Universitat de València
Doctoral Dissertation / *Tesis Doctoral*

Computational Measures of Information Gain and Reinforcement in Inference Processes

Medidas Computacionales de Ganancia de Información y Refuerzo en Procesos de Inferencia

José Hernández Orallo

Supervisor/*Director*: **Prof. Dr. Rafael Beneyto Torres**
Catedrático de Lógica y Filosofía de la Ciencia
Universitat de València

A thesis submitted to the *Universitat de València* in accordance with the requirements of the degree of Doctor of Philosophy in the Department of Logic and Philosophy of Science.

September 1999

Abstract and Keywords

This work is devoted to the formal study of inductive and deductive concept synthesis usefulness and aftermath in terms of information gain and reinforcement inside inference systems. The set of measures which are introduced allow a detailed and unified analysis of the value of the output of any inference process with respect to the input and the context (background knowledge or axiomatic system).

Although the main measures, computational information gain, reinforcement and intensionality, are defined independently, they (alone or combined) make it possible to formalise or better comprehend several notions which have been traditionally treated in a rather ambiguous way: novelty, explicitness/implicitness, informativeness, surprise, interestingness, plausibility, confirmation, comprehensibility, 'consilience', utility and unquestionability.

Most of the measures are applied to different kinds of theories and systems, from the appraisal of predictiveness, the representational optimality and the axiomatic power of logical theories, software systems and databases, to the justified evaluation of the intellectual abilities of cognitive agents and human beings.

Keywords: *Inference Processes, Evaluation Measures, Induction, Deduction, Information, Kolmogorov Complexity, Reasoning, Inference Paradox, Information Gain, Inference Confirmation, Reinforcement, Intensionality, Measurement of Cognitive Abilities, Evaluation of Logical Theories, Knowledge-Based Systems, Machine Learning, Inductive Logic Programming, Intensionality.*

Resumen y Palabras Clave

Esta tesis se centra en el estudio formal de la utilidad y resultados de la síntesis de conceptos inductivos y deductivos en términos de ganancia de información y refuerzo en sistemas de inferencia. El conjunto de medidas que se introducen permiten un análisis detallado y unificado del valor del resultado de cualquier proceso de inferencia con respecto a la entrada y el contexto (conocimiento previo o sistema axiomático).

Aunque las medidas más importantes, ganancia computacional de información, refuerzo e intensionalidad, se definen de manera independiente, permiten (solas o combinadas) formalizar o comprender mejor varias nociones que han sido tratadas tradicionalmente de una manera bastante ambigua: novedad, la diferencia entre explícito e implícito, informatividad, sorpresa, interés, plausibilidad, confirmación, comprensibilidad, 'consiliencia', utilidad e incuestionabilidad.

La mayoría de las medidas se aplican a diferentes tipos de teorías y sistemas, desde la estimación de la capacidad de predicción, la optimalidad de representación, o el poder axiomático de teorías lógicas, sistemas software y bases de datos, hasta la evaluación justificada de las habilidades intelectuales de agentes cognitivos y seres humanos.

Palabras Clave: *Procesos de Inferencia, Medidas de Evaluación, Inducción, Deducción, Información, Complejidad Kolmogorov, Razonamiento, Paradoja de la Inferencia, Ganancia de Información, Confirmación de la Inferencia, Refuerzo, Medición de Capacidades Cognitivas, Sistemas Basados en el Conocimiento, Aprendizaje Computacional, Programación Lógica Inductiva, Intensionalidad.*

Contents

Extended Abstract

This work introduces several evaluation measures which are applicable, in a consistent and effective way, to different inference processes. In particular, these measures are established through two main tools:

- The theory of Kolmogorov Complexity, and especially Levin's space-time variant, allow the definition of a measure of information gain which depends on the effort that has been invested in any given inference process.
- The theory of reinforcement, understood as the propagation of truth or certainty degrees from some statements to others, makes it possible to define a theory of confirmation which includes in a quantitative way both deductive and inductive confirmation.

Both tools are not (strictly) semantical, and it is precisely this fact which allows the measurement of different dimensions which have not been tackled successfully to date from purely semantical approaches: informativeness, plausibility, 'consilience', intensionality, intelligibility and utility.

The first part of this thesis is based on the fact that processes that are apparently so unlike as induction and deduction can be explained in a computational framework as inference processes that both generate an output from an input. Obviously, they must observe different criteria or restrictions, which have been widely studied in philosophy of science and mathematical logic, respectively. In this computational framework, both processes are regarded as non-omniscient processes, i.e., resource-demanding processes. Levin's variant, which weighs the additional amount of information and computational time which is required to perform an inference, is used to define a *single* gain measure of that inference.

The leading results are obtained by applying the gain measure in an equally clarifying and unifying way to both inductive and deductive processes. In the case of induction, the information gain represents how informative the hypothesis is with respect to the observations, in Popper's sense, and it is compared with other evaluation criteria for induction, mainly simplicity. In the case of deduction, information gain also represents how much informative the conclusion is from the premises, which establishes a generic measure of the gain obtained whenever an explicit knowledge is extracted from an implicit knowledge. In fact, this represents a generalisation of Hintikka's notions of surface and depth information for first-order logic.

Apart from its unifying and explanatory power, the measure of information gain which is presented, although computable, is, as expected, computationally intractable, and it is not directly applicable to concrete systems. Accordingly, a more efficient and *detailed* measure is introduced, based on the reinforcement or use of the components

of an inductive theory or axiomatic system. Reinforcement represents a measure of the confirmation of a theory, which includes the propagation of confirmation by deductive and inductive inference (thus giving a measure of plausibility or utility, too). Moreover, reinforcement is easy to compute and it is positively related to information gain.

Another connection is established between the idea of implicitness and the notion of intensionality of a description. It is shown not only that extensional definitions have no gain at all but also that intensional definitions, the latter understood as definitions without exceptions, have a great probability of showing a high information gain. Moreover, the theory of intensionality allows the formalisation of the idea of comprehension, and helps to make the difference between descriptive induction and explanatory induction, the latter requiring that all the evidence should be 'consiliated' by the theory, by avoiding exceptions or extensional cases.

The previous measures are particularised for logical theories and are compared with other measures in the literature, especially the Minimum Description Length (MDL) principle. It is shown that the measure of reinforcement is more detailed and comprehensive. Furthermore, it is more robust, because it avoids the problem of induction for finite and random evidences, where the MDL principle suggests the evidence itself and, consequently, nothing is learnt.

The non-omniscient view of inference processes makes it possible to relate the computational capability of a rational agent with several inference problems. More precisely, the difficulty of an instance (or problem) can be defined in terms of the information gain from the problem to the solution and the intrinsic complexity of the solution. A comprehension test is then devised, and correlates, at the sight of results, with classical psychometric tests, representing a formal and non-anthropomorphic alternative to the Turing test.

Finally, several applications of information gain and reinforcement are shown for other inference processes such as abduction or analogy, and many others are sketched for artificial intelligence and computer science: rational agents with limited resources, knowledge-based systems, and knowledge discovery in databases.

All in all, the most important result of this work is an operative clarification of the relationship between the notions of inference, information and confirmation. As a conclusion, the view of induction and deduction as inverse processes in terms of information gain is definitively dismissed for non-omniscient systems and for agents with limited-resources, human beings and computers included among them.

Resumen Extendido

Esta tesis introduce varias medidas de evaluación que son aplicables, de una manera consistente y efectiva, a diferentes procesos de inferencia. En particular, dichas medidas se establecen a partir de dos herramientas fundamentales:

- La teoría de la complejidad de Kolmogorov, en especial, la variante de Levin espacio-temporal, permite definir una medida de ganancia de información que depende del esfuerzo que se haya invertido en cualquier proceso de inferencia.
- La teoría del refuerzo, entendida como propagación del grado de verdad o certeza de unos enunciados a otros, permite definir una teoría de la confirmación que incluye de una manera cuantitativa tanto la confirmación deductiva como la confirmación inductiva.

Ambas herramientas no son semánticas, y es precisamente este hecho lo que permite medir con éxito diferentes dimensiones que no han sido bien abordadas hasta ahora desde aproximaciones puramente semánticas: informatividad, plausibilidad, 'consiliencia', intensionalidad, inteligibilidad y utilidad.

La primera parte de esta tesis se basa en el hecho de que procesos aparentemente tan diferentes como son inducción y deducción pueden explicarse en un marco computacional como procesos de inferencia que generan una salida a partir de una entrada, y que deben cumplir ciertos criterios o restricciones, ampliamente estudiados en filosofía de la ciencia y en lógica matemática, respectivamente. En este marco computacional, ambos se pueden ver como mecanismos no omniscientes, es decir, procesos que consumen recursos. La variante de Levin que pondera la cantidad de información adicional para llevar a cabo la inferencia y el tiempo computacional empleado en ella, es utilizada para definir una medida *única* de la ganancia de dicha inferencia.

Los primeros resultados se obtienen al aplicar dicha medida de manera igualmente clarificadora y unificadora tanto en procesos inductivos como deductivos. En el caso de la inducción, la ganancia de información representa cuán informativa es la hipótesis respecto a las observaciones, en el sentido de Popper y se compara con otros criterios de evaluación de teorías inductivas, principalmente el de simplicidad. En el caso de la deducción, la ganancia de información también representa cuán informativa es la conclusión a partir de las premisas, estableciendo una medida genérica de la ganancia obtenida al extraer un conocimiento explícito a partir de un conocimiento implícito, tal como fuera apuntado por Hintikka con las nociones de información superficial y profunda para la lógica de primer orden.

Aparte de su poder unificador y explicativo, la medida de ganancia de información que se presenta, aunque computable, es, como era de esperar, intratable computacionalmente, y no es aplicable directamente a sistemas concretos. Por esta

razón se introduce una medida más eficiente y *detallada*, basada en el refuerzo o uso de los componentes de una teoría inductiva o de un sistema axiomático. Aunque el objetivo inicial, como se ha apuntado antes, es proporcionar una medida de confirmación de una teoría que incluya la propagación de la confirmación por inferencia deductiva o inductiva, dicha medida se relaciona positivamente con la ganancia de información.

Por último, se establece la conexión entre la idea de conocimiento implícito y la noción de intensionalidad de una descripción, mostrando no sólo que las definiciones extensionales tienen ganancia cero sino que las definiciones intensionales, entendidas éstas como definiciones sin excepciones, tienen gran probabilidad de tener ganancia de información alta. Además, la teoría de la intensionalidad permite formalizar la idea de comprensión, y ayuda a diferenciar entre inducción descriptiva e inducción explicativa, requiriendo ésta última que todas las observaciones sea 'consiliada' por la teoría, evitando excepciones o casos extensionales.

Las medidas anteriores se particularizan para teorías lógicas y se comparan con otras medidas de la literatura, especialmente el principio de la descripción de longitud mínima (MDL), y se muestra que la medida de refuerzo es más detallada y comprensiva. Asimismo, es más robusta, ya que evita los problemas de inducción para evidencias finitas y aleatorias, donde el principio MDL sugiere la evidencia misma, no dando así ninguna explicación para la evidencia y, más aún, no aprendiendo nada.

La visión de los procesos de inferencia como no omniscientes permite relacionar la capacidad computacional de un agente racional con diversos problemas de inferencia. En particular, se define de una manera formal la dificultad de una instancia (o problema) a partir de la ganancia de información desde el problema a la solución y la complejidad intrínseca de la solución. Esto permite realizar tests de comprensibilidad, que correlan, a la vista de los resultados, con los clásicos tests psicométricos y suponen una alternativa formal no antropomórfica al test de Turing.

Finalmente, se muestran diversas aplicaciones de la ganancia de información y el refuerzo a otros procesos de inferencia como la abducción o la analogía, y se esbozan otras muchas en el campo de la inteligencia artificial y la computación, desde los agentes racionales con recursos limitados, los sistemas software basados en el conocimiento, hasta el campo de "descubrimiento automático de conocimiento" en bases de datos.

En definitiva, el resultado más importante de esta tesis es el esclarecimiento de la relación entre las nociones de inferencia, información y confirmación. Como conclusión, la visión de inducción y deducción como procesos inversos en términos de ganancia de información se descarta definitivamente en sistemas no omniscientes y de recursos limitados, entre ellos los seres humanos y las computadoras.

Authorship

The work in this thesis dissertation is the independent and original work of the author, except where explicit reference to the contrary has been made. No portion of this work has previously been submitted in support of an application for a degree of this or any other university.

Autoría

El trabajo que se muestra en la presente tesis no ha sido previamente presentado para la obtención de alguna titulación o diploma en ésta o cualquier otra institución educativa superior. Por lo que conozco, ninguno de los materiales que aquí se presentan han sido previamente publicados o escritos por otra persona excepto en aquellas partes donde se hacen referencias de manera explícita.

José Hernández Orallo, martes, 17 / mayo / 2011

Acknowledgements

My first notion of resource-bounded complexity was experienced from my first computer, a slow but powerful Z80 with only 48K. As a teenager, on some occasions I even dreamt about how to make it reason in some way. A few years later, after a computer science degree, I already knew that the challenge would surely take a much more important part (or the whole) of my life.

In 1995, after six months and a day of reflections in France, I felt that I still needed to learn a great deal if I intended to do something positive about the matter. The attractive contents of the doctorate program of the Department of Logic and Philosophy of Science of the University of Valencia, and the first interview with my supervisor Rafael Beneyto, arose inside me the firm conviction that I would never give up that challenge, at least the aim to be always aware of its state of the art.

During the first year and a half of my doctorate courses, the major problem was to combine jobs in different companies with a thirteen-month-long substitutive social service, and still make the lectures something profitable, with a sort of juggling and sleeplessness. The major gratitude is given to those who *suffered* me in those difficult moments, especially Neus and my mother.

In the summer of 1996 I was granted with a six-week visit to the Dept. de la Ciencia de la Computación de la Pont. Universidad Cat. de Chile, under the Intercampus Programme, with subject "logical knowledge representation". Javier Pinto, who tutored me there, was a great stimulus and a source of knowledge about many different areas, from situation calculus to modal logics.

My position as a full-time teaching and research assistant in the Dept. de Sistemes Informàtics i Computació (DSIC) de la Universitat Politècnica de València (UPV) since late 1996 has provided an exceptional platform from which to go on with more dedication (and motivation) with this work. I must acknowledge the resources the department has afforded me, in particular, the DSIC research commission, which defrayed part of the participation costs in some conferences during 1998. The Extensions of Logic Programming (ELP) group, led by María Alpuente, has shown its availability and support since the first moment, and the joint work with María José Ramírez has been so productive partially due to this atmosphere.

Nonetheless, it is in this Dept. de Lògica i Filosofia de la Ciència de la Universitat de València where I have found the main source of wisdom and kindness (and I hope it will still be so for many years). I am particularly grateful to Jesús Alcolea (I pleasingly remember his under-graduate lectures on philosophy of mathematics), José Pedro Úbeda, Juan Manuel Lorente and Enric Casaban. They made a computer scientist feel comfortable in a department of logic and, essentially, I learnt a lot from them.

During the last two years of development of this dissertation, parts of it have been submitted to several journals and presented in diverse conferences. Many comments, suggestions and, of course, critiques have been extremely helpful for directing and growing it. Concretely, the following people have improved this dissertation with fruitful discussions, corrections, material or technical support: Kenneth Wexler, Peter Flach, Boris Siepert, Atocha Aliseda, Mark Derthick, John Lloyd, Stephen Muggleton, Paul Thagard, the anonymous referees of CCIA'98, Nigel Crook, MªCarmen Juan, Carlos Monserrat, Kike Araque, Rose Barreiro, Enrique Fueyo, Peter Wegner, Lorenzo Magnani, Vincent F. Hendricks, Enrique Hernández, Alen Varsek, Joan Carles Micó, Neus Minaya, and Ismael García (some publications derived from this work have been co-written with the latter two). Finally, I am much obliged to the main inspirers of chapter 8, Greg Chaitin and Douglas Hofstadter, for their encouraging comments, especially during 1997, when the main ideas were taking shape.

Finally, I would like to thank my supervisor, Rafael Beneyto, for letting me take some risks, but not too many, in this free quest in quicksand. I also appreciate the effort to realise the venture of this thesis to him and the other members of the reading committee.

En fi, tots els adés citats saben que la Ciència és una companya exigent. Ens furta temps, som i esforços, dels quals només a voltes ens en torna fruits. Més bé encara ho sap la meua dona, i a ella promet tornar-li part del temps que, fins ara, no li n'he pogut dedicar.

To my mother,
From Pradilla to London, that's certainly an odyssey life.

1. Introduction

Certa amittimus, dum incerta petimus[1]
Plauto, II century BC, Pseudolus, 685

Abstract: *This chapter introduces the motivations for this work, by realising some problems that pervade the conception of inference processes, mostly their joint interpretation in terms of information and plausibility, or its traditional view as inverse processes derived by Carnap's probabilistic calculus. Some precedents that recognise these problems, mainly the deductive inference paradox and Popper-Miller's argument, and some partial solutions, such as Hintikka's distinction between depth and surface information, are discussed. New tools, such as Kolmogorov Complexity, especially Levin's space-time variant, are required to account for non-omniscient deduction, where the effort of any inference, either deductive or inductive, would be recognised. Moreover, a constructive extension of a theory of reinforcement could also address the confirmation problem of both inductive and deductive inference. These non-strictly semantic tools also centre the scope of this work. Hence, the concrete aims are given by the measurement of different dimensions under this machinery: informativeness, plausibility, consilience, intensionality, intelligibility and utility. The end of the chapter includes an overview of each of the chapters, and some necessary notation.*

Keywords: Evaluation Measures, Inference Processes, Induction, Deduction, Semantic Information, Kolmogorov Complexity, Reasoning, Inference Paradox.

[1] *We abandon the certain for seeking the uncertain.*

1.1 Introduction

Reasoning involves different inference processes. Logic has traditionally studied deduction from a semantic point of view, according to completeness and correctness, devoted to ascertain which inference rules and conclusions are sound. Induction, however, is a quite different matter. For any sufficient expressive language, there are infinite valid hypotheses for any given evidence. The main and pristine question of induction has always been the establishment of hypotheses selection criteria, either epistemological or methodological. Plausibility (or likeliness), utility, comprehensibility and informativeness have been the most vindicated criteria, although they can all be understood in many different (even contradictory) ways. These measures have been adapted and applied to other hypothetical inference processes, such as abduction and analogy, but rarely addressed as fundamental issues for deduction. The reason is quite simple: a theorem prover is not intended to rate its theorems, it must only state which formulae are theorems and which are not.

However, reasoning is much more than theorem proving, much more than inductive generalisation and much more than abduction, analogy and other partial inference processes. And it is much more than the sum of all of them. If we can combine consistently and profitably different inference processes, we will enlarge the power and applications of the separate advances in different fields of logic, philosophy of science, artificial intelligence, automated reasoning and machine learning that have taken place in the last half of the XXth century. This would allow that the progress in these different areas would be applied to make intelligent systems, capable of acquiring and deriving new knowledge.

The *subject* of this work is the unified evaluation of inference processes, under different dimensions, mainly plausibility, utility, comprehensibility and informativeness. The interest and extension of this work will be devoted to those pairs (dimension, process) which lack appropriate measures. On the other hand, in the case an existing theory accounts for a given pair of dimension and inference process, it will be referred, as the case of plausibility for monotonic and non-monotonic deduction.

For such a study it is necessary to evaluate the contribution of each inference, in order to invest resources towards useful and valuable results. The result of an inference process, in many cases, is a (new) *concept, assertion* or *fact* (as a simple hypothesis, a theorem or property, a proof, a whole (or part of a) theory, a change of confirmation degree...). A concept may appear in all its varieties, including necessary, auxiliary, inductive and synthetic concepts. Apart from a proper concretisation of the term ‘concept’ (determining different approaches depending on the representation), it is first required to settle on such fundamental issues as deduction, induction, information and complexity, as they are known to date.

Different tools will be used in this work, which will also determine its *scope*. The first necessary tool is a universal and independent measure of information, which could be used for both induction and deduction. This measure of information is represented by Kolmogorov Complexity. Since the appearance of the idea of the shortest algorithmic description for a given object in the sixties, the theory has been successfully applied to fundamental areas of mathematics, computer science, artificial intelligence, physics and philosophy of science. To our concern, Kolmogorov Complexity has addressed fundamental problems of statistics, induction, information and complexity. However, little work has been done to address two important topics: representation and deduction. Representation issues have been neglected due to the invariance theorem (i.e. any universal machine can emulate any other universal machine with a constant additional space and time cost) and the process of deduction has been usually forgotten because the absolute version of Kolmogorov Complexity, denoted by $K(x)$ and defined in section 1.5, does not consider the time of computation.

As we will see, many insights can be drawn about information transformation, be it inductive or deductive, by using as theoretical tools the absolute Kolmogorov Complexity $K(x)$ and, especially, Levin's version, denoted by $Kt(x)$ and also defined in section 1.5, which weighs space and time. Derived concepts from K and Kt will allow to account for some dimensionalities: informativeness, comprehensibility, and, partially, plausibility.

The other important tool is the idea of reinforcement, well-known in psychology and machine learning but under-exploited in theoretical and philosophical considerations. The idea can be found in a philosophical context by many empiricists, and concretely Quine's "empiricism without dogmas" [Quine 1953], where any conflict with experience on the periphery of knowledge entails readjustments in the inner parts of the field: truth values must be re-distributed between some of the previous statements. The idea of propagation, later on exploited by artificial neural networks, is clearly stated by him: "*Once the values are re-distributed among some statements, it is also necessary to re-distribute the values of other statements which can be logically connected with the first (...)*" [Quine 1953].

Although the idea of a propagation of confirmation from the outer part of perception to the inner part of cognition has been successful for artificial neural networks, it has not been extended to more constructive and expressive frameworks. The reason may possibly be found in that a direct use of reinforcement for constructive representational languages leads to paradoxes since it is possible to add *fantastic* concepts which are used for the rest of a theory, and reinforcement of the whole theory is increased in a tricky way. This problem will be solved in chapter 5, as a point in between Hempel's quantitative and Carnap's qualitative solutions to the problem of inference confirmation, so triggering many different applications[2] that were handicapped by this paradox.

1.2 Motivation and Precedents

Traditionally, induction and deduction have been seen as complementary inference processes. However, during the last half of the XXth century, induction has frequently been seen as the *inverse* of deduction, in terms of information gain, i.e., induction increases information and deduction decreases it. This view was axiomatised by Carnap, who formalised in the fifties [Bar-Hillel and Carnap 1953] Popper's notion of semantic information, informally introduced in the thirties, as a counterpart to statistical information, represented by Shannon's mathematical theory of communication.

Semantic Information was defined as the negative logarithm of the probability as given by Carnap's Probabilistic Interpretation of First-Order Predicate Calculus. It has the following well-known properties:

$$p(\top) = 0$$
$$p(\bot) = 1$$
$$p(P \wedge Q) = p(P \cap Q)$$
$$p(P \vee Q) = p(P \cup Q)$$
$$p(\neg P) = 1 - p(P)$$
$$p(P) \leq p(Q) \text{ if } P \models Q$$

From the very beginning the difficulties to harmonise semantic information and statistical information theory were shown clearly. As we will comment later, Hintikka (see for instance [Hintikka 1970a]) introduced the difference between surface and depth information in order to establish the intuitive fact that both deduction and other truth-preserving processes (such as the introduction of auxiliary concepts) could as well generate information, improve the utility and manageability of the deductive system by changing syntax while preserving its semantics.

Without any note related to Hintikka's work, the justification of induction which is used in the field of Inductive Logic Programming (ILP) (see e.g. [Muggleton 1996], resorts to "Shannon's information theory" for introducing, without more discussion, the usual assignment $I(P) = -\log p(P)$. From here, if $P \models Q$, then we have, from the

[2] Utility is not directly addressed by reinforcement but what I will show in chapter 5 is that reinforcement can be used for utility if actions are also considered to be reinforced. Even more, reasoning actions can also be reinforced, and this makes it possible for improving the reasoning abilities of a system.

last property of Carnap's Probabilistic Calculus, that $p(P) \leq p(Q)$. By using this assignment we have that $I(P) \geq I(Q)$, i.e., the premise must have more information than the consequence.

After this rationale, the popular assertion "*deduction loses information and induction increases it*" seems compelling. However, this restricted view of information precludes any possibility of deduction as a useful, valuable process, known as the *inference paradox*, which I will discuss below. But even more, Shannon's information theory has been substituted by the more general and accurate view of information, Descriptional or Kolmogorov Complexity, making popular the Minimum Description Length (MDL) principle for induction. The MDL principle, a formalisation under information theory of Occam's razor, chooses the theory which minimises $I(T \mid E)$ with T being the hypothesis and E the data. Since the evidence is correct, $I(E) = 0$, and taking logarithms to Bayes' rule we have:

$$I(T \mid E) = I(T) + I(E \mid T)$$

In this moment, $I(x \lfloor y)$ is sometimes [Muggleton et al. 1992] [Muggleton and Page 1994] computed by its Kolmogorov Complexity $K(x \lfloor y)$, i.e., the minimum encoding of T. The result is that the shorter the hypothesis with respect to the data, the more likely. Although this usually works for logic programs, we have a very counterintuitive result of using $K(x)$ instead of $I(x)$: deduction increases length and induction decreases length. Thus the length (or $K(x)$) cannot be a good approximation to the intuitive idea of information according to Carnap. This seems to comply with Popper in his denial of an objective measure for information.

The roots of these problems, however, can be found in the view of deduction as an omniscient, complete and perfect process, which leads to the *inference paradox* and, as a consequence, the *inductive paradox* (or impossibility of inductive probability), also known as the scandal of deduction [Hintikka 1973] and the scandal of induction (advocated by Hume), respectively.

Let us see first the inductive paradox. Popper and Miller started a vigorous debate (see [Mura 1990] for an extensive account) on the relationship between deductive relations and probabilistic support with their paper *A Proof of the Impossibility of Inductive Probability* [Popper and Miller 1983]. Popper and Miller made the claim that any positive probabilistic support of evidence e for a hypothesis h, as measured by $s(h,e) = p(h,e) - p(h)$ is due solely to deductive relations (properly understood) between e and h. An immediate corollary is that inductive (i.e. non-deductive) probabilistic support does not exist. In other words, Popper and Miller claim that all probabilistic support is deductive.

Their argument is based on an omniscient view of logic, i.e., complete: "we find that what is left of h once we discard from it everything that is logically implied by e, is a proposition that in general is counter-dependent on e" [Popper and Miller 1983].

Let us see this result by using elementary logic, directly from [Cussens 1998]:

Definition 1.1 For any proposition b, $Cn(b)$ is the class of all consequences of b which are not logical truths. $Cn(b) = \{x: b \models x \text{ and } \not\models x\}$.

Definition 1.2 Two propositions, a and b, are deductively independent if and only if $Cn(a) \cap Cn(b) = \varnothing$. Otherwise they are deductively dependent.

Lemma 1.1 For any two propositions a and b, $Cn(a) \cap Cn(b) = Cn(a \vee b)$.

PROOF.

$$\begin{aligned} x \in Cn(a) \cap Cn(b) &\Leftrightarrow a \models x, b \models x, \not\models x \\ &\Leftrightarrow \models \neg a \vee x, \models \neg b \vee x, \not\models x \\ &\Leftrightarrow \models (\neg a \vee x) \wedge (\neg b \vee x), \not\models x \\ &\Leftrightarrow |= (\neg a \wedge \neg b) \vee x, \not\models x \\ &\Leftrightarrow |= \neg(a \vee b) \vee x, \not\models x \\ &\Leftrightarrow a \vee b \models x, \not\models x \\ &\Leftrightarrow x \in Cn(a \vee b) \end{aligned}$$

□

Corollary 1.2 For any two propositions a and b, a and b are deductively independent if and only if $\models a \vee b$, i.e., $(a \vee b)$ is a logical truth.

PROOF. Using Lemma 1.1, $Cn(a) \cap Cn(b) = \varnothing \Leftrightarrow Cn(a \vee b) = \varnothing$ □

Corollary 1.3 b is deductively independent of a if and only if $\neg a \models b$.

PROOF. $\models a \vee b \Leftrightarrow \neg a \models b$. The result follows immediately from Corollary 1.2 and Definition 1.2. □

There are two problems with Popper-Miller's focus on pure inductive support. Firstly, for the dependence between a and b to be purely inductive, it apparently has to be the case that a and b are deductively independent, but this is equivalent to having $\neg a \models b$. So, if the dependence between a and b is purely inductive, the deductive dependence between $\neg a$ and b has to be maximal. This connection between pure inductive dependence and deductive consequence shows that it is not possible to define a notion of purely inductive dependence, which is free from deductive contamination[3].

[3] It is remarkable to see that, from the descriptional point of view, i.e. $K(x)$, deductive dependence and independence are extremely close, namely $a \models b$ and $\neg a \models b$.

As [Cussens 1998] points out "*However, there seems no reason to suppose that inductive (i.e. ampliative) inference should not be deductively contaminated. There can be a* <u>*relation*</u> *between deduction and induction, without the two types of inference being equivalent, or one reducible to the other. In fact, I take the investigation of this relation by Popper and Miller to be the most useful contribution made by the Popper-Miller argument*".

This clearly neglects the modern view of induction and deduction as inverse inference processes. Even more, "*any notion of 'induction' as a sort of complement to deduction seems untenable*" [Cussens 1998].

A more proper conception is based on the idea that deduction preserves semantics whereas induction amplifies it, so in this way they can be very different processes but not exactly inverse. The paradox also arises when these two main inference processes are evaluated in terms of knowledge. Simplistically, the role of acquiring new knowledge is left to induction whereas deduction is just used to retrieve this knowledge when necessary (the deductive database viewpoint).

But whoever has some original view of deduction, namely some non-omniscient view of deduction (such as any who has worked on automated deduction or has practised mathematics), knows well that many deductive inferences give us much information inside the same axiomatic system, without changing the model (or set of inferable facts). Even more, this information is worth enough being 'remembered' or maintained explicitly.

As it has been said, the problem of deduction is derived from the classical "inference paradox", mainly indicated by Mill (he is not the first to insinuate the problem but the classical source of the discussion), also recognised as the main unsolved problem for the justification of deduction by [Dummett 1973] and expressed in these terms by [Cohen and Nagen 1935]: "*if the conclusion of an inference is not contained in the premises, it cannot be valid; and if it is not different from them, it is useless; however, the conclusion cannot be contained in the premises and be at the same time novel; consequently, inferences cannot be both valid and useful.*"

Both the inductive paradox and the inference paradox are motivated by the thought that deduction is omniscient while induction is not. Two escapes are possible. The first one is to consider both deduction and induction omniscient, which would mean that reasoning is useless and the only increase of information can be given by perception. The second possibility is much more conspicuous: to consider both deduction and induction as imperfect or incomplete inference processes which require some effort, and, consequently, they are valuable because they help to make explicit what was implicit.

The main idea of this thesis is to consider *information* and *use* dependence instead of deductive dependence. Nonetheless, information dependence cannot be used in an absolute (time independent) way as it is given by absolute Kolmogorov Complexity. Namely, if we have $a \models b$ then $K(b \mid a) = 0$ (unless b is a subset of all the consequences of *a*), however $K(a \mid b)$ is usually greater then 0. In other words, if time is not considered (omniscient deduction) the classical result is obtained once again: deduction does not give information and induction can provide it. The things change radically if we consider the same relation using time-space variants of *K*. That is to say, if we consider Levin's *Kt*, we have that both $Kt(b \mid a)$ and $Kt(a \mid b)$ can be greater than 0, because if the deductive system has limited resources, it has not omniscience or it is incomplete, the deductive inference of *b* from *a* can provide information as well, depending on how explicit $a \models b$ is in the system.

For the case of use dependence, valid and interesting inferences help to shorten proofs and are useful to compact a theory or make it more comprehensible. For instance, consider $a \models b$ and $c \models d$. The discovery of the following *deductive* connections $a,c \models e$, $e \models b$ and $e \models d$ is valuable because both *b* and *d* can be derived only from *e* and this derivation could be much easier. Consider now $a \models c$ and $b \models c$. The discovery of the *inductive* concept *d* such that $d \models a$ and $d \models b$ is valuable because both *a* and *b* are a consequence of *d*, and this *d* may help to explain both facts or to unify the evidence. Although induction has recognised this gain, it is deduction which still lacks a general and widely accepted account of deductive informativeness and novelty, despite the fact there are some fields of artificial intelligence which urgently demand such an assessment.

The first field which requires measurements of information gain is automated reasoning, or more properly, automated theorem proving (ATP), which is highly interested in avoiding useless inferences and maintaining explicit those properties that can be useful to find a proof of a theorem.

The second field is more related with artificial intelligence, and is known as bounded-rationality, in the search of deduction methods which are efficient and where the cost of a deductive inference is recognised as an important factor which should be minimised, by many different means, such as avoiding the re-derivation of useful and common facts which are costly to derive from the axioms. A rational system is seen in a dynamical way [Girard et al. 1989], distinguishing what is potentially derivable (i.e. provable), what is feasibly derivable, and what is explicitly known in a given situation.

Finally, in my opinion, there is a heritage in Philosophy and a practice in Artificial Intelligence of studying deduction and induction in a separate way. As we have seen, many works are untenable when they are contrasted with the deductive/inductive counterpart in terms of information. So it seems attractive to study a possible 'conciliation' among induction, deduction, use and information. In particular, a clear account of knowledge and representation shifts via conceptualisation is not possible if the previous question is not clarified first.

Although the view of omniscient deduction still pervades many philosophical investigations about the relation between induction and deduction, there are, of course, some precedents. Different non-semantic evaluation measures for deduction and induction have been introduced in the literature. However, they have almost always been studied *separately*.

Surely, it is induction which has introduced more evaluation criteria, motivated by the problem of the justification of induction. From the point of view of *plausibility*, many *selection* criteria have been advocated. Simplicity, exemplified by Occam's razor and its modern formalisation as the Minimum Description Length (MDL) principle is the most vindicated and successful one, because it is both a plausibility and methodological criterion. Other related criteria are cross-validation, maximum likelihood estimators (MLE) according to a selected prior, generality, specificity, explanatory power, etc. Some of them will be reviewed in the next chapter.

In general, if we consider a concept as a new creation (either as a result of and inductive, abductive, analogical or deductive process) one must distinguish some main characteristics of that concept[4].

According to *hardness*, many subjective and informal distinctions have been presented between hard and easy concepts. Basically, a concept is hard if the disjuncts of the concept to be obtained are "spread out" in the instance space, but it can yet be formulated with the given features [Kramer 1995]. On the contrary, easy concepts are those that can be secured even by simple reasoners [Holte 1993].

According to *necessity*, [Ling 1991] distinguishes informally between accessory or useful concepts, which are not crucial but help to compress or better express a theory, and necessary, i.e., concepts that make it possible to learn or define a given concept. The first ones can only improve the form of the solutions but the latter can be necessary to find the solution.

According to *nature* [Kramer 1995] we can distinguish among extensional and intensional. Extensional concepts, often known as features, are defined as an extensional definition of a set. On the contrary, intensional concepts are defined as a comprehensive property or function for them. A finer division may be established between intensional concepts with and without recursion. Concepts without recursion can be intensional, but just if they are derived or defined from other concepts. [Thagard and Nowak 1990] call them *concept combinations*. On the other hand, recursive concepts are usually intensional: they are derived concepts such that in the definition the concept itself appears directly or indirectly. The connection between recursive concepts and necessary concepts has been studied in logic[5] and also in machine learning [Stahl 1995], and some authors [Ling 1991] state that they are the only ones that are *necessary*.

[4] It is relevant to highlight that if one allows definition rules in a deductive system, one may observe the similarities between learning a concept (which can be identified as an induction) and defining a new concept from other (which can be identified as a usual deductive task).

[5] Frege was the first to clarify the notion of concept invention in formal theories, with its informal explanation of the difference trivial and non-trivial definitions in logic and mathematics.

A last additional aspect is the *comprehensibility* of induced concepts (generally, referred to humans) and this is not the same as compressibility or syntactical simplicity (although sometimes *elegance* has been seen as a unified term for both comprehensibility and simplicity [Quine 1953] [Chaitin 1998]), as [Kramer 1995] points out "[...] *comprehensibility and syntactical complexity might be correlated, but surely are not the same.*"

Despite all this variety, only plausibility criteria have been formalised in a convenient way. Hardness, necessity, nature and comprehensibility have generally been studied in an informally and usually unrelated way.

For the case of deduction, the scene is still worse. Deduction has traditionally lacked from this interest, and logic has mainly been devoted to ascertain the correction of a deductive inference, its semantic validity, its completeness, and not its practical or informational value. Even epistemic and modal logics have almost always (see e.g. [Duc 1997]) taken logical omniscience for granted. Apart from Hintikka's approach, only recently has there been a renewed interest about the evaluation of deductive inference, precisely in the fields of automated theorem proving and bounded rationality seen before. It is then compelling to recognise the importance of auxiliary concepts in a formal framework and state in a clear way that deduction can also increase information. An expected question to this urgency is whether we are in a better situation now to address this problem than some decades ago.

Since the time Hintikka tried to clarify the paradoxes of induction and deduction in terms of information gain and utility, the accepted view of information has changed in a very important way, towards the more general and universal view based on Kolmogorov Complexity. Moreover, some fields of AI, such as machine learning and knowledge-based systems have helped to clarify the problem and recognise the cost and non-omniscient character of both induction and deduction. All this allows much more challenging goals with a reasonable chance of success.

1.3 Aims

The main aim of this work is the formal study of concept synthesis usefulness and aftermath in terms of information gain and reinforcement inside inference systems. The measures to be developed should be consistently and equally applicable for both deductive and inductive inference.

The central concern will be the *evaluation* (and not the generation) of concepts, although some issue in this regard will be punctually addressed. For this evaluation, several coherent measures should be devised, valid both for deduction and induction. Concretely, the specific objectives of this work are given by the *measurement* of the following dimensions:

- **Informativeness**: a new measure will assign the *information gain* of a given inference from a concept x to a concept y. This will allow to clarify the notions of explicitness and implicitness, and to give general and alternative notions to Hintikka's surface and depth information of deductive systems and Popper's informativeness for induction.

- **Plausibility**: this dimension is not applicable for classical deduction. We will measure this dimension for non-truth-preserving processes (induction, approximate or non-monotonic deduction, abduction, ...) by means of a theory of *reinforcement*, given by the necessary use of each part of a theory or system in the rest and the evidence.
- **Consilience**: this dimension, informally introduced by Whewell in 1847, is related to the degree of uniformity to which a theory covers its consequences, and it is also usually referred (with slightly different nuances) as coherence or unification. It has usually vindicated in explanatory induction, where the theory must be comprehensive with the evidence, in the way that all the examples must be covered or unified by the same general rule or *cause*.
- **Intensionality**: a pristine question associated with any definition is whether it is extensional (by extension) or intensional (by comprehension). A first analysis of this question will show that it is not appropriate to assign a Boolean answer to it. Consequently, a degree of intensionality will be introduced, closely related with the idea of exception.
- **Comprehensibility / Intelligibility**: A measure will be introduced to sacle the difficulty of comprehend, namely the degree of comprehensibility or intelligibility of a given concept. It will also be particularised to estimate the difficulty of different problems of inductive and deductive character. This will allow the measurement of intellectual abilities, without anthropomorphic contamination.
- **Utility**: in deductive systems, the utility of the introduction of new concepts for different purposes is informally clear: a better understanding of the whole theory, a more concise expression of the *same* idea, a reduction of the computational time and size of future deductions (i.e. proofs), etc. This necessity of intermediate information will be formally shown. In the case of induction the notion of utility is closely connected to plausibility, as it will be represented by the use of reinforcement for measuring utility in deductive and inductive inference.

It is obvious to see that most of these dimensions are dependent or counter-dependent, and this is admissible provided they represent intuitively different and useful measures. This phenomenon is also motivated by the use of different representational mechanisms, and some of these 'dependences' cannot be elucidated in an absolute way because they may depend on the descriptional mechanism. In this sense, it is stronger the intention to allow that these measures could be applied to any representational mechanism, although in some cases some minor restrictions could be assumed, in order to allow finer and more practical measures.

Some derived measures for optimality of the representation and the whole behaviour of any deductive system will be sought. For deduction, this can be done without the collation with the outer experience. This inner feedback, similar to the notion of experimental mathematics, is clearly shown in game theory, a topic that was also tackled by Hintikka jointly with the distinction between depth information and surface information [Hintikka 1973]. Some classical notions of philosophy of mathematics will be attempted for a clarification, as the choice of the set of axioms and important theorems to work on. Utilitarian and simplicity justifications [Tymoczko 1986] are usually pointed out but rarely formalised.

For induction, though, the outer evidence is the main (but not exclusive) factor that determines the goodness of a theory. Nonetheless, an inductive theory can be constructed for different purposes: explain the evidence, predict future evidence, to describe the evidence, to be comprehensive, etc.

Finally, the combination of induction, deduction, confirmation and information gain will be particularised for the evaluation of logical theories, but its application to different aspects of modern databases and complex software systems will be essayed.

1.4 Overview and Organisation

The rest of this work is organised as follows:

Chapter 2, *On Inference Processes and Their Relationship*, reviews some necessary background about inference processes and their relationship. Although a brief description and history of deduction, induction, abduction and analogy is discussed, the emphasis is lain on computational approaches for both induction and deduction, mainly in a logical framework. Automatic Theorem Proving (ATP), usually known simply as automated reasoning, and resource-bounded rationality are highlighted as the computational approaches of deduction which more urgently require the integration of evaluation measures because inductive techniques are beginning to be used. On the other hand, Inductive logic programming (ILP) is shown as a machine learning paradigm which uses logical theories as representational language and the role of deduction could also be exploited. Other inductive paradigms are just referred to the literature, such as grammatical inference, propositional learning, and artificial neural networks. Obviously, it is vain to address the nature of even a single inference process in one chapter, or even in a book, but a brief review of them may show their differences, relationships and similarities. Among the latter we have some important ones that motivate this work: first, every inference is usually guided by an interest to obtain a new assertion or new knowledge, not *explicitly* present previously and, secondly, the result of an inference process must be evaluated in order to discern if the result is *valuable* enough to be preserved or discarded (forgotten), according to the *effort* which has been performed to obtain it, its plausibility or degree of *confirmation*, and its interest or *utility*.

Chapter 3, *Information and Representation Gains*, introduces the first and most theoretical measure of the thesis. The main purpose is to evaluate the amount of information that has been made explicit in a reasoning step. Initially, a measure of time-ignoring information gain $V(x|y)$ represents the degree of information of x which is implicitly in y. For *non-omniscient systems*, where the notion of effort makes sense, the intuitive notion of information is re-understood in terms of resource consumption. The choice of the function LT, which weighs space and time, as an appropriate measure of effort, neglects the idea of effort exclusively based on time or space. A new effective function, called *computational information gain* $G(x|y)$, which depends on the computational effort (time and space), measures the proportion of x which can be easily obtained by the help of y. Some of its properties are studied, and it is compared with different informal but outstanding notions: implicitness vs. explicitness, some questions about aesthetics and interestingness. Finally, some notions for whole systems or theories are introduced, such as Representation Gain, a general notion of Simplification and the definition of a Representational Optimality criterion.

Chapter 4, *Information Gain and Inference Processes*, takes advantage of the definitions and measures given in the previous chapter. Computational Information Gain, namely G, is used to explain the informativeness of a hypothesis with respect to some evidence and to explain the gain or the reduction of effort that takes place when a conclusion or theorem is deductively established from an axiomatic system. In the case of induction, Popper's idea of informativeness is grasped by the use of G. Moreover, a new notion of authentic learning is introduced, ensuring that learning has taken place, independently of how compressible the evidence is, unlike the MDL principle. In the case of deduction, different adaptations of G are introduced for several deductive paradigms. Appropriate approximations for logical programs are derived and illustrated, which make it possible for measuring in practice these gains. This chapter also includes a comparison with Hintikka's ideas, establishing the relationship between G and Surface Information, and between V and Depth Information. Several general measures of System Optimisation and Systematic Power are also introduced, which show the usefulness of Intermediate Information in ATP and mathematical practice. The conciliation among induction, deduction and information is made possible if omniscience is neglected, although it is recognised that a measure of utility or plausibility is also necessary to account for the whole *value* of an inference process.

Chapter 5, *Constructive Reinforcement,* presents an operative measure of confirmation for general constructive theories, studying the growth of knowledge, theory revision, abduction and deduction in this framework. The new approach performs an apportionment of credit with respect to the 'course' that the evidence or set of derivables makes through the rules of the learnt/axiomatic theory. For the case of induction it is shown to be both a utility and plausibility criterion, and it is connected with other classical evaluation criteria, such as cross-validation and the MDL principle. For the case of classical deduction, since confirmation is fully propagated, it turns out to be a utility criterion that establishes how useful a property, lemma or theorem is for the rest of the theory. It is also applied to other inference mechanisms, such as analogy, approximate deduction, abduction and explanatory induction, the latter represented by a balanced distribution of reinforcement, so formalising the notion of consilience. The theory is also extended with negative reinforcement, so connecting this approach with more classical notions of reinforcement, based on rewards and penalties. In the end, reinforcement and information gain are compared.

Chapter 6, *Intensionality and Explanation,* addresses the problem of formally distinguishing between an extensional definition or description and an intensional one (or by comprehension)[6]. This notion is quite difficult to grasp formally for finite concepts because there are many different ways to *disguise* an extensional description to look like an intensional description. A first formalisation for the case of logical theories of the idea of intensionality is introduced in terms of avoidance of exceptions, these seen as extensional or non-validated parts of a theory. The definition of intensionality and reinforcement to any descriptional language is essayed, based on a formal and general definition of subprogram, but the formalisation is much more complicated that the one made for rule-based representational languages. Different concepts based on descriptional complexity are introduced, such as projectible descriptions and stable descriptions to account more easily for the notion of intensionality in general. The final approach allows the definition of an explanatory variant of Kolmogorov Complexity, which allows to define an explanatory counterpart to the MDL principle. Some connections are also established. First, intensionality is closely related to information gain, since

[6] Intensional (or indexical) logic is the study of assertions and other expressions whose meaning depends on an implicit context or index, such as time or spatial position. This type of logic was originally developed (by Kripke, Carnap, Montague, Church, Tarski and others) to help understand natural language, in which such expressions abound. Obviously, an indexical definition cannot be extensional because it depends on other definition or concept in order to find the thing referred. However, our notion of intensional definition is self-contained and is quite different from the notion of indexical expression. The use of this logic should not be confused with the proposal of chapter 6, although some roots of the difference between induction and abduction may be found in the distinction between self-contained explanations or contextual explanations.

extensional descriptions are neither intensional nor informative. Secondly, explanation is also related to the notion of unquestionability, which is given when there are not alternative explanations, and the notion of comprehension, both notions being necessary for chapter 8.

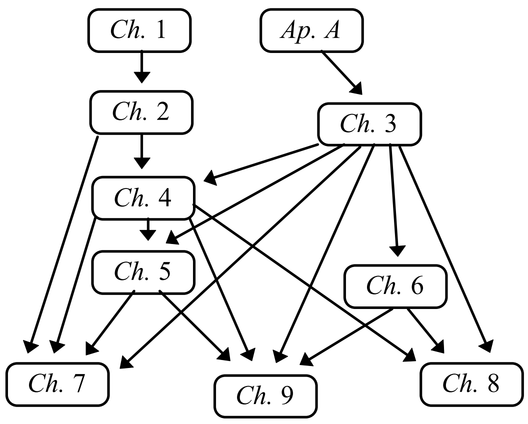
Chapter 7, *Evaluation and Generation of Logical Theories*, initially reviews the most classical criteria for the evaluation of Logic Programs used in ILP, especially two variants of the MDL principle. Next they are compared with reinforcement, intensionality and gain, as defined in the preceding chapters, and the first positive results are shown. In terms of plausibility, reinforcement is manifestly better than the MDL principle, either for whole positive evidence, partial positive evidence and partial positive and negative evidence. Intensionality can be computed to know in which degree the data is 'conciliated' by the theory, and in some cases it can be a prerequisite (abduction, explanatory reasoning, etc.). Finally, for the case of evaluation, gain has only some auxiliary use, mainly for ascertaining when a real learning has taken place, i.e., the theory is original with respect to the data. Apart from evaluation, the question of how reinforcement and gain can be combined for guiding a machine learning algorithm is discussed. First, it is shown that the enumeration algorithm is compatible with an increase of gain, because the theories are not data but hypothesis driven. Secondly, a data-driven approach can still be constructed with the help of randomised approaches such as genetic programming, where the selection criterion (oblivion criterion) is a combination of the optimality of the program (the individual) and the gain (unusual or rich genotype).

Chapter 8, *Measurement of Intellectual Abilities*, presents the most fascinating application of this thesis. Initially, the main factor of intelligence is identified as the ability to comprehend, derived from the notion of comprehension introduced in chapter 6. However, some technical problems arise when this factor is to be measured, especially unquestionability, as originally defined in chapter 6, and an absolute scale of difficulty of comprehension. Both problems are solved in this chapter and the result is a comprehension test, or C-test, exclusively defined in terms of universal descriptional machines. Despite the absolute and non-anthropomorphic character of the test it is equally applicable to both humans and machines. Moreover, it correlates with classical psychometric tests, thus establishing the first firm connection between information theoretic notions and traditional IQ tests. From here, a factorisation is outlined, considering other inductive and deductive factors, thus allowing a theoretical study of their inter-dependence, something that has only been possible in an experimental way, by the statistical correlations studied in psychometrics.

Chapter 9, *Prospective Applications,* includes some proposals that are mainly at a theoretical stage. The first application is given for *information systems*, popularly known as databases. According to the optimal representation measures seen in chapters 3 and 4, the best organisation for deductive databases is discussed, in order to improve the performance of database operations depending on which operations are more frequent and the degree of regularity of the data. Finally, both deductive and inductive processes (and their integration) will be increasingly more important in future databases, which will be better known as knowledge bases or knowledge systems, with data-mining (inductive) abilities. Another application is the study of validation and maintenance characteristics of software systems under the analogy between software science and philosophy of science or, more precisely, between software construction and machine learning. Reinforcement measures from chapter 5 are adapted to define a measure of software 'predictiveness', which is identified with software validation, to represent the stability of a system. An inversely related measure, the probability of modification, is also obtained for each component and for the whole system. Some models of maintenance cost are presented, based on a detailed combination of predictiveness and modifiability, and different software arrangement topologies are studied theoretically under these models. Finally, some other applications are outlined, especially related with language, meaning, and communication, and their applications to agents communication.

Finally, Chapter 10, *Conclusions*, comments on the results of this work, its main contributions, the open questions and the future work. The Appendix A is devoted to give a quick and comprehensive review on Kolmogorov Complexity and some of the properties and related concepts which are used from chapter 3 to chapter 8. The Appendix B references the publications originated from this thesis. The work is closed by Appendix C, which contains any reference which is alluded to in this dissertation or have been used as a base or motivation for this work, Appendix D, which includes the acronyms used throughout the dissertation and Appendix E, an analytical index.

The following figure illustrates a graph of dependences of this dissertation. The graph has only suggestive character because the chapters have not been written to be read independently, and a sequential reading is more recommended:



Graph of Dependences

Since chapter 2 is a review of inference processes, anyone who is familiar with them, namely deduction, induction, abduction and analogy may prefer to take only a reading to its last section. This section is devoted to their relationship, which is aimed to be clarified in this work.

1.5 Terminology and Notation

In general, notation will be introduced when needed in each chapter. Even acronyms are always quoted in their extended form the first time they appear (and a list of acronyms is included in appendix D).

Nonetheless there is some basic terminology, which will be used throughout all the dissertation, that I have preferred to include here. The reader may come back here from any subsequent chapter if any notation seems odd.

Unless specified, a finite alphabet Σ composed of symbols will be used. If not specified, $\Sigma = \{0, 1\}$. A string, concept or object is any element from Σ^*, with $\cdot$ being the composition operator, usually omitted or represented by $<a,b> = a \cdot b$. The empty string or empty object is denoted by ε. The term $l(x)$ denotes the length or size of x in bits and $log\ n$ will always denote the binary logarithm of the value n. The relation $<_{lex}$ between two objects denotes precedence in the left-to-right lexicographic order, considering $0 <_{lex} 1$. The term $y_{n..m}$, with $n \leq m$, denotes the symbols from position n to position m. Note that by a position we refer to the virtual space between two symbols, i.e., $y_{0..l(y)}$ denotes the whole string y, and $y_{0..l(y)-1}$. Consequently there is always one more position than symbol in any string. For every string y and every natural number n, $y_{n..n} = \varepsilon$. With $y_{..m}$, $y_{n..}$, and y_k we denote $y_{0..m}$, $y_{n..l(y)}$, and $y_{k-1..k}$, respectively. A string x is a substring of y iff there exist two strings z,w such that $y = zxw$, or, what is equivalent, $y_{n..m} = x$ with $n = l(z)$ and $m = l(z) + l(x)$. A string x is a prefix of y iff there exists a string z such that $y = xz$, or, what is equivalent, $x = y_{0..m}$ being $m = l(x)$. Given any string x, $x_{-d} = x_{0..l(x)-d}$ denotes the prefix of x with length $0..l(x)-d$, i.e. the string x without its d elements. The term $x_{\neg i}$ represents exactly the same string as x but changing the *ith* bit by its complement., i.e. $x_{\neg i} = x_{0..i} \cdot 1 \cdot x_{i+1..l(x)}$ iff $x_i = 0$ and $x_{\neg i} = x_{0..i} \cdot 0 \cdot x_{i+1..l(x)}$ iff $x_i = 1$.

Some theoretical results which are obtained in this work are asymptotic, and the following notation will be quite useful: $a \overset{+}{=} b$ means $a = b + O(1)$, $a \overset{+}{<} b$ means $a < b + O(1)$, and $a =^{\log} b$ means $a = b + O(log(var(b)))$, where $var(b)$ denotes the variable or size of b.

A universal machine ϕ will be any machine which can emulate a universal Turing machine. $\phi(p,y)$ denotes the result of the execution of p in ϕ with input y. $Cost_\phi(p,y)$ denotes the computational cost (steps of the machine ϕ) of executing the program p with input y. From here, the following definitions can be given,

Definition 1.3 Kolmogorov Complexity.

$$K(x|y) = \min \{ l(p) : \phi(p,y) = x \}$$

It is also supposed that the machine ϕ uses a prefix-free codification method for programs.

The term x^* denotes the first minimal program for x in enumeration order. Consequently $l(x^*) = K(x)$.

Definition 1.4 Levin Complexity.

$$Kt(x|y) = \min \{ LT(p) : \phi(p,y) = x \}$$

where $LT_\phi(p) = l(p) + log_2\, Cost_\phi(p,y)$.

These two definitions just only give a hint about Kolmogorov (or Algorithmic) complexity K and its space-time variant Kt. Many other properties and derived concepts will be used or referenced throughout all the dissertation. Hence, it is recommended that the reader takes a look first at appendix *A* if she is not familiar with the theory.

2. On Inference Processes and their Relationship

For every belief comes either through syllogism or from induction.
Aristotle, Prior Analytics, Book II, Chapter 23, 330 BC

Abstract: *This chapter gives a quick account (or recall) of inference processes: deduction, induction, abduction and analogy. More specifically, the emphasis is lain on computational approaches of induction and deduction, mainly in a logical framework, such as Automatic Theorem Proving (ATP), Resource-bounded Rationality, and Inductive Logic Programming (ILP). Instead of highlighting their differences, the major similarities among them are considered. Any inference process requires an effort to make explicit something that was implicit, and how this is distributed determines the difference between lazy inference methods and eager ones. It is also shown how inference processes can be equally understood in terms of confirmation, provided a quantitative (and not a qualitative) propagation is used. Finally, the combination of different inference processes is discussed and some precedents in this line illustrate the need of unified evaluation measures for all of them.*

Keywords: Inference, Logic, Reasoning, Induction, Deduction, Abduction, Analogy, Information, Semantic Information, Explanation-Based Learning, ILP.

2.1 Introduction

Any work that deals about deductive and inductive inference cannot begin from scratch. It would be folly to do that. However, the acceptance of the heritage of centuries of philosophical enquiries about the matter has the advantage of using what has been solved and clarified, but also has the drawbacks of reluctant paradoxes and weak foundations, carried along the way. It is not my intention to open a discussion about these foundations, especially about the problems of inductive inference. The reader should not consider this a disclaim from discussion, but the first sections of this chapter must be understood as a kind of anthology and introduction of notions which may be useful for the rest of the dissertation. Anyone familiar with deduction and induction can step directly to the last section of this chapter, which discusses the problems and utility of the combination of different inference methods.

Let us begin by introducing a definition of inference: "a process of reasoning by which an agent modifies (part of) its beliefs". According to this common definition, there are some important traits of any inference process that are implicit in it. First of all, there is an intentional character in any inference process, with the aim to obtain new knowledge from some new data or previous beliefs, either real or imaginary. Secondly, there is an epistemological character of any inference, because it provides *novel* information, which was not explicitly known before the inference process. This *new* knowledge may be a concept (a fact, a rule or property), or the refutation or confirmation of a previous or assumed belief. Any intermediate plausibility assignment for a belief, between refutation (this is not the case) and confirmation (this is the case), is also possible, and is given by the degree of reinforcement that several inference processes have assigned from other beliefs and their corresponding 'plausibilities'. Note that, in this context, the fundamental issues for understanding inference seem to be the notions of information, novelty, belief, explicitness and reinforcement.

In some way, any inference process involves an argumentation with own's beliefs, and, in principle, is more related with *dialectics* than with *analytics*. However, in the context of logic, it is usual to see a distinctively different notion of inference, as *correct argumentation*. Hence, the interest is to discover sound rules of inference to make a *logic*, in order to associate "what can be inferred" with "what follows logically". This is what is sometimes called "the logico-maniacal version of inference" [Vega 1987] which is usually accompanied with assumptions such as "the result of any inference is a necessary result from its premises", "an inference is only justified if and only if it is a correct application of a rule of inference", which leads to the paradox of inference that I have commented in the previous chapter and I will also discuss in this one.

This view of deduction has been majority since Aristotle (384/3-322 BC) until the XXth century, and logic has been mainly concerned in distinguishing valid inference schemata from invalid ones. This view, however, precluded the study of some other kinds of plausible inference to account for a theory of plausible inferences which are used in everyday situations.

Accordingly, the kind of valid or truth-preserving inference, or what [Flach 1995a] properly calls satisfaction-preserving inference[7], more commonly known as *deduction*, was, during many centuries, granted the prominence (when not the exclusivity) of reasoning. On the contrary, *induction*, as the process of inferring plausible general rules or concepts from a factual evidence, had been indirectly addressed by Plato (427-348 BC), in his study of perception and reality, and by Aristotle himself, but it is not until the works of the philosophers Bacon (1561-1626), Locke (1632-1704), Berkeley (1685-1753), Hume (1711-1776), Kant (1724-1804) and Mill (1806-1873) when an important and deserved role is given to inductive reasoning.

More recently, during the XXth century and mostly under the field of Artificial Intelligence, there has been an increasing interest on *other* inference processes. Abduction is a kind of hypothetical inference introduced in the XIXth century by Peirce (1839-1914) because, in his opinion, neither deduction nor induction, alone or combined, could unveil the internal structure of *meaning* [Yu 1994]. Recently, abduction has been shown to be one of the most practical inference mechanisms for AI applications such as diagnosis or problem solving. Analogical reasoning has also been elected as a different reasoning mechanism that *pervades all our thinking* [Polya 1957]. However, the great difference of analogy with respect to deduction, induction and abduction is that the result of an analogical inference process is not intended to be plausible, and this process is more related with creativity, value, problem-solving abilities, intelligence, etc., than with validity or correctness.

The existence of different inference processes has led to the accepted view that *reasoning* must be composed of many of them, with diverse characteristics, aims and mechanisms, and in a *convenient combination.* In order to achieve a successful combination it is necessary to ascertain the characteristics and differences of each of them. A first classical characterisation of deduction as a truth-preserving inference process and the rest of processes as plausible (hypothetical) ones has changed radically after the development in the XXth century of different modal and non-classical logics. Although in Prior Analytics, Aristotle also introduced the four most important modal particles; the extension of the figures was never intended to fall outside the framework of what is valid or what is not. However, recent non-classical logics, such as probabilistic, fuzzy and nonmonotonic logics, are not truth-preserving, in the way that *deductive* inferences are assigned a degree of probability or,

[7] Michalski affirms that deduction is truth-preserving while induction is falsity-preserving [Michalski 1993].

simply, they are just plausible in a more or less vague way and, finally, can be defeated. Hence the name defeasible logics for them.

Consequently, the dual classification between deduction and other inference processes cannot be found, in general, in terms of validity or truth-preservation. It cannot also be found in a difference among general to specific (deduction), specific to general (induction) and specific to specific (abduction and analogy) since it is easy to find different counterexamples. In fact, mathematical induction is a truth-preserving mechanism by which a general property can be derived from single facts, which, in fact, was also first introduced by Aristotle in his Posterior Analytics.

Simplistically, but more pragmatically, it is usual to characterise these inference processes in the following way, under semantic considerations. Given the following implication,

$$A, B \models C$$

each process can be understood as taking *A*, *B*, *C* as inputs or outputs.

- **Deduction**: if *A* represents the axioms or premises, *B* the background knowledge, and *A* and *B* are given, then *C* can be obtained by deduction, and it is called the consequence of *A* in the context of *B*. An example of deduction could be "Input: Every banana is yellow. This fruit is a banana. Output: this fruit is yellow".
- **Induction**: if *B* represents the background knowledge, and *C* represents the evidence, and *B* and *C* are given, then *A* can be obtained by induction and it is called a rule or description of *C* under *B*. An example of induction could be: "Input: These 6 fruits are yellow and 5 of them are bananas. Output: Presumably, every banana is yellow".
- **Abduction**: if *B* represents the background knowledge, and *C* represents the evidence, and *B* and *C* are given, then *A* can be obtained by abduction and it is called the explanation or assumption of *C* under *B*. An example of abduction could be: "Input: This fruit is yellow. Bananas are yellow. Output: This fruit may be a banana".

Note that the distinction between induction and abduction as it has been illustrated is only terminological. More concretely, in the case of induction, it is usually assumed that *C* is a set of examples and *A* is a general *rule* for them. In the case of abduction, *C* may be a single fact and *A* is usually also a fact that explains the occurrence of *C* under the context *B*. In many cases, this *A* should be selected from a set of *abducibles*.

Sometimes the difference between deduction, induction and abduction is made in the context of causality. Given a rule, deduction would discover the future effect of a perceived cause. Abduction would discover the past cause of a perceived effect. Induction would discover the rule given the cause and the effect repeatedly. However, this view also presents some problems because deduction is sometimes propagated backwards to the cause or conditions (e.g. "The light is on. Necessarily the lamp is ok"), although some authors would say that this is an abduction.

Distinctively, analogy cannot be seen in the context of an implication of the form $A, B \models C$. On the contrary, it must be seen in a context of similarity or association. In fact, analogy is sometimes considered a suggestive process (or source of connections) rather than an inference process.

- **Analogy**: Given A is-to A' and C, obtain C' such that A is-to A' as C is-to C' in the context of B. An example of analogy could be: "Input: John has three children. Yesterday, he entered Susan's shop and bought three scarves. Peter has two children. Today, Peter has entered Susan's shop. Output: Possibly, Peter has bought two scarves.".

In some way, analogy can be described as an induction to a temporary rule, followed by a deductive or abductive inference over this rule. For the previous example, the generalisation is that "every father buys in Susan's shop as many scarves as the number of children he has", which, although quite unbelievable in general, may serve to draw the deductive conclusion that "Peter has bought two scarves". An abductive conclusion would be given in the case the example would be changed to "Input: John has three children. Yesterday, he entered Susan's shop and bought three scarves. Today, Peter has entered Susan's shop and has bought two scarves. Output: Probably, Peter has two children". The context is extremely important for analogy. Consider for the first example that yesterday it was terribly cold and today it is sunny and hot. Maybe, Peter has bought two bathing shorts.

Analogy suggests another way of classifying an inference process. Some inference processes work *on the fly*, i.e. they are lazy, in the way that they are only used when needed, such as analogy or abduction. Other inference processes, though, are more eager, in the way that they try to obtain concepts or rules that would be necessary in the future, as constructive induction performs. Finally, deduction is sometimes lazy, such as everyday deductive inferences, and sometimes eager, such as mathematical practice.

In this chapter we will make a quick review of these different inference processes and some of their most successful computational realisations and, finally, their relationship and the problems of their combination. Obviously, it is vain to address the nature of even one of them in a single chapter, but a brief review of them may show their differences, relationships and similarities. This would highlight some deficiencies of their combination which are given mainly due to a poor measurement of their value, as it was shown in the motivations of the previous chapter and as it is going to be further justified here.

2.2 Deduction

Surely, deduction is the inference process that has been studied more deeply in the history of philosophy, mathematics and logic. In fact, it is the inference process *par excellence* and it is even still considered a synonym for inference or reasoning, although this chapter is partly devoted to make the reader forget that idea. The word deduction has usually been related with logic, consequence, entailment, proof, syllogism, truth, etc. In the end, although it is the best-known inference process, there still many open and fundamental questions, what gives a hint of how poorly known the other inference processes are.

The formulation of logic, and the formalisation of deduction within it, is known and used nowadays in the way that Boole and, mainly, Frege, developed in the XIXth century. Later on, up to the half of the XXth century, some crucial ideas for modern logic were developed: Russell's theory of types, Gödel's incompleteness theorems, Skolem functions and Herbrand Universe (also due to Skolem), the notion of interpretation and unification (Herbrand), Natural Deduction and Sequent Calculus [Gentzen 1935][Prawitz 1965].

But it is the advent of computer science and the notion of computation that forces a re-understanding of deduction. The works of Turing, Post, Church and Kleene established the major connections among lambda-definable functions, Turing-computable functions, general recursive functions and the intuitive notion of computable functions. The notion of computable function is crucial for different reasons, but for the case of deduction is especially important for two things. First, the most important notion which is derived from computation is precisely that of incomputability, which, translated to deduction, corresponds to the notion of not provable or undecidable propositions. Secondly, the connection between lambda-definable functions and computable functions allows the extension of most results given in computation to deduction, mainly complexity results derived from Blum's notion of 'computational cost' and 'complexity classes' [Blum 1967]. In other words, not every feasible deduction takes the same amount of time to be performed, i.e., computed.

This has allowed talking about three dimensions: consistency, completeness and pragmatics of any deductive system. The connection between the first two dimensions began with Gödel's famous incompleteness result: it is well known that a consistent system that is able to express arithmetic cannot be complete. More recently, the study has centred on the relation between the two last dimensions: completeness and pragmatics. It has been shown that even for very restricted representation mechanisms (such as propositional logic) if completeness is desired, there is no efficient deduction method[8]. Only very reduced representations, such as

[8] Provided NP $\neq$ P, i.e. Non-Polynomial problems are not reducible to Polynomial ones.

strict subsets of propositional logic are consistent, complete and computational feasible (or, what is equivalent, polynomial).

The relationship among these three dimensions was also studied in the most general case by Chaitin who established the correspondence of Gödel incompleteness results in terms of descriptional complexity [Chaitin 1982]. More importantly, he showed the apparently intuitive conclusion that a compromise is to be found among the space of the theory, its degree of (in-)completeness and its computational complexity. In his words: "*any formal system in which it is possible to determine each string of complexity less than n has either (...) few bits of axioms and needs incredibly long proofs, or it has short proofs but an incredibly great number of bits of axioms. (...)This is analogous to the dilemma of a scientist who must choose between directly publishing his observations, or publishing a theory that explains them, but requires very extended calculations in order to do this.*" [Chaitin 1974].

As a result of these many negative results (and the strictness of classical logic for artificial intelligence applications), there has been an enormous effort towards non-classical logics (see e.g. [Haack 1978] for a review). Modal logics, although already introduced long ago by Aristotle, have been formalised recently, under the works of Lukasiewicz, Lewis, Carnap, Kripke, Hintikka and Lemmon. From the possible worlds semantics [Moore 1984] and Kripke's semantics [Kripke 1963], different logics of believe and modal logics are studied under the following axioms [Konolige 1992]:

$$(K)\!: L(\phi \supset \psi) \supset (L\phi \supset L\psi)$$
$$(D)\!: L\phi \supset \neg L \neg \phi$$
$$(T)\!: L\phi \supset \phi$$
$$(4)\!: L\phi \supset LL\phi$$
$$(5)\!: \neg L\phi \supset L\neg L\phi$$
$$(P)\!: \phi \supset L\phi$$

Axiom (K) is known as the axiom of deduction, (D) is the non-contradiction axiom, (T) is the axiom of infallibility, (4) is the axiom of the conscience of own's knowledge (or positive introspection axiom), (5) is the axiom of the conscience of ignorance (or negative introspection axiom) and (P) is the axiom of complete wisdom.

Some combinations of these axioms have intuitive interpretations and others are just impossible or lead to very intuitive results (see e.g. the discussion about the different uses of S5, KD45, K45 in [Halpern 1997]).

Other logics are many-valued logics (Lukasiewicz[9], Rosser), free logics (Lambert), intuitionistic logic ([Brouwer 1907] and [Heyting 1920]), constructive logic ([Martin-Löf 1982]), linear logic (Girard), dialogics (Lorenzen), combinatorial logic ([Curry 1920]), deontic logic ([von Wright 1951]), epistemic logic (Hintikka 1962), pragmatic

[9] See the essay "On three-valued logic" (1920) in [Lukasiewicz 1970]).

logic (Montague[10]), intentional logic ([Zalta 1998]), restrictions of predicate logic formalising semantic networks ([Woods 1975]), such as terminological or description logics [Brachman 1977] [Baader et al. 1992] [Patel-Schneider and Swartout 1994] [Donini et al. 1997], fuzzy logic [Zadeh 1965, 1972], probabilistic logic [Nilsson 1986] and other non-monotonic logics [Marek and Truszczynski 1993] [Antoniou 1997]: default logic (Reiter 1980), defeasible logic (Nute 1988, 1991), possibilistic logic, temporal logics (situation calculus [McCarthy 1968], event calculus [Kowalski and Sachi 1997]), etc.

Although some of them have been applied successfully (deontic, fuzzy, temporal, many-valued) or have provided interesting theoretical results (intuitionistic, linear), many of the problems of *pragmatics* have not been solved by these variants (except to those which restrain considerably the expressiveness such as description logics [Borgida 1996]). Additionally, higher-order logic [Leivant 1994] has developed pragmatic and creative inference methods and techniques, because higher-order deduction is incomplete, and the relevance is then given to tractability.

There are two areas where pragmatics has been addressed as a fundamental or even foundational issue. The first one is *automated deduction*, which deals with feasible computational deduction and, consequently, has given one of the most successful applications for computer science, the logic programming paradigm. The second one is called resource-bounded deduction, and it is devoted to employ optimally the computational resources available (space and time) for deductive problems. The paradigm is also applied to other problems that require reasoning (and this is the reason why it is also called resource-bounded rationality). Let us review both areas.

2.2.1 Automated Deduction and Logic Programming

The idea of an automated calculus of logic is firstly introduced by Llull (1235-1315) with his Ars Magna, which would be determinant for Leibniz's (1646-1716) "De Arte Combinatoria" where the notion of logic as reckoning or calculus is explicitly vindicated or, more exactly, longed for. But it is not until the advent of the first computers that automated deduction becomes a reality.

The aim of the first systems was Automatic Theorem Proving (ATP) and this may be the reason for ATP being an alternative name (although more restrictive) for automated deduction. These first essays evolved very closely to the development of the first years of artificial intelligence. There were two main streams. A first one tried to emulate human mechanisms of reasoning justified by the fact that mathematician do no use strict rules and symbolic rules to prove a theorem. The second stream considered logic a better tool to obtain their goals. The best results have always been given by the second approach, as it is clearly justified by Alan Bundy: "[…] *logic-based approaches have been frequently criticised, but have survived due to the failure of alternative*

[10] See (Dowty et al. 1981).

approaches. In fact, most attempts to produce non-logic based automated reasoning end up reinventing logic — except that the new wheel is usually more hexagonal than circular. Techniques such as semantic nets, frames and production rules have each had a brief flowering, before being recognised as the old wolf in sheep's clothing. There are successful non-logic based techniques, for instance neural nets, but they cannot simulate sustained arguments, e.g. mathematical proofs." [Bundy 1991].

Inside of the logical stream, the initial idea was to put into practice the most popular deductive systems, in particular Gentzen's sequent calculus. Accordingly, for the first systems that were implemented, such as SAM (Semi-Automated Mathematics), human intervention was crucial. In 1966, the release 5 of this interactive prover proved the first new lemma for mathematics (in *lattice-theory*), that, in its honour, is called SAM's lemma.

However, these efforts drew little attention to mathematicians. On the contrary, and due to the growth of the field of program verification motivated by the software crisis of the sixties, computer scientists were indeed the most interested in an automatisation of proofs, which, in the case of program verification, were especially tedious.

But in 1963, when the field seemed more open, Robinson introduces the resolution principle [Robinson 1965]; it is in the Argonne Laboratories where Wos, Robinson and Carson programmed in 1965 the first prover based in binary resolution. The same year, Robinson introduces the concept of hyper-resolution [Robinson 1965b]. This turns out to be a major swift towards the paradigm introduced by Herbrand in 1930, especially the unification algorithm, (re-discovered by Prawitz in 1960), and the theoretical concepts of universe, interpretation and model.

In 1969 Green discerned the possibility of using the resolution principle to solve problems of any kind [Green 1969] and, just in 1970, the ATP community provided one of the most contributions to AI, the programming language Prolog. The same year, Kowalski presented its formal semantics and Colmerauer designed its first interpreter.

The use of a theorem prover as a programming language [Kowalski 1974] boosted the paradigm of logic programming. In addition, first-order logic was even more used than initially expected in many applications of artificial intelligence and computer science (e.g. databases). In this context, the works on variants of resolution have been countless since then. Only in 1978, Loveland [Loveland 1978] already included in his book 25 variants of resolution.

From this moment and after the initial important applications and successes of SLD (selective linear resolution for definite clauses) and SLDNF (selective linear resolution for definite clauses with negation as failure), discouragement soon appeared; resolution is complete and improves the combinatorial explosion but it does not eliminate it. The negative results re-appeared, as if they had been forgotten: the proof of an arbitrary fact under resolution is NP even for the case of propositional logic. ATP Programs of the time got lost in many useless ways to find the proof of complex theorems.

In the three last decades, Automatic Theorem Proving has advanced remarkably in techniques, results and applications, according to the proof of new non-trivial theorems, as well as the time that is required to prove some existing ones with current systems. According to [Wos 1994]: "*to gain an appreciation of the progress that has occurred in this young field in but three decades, one need only realize that the more effective programs used now are more capable of solving problems that require reasoning than are the vast majority of people, even if restricted to university students. Perhaps the error that underlies the position taken by those who assert that automated reasoning programs offer little and that the field has made essentially no progress rests with comparing these programs to the better* (sic) *mathematicians and logicians*".

Most of these advances do not come from new theoretical results in logical systems, but from the use and development of AI techniques, e.g. heuristics, the realisation of the relevance of *intermediate* information and a convenient use of resources (memory and time). However, the techniques that are beginning to being used are more related with hypothetical inference than deduction: "*Concepts like analogy, use of examples and counterexamples, special cases, and much more of this kind would account for an additional dimension to be added to our space of methods. It is only then that mathematicians would begin to get truly interested in using such systems in their daily work and discover proofs perhaps even for long-standing and well-known conjectures in a cooperative way*" [Bibel 1991]. For instance, some new approaches such as that of [de la Tour et al. 1987] and some new tools such as the resonance strategy [Wos 1996] use analogy.

Abstraction has also been one of the main techniques in automated deduction, which, if it is used hand-made corresponds to the use of schemata, as they were introduced by [Plaisted 1980], where the skeleton of first-order proof is extracted to apply to other proofs. The idea has been more and more sophisticated since then and nowadays the proofs are made through the use of 'proof plans' [Bundy 1991].

Furthermore, these plans and meta-information are beginning to be incorporated in higher-order deductive systems. An example of this is the HOL system, a typed higher-order deductive system, based on the LCF system, which used rewriting techniques. Additionally, HOL also includes backward proofs, based on the notion of tactic, introduced by Robin Miller [Milner 1978]. A remarkable feature of HOL is that the system is extensible and the set of consequences (usually infinite) of a theory are not considered until they have been proven in the system. In other words, theories are extensible and dynamical systems.

The current stage, as we have said, is the introduction of inductive generalisation and hypothetical reasoning in these systems, in order to discover new schemata. This is easier to do in higher-order systems because they are able to represent inside the language the concepts of proof, tactic, schema, etc. Finally, a new question arises: is the use of inductive techniques for deduction a paradox?

2.2.2 Resource-Bounded and Non-omniscient Deduction

Resource-bounded reasoning was introduced by I.J. Good [Good 1971] and H. Simon [Simon 1982] as a way to make reasoning in the large feasible for artificial intelligence applications [Russell and Wefald 1991]. The most technical stream of this proposal evolved from the theoretical notions of approximate or anytime algorithms. "Anytime algorithms offer a trade-off between computation time and quality of results" [Zilberstein 1995]. This resource-bounded rationality (see [Zilberstein 1996] for a survey) involves randomised heuristics in order to obtain approximately the solution of a problem. The reason for this approach is simple: for some problems "*it is not feasible (computationally) or desirable (economically) to compute the optimal answer*" [Zilberstein 1996]. In the case of deduction, under this context, the idea is not to find a proof for a theorem in some system but obtain a degree of plausibility for it. Since it is inspired in approximate algorithms, the techniques are usually based in Monte-Carlo like methods or genetic algorithms, which usually give a probabilistic correct answer whose probability depends on the time the algorithm is supplied. For instance, there are approximate algorithms that give with an astonishing high probability of success, whether a number is a prime, and they are used in practice instead of other exact, but intractable, algorithms.

In the context of first-order logic, entailment and subsumption can be re-defined into resource-bounded entailment or resource-bounded subsumption, respectively [Sebag and Rouveirol 1997]. Obviously, this approach entails new problems, since deduction may be inconsistent. In Zilberstein's words, "*(there are) two sources of uncertainty. The first source is internal to the system and relates to its capability to produce incrementally improving solutions and to assess their quality. The second source of uncertainty is external and relates to unpredictable change in the environment in which the system operates*" [Zilberstein 1999]. The effort has been centred to deal with these inconsistencies.

Non-omniscient deduction is closely related with this entire problem, although it is something more general that must not be necessarily based on approximated deductive methods but originated from different sources, such as multi-agent systems. In the end, non-omniscience is something much more frequent that could be expected because, for any high expressible system (e.g. able to formalise arithmetic), deduction is incomplete, so any deductive inference is valuable, since it not only clarifies the doubt of whether something holds or not (which is implicitly in the premises for a complete system), but provides information about whether it is possible to know whether it holds or not. Even in complete systems (e.g. resolution for Herbrand logic), non-omniscient deduction may also come from a lack of resources, and this is more conspicuous in real-time applications.

In any case, since we are dealing with inconsistencies, it is necessary to consider theory revision. As a result, other inference processes different from deduction are much more necessary and the term non-omniscient rationality (for a review see [Moreno 1998]) is used instead. The idea is to loose the standard epistemic modal axioms of the system and let other inference processes arrive to some conclusions and increase and *improve* its knowledge.

Moreover, when resources are scarce it is necessary to have a measure to evaluate which reasoning actions are valuable to perform according to the expected gain and the resources they require. And this depends highly on the distribution of time-space limitations. In the case of hard spatial limitations, only few costly things should be maintained in an explicit way and time will be used to derive everything from them. On the contrary, if time is crucial and memory is large, then the system will use a lot of intermediate information, in order to accelerate any further needed inference (this is the deductive database viewpoint). However, comprehensive measures to evaluate and act in these situations have not been introduced to date, and as I stated in chapter one, and they are the main goal of this work.

For the approach that begins in the next chapter, not many other concepts about deduction will be needed. A basic notion of deduction as given from an introductory logic course is sufficient, although for some chapters it is required some familiarity with logic programming, where the classical source is [Lloyd 1987]. For the reader who is not familiar with computational notions is also advisable to take a look at some book about logic and computation (e.g. [Boolos and Jeffrey 1989]). Additionally, but not necessarily, the reader may be interested in extending some of the topics which have been so briefly introduced in this section: for a history of logic see e.g. ([Kneale and Kneale 1984] or [Heijenoort 1967]) and for more information about ATP see e.g. [Bibel 1991] or [Mackenzic 1995].

2.3 Induction

The term 'induction' has always been surrounded by controversy. There is no widely accepted definition for it and even some philosophers deny that it can exist such a thing as "inductive reasoning". Nonetheless, almost everyone associates the word induction with the process of theory abstraction from facts, the problem usually faced by a scientist. The main problems arise when it has been tried to "logicise", trying to imitate the same treatment as deduction. It has been even known (in my opinion improperly) as the "logic of discovery", but there has not been any appropriate formalisation of induction under a logical system.

Although the process of induction is still poorly known, its objectives are widely recognised as the construction of a hypothetical theory to *explain* past facts and/or *predict* future experiences. The set of facts is usually called *evidence* or *observations* and the theory is usually known as *hypothesis*. In many occasions, there is a *background* *knowledge* or bias that can constrain the inductive problem. In practice, the synthesis of theories from facts is still nowadays a fundamental problem to cognitive science and philosophy of science. There is a common and broader view of induction, called *learning*, which is used by every human being and many animals, that is still less known and even more intriguing.

As said before, the attention to induction was highlighted by Bacon in his *Novum Organum* (1960) and later converted into methodology by Mill in his *System of Logic* (1843). However, there is also a statistical view of induction, which begins with Bayes (1702-1761) in his "*Essay towards solving a problem in the doctrine of chances*" [Bayes 1764], continues with Laplace and, finally, Boole in his "The Laws of Thought", which was the first applications of logic algebra to probability. Later on, this would allow Carnap to develop its probabilistic calculus. In addition, many purely statistical induction tools have been developed in the last two centuries, e.g. regression techniques.

There are three stages in induction (and hypothetical inference in general): generation, evaluation and confirmation. *Generation* has only been addressed without mysticism recently, under the machine learning community, initially motivated by Turing's paper 'Can machines think?" [Turing 1950], where he argued that a machine could learn if it is programmed to programme itself. Simplistically, there are three main methods for induction: data-driven induction, schema-driven induction and enumeration approaches. Obviously, most systems use a combination of them.

Evaluation is the most remarkable and discussed issue of induction. The concept of *verisimilitude* [Popper 1968] is the level of agreement with the facts. A theory t_1 has more verisimilitude that t_2 if t_1 implies as many true observational sentences as t_2 and has less false observational sentences (exceptions). However, he then argued that the most scientific criterion was that of "falsifiability", i.e., the best theory is the one that is the easiest one to falsify. Intuitively, if a theory is intrinsically easy to falsify and it cannot be falsified by essaying possible examples then it is much more reliable than a theory that provides no way to make experiments to refute it.

Kuhn was more pragmatic. In his opinion, a theory or paradigm that is refuted by a single or more facts can still be used if there is no alternative better paradigm. Obviously, after some time, a great amount of anomalies forces the introduction or generation of another paradigm [Kuhn 1970].

Machine Learning (ML) is the subfield of artificial intelligence that studies the techniques and possibilities for making machines learn. It has been shown that both generation and evaluation issues are extremely coupled in practice. After some initial systems which learned some simple problems from IQ tests, plans and some toy domains, the study took a more theoretical character. The seminal paper of Gold in inductive inference [Gold 1967] introduced the notion of "identification in the limit". In this paradigm several strong results were proven, mainly that even regular or propositional concepts were not learnable in the limit from positive data only [Blum and Blum 1975] [Angluin and Smith 1983].

Due to this discouraging result, [Valiant 1984] introduced Probably Approximate Correct learning (PAC-learning) in an effort to make a more realistic theory of learning, which could be made feasible for more expressible representation languages. The machine learning community implicitly accepted Kuhn's paradigm, because most of its applications deal with approximate learning. With the same aim, Angluin introduced the paradigm of query learning [Angluin 1987], as a learning session that is helped by the possibility of making arbitrary queries to a teacher. After that, most of the papers in computational learning theory are concerned with these two paradigms [Blumer et al. 1987] [Blumer et al. 1989] [Board and Pitt 1990].

In the last decade, there have been many revisions and critiques of these paradigms because they are still too pessimistic, in the way that they always consider the worst case and not the mean case. See for instance [Abe 1997] or [Freivalds et al. 1995].

Finally, *confirmation* is the last aspect of induction. At first sight, it seems the easiest thing: a hypothesis can be refuted or confirmed by the evidence. As we will see, confirmation is not that easy, because "*theories may be refuted, but they cannot be confirmed beyond any doubt*" [Popper 1968]. We will return on *confirmation* in the last section of this chapter.

2.3.1 The MDL principle and other Selection Criteria

From all the selection criteria that have been discussed in the literature of induction, simplicity is the most vindicated and recurrent one. It is attributed to William of Ockham 1290?-1349? (although the same idea was held by John Duns Scotus twenty years before) this theme in philosophy of science and induction:

Occam's Razor Principle: "if there are alternative explanations for a phenomenon, then, all other things being equal, we should select the simplest one".

This principle was rejected by Popper because he said that there *was* no objective criterion for simplicity. But descriptional complexity, $K(x)$, gives an objective criterion for simplicity, as it is seen in appendix A. This is precisely what R.J. Solomonoff proposed [Solomonoff 1964] as a 'perfect' theory of induction, in [Li and Vitányi 1997] words.

Moreover, Descriptional Complexity inspired J. Rissanen in 1978 to use it as a general modelling method, giving the popular MDL principle [Rissanen 1978, 1986, 1996]:

Minimum Description Length (MDL) principle:

The best theory to explain a set of data is the one that minimizes the sum of:

- *the length, in bits, of the description of the theory; and*
- *the length, in bits, of data when encoded with the help of the theory.*

In the second term we enclose the exceptions, if any.

Philosophically, the MDL principle matches with Kuhn's notion of "changing paradigms" [Kuhn 1970]: Exceptions are patched until they are long enough to force the revision of the paradigm (or model) of the theory.

Apart from its success, the first justified reason to use the MDL principle is avoiding overgeneralisation. When two generalisations cover all the cases, we select the shortest one. In some way, the MDL principle finds a good compromise between generality and specificity that improves the predictability of the hypotheses. [Li and Vitányi 1997] give an informal justification of the predictability of the MDL principle in the following way:

> [...] a priori we consider objects with short descriptions more likely than objects with only long descriptions. That is, objects with low complexity have high probability while objects with high complexity have low probability. Pursuing this idea leads to the remarkable probability distribution $2^{-K(x)}$ [...].

If not explained, the term "a priori" makes of this no justification. The arbitrariness of its use is given by the following explanation: when we decide to study a phenomenon that we deem is not random (for instance the course of the planets) it is because we expect some mechanism behind it. A mechanism is an algorithm, so if we expect to know it and to be usable, it has to be relatively simple. So this "a priori" distribution is not so arbitrary, it is a methodological criterion. From here a formal proof derived from Bayes Rule can be found in [Li and Vitányi 1993] (pp. 308-309).

This is the reason why it is commonly said that "the shorter the hypothesis the more predictable it is", which supports the MDL principle. In conclusion, it has been usually recognised by the machine learning community that learning and induction can be understood as data compression [Blumer et al. 1987] [Blumer et al. 1989]. However, as it will be discussed, the MDL principle gives many problems for explanatory induction and, mostly, when combined with deduction.

There are many other evaluation criteria. Among the probabilistic selection criteria, Maximum Likelihood Estimators [Case and Smith 1983] and Bayesian Learning are based on posterior probability. A special case of these is Quinlan's Gain Ratio, which has given the most successful machine learning algorithm: ID3 [Quinlan 1986, 1990]. Other criteria are not probabilistic but still based on statistical roots, such as cross-validation. Cross-validation is based on a simple idea. Consider an evidence, which is composed of n examples. Select a random sample of m examples ($m < n$). Use whatever generation algorithm to obtain different hypotheses for these *m* examples. The best hypothesis would be the one that better behaves for the data which has been reserved (*m* – *n* examples).

Another evaluation criterion is known as 'consilience', introduced by Whewell in 1847 for the evaluation of scientific theories. Informally, as it has been used to date, a model or theory is 'consilient' if it is predictive, explanatory and unifies the evidence. Since all of these criteria are desirable, consilience was informally introduced as a fundamental issue for theory construction and modelling. Consilience has always been alluded in the context of scientific explanation or explanatory induction [Harman 1965] [Hempel 1965] [Ernis 1968]. Moreover, one of the important traits of abduction, seen as the inference to the best explanation, is that the abductive hypothesis (known as assumption) must be the most 'compliant' with the background knowledge. This can also be identified with the notion of 'coherence' [Thagard 1978].

Just another popular (and simple) criterion is to choose the most general hypothesis. This criterion is only useful for some restricted representations, because for positive data only, the most general theory would be "everything is true". On the contrary, if we have (or may have) negative data also, it is quite related with the most falsifiable hypothesis, as vindicated by Popper.

To make an idea of the disparity of some selection criteria, there are many other applications where the most specific theory is preferable. For instance, the first rule used for induction of logic programs was the relative least general generalisation (*rlgg*) introduced in the late 1960s by Reynolds and Plotkin [Plotkin 1970]. For more ambiguity, another term exists for this rule, which is known as the *subset principle* [Wexler 1992], i.e., if two hypotheses cover the data, we select the most specific one. It is curious that this principle was identified as an intensional principle because it solved some cases of 'hyperlearning' or "Plato's problem on the poverty of stimulus", when the number of samples were two small or only positive examples were given. In general, however, the most specific hypothesis is the data itself, which is useless.

Finally, some connections between the former evaluation criteria have been established. Theoretically, the MDL principle is closely related to the Minimum Message Length (MML) principle and Maximum Likelihood Estimators [Case and Smith 1983]. It has also been compared with cross-validation [Kearns et al. 1999] and Bayesian Learning [Gull 1988].

There have been little interest about other quality criteria not related with plausibility (accuracy and coverage). For instance, according to understandability, the most important work is [Sommer 1995b], which recapitulates old criteria and introduces new ones for evaluating the quality, in terms of understandability, of induced theories. Some of them, although rudimentary, are closely related with some of the measures that will be introduced in this work, especially reinforcement and intensionality.

2.3.2 Grammatical Inference and Induction of Functional Programs

A great amount of the ML literature during the last decades is devoted to non-symbolical learning technique such as regression, neural networks, fuzzy systems, and Bayesian trees. Since deduction is not relevant for them, the combination of inference processes is not a problem nor a necessity at the moment.

Therefore, we are interested in the important development of symbolic approaches, initially originated by universal representational languages, such as Turing machines and LISP. Initially, universal Turing machine programs were used from the most theoretical point of view to study induction, serving as an *arbitrary* machine for the works of [Solomonoff 1964], [Kolmogorov 1965] [Levin 1973] [Chaitin 1969][11] and many others. The relation between algorithmic (or Kolmogorov) complexity and induction appeared soon, reformulating the foundations of *statistics*. Despite its enormous importance in any work on induction in the last three decades, only recently these ideas have been used practically *directly* [Schmidhuber et al. 1997] or indirectly in ILP [Muggleton et al. 1992] [Muggleton and Page 1994].

Grammatical inference (see a classical source [Angluin and Smith 1983] or a more recent one [Sakakibara 1997]) began after the seminal paper of Gold in inductive inference [Gold 1967], studying the learnability of different kinds of languages and other theoretical issues. Nowadays, a *second* grammatical inference *stage* is more centred in efficient induction (generally using the *state merging operators* instead of enumeration techniques) of regular or context-free grammars for pattern recognition applications, generally using sequential [Velenturf 1978] or subsequential automata [Oncina et al. 1993].

Apart from Chaitin, LISP was also initially used as a representational language for induction in the works of [Summer 1975] [Biermann 1978] (see [Smith 1984] for a survey), with execution *traces* techniques. The goal of these systems was more concerned with automatic programming than learning. Later, the induction of functional programs has been retaken with modern techniques (evolutionary programming, MDL principle, folding) [Olson 1994], [Olson 1995] also with the aim of program synthesis from examples.

Recently, ILP has been presented as a means to undertake both the learning and the automatic programming views in the same framework, since program synthesis can be understood also in these terms, if the hypothesis must cover exactly all the data. It also unveils novel uses: relational data mining, and the semi-automatic generation of *new scientific theories*.

[11] Nowadays, Chaitin uses to work with *elegant* LISP programs.

2.3.3 Inductive Logic Programming (ILP)

Despite its popularity in the last decade, inductive Logic Programming (ILP) is *just* a framework of learning in first order logic. Seen more pompous, "ILP bridges Machine Learning, Computational Learning Theory, Statistics and Logic Programming" [Muggleton et al. 1995].

Inside the general framework of learning and induction, the recent importance of ILP may be justified by many reasons. First, one of the advantages of ILP is the ability of using background knowledge and the understandability of theories, differing radically with other novel approaches such as fuzzy systems or neural networks. Second, ILP is a more tractable and natural framework for many problems and has all the *hypothesis validation* efficiency of SLD-resolution. Third, due to the logical representational language, it is easier to state formal considerations about the hypotheses, the evidence and their relationship. Fourth, the induction time of some classical problems (list reversal, sorting) has been reduced to the order of seconds[12] instead of minutes [Olson 1994], which makes ILP feasible for new applications. A last reason may be found in the aim to reconvert those people who many years ago entered Logic Programming with AI views, remarkably the Japanese heritage of the fifth generation, being *relegated* in the late eighties to theorem-proving, databases applications and automatic programming.

The problem addressed by ILP can be simply stated as the inference of a theory (a logic program) from facts (or evidence logic theory) using a background logic theory. Evidence can be only positive E^+ or both positive and negative $\{E^+, E^-\}$. Additionally, a background knowledge theory B (another logic program) can be used. Finally, the desired theory or hypothesis H must observe the following properties.

$B \wedge H \models E^+$ (posterior sufficiency)

$B \wedge H \wedge E^- \not\models \Box$ (posterior satisfiability)

Besides, it is assumed that,

$B \wedge E^- \not\models \Box$ (prior satisfiability)

$B \not\models E^+$ (prior necessity)

In a broader sense and under the conditions of perfect learning (no error allowed in the posterior sufficiency property), ILP can be considered just as a special case of logic program synthesis from formal specification. According to [Deville and Kung-Kiu 1994] there are three main methods of program synthesis: *constructive synthesis* where a program is the result of a constructive proof of a conjecture (the specification), *deductive synthesis* including a lot of techniques where some deduction rules make the final program correct with respect to the specification, and, *inductive synthesis*, which is exemplified by ILP.

[12] The computational power of modern workstations has also influenced in these results.

The evidence E^+ and E^- is usually given in an extensional manner (i.e., as facts) but the framework does not exclude intensional (i.e., theories) as evidence. However, most of ILP systems cannot deal with intensional evidence and usually make a pre-processing to generate sample outputs from it or make some kind of saturation to complete it from the background knowledge [Rouveirol 1994]. It is important to note here that if the evidence is given as a theory, it can be regarded as a program-transformation problem. Clearly, there is a strong relation between generalisation-specialisation techniques in ILP and program transformation techniques.

[Shapiro 1981] can be considered the seminal work for ILP. Shapiro's Model Inference System (MIS) is the best representative of the incremental family of ILP systems. In MIS the initial program is empty and the model is refined using a so-called refinement operator. He claimed that there is a most general refinement operator, complete for clausal logic. But, as it is shown in [Niblet 1993], his operator is not complete and no "natural" refinement operator can exist without introducing redundancy or incompleteness. This means that Shapiro's enumeration of logic programs according to the order that is established by his refinement operator is not complete. Despite this fact and the combinatorial explosion for complex problems [Furukawa et al. 1997], most incremental top-down ILP systems are inspired in it. From the current top-down systems, FOIL [Quinlan 1990] [Quinlan and Cameron-Jones 1995] is the most important and efficient one.

The bottom-up solution consists in finding the Most Specific Hypothesis (MSH) according to Plotkin's Relative Least General Generalization (RLGG) [Plotkin 1970] (see [Mizoguchi and Ohwada 1995] for extensions in ILP) which means that the minimal hypothesis (according to set inclusion) is sought. Popular systems of this kind are Duce and Cigol (in propositional logic and first-order logic respectively), which use the so-called refinement operators:

Absorption:

$$\frac{q \leftarrow A \qquad p \leftarrow A, B}{q \leftarrow A \qquad p \leftarrow q, B}$$

Identification:

$$\frac{p \leftarrow A, B \qquad p \leftarrow A, q}{q \leftarrow B \qquad p \leftarrow A, q}$$

Intra-Construction:

$$\frac{\begin{array}{l} p \leftarrow A, B \\ p \leftarrow A, C \end{array}}{q \leftarrow B \qquad p \leftarrow A, q \qquad q \leftarrow C}$$

Inter-Construction:

$$\frac{\begin{array}{l} p \leftarrow A, B \\ q \leftarrow A, C \end{array}}{p \leftarrow r, B \qquad r \leftarrow A \qquad q \leftarrow r, C}$$

Intra-Construction is specially interesting for predicate invention. These operators can be used to modify the configuration of the bottom-up search.

Golem [Muggleton and Feng 1990] also uses a subsumption lattice. To ensure a complete and non-redundant lattice, the rlggs constructed by Golem are forced to have only a tractable number of literals by requiring ij-determinate definite clause theories [Muggleton and Feng 1990]. Golem was the first ILP system used successfully in many applications, some of them traditionally studied with regression techniques.

Recently, the system Progol has been devised [Muggleton 1995] based also on a subsumption lattice over MSH. The power of Progol originates from its new mode operations and some new ad-hoc statistical considerations to balance the length of the hypothesis with its generality in the case of positive data only and to maximise compression in the case of both positive and negative data. According to [Muggleton 1995], the Progol approach is of a more fundamental nature, based on the inversion of model-theory deduction (inverse entailment), than that of inverse theorem proving techniques (inverting resolution) of Duce and Cigol.

Progol has been applied for learning relations on scientific problems, mainly the prediction of mutagenic molecules [Srinivasan et al. 1994] and other molecular chemistry applications. This use for practical applications [Bratko and Dzeroski 1995] has been highlighted as the most relevant feature of ILP.

One of the most remarkable things about ILP is that it has made clear the necessity of inventing new predicates as bias shift operation [Stahl 1995]. Moreover, they have popularised the difference between useful and necessary predicates. An example of useful predicate is "parent(X,Y)" to express the idea of "grandparent(X,Y)". It is not necessary because we can make do with "father(X,Y)" and "mother(X,Y)". An example of necessary predicate is "ancestor(X,Y)" to express the idea of "relative(X,Y)". As it was seen in the first chapter, useful predicates allow the hypothesis to be shorter, but the generalisation is not strictly necessary. The second ones are recursive ones that are strictly necessary to learn a concept.

In our opinion, predicate invention could be better tackled using high-order logic. In this sense, but with different purposes, Lloyd [Bowers et al. 1997] proposed a jump from ILP to Higher-Order Logic, using a new programming language called Escher [Lloyd 1995], based on Church's simple theory of types [Church 1940], with emphasis on constructs that would be useful for induction.

Finally, there are many other non-symbolic techniques for learning in the machine learning community: artificial neural networks, fuzzy approaches, regression... but, as said before, are less interesting for this thesis and can be found in [Mitchell 1997]. For an up-to-date covering of a more philosophical and logical view of induction I recommend Flach's thesis dissertation [Flach 1995a]. For ILP I recommend [Lavrac and Dzeroski 1994][Muggleton and De Raedt 1994][De Raedt 1996] [Nienhuys-Cheng and de Wolf 1997].

2.4 Abduction

Abduction (sometimes wittily called Sherlock Holmes' intelligence [Josephson and Josephson 1994]) is a kind of hypothetical inference process introduced by Sanders Peirce (1839-1914). Peirce asserted [Peirce 1867/1960] that neither deduction nor induction can help us to unveil the internal structure of *meaning* [Yu 1994]. He thought that another kind of reasoning was necessary to account (jointly with induction and deduction) with all the aspects of human reasoning.

Although we will get back on the problem of meaning in subsequent chapters, nowadays, abduction is usually considered as a special kind of induction or, at most, both are seen as different kinds of hypothetical inference, as [Michalski 1987] points out: "*inductive inference was defined as a process of generating descriptions that imply original facts in the context of background knowledge. Such a general definition includes inductive generalisation and abduction as special cases*".

More concretely, it is usually accepted that abduction is a mechanism for completing knowledge about a certain individual (generally inventing a fact to fit with a theory that is given), thus explaining why the given observations were not predicted by the initial knowledge. On the contrary, induction tries to extend knowledge (or to make a new theory) for predicting future observations.

In our view, the difference may be more of nature than of purpose: induction works without constraint (although an auxiliary background theory can be used) whereas abduction tries to find a hypothesis that is 'compliant' with some higher law that constrains how hypotheses can be. In this way, abduction may be seen as induction in a fixed context, closer to Peirce's original postulate [Flach 1996]:

The surprising fact, C, is observed;

But if A were true, C would be a matter of course.

Hence, there is reason to suspect that A is true.

This "matter of course" is usually represented as a background theory or common-sense theory T (known as paradigm in philosophy of science or a constraint bias in inductive learning). Accordingly, abduction can be represented as usually:

$$A \cup T \models C$$

but with the additional condition that A cannot be an anomaly in the context of T and it cannot be an invention either (a fantastic but possible assumption). As any hypothetical inference process, many A's could be found; consequently, some selection criteria must be chosen in order to find the most appropriate one.

In general, a clear distinction between induction and abduction has not been presented to date (see e.g. [Flach and Kakas 1999] for a state of the art). Abduction is then generally seen as the process of making assumptions to explain some facts. It is related to explanatory induction (completion of the data), commonly described as "inference to the best explanation" [Harman 1965]. For instance the fact "the shoes are wet" can be explained because "the grasp was wet" and the latter because "it rained last night". "Explanatory induction" distinguishes from enumerative induction [Ernis 1968] by using some coherence criteria (or metrics [Ng and Mooney 1990]). Given an observation, in the absence of noise, an explanation must give the causes for the whole observation. For instance, if we have seen smoke and fire co-occurring 999 times over 1000 times, we can describe this observation as "P(smoke-fire) = 0.999" and we have a reliable prediction. However, no explanation is given, mainly because there is not any underlying mechanism justifying the co-occurrence nor the anomaly.

Finally, as we said, there is another trait of abduction related with causation, and this has motivated the alternative name 'retroduction' for abduction [Hanson 1958]. Different frameworks for formalising causation soon appeared with the early expert diagnosis systems, exemplified by causal networks, especially Peng and Reggia's *causal abductive network* [Peng and Reggia 1987], along with other probabilistic or possibilistic frameworks such as Pearl's Causal Theories, using a Bayesian belief network [Pearl 1988], [Pearl 1993]. In this context, an explanation that consists of an only cause for all the data is frequently preferable over separate causes for co-occurring phenomena, following Reichenbach's *principle of common cause* [Reichenbach 1956]. We will get on these topics in subsequent chapters.

Many people (Peirce among them) have considered abduction a rather different reasoning mechanism. As a result, ALP (Abductive Logic Programming) [Kakas et al. 1993] has appeared in the area of logic programming as an isolated paradigm, with numerous applications in databases. Impressively, in some reviews (e.g. [Eiter et al. 1997]) no reference appears to ILP or any induction-related paradigm. [Adé and Denecker 1994] proposed unsuccessfully AILP (Abductive Inductive Logic Programming): "*... we argue that both abduction and induction are different, yet related forms of reasoning on incomplete knowledge. They are both forms of hypothetical reasoning and attempt to "complete" the knowledge by proposing additional hypotheses. However, they differ in the sort of hypothesis*". They just see the role of ILP in finding general rules to explain some data and see ALP in finding some data that fills some gaps in proposed theories.

Also, EBL (Explanation-based Learning) and Abductive Explanation Based Learning can be considered more specific cases of abductive reasoning.

To settle momentarily the question, in my opinion, the major difference originates mainly from the different aims and utility, and what is given, which usually turns abduction much easier and constrained than induction. In *abduction* we try to *complete* our knowledge about a certain individual, by generally inventing a (*past*) fact to fit with a theory that is given, while in *induction* we want to extend (make a new theory) our knowledge to predict (*future*) individuals.

For an up-to-date covering of abduction I recommend Aliseda's thesis dissertation [Aliseda-Llera 1997]. For the relation between induction and abduction I recommend [Flach and Kakas 1999].

2.5 Reasoning by Analogy

As it has been said, an analogy is a transference of information from the characteristics of one situation to the characteristics of another situation. Analogy is also a hypothetical inference process that can sometimes provide correct results (such as whales are mammals) or incorrect (such as the ancient view of the stars as wholes in a black sphere which separated the world from fire). In fact, analogy has always been used as a pragmatic tool and also as a source of metaphors, even for aesthetic purposes.

How to distinguish plausible analogies from fantastic ones has also been the main question of analogy, as an augmented version of the main problem given in induction. Recently, a pragmatic use of analogy has been addressed in AI, which has generated an extensive bibliography about learning by analogies [Kling 1971] [Gentner 1983] [Greiner 1988] [Hall 1989] [Derthick 1990] [Winston 1992] [Hofstadter et al. 1995]. Analogy can be seen as two-layer *explanatory* induction (or as [Holland et al. 1989]'s second-order morphisms), following the known fact that some rule or structure [Gentner 1983] is shared between two previously independent facts, so making possible the transfer of rules and methodologies from one case into the other one. This kind of *creative* analogy or *inductive* analogy is known also as *interpretative analogy* [Indurkhya 1991].

There is a close connection between analogy and induction, analogy and abduction, and even analogy and deduction, as it is seen in mathematical practice. As we said in the introduction, an analogical inference can be seen as an inductive step followed by an abductive (or deductive) step.

2.5.1 Case-Based Reasoning

Case-based reasoning appeared to address the problem of everyday abductive explanation [Kass 1986], [Leake and Owens 1986], [Schank 1986], [Leake 1992], [Shank et al. 1994]. Its main feature is its non-constructive character. Instead, it is schema-based and analogy-based, i.e., every explanation is based on a similar case on prior episodes from memory.

The question arises: how can a case-base reasoning system start? It must record always the first episodes (and after a time it will have an intractable amount of samples to compare with) or "*it assumes that the system begins with a fairly extensive set of schemas —a sufficient set to capture the classes of events that are of interest*" [Leake 1995]. This implies accounting only for "stereotyped events" schematised in "frames", "scripts" or "MOPs" [Charniak 1978], [Cullingford 1978], [DeJong 1979], [Lebowitz 1980], [Minsky 1975], [Schank and Abelson 1977], [Schank 1982] [Plaza 1992]. It is supposed that some kind of analogy has to be made between past experiences and new facts. The difference is that the required mappings are used for just one situation and not constituted as a general rule.

Despite the limitation of *pure* case-based reasoning (in our view), one remarkable thing is the clarification of the issue *when to explain*. The case-based model proposed a method for automatically generating appropriate targets to fill in gaps in understanding, based on detection of comprehension failures revealed by *anomalies* that arise during the understanding process. In [Leake 1995] words: the "explanation process is triggered when *anomalies* arise". But "anomalies not only provide guidance about when to explain, but of what to explain as well" [Leake 1995]. In this way, statistical and MDL-based induction is not useful for explanations, because anomalies are not detected until a great number of them force a change of model. The 'best' explanation is based on "probabilities" or "costs" of the assumptions. These 'costs' are usually measured in terms of "Occam's razor" but based on the "number of assumptions" and "structure coherence" [Ng and Mooney 1990] [Thagard 1989] much more than on a simplistic MDL principle. Case-based explanation is strongly influenced by *similarity* to previous-explained episodes and to stereotyped patterns. Some methods [Charniak and Shomony 1994], [Hobbs et al. 1993], [Pearl 1988] assume that the information on the probability or "cost" of each assumption and rule is available to the explainer. Case-based reasoning, in general, assumes that only "coarse-grained" likelihood information is available [Leake 1995].

For more information and technical details about analogy I recommend [Winston 1992]. [Hofstadter et al. 1995] is suited for more philosophical or psychological issues of analogy.

2.6 On the Relation between Inference Processes

There are many open questions about the nature of each inference process, as long as many more technical problems. The things get even more complicated when one tries to understand the relation between different inference processes and to study their combination. The emphasis has almost always been lain in highlighting their differences and seeing them as opposed inference processes rather than complementary.

In principle, it may seem clear that induction must, in some way, depend on deduction, because every hypothesis must be deductively checked with the evidence. However, this should not boldly lead to the view of induction as the inverse of deduction[13]. The following words of Stanley Jevons [Jevons 1874] state clearly this view:

> Induction is, in fact, the inverse operation of deduction, and cannot be conceived to exist without the corresponding operation, so that the question of relative importance cannot arise. Who thinks of asking whether addition or subtraction is the more important process in arithmetic? But at the same time much difference in difficulty may exist between a direct and inverse operation; the integral calculus, for instance, is infinitely more difficult than the differential calculus of which it is the inverse. Similarly, it must be allowed that inductive investigations are of a far higher degree of difficulty and complexity than any questions of deduction; ...

They are “both modern sounding and relevant” [Muggleton 1995] to current techniques. The view is also endorsed by from the logical interpretation of probability, initiated by John Maynard Keynes [Keynes 1921] and Rudolf Carnap [Carnap 1950, 1952], although the main ideas can be found in the work of George Boole.

Carnap acknowledged that probability “*has two distinct, legitimate meanings: that of a degree of confirmation (related to the subjectivists' concept of probability), and that of a relative frequency (as used by frequentists). To make the distinction clear, Carnap uses Probability$_1$ to denote the former, and Probability$_2$ to denote the latter. Carnap's main interest was with Probability$_1$. Unlike the subjectivists, however, Carnap postulates that Probability$_1$ is a purely logical concept, and that values Probability$_1$(h | e) can be correctly determined by a purely logical analysis of h and e.*” [Jaeger 1998], *h* being the hypothesis and *e* the evidence.

Carnap aimed to define $P(h, e, S, N)$, where S is a finite subset of an infinite vocabulary of *unary* relations and N a finite domain of constants[14]. Carnap takes for granted that this will actually be independent of S and reduces the problem to obtain $P(h, e, N)$. Even with these restrictions and assumptions, Carnap concludes, after studying some possibilities, that the best of them “*is not entirely inadequate*” ([Carnap 1950, p. 565]). In Jaeger's words “*Looking back at these works by logicists we can probably*

[13] Except the 'abductivistics', considering abduction as a different category from deduction and induction.

[14] The reason for this restriction is simple, if the language is infinite and we consider non-unary relations it is not always possible to determine all the consequences of a statement, by Gödel's First Incompleteness Result, and this probability is ill-defined.

rightfully say that their original program has failed. A purely logical concept of probability has not proven to be viable. An obvious reason for this is the fact that we have essentially unlimited and arbitrary choices for which language to choose for expressing probabilistic information" [Jaeger 1998]. A purely logical concept of induction has not been possible either as it is shown in [Flach 1995a], who is able to give a logical account for abduction but not for induction. Note that a similar problem was found by Hintikka in his formulation of surface and depth information [Hintikka 1970a] for deduction.

There have been other investigations on the search of a proper view of semantic information (e.g. [Wittgenstein 1922], [Mackay 1959] [Devlin 1992] [Maddox 1993]), but they have never obtained the popularity of probabilistic accounts of information and logic, or other utilitarian views of information ([Howard 1966] [Aisbett and Gibbon 1999]).

In the end, the best that could be drawn from the probabilistic view of logic (or a logical view of probability) is Carnap's Probabilistic Calculus [Bar-Hillel and Carnap 1953], exhaustively developed in [Kemeny 1953], which, in fact, has been highly influential to the view that induction and deduction are inverse processes in terms of information. The main rule of Probabilistic Calculus is, as we reminded in the previous chapter, the following one:

$$\mathrm{p}(P) \leq \mathrm{p}(Q) \text{ if } P \models Q$$

According to it, deduction decreases information and induction increases it. Note, however, that this omniscient view of deduction is incompatible with the notion of inference we have begun this chapter "a process of reasoning by which a person modifies (part of) its beliefs.". Deduction decreases information and, consequently, does not change belief (provided premises are not forgotten). This does not motivate at all a system to perform deductive inference, because, under this paradigm, it is useless.

Nonetheless, the omniscient view of deduction was already rejected by Kant. He was especially concerned to study what kind of knowledge can be provided by both induction and deduction. He distinguished truths that are a priori and a posteriori (empirical) but, more importantly, he recognised that some a priori propositions may be synthetic, in the way they are derived from other a priori propositions[15]. In this sense, synthetic (both a priori and a posteriori) propositions are valuable in the way that they can amplify our knowledge (i.e., informative). However, Kant was flustered by the following question: How is it possible to establish propositions that are universally valid and, at the same time, amplify our knowledge? Or in its more modern formulation, how deduction can provide some new knowledge? By that time, the idea of computational cost or resource consumption was not even augured,

[15] Even for the empiricists (like Quine), for which there cannot be an absolute distinction between the analytic and the synthetic, Kant represents the recognition of inference as a mean of synthesis of new concepts, not explicitly given before the inference process (or processes) has taken place.

so Kant's solution was found in what he called *Transcendental Aesthetics*. Curiously, this transcendental aesthetics was based on the notions of space and time, although from a much more existential point of view from the notions of space (bits) and time (computation steps) that will be used in this work.

2.6.1 Inference Processes, Effort and Lazy/Eager Methods

The Ancient Greeks were not interested in the effort that any inference requires. Their passion for philosophical and scientific meditation did not motivate an interest to reduce this effort of reasoning, which, for the Greek Culture, was even pleasant. It is only with the advent of the notion of machine and its applications to artificial intelligence and other many different practical problems, that effort and cost turn out to be relevant.

I have been discussing the differences between inference processes. When we consider information and effort, some similarities begin to shine: first, every inference is usually guided by an interest to obtain a new assertion or new knowledge, not *explicitly* present previously and, secondly, the result of an inference process must be evaluated in order to discern if the result is valuable enough to be preserved or discarded (forgotten), according to its interest and the effort which has been performed to obtain it.

This engages with the field of resource-bounded reasoning, where this effort is beginning to be weighed with the value of the expected results: "*Instead of building systems that find a 'good' answer, the goal of resource-bounded reasoning techniques is to find an 'optimal' answer. Optimality, however, is defined with respect to the system knowledge and computational capabilities*" [Zilberstein 1999]. For such a measure of optimality it is necessary to evaluate the effort in computational capabilities from what was known before an inference process and what is known after it. In other words, a resource-dependent *information gain* measure is required.

The case of analogy also suggests another way of classifying inference processes. Some inference processes work *on the fly*, i.e. they are *lazy* [Aha 1997], in the way they are only used when needed, such as analogy or abduction, and other inference processes are more *eager*, in the way they try to obtain concepts or rules that would be necessary in the future, as constructive induction performs.

More concretely, eager learning extracts all the regularity from the data in order to work with intensional knowledge, i.e., a model. Examples of eager learning are Model Based Reasoning (MBR) and Inductive Logic Programming (ILP). The MDL principle gives a theory that is usually eager for compressible data and lazy for uncompressible data. I will investigate in this thesis more eager criteria, in order to anticipate or 'invest' in more complex (or intensional) theories.

There has also been a very important and fruitful research in *lazy* learning methods [Aha 1997]. Examples of lazy learning methods are *k*-neighbouring or distance-based techniques, case-based reasoning (CBR) or instance-based reasoning and analogical reasoning. As we have seen in some of these methods, examples are *memorised* as extensional knowledge with some information about their results and other characteristics. In the moment of a query or a new problem, the system works hard to extract which previous experiences are more appropriate to the new problem, by selecting the most similar cases or by making the most plausible analogy.

A quite updated comparison of lazy and eager (also called inductive in this paper) methods can be found in [López de Mántaras and Armengol 1998]. Some works [DeJong and Mooney 1986], [Mooney 1990] have introduced flexible frameworks to combine EBR and induction, known as "explanation-base schema acquisition" (EBSA). The difference in laziness of CBR and EBSA is illustrated by [Leake 1995]:

> A key difference between case-base explanation and explanation-based schema acquisition concerns the preferred level of generalization. Explanation-based schema acquisition assumes a sufficiently high-quality domain theory to allow immediate generalization of new episodes whenever licensed by the rules of the domain theory. Whenever explanation-based schema acquisition systems encounter new situations that do not fit previous generalizations, they first explain the situation by doing backwards chaining, using their library of previous rules and schemas. After completing an explanation, they immediately perform explanation-based generalization of the explanatory chain to form a new generalized schema for future use. Case-based explanation instead takes a very conservative approach to generalization. At the time an explanation is generated, case-based explanation simply stores *that specific explanation*. If that explanation must be generalized to apply to another situation, the generalization is done only at the time that the explanation must be re-applied, and only to the extent required to explain the new situation.

Despite this continuous repetition of things that have been done several times before, the advantages of lazy methods are their flexibility and the economy of resources in the short and medium terms, because a reasoning effort is only done when a new problem appears. Another important advantage is that revision is unnecessary, because no model of reality is constructed.

In contrast, the advantages of eager methods are given by the fact that they can be constantly pre-processing all the received information and they can profit idle time resources. If the model is accurate, the answer to a new problem is immediate. Moreover, most of the given examples can be forgotten when their model is reliable enough, reducing storage and increasing manageability in the large.

This distinction, however, has not been clearly stated in the literature for deduction. Nonetheless, deduction can also be sometimes lazy, such as everyday deductive inferences, and sometimes eager, such as mathematical practice. Only program specialisation and transformation techniques [Pettorossi and Proietti 1990, 1996a, 1996b] [Dershowitz and Reddy 1992] deal with the transformation of deductive systems or programs into more efficient ones, preparing the program representation to the expected facts it should cover, something that could be seen as eager or anticipative deduction.

The reaction time of both inductive and deductive inference is crucial in action systems, and the quotient between reaction time and quality of response is the main point, highlighted by [Horvitz 1990]. Since then, there has been an increasing interest in the context of resource-bounded systems, and it has been shown that the question not only depends on the expectation of kind of problems (problem type prediction) but also on the preparation and representation of the background knowledge, i.e. the theory, from which quick inferences must be generated. According to this expectation (or past problems) a measure of *representational optimality* could be quite useful.

The choice between lazy and eager inference methods clearly depends on time and space resources but also on the frequency of use. For instance, almost everyone of us reminds explicitly how old we are but usually do not store explicitly how many years have passed since we finished secondary school. Hence, some time (and effort) must be employed to derive that information when it is needed. In other words, an *oblivion criterion* is required to discern which things should be maintained explicitly.

Finally, complexity connections between different inference processes have also been established in terms of computation, which make even clearer that deduction and induction are not inverse processes. It is little surprising then the recent result that "*Some Learning Systems are Interactive Proof Systems*" [Sempere 1998].

2.6.2 Inference Processes and Confirmation

Another common trait of any inference process is that an inference can be confirmed or refuted. Even in the case of non-hypothetical inference, i.e. classical deduction, it is completely different to state "B is a logical consequence from A" that to state "B is a logical consequence from A due to proof C". This also highlights that a theorem prover provides useful information, because a proof does supply new knowledge, meta-knowledge about the certainty of other pieces of knowledge. In fact, mathematics is full of conjectures, which may or may not be confirmed or refuted. But even in the case of computational deduction we must admit some possibility of error, and, consequently, additional confirmations are useful. Obviously, for hypothetical inference, the role of confirmation is more blatant because evaluation criteria are usually not sufficient to select the 'right' hypothesis with certainty.

The consideration of confirmation propagation motivates a refinement in terminology, perfectly illustrated by these words from [Li and Vitányi 1997]:

> The Oxford English Dictionary defines induction as "the process of inferring a general law or principle from the observations of particular instances". This defines precisely what we would like to call *inductive inference*. On the other hand, we regard *inductive reasoning* as a more general concept than inductive inference, namely, as a process of reassigning a probability (or credibility) to a law or proposition from the observation of particular instances.
>
> In other words, inductive inference draws conclusions that *accept or reject* a proposition, possibly without total justification, while inductive reasoning only changes the degree of our belief in a proposition. We need also to distinguish inductive reasoning from *deductive reasoning* (or *inference*). In deductive reasoning one derives the absolute truth or falsehood of a proposition. This may be viewed as a borderline case of inductive reasoning.

It is somehow startling for the author of this work to note that the solution for this confirmation propagation, which will be given in chapter 5, accounts both for deductive and inductive confirmation by regarding deductive confirmation as a limit (or borderline) of inductive confirmation. Moreover when [Li and Vitányi 1997] are strenuous upholders of the MDL principle.

After this regard to terminology, let us review two different solutions for the confirmation problem. Two philosophers and logicians from the Wiener Kreis addressed the problem: Carnap and Hempel. A quantitative concept of degree of confirmation, as a value between 0 and 1 for a hypothesis given an evidence, was developed by Carnap, who associated it, as we have said, with a notion of probability. On the contrary, Hempel introduced a qualitative concept of confirmation, i.e., a Boolean relation between hypothesis and evidence, in the way that E confirms H or E does not confirm H. In order to devise such a *logical* relation of confirmation, Hempel introduced five adequacy conditions [Hempel 1943, 1945]. Let us recall them (from [Flach 1995a]):

(H1) *Entailment condition*: any sentence which is entailed by an observation report is confirmed by it

(H1.1) Any observation report is confirmed by itself.

(H2) *Consequence condition*: if an observation report confirms every one of a class K of sentences, then it also confirms any sentence which is a logical consequence of K.

(H2.1) *Special consequence condition*: if an observation report confirms a hypothesis H, then it also confirms every consequence of H.

(H2.2) *Equivalence condition*: if an observation report confirms a hypothesis H, then it also confirms every hypothesis which is logically equivalent with H.

(H2.3) *Conjunction condition*: if an observation report confirms each of two hypotheses, then it also confirms their conjunction.

(H3) *Consistency condition*: every logically consistent observation report is logically compatible with the class of all the hypotheses which it confirms.

(H3.1) Unless an observation report is self-contradictory, it does not confirm any hypothesis with which it is not logically compatible.

(H3.2) Unless an observation report is self-contradictory, it does not confirm any hypotheses which contradict each other.

(H4) *Equivalent condition for observations*: if an observation report *B* confirms a hypothesis *H*, then any observation report logically equivalent with *B* also confirms *H*.

(H5) *Converse consequence condition*: if an observation report confirms a hypothesis *H*, then it also confirms every formula logically entailing *H*.

Loosely, H1 and H2 can be identified as deductive (downward) confirmations, H3 is an inductive confirmation in Popper's sense (the theory has not still been refuted by the evidence), and H5 is an abductive (upward) confirmation. H4 is the most natural and doubtless one, at least if modalities are not taken into account. However, H2 and H2.1 turn out to be inconsistent with H5, a problem known as the "confirmation paradox" [Hempel 1943, 1945] [Hesse 1974]. His solution is to drop one of the two conditions, but, as Flach points out, "*Hempel solves the problem on the formal level by dropping the converse consequence condition in favour of the consequence condition. However, on the intuitive level the paradox remains, since Hempel does not provide a clear justification of his choice*" [Flach 1995a]. Moreover, H2.2 generates some problems with general formulae that Hempel tries to solve through a narrower relation of direct confirmation, which is somehow closely related to the subset principle, i.e., if two hypotheses cover the data, choose the most specific one. Flach's qualitative solution is much more convincing; he separates two subsets of adequacy conditions which account separately for explanatory (abductive) and confirmatory (descriptive) reasoning, so enlightening some classical distinctions between inductive and abductive reasoning. However, in my opinion, the original view of confirmation is not represented by the second choice alone, nor by the first one. The problem is that a qualitative account of confirmation cannot conciliate H2 with H5, i.e., downward or forward (deductive) confirmation with upward or backward (abductive) confirmation, because both have different strength.

One claim of this thesis, which is especially defended by the results of chapter 5, is that it is possible to weigh consistently both sources of confirmation, although it cannot be done with a measure of probability, in a strict sense, but a measure of plausibility, which does not comply with Carnap's Probabilistic Calculus. Another reason for this is to avoid the non-informativeness problem of probabilistic approaches of confirmation, as pointed out by Popper: "*Those who identify confirmation with probability must believe that a high degree of probability is desirable. They implicitly accept the rule: 'Always choose the most probable hypothesis!' Now it can be easily shown that this rule is equivalent to the following rule: 'Always choose the hypothesis which goes as little beyond the evidence* *as possible!*" ([Popper, 1963, pp. 289-90]), or maybe, just take the evidence itself as a complete extensional hypothesis. Carnap, on the contrary, obviates this problem by separating the problem of probability from that of interestingness: "*Inductive logic alone does not and cannot determine the best hypothesis on a given evidence... This preference is determined by factors of many different kinds...*" ([Carnap 1950, p.221], from [Flach 1995a], p. 30). Maybe an *intensionality degree* or *an information gain measure* could be in Carnap's mind.

More precisely, the approach for a theory of confirmation that will be undertaken in the chapter 5 of this dissertation is based on a gradual (non-Boolean) propagation of confirmation, a solution in between Hempel's and Carnap's, which allows to include both H2 and H5, a theory which is also between the MDL principle and Popper's informativeness criterion. This measure of reinforcement will be shown to be useful for deduction, induction and abduction.

2.6.3 Towards a Combination of Inference Processes

The research in artificial intelligence has usually studied inference processes in a separate way. Although abduction has sometimes been seen in conjunction with deduction in nonmonotonic models of reasoning or probabilistic logics, induction, as the way to generate theories from facts or learn in an automated way, has usually been a separate thing, addressed by the machine learning community.

Apart from Popper and Miller's view that every inductive support is deductive (quite reasonable if we assume that for all a and b, $a \rightarrow b \equiv \neg b \rightarrow a$), there have even been some essays to see induction as deduction, with the illusion that all the problems of combination would be solved, because there would only be a unique inference process. [Shanahan 1989] studied the use of deduction for prediction and abduction for explanation and [Gregoire and Saïs 1996] claimed that inductive reasoning is sometimes deductive. In my opinion, these results are obtained by misconceptions or a different understanding of some of the inference methods involved, which, in any case, do not solve the main problem of their combination.

In the last decade, the first successes of ML have motivated the punctual use of ML techniques for other problems of deductive character, such as software engineering and automated deduction [Langley and Simons 1995]. But only recently, agent theory addressed the problem of reasoning with combined inference processes, at least in an informal way. There is again some interest about the association of different inference processes in order to make more intelligent systems. In fact, it has been realised that many different processes of learning or knowledge acquisitions can be explained as suitable combinations of basic inference processes: induction, abduction and deduction. In [Michalski 1993], different combinations and variants of hypothetical reasoning are given in the following table:

Variant	INPUT	Background Knowledge	OUTPUT
Empirical Inductive Generalisation	Dawski's paintings, "A girl's face" and "Lvov's cathedral" are beautiful		Maybe all Dawski's paintings are beautiful
Constructive Inductive Generalisation **(generalisation + deductive derivation)**	Dawski's paintings, "A girl's face" and "Lvov's cathedral" are beautiful	Beautiful paintings tend to be expensive (and opposite)	Maybe all Dawski's paintings are expensive.
Inductive **Specialisation**	John lives in Virginia	Fairfax is a "subset" of Virginia (Living in x implies living in superset of x)	Maybe John lives in Fairfax
Concretion	John is going to New York	John likes driving ("Driving to" is a special case to "going to") Liking to drive m-implies driving to places	Maybe John is driving to New York
Abduction	There is smoke in the house	Smoke usually indicates fire (and conversely)	Maybe there is fire in the house
Constructive abductive generalisation **(generalisation + abduction)**	Smoke is in John's apartment	Smoke usually indicates fire (and conversely) John's apt. is in the Golden Key building	Maybe there is fire in the Golden Key building

Table 2.1. Different combinations and variants of hypothetical reasoning.

The purpose of the preceding table is to illustrate how classical inference processes can be 'camouflaged' under different names. Moreover, it shows that the view of deduction as specialisation and induction as abstraction or generalisation is completely erroneous; abduction can generalise or specialise and induction can also generalise or specialise.

The question is then, if these complex or derived inference processes are composed of simpler or basic inference processes, does this mean that plausibility, informativeness and confirmation must be assigned as a sum of its parts? And, if this is the case, is it possible to combine plausibility criteria of induction with plausibility criteria of nonmonotonic deductive inference? Does have the same meaning informativeness for abduction, induction and deduction? The things seem even more complicated if we intend to measure the value, novelty, or internal utility of these inferences, depending of a background knowledge that has been constructed as well from varied inference processes. In other words, which results are to be maintained explicitly?

There have been, of course, some approaches to solve this problem. The most successful ones, although in limited domains, are based on combinations of reinforcement learning and effort or resource optimisations [Barto et al. 1995] [Schmidhuber et al. 1997a, 1997b] [Martin 1998].

The first symbolic approach in this direction is SOAR [Newell 1990], but it is based on a single learning method called chunking. The main feature of SOAR but it highlights when and why a reasoning process should be triggered. The system THEO [Mitchell et al. 1991] is also a self-improving system that integrates more learning methods.

The system Noos [Arcos and Plaza 1996] combines problem-solving (deductive) techniques with multiple learning methods [Plaza and Arcos 1993] (induction and CBR [Armengol and Plaza 1994]). The major feature of this latter system is that includes metalevel capabilities and reflection about the learning strategies and how the goals have been achieved.

However, there has not been presented to date a general theory that would account for this combination in general. A partial effort in this line [Aisbet and Gibbon 1998, 1999] is based on the use of utility to account for information in a logical framework, or the MOBAL system [Morik et al. 1993][Sommer et al. 1995], which restructures a theory according to different criteria while retaining the set of computed answers.

In the end, there is a need for evaluation criteria that solve the main characteristics of inference, as we saw in the introduction: information, novelty, belief, explicitness and confirmation. In the end, as an outcome of this problem of inference combination, a necessity arises: new unified and coherent evaluation measures.

3. Information and Representation Gains

Al funesto aforismo de que "el saber no ocupa lugar" —lo cual, en rigor, es falso– opongo siempre este otro: "pero el aprender ocupa tiempo" y mientras se aprende una cosa podría aprenderse otra de más sustancia.

Miguel de Unamuno, 1865-1937, Sobre la enseñanza del clasicismo.

Abstract: *this chapter introduces a theoretical measure for evaluating the amount of information that has been made explicit by the effort of a reasoning step. The properties such a measure should observe are discussed. Initially, a measure of time-ignoring information gain* $V(x|y)$ *is essayed, which represents the degree of information of* x *which is implicitly in* y*. However, it does not take into account time, and, consequently, it does not grasp the idea of effort. For non-omniscient systems, where the notion of effort makes sense, the intuitive notion of information is re-understood in terms of resource consumption. The choice of the function LT, which weighs space and time, as an appropriate measure of effort, neglects the idea of effort exclusively based on time or space. A new effective measure of computational information gain* $G(x|y)$*, which depends on the computational effort (time and space), measures the proportion of* x *which can be easily obtained on the help of* y*. Some of its properties are studied, and it is compared with different informal but outstanding notions: implicitness vs. explicitness and some questions about aesthetics and interestingness. Finally, some definitions for whole systems are introduced, such as Representation Gain, a general notion of Simplification and a Representational Optimality criterion.*

Keywords: Reasoning, Information, Bounded Rationality, Computational Resources, Information Gain, Transformation Gain, Explicit vs. Implicit, Interestingness.

3.1 Introduction

Reasoning can be characterised as a kind of computational process that transforms information. As we said in the preceding chapter, for many centuries, the attention was focused on characterising reasoning in terms of *truth*. Long before the modern notions of computation and information were developed, philosophers strove for understanding and formalising logic as a truth-preserving deductive process. Later on, induction was distinguished as a probabilistic or non-truth-preserving process where knowledge could be amplified from.

Nonetheless, it has been patently clear in artificial intelligence that truth considerations are not sufficient for characterising reasoning. Infinite many different extensions of logic have been introduced accordingly: non-monotonic, modal, multi-valued, fuzzy, default, ... with more or less outstanding theoretical results and applications in different areas. However, there is a feature of formulae that is as important as truth. It is their *value* or *utility*. Tautologies are true formulae, but most of them are useless. This suggested the classical conception of information related to the number of excluded worlds, so leading to the assignment $I(x) = 2^{-P(x)}$. This entails that any two different representations of the same concept have always the same information and probability. For instance, "log (x · y) = log x + log y" has the same information as "x+x = 2x", and, in this world and at this moment, "the capital of France is Paris" has the same information as "Paris is Paris". However, their informativeness is quite different. Apart from their significance in the philosophy of language and the study of meaning, it is a general issue that pervades reasoning. Modalities, quotes, partial derivations/evaluations have been exhaustively studied in the literature, especially in automated deduction, but only utility criteria have given auspicious results. Despite relative utility criteria, there are absolute criteria that affect the value of a concept. These absolute criteria are centred in the degree of *intension* of a concept. We will get back on this question on chapter 6. In this chapter and the following one we will study reasoning dynamically, as a computational process that transforms information into more convenient representations.

Recently, as it has also been commented, there has been an increasing interest in regarding reasoning as a resource-bounded process ([Simon 1982], see [Zilberstein 1996] for a survey) as a reaction [Moreno 1998] to the wide use of Kripke's semantics [Kripke 1963] of possible worlds for formalising rational agents. Reasoning involves effort, which can be computationally expressed as resource consumption, especially space and time. Accordingly, cognitive systems organise their space resources (i.e. their memory) in order to minimise this effort in the future. Since memory resources are finite, intelligent cognitive systems usually memorise the information they receive in a selective and intensional way. Some information that was already implicitly or explicitly present in the system's knowledge is discarded whereas new and interesting information is included in a convenient way.

If a system is omniscient, any implicit information can be made explicit without effort. In this case, it is easy to determine what is 'new' information: any piece of data such that is not covered or subsumed by previous knowledge, i.e. it is independent to it. Thus, the knowledge of an omniscient system increases as new and independent information is being added. Under this classical view, the knowledge of a system can only change from the interaction with a world or reality. A paradoxical and usually neglected consequence of this is that an omniscient system does not need to think, because only external perceptions have influence on knowledge. In this way, information can be seen as energy; a close system cannot increase its energy/information.

On the contrary, if a system is not omniscient, reasoning is mainly devoted to make explicit what was implicit[16], which includes the connection of different parts of knowledge, the detection of redundancies, the construction of plans and its consequences, the imagination of what-if, etc. Moreover, if it is neither omniscient nor completely consistent, it may detect inconsistencies and improve the robustness and ontology of knowledge. In other words, non-omniscient systems have many more functionalities to study on, much more dynamics, and, mainly, they are more realistic.

As a result, a non-omniscient system has *another* kind of uncertainty. This is precisely what motivated Hintikka's difference between shallow and deep information: "*The alleviation of this kind of uncertainty must be reflected by any realistic measure of information that we have effectively available (insofar as distinct from the information that in some way we have potentially available)*" [Hintikka 1973]. Therefore, a reasoning step from *y* to *x* will be more valuable as more implicit information of *y* is made explicit. In other words, given a *y* and an *x* we want to measure how much information of *x* is explicitly in *y*. If all the information of *x* is already explicitly in *y*, then *x* is not much valuable with respect to *y*, i.e. *x* is obvious from *y* and there is no significant reduction of uncertainty. On the contrary, if most of *x* is not explicitly or implicitly in *y* or it is hard to make it explicit, then there is an information gain of *x* with respect to *y*.

The goal is then a measure that evaluates the information gain that any reasoning process can obtain from a given knowledge *y* to the result *x*. Among theses processes we have:

a) Problem Solving: *y* as a problem and *x* is one solution.
b) Consistence Checking: *y* can be inconsistent and *x* makes it manifest.
c) Deductive Inference: *x* a deduction from *y*.

[16] Although Kirsh gives this role to computation: "computation is a process of making explicit, information that was implicit" (Kirsh 1990).

d) Inductive Inference: *x* an induction from *y*.
e) Abductive Inference: *x* an abduction from *y*.
f) Analogical Inference: *x* is analogous to *y* (*or x shows the analogy between* y_1 *and* y_2*, with* $y = < y_1, y_2 >$).
g) Representational Transformation: *x* and *y* are alternative representations for *z*.
h) Conceptual Simplifications: *x* is a simplification of *y*.
i) Conceptual Optimisation: *x* is a better representation for *z* than *y*.

It is clear that process a) can be conveniently defined to subsume processes b), c) and d), and g) subsumes processes h) and i). In this chapter we will centre on processes a) and e) in a generic way, in order to introduce different measures of information gain. The next chapter will be devoted to study inference processes b) c) d) and e) f) in a detailed way.

As we have said, a function $F(x, y)$ of information gain from *y* to *x* must evaluate how informative is *x* with respect to *y*. This measure should conform with the following fundamental properties:

1) $F(x, y)$ should be smaller as long as x is more obvious from y and it should be greater as long as x is more difficult to obtain from y.

2) $F(x, y) = min_{\forall x,w}(F(x, w))$ iff $x = y$. In other words, no transformation, minimum gain.

3) $F(x, y) = max_{\forall v,w}(F(v, w))$ iff y is useless for x.

The last property can be understood in two ways: *y* can be useless for obtaining *x* because *x* and *y* are absolutely independent, or, *y* has common information with *x* but it is useless because it is extremely intricate or difficult to discover. Since, as we will see, it is not computable to know whether two objects are absolutely independent, we will consider the last interpretation.

In what follows we will introduce different measures and we will study their properties. We will dub "information values" those measurements which can be negative and positive, comparing the effort from *y* to *x* with the effort from *x* to *y*. In the cases studied, they all result to be inconsistent with most of the preceding properties. Their introduction is justified because they are useful to understand the measures that are presented later on.

Finally, we will establish some technical properties in order to make the function measure the proportion and degree (0...1) of the information of *x* which is explicitly in *y*:

4) $0 \leq F(x, y)$

5) $F(x, y) \leq 1$

We will only use the name of Information Gain for any measure for which properties 1, 2, 3 hold and Normalised Information Gain if it is also compliant with properties 4 and 5.

Hintikka claimed that “a measure of information that would not be effectively computable is almost absurd” [Hintikka 1973]. According to this, a last property will be taken into account:

6) F must be computable.

This last property depends on how the effort from implicit information to explicit information is measured. As we will see, F will only be computable if time is considered as a factor of the effort.

3.2 Resource Consumption and Gain

The difference between explicit and implicit can only be made if reasoning entails an effort or difficulty (if not, everything implicit would be explicit). Any effort must be measured according to some resource consumption (time, energy, external additional data, accuracy loss, ...). Consequently, depending on which measures of resources were chosen, different measures of information gain could be obtained.

From a computational point of view there are two main resources to be considered: space and time. Without loss of generality, given a system ϕ, any piece of information x which represents an object or a fact from the world can be coded as a binary string. However, if ϕ is a universal descriptional system, there are infinite many representations for x in ϕ, i.e., different strings d such that $\phi(d) = x$, that we denote by $\{d_x\}$. In order to measure the resource-optimality of each d_x it is sufficient to define a resource-function over space and time, i.e., a function over the length of the description, denoted by $l(d_x)$, the additional space which is required to go from d_x to x, denoted by $Space(d_x)$, and the time cost, denoted by $Cost(d_x)$. In this way, if we define the resource-function $R = F(\text{l, Space, Cost})$, the best R-representations of x can be generically obtained in the following way:

$$\text{Opt}(x) = \text{argmin } \{ \text{R}(d_x) \mid \phi(d_x) = x \}$$

Note that $Opt(x)$ is a set since there can be more than one representation that makes $R(\cdot)$ minimal. Finally, the resource-complexity can be defined as $\text{RC}(x) = \min \{ R(d_x) \mid \phi(d_x)=x \}$.

If we simply define R as $l(d_x)$, we have that $\text{RC}(x) = \text{K}(x)$, i.e., the Kolmogorov Complexity of x. This measurement of resource as space of the description ignores time. As a result, it only recognises the implicit information that cannot be made explicit unless some extra information is added. It does not recognise that something can be implicit because it requires time to make it explicit. Some unintuitive consequences are derived. For instance, an encrypted document would be explicit information, because there is a very short program (try all the combinations for the key) that would finally decrypt the document to make it ‘explicit’).

On the other side, if we define R as $Cost(d_x)$, we would have that $\mathrm{RC}(x) = l(x)$ if x is finite and $\mathrm{RC}(x) = \infty$ if x is infinite. If we measure the information gain under this resource we would have even more unintuitive consequences, because any extra information is not included in the resource function and the information gain from y to x would always be the time it takes to print y, because y could always be given without cost.

As we will see, the compromise is found by measuring both time and space. In this case, it is possible that some implicit information cannot be made explicit unless some extra information is added. Moreover, it recognises that something can be implicit because it requires time but no extra information to make it explicit. *The* question is how to weigh space and time. In 1973 Levin showed that the weighing $R(d_x) = l(d_x) + \log Cost(d_x)$ can be used to construct a "universal optimal search algorithm" as an enumeration algorithm ordered by $R(d_x)$ that were optimal (up to a multiplicative constant factor) for any inversion problem (given a y and a function f, obtain x such that $f(x) = y$). This function R, which is usually denoted by LT, produces the well-known Kt complexity.

The rationale for LT is simple: suppose the solution of a problem has size n and to check the solution is made by an oracle. In order to obtain the correct solution we can provide the n bits of the solution extensionally or we can essay all the possible 2^n combinations without providing any additional information to guess the correct answer. The first method has the cost of n bits of additional information whereas the second method has the cost of 2^n computations (and questions to the oracle). By using LT both methods have the same cost: n. Normally, the optimal way from the problem to the solution is given by a balance between both methods: some information (hints) is given extensionally and some other information is computed. As Kirsh points out: "*it seems that there is a principled difference between space and time. But we have learned otherwise. Accordingly, just taking computational effort as the measure of explicitness, there is no way of choosing whether to represent a given block of information by a powerful procedure plus limited data or by a weak procedure plus exhaustive data*" [Kirsh 1990].

For the purposes outlined at the beginning, it provides a good compromise between space and time[17]. However, different parameterised factors could be added according to the system's characteristics. After these notions of 'effort' it would seem easy to define a notion of 'value' increase or 'gain' from an object to another. However, an effort can be done in vain, so it may entail no information gain. We will see how to avoid partially or completely this 'vain effort' phenomenon.

Sections 3.3 and 3.4 investigate the possibilities of $K(x)$ for defining an information gain, inspired in well known derived notions of $K(x)$, as mutual

[17] Using the logarithm of the cost instead of the cost or the product of l(p) * cost(p) allows the consideration of short programs that are NP-hard (or exponential), that otherwise would be replaced by the program "PRINT x", because it would have less complexity.

information and information distance. Section 3.5 introduces Computational Information Gain based on $Kt(x)$, which will be shown to comply with the properties sketched in the previous section. Section 3.6 relates it with some classical concepts of computational complexity, showing the robustness of the definition. Section 3.7 introduces a variant that it is theoretically free from the addition of new and unrelated information (vain effort phenomenon) in order to inflate information gain. Section 3.8 presents a measure that compares 3 objects instead of only 2 to formalise the idea of representational transformation. This special case is dubbed representational gain. Finally, section 3.9 compares information gain and other measurements that are introduced in this chapter with existing related notions appeared in the literature.

3.3 Relative Information Value

Kolmogorov Complexity ($K(x)$) is an objective measure of the absolute amount of information of an object, up to a fixed constant, which depends exclusively on the descriptional system that is used. Conditional Kolmogorov Complexity ($K(x|y)$) is an objective measure of the relative amount of information of an object x with respec to an object y.

One of the most significant properties of $K(\cdot|\cdot)$ is that, in general, $K(x \mid y) \neq K(y \mid x)$, known as the asymmetry of Kolmogorov Complexity. It is straightforward then to understand $K(x \mid y)$ as the data 'cost' or 'effort' necessary for going from y to x. In the same way, the time-weighted variant $Kt(x \mid y)$ can be viewed as the data and time 'cost' or 'effort' necessary for going from y to x. Moreover, it is easy to adapt these functions to follow two of the three properties of a metric (at least asymptotically), namely, (1) $F(x, y) = 0$ iff $x=y$, because $K(x \mid y)$ and $Kt(x \mid y)$ are always strictly greater than 0 and (2) $F(x, y) + F(y, z) \geq F(x, z)$. As we have just commented on, they do not follow the third property of a metric, symmetry.

In fact, this has given many problems for the definition of a proper measure of universal distance [Bennett et al. 1998] between objects because "*The conditional complexity $K(y|x)$ itself is unsuitable* [...]. *$K(\varepsilon|x)$, where ε is the empty string, is small for all x, yet intuitively a long random string x is not close to the empty string*". Nonetheless, it is precisely this asymmetry which makes information transformation *worthy* and allows the possibility of measuring information gain.

The first idea of information value can be based on measuring the cost or effort in one way with respect to the cost or effort in the other way. For instance, if x is a program for y, it is usually more valuable to have x than to have y, because, intuitively, if one has x then one has y. The idea is to compare the way from y to x with the way back from x to y. This leads exactly to the definition of relative information value:

Definition 3.1 The relative information value of x with respect to y, denoted $W'(x \mid y)$ is defined as:

$$W'(x \mid y) = K(x \mid y) - K(y \mid x) \qquad \text{[Zurek 1989a]}$$

This measure W' was introduced by Zurek as the thermodynamic cost of computation. Some of its properties are studied in [Zurek 1989a].

For our purposes, it is easy to check that $W'(x \mid y)$ does not follow any of the properties of the introduction. For instance, $W'(x \mid y)$ would be negative if y is more valuable than x and positive if x is more valuable than y.

However, it is interesting to study some other properties in order to know where the definition can be modified to make the way towards better definitions such as those introduced in sections 3.4 and 3.5. The first reader can perfectly step to these sections directly.

3.3.1 Properties

Theorem 3.1 If x and y are independent, i.e., the common information $I(x{:}y) \overset{+}{=} 0$, we have that $W'(x \mid y) \overset{+}{=} K(x) - K(y)$, so the value of x with respect to y only depends separately on the minimal lengths of objects x and y.

PROOF. By the following property $I_c(x : y) \overset{+}{=} I_c(y : x) \overset{+}{=} I(x{:}y)$, the result is straightforward. If x and y are independent, we have that the contained information of y with respect to x is 0, i.e. $I_c(\mathrm{x} : y) = K(y) - K(y \mid x) \overset{+}{=} 0$ and the contained information of x with respect to y is 0, i.e., $I_c(y : x) = K(y) - K(y \mid x) \overset{+}{=} 0$. Hence $K(y) \overset{+}{=} K(y \mid x)$ and $K(x) \overset{+}{=} K(x \mid y)$ resulting in $W'(x \mid y) \overset{+}{=} K(x) - K(y)$. □

Our main concern, however, is precisely centred when x and y have a close relation as problem-solution, premise-conclusion, evidence-hypothesis, etc. For instance, the closest relation is given by the following theorem, when x is a program for y.

Theorem 3.2 If x is a program for y we have that $W'(x \mid y) \overset{+}{=} K(x \mid y)$.

PROOF. Since x is a program for y we can construct the program p = "`Execute the input`", which is of constant size. Hence, $K(y \mid x) = l(p)$. From here, $W'(x \mid y) = K(x \mid y) - l(p) \overset{+}{=} K(x \mid y)$. □

Consequently, if x is a program for y, the information value is reduced to the measurement of the relative information of x with respect to y. This suggests the use of some properties of relative information and their extension to information value. For instance, obtaining the first shortest program for a given y is not much valuable:

Theorem 3.3 If $x=y^*$, i.e., the first minimal program for y, we have that:

$$W'(x \mid y) \overset{+}{<} log\ l(y) + 2\ log\ log\ l(y)$$

PROOF. From Theorem 3.2, since x is a program for y, $W'(x \mid y) = W'(y^* \mid y) \overset{+}{=} K(y^* \mid y)$. Additionally, for all x, $K(x^* \mid x) \overset{+}{=} K(K(x) \mid x)$. The > sense is obvious and the other sense is explained by the following construction: if we know $K(x)$, then we know $l(x^*)$ and we can construct the $2^{l(x^*)}$ programs of length $l(x^*)$ and execute one step of each of them (in lexicographical order) instead of running them sequentially. The first one that gives x is x^*.

Moreover, in [Li and Vitányi 1997] it is shown that the complexity of the complexity function K results to be $K(K(x) \mid x) \overset{+}{<} log\ l(x) + 2\ log\ log\ l(x)$. Consequently, $W'(x \mid y) = W'(y^* \mid y) \overset{+}{=} K(y^* \mid y) \overset{+}{<} log\ l(y) + 2\ log\ log\ l(y)$. □

Since *log log* $l(y)$ is negligible with respect to *log* $l(y)$, the first shortest program for a string y is at most *log* $l(y)$ worthy from having simply y. In other words, finding y^*, which is not only a very hard problem but also not computable in the general case, turns out to be only worthy *log* $l(y)$. The explanation can be found in that the shortest program for y has a short description "the shortest description for y". An important question is whether this phenomenon generalises for any compression ratio between x and y, or this unintuitive result only happens for maximal compression.

Let us first formalise the notion of compression ratio:

Definition 3.2 Let x be a program for y. We define the compression ratio of y with respect to x as:

$$R(x{:}y) = l(y)/l(x){:}1$$

or simply $R(x{:}y) = l(x)/l(y)$.

From here, the following theorem shows that there is no monotonic relation between W' and the compression ratio.

Theorem 3.4 Consider the set $S=\{<x,y>, x \text{ is a program for } y\}$. In S, $W'(x \mid y)$ is non-monotonic with respect to $R(x{:}y)$ and non-monotonic with respect to $R(y{:}x)$ either.

PROOF. The non-monotonicity with respect to $R(x{:}\ y)$ can be clarified if we see that there is a y such that we can find a non-minimal compressed program x for y with $l(x) > K(y)$ with $K(x \mid y) > log\ l(y) + 2\ log\ log\ l(y)$. This is justified because if we choose $x= <z, r>$ where z is any program for x and r is an uncompressible string such that $K(r) = K(r/y) = K(r/y^*) = l(r)$, and x is still a program for y. By choosing $l(r) > log\ l(y) + 2\ log\ log\ l(y)$ we have $K(x \mid y) > K(z) + log\ l(y) + 2\ log\ log\ l(y)$.

By Theorem 3.3, $W'(y^* \mid y) \overset{+}{<} log\ l(y) + 2\ log\ log\ l(y)$, but we have just seen that there exists a z such that , $W'(x \mid y) > log\ l(y) + 2\ log\ log\ l(y)$,. By the way x was constructed it is necessarily longer than y^* so $R(y^*{:}y) > R(x{:}y)$.

The non-monotonicity with respect to $R(y{:}x)$ is easy to show. Just choose $x=y^p$, i.e. the program "PRINT y". Obviously $W'(x \mid y) \overset{+}{=} 0$. Just choose a y that is compressible. There is a program y' $l(y') < l(y)$ where $W'(y' \mid y) > 0$ (just use y^* and add random bits). Since y is compressible $R(y'{:}y) > 1 = R(y^P : y)$. □

It is reasonable to accept that most non-first-minimal programs for y are more informative than the first-minimal program. This makes the first non-monotonicity. The contrary non-monotonicity is more expectable. In the end, Theorem 3.1, Theorem 3.2 and Theorem 3.3 are sufficient to discard W' as a measure of gain. Moreover, Theorem 3.4 shows that W' is not appropriate for measuring a clear relation (ignoring time) between the compression achieved between x and y, either, so the applications of W' for measuring some kind of information value or gain are discarded.

3.4 Time-Ignoring Information Gain

The previous definitions did not follow any of the properties of the introduction. Here we will introduce new measures to approach the final solution. In addition, in the previous section, we have unveiled the main problems of the preceding measure: time-independence and the arbitrary addition of random information. In this section we present two different ways to avoid or reduce the latter problem whereas the following section introduces a time-dependent version to solve the former.

As it has been said, the aim of the following definition is to avoid the increase of $W'(x|y)$ by the addition to x of random information.

Definition 3.3 The weighed and normalised (time-ignoring) relative information value of x with respect to y is defined as:

$$V'(x \mid y) = W'(x \mid y) \mid K(x)$$

The function is undefined iff $x = \varepsilon$ and it is well defined for any $x \neq \varepsilon$ since $K(x) > 0$.

It is easy to show that $V'(x \mid y) \leq 1$. However it can be a very large negative number if $K(y \mid x)$ is much greater than $K(x \mid y)$.

Theorem 3.5 If x and y are independent then $V'(x \mid y) = 1 - K(y) / K(x)$.

The proof is trivial from Theorem 3.1.

Once again, a relevant case is precisely when x is a program for y.

Theorem 3.6 If x is a program for y we have that $V(x \mid y) \overset{+}{=} K(x \mid y) / K(x)$.

The proof is trivial from Theorem 3.2.

This last theorem and the fact that new and arbitrary information can still be unrelated in any case without being compensated by $K(y \mid x)$ suggests to neglect this term and propose a new definition.

Definition 3.4 The normalised relative (time-ignoring) information gain of x with respect to y, denoted $V(x \mid y)$ is defined as:

$$V(x \mid y) = K(x \mid y) / K(x)$$

The function $V(\cdot|\cdot)$is undefined iff $x = \varepsilon$ and it is well defined for any $x \neq \varepsilon$ since $K(x) > 0$. For every x and y, it is obvious that $1 \geq V(x \mid y) > 0$. These are properties 4 and 5 of the introduction. The upper limit is precisely given when y does not contain any information about x, i.e., $I_c(x{:}y)= 0$, as in Theorem 3.5. This complies with property 3 of the introduction if we understand it in one of the two possible ways: y is useless for obtaining x because x and y are absolutely independent (and not the other interpretation: y has common information with x but it is useless because it is extremely intricate or difficult to discover).

The lower limit is given when $x = y$ (which is the property 2 of the introduction) but also when y is a program for x. In the latter case, when the execution of y is complicated, we may have an important transformation from y to x, so property 2 is not completely fulfilled. Apparently, x can be obtained from y, but the computational time may be extremely high, so y is not so useful to obtain x, something that is not reflected by V.

Moreover, property 1 is only accomplished if the terms 'difficulty' or 'effort' are made equivalent with data size. Finally, the major problem is that $K(\cdot)$ is not computable, which makes V not computable, contrarily to property 6. Let us study in the next subsection a computable version of information gain, which, in addition, solves many of the preceding problems.

3.5 Computational Information Gain

In section 3.2 we saw that the solution relies in not to measure only the data nor only the time. A much more appropriate solution is based on a weighed length-time (LT) foundation.

Definition 3.5 The normalised relative time-space information gain of x with respect to y, denoted $G(x \mid y)$ is defined as:

$$G(x \mid y) = Kt(x \mid y) \; / \; Kt(x)$$

where $Kt(x \mid y) = argmin \{LT(p) \colon \phi(<p,y>) = x\}$ and $Kt(x) = Kt(x \mid \varepsilon)$ and $LT(p) = l(p) + log\ Cost\ (p,y)$, with *Cost* being the computational cost[18] of executing p with input y, (the same weighing as Levin's *Kt*) because it provides a good compromise between space and time, as seen before, but another relation could be tuned. As before, the function $G(x|y)$ is undefined iff $x = \varepsilon$ and it is well defined for any $x \neq \varepsilon$ since $Kt(x)>0$.

The major advantage of G is that it is computable, and due to this fact, in the following we will use the name 'computational information gain' or, simply, if there is no confusion with V, information gain.

Since $G(x \mid y)$ is pondered by $Kt(x)$, information gain measures the proportion (between 0 and 1) of x which is obtained on the help of y. The use of $Kt(x \mid y)$ implies that the measure mixes both the *portion* (which parts of x are obtained from y) and the *degree* (if each of these parts are obtained in a more or less difficult way from x).

For instance, given a 'problem' y = "$(a - 3)(a + 2)(a^5 + 4a^4 - 18a^3 - 64a^2 + 17a + 60)$" and its solution x ="the roots are $a = 3$, $a = -2$, $a = -3$, $a = -5$, $a =4$, $a = -1$, $a = 1$", and a system that has the knowledge for finding roots of arbitrary polynomials (among other abilities), the function $G(x \mid y)$ measures the length of stating correctly the problem "Find the roots of the following polynomial:", the logarithm of a negligible time for discovering the roots of "$(a - 3)(a + 2)$" and finally, the minimum between the logarithm of a considerable time for obtaining the roots of "$(a^5 + 4a^4 - 18a^3 - 64a^2 + 17a + 60)$" and the length of the direct quoting of the solution "$a = -3$, $a = -5$, $a =4$, $a = -1$, $a = 1$". In the case that the best $Kt(x \mid y)$ is obtained through solving "y" there is a part of x which is not helped by y, which is exactly the length of "the roots are:". However, the rest is not profited in the same way, there is a part of x, where y turns to be extremely useful, which is "$a = 3$, $a = -2$", but there is another part where y is only slightly useful, which is "$a = -3$, $a = -5$, $a =4$, $a = -1$, $a = 1$".

[18] The term *Cost* $(p(y))$ does not take into account the cost of a *first* reading y if it is not necessary for computing x, so for every y we have $Kt(\varepsilon \mid y) = 0$.

It is quite doubtful that 'portion' and 'degree' could be isolated in general by any measure (a very similar problem is realised in chapter 6). Fortunately, it is natural to consider both factors in a measurement of information gain, due to property 1: "$F(x, y)$ should be smaller as long as x is more obvious from y and it should be greater as long as x is more difficult to obtain from y".

Finally, it is necessary to state clear now that $G(x \mid y)$ does not measure the degree of certainty of x with respect to y. For this reason, if y is a Boolean problem, there are only two possible values for x, and both of them, *true* or *false*, have a gain of $Kt(x \mid y) / Kt(x) \approx 1$ provided y has a significant complexity.

Once we have unveiled part of the meaning of the function G, it is necessary to study more properties of it.

3.5.1 Fundamental Properties

Let us show in this first subsection some of the properties that were claimed on the introduction. The second subsection is motivated but some difficulties to comply with property 2. The last subsection discusses other properties.

Apart from the first property which we have seen, namely that G is computable (property 6), we can see that all the other properties of the introduction also hold. For instance, properties 4 and 5 are shown by the following theorem:

Theorem 3.7 There exists a constant c such that for every x and y, $\log l(x)/(l(x) + \log l(x) + c) < G(x \mid y) \leq 1$.

PROOF. The second inequality $G(x \mid y) \leq 1$ is obtained by considering that y must only be read if it is necessary for obtaining x, so $\forall x,y\ Kt(x \mid y) \leq Kt(x)$. The limit 1 is obvious by choosing $y = \varepsilon$ and the definition of $Kt(x)$ as $Kt(x \mid \varepsilon)$.

The first inequality is justified by the fact that the numerator follows

$$Kt(x \mid y) \geq \log l(x) \qquad (1)$$

because x must be printed and this takes at least $l(x) + c_2$ units of time. In fact this limit can be come close if $x = y$ because the program "print the input" has a temporal cost of approximately $2 \cdot l(x)$. The denominator must follow this disequality.

$$Kt(x) < l(x) + \log l(x) + c \qquad (2)$$

because in the worst case, when x is random, $l(x) + c_1$ bits of information are needed for the program "print x" and at most $l(x) + c_2$ units of time to be printed. Both constants can be represented by a negligible new constant c. By (1) and (2) we have that $\log l(x)/(l(x) + \log l(x) + c) < G(x \mid y)$. □

When G is near to the maximum 1, a great computational effort (in information and time) is needed to compute x from y. Therefore, y is useless to describe x in less time-space, as property 3 stated. Contrariwise, when G is near to the minimum $log\ l(x) / l(x)$ (very close to 0) it means that y is very useful for describing x.

At first sight, it may seem that G complies with property 2 (i.e. if $x = y$, G is minimum), but this is not the case as the following counterexample shows:

If $x = y$ then $Kt(x \mid y) \leq c_1 + \log(2 \cdot l(x) + c_2)$ because x must be read from the input and written on the output and this takes $l(x)$ steps for reading and $l(x)$ for writing. However, the LT-best program p for x can be such that $\log l(x) << l(p) << l(x)$ and it outputs x in $l(x)$ steps. In this case, $Kt(x \mid p) \leq c_3 + \log(l(p) + l(x) + c_4) << c_3 + \log(l(x) + l(x) + c_4) = c_3 + \log(2 \cdot l(x) + c_4)$. Since $\log l(x) << l(p)$, it is not worthy to use p for $Kt(x \mid y)$ and since p is the LT-best program for x we have that $Kt(x \mid y) = c_1 + \log(2 \cdot l(x) + c_2)$. Depending on these c_1, c_2, c_3, c_4, it may be the case that $Kt(x \mid p) < Kt(x \mid y)$ and consequently $G(x \mid p) < G(x \mid y)$. So we have found a counterexample when $G(x \mid y)$ is not minimum.

To give a more concrete example, consider x = 1,2,3,4,5, ..., n. It is natural to think that there is an n such that the best program p for x follows that $\log n << l(p) << n$. It is just a matter of properly selecting this n to make $Kt(x \mid p) < Kt(x \mid x)$.

The rationale is that for long but highly (and easily) compressible strings, it is not much valuable to read the whole string as an input but describing it in a more intensional way. This is something intuitive and it is clearly reflected by situations when $G(x \mid x) = 1$. In the end, the difference that is necessary for making property 2 hold for G can be at most equal to the difference between $G(x \mid y) = log\ l(x)/(l(p) + log\ l(x) + c)$ and $G(x \mid y) = log\ (2 \cdot l(x))/(l(p) + log\ l(x) + c)$ which is precisely $1 / (l(p) + log\ l(x) + c)$. Since this situation appears only for long strings, this term can be ignored in practice. However, let us see how to solve this minor problem in general.

3.5.2 Unique Interface Formulation

The preceding definitions are applicable to any universal descriptional mechanism: Turing machines, lambda calculi, logical theories, programming languages, etc. In this section we present a particularisation of G that has additional properties, property 2 being among them

Just choose a Turing machine ϕ with two tapes, an input-output tape and a work tape with three symbols: 0, 1 and a delimiting symbol δ. Initially, the position of the machine is at the first cell of the input-output tape and the input is considered from this cell up to the delimiting symbol. When the machine halts, the output is precisely defined from the position where the machine has stopped at the input-output tape up to the delimiting symbol. We dub this kind of machines *unique interface machines.*

If we also define $Kt_\phi(x) = Kt_\phi(x \mid \varepsilon)$ in this reference machine, the following properties are straightforward:

- For every x, $Kt_\phi(x \mid x) = 0$, since the input-output tape does not need to be affected.
- $Kt_\phi(\varepsilon \mid x) = c$, being this constant term c small. The necessary time and information to put the delimiting symbol at the current position of the output tape.
- $Kt_\phi(\varepsilon) = Kt_\phi(\varepsilon \mid \varepsilon) = 0$ because the delimiting symbol is at the current position in the output tape.

It is clear then to see that for every x and y, $0 \leq G_\phi(x \mid y) \leq 1$, and precisely the minimum happens only when $x = y$, i.e. for every x and y, $G_\phi(x \mid y) = 0$ iff $x = y$, which is still stricter than property 2 of the introduction.

Any universal descriptional system can be 'wrapped' into a unique interface machine in order to work with a single input-output tape as the only external interface. Consequently, all the properties of the introduction hold if the system is modified accordingly. In the following I will refer to G in general and we will concretise to G_ϕ if necessary.

3.5.3 Other Properties

In the same way we made for other measures, we can express $G(x \mid y)$ in some other ways, be it for the general case or for special cases. For instance, in Theorem 3.5 we could express $V'(x \mid y) = 1 - K(y) / K(x)$ if x and y were independent.

In the case of G, the 'space-time' independence between x and y is given when $Kt(x \mid y) = Kt(x)$ and $Kt(y \mid x) = Kt(y)$. For G only the first equality has some significance, and this is precisely when y is useless for x and $G(x \mid y) = 1$.

Finally, we will express $G(x \mid y)$ in function of $Kt(y)$. As expected, the following theorem only relates them under a disequality.

Theorem 3.8 There exists a c, such that for every x and y, it holds that $G(x \mid y) > 1 - (Kt(y) + c) / Kt(x)$.

PROOF. Let us prove first that there exists a c such that for every x and y, we have $Kt(x) < Kt(x \mid y) + Kt(y) + c$. Use the Kt-minimal program y' that constructs y. Then x can be generated from it by the less Kt-minimal program p for x given y. Both can be joined for constructing a program for x directly with length $l(y') + l(p) + c_1$ and $\mathrm{Cost}(y') + \mathrm{Cost}(p) + c_2$. So there exists a constant c' such that $Kt(x) < l(y') + l(p) + c' + \log(\mathrm{Cost}(y') + \mathrm{Cost}(p))) \leq l(y') + \log(\mathrm{Cost}(y')) + l(p) + \log(\mathrm{Cost}(p)) + c'$. If we separate both processes, there exists a constant c such that $Kt(x) < Kt(x \mid y) + Kt(y) + c$.

From here, $Kt(x) - Kt(y) - c < Kt(x \mid y)$ and hence $G(x \mid y) > (Kt(x) - Kt(y) - c) / Kt(x) = 1 - (Kt(y) + c) / Kt(x)$. □

For instance, the previous theorem serves to make patently clear that if $Kt(x) >> Kt(y)$ then $G(x \mid y) \cong 1$. In other words, complex concepts (in terms of Kt) are always an information gain over simple concepts.

3.6 Information Gain and Complexity

In this section we will study the computational complexity required for increasing the information gain and which are the relationships with other concepts of Kolmogorov complexity, such as information potential.

The following theorem states the difficulty of obtaining, in an exclusive algorithmic way, an x from a y with a gain close to 1:

Theorem 3.9 Consider a *learning* algorithm A^* in $\mathcal{P}$ (i.e. polynomial), namely $\exists p \in \mathbb{N}^+ : O(n^{p-1}) \leq O(A^*) \leq O(n^p)$, A^* being of constant size, i.e., $l(A^*)= c$. This algorithm deterministically transforms y into x, where x is a program for y, with $n = l(y)$. There is a τ such that for all x and y, if $n > \tau$ and $Kt(x) > k \cdot p \cdot \log n$, then $G(x|y) \leq 2 / k$.

PROOF. For every string of data y, let us construct x in the following way: $x =$ *"apply A^* to y"*. Since we can construct x from $<A^*,y>$ in an easy way $p=$ "apply 1st argument to 2nd argument" $Kt(x \mid <A^*, y>) \leq LT(p) = l(p) + \log cost\,(p) < c' + \log n^p)$. It is obvious that $Kt(x|y) < Kt(x \mid <A^*, y>)$. So we have that $Kt(x|y) < c' + \log n^p = c' + p \log n$.

If, as supposed, $Kt(x) > k \cdot p \cdot \log n$, then the quotient $G(x|y) = Kt(x|y) / Kt(x) \leq ((c' / (p \cdot \log n)) + 1) / \text{k}$. Since $p > 0$, just choose $\tau = n$ such that $c' / (p \cdot \log n) < 1$. From here, $G(x|y) \leq 2 / k$. □

More plainly, the theorem states that if both x and y are long enough and there exists a polynomial algorithm from y to x, then $G(x|y)$ must be low. This means that the measurement of $Kt(x|y)$ is very dependent on the existence of an algorithm from y to x and which complexity this algorithm has. Moreover, it shows the difference between deduction (you have to tell which algorithm to use) and deterministic computation (the algorithm is systematically used). This difference is represented in G by the length of A^*, which can be significant. In other words, it is very different the assertion that A^* can be used from the assertion that A^* with input y necessarily gives x. In the next chapter we will concretise this difference.

Before, we have seen that information gain is very dependent to the complexity of x per se. For instance, if x is a Boolean answer to a question y, the complexity of x is very simple, namely 1 bit, so $Kt(x)$ is usually an exiguous $1 + c$ and this usually forces $Kt(x|y) = Kt(x)$ and logically $G(x|y) = 1$.

There is a concept derived from Kolmogorov Complexity that captures and generalises this idea with respect to the length of x. Let us take the definition from appendix A:

Definition 3.6 A string x is *k-potent* if k is the least positive integer such that $Kt(x) \leq k \log l(x)$.

For instance, the string 1^n is 1-potent because $Kt(x) \approx 1 \cdot \log(n)$ whereas an incompressible string s is $(l(s)/\log l(s) + 1)$-*potent* since $\mathrm{Kt}(s) \approx l(s) + \log l(s)$.

If we regard set of concepts as sequences, a potent sequence can be the formal correspondent to the notion of a hard-to-obtain concept. Obviously, for a potent x there cannot be a short y such that $Kt(x \mid y)$ would be small, because this would entail that x could be obtained and described from y and hence $Kt(x)$ would be also small. The following theorem formalises and limits this idea:

Theorem 3.10 For every x and y, if x is k-potent then $G(x \mid y) > 1 / k$.

PROOF. If x is k-potent then $(k-1)\log l(x) < Kt(x) \leq k \cdot \log l(x)$. Since $Kt(x \mid y) > \log l(x)$ because it must print x we have that $G(x \mid y) = Kt(x \mid y) / Kt(x) > \log l(x) / (k \cdot \log l(x)) = 1 / k$. □

Since Theorem 3.10 marks a lower limit, it is only illustrative when k is low, namely when x is very easy to obtain, and whatever y cannot be used for obtaining x with less effort. For the previous example, $x = 1^n$, since there is a constant c such that x is 1-potent for every $n > c$, we obtain that for every y, $G(x \mid y) = 1$.

Theorem 3.10 holds for every y, even the case $y = x$, although in this special case we have that $G(x \mid x) = Kt(x \mid x) / Kt(x) < Kt(x \mid x) / ((k-1) \cdot \log l(x)) < 2 \log l(x) / ((k-1) \cdot \log l(x)) = 2 / (k-1)$ which leaves an interval $(1 / k, 2 / (k-1))$.

Finally, let us see with two examples how information gain applies for NP problems. Consider the SAT problem, i.e. to decide whether a given Boolean formula in conjunctive normal form is satisfiable. If we consider y the problem and x a certificate of the true solution (and we consider NP ≠ P), and y has n variables, this certificate requires n bits to be expressed (a bit for each variable). Since the best algorithm for solving the SAT problem, as far as it is known to date, is exponential, y is not directly useful for x because the cost of solving the problem would be 2^n and $\log 2^n = n$ so $G(x|y) \cong n / n = 1$.

Even in the case that part of y can be profited if some extra information z is provided such that $m = l(z) < n$ and x can be computed from y and z. The question is the cost of computing x from y and z. This cost cannot be less in the general case than 2^{n-m} , because otherwise x could be computed from y by evaluating all the possible z (and there are 2^m possibilities) and then it would have cost less than $2^m \cdot 2^{n-m} = 2^n$, placing the problem in P. As a conclusion, we can only obtain again an upper limit 1 for the general case of this NP problem (although in many particular cases it may happen that $G(x \mid y) << 1$).

As a second example, consider the colouring problem, i.e., the problem of deciding whether a given graph of n nodes can be node coloured with k colours, such that no edge connects two nodes of the same colour. It is known that this problem is an NP-complete problem, due to its reducibility to the decision problem SAT. Let us consider again y the problem and x the solution (and we consider NP ≠ P). If the graph has n nodes, the solution will consist of a string of length n as $(c_1, c_2, ..., c_n)$ where each c_i represents that node i has colour c_i. In the worst case, this string has length (in bits) $n \cdot \log k$. In this case the cost of solving the problem still depends on the essay of the $2^{n \cdot \log k}$ combinations. Since $\log 2^{n \cdot \log k} = n \cdot \log k$ we are still close to the same upper limit $G(x \mid y) \cong n \cdot \log k \; / \; n \cdot \log k = 1$.

3.7 True Information Gain

Although $G(x \mid y)$ follows all the requirements of the introduction and many stability and robustness properties that have been shown in the previous section, it is important to be able to distinguish whether the gain $G(x \mid y)$ is obtained by an addition of random and unrelated information or, on the contrary, is obtained because the computational effort from y to x is high, but no additional information is needed. Since the result is relative to the complexity of x, this effect is somehow reduced in $G(x \mid y)$ but not eliminated. It is possible to compare $G(x \mid y)$ with $G(y \mid x)$ to exclude some cases of unrelated information. In general, however, it is impossible to know *effectively* when x does not contain random and unrelated information, because this information can be interlaced with the rest in many intricate ways (even in a cryptographical way). However $K(x \mid y)$ exactly represents (but not computes) this common information. This allows the following definition:

Definition 3.7 The true information gain of x with respect to y, denoted $TG(x \mid y)$ is defined as:

$$TG(x \mid y) = (Kt(x \mid y) - K(x \mid y)) \; / \; Kt(x)$$

The name *true information gain* is justified by the fact that it compensates what it is not easily in y or it is not at all in y ($Kt(x \mid y)$) and what it is not at all in y ($K(x \mid y)$). The result is a measure of exclusively what is in *y* but it is not easy to obtain[19]. In other words, *TG* measures how much information of *x* is hardly implicit in *y*.

For example, in the previous NP problems we have that $K(x \mid y)$ is very low (just essay all the possibilities), so in these cases $TG(x \mid y) \cong G(x \mid y)$. This makes *TG* a measure of computation time gain.

Consequently, if new unrelated information *w* is added to *x* and the rest *z* can be obtained from *y* without regarding time-resources, $TG(x \mid y) = (Kt(x \mid y) - K(x \mid y)) / Kt(x) = (K(w) + Kt(z \mid y) - K(w)) / Kt(x) = Kt(z \mid y) / Kt(x)$, which only considers the gain which is produced from *z* and ignores the unrelated information *w*.

One can wonder why we have not restricted before this additional information in this way. The rationale is simple. It is useful for the measurement of the value of a solution with respect to a problem, because the answer is *implicitly* in the question. However, it would not be useful in the case of inference processes such as deduction and induction because both of them require some additional information. In the case of deduction we must select which conclusion of all the possible ones is *x* and in he case of induction we must provide a selection criterion.

In the end, *TG* is not necessary if we use both $V(x \mid y)$ and $G(x \mid y)$ to perfectly distinguish both cases. For instance if $V(x \mid y)>0$ there is new and unrelated information which has been used to 'inflate' *x*. On the contrary, if $V(x \mid y)=0$ and $G(x \mid y)>0$ we know that all that is measured by *G* is a true information gain. This serves as a distinction between explicit and implicit information. Explicit information is characterised when $V(x \mid y) = 0$ and $G(x \mid y) \cong 0$ or, in Kirsh's words "*information is explicitly only when it is ready to be used. No computation is necessary to bring the content into a usable form*" [Kirsh 1990]. On the other side, implicit information is given when $V(x \mid y)=0$ and $G(x \mid y)>0$. Again, Kirsh's words show the correspondence with the informal notion of implicitness "*the hallmark of implicit information is (what) it is not explicit but could be made explicit*".

Moreover, $G(x \mid y)$ is still useful for problem-solution matters because one can constrain externally the introduction of random information. The advantage is that it is more flexible and, more importantly, $G(x \mid y)$ is computable whereas $TG(x \mid y)$ and $V(x \mid y)$ are not. In other words, to know whether some piece of information is implicitly in another piece of information is not computable.

3.8 Representation Gain

Information gain allows to compare any two objects *x* and *y*. In the special case where *x* is a program or representation for *y* the gain can still be between almost 0 to

[19] Note that $Kt(x \mid y) - K(x \mid y) = min \{ LT(p_x) : \phi(p_x \mid y) = x \} - min \{ l(p_x) : \phi(p_x \mid y) = x \}$ is not equal to $min \{ log\ Cost(p_x) : \phi(p_x \mid y) = x \}$.

1. However, there may be the case one wants to compare three objects x, x' and y with the following relationships: x is a representation for y and x' is also a representation for y.

We can use directly G to compare x and x'. In the case x and x' are representations for y we will say that $G(x' \mid x)$ is the *representation gain* from x to x'. However for most cases, we will have that $G(x' \mid x) = 1$ and $G(x \mid x') = 1$ because x and x' are unrelated descriptions for y.

For example, consider a system ϕ which solves arithmetic expressions. It reduces the following expressions:

sqrt(81) / 3 = 9 / 3 = 3

45 – 42 = 3

and let us denote the different terms as x = "sqrt(81) / 3", x' = "9/3", y = "3" and x''= "45 – 42". In the system, if x, x' or x'' are computed, all of them produce the output y. It is natural to obtain that $G(x'' \mid x) = 1$ and $G(x \mid x'') = 1$. However, it is expectable that $G(x' \mid x) < G(x \mid x'')$. This is because $G(x' \mid x)$ is small, although not minimal because it is necessary to say where the evaluation must take place and up to which extent. G shows precisely what gives it name, the informativeness of the description: x is more informative than x' but it cannot be said that x is more or less informative than x''. In addition, it seems that the expression x''' = "$\log_5(7^8 - 10! + 1218567124) + 27 - 37$" is not more informative than "3". Accordingly, $G(x'''|y) = G(y |x''') = 1$, despite the fact that the first term results into the second one, the computational cost is high and then it is not useful for obtaining "3" since it is more economical to quote "3" directly.

But it is precisely this meaning of information gain what makes G inappropriate for the discernment of a closely related notion, simplification. We would like a non-semantical function that says that y is the result of all of them, that x' is a simplification of x and that x'' and x are alternative representations for "3".

3.8.1 Universal Simplification

For the previous example, we know that $V(y \mid x) = V(y \mid x') = V(y \mid x'') = 0$ and it is expectable that $G(y \mid x) > G(y \mid x')$. But $V(x' \mid x) \neq 0$ because we must say that the derivation must stop before the complete evaluation of the term. If we choose that x' is a simplification of x exclusively because $G(y \mid x) > G(y \mid x')$, the problems arise immediately because usually $G(y \mid x) > G(y \mid x'')$ despite the fact that x'' is not a simplification of x. As a result, we must add the condition that x' can be obtained from x (and not the reverse way), i.e., $V(x' \mid x)$ is low but not necessarily $V(x \mid x')$. Note that we are using V and not G because a simplification may take a long time.

The following definition formalises this idea:

Definition 3.8 A concept or formula x' is a (r_1,r_2)-simplification of x in ϕ iff $\exists y = \phi(x)=\phi(x')$, i.e. x and x' are program for y in ϕ, and

$G(y \mid x') < G(y \mid x)$ and $K(x' \mid x) \leq r_1 \cdot log\ K(x')$ and $K(x \mid x') > r_2 \cdot log\ K(x')$

Intuitively, y is easier to obtain from x' than from x, and x' is implicitly in x but not otherwise. We have used the logarithm in order to make the parameters (r_1,r_2) range more appropriately. In addition, the last disequality has denominator $K(x')$ and not $K(x)$. This is because $K(x)$ is usually greater than $K(x')$.

Note that the condition $\phi(x)=\phi(x')$ forces that x and x' are denotationally equivalent. The first disequality says that x' is a simpler representation than x for y. The last disequalities favour that $K(x)$ would be usually greater than $K(x')$ (but it is not always the case).

Since there is no definite limit when a transformation can be considered an actual simplification, the parametres (r_1,r_2) can be used to adjust the definition for different purposes. Normally, $r_2 > r_1$. The choice of these constants is even more important because we are talking about *programs*, which are usually of small size, and constants are important.

A direct property of the preceding definition is that if x' is a (r_1,r_2)-simplification of x, $V(x \mid x') > (r_2-r_1) \cdot \log K(x')$ and $V(x' \mid x) < (r_1-r_2) \cdot \log K(x')$, which illustrates the difference in effort of both ways.

The reader could be tempted to recover more traditional notions of simplification: "x' is a simplification of x if x' appears in a subsequent step of a derivation from x' to x". This is a quite comfortable definition but it is also pretty restrictive. First of all, it requires the notion of deduction instead of computation, because computation has only one possible derivation. Secondly, it only contemplates the possible derivations that are allowed by the system, and it excludes some other possible simplifications. Thirdly, and most importantly, there are intermediate steps in a derivation that are not in any way a simplification.

Consider the alternative derivations for the previous example:

sqrt(81) / 3 = sqrt(81 / 9) = sqrt(9) = 3

45 – 42 = 5 – 2 = 3

45 – 42 = 45 – 42 + 0 = 3

The first one should be excluded by most *innermost* functional evaluators, although it is the most efficient way in this case. Moreover, few would recognise sqrt(9) as a simplification of sqrt(81) / 3 without knowing the middle step. The second one is a rule that is only used by humans, because few arithmetical systems have imbedded that kind of rules. The third case shows a derivation where the second step is not a simplification of the first one.

To account for these cases, there is a possible mixture of both approaches:

Definition 3.9 A concept or formula x' is a derivational simplification of x in ϕ iff $\exists y = \phi(x)=\phi(x')$, i.e. x and x' are program for y in ϕ, and

$G(y|x') < G(y|x)$ and there is a derivation from x to y which uses x' as an intermediate step.

This does not solve all the preceding problems because it allows that the following possible derivation:

sqrt(81) / 3 = 9 / 3 = 3 = 5 – 2 = 45 – 42 = 3

would make "45 – 42" a simplification of "sqrt(81) / 3".

In this case, it is better to use Definition 3.8, because the notion of simplification must depend on the notions of effort and complexity.

Definition 3.8 does not only mean that the cost from x to x' is less than x' to x but that the information cost from x to x' must be small. This implies that if x' and x are programs for y and x' is shorter than x but a great extra-data effort is required to transform x into x', we cannot talk about a simplification, because x and x' are alternative and differentiated programs. We will get back on the questions of evaluation and simplification in the next chapter. For the moment, let us introduce the notion of reduction.

Definition 3.10 A concept or formula x is (r_1,r_2)-reduced iff $\neg\exists x'$ such that x' is a simplification of x.

This is the descriptional correspondence to normal form or completely evaluated formula of functional programming.

3.8.2 Representational Optimality

The previous subsection formalised the ideas of simplification and normal or reduced form in a generic and non-semantical way. However, we have not dealt with representational optimality, i.e., the fact that there are better representations than others. For instance, 256 is the decimal representation for s(s(s(... *256 times* ... (s(s(0)))...))), and 2^8 is another representation for 256, and consequently for s(s(s(... *256 times* ... (s(s(0)))...))). Naturally, s(s(s(...256 times... (s(s(0)))...))) is a representation of itself. The question is: is there any optimal representation for this object?

Given any three objects x, x' and y, x and x' being representations of y, by using information gain directly we could compare that x' is better than x by $G(y \mid x') < G(y \mid x)$. As we have seen this would yield the normal form as the best representation. On the contrary, if we say that x' is better than x if $G(y \mid x') > G(y \mid x)$, we would have that long and intricate descriptions are optimal. True information gain is not valid either because the information of the representation is not implicitly coded in what is represented (it is just otherwise).

Fortunately, the notion of optimal descriptions is the major issue of descriptional complexity and, as was remarked in the introduction (and Appendix A), we can measure the length (the shorter the better), the time (the faster the better) or a combination of both. If we choose this last option (and the function *LT* again) we can say that a representation x' is LT-better than a representation x iff $\exists y = \phi(x) = \phi(x')$, i.e. x and x' are programs for y in ϕ, and $LT(x') \leq LT(x)$.

In the previous case the representation 256 is usually better than both s(s(s(... *256 times* ... (s(s(0)))...))), and 2^8 because either it is shorter and still efficient for obtaining the normal form (in unary notation) or it is approximately of the same size but much more efficient. We can generalise this idea:

Definition 3.11 The representation enhancement between two representations x' and x for y is defined as

$$RE(x', x) = (LT(x) - LT(x')) \, / \, Kt(y)$$

This allows the definition of a concept that can be understood as a "maximum" or optimal representation:

Definition 3.12 A representation x for y is LT-optimal iff $\neg\exists x'$ such that $RE(x',x) > 0$.

This definition is equivalent to the following one: x is LT-optimal iff $LT(x) = Kt(y)$, the less complex (in *LT* terms) description. In chapter 2 we commented on the choice of simple descriptions for induction and we will get back on this in the next chapter (instead of y as the normal form, we will use y as the best representation).

A different result can be obtained if we combine the definition of simplification with the definition of representation enhancement to obtain the following definition of local optimality:

Definition 3.13 A representation x for y is locally optimal iff $\neg\exists x'$ such that x' is a simplification of x or x is a simplification of x' such that $RE(x',x) > 0$.

In some way this represents a topological concept. There are infinite many representations for a given string y, but they can be classified into different derivation lines (a representation can be in different lines). In the previous example, if we consider that the representation 2^8 is better than s(s(s(... *256 times* ... (s(s(0)))...))) then we have that 256 and 2^8 cannot be simplified one into another. In other words, they are alternative representations. Moreover if both are local optima, we have that they are alternative canonical representations for s(s(s(... *256 times* ... (s(s(0)))...))).

Since for every finite string y there is always a program y^P= "PRINT y", we can abuse notation and say there is a representation enhancement (or optimisation) between y and x, being x another program for y.

Finally, it is important to distinguish the different notions that have been seen in this section: representation gain, simplification, and optimisation. Representation gain and simplification are inversely related notions in general, since gain usually increments information and a simplification usually decrements it. Optimisation is a more classical notion that it is somehow between the other two. We will get back on them in the next chapter.

3.9 Comparison with Related Information Measures

The motivation of the introduction of new measures of information gain was the fact that not only new and independent information can augment an agent's knowledge. The rationale is clear: first, there are no omniscient systems in practice, and, secondly, there cannot be such a thing, because to know if two pieces of information are independent is undecidable in general.

Nonetheless, almost every measure of information or information gain that has been introduced to date has been given in unbounded terms (i.e., without considering space or time resources). We will comment in the next chapter some exceptions from the literature. However, there are related notions dubbed with different terms that consider the notion of effort or resource consumption. Some measures have led to complete theories. In this section we will revise three of these theories, Kirsh's theory of explicitness, Nake's Theory of Aesthetics and Schmidhuber's Interestingness, which are mainly informal. The use of information gain is useful to formalise them, and, in some cases, the outcomes are direct and enlightening.

Many other measures are related with induction, with deduction or with both, such as Pietarinen's Systematic Power (both), Hintikka's Deep Information (deduction), Quinlan's Gain Ratio (induction), MDL principle (induction) and Bounded Rationality (both). We leave the discussion about these measures for the next chapter. In fact, Quinlan's *gain ratio* is the measure which is perhaps most related with the information gain that we have presented in this chapter, because *gain ratio* turns out to be essentially the same as $V(x|y)$ (or, more properly, essentially its contrary).

Let us begin with a problem that solely justifies the introduction of information gain: the difference between explicit and implicit information.

3.9.1 Kirsh's Theory of Explicitness

In "*When Is Information Explicitly Represented?*" [Kirsh 1990] Kirsh affirms that most discussions of knowledge and representation fall into paradoxes due to weak and ambiguous notions of the terms 'explicit' and 'implicit'. Although initially moved by the 'deeper' notion of implicitness, he soon recognises that explicitness has also been problematic. Under the premise that clarifying explicitness is a prerequisite to clarify implicitness because "*implicit is that which is not explicit but which could be made so*" [Kirsh 1990], he introduces a theory of explicitness, postponing a theory of implicitness.

The four conditions of explicit information are stated as follows:

- Locality: "the symbols which explicitly encode information must be easily separable from each other".
- Movability: "An ambiguous language may explicitly encode information only if it is trivial (non-ambiguous) to identify the syntactic and semantic identity of the symbol"
- Immediately readable: "symbols explicitly encode information if they are readable in constant time".
- Meaning: "the information which a symbol explicitly encodes is given by the set of associated states, structures, or processes it activates in constant time"

If we consider x the information to be represented explicitly by y in a computational way, then y is a program for x. The first two conditions can be easily fulfilled under the notion of computation. The first one requires only to consider a code such that if $\phi(x) = a$ and $\phi(y) = b$ then $\phi(xy) = ab$, i.e., incontextual. The second condition is imbedded by the non-ambiguous nature of computation.

By the third condition, we are forced to have $Kt(x \mid y) \leq k \cdot \log l(x)$ (linear time) or even stricter $Kt(x \mid y) \leq k$ because y is a program for x and x must be obtained from y in constant time. This usually implies that $G(x \mid y)$ would be low if x has some complexity.

Finally, the last condition can also be understood in this framework if we consider y as an expression, x as its meaning and z as the background knowledge (associated states, structures, or processes it activates). In this case we have that $Kt(x|<y,z>) \leq k$ because the meaning of x can be obtained from y and z and it must be obtained in constant time. Again, this implies a low value of $G(x|<y,z>)$.

If we relax the condition of y being a program for x, we can generalise the previous results on explicitness as a degree between implicitness and explicitness. Moreover, this has been done in this chapter; we have already presented two functions that formally state the degree of implicitness and explicitness. In addition, most of Kirsh intuitions about the matter are fulfilled by these functions. Concretely, $V(x \mid y)$ measures how much information of x is implicitly present in y. If $V(x \mid y) \cong 0$, then x is all implicitly in y. If $V(x \mid y) \cong 1$, then no part of x is implicitly in y. On the other side, $G(x \mid y)$ measures how much information of x (and in which degree) is explicitly present in y. If $G(x \mid y) \cong 0$, then x is all explicitly in y. If $G(x \mid y) \cong 1$, then x is not explicitly in y.

But all this also shows that Kirsh's premise is false: explicitness is not necessary to ascertain implicitness. Moreover, $V(x \mid y)$ and $G(x \mid y)$ generalise the notions of implicitness and explicitness. The following table summarises this relation:

$V(x \mid y)$	$G(x \mid y)$	$TG(x \mid y)$	$V(x \mid y)/G(x \mid y)$	*Meaning*
1	1	0	1	x is neither implicitly nor explicitly in y
0	1	1	0	x is deeply implicitly in y
1	$\cong 0$	-	-	*Impossible*
0	$\cong 0$	$\cong 0$	0	x is explicitly in y

Table 3.1. *Different cases and degrees of implicitness and explicitness.*

Although $V(x \mid y) / G(x \mid y)$ is well-defined and it is always between 0 and 1, it is not sufficient to separate the three different cases (rows) covered by table 3.1, and TG cannot differentiate them either. It is necessary then to use two functions. However, from the definition of V, it is important to realise that only explicitness is computable, so the first two cases of table 3.1 are effectively indistinguishable. This, once again, justifies the use of $G(x \mid y)$ as the only practical function to discern how much information is explicitly or implicitly present between two concepts.

3.9.2 Nake's Theory of Aesthetics and Schmidhuber's Interestingness

Nake [Nake 1974] suggested that maximally interesting and aesthetically pleasing input data exhibits an ideal ratio between expected and unexpected information. In other words, things are considered boring if they are either too random or too predictable.

Although this view of interestingness can be found earlier than Nake more or less definitely, it is only recently that this idea has been adopted for AI. In particular, Schmidhuber gets inspired by this notion of interestingness to propose a 'curious' agent [Schmidhuber 1997b] and a theory of incremental self-improvement [Schmidhuber 1997a], which tries to augment its knowledge incrementally from interesting things, ignoring what is known and what is too complicated for the agent. Although Schmidhuber only formalises this idea for a particular agent architecture, the issue is to measure the effort from the agent's knowledge to a given knowledge to obtain its "difficulty".

In an absolute way, this theory of interestingness could be well formalised by $K(x)$ or $Kt(x)$, namely x is interesting if $0 << K(x) << l(x)$ or in the space-time variant x is interesting if $\log(l(x)) << Kt(x) << l(x) + \log(l(x))$. A more detailed study could even lead to the notion of logical depth or sophistication, which will be treated in chapter 6.

However, if we relativise for an agent, this turns out to be the following definition: x is interesting iff $\log(l(x)) << Kt(x \mid y) << l(x) + \log(l(x))$, being y the knowledge of the agent. In other words, if $Kt(x \mid y)$ is close to the minimum, then x is known by the agent. On the contrary, if $Kt(x \mid y)$ is close to the maximum $Kt(x)$, then x is extremely novel but also interestless.

Nonetheless, this formalisation of Schmidhuber's ideas would make that short random objects would always be interesting since $Kt(x \mid y)$ is low but still significant. This problem is not clarified by Schmidhuber, because he always refers to long pieces of knowledge.

Fortunately, the solution to this problem is precisely information gain, because it weighs the complexity $Kt(x)$:

Definition 3.14 Interestingness

A concept x is interesting to an agent with a knowledge y iff

$$b - c < G(x \mid y) < b + c$$

where $0 \leq b \leq 1$ is the agent's boldness and $0 \leq c \leq 1$ its curiosity threshold.

For instance, an agent with high boldness $b= 0.7$ and low curiosity $c= 0.1$ would be a presumptuous agent (a know-all).

A final note about interestingness is that it is difficult to separate it from the notion of purposed interest, which depends on many more things than knowledge only; agent's goals and desires. However, as we have shown, not only "there is no knowledge without interest" [Habermas 1972] but knowledge also affects interest, and this bi-directional influence should not be neglected.

3.10 Summary and Contributions of This Chapter

In the introduction we argued on the necessity and possibility of a measure of information gain that evaluates the amount of information which has been made explicit in a reasoning step. In order to make sense, we must be involved with non-omniscient systems, where reasoning has the main goal of transforming knowledge and obtaining 'new' results that were not obvious initially. We proposed several properties that a function of information gain should comply with. Consequently, the first part of this chapter is devoted to give a measure under these requirements.

In section 3.2 we made clear that the difference between explicit and implicit must be based on a measure of effort, and this must be based on a measure of resource consumption or resource complexity. In particular, we justified the use of $\mathrm{LT} = l(x) + \log \mathrm{Cost}(x)$ because it weighs in a very convenient way space and time resources.

Sections 3.3 and 3.4 have discussed in some previous definitions from Kolmogorov Complexity which were introduced for other purposes. We showed that these measures are not valid for our goals but, more importantly, we have discovered the reasons-why they are not valid and what they lack. This insight is exploited in Section 3.5, which introduces Computational Information Gain $G(x \mid y)$, the main contribution of this chapter. Despite its simple definition based on Levin Complexity ($Kt(x \mid y)$), it fulfils all the properties stated in the introduction. In section 3.6 we studied other properties of Computational Information Gain. Especially, we related it with some classical concepts of computational complexity, showing the robustness of the definition in front of polynomial transformations. We have applied $G(x \mid y)$ to some NP problems and we have obtained intuitive results.

In Section 3.7 True Information Gain $TG(x \mid y)$ is introduced as a variant of G that avoids the addition of new and unrelated information. This definition and some previous ones are used later to definitely clarify and formalise the difference between implicit and explicit.

Section 3.8 deals about representations, that is to say, concepts that are descriptions of other concepts. In this case we are interested in comparing two different descriptions of the same concept, so we needed a measure that compares three objects instead of only two. This leads to the notion of representation gain between x and x', which is just a particularisation of gain when x and x' are descriptions of a third concept y. If both x and x' are programs (representations) for y we should retain the best one according to some criterion. Then we talked about simplifications, and we clarified this notion, usual in computational and deductive systems. Finally, we studied optimality, and we used again $Kt(x)$ to measure this resource optimality. Representation gain and simplification are inversely related notions in general, since gain usually increments information and the simplification usually decrements it, whereas optimisation between the other two.

Section 3.9 compared information gain with other information measurements presented by other authors which are closely related with our proposal and we have discussed the problems they have left open or have not formalised. They are easily covered and clarified under our theory. In particular, we have formalised Kirsh's theory of explicitness, Nake's Theory of Aesthetics and Schmidhuber's Interestingness, which were mainly informal. Information gain has enlightened and, in some cases, generalised them.

As leading outset, the main contributions of this chapter are:

- The justification of a necessity of re-connecting the intuitive notion of information with resource consumption or computational/reasoning effort.
- The choice of LT as an appropriate measure of effort, neglecting the idea of effort exclusively based on time or space.
- A measure of time-ignoring information gain $V(x \mid y)$ which represents the degree of information of x which is implicitly in y.
- A new effective measure of *computational* information gain $\mathrm{G}(x \mid y)$, which depends on the computational effort (time and space) and measures the proportion of x which can be easily obtained on the help of y. In other words, the degree of information of x which is explicitly in y.
- The study of its properties and the verification of their robustness with respect to non-polynomial algorithms.
- Representation Gain as a special case of information gain. A general notion of simplification and the definition of a representational optimality criterion.
- The comparison and clarification of different informal but outstanding notions: implicitness vs. explicitness, aestheticism, and interestingness.

Some of the previous results could be, *per se*, quite outstanding. However, the potential of these definitions, especially G, and their full utility will be still unveiled in the next chapter.

4. Information Gain and Inference Processes

Quaerendo invenietis[20]
J.S. Bach, from a canon of Musical Offering (1747)

Abstract: *the notions introduced in the previous chapter are exploited here. The function G is used to explain both the informativeness of a hypothesis with respect to some evidence (in Popper's sense) and the gain that takes place when a conclusion or theorem is deductively established from an axiomatic system. Additionally, a new notion of authentic learning is introduced, ensuring that learning has taken place, independently of how compressible the evidence is, unlike the MDL principle. In the case of deduction, different adaptations of G are introduced for several deductive paradigms, especially for logical programs, which illustrate the measuring in practice of these gains. This chapter also includes a comparison with Hintikka's ideas, establishing the relationship between G and Surface Information, and between V and Depth Information. Several general measures of System Optimisation and Systematic Power, where Intermediate Information is recognised useful in ATP and mathematical practice, are also introduced. Finally, an oblivion criterion is defined, as well as a characterisation of eager and lazy inference methods.*

Keywords: Inference processes, Induction, Deduction, Bounded Rationality, MDL Principle, Information Gain, Discovery, Machine Learning, Informativeness, Creativity, Eager and Lazy Inference, Systematic Power.

[20] Seeking you will find.

4.1 Introduction

The characterisation of inference processes was reviewed in chapter 2, following the current practice in artificial intelligence, which likewise has taken the notions of deduction, induction and abduction from philosophy. However, some adaptations of these processes to artificial intelligence have been quite careless, mainly terminologically. In general, the applications have usually been performed by using inference process alone, with the view that they are *separate* inference processes. In fact, the main branch of artificial intelligence, mostly devoted to deduction, has held little relation with the machine learning community, mostly devoted to induction. More recently, though, traditional paradoxes and inconveniences have been re-appearing whenever more than one inference process is required to be integrated in one application or system. The thing is especially blatant in the conjunction of induction and deduction, because measures and paradigms used for deduction are not only useless for induction but many times are inconsistent with it. Some reasons of this failure of a consistent integration are:

- The deductive-nomological fallacy of explanatory induction introduced in 1949 by Hempel and Oppenheim [Hempel 1965] is a mistake (see e.g. [Thagard and Shelley 1997]), because the necessary general laws (*nomos*) must be frequently discovered by the process and not initially given, as the very special case of non-monotonic deduction. This is only possible for abduction, i.e., non-constructive induction, that can be seen as non-monotonic inference, although some authors are not precise about terminology [Grégoire and Saïs 1997, even using the term induction for abduction [Helft, 1989] [Núñez et al. 1995].
- The dilemma between selection criteria based on informativeness and selection criteria based on likelihood. Despite the fact that Occam's razor (and its incarnation in the MDL principle) has been successful, any criterion based on likelihood requires the definition of a prior distribution, which always is an arbitrary choice. It seems more reasonable to understand any selection criteria from a methodological point of view and let reality refute the wrong hypotheses. In this way, more easily refutable hypotheses are preferable, as Popper has always argued.
- The confusion among three related but different processes: amplificative deduction, as the process of obtaining new theorems of a given axiomatic system, theorem proving, as the process of obtaining the proof of given theorems, and computational deduction (e.g. logic programming) which identifies theorem proving with program computation. Although the last equivalent was neglected some time ago [Fetzer 1988] [Fetzer 1991], it is still common in AI the erroneous view of deduction as a deterministic process.

- The thought of deduction as a perfect process, which is omniscient and truth-preserving. We discussed in the previous chapter that omniscient reasoners are not only unrealistic but also paradoxical in general. But the idea of deduction as *always* truth-preserving is also unrealistic in real systems (even computers). For instance, humans make errors and this does not preclude them to make deductive reasoning. Consequently, we must accept that *deductive reasoning can also be non-truth-preserving*, but not only in the way that it can be approximate (such as many modern and non-monotonic logics have formalised) but possibly erroneous.
- The thought that deduction is non-informative impedes that a system can be motivated to obtain new deductive inferences, because the process from premises to theorems is not valuable, because the conclusion has less information than the premise.
- The apparent computational indistinguishability of deterministic induction and deduction. Consider the following paradox: under a descriptional system ϕ, we have that data x can be interpreted as a valid program for ϕ. If executed in the system ϕ, it gives $\phi(x) = y$, which, casually, turns out to be also a valid program for ϕ. Even more casually, the result of executing the program y, i.e., $\phi(y)$, is the data x. In other words, $\phi(x) = y$ and $\phi(y) = x$, giving a fix point $\phi(\phi(x)) = x$. In this case, it is difficult to say whether y is a deduction of x or is an induction of x.

To clarify most of the preceding questions it is necessary to distinguish between deterministic and non-deterministic inference processes. Given a computational system ϕ, deterministic deduction is the same process as computation in ϕ, i.e., given some axiomatic system x there is only one possible deduction, $\phi(x)$. In the other way, deterministic induction, transforms a given data d in a more intensional representation x, either exact or approximated, such that d is a deterministic deduction of x, namely that x is a program for d, i.e., $\phi(x) = d$.

On the other hand, non-deterministic deduction is an information-demanding process, in the way that an axiomatic system x can potentially give many different consequences, and it is necessary then to provide more information to select which one. In other words, if d is a consequence of x, then there is a need of information in order to obtain d alone, even in the case that the deductive system would be omniscient and ideal.

As it is well known, non-determinism can be formalised by computation by the use of non-deterministic Turing machines. Although the extreme complexity of physical systems can disguise their final deterministic nature (at least to the level of atoms), a fully deterministic system can always introduce some randomness in part of its processes, thus emulating non-deterministic systems. Consequently, it is sufficient to characterise non-deterministic deduction as deterministic computation in the following way. Given a computational system ϕ, d is a consequence of an axiomatic system x iff there exists some selection data w such that $\phi(<x, w>)= d$, where w represents which axioms of x should be used, in which order and up to which extent. More formally, we can distinguish three kinds of deductive systems:

Definition 4.15 A **Computational Deterministic Derivational** (or simply Deductive) **System (DS)** is defined as a deterministic computer ϕ which only accepts programs of the form $\phi(<x, w>)$, where x is an axiomatic system and w is the selection information which indicates which axioms from x must be used, which occurrences must be selected and in which order. Finally, the following condition must also be satisfied:

$$\phi(<x, w>) = d \quad \rightarrow \quad x \vdash d \quad \text{and } w \text{ is a proof for } d \text{ in } x.$$

In contrast,

Definition 4.16 A **Computational Theorem Prover (TP)** is defined as a deterministic computer ϕ, which only accepts programs of the form $\phi(<x, t>)$, where x is an axiomatic system and t is a well formed formula of x such that:

$$\phi(<x, t>) = 1w \quad \rightarrow \quad x \vdash t, \text{ and } w \text{ is a proof for } t \text{ in } x \text{ and}$$

$$\phi(<x, t>) = 0 \quad \rightarrow \quad x \nvdash t$$

Note that for highly expressible axiomatic systems, ϕ may not end for some theorems.

Aside from randomness, non-determinism is usually simulated by the use of backtracking (e.g. Prolog), which gives different proofs for t. Note that Definition 4.15 and Definition 4.16 are 'structurally equivalent'. The difference only depends on the interpretation of the input (w and t respectively) and the output (d and the proof respectively). Note also that Definition 4.16 gives only one possible proof. The definition could be modified to give the best proof according to some criterion.

Definition 4.17 A **Computational Accepter (AC)** is defined as a deterministic computer ϕ, which only accepts programs of the form $\phi(<x, t>)$, where x is an axiomatic system and t is a well formed formula of x such that:

$$\phi(<x, t>) = 1 \quad \rightarrow \quad x \vdash t, \text{ and}$$

$$\phi(<x, t>) = 0 \quad \rightarrow \quad x \nvdash t$$

A clear example of an accepter is a grammar, and these can be classified according to Chomsky's hierarchy, and both deduction and induction are increasingly complex from type 3 to type 0. In addition, instead of a Boolean accepter, where there are two values {0, 1}, this definition can be generalised for any set of values or classes { c_1, c_2, ... , c_n } and we have a classifier system, where the system tells to which class c_i the input t belongs to.

The difference between Definition 4.16 and Definition 4.17 is notorious and fundamental in proof theory. For our purposes it is still more important, since $G(w \mid <x, t>)$ for Definition 4.16 can be high but $G(a \mid <x, t>)$ for Definition 4.17 will be usually low because both outputs (0 or 1) are easy to describe extensionally. In practice, however, many deductive systems (e.g. a Prolog interpreter under slight modifications) are able to perform the three kinds of deductive paradigms mentioned above.

Once the main deductive paradigms are defined in terms of deterministic computation, we can also define the main non-deterministic inductive paradigms.

> **Definition 4.18 Non-deterministic Induction.** Given a computational deterministic deductive system ϕ, non-deterministic induction is defined as a transformation from a given data or evidence d into a more intensional representation x, according to some criterion κ, represented by $\vartheta(<d, \kappa>) = x$, such that exists a w which makes $\phi(<x, w>) = d$. In this case, $\phi(<x, \cdot>)$ represents the generalisation that has been produced by the inductive process (the predictions).

A more general view of induction as theory acquisition and revision can also be seen in this context.

> **Definition 4.19 Incremental Theory Construction.** A first piece of data or data example d_1 generates an x_1 such that there exists a w_1 such that $\phi(<x_1,w_1>) = d_1$. The second piece of evidence d_2 modifies (extends or revises) x_1 into a new theory x_2 such that there exist two $w_{2,1}$, $w_{2,2}$ such that $\phi(<x_2,w_{2,1}>) = d_1$ and $\phi(<x_2,w_{2,2}>) = d_2$. This kind of incrementality is seen in the following chapter.

Concept learning, which is just a kind of inductive inference, can be defined directly from an accepter or classifier system in the following way:

> **Definition 4.20 Concept Learner.** Given an accepter or classifier system ϕ, a concept learner transforms an evidence $d = \{ <t_1, c_1>, <t_2, c_2>, ..., <t_n, c_n> \}$ into a more intensional representation x, such that $\phi(<x,t_1>) = c_1$, $\phi(<x,t_2>) = c_2$, ..., $\phi(<x,t_n>) = c_n$. Once again, $\phi(<x, \cdot>)$ represents the generalisation that has been produced by the inductive process (all the new values that can be classified).

It is important to realise that the preceding definitions have given account for the main paradigms of deterministic and non-deterministic induction and deduction, exclusively based on computation ($\phi(<\cdot, \cdot>) = \cdot$, $\vartheta(<\cdot,\cdot>) = \cdot$). Deduction and induction have been defined avoiding the use of the notion of semantics or truth, i.e., without making difference between a truth-preserving process such as deduction and a hypothetical process such as induction. This allows the use of computation-based concepts such as $K(\cdot)$ and $Kt(\cdot)$, and the derived notions of the previous chapter, to deal with deduction and induction. As a result of the course of this chapter, we will see that deduction and induction are not inverse processes *in terms of information or transformation gain*, or, in other words, it is possible to apply the same measure of gain for both of them.

In this chapter we will apply computational information gain $G(x \mid y)$ for both induction and deduction, according to the paradigms that have been defined in this section. In the case of induction, information gain forces the hypothesis to be informative (or computationally hard to discover) with respect to the evidence, so it provides a formal account of what is to discover and what is to learn. In the case of deduction, computational information gain distinguishes easy deductions from hard ones, and allows, finally, to find a compromise for distinguishing which deductions are worth leaving explicitly, and for which ones it is preferable to deduce them when needed.

4.2 Information Gain and Induction

It seems more natural to study first our measure of information gain for induction, because it has been usually argued that induction is the *only* informative inference process. More precisely, it has been said that the more general the more informative and the more specific the less informative. This idea was informally introduced by Popper in the thirties and then formalised by Bar-Hillel and Carnap in 1953 for first-order theories, giving the famous Carnap's Probabilistic Interpretation of First-Order Predicate Calculus. However, this relation between probability and semantic information has some counterintuitive properties, as we commented in chapter 2. For instance, $p(\top) = 0$, which implies $I(\top) = \infty$, or, in other words, the theory "everything is true" is the most informative one.

On the other hand, there is an exclusively syntactic view of information, represented by Kolmogorov Complexity. This soon motivated the view of "induction as compression" [Solomonoff 1964], popularised by Rissanen's Minimum Description Length (MDL) Principle [Rissanen 1978, 1996]. Although it has been successfully applied in many fields, this view gives many paradoxes too, as we also saw in chapter 2.

For instance, the first unintuitive consequence arises when one intends to assign probabilities to theories. In order to do this, it is necessary to use the prefix-free version of Kolmogorov Complexity $K(h)$, that makes the usual prior $P(h) = 2^{-K(h)}$ be a probability. But this prior provokes the following paradox. Suppose a given evidence y and two deterministic theories (descriptions) x and x' such that $\phi(x) = \phi(x') = y$ where $K(x') < K(x)$. Under this prior, x' should have more probability than x. However, this is not intuitive since both represent the same theory (the prefix-free condition still allows this). The point is that $K(\phi(x)) = K(\phi(x'))$ but $K(x')$ is not equal to $K(x)$. It seems that this problem can be solved by using $P(h) = 2^{-min\{l(h'):\ \phi(h) = \phi(h')\}} =$

$2^{-K(\phi(h))}$, taking into account that the evidence is a prefix of $\phi(h)$. However, this solution has not been commented in the literature (to my bibliographical knowledge) probably because the resulting value $P(h)$ is not a probability ($\Sigma P(h) > 1$).

In the case of non-deterministic induction, the same problem can be reproduced as well. For most criteria, there can be two theories x_1 and x_2 such that not only there exists a w_1 which makes $\phi(<x_1, w_1>)= d$ and there is also a w_2 which makes $\phi(<x_2, w_2>)= d$ but $\phi(<x_1, \cdot>) = \phi(<x_2, \cdot>)$, i.e. they are semantically equivalent. Moreover, in the case that $\vartheta(<d, \kappa>)= x_1$, one theory would be better than the other, simply because it is better according to some criterion. In other words, it is reasonable to use a selection criterion for choosing between different possible theories, but not a prior criterion to assign probabilities.

By using the MDL principle even without a prior, another important problem appears: nothing can be learnt from random strings. Prediction is not possible since the program x= "PRINT y" (the shortest program if y is random) is not enumerative and it cannot even predict the following bit of y. Although we can easily force x to be enumerative by converting it into "PRINT y FOREVER", several questions arise: Is this the best option for prediction? Is it informative? Is it valuable?

Instead of talking about the best model, it would be even more insightful to evaluate how valuable it is to obtain a concrete description and whether it is worthy to remember or forget it. This would be especially useful if an inductive method can consider different hypotheses at a time, because some surprising, strange, difficult to obtain, or curious hypotheses which have not been still refuted can be kept for future use. On the other hand, obvious or easy hypotheses can be forgotten because they would be easily generated again when needed.

In this way, $G(x \mid y)$ provides a uniform measure of the relative value of the hypothesis with respect to the data, the gain of the computational effort which has been invested in the process from the data to the hypothesis. More precisely, if x is the theory and y is the data, the two extreme cases are illustrative:

- Minimum: $G(x \mid y) = \log l(x) \,/\, (l(x) + \log(l(x)) \approx 0$. The theory is evident from the data. In other words, it is very easy to describe the theory from the data. Some examples of this minimum are: a description full of exceptions or with great extensionalities since they can be quoted easily from the data, or the nth order polynomial obtained from n+1 data.
- Maximum: $G(x \mid y) = 1$. The theory is surprising or creative with respect to the data. The data is useless (in time-space terms) to describe the theory ($Kt(x \mid y) = Kt(x)$). It is necessary a great computational work on the data y to obtain the theory and/or there is a need for external information. In other words, the computational effort invested justifies x to be retained, because it is valuable.

It is important to highlight that $G(x \mid y)$ measures the gain from y to x and not the plausibility of x. For instance, the data "*aaa...a*" suggests the plausible theory "repeat *a* for ever" which is easy to obtain from the data. At this point a clear distinction between plausibility criteria and methodological criteria should be done:

- A Plausibility Criterion: a classical selection criterion that chooses the most likely hypotheses. This criterion must be based on prior information, bias information, context, etc. In chapters 5 and 6 we will address this question again.
- A Methodological Criterion: for want of a reasonable plausibility criterion, a methodological criterion can be used to select the most convenient theory for operational purposes (reduction of complexity, optimisation of the whole process, robustness).

For instance, Popper's falsifiability criterion is of methodological kind and it aims for more robust theories, although, in the long run, if the hypotheses cannot be falsified, they become more plausible. The Maximum Likelihood Estimator [Case and Smith 1983] is a plausibility criterion, provided the prior is conveniently supplied, in this case compared with cross-validation [Kearns et al. 1999] and Bayesian Learning [Gull 1988]. Finally, the MDL principle is both a plausibility criterion and a methodological one, because shorter theories are more manageable.

Information Gain is a purely methodological criterion, because there is no reason to think that 'harder' theories are more probable (more on the contrary, very intricate processes in nature are only present in biological systems). However, methodologically, information gain can be used to obtain a good "oblivion criterion". Let us explain this idea. It is well known that the plausibility of a hypothesis depends on the data, the hypothesis itself and the context. But it is also well known that the 'confidence' or 'reliability' of a hypothesis depends mostly on the ability of the agent or learner to find alternative hypotheses. In this case, the classical notion of a learner as a black box that outputs the best hypothesis according to some plausibility criterion is misleading, because if the best hypothesis fails with new evidence, the process must start again. On the contrary, it is more natural to consider the learner separately from the selection criteria. In this case, a set of possible hypotheses is generated by the learner: some of them are more plausible than others and some of them are more informative than others. In this case, it is interesting to maintain those hypotheses that have not been selected by the plausibility criteria but are alternative explanations that could be selected later, if the evidence discards other momentarily better hypotheses. The learner has invested some effort to obtain these hypotheses and they must be preserved to recoup more effort than it would be profited if only one hypothesis is remembered. Let us formalise this idea:

Definition 4.21 Oblivion Criterion. Given a plausibility criteria $PC(h \mid d)$, and a learner with alternative hypotheses and limited memory resources, its memory politics can be ruled by the following oblivion criterion:

$$OC(h \mid d) = G(h \mid d) \cdot PC(h \mid d)$$

The hypotheses with lower *OC* should be forgotten. For instance, if the plausibility criterion is the MDL principle we have $OC(h \mid d) = G(h \mid d) \cdot 2^{-l(h)}$. By the use of the *G* factor, for instance, a hypothesis "print *d*" for a random data *d*, would have $OC(h \mid d) = 0$. This criterion turns out to be a quite reasonable compromise between informativeness and simplicity. For compressible data, a short hypothesis is usually informative and the final value of *OC* is not too much affected by *G*. On the contrary, for random data, hypotheses generated by the MDL principle are discarded because *G* is low[21].

This engages with the classical dilemma between informative and probable hypotheses. It is clear that an explanation must have some degree of plausibility to avoid fantastic hypotheses, but in many applications, such as scientific discovery or abduction, we must regard an explanation as an investment, even a "risky bet" that could be soon falsified. This is merely Popper's criterion of falsifiability [Popper 1962]: one does not always want the most likely explanation, because sometimes it is the less falsifiable / informative too. More precisely, falsifiability is related with the number of restricted worlds that can be testable whereas informativeness is not directly related with testability.

The issue is clear when the data is random (and this usually happens with short data because it makes no worthy any compression). The MDL principle (or the simplicity criterion) just gives the data themselves, which does not correspond to the idea of 'model'. By using *OC*, different informative hypotheses can be induced. This gives clues to the enigma of "hyper-learning" or "poverty of stimulus" [Reuland and Abraham 1993] in those cases where the data suggests some obvious (but useless) hypotheses instead of more creative ones.

4.3 Creativity, Learning and Discovery

The view of learning as compression [Solomonoff 1964] is also supported on the fact that compression[22] and informativeness are positively related in general. Let us show that this is true, at least for 'thorough' learners:

[21] There are, obviously, other traits that have influence over an oblivion criterion. In particular, frequency of use and interest are, in many cases, more important than plausibility or gain. However, frequency of use, plausibility and interest (what might be useful in the future) are included in the measure of reinforcement which will be presented in the next chapter.

[22] Note that the MDL principle does not ensure any compression at all in the general case.

Theorem 4.11 Consider a thorough learner Λ, which examines all the data or nothing of it. If the hypothesis is very short with respect to the data (the compression ratio $R(d{:}h) > l(d) / log\ l(d)$) then $G(h \mid d) = 1$.

PROOF. If the data is $n = l(d)$ bits long, this takes at least n steps to read it, so $Kt_\Lambda(h \mid d) > \log n$ if the learner examines the data and $Kt_\Lambda(h \mid d) = Kt_\Lambda(h)$ if it ignores it. Since the compression ratio $R(d{:}h) > n / \log n$ and $R(d{:}h) = l(d) / l(h)$, we have that $l(h) < \log n$. Consequently, $Kt_\Lambda(h) < \log n$. So it is preferable to ignore the data and obtain $Kt_\Lambda(h \mid d) = Kt_\Lambda(h)$ directly. This gives $G(h \mid d) = 1$. □

The condition of thorough learners may seem arbitrary. First of all, for the two formulations which were considered in the previous chapter (two interfaces and unique interface formulation), a given data "111...^{n}...111" can be induced by reading only m bits and not the n 1's of the data, because the hypothesis "1 forever" is obvious when sufficiently bits (m) are read. However, this can only be made judiciously when there is enough redundancy (m is large).

This can prompt the reader to think that usual learners are not thorough, but reality tells us otherwise. The overwhelming majority of machine learners *always* examine *all* the evidence, so G could be even greater than 1. Note that in the previous chapter we showed that $G \leq 1$ because all the evidence needed not to be read.

The aim of Theorem 4.11 jointly with Theorem 3.9 of the previous chapter is only to show that efficient learners cannot obtain informative hypotheses. More concretely, when using a polynomial learning algorithm for learning an evidence of length n, a compression ratio much greater than $n / (p \log n)$ is required, which, for n great and p low (as it is usually the case) is a very strict and difficult requirement. This supports the thesis that efficient *algorithms that work exclusively from the data* cannot learn valuable hypothesis or, seen the other way, efficient algorithms always quote 'shallowly' part of the data. This highlights the relevance of context, that thing which is given and not learned, which serves to generate the hypothesis by trial and error (using reality as an oracle), or by less drastic approaches, such as genetic algorithms.

In this sense, under the most intuitive notion of creativity, it is considered that a creative concept cannot be easily obtained from anything else that was known before, because in this case it would not be novel. Consequently, the clue to creativity, if there is any, is to avoid repetition of old structures and patterns, in order to make $Kt(x \mid b) = Kt(x)$ where x represents the 'novel' concept and b is the background knowledge. In Aesthetics, this novelty should not be extreme in order to make the concept minimally comprehensible and hence interesting, as we discussed in the previous chapter.

In the context of induction and learning, creativity is usually known as 'discovery', or in Kirsh's terminology, to make explicit something that was deeply implicit.

According to this, not every inductive theory is a discovering; there are some conditions that must be observed:

1. the theory must be 'implicitly' in the evidence *e* and the background knowledge *b*.
2. the theory must not be 'explicitly' in the evidence *e* and the background knowledge *b*.
3. the theory must cover/explain the data jointly with the background knowledge.
4. the theory should be confirmed (otherwise it is just a creative theory or hypothesis but not a discovering).

We saw in the previous chapter how to formalise the first two conditions, taking $d=<e,b>$ namely $V(h \mid d)$ is close to 0 for the first condition and $G(h \mid d)$ is close to 1 for the second one. The third one is semantic and can be formalised in any of the non-deterministic deductive systems of the introduction. The fourth condition depends on further experimentation with more evidence or additional context knowledge (it even could be deductively confirmed). However, the stricter the first condition is followed, the less possibility that there are alternative plausible theories for the evidence, because if not, $V(h \mid e)$ should contain information about which one to select, and it would be high.

Note that the theory can be implicit mainly due to the background knowledge. The thing is completely different if the selection criterion is fixed a priori. In this case, $V(h \mid e)$ can be very low. For instance, in the case of the MDL principle, the shortest description of a given data can be easily described as "the shortest description for the evidence", and the information conveyed by the shortest description given the evidence ($K(x^* \mid x)$) is very reduced, as we saw in Theorem 3.3. This can be extended for V, as the following theorem shows:

Theorem 4.12 Given a thorough learner Λ, there exists a constant c which only depends on Λ such that for every piece of data x longer than c such that $K(x) > k \cdot \log l(x)$, where $k > 1$, i.e. it is not excessively compressible, the first shortest theory x^* for it follows condition 1, more precisely, $V(x^* \mid x) < 2 / k$.

PROOF. By Theorem 3.3, and since $V(x \mid x^*) = 0$, we have that the first shortest theory for a given piece of data x has $V(x^* \mid x) < (\log l(x) + 2 \log\log l(x) + c_1) / K(x^*)$. Since $K(x^*) = K(x) + c_2$ and $K(x) > k \cdot \log l(x)$, then $V(x^* \mid x) < (\log l(x) + 2 \log\log l(x) + c_1) / (k \cdot \log l(x) + c_2)$. Just choose $c = 2^{2^{c_1}}$, and $V(x^* \mid x) < (\log l(x) + 2 \log\log l(x) + \log\log c) / (k \cdot \log l(x)) < (\log l(x) + 3 \log\log l(x)) / (k \cdot \log l(x))$. It is natural to expect that c is great enough to force $3 \log\log l(x) / (k \cdot \log l(x)) \leq (1 / k)$. Otherwise, just choose $c = \max(2^{2^{c_1}}, 2^{16})$, because $3 \cdot \log\log 2^{16} = 12 \leq 16$. Consequently, there is a constant c, such that $V(x^* \mid x) < (\log l(x) / (k \cdot \log l(x)) + (1 / k) = 2 / k$, which for k great means that the theory is implicitly in the evidence. □

But it is condition 2 which is questioned by the MDL principle, because it is not ensured that the simplest theory is deeply present in the evidence. On the contrary, many times it is explicitly found, as the following rationale shows: note that the condition $K(x) >> \log l(x)$ of Theorem 4.12 is precisely the limit of the condition of Theorem 4.11: ($l(x)$ / $l(h) = R(x{:}h) > l(x) / \log l(x)$ implies $l(h) = l(x^*) = K(x) < \log l(x)$), thus it seems difficult to find an interval where the MDL principle ensures discovering in general.

A further insight on the previous results for the field of machine learning suggests that if the hypothesis is evident from the data, no much learning should have taken place. The issue is clear when the data is random (and this usually happens with short data because it is no worth compressing it). For instance, the MDL principle just gives the data itself, which does not correspond to the idea of 'model' or 'explanation'. However, the most important learning paradigms are based on the idea of identification: identification in the limit [Gold 1967], PAC model [Valiant 1984], Query-Learning [Angluin 1988]. These paradigms are designed for infinite data, but a learning algorithm that always gives a completely extensional (and not valuable) description "print x" for any finite data x would formally learn, something that is quite counterintuitive.

From here, and very far from the classical notion of 'identification' [Gold 1967], we propose a different notion of learning (or discovering): the more a system learns the more valuable the description is with respect to the data.

Definition 4.22 Authentic Learning or Discovering. We say that a concept or theory x is an *authentic learning* or *discovering* with respect to y in a context β iff x is a theory or description for y and $G_\beta(x \mid y)$ is close to 1.

This engages with the intuitive notion of learning as "knowing something that was not known" and it is also applicable for other non-omniscient inference processes, even deduction. In a proper way, discovering, as a special case, should be accompanied by a confirmation (condition 4), whereas learning must not necessarily be confirmed, because x is valuable *per se*. Condition 1 is not required for authentic learning, either.

Finally, the idea of *surprise* is more related with anomaly, a failure of prediction, and not only with the idea of novelty or new information. This issue will be addressed later.

4.4 Quinlan's Information Gain and Gain Ratio

In the previous chapter we studied the relation between computational information gain and other related measures, such as Kirsh's theory of explicitness and Schmidhuber's interestingness. In this section we will study the relation with

Quinlan's Information Gain, which will turn out to be the closest measure to *G*, apart from their common name.

The algorithm ID3 for inducing decision trees [Quinlan 1986] and its implementation and last version C4.5 [Quinlan 1993] is the most popular and widely used program of the machine learning community. Its success relies on two facts: firstly, many inductive problems can be reformulated as the problem of inducing a decision tree, and, secondly, its evaluation measure, a modification of classical information gain, allows a compromise between the optimal solution and efficiency.

For instance, let us consider the same observations from Quinlan's classical example, which contains four attributes and two classes:

Outlook	Temp (ºF)	Humidity (%)	Windy?	Class
sunny	75	70	true	Play
sunny	80	90	true	Don't Play
sunny	85	85	false	Don't Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
overcast	72	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play
rain	65	70	true	Don't Play
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

Figure 4.1. *A small training set*

If *C* is the set of class labels, we can compute the entropy of *C* in the classical probabilistic way (Shannon's entropy):

$$info(C) = H(C) = -\sum_{c \in C} P(c)\, log_2 P(c)$$

In this case $H(C) = -9 / 14 \times \log_2(9/14) - 5/14 \times \log_2(5/14) = 0.940$ bits.

If we choose the attribute 'windy' to split the evidence into two different problems we can compute the entropy of each of them as $H(C|windy = true)$ and $H(C \mid windy = false)$. A weighed sum of these two entropies gives the entropy after the split. Generalising this we have:

$$info_X(C) = \sum_{v \in X} P(v) \cdot H(C \mid v)$$

where each $H(C \mid v)$ is the entropy of each subtree which has been generated, knowing *v*. For the previous case, if X_w = { windy = true, windy = false } then

$info_{X_w}(C) = 6/14 \times (-3/6 \times \log_2(3/6) - 3/6 \times \log_2(3/6) + 8/14 \times (-6/8 \times \log_2(6/8) - 2/8 \times \log_2(2/8) = 0.892$ bits.

Then, it is natural to think that what we have gained after the split is the difference in information between the whole evidence and the split evidence. This is precisely what classical information gain formalises; the gain of considering feature X is measured by computing the difference in uncertainty (i.e. entropy) between the situations without and with knowledge of the value of that feature.

Definition 4.23 Classical (or Probabilistic) Information Gain [Quinlan 1986]

$$gain(X,C) = info(C) - info_X(C) = H(C) - \sum_{v\in X} P(v) \cdot H(C \mid v)$$

We have used the term 'classical' because this is just a generalisation of the traditional equation of information given by x about C when X has only one element x:

$$I(X : C) = H(C) - H(C \mid x)$$

where $H(C \mid x)$ is the conditional entropy, defined as $H(C|x) = -\sum_{c\in C} P(c|x) \log_2 P(c|x)$.

The previous example gives $gain(X_w,C) = 0.940 - 0.892 = 0.048$. If we choose X_o = { outlook = sunny, outlook = overcast, outlook = rain } we have that $gain(X_o, C) = 0.940 - 0.694 = 0.246$.

Information Gain, however, tends to overestimate the relevance of features with large numbers of values. Imagine a data set of hospital patients, where one of the available features is a unique "patient ID number". This feature will have very high Information Gain, because it exactly determines the class, but it does not give any generalisation to new instances. To normalise Information Gain for features with different numbers of values, Quinlan [Quinlan 1993] introduced a normalised version, called Gain Ratio, which is Information Gain divided by split info(X), the entropy of the feature-values.

Definition 4.24 Split Info [Quinlan 1993]

$$split\ info(X) = -\sum_{v\in X} P(v) \cdot \log_2 P(v)$$

This last definition is a measure of the "complexity" of X. In the case X has two values and each half of the evidence corresponds to each value we have a split info equal to 1. For the previous example, $split\ info(X_w) = -6/14 \times \log_2(6/14) - 8/14 \times \log_2(8/14) = 0{,}985$, and $split\ info(X_o) = -5/14 \times \log_2(5/14) - 4/14 \times \log_2(4/14) - 5/14 \times \log_2(5/14) = 1.577$ bits.

Definition 4.25 Gain Ratio [Quinlan 1993]

$$gain\ ratio(X,C) = gain\ (X,C) / split\ info(X)$$

In the previous case $gain\ ratio(X_w,C) = 0.048 / 0.985 = 0.049$ and $gain\ ratio(X_o,C) = 0.246 / 1.577 = 0.156$.

This formula is justified experimentally but not theoretically. In fact, both *gain* and *gain ratio* are currently used in practical applications (many times in the same application), although the latter is more robust to large number of values.

By the theorem of (asymptotic) equality between Shannon's Stochastic Entropy and expected Algorithmic Complexity (first proved by Kolmogorov, see e.g. [Li and Vitányi 1997]) we can make the translation to descriptional complexity of the previous definitions:

Definition 4.26 Descriptional Gain and Descriptional Gain Ratio

$$\textit{desc gain}\,(X,C) = \textit{desc info}(C) - \textit{desc info}_X(C) = K(C) - K(C\,|X)$$

$$\textit{desc gain ratio}(X,C) = \textit{desc gain}\,(X,C)\,/\,\textit{desc info}(X) = \{\,K(C) - K(C\,|X)\,\}\,/\,K(X)$$

These last definitions resemble those of information gain of the previous chapter. In particular, the following relation can be established:

Theorem 4.13 For every X and C, $\textit{desc gain ratio}(X,C) = 1 - V(X \mid C)$ up to an independent additive value $c \leq c' / K(X)$, with c' being another independent constant.

PROOF. From Definition 4.26 we have that $\textit{desc gain}\,(X,C) = (\,K(C) - K(C \mid X)\,)\,/\,K(X)$. A well-known property of K is that $K(y|x) \overset{+}{=} K(<x,y>) - K(x)$ so $\textit{desc gain}\,(X,C) \overset{+}{=} K(C) - K(<X,C>) + K(X) \overset{+}{=} K(C) - K(<C, X>) + K(X)$. By the same property, $K(C) - K(<C,X>) \overset{+}{=} -K(X \mid C)$, we have that $\textit{desc gain}\,(X,C) \overset{+}{=} -K(X \mid C) + K(X)$. In other words, there exists a c' such that $\textit{desc gain}(X,C) = K(X) - K(X \mid C) \pm c'$.

By Definition 4.26 we also have that $\textit{desc gain ratio}\,(X,C) = \textit{desc gain}(X,C)\,/\,K(X)$. From the previous result $\textit{desc gain ratio}\,(X,C) = (K(X) - K(X \mid C) \pm c')\,/\,K(X) = 1 - K(X \mid C)/K(X) \pm c'\,/\,K(X) = 1 - V(X \mid C) \pm c'\,/\,K(X)$. By taking $c = c'\,/\,K(X)$, the theorem is proven. □

Since V is always between 0 and 1, for large values of X, *desc gain ratio* and V are just complementary. If $V(X \mid C) \cong 1$, X is completely new or independent to C, so it is useless for making a split in C, and $\textit{desc gain ratio}(X,C) \cong 0$. On the contrary if $V(X \mid C) \cong 0$, X is fully imbedded in C, so it is extremely useful for making a split in C, and, naturally, $\textit{desc gain ratio}(X,C) \cong 1$.

The reason for dividing by $K(X)$ both in G and *desc gain* is quite the same: we are not interested in an absolute value or a ratio, for if not, large X would give always the best gains. Note that both ignore time, in contrast to $G(X\,|Y)$. It would be interesting to include time in the induction of decision trees by a measure of computational gain ratio. Obviously, this falls out the scope of this thesis.

4.5 Information Gain and Deduction

As it is discussed in chapters 1 and 2, Carnap's Probabilistic Interpretation of First-Order Predicate Calculus [Bar-Hillel and Carnap 1953] spread the view of deduction as an inference processes where the result had always less information than the premises, as the property $p(P) \leq p(Q)$ if $P \models Q$ clearly stated.

In fact, inductive criteria such as the generality degree criterion or the MDL principle also corroborate this view. A deduction is always more specific than the premises, and a theory always grows in size if its consequences are obtained and adjoined.

The strongest response to the view of deduction as a non-informative process was endeavoured by Hintikka [Hintikka 70b]: "*there is, in addition to the scandal of induction, a closely related and equally disquieting scandal of deduction (...) How does deductive reasoning add to our knowledge (information)?*".

Once again, the answer must be associated to the notion of effort and the reality of non-omniscient systems; deductions are frequently costly and their results are therefore valuable, informative, novel and, in some cases, surprising.

In this way, $G(x \mid y)$ also provides a uniform measure of the relative value of the conclusions with respect to the premises, the gain of the computational effort which has been invested in the process from the premises to the conclusions. More precisely, if x is the conclusion and y is the premise, the two extreme cases are illustrative:

- Minimum: $G(x \mid y) = \log l(x) \,/\, (l(x) + \log(l(x)) \approx 0$. The conclusion is evident from the premises. It is very easy to describe the conclusion from the premises. Some examples which can produce this minimum are: a conclusion that adds an easy tautology to the premises, a conclusion that just changes the order of some logical components, a conclusion as a direct instance of a premise, or a conclusion composed mostly of the premises and a few derived things.
- Maximum: $G(x \mid y) = 1$. The conclusion is surprising or even creative with respect to the premises. The premises are finally useless (in time-space terms) to describe the conclusion ($Kt(x \mid y) = Kt(x)$). It is necessary a great computational work on the premises y to obtain the conclusion x or there is a need for external information. In other words, the computational effort invested justifies x to be retained. An example of this is a very difficult theorem. For instance, Fermat's theorem is much easier to describe per se that by deriving it from many premises by using Wiles' proof.

Most of deductions fall in between these two extreme cases. However, these extreme instances give a hint of what G measures for deduction. It is interesting to compare with the same analysis we made about the use of G for induction.

The first case ($G(x \mid y) \approx 0$) is somehow much clearer than the second case ($G(x \mid y) \approx 1$) as it is clearly illustrated by the following example from [Holland et al. 1989]: "*The fact that an inference is a valid deduction, however, is no guarantee that it is of the slightest interest. For example, if we know that snow is white, we are free to apply a standard rule of deductive inference to conclude that "either snow is white or lions wear argyle socks." In most realistic contexts such deductions will be as worthless as they are valid*". In fact, if we consider $y = a$ and $x = a \vee b$, and b is much greater than a, the result of applying G is that $G(x \mid y) \cong 1$, which is quite counter-intuitive. There are, fortunately, at least two solutions for this.

The first one is the use of the measure *true information gain*, seen in the previous chapter, defined as $TG(x \mid y) = (Kt(x \mid y) - K(x \mid y)) \,/\, Kt(x)$. Using G we would have that there would be almost no difference between $Kt(x \mid y)$ and $K(x \mid y)$, and we would have $TG(x \mid y) \cong 0$. The second solution is the use of utility criteria like the one that will be introduced in the next chapter. For this reason, in the following we will consider G and not TG because we will suppose there are external restrictions (or utility criteria) that avoid the inclusion of dummy information like "*lions wear argyle socks*".

It is also important to note that this phenomenon also happens in induction. For instance, given the data "1,2,3,4,5,...", a hypothesis "the natural numbers + 1342521515 – 1342521515" would be informative.

Exception made from the cases where dummy information is added, G recovers the intuitive meaning of the word information for deduction, as when it is said that logical and mathematical inference is valuable. In particular, we would like to clarify different cases: given a theory and a proof sketch, how much valuable is the theorem?. Or, given a theory and a theorem, how much valuable is the proof?.

Let us see how this measure particularises for these situations by using the different deductive systems seen in the introduction. For instance, let us first consider a computational deterministic deductive system ϕ as it was given by Definition 4.15 as an oracle from theory x (and proof w) to theorem d (i.e. $\phi(<x,w>)= d$). The time used by the oracle is considered, we have the following results.

First, if we only have the theory x, the gains of obtaining a theorem d that is effectively derivable from it are:

$$V_\phi(d \mid x) \leq min_{w\,:\,\phi(<x,w>=d)}\, K(w) \,/\, K(d)$$

$$G_\phi(d \mid x) \leq min_{w\,:\,\phi(<x,w>=d)}\, \{\, Kt(w) + log\, Cost(\phi(<x,w>)) \,\} \,/\, Kt(d)$$

The gains are limited by the cost of obtaining the proof ($K(w)$ and $Kt(w)$ respectively) and then using it to obtain d in the system ϕ, which in the last case, takes some time. In both cases, it is selected the proof which minimises the whole cost. Since proofs are always longer than the final result, these measures will usually be still high if the *derivation* (i.e. the proof) is not given. In other words, these are upper limits, sometimes w is not used because d is short.

In the case w is also given, we have that $V_\phi(d \mid <x,w>) = 0$ as expected but $G_\phi(d \mid <x,w>) = \min (\log \text{Cost}(\phi(<x,w>)) \ / \ Kt(d), G_\phi(d, x))$ which can be still high if the proof w takes a lot of time and there is no another much shorter proof. A high value of G will then only happen for very intricate proofs and when the conclusion d is very simple and it is more comfortable to quote d directly.

In contrast, a computational theorem prover ϕ, as we saw in Definition 4.16, which outputs a proof w given a theory x and a effective theorem t, gives a $V_\phi(w \mid <x,t>) = 0$ and $G_\phi(w \mid <x, t>)$ is usually low if w is precisely the proof which is generated by ϕ and it is efficiently obtained by ϕ. But if w is a different proof of the canonical one generated by ϕ, and there are too many alternative proofs, $G_\phi(w \mid <x, t>)$ can take practically any value between 0 and 1, because it is easier to quote the proof. We will see later that, in a Horn logic program, the *first* SLD proof for a given atom does not need any extra information. It is implicitly coded if given the program, the deductive method and the selection criteria.

Finally, a computational accepter, given by Definition 4.17, which just outputs a value $a \in \{0, 1\}$ to say if t is a derivable theorem from x, then $V_\phi(a \mid <x, t>) = 0$ if a program for querying the oracle is shorter than quoting extensionally the answer and $G_\phi(a \mid <x, t>) = 1$ because to quote a single bit would be usually much more efficient than to wait for the oracle. Although the use of gain for a computational accepter does not make too much sense for a single atom, we can consider a classifier system, where $a \in \{ c_1, c_2, \dots , c_n \}$. In this case, $V_\phi(a \mid <x, t>) = 0$ if n is great and $G_\phi(a \mid <x, t>)$ would depend on the time cost of the oracle if n is great. In addition, the use of information gain would be useful for computational accepters for checking whether a *set* of atoms are true or false. The result is that a set of Boolean answers $(a_1, a_2, \dots, a_n)$, which would usually larger to quote alone than if one has the theory and the evidence.

Type of Deductive System	*V*	*G*
Ded. Inference System		
• Without proof: $F_\phi(d \mid x)$	$\leq \min_{w \,:\, \phi(<x,w>)=d} K(w) \ / \ K(d)$	$\leq \min_{w \,:\, \phi(<x,w>)= d} \{ Kt(w) + \log \text{Cost}(\phi(<x,w>)) \} \ / \ Kt(d)$
• With proof: $F_\phi(d \mid <x,w>)$	$= 0$	$\leq \min (\log \text{Cost}(\phi(<x,w>)) \ / \ Kt(d), G_\phi(d, x))$
Theorem prover: $F_\phi(w \mid <x,t>)$	$= 0$	*quite variable*
A Boolean Accepter: $F_\phi(a \mid <x, t>)$	$= 0$	1

Classifier System: $F_\phi(a \mid <x, t>)$	$= 0$ for large n	*quite variable.*

Figure 4.2. *Different approximations for V and G for several deductive systems.*

The difference between a theorem prover and an accepter highlights the two components, descriptional and confirmative, of the idea of proof, and it also shows that G only measures the former. There is an informative or descriptional component, represented by G, which therefore is only present by obtaining the proof trace w and not by obtaining only true and false. In this sense, it seems that accepters are useless, because they do not provide information in G terms. But there is also the confirmative component of a proof, at least so important as the informative component, ignored by G, that is contemplated by theorem provers and accepters. By regarding the informative component alone, which has been less studied than the semantic or confirmative component, we are able to distinguish more definitely what is a proof. More precisely, it is not the detailed relation of all the steps from the axioms to the consequence, but only the information that is exclusively required from the axioms to the consequence, under certain constraints (the description must show how to perform valid and necessary combinations).

Regarding all the particularisations: derivational system (DS), theorem prover (TP) and accepter (AC), we have tried to show that the information value of a result is extremely dependent of the particular deductive process that is considered. In other words, there is no descriptional information gain in some deductive processes where the result is easily obtainable from the premises, although this provides a confirmation.

4.5.1 Example: Information Gain for Logical Theories

Figure 4.2 summarises the results that can be obtained in general for several deductive systems. These are rough approximations and, moreover, they are difficult to compute. If we restrict the descriptional language we can be more precise and obtain effective approximations for the preceding measures of gain. In particular, logical theories allow the use of several metrics which have been introduced in the literature for different purposes, and that can be adapted for much more concrete and illustrative measurements of gain.

First order logic is undecidable in general, so the former limits only apply when we know a priori that some theorem or derivation is effectively obtainable from the theory. In this case, V is equal to 0 (except in the case where the proof is required), and G depends on the computational cost of the derivations.

Herbrand theories are a subset of first order logic theories that are semidecidable, resolution is a complete and correct method for them, all this automatisable in any Prolog interpreter. By default, a Prolog interpreter acts as an Accepter, but it can easily be adapted to work as a Deductive System or a Theorem Prover.

In order to apply G to logic theories we must find an approximation for $K(T)$, with T being a logical theory or logic program. The most common and easy solution is to code a logic program as if it were to be transmitted, and $K(T)$ is computed as $H(T)$, the entropy of T. Moreover, in order to express how much information is required to describe a concrete proof of a given evidence with respect to a theory, we will require as well the space complexity of a proof. Finally, it will also be necessary to define the computational cost of a given derivation.

Let us define several approximations for these measures before using them in particular definitions of V and G for logic programs.

4.5.1.1 The Space Complexity of a Theory: *L*(*T*)

Imagine the problem of transmitting a logic program. If we assume that the peer does not know the Herbrand Universe and this is finite, we should also transmit it. Since the names of the predicates, constants and variables are not relevant for the theory, it is only necessary to transmit their number. In this way, the length of a logic program is defined as in [Conklin and Witten 1994]:

Definition 4.27 Space Complexity of a Herbrand Theory

[Conklin and Witten 1994]

$$L(P) = \log(v+1) + 1 + l\,(\log p + 1) + \textstyle\sum_{l\in P} \text{size}(l)$$

v being the number of *different* variables used in the program, p the number of different predicates and l the number of literals. The size of a literal is defined as:

$$\text{size}(l) = a \log (v + c)$$

a being the arity of the predicate and c the number of constants in the Herbrand Universe of the program.

The term $log(v+1)$ serves for determining how many variables are in order to differentiate constants from variables, because they are codified together in size(l). The term $l(\log p + 1)$ specifies which predicate is used in each literal plus an extra bit to say whether it is the last literal of a rule.

This measure does not cover the case of logic programs with function symbols, but this could be solved by using flattening techniques before calculating $L(P)$ [23].

For instance, the following program P_1:

```
{ p(X,a,c).
  g(Z,b) :- h(a), p(W,Z,c).
  h(Y).
```

[23] Another option $L'(P)$ could be size(l) = $\sum_{i=1..a}$ size(arg_i) and size(arg) = log (v+c) + 1 if it is a variable or a constant) or size(arg) = log (f) + 1 + $\sum_{i=1..a}$ size(arg_i) if it is a function.

h(Z,a). }

has 2 variables (after renaming, the second rule is the only one with 2 variables), 3 constants (a,b,c), 4 predicates (p/3, g/2, h/1, h/2), 4 rules and 6 literals. Consequently,

$$L(P_1) = \log(2+1) + 1 + 6\,(\log 4 + 1) + 3 \log (5) + 2 \log (5) + \log (5) + 3 \log (5) + \log (5) + 2 \log (5) = \log(3) + 19 + 12 \log (5) = 48.4 \text{ bits.}$$

The expression of $L(P)$ can be simplified into $L(P) = \log(v+1) + 1 + l\,(\log p + 1) + d' \log (v + c)$ with d' being the total number of arguments in all the literals.

4.5.1.2 The Space Complexity of a Proof of a given evidence: *PC*(*W*| < *T,E*>)

It is easy to describe a proof for a given theory and evidence in logic programs. The measure is called the proof complexity for logic programs and it was introduced by [Muggleton et al. 1988], for a very different purpose: defining an *inductive evaluation criterion*. Although we will get back later on this original use, let us describe this measure right now.

The proof complexity measure was originally denoted $PC(E|T)$ [Muggleton et al. 1988] [Muggleton et al. 1992] and is also known as *derivational complexity* [Feldman 1972] for context-free grammars. It is defined in the following way:

Definition 4.28 Proof Complexity

[Muggleton et al. 1988, Muggleton et al. 1992]

Given a theory T, let :- G_1, …, G_n be the current goal and the root of a success branch in the SLD-tree. In this moment k rules can be selected where G_1 (assuming leftmost computation rules) unifies with the rule head. So, it will require log k bits to select the rule and the proper substitution, and requiring log $(c+v)$ for the substitutions supposing function-free programs ([Conklin and Witten 1994] do not take into account the substitution between variables, either) for every variable in a non-generative rule, that is, a rule where the head contains one or more variables not occurring in the rule body. Let us call $PC(w|<T,a>)$ the information which is required in this way to code the proof w of an atom. So:

$$PC(w|<T,E>) = \sum_{a\in E} PC(w|<T,a>)$$

However, if the evidence is given, then to code the proof does not require to quote the substitutions, because SLD finds them by mgu, and the information depends only to select the leaves that give to different proofs. This means that the information required is equal to 0 if there is only one possible proof. We will denote with $PC(w|<T, a>)$ the information to select a concrete proof w.

For instance, we want to describe p(a,b,b), p(a,c,d), p(b,d,e) and we have the program

$P = \{$	$r_1 \equiv$	p(a,Y,X) :- r(Y), q(Y).
	$r_2 \equiv$	p(Y,b, X) :- q(X).
	$r_3 \equiv$	p(b,X,Y).
	$r_4 \equiv$	r(b).
	$r_5 \equiv$	r(X).
	$r_6 \equiv$	r(e).
	$r_7 \equiv$	r(g).
	$r_8 \equiv$	q(b).
	$r_9 \equiv$	q(d). $\}$

In this case we have:

$PC((r_1, r_4, r_8) \mid <P,$ p(a,b,b)$>) = \log 2 + \log 2 = 2.$

$PC((r_1, r_4, r_8) \mid <P,$ p(a,b,d)$>) = \log 2 + \log 2 = 2.$

$PC((r_1, r_4, r_8) \mid <P,$ p(a,b,c)$>) = \log 2 + \log 2 = 2.$

$PC((r_2, r_9) \mid <P,$ p(a,b,d)$>) = \log 2 = 1.$

$PC((r_3) \mid <P,$ p(b,d,e)$>) = 0.$

$PC((r_2, r_9) \mid <P,$ p(c,b,d)$>) = 0.$

In fact, if we determine the search strategy (top-down), we must only say which of all the possible proofs is, and in this case:

$PC'((r_1, r_4, r_8) \mid <P,$ p(a,b,b)$>) = \log 3 = 1.58.$

$PC'((r_1, r_4, r_8) \mid <P,$ p(a,b,d)$>) = \log 3 = 1.58.$

$PC((r_1, r_4, r_8) \mid <P,$ p(a,b,c)$>) = \log 2 = 1.$

$PC'((r_2, r_9) \mid <P,$ p(a,b,d)$>) = \log 3 = 1.58.$

$PC'((r_3) \mid <P,$ p(b,d,e)$>) = 0.$

$PC'((r_2, r_9) \mid <P,$ p(c,b,d)$>) = 0.$

This usually gives lower values for PC' than PC, in general. On the contrary, if we do not determine a priori the selection rule, nor the search strategy, the information would be greater, in general.

4.5.1.3 The Proof-Relative Space Complexity of an Evidence: $L_{PC}(E \mid T)$

Imagine that we want to reckon the information which is required to select a subset of the consequences of a logic program, for instance, we want to describe p(a,b,b), p(a,c,d), p(b,d,e) and we have the program

$P = \{$	$r_1 \equiv$	p(a,Y,X) :- r(Y), q(Y).
	$r_2 \equiv$	r(b).
	$r_3 \equiv$	p(b,X,Y).
	$r_4 \equiv$	r(c).
	$r_5 \equiv$	r(d).
	$r_6 \equiv$	r(e).
	$r_7 \equiv$	r(g).
	$r_8 \equiv$	q(b).
	$r_9 \equiv$	q(d). }

Before, we have modified Definition 4.28 because the substitutions were not necessary. Now they are necessary, but in this case we also require to modify Definition 4.28 slightly because we must select which predicate opens the search tree.

Definition 4.29 Proof-Relative Space Complexity of an Evidence: $L_{PC}(E| T)$

Given a theory T, and a fact or atom a, we select the predicate from all the possible predicates (only the predicates appearing in the heads are reckoned), which takes log p bits. Then we construct a term G with new fresh variables for each argument of the predicate and we generate a goal *:- G*.

Now let :- G_1, …, G_n be the current goal and the root of a success branch in the SLD-tree. In this moment we can select k rules where G_1 (assuming leftmost computation rules) unifies with the rule head. Thus, log k bits will be required to select the rule and the proper substitution, and log $(c+v)$ bits for the substitutions (supposing function-free programs) for every variable in a non-generative rule, that is, a rule where the head contains one or more variable not occurring in the rule body. There is no extra bit to code which of these non-generative variables, because they all will be given a substitution and they will be coded in the same order as they appear in the head of the rule. However, log $(c+v)$ bits are required because there can be facts which contain variables and substitutions such as W/X (as coding p(X, X, a) from p(W,Y,Z)). However this v varies from rule to rule being exactly equal to the number of non-generative variables of the rule.

Let us call $L_{PC}(a|T)$ the information which is required in this way to code an atom from a given program T. Hence,

$$L_{PC}(E|T) = \sum_{a\in E} (L_{PC}(a|T) + 1)$$

In the previous example we had 6 constants and a non-generative variable in the first rule and two non-generative variables in the third rule, then:

If only non-generative (in order) are generated and there are no function symbols we have:

L_{PC}(p(a, b, b) | P) = log 3 + log 2 + log 7 + log 5 = 7.72 bits.

If we consider functions we would need log c + 1 if the substitution is for a constant and log f + sum(argi) +1 if it is a function term.

L_{PC}(p(a, b, b) | P) = log 3 + log 2 + log 7 + 1 + log 5 = 8.72.

In this case, compare with L(p(a, b, b)) = log 3 + 3· (log 6 + 1) = 12.35, i.e. quoting p(a, b, b) without the program.

4.5.1.4 The Model Complexity of an Evidence: $L_{MC}(E|T)$

The preceding measure is convenient for single atoms or when the size of the evidence is small. However, when E is large, it is usual that $L_{PC}(E|T)$ will be even higher that $L(E)$. Consider for instance:

the theory P_1 = { p(a), p(b), q(X):-p(X), p(f) } and

the evidence E_1 = { p(a), p(b), q(a), q(b) }.

It would be more appropriate to code $K(E_1| P_1)$ as "M(P_1) except p(f), q(f)" being M the Herbrand Model of P_1. Now imagine that E_2 = { p(a), p(b) }. It would be better in this case to code $K(E_2| P_2)$ as "{ p(a), p(b) }". This leads to the Model Complexity measure [Conklin and Witten 1994]:

$$L_{MC}(E|T) = 1 + \min (L(M(T) - L(E)), L(E))$$

which measures the best of the two ways. The first bit is to distinguish which option has been selected. With another additional bit, this could also be combined with L_{PC} into a new measure $L_{MC,PC}$.

4.5.1.5 The Time-Complexity Measure: $Cost(E| T)$

The complexity of logic programs was first studied by [Shapiro 1984]. One of the measures that we introduced is the time-complexity measure $Cost(E | T)$

The cost of checking a single goal $Cost(G | T)$ against a logic program can be measured as the number of rules that have been *essayed* by SLD-refutation, using the standard Prolog leftmost computation rule (i.e., the number of successful or failed unifications) without counting backtracking. This is equivalent to the number of leaves that are traversed by SLD.

Let us consider the following example to illustrate this measure:

```
P = { member(X, [X|Y]).
      member(X, [Y|Z]) :- member(X, Z). }
```

From here we have:

cost(member(a, [c,d,e,a,b,c]) | P) = 7

This cost is a rough approximation, since it is computed independently to the cost of the mgu, which depends on the size of the goal. Moreover, the use of proper compilations or implementations (see [Clark 1991]) can make this measure very variable because some comparisons can be avoided in some cases, even more if partial evaluation techniques are employed. For other important topics about the computational complexity of logic programs see [Shapiro 1984].

Without forgetting these limitations, we can give a measure of the computational cost of a finite set of facts:

$$\mathrm{Cost}(E|T) = \sum_{G \in E} \mathrm{Cost}\ (G|T).$$

For the preceding example, being E = { member(a, [c,d,c]), member(a, []), member(a, [c,d,e,a,b,c]) }

$\mathrm{Cost}(E|T) = 6 + 2 + 7 = 15$

Note that this can be computed with mixed facts (positive or negative evidence).

A slight problem of this approach is that if we are given an infinite evidence (i.e. E is given intensionally and not extensionally) we will not be able to compute $\mathrm{Cost}(E|T)$. The solution can be found by using an arbitrary distribution to extract a finite set from it, and then compute an average cost.

However, in the following, we will also be interested in the computational cost if the proof is also given. In this case, let us denote with $Cost(G \mid <T, w>)$ the computational cost of following the proof w of G, and it is measured in the same way as $Cost(G \mid P)$ but not reckoning the failed branches, just the successful way. This gives the following result for the previous example:

$$\mathrm{Cost}(\mathrm{member(a, [c,d,e,a,b,c])} \mid <\mathrm{P}, (r_2, r_2, r_2, r_1)>) = 4$$

The measure can be generalised for a set of goals as $Cost(E \mid <T, W>)$, where W is a set of proofs. In this case, only positive evidence is possible, because negative ones do not have a proof.

4.5.1.6 Derived Information Measures for First-Order Logical Theories

We have just introduced $L(E)$, $L_{PC}(W \mid <T,E>)$, $L_{PC}(E \mid T)$, $L_{MC}(E \mid T)$, $Cost(E \mid T)$, $Cost(E|<T,W>)$ and, finally, it would be easy to see that $L(E|<T,W>)=0$. By using all these measures we are able to approximate $V(\cdot|\cdot)$ in different ways for different kinds of first-order logical systems. The $G(\cdot \mid \cdot)$ version, however, requires a weighing between space and time, and a proper selection of a L measure with a Cost measure.

For instance, in the case of $L_{PC}(a \mid P)$, it gives the minimal information for selecting a concrete proof of a in P. But this measure does not obtain the whole proof. This does not include the information for selecting from different predicates when only one of them is valid, and extra computation is required to show that one of the others fails. For instance, the program P = { p(X,Y) :- t(X,Y). p(a,b) :- q(a,X). q(a,X) :- r(a,X). r(X,Y) :- s(X,Y). s(X,Y). }, makes that $L_{PC}(p(a,b)|P) = \log 4 = 2$. The first rule does not affect the measure although there is the computational cost for the negative branches. So, $\log Cost(E|T)$ will be used instead of $Cost(E \mid <T, W>)$, giving a proper $LT(E|T) \leq L(E|T) + \log Cost(E|T)$.

With these considerations, let us give first the measures of gain for deductive first-order systems:

$$V(E|T) \leq L(E|T) \; / \; L(E)$$

$$G(E|T) \leq \{ L(E|T) + \log Cost(E|T) \} \; / \; (L(E) + \log CostPrint(E))$$

Note that for $L(E|T)$, both L_{PC} and L_{MC} can be used.

If the proof w is also given, we have:

$$V(E|<T, W>) = 0$$

$$G\,(E|<T, W>) \leq \log Cost(E \mid <T, W>) \; / \; (L(E) + \log CostPrint(E))$$

In the case of a first-order theorem prover the results are slightly different. If W is composed exclusively of the first or canonical proofs (W_c) that the theorem prover generates for each fact we have that:

$$V(W_c|<T, E>) = 0$$

$$G\,(W_c|<T, E>) \leq \log Cost(E|T) \; / \; (L(W_c) + \log CostPrint(W_c))$$

and for any other proof:

$$V(W|<T, E>) \leq L_{PC}(W|<T,E>) \; / \; (L(W)$$

$$G(W|<T, E>) \leq \{ L_{PC}(W|<T,E>) + \log Cost(E \mid <T, W>) \} \; / \; (L(W) + \log CostPrint(W))$$

And, finally if we have an accepter, then to know if a set of evidences E are true or false, we have:

$$V(<a_1, a_2, ..., a_n) \,|<T, E>) = 0$$

$$G(<a_1, a_2, ..., a_n) \,|<T, E>) \leq \log Cost(E|T) \; / \; (n + \log n)$$

The following table summarises these results:

Type of First-Order Deductive System	*V*	*G*
Ded. Inference System		
• Without proof: $F_\phi(E \mid T)$	$\leq L(E\|T)/L(E)$	$\leq \{L(E\|T)+\log Cost(E\|T)\} / (L(E)+\log CostPrint(E))$
• With proof: $F_\phi(E \mid <T, W>)$	$= 0$	$\leq \log Cost(E \mid <T, W>) / (L(E) + \log CostPrint(E))$
Theorem Prover		
• Canonical proof: $F_\phi(W_c \mid <T, E>)$	$= 0$	$\leq \log Cost(E\|T) / (L(W_c) + \log CostPrint(W_c))$
• Other Proof: $F_\phi(W \mid <T, E>)$	$\leq L_{PC}(W\|<T,E>) / L(W)$	$\leq \{ L_{PC}(W\|<T,E>) + \log Cost(E \mid <T, W>) \} / (L(W) + \log CostPrint(W))$
A Boolean Accepter: $F_\phi(a \mid <T, E>)$	$= 0$	$\leq \log Cost(E\|T) / (n+\log n)$

Figure 4.3. *Different approximations for V and G for several first-order systems.*

4.5.1.7 Example

Let us see all these measures in an example:

$P = \{$ $r_1 \equiv$ father(john, michael)

$r_2 \equiv$ father(john, henry)

$r_3 \equiv$ father(john, steve).

$r_4 \equiv$ father(steve, susan).

$r_5 \equiv$ male(john).

$r_6 \equiv$ mother(ann, susan).

$r_7 \equiv$ parent(X,Y) :- father(X,Y).

$r_8 \equiv$ parent(X,Y) :- mother(X,Y).

$r_9 \equiv$ grandfather(X,Y) :- father(X,Z), parent(Z,Y).

$r_{10} \equiv$ sibling(X,Y) :- parent(Z,X), parent(Z,Y). $\}$

P has 3 variables , 6 predicates, 16 literals, 7 constants, which gives $L(P) = \log(3 + 1) + 1 + 16 (\log 6 + 1) + 31 \cdot \log 10 = 163.3$ bits.

Given the following evidences:

$E_1 = \{$ father(steve,susan) $\}$ with $L(E_1) = \log(4)+1+1 (\log 6 + 1) + 2 \cdot \log 10 = 13.2$ bits

$E_2 = \{$ granfather(john,susan) $\}$ with $L(E_2) = 13.2$ bits

$E_3 = \{$ sibling(steve,michael) $\}$ with $L(E_3) = 13.2$ bits

E_4 = { granfather(john,susan). sibling(steve,michael) } with $L(E_4) = \log(4) + 1 + 2\,(\log 6 + 1) + 4\cdot\log 10 = 23.5$ bits

E_5 = { father(steve,susan). granfather(john,susan). sibling(steve,michael). } with $L(E_5) = \log(4) + 1 + 3\,(\log 6 + 1) + 6\cdot\log 10 = 33.7$ bits

$E_6 = M(P)$ = { father(john, steve). father(steve, susan). male(john). father(john, michael). father(john, henry). mother(ann, susan). parent(john, steve). parent(steve, susan). parent(john, michael). parent(ann, susan). granfather(john,susan). sibling(steve,michael). sibling(steve,henry). sibling(michael,steve). sibling(henry,steve). sibling(michael,henry). sibling(henry,michael). sibling(steve,steve). sibling(michael,michael). sibling(henry,henry). . sibling(susan,susan).} with $L(E_6) = \log(4) + 1 + 22\,(\log 6 + 1) + 40\cdot\log 10 + 1\cdot\log 10 = 214.5$ bits

And ignoring the cost of printing, we have for the first measures (without proof),

$V(E_1|P) \leq L_{PC}(E_1|P)\ /\ L(E_1) = (\log 6 + \log 4)\ /\ 13.2 = 4.6\ /\ 13.2 = 0.35$

$G\,(E_1|P) \leq \{\, L_{PC}(E_1|P) + \log \mathit{Cost}(E_1|P) \,\}\ /\ L(E_1) = (4.6 + \log 1)\ /\ 13.2 = 0.35$

$V(E_5|P) \leq L_{PC}(E_5|P)\ /\ L(E_5) = (\log 6 + \log 4 + 1 + \log 6 + \log 1 + 1 + \log 6 + \log 1 + \log 2 + \log 2 + \log 4 + \log 4 + 1)\ /\ 33.7 = 18.8\ /\ 33.7 = 0.56$

$G\,(E_5|P) \leq \{\, L_{PC}(E_5|P) + \log \mathit{Cost}(E_5|P) \,\}\ /\ L(E_5) = (18.8 + \log(1 + 19 + 5))\ /\ 33.7 = 0.70$

In the last case, the effect of the 16 failed unifications of "granfather(john,susan)", supposing top-down search strategy, affect more sensitively to the difference between V and G.

When the evidence increases in number of facts, we see that the gain tends towards a significant value of gain (usually > 0.5), which depends highly on the number of rules with the same predicate in the head.

$V(E_6|P) \leq L_{PC}(E_6|P)\ /\ L(E_6) = (4\cdot(\log 6 + \log 4 + 1) + 2\cdot(\log 6 + 1) + (\log 6 + \log 1 + 1) + 4\cdot(\log 6 + \log 2 + \log 4 + 1) + 10\cdot(\log 6 + \log 1 + \log 2 + \log 2 + \log 4 + \log 4 + 1))\ /\ 214.5 = 155.3\ /\ 214.5 = 0.72$

$G\,(E_6|P) \leq \{\, L_{PC}(E_6|P) + \log \mathit{Cost}(E_6|P) \,\}\ /\ L(E_6) = (155.3 + \log(6\cdot 1 + 4\cdot 2 + 19 + 10\cdot 5))\ /\ 214.5 = 161.7\ /\ 214.5 = 0.75$

For large evidences, though, it could be more efficient to code using L_{MC}.

$V(E_6|P) \leq L_{MC}(E_6|P)\ /\ L(E_6) = 1\ /\ 214.5 = 0.01$

$G\,(E_6|P) \leq \{\, L_{MC}(E_6|P) + \log \mathit{Cost}(E_6|P) \,\}\ /\ L(E_5) = (1 + \log(6\cdot 1 + 4\cdot 2 + 19 + 10\cdot 5))\ /\ 214.5 = 7.4\ /\ 214.5 = 0.03$

This suggests the use of a combined $L_{MC,PC}$ which will give low gains with evidences which have most of the facts from M(T) (almost all the theory) or with few facts which are derivable from a very instanced rules in the theory (exceptions).

If the proofs are provided, the gains get down significantly, $V(E|<T, W>)$ is always 0 as we have seen, and G is only slightly greater than 0 when the proof is difficult:

$G(E_1|<P,W>) = 0$

$G(E_5|<P,W>) \leq \log Cost(E_5|<P,W>) / L(E_5) = \log (1 + 4 + 5) / 33.7 = 0.10$

$G(E_6|<P,W>) \leq \log Cost(E_6|<P,W>) / L(E_6) = \log (6 \cdot 1 + 4 \cdot 2 + 4 + 10 \cdot 5) / 214.5 = 6.1 / 214.5 = 0.03$

Once seen the gains of obtaining consequences of a logic theory we can see that the gain of obtaining a proof (the first proof) is equal to 0 for $V(W_e|<T, E>)$ and

$G (W_1|<P, E_1>) \leq \log Cost(E_1|P) / L(W_1) = \log 1 / (\log 6 + \log 4) = 0$

$G (W_5|<P, E_5>) \leq \log Cost(E_5|P) / L(W_5) = \log (1 + 19 + 5) / (\log 6 + \log 4 + 1 + \log 6 + \log 1 + 1 + \log 6 + \log 1 + \log 2 + \log 2 + \log 4 + \log 4 + 1) = 4.6 / 13.8 = 0.33$

$G (W_6|<P, E_6>) \leq \log Cost(E_6|P) / L(W_6) = \log (6 \cdot 1 + 4 \cdot 2 + 19 + 10 \cdot 5) / (4 \cdot (\log 6 + \log 4 + 1) + 2 \cdot (\log 6 + 1) + (\log 6 + \log 1 + 1) + 4 \cdot (\log 6 + \log 2 + \log 4 + 1) + 10 \cdot (\log 6 + \log 1 + \log 2 + \log 2 + \log 4 + \log 4 + 1)) = 6.4 / 155.3 = 0.04$

The only result with a high value is $G (W_5|<P, E_5>)$ which is again due to failed unifications.

Finally if we have an accepter then to know if a set of evidences E are true or false, we have that $V(<a_1, a_2, ..., a_n) |<T, E>) = 0$ but

$G (W_1|<P, E_1>) \leq \log Cost(E_1|P) / card(W_1) = \log 1 / (1 + \log 1) = 0$

$G (W_5|<P, E_5>) \leq \log Cost(E_5|P) / card(W_5) = \log (1 + 19 + 5) / (3 + \log 3) = 4.6 / 4.6 = 1$

$G (W_6|<P, E_6>) \leq \log Cost(E_6|P) / card(W_6) = \log (6 \cdot 1 + 4 \cdot 2 + 19 + 10 \cdot 5) / (21 + \log 21) = 6.4 / 25.4 = 0.25$

It is in these accepter systems when G is more useful. For instance, it is not worthy to memorise the result of the truth value of E_1 and E_6 because they can more easily be recovered than quoted. However, W_5 is useful to recall for E_5, i.e., it is better to memorise that "granfather(john,susan)" is true than to obtain it from the rules each time is required.

4.6 Hintikka's Surface and Depth Information

As we said before, the first response to the view of deduction as a non-informative process was endeavoured by Hintikka. While depth information is constant in a deductive system, surface information can change. [Hintikka 70b] intended to discern some properties that are found in deductive systems and to explain the intuitive idea that logical truths do provide information.

To distinguish 'cash' information from 'accessible' or 'potential' information, he developed a new theory of *semantic information* based on the probability of constituents in First Order Logic. Therein he formalised the notions of depth and surface information. The theory is difficult to use and has an important drawback, the probabilities depend on the number of individuals taken into account.

Descriptional (or Kolmogorov) complexity allows the formalisation of Hintikka's claims independently of the world of individuals and independently of the descriptional language used (Hintikka's approach was restricted to first-order theories). Moreover, $G(\cdot|\cdot)$ is based on the computable variant of Kolmogorov Complexity, hence it is computable, and it takes into account time, clearly differentiating what is currently or easily available from intricate information that requires a lot of computational effort.

Nonetheless, the idea which is represented by $G(\cdot|\cdot)$, that any effort of transformation must be converted in an information gain, was already formulated by him: "*There cannot be any objective obstacles to seeing the conclusions right there in the premises, for if there were, their removal would constitute an objective gain in information. In fact, the bluntest and frankest of the philosophers taking this line, Ernst Mach, explicitly assented to this conclusion*" [Hintikka 1970b].

$G(\cdot|\cdot)$ can be used in a similar way as Hintikka's notion of surface information, to recover the intuitive meaning of the word information, as when it is said that logical and mathematical inference is valuable, mathematicians can be creative, original and their results are worthy, as Hintikka also asserted[24].

In the examples of logic programs, we have seen the difference between having the theory in an intensional way and having the evidence expressed in atoms. In many cases there is a gain between the intensional form and the extensional form, because there is an effort. The same rationale was used by Hintikka from constituents to atomic statements.

In our case, depth information is the result of considering the omniscience and infinite resources of the deductive system, i.e. using $V(x|y)$ instead of $G(x|y)$. This matches with the undecidability of depth information [Hintikka 1970b] since V is not effective. In fact, Hintikka shows that depth information is the limit of surface information, and we can easily establish a corresponding theorem between V and G.

[24] Moreover, there is nothing subjective or psychological in this notion of surface information, nor therefore in the measures of the additional information which a logical argument gives us.

Consider the notation $speedup(\phi_a, \phi_b) = n$ to denote that ϕ_a is n times faster than ϕ_b, i.e., for each step that ϕ_b performs, ϕ_a performs at least n in the same time, or, alternatively, that ϕ_a makes in 1 step more than n operations of ϕ_b.

Theorem 4.14 Select any universal machine ϕ, and define G_n in a machine ϕ_n such that $speedup(\phi_n, \phi) = n$ and that for any effective program p we have that $\phi_n(p)=\phi(p)$. Then for any pair of finite concepts x,y then $\lim_{n\to\infty} G_n(x\,|\,y) = V_\phi(x\,|\,y)$.

PROOF. The proof is direct from the definitions of G and V. For any pair of finite concepts x and y we have that $\lim_{n\to\infty} G_{\phi_n}(x\,|\,y) = \lim_{n\to\infty} (Kt_{\phi_n}(x\,|\,y) \;/\; Kt_{\phi_n}(x)) = \lim_{n\to\infty} Kt_{\phi_n}(x\,|\,y) \;/\; \lim_{n\to\infty} Kt_n(x) = \lim_{n\to\infty} (\min\, (LT_{\phi_n}(p)\text{: } \phi_n(<p,y>)= x)) \;/\; \lim_{n\to\infty} (\min\, (LT_{\phi_n}(p)\text{: } \phi_n(<p,y>)= x))$. Since for any effective program p, $LT_{\phi_n}(p) = l(p) + \log \mathrm{Cost}_{\phi_n}(p)$ and $\phi_n(p)=\phi(p)$, we have that for all p $\lim_{n\to\infty} LT_{\phi_n}(p) = l(p) + \log \lim_{n\to\infty} (\max(\mathrm{Cost}_\phi(p) \;/\; n, 1))$ since ϕ_n must perform at least one operation. This gives $\lim_{n\to\infty} LT_{\phi_n}(p) = l(p)$ which leads to $\lim_{n\to\infty} (\min\, (LT_{\phi_n}(p) : \phi_n(<p,y>)= x)) \;/\; \lim_{n\to\infty} (\min\, (LT_{\phi_n}(p) : \phi_n(<p,y>)= x)) = \min\, (l(p) : \phi(<p,y>)= x) \;/\; \min\, (l(p) : \phi(<p, y>)= x) = K_\phi(x\,|\,y) \;/\; K_\phi(x) = V_\phi(x\,|\,y)$. □

The distinction between depth and surface information was motivated by the resolute thought that deductive systems are resource-limited and not omniscient (or in Hintikka's words, logical inference is not tautological [Hintikka 1970b]). In this sense, deduction is a much valuable (and informative) process.

Although the approach is qualitative different from Hintikka's, his motivations and results have a strong parallelism with the theory that is presented in this work, especially in this chapter. In addition, Hintikka (and some of his colleagues) also addressed his theory to account for different phenomena in inductive probability, the problem of meaning, mathematical arguments, Frege's distinction between trivial and non-trivial definitions and many other topics directly or indirectly related with the distinction between implicitness and explicitness. Many of these topics are also addressed in this thesis.

4.7 Axiomatic Systems Optimisation

In the case of deterministic systems, where $\phi(T) = E$, and E is finite, we can consider the optimal T as the shortest one (i.e. $l(T) = K(E)$), the best one according to LT (i.e. $LT(T) = Kt(E)$) or, as usual in computer science, the fastest one (in this case T = "print E"). If E is infinite, we have that the shortest theory is still represented by $K(E)$ but in this case the best one according to LT and the fastest one match, and must be obtained in the following way: if E is infinite we have that $Kt(E) = \min \{ LT(T) : \phi(E) = T \} = \min \{ l(T) + Cost(T) : \phi(T) = E \} = \infty$.

In other words, it seems that time cannot be used for infinite evidences. There is a way out for this. We could still obtain the best program according to *LT*: $opt_{LT}(E) = \{ x : \exists c\ \forall n \geq c,\ Kt(E_{1..n}) = x(n) \}$ where $E_{1..n}$ represents the first *n* bits of *E* and *x*(*n*) represents the same program *x* but limiting its output to size *n*. Note that this is not possible to do for any program *x*, so the result of $opt_{LT}(E)$ is a program that outputs one by one, without going back or modifying what has been output so far. Since *E* is infinite, it is straightforward to see that the size of *T* can be ignored, since there exists an *n* such that Cost(*T*(*n*)) would be greater, and we have that $opt_T(E) = opt_{LT}(E)$.

However, for finite evidence (or due to practicality), it is necessary to find a compromise between the size of the theory and its computational complexity. As we commented in chapter 2, this was profoundly studied by Chaitin, who established the correspondence of Gödel incompleteness results in terms of descriptional complexity [Chaitin 1982]. The more powerful a system is the more blatant the dilemma is: "*any formal system in which it is possible to determine each string of complexity less than n has either (...) few bits of axioms and needs incredibly long proofs, or it has short proofs but an incredibly great number of bits of axioms. (...)This is analogous to the dilemma of a scientist who must choose between directly publishing his observations, or publishing a theory that explains them, but requires very extended calculations in order to do this.*" [Chaitin 1974]

Nonetheless, the parallel is more figurative with mathematical practice: "*One does not really want the most compact axiom for deducing a given set of assertions. Just as there is a trade-off between the number of bits of axioms one assumes and the size of proofs. Of course, random or irreducible truths cannot be compressed into axioms shorter than themselves. If, however, a set of assertions is not algorithmically independent, then it takes fewer bits of axioms to deduce them all than the sum of the number of bits of axioms it takes to deduce them separately, and this is desirable as long as the proofs do not get too long. This suggests a pragmatic attitude toward mathematical truth, somewhat more like that of physicists.*" [Chaitin 1982].

In general, though, we are not presented against deterministic deduction, such as $\phi(x) = y$, where there is only a consequence for a given description, but an axiomatic theory, where we have to select the best theory with respect to a set, maybe infinite, of consequences. This generates a more complex problem. Let us see it first with single evidence and then with multiple evidence.

4.7.1 Single Evidence Representational Optimality

We can use the notions of representation simplification and representation optimisation introduced in the previous chapter to show that this trade-off can be made in practice.

Consider the following equational theory *T* for addition. Its shortest representation is:

$$X + s(Y) = s(X + Y)$$
$$X + 0 = X$$

where the rules are oriented left to right. We could derive similar measures for equational programs than the ones that we had for logical programs. However, in this case, the information gain should be measured from any term to its normal form, acting as a theorem prover or deductive inference system but not as an accepter. For instance, if we have that s0 + s(s0 + ss0) = sssss0 then G(sssss0 | <T, s0 + s(s0 + ss0)>) is low, showing that the program and the left hand side of the equation is useful for obtaining the right hand side.

By using the definition of derivational simplification, we have that, e.g., s(sss0 + s0) and s0 + s(s0 + ss0) are alternative representations for sssss0, because both can be derived into sssss0 and both are more complex than sssss0 (since G(sssss0 | <T, s(sss0 + s0)>) < G(s(sss0 + s0) | <T, sssss0>) and G(sssss0 | <T, s0 + s(s0 + ss0)>) < G(s0 + s(s0 + ss0) | <T, sssss0>).

Note that for the previous theory it is not strange to consider x_3 = ssss...45 times...sss0 the best representation for 45d, because addition (alone) does not provide any representational advantage.

Let us extend the previous theory with the equations of product:

$0 \times X = 0$

$sX \times Y = X \times Y + Y$

This new theory (let us denote it by T') has now 4 function symbols (0, s, +, ×). Consider these three terms:

x_1 = ((((s0 × ss0 + 0) × ss0 + s0) × ss0 + s0) × ss0 + 0) × ss0 + (s0),

x_2 = s0 + ss0 × (0 + ss0 × (s0 + ss0 × (s0 + ss0 × (0 + ss0 + s0)))), and

x_3 = ssss...*45 times*...sss0 both equal to 45d with respect to to T'.

We can obtain their optimalities:

LT(x_1 | T') = 35 · (log 4) + log Cost(x_1 | T') = 70 + log 1340 = 80.4

LT(x_2 | T') = 35 · (log 4) + log Cost(x_1 | T') = 70 + log 324 = 78.3

LT(x_3 | T') = 46 · (log 4) = 92

The best representation is x_2, and x_3 turns out to be the worst, which agrees with the usual representation of numbers in most developed cultures [Ifrah 1994]. And this holds as T' is not only better than T for representing the natural number 45 but the complete set of natural numbers and other arithmetical evidence.

Finally, as we saw in the preceding chapter, we can compute the representation enhancements between these representations. For this, we must only find $Kt(y)$, where y is the absolutely best representation, which in this case can be a simplification of x_2, more concretely y= s0 + ss0 × (ss0 × (s0 + ss0 × (s0 + ss0 × (sss0)))) with LT(x'_2) = 29 · (log 4) + log 240= 65.9.

Then, the representation enhancements with respect to y are $RE(x_2,x_1) = (\mathrm{LT}(x_1) - \mathrm{LT}(x_2)) / Kt(y) = 0.03$, $RE(x_2,x_3) = (\mathrm{LT}(x_3) - \mathrm{LT}(x_2)) / Kt(y) = 0.20$ and $RE(x_1,x_3) = (\mathrm{LT}(x_3) - \mathrm{LT}(x_1)) / Kt(y) = 0.18$, and $RE(y,x_1) = (\mathrm{LT}(x_1) - \mathrm{LT}(y)) / Kt(y) = (80.4 - 65.9) / 65.9 = 0.22$.

4.7.2 Theory Optimisation for Multiple Evidence

We have seen the optimality of a single representation. Let us begin to explore if this notion can be extended to a set of consequences or evidence E.

Given an evidence E, we are concerned with finding a theory T, such that E is covered optimally by T. Since E is a set, we can weigh each element of E in some other way than uniformly (we will get on this problem in chapter 6). Moreover, we must select once again the criterion of optimality. Finally, we must distinguish between a deductive system (DS), a theorem prover (TP) and an accepter (AC), or, in other words, the theory should behave reasonably well for obtaining deductive inferences, obtaining proofs or obtaining the truth value of formulae, respectively.

If we parameterise the optimality criteria (or the effort criteria), and we want a uniform measure for all the evidence, we can measure the optimality of T with respect to E as:

$\mathrm{opt}_{\mathrm{DS}}(T| E) = argmin_T(\Sigma_{e\in E} \mathrm{Effort}(e|T))$.

$\mathrm{opt}_{\mathrm{TP}}(T| E) = argmin_T(\Sigma_{e\in E} \mathrm{Effort}(w|<T,e>))$.

$\mathrm{opt}_{\mathrm{AC}}(T| E) = argmin_T(\Sigma_{e\in E} \mathrm{Effort}(a|<T,e>))$.

for deductive systems, theorem provers and accepters, respectively. Or maybe we want to measure the best theory for these three purposes:

$\mathrm{opt}(T| E) = argmin_T(\Sigma_{e\in E} \alpha\cdot\mathrm{Effort}(e|T) + \beta\cdot\mathrm{Effort}(w|<T,e>) + \gamma\cdot\mathrm{Effort}(a|<T,e>))$.

As we said before, if the evidence is infinite, a finite sample should be randomly extracted from it in order to be able to compute the previous measures.

The question is now centred in selecting the criterion. As we saw, G and V are measures of effort, and, in the cases of LT and L, the goal would be to minimise G and V respectively. However, LT and L are not the only measurements of effort. For instance, if the criterion is efficiency of inference, the best theory for deductive inference is given by $\mathrm{opt}_{\mathrm{DS}}(T| E) = argmin_T(\Sigma_{e\in E} Cost(e|T))$. It is important to note that even in this case of unlimited memory, the best system for E usually is not E itself (apart from the case where E is infinite because an extensional description does not exist). For instance, given the evidence "p(2), p(4), p(6),...., p(100000)," it is better to have "p(X) : - even(X), X < 100000" than to search among the 100000 elements, having a cost of about 100000/2 = 50000 mean time access.

On the contrary, if the criterion is the size of describing the evidence we have that $\text{opt}_{DS}(T|E) = argmin_T(\Sigma_{e\in E} L(e|T))$. For a logical program we could use $\text{opt}_{DS}(T|E) = argmin_T(L_{PC}(E|T))$ or $argmin_T(L_{MC}(E|T))$ instead.

Most of these variants have a direct utility for different situations:

- The first measure, $\text{opt}_{DS}(T|E) = argmin_T(\Sigma_{e\in E} Cost(e|T))$ is applicable to solvers and programs, because the necessary w can be seen as the input and e as the output, and it is required that his process would be efficient. However, the length of the theory should also be reduced, thus a reasonable $L(T)$ must be maintained. This dilemma is habitual when using partial evaluation [Alpuente et al. 1998] and transformation techniques [Pettorossi and Proietti 1996b], where specialised theories, which exhibit low computational cost to some part of their consequences, may be quite large.
- If we measure the descriptional complexity, i.e., $\text{opt}_{DS}(T|E) = argmin_T(\Sigma_{e\in E} L(e|T))$, it is applicable to Conceptual Theories or Scientific Models, where the evidence should be described with the help of the theory. In order to avoid very complex theories, it would be better to use $\text{opt}_{DS}(T|E) = argmin_T(\Sigma_{e\in E} LT(e|T))$ instead. This extends the previous notion of representation simplification to that of theory simplification from T to T', if $\Sigma_{e\in E} LT(e|T')$ is less than $\Sigma_{e\in E} LT(e|T)$.
- If we measure the cost of obtaining proofs, i.e., $\text{opt}_{PS}(T|E) = argmin_T(\Sigma_{e\in E} Cost(w|<T,e>))$ we can apply it especially to Automated Theorem Provers (ATP) where different benchmarks for set of characteristic proofs are applied in order to discern which system is better than others [Suttner and Sutcliffe 1996].
- In mathematics, though, we are not interested in the time a proof takes to be discovered (except some famous theorems, such as Fermat's). Here $\text{opt}_{PS}(T|E) = argmin_T(\Sigma_{e\in E} L(w|<T,e>))$ favours shorter proofs, which are something very valued in mathematical systems. Even in ATP, as Wos pointed out (Wos 1996), the 'elegance' of a proof is important, according to three criteria (length, structure and compactness). In many cases, we introduce lemmata or intermediate information in order to shorten the proofs although the system size increases.
- Specialised to accepter systems, $\text{opt}_{AC}(T|E) = argmin_T(\Sigma_{e\in E} Cost(a|<T,e>))$ is applicable for time optimisation of digital circuits and, if a is not Boolean, to classifier systems.
- Finally, $\text{opt}_{AC}(T|E) = argmin_T(\Sigma_{e\in E} L(a|<T,e>))$ is not sensible, and it must be replaced by $argmin_T(LT(A|<T, E>))$ being A an array giving the answers to each e in E. This, finally, turns out to be very similar to the previous measure.

4.7.3 Pietarinen's Systematic Power

In the previous subsection we have seen that many of the theory optimality measures were made in order to minimise G and V of the evidence with respect to the theory. Let us see how a similar outcome was obtained from a different departure, based on Popper's idea of explanatory power.

Juhani Pietarinen, in his paper "Quantitative Tools for Evaluating Scientific Systematizations" [Pietarinen 1970] introduces a measure (and four different variants of it) to account for the explanatory capacity, namely systematic power, of hypotheses with respect to certain determinate data.

The idea is inspired in Popper's measure of explanatory power [Popper 1959], namely

$$E(h, d) = \{ \mathrm{p}(d \mid h) - \mathrm{p}(d) \} \;/\; \{ \mathrm{p}(d \mid h) + \mathrm{p}(d) \}$$

where h represents the hypothesis and d the data. Popper's measure is modified and generalised by Pietarinen:

$$\mathrm{syst}(h, d) = \{ \mathrm{unc}(d) - \mathrm{unc}(d \mid h) \} / \mathrm{unc}(d)$$

By using four different measures of relative uncertainty ($\mathrm{unc}(d \mid h)$) and absolute uncertainty ($\mathrm{unc}(d)$), all of them based on probability, Pietarinen gives four different measures of systematic power, studying their specific properties.

Since the four measures had the advantages and drawbacks of being based in probabilities, it would be interesting to introduce a fifth and a sixth variant, by using the descriptional version of uncertainty, namely, relative complexity $K(d \mid h)$ and absolute complexity $K(d)$ and their corresponding Kt versions.

Definition 4.30 Time-Ignoring Descriptional Systematic Power

$$\mathrm{syst}_5(h, d) = \{ K(d) - K(d \mid h) \} / K(d)$$

Correspondingly, we define

Definition 4.31 Space-Time Descriptional Systematic Power

$$\mathrm{syst}_6(h, d) = \{ Kt(d) - Kt(d \mid h) \} / Kt(d)$$

However, both definitions highly resemble previous notions seen in this chapter, as the following theorem shows.

Theorem 4.15 $\mathbf{syst_5(\boldsymbol{h}, \boldsymbol{d}) = 1 - \boldsymbol{V}(\boldsymbol{d} \mid \boldsymbol{h})}$ and $\mathbf{syst_6(\boldsymbol{h}, \boldsymbol{d}) = 1 - \boldsymbol{G}(\boldsymbol{d} \mid \boldsymbol{h})}$

In other words, under this view of uncertainty, the more powerful a system is, the less the gain of the evidence with respect to the theory. This is equal to descriptional Gain Ratio as it was derived from Quinlan's Gain Ratio in Section 4.4, if H and D are swapped, i.e. $1 - V(X|C)$ where H corresponds to the attribute X and D correspond to the class C. Although the interpretation is quite different in both cases, a measure $1 - G$ indicates that a theory or a tree split must be invested in order to minimise subsequent effort (value of G).

The measure $syst_6$ is intuitive and applicable to the different kinds of deductive systems we have just seen previous subsections. For instance, a system where its deductions are hard to obtain, i.e., $V(E|T)$ and/or $G(E|T)$ are high, is difficult to test, because it is hard to extract new evidences. This matches with Popper's relative potential satisfactoriness, or potential progressiveness, of scientific theories. This criterion "*characterizes as preferable the theory which tells us more: that is to say, the theory which contains the greater amount of empirical information or content: which is logically stronger: which has the greater explanatory and predictive power; which can therefore be more severely tested by comparing predicted facts with observations*" [Popper 1962].

On the contrary, $V(W|<T, E>)$ and $G(W|<T, E>)$ can be used to define the systematic power of a mathematical systems, where proofs are required to be short, given the theory and the theorems to prove.

4.8 A Characterisation of Lazy and Eager Inference Methods

In chapter 2 we discussed two kinds of inductive inference methods: lazy methods such as Explanation-Based Learning (EBL) and Case-Based Reasoning (CBR) and eager (or inductive) methods, such as Model Based Reasoning (MBR) or Inductive Logic Programming (ILP). The difference lays in the time where the inference effort is made. For the case of lazy methods, reasoning is triggered whenever a new problem or evidence appears, and all the history of previous cases is reviewed. This is the reason why they are also called memory-based methods. On the contrary, eager methods construct a model of the problem as this is fed, and when a new problem case appears, the solution comes quickly by the application of the model. Since back cases are no longer necessary if the model is reliable, they can be forgotten. This is the reason why these methods are also called forgetful methods. See [López de Mántaras and Armengol 1998] for an up-to-date state of the art of both methods.

Nonetheless, the difference between lazy and eager methods has always been discussed in an informal way. A theoretical account of this distinction would help to establish for which kind of problems each type of method would be more appropriate. Moreover, their integration could be studied in a much more general way than some fruitful but particular approaches [Armengol and Plaza 1994].

This formalisation can be made by the use of concepts related with information gain. Consider the evidence as an ordered (indexed) set of examples: $E_n = \{ e_1, e_2, ..., e_n \}$. Each example is a pair (x_i, y_i) where x is the input and y is the output. A Boolean evidence can always be transformed by restricting $y \in \{ \text{False, True} \}$. Consider a method or algorithm A that, for all $m < n$, if given the first m examples and the input x_{m+1} predicts (correctly or not) the example $m+1$, i.e., y_{m+1}.

Let us define $Time(A, i)$ = the time from the moment that the algorithm is given the next input x_i till the moment it gives the value y_i, i.e., the reaction time. This time includes any process that the algorithm performs, including theory revision. In other words, in the case that y_i is wrongly predicted, the algorithm may perform some operations to remake its model (if it has it) and this must be included in the measure (which would be in this case a reaction and reflection time). From here the following definition is straightforward:

Definition 4.32 Time Laziness

$$\text{Laziness}^t(A) = \lim_{i\to\infty} \{ [\textstyle\sum_{j \leq i} Time\ (A,j)]/\ i \}$$

Obviously, in the case of good eager methods, their laziness would usually be low (it would be always low if we do not incorporate the revision time but this would not be fair). In the best case, when the algorithm identifies in a certain i the correct model M of the evidence, then the time would be very reduced for the rest of the evidence, since it would only be required to apply the model M, and $\text{Laziness}^t(A) \in O(\text{Cost}(M))$. On the contrary a lazy algorithm is constantly comparing with past examples and the time cost will be more and more expensive because there will be more and more past examples to compare, up to the limitation of memory resources.

Accordingly, it is not only time but space which mostly distinguishes lazy and eager methods. As we have seen, an eager algorithm forgets the evidence whereas a lazy method must store a great portion of it in order to make the comparisons.

Let us define $Space(A, i)$ = the space in bits that is required to store the necessary evidence and model for the algorithm to operate.

Definition 4.33 Space Laziness

$$\text{Laziness}^s(A) = \lim_{i\to\infty} \{ [\textstyle\sum_{j \leq i} Space(A,j)]/\ i \}$$

The space laziness of an eager method will depend on a model but it is again trivial that this would always be $l(M)$, i.e. a constant. On the contrary, the space laziness of a lazy method will, in general, be equal to its memory capacity.[25]

We can combine in a single measure both time laziness and space laziness.

Definition 4.34 Time-Space Laziness

$$\text{Laziness}(A) = \log_\tau(\text{Laziness}^t(A)) + \sigma \cdot \text{Laziness}^s(A)$$

In general, if idle times can be used, inductive methods would be better (in the large) for real-time applications. However, in most cases, a combination of lazy and eager methods could be the best solution (such as [Armengol and Plaza 1994]). The

[25] Note that the question for approximate algorithms is different, because the model is always an approximation and more and more space will be needed to obtain better precision.

previous measure allows that a joint value of laziness can be computed from the separated algorithms, provided their separated resources and the frequencies of use of each of them are known.

It is important to highlight that these measures are valid for induction, abduction, analogy, and deduction (consider input-output pairs as evidence).

For abduction (EBL) and analogy (CBR) the result would be a high value of laziness. For constructive induction (MBR and ILP), a low value of laziness is expected. Finally, for deduction the result can be lazy (such as knowledge-based systems) or eager (mathematics and software). Specialisation and transformation techniques are used for software programs [Pettorossi and Proietti 1990, 1996a, 1996b] [Dershowitz and Reddy 1992], in order to make them more eager, in the sense of $\text{Laziness}'(A)$.

The notion of eagerness and the oblivion criterion seen in section 4.2. can be used together in order to optimise resources. Obviously, accuracy should not be significantly affected by this.

4.9 Induction, Deduction and Information

In the scientific method, there is a traditional view that inductive processes are primarily responsible of providing information to our knowledge, whereas deductive processes reorganise them. We have seen that deductive processes also provide information by this reorganisation, comparison and collation of previously induced information. Knowledge acquisition and revision are thus continuous and highly inter-related processes.

The classical view of mathematics, on the contrary, does not change its truths frequently, but gives more relevance to some theorems, systems and methods, depending to their applicability to other sciences or other parts of mathematics itself. Axiomatic changes are motivated by these changes of interest, the addition of new axioms or the discovering of new relationships or theorems. Recently, there is a an increasing trend towards the view of mathematics as an experimental science [Tymoczko 1986], which sees itself not so different to other experimental sciences.

Induction, deduction and information have been difficult to conciliate in the literature. In the first chapter we commented on many paradoxes of the view of induction as an inverse process of deduction. In this chapter, we have seen that both deduction and induction are either informative or non-informative processes depending on $G(x \mid y)$, y being the data and x being the inferred result (an inductive hypothesis or a deductive derivation). This contrasts, as we saw in the initial part of this dissertation, with the traditional idea of induction as always information increasing inference process and deduction as always information decreasing inference process [Bar-Hillel and Carnap 1953].

Before, we have talked about the best theory for a given evidence (system optimisation), but without considering prediction purposes (i.e. plausibility). If we want to join both things we are forced to study first the relation of induction and deduction under information gain and transformation. This implies the comparison between $G(T \mid E)$, which represents the inductive gain, and $G(E \mid T)$, which represents the effort of obtaining the evidence from the theory and, thus, its testability and practicality. Following the asymptotic relationship $Kt(T \mid E) + Kt(T) = Kt(E \mid T) + Kt(E)$, a compromise between high inductive gain, plausibility and low deductive gain should be found. In order to obtain this, we have that $Kt(T)$ should be low and $Kt(E)$ should be high, which suggests that the theory should be much simpler (in LT terms[26]) than the evidence. This may suggest the use of a LT-modified MDL principle, for obtaining highly compressed and efficient theories.

However, we can recall one of the main problems of combining the MDL principle with deduction. Probabilities do not behave well under deduction transformations. Usually, if we have $p = x \vee y$, the deduction $x \vee y \vee \neg y$ should have more probability, according to Carnap, but the MDL principle assigns it less probability because it is longer. A probabilistic account based on an information-theoretic framework is not possible since $x \wedge y$ is as easy to construct as $x \vee y$ but their semantics are completely different.

4.9.1 Intermediate Information

Contrarily, information gain measures the internal increase of information and allows the distinction of which concepts are valuable and usable from an internal point of view. However, this is not sufficient as Holland et al. points out: "*One of the most impressive research efforts on induction, that of Lenat (1983), has yielded programs for generating concepts and heuristics for mathematics and other domains. However, the programs all encounter the problem of "mud", which is Lenat's informal designation for uninteresting definitions and tasks. Mud sooner or later accumulates to the point that a system becomes totally involved in a round of tasks that contribute nothing to the expansion of concepts and heuristics.*" [Holland et al. 1989].

In section 4.7 we have talked about partial evaluation techniques, lemmata, and other forms of intermediate information that make a system better with respect to some evidence. For instance, that addition is commutative ($X + Y = Y + X$) is a property that allows to shorten many derivations in arithmetic. However, the property ($X + X + 3 = (X + 1) \cdot 2 + 1$) is not so useful in arithmetic. Nonetheless, both are true from Peano's axioms, i.e., both are *theorems*. Moreover, both have a high value for G, but only one of them is worthy to maintain explicitly. In other words, it has been shown that for the common evidences arithmetic deals with, commutativity is extraordinary useful to memorise as an *incarnate* theorem whereas the other should be deduced if ever is needed. Similarly, lemmata are a special kind of theorems (also

[26] Note that this would be impossible for K, since $K(T) \geq K(E)$.

with high G) which has no practical use for E, but it is useful for the proof of a theorem.

Consider the same equational theory T for addition and product as before:

$$X + s(Y) = s(X + Y)$$

$$X + 0 = X$$

$$0 \times X = 0$$

$$sX \times Y = X \times Y + Y$$

It can be proven from here that $X + Y = Y + X$ and $Y \times X = X \times Y$. Since the proofs are not short, their G is high. The question is whether these theorems should be maintained in T or they should be removed because they are redundant (omnisciently non-informative). Moreover, they can give problems of endless loops.

Recalling $x_1 = ((((s0 \times ss0 + 0) \times ss0 + s0) \times ss0 + s0) \times ss0 + 0) \times ss0 + (s0)$, and $x_2 = s0 + ss0 \times (0 + ss0 \times (s0 + ss0 \times (s0 + ss0 \times (0 + ss0 + s0))))$, we have that using both properties properly we have that $LT(x_1) \approx LT(x_2)$ because now the proof of x_1 is much shorter. This is expectable for many other evidences and, consequently, $opt_{DS}(T| E) = argmin_T(\Sigma_{e \in E} Cost(e|T))$ should increase for an arithmetic solver constructed with commutativity of addition and product. One may not preserve any property, though, because this takes space, or may also take time because there are more possibilities to essay, and not every property is equally useful. The oblivion criterion can be adapted to select which set of deductive inferences are maintainable and which are not, depending on the effort of obtaining them and their utility (aside from their plausibility).

In the next chapter, we will centre on distinguish and effectively measuring why $(X + Y = Y + X)$ is a property *reinforced* by the evidence and the property $(X + X + 3 = (X + 1) \cdot 2 + 1)$ is not *reinforced*. We will see that reinforcement is what is lacking, both a measure of plausibility and utility.

4.9.2 Resource-Bounded and Fallible Inference

Classical and Mathematical logic are omniscient. A logic formula is a theorem in a system or not, independently of how many steps are necessary or how difficult is to obtain them. The possible worlds semantics and Kripke's semantics [Kripke 1963] triggered the research over different logics of believe and modal logics. However, an agent which is modelled by this semantics is logically omniscient, because they must believe any classic tautology, and perfect reasoners, because they must believe every classical logical consequence of its believes. This is not a realistic model of actual agents (either human or computational) because these always suffer a resource limitation, which makes impossible to become ideal reasoners. Maybe the major problem of Artificial Intelligence has been to neglect this difference during too time.

The axioms of modal logics were essayed for each of their combinations, these axioms being the following [Konolige 1992], as were introduced in chapter 2:

$$(K): L(\phi \supset \psi) \supset (L\phi \supset L\psi)$$
$$(D): L\phi \supset \neg L\neg\phi$$
$$(T): L\phi \supset \phi$$
$$(4): L\phi \supset LL\phi$$
$$(5): \neg L\phi \supset L\neg L\phi$$
$$(P): \phi \supset L\phi$$

Their names depend on the authors but generally (*K*) is known as the axiom of deduction, (*D*) is the non-contradiction axiom, (*T*) is the axiom of infallibility, (*4*) is the axiom of the conscience of own knowledge, (*5*) is the axiom of the conscience of ignorance and (*P*) is the axiom of complete wisdom.

Some of these axioms are related (for instance *T* and *K* imply *D*) and some combinations have intuitive interpretations (see e.g. the discussion about the different uses of S5, KD45, K45 in [Halpern 1997]).

For the use of inference processes such as induction and deduction, we are precisely interested in *K*, *D* and *T*:

- Ontological Fallibility ($\neg T$): An agent whose knowledge cannot be false is not realistic, even more when we consider learning agents, whose knowledge is empirical and, consequently, refutable. More clearly, induction would be nonsensical if the axiom *T* holds, because inductive inference is necessarily associated with error.
- Knowledge Extensibility ($\neg K$): if this axiom is assumed, all the logical consequences of a belief are also believed, i.e., every deep knowledge is made shallow automatically (omniscience). Apart from the fact that this in only possible for reduced representations, this would make that there would be no difference between depth information and surface information in the sense of [Hintikka 1970b]. On the contrary, if *K* is not assumed, deduction is more related with ontology and deductive reasoning is useful to make explicit what was implicit. Axioms 4 and 5 could also give raise to different degrees of extensibility.
- Reasoning Fallibility ($\neg D$): Since agents are based in languages with well-defined semantics, it is not usually considered that the system could have internal inconsistencies. But as well as *T* and *K* imply *D*, it is reasonable to accept that if the system has ontological fallibility and its reasoning abilities are not immediately omniscient., it is quite possible that an inconsistency of two separate *inductive* theories could persist during a time until it is detected by deductive reasoning. It is important to highlight that this makes deduction even more useful. Finally, this is not equal to the possibility of wrong deductions, which, for small applications, need not to be considered, but a

complex cognitive system should take that possibility into account for the sake of robustness, because any internal failure, could make the system collapse.

A fourth property, which only appears from a dynamical point of view, is the possibility of oblivion, which has been discussed previously. However, oblivion is much easier to model in the case of knowledge extensibility, because if *K* is assumed, forgetting something implies forgetting everything that was implied by it.

Let us summarise in a table, the six reasonable combinations of these axioms and the influence of these combinations for inductive and deductive inference:

Type of Knowledge	Omniscient (K)	Extensible (¬K)
Infallible (T and D)	Induction Nonsensical, Deduction Useless	Induction Nonsensical
Fallible without Contradiction (¬T and D)	Deduction Useless	Induction and Deduction useful and compatible (Horn theories, ILP)
Fallible with Contradiction (¬T and ¬D)	Contradictions are immediately detected	Both Induction and Deduction affect positively to ontology

Figure 4.4. *Combinations of modal axioms T, K, and D and influence to inference processes.*

For expressible languages, only the combinations (¬K, ¬T and D) and (¬K, ¬T and ¬D) are possible. The last combination, (concretely ¬T and ¬D), allows two kinds of refutations: inductive and deductive. The first one is the most classical one in the literature: a new fact is inconsistent with the theory and it must be revised. The second one is only possible if D is not assumed, which means that inconsistencies are not detected immediately. Two inconsistent inductive theories can coexist a time in a system until a deductive inference makes their contradiction explicit. In this case, deduction also affects plausibility, because the better deductive inference works, the surer the system that internal contradictions do not exist, and its predictions would have more plausibility. Both kinds of refutation do not mean that the theories must be necessary withdrawn, but they still can be used if they are practical to cover the evidence. This is usual for scientific theories, which can coexist even when are formally inconsistent (e.g. Relativity Theory and Quantum Theory).

The mechanism to resolve or work with inconsistent knowledge is open to different applications. The most usual approaches are based on priorities, confidences or credits, as we will use in the next chapter. Other approaches are more semantical such as Nute's *defeasible logic* [Nute 1988][Nute 1994], which opts for the strongest inference step whenever a contradiction takes place, weighing the deductive derivations performed.

Some authors think that weakening the standard epistemic systems results in many intuitions about the concepts of knowledge and belief being lost [Duc 1997]. The reason of this unintuitive results may be due to the possible world semantics [Moore 1984]. Duc's solution is based in a logical 'dynamisation' of epistemic logic inspired in dynamic logic [Harel 1984]: "*if we say that the epistemic agent knows the laws of logic, we do not mean that she knows some facts of the world, but rather that she is able to use these laws to draw conclusions from what she already knows. The laws of logic are what the agent knows implicitly; she does not need to possess them permanently. It suffices if she can recall them when she needs them in order to infer new information from her explicit data base*" [Duc 1997]. This forces a re-understanding of modal epistemic logic: "Modal epistemic logics should be interpreted as logics of possible, or implicit knowledge, and not as logics of actual, or explicit knowledge". Then, the clarification of the words implicit and explicit, as it has been done in these two chapters, is also of the greatest relevance for modal epistemic logic.

Another non-omniscient proposal is the idea of "conceivable worlds" [Moreno and Sales 1997]. Instead of considering all the possible worlds, the agent can only consider 'conceivable worlds': "*A conceivable situation is any state that the agent can consider, independently from its possible partiality or inconsistency*" [Moreno and Sales 1997]. Moreover, this kind of agents "are continually analysing their beliefs, in order to make them more resembling with real world". The processes for this analysis to remove inaccurate beliefs and refine true ones, dubbed "rational investigation", is characterised by these components [Moreno and Sales 1997]:

- Logical Analysis: the agent could make (limited) deductive inferences.
- Exploratory Analysis: the agent could raise questions and pose doubts such as whether it implicitly believes or not some fact.
- Experimental Analysis: the agent could ask for data from the environment to confirm or refute some of its beliefs.
- Knowledge Acquisition: the agent could incorporate to its beliefs the information that is received from the environment.

According to this view of bounded rationality, it is necessary a measure of plausibility would not always negatively affected by the addition of new intermediate information to the theory.

On the contrary, we look for a measure where positive deductive results should increase the reliability of the theory, and hence its plausibility, and negative deductive results should decrease the reliability of the theory. This kind of measure is endeavoured in the following chapter.

4.10 Summary and Contributions of This Chapter

This chapter has taken advantage of the definitions introduced in the previous chapter for inference processes, mainly induction and deduction. Both processes have been expressed in terms of computation and not in terms of truth, and we have also neglected any probabilistic account. Different kinds of deductive systems are particularised: Derivers (DS), Theorem Provers (TP) and Accepters (AC). Without truth or probability considerations it has been still possible to account for many phenomena that inference processes deal with. Information Gain, namely G, is both useful to explain the informativeness of a hypothesis with respect to some evidence and to explain the gain or reduction of effort that takes place when a conclusion or theorem is deductively established from an axiomatic system.

Section 4.2 concretises this idea for induction and recognises that if $G(h \mid e) \cong 0$, the theory is evident for the data. On the contrary, $G(h \mid e) \cong 1$, an informative induction has taken place, according to Popper's informativeness. Induction is then seen as an investment. However, this must be somehow restricted, and an oblivion criterion is introduced to weigh valuable and plausible hypotheses.

In Section 4.3, G is compared with the idea of learning as compression and the MDL principle. In the previous chapter we saw that efficient learners are shown to be non-informative and in this section it is shown that compression favours gain. However, the MDL principle does not ensure compression for most evidence, the vast majority of them being incompressible. Gain is a much more robust measure and with its help, notions such as creativity and scientific discovery are clarified, and the classical view of learning as identification [Gold 1967] is neglected.

Section 4.4 revises Quinlan's Gain Ratio, which is part of C4.5, the most famous machine learning algorithm. Gain Ratio Measures the value that a split on an attribute X has, when learning a decision tree for class C. The definition is adapted to descriptional complexity and it is shown that descriptional gain ratio = $1 - V(X \mid C)$.

Deduction is addressed in Section 4.5. $G(c \mid p) \cong 0$ if the conclusion c is evident from the premises p. On the contrary, $G(c \mid p) \cong 1$ when the conclusion is difficult and surprising from the premises p. This idea is particularised for the different deductive paradigms presented in the introduction. They are approximated for logic programs, sometimes adapting classical measures in the LP or ILP literature. Finally, an example illustrates these measures.

Section 4.6 compares our approach with Hintikka's Surface and Depth Information. Surface Information is identified with G and Depth Information with V. We prove that V is the limit of G in the sense of Hintikka.

The notions of representational optimisation from the previous chapter are taken up again and generalised in Section 4.7, by finding a compromise of size and time that it takes to obtain the evidence from a theory. This is particularised for different deductive paradigms, showing the role of Intermediate Information in ATP and mathematical practice. In the end, Pietarinen's Systematic Power suggests two different descriptional variants of systematic power as $1 - V(d \mid h)$ and $1 - G(d \mid h)$, corresponding with Popper's relative potential satisfactoriness.

Section 4.8 formalises the distinction between eager and lazy inference methods, by considering the time when the computational effort is made and / or how space resources are used.

Section 4.9 finally undertakes the relation among Induction, Deduction and Information, without forgetting plausibility. Classical induction criteria are incompatible with intermediate deductive information. It is discussed that it is only possible to conciliate them under resource-bounded rationality, avoiding omniscience and allowing inconsistencies in knowledge. The first steps towards the theory of reinforcement of the following chapter are taken.

The main contributions of this chapter are:

- It has been shown that a single measure of information gain can be applied to both deduction and induction in a uniform way.
- Popper's idea of informativeness is gathered by the use of G for induction.
- A new notion of authentic learning is introduced, ensuring that learning has taken place, independently of how compressible the evidence is.
- Quinlan's Gain Ratio is closely connected with V.
- Deduction can be informative and different measures are introduced for several deductive paradigms.
- Appropriate approximations for logical programs are derived and illustrated, which make possible to measure in practice these gains.
- Comparison with Hintikka's ideas, establishing the relationship between G and Surface Information, and between V and Depth Information.
- General measures of System Optimisation and Systematic Power, where Intermediate Information is recognised useful in ATP and mathematical practice.
- A formal account of the notion of lazy and eager methods, according to response time and necessary memory resources.
- The conciliation among induction, deduction and information is made possible if omniscience is neglected. However, when omniscience and infallibility are neglected, the semantic tools are weakened. Consequently, other mechanisms are needed to guide a system's ontology.

The next chapter introduces a theory that makes possible and practical this conciliation.

5. Constructive Reinforcement

Custom then, is the great guide of human life
David Hume, 1711-1775, An Enquiry Concerning Human Understanding 1748

Abstract: *this chapter presents an operative measure of reinforcement for general constructive theories as a quantitative theory of confirmation, studying the growth of knowledge, theory revision, abduction and deduction in this framework. This approach performs an apportionment of credit with respect to the 'course' that the evidence or set of derivables makes through the learnt/axiomatic theory. For the case of induction it is shown to be both a utility and plausibility criterion, and it is connected with other classical evaluation criteria, such as cross-validation and the MDL principle. For the case of deduction it behaves like a utility criterion that establishes how useful a property, lemma or theorem is for the rest of the theory. It is also applied to other inference mechanisms, such as analogy, abduction and explanatory induction, the latter represented by a balanced distribution of reinforcement, thus formalising the notion of consilience. The theory is also extended with negative reinforcement, thus connecting this approach with more classical notions of reinforcement, based on rewards and penalties. In the end, reinforcement and information gain are compared.*

Keywords: Reinforcement Learning, Incremental Learning, Useful Theorems, Inference Processes, Apportion of Credit, Knowledge Acquisition and Revision, Consilience, Analogy, Theory Evaluation.

5.1 Introduction

In the previous chapter we finally arrived to the necessity and possibility of finding a measure for conciliating induction, deduction, information and plausibility. Under this measure, a positive deductive operation that connects two unrelated things should also increase the reliability of the theory. On the contrary, negative connections originated from internal or external inconsistencies should decrease its reliability. Moreover, this measure should comply with plausibility, not differing too much with some of the usual selection criteria in machine learning or inductive inference, and should favour informative theories without compromising tractability and applicability. At first sight this seems rather pretentious. However, this chapter introduces an effective way to evaluate a system with respect to a given evidence, where induction and deduction behave properly and not contradictorily.

In the case of deduction, the quality, robustness or reliability of a system is given by how many connections can be established among their theorems and their final evidence. In Kneale's words: a "*system is interesting mathematically if it is rich in theorems and has many connections with other parts of mathematics*" (from [Lakatos 1979]). In other words, concepts and theorems with high connectivity are useful, ordinarily known as properties.

In the case of induction, it has been shown that high connectivity is also positive with induced theories. In fact, many inductive algorithms are based on the idea of propagating reinforcement, as in RL (Reinforcement Learning) or ANN (Artificial Neural Networks). However, the representational language that is used in these inductive paradigms is poor and deduction is an ignored matter (i.e. deterministic and supposedly efficient).

The question is whether we can extend the notion of reinforcement to more expressive languages, where deduction is not so easy and may be informative.

The problem of propagating reinforcement from the evidence into the theory has been shown especially troublesome in high-level languages, such as ILP, but the same problem pervades other representations that allow redescription (e.g. neural networks).

In this chapter, we present an operative measure of reinforcement for general constructive theories, studying the growth of knowledge, theory revision, abduction and deduction in this framework. Our approach performs an apportionment of credit with respect to the 'course' that the evidence makes through the learnt theory. The result is compared with other evaluation criteria, in the case of induction, such as the MDL principle, and other utility criteria, in the case of deduction.

Finally, we will study a more common view of reinforcement, where the actions of an intelligent system can be rewarded or penalised, and we discuss whether this should affect the distribution of reinforcement.

In the end, we will relate the theory of reinforcement with deduction and axiomatic systems optimisation, and we will establish the connection with the notions and results of chapter 3 and 4: information gain, representation gain, maximisation of reinforcement of resource-bounded systems, etc.

5.1.1 Reinforcement as Selection Criterion

As we said in the second chapter, the aim of Machine Learning is the computational construction of hypothetical inferences from facts.

However, we saw that given some evidence *E*, infinite many hypotheses *H* can be induced ensuring $H \models E$. Obviously, some selection criteria are needed. Depending on different applications, some criteria have been used (e.g. the most specific hypothesis, the most general one, the shortest one, the most informative one, ...). In general, this choice implies the assumption of a prior distribution, which can be used to derive the likeliness of the hypotheses. The MDL principle is the most famous and used criterion. We also saw in chapter 2 many of its problems.

In this chapter, we intend to handle these difficulties with a dynamical reinforcement. However, our approach has additional advantages: (1) no prior assumption has to be made (apart from how to distribute this reinforcement, which is the topic of this chapter), and (2) reinforcement can be more flexibly managed than probabilities, and allows further insight on the relation between the evidence and the theory.

5.2 Reinforcement Learning

As we have seen, whatever the approach to knowledge construction, the revision of knowledge must come either from an inconsistency or from a lack of support. In the latter case, a partial or total weakness of the theory can be detected by a loss of *reinforcement* (or apportionment of credit [Holland et al. 1986]. There have been several empirical and theoretical justifications for reinforcement in different fields, from many empirical observations from on learning processes in animals or humans to theoretical and practical verifications by cross-validation.

The study of reinforcement learning in restricted representations has been especially fruitful in this decade (see [Kaelbling et al. 1996] for a survey) and it has been recently related with EBL (see [Dieterich and Flann 1997]). One of the main problems of reinforcement learning is that it is increasingly more difficult to assign and 'propagate' the reinforcement (or apportionment of credit [Holland et al. 1986]) depending on two factors (which are as well related): (1) how eager the inductive strategy is (vs. lazy methods such as instance-based and case-based reasoning [López de Mántaras and Armengol 1998]) and (2) how expressible the language where induction must take place is. Explanation Based Learning (EBL) and Inductive Logic Programming (ILP) are two areas where the propagation of reinforcement faces these issues in a more arduous way.

In this chapter we shall address the problem of reinforcement with eager learning methods. As we saw in the previous chapters, eager learning methods extract all the regularity from the data in order to work with intensional knowledge (instead of the extensional knowledge of lazy methods [Aha 1997]).

Additionally, we will consider the problem with constructive languages. A constructive language is a language that allows dynamical change of its representational bias (what is sometimes known as the possibility of 'redescription'), i.e., new constructed terms can be created to express the evidence more compactly in a more compact way. This is usually known in ILP as predicate invention.

In decision trees or attribute languages, no invented terms are induced and reinforcement is distributed among the initial attributes. The main drawback of these approaches is the lack of flexibility: when arrived to a 'saturation' point, the data is not abstracted further and the mean reinforcement arrives to a limit. Consequently, the ontology must be given and not constructed (a model of the 'world' is embedded in the system) and the possible extensions of this world are very restricted.

In the case of learning in highly expressible frameworks, a main problem is presented (apart from efficiency): the ontology of the new constructed concepts is indirect. The usual solution to this problem is the assumption of a prior probability. Once the probabilities are assigned, a Bayesian framework can be used to 'propagate' the distribution. In general, there is not justification at all of which prior distribution to choose. In the absence of any knowledge, as we saw in chapter 2, the most usual one is the MDL (Minimum Description Length) principle [Rissanen 1978, 1996]. The MDL principle is just a formalisation of Occam's razor. Theoretically, its close relation with PAC-learning [Valiant 1984] has been established by (Blumer et al. 1987). Some high-level representation inductive methods have adapted these ideas (e.g. U-learnability in ILP [Muggleton and Page 1995]). All of them are based on the assumption of a prior. However, there are many riddles with the management of probabilities and, in particular, the best choice, the MDL principle, has additional ones.

As we will see, most of these difficulties would disappear if no prior distribution is assumed and the knowledge is constructed by reinforcement, as the data suggest. However, the translation of these ideas to general representational frameworks seems difficult. First, the length of the structures which supposedly are to be reinforced is variable. Second, and more importantly, it seems we can always invent 'fantastic' concepts that can be used in the rest of knowledge. Consequently, these 'fantastic' concepts are highly reinforced, increasing the reinforcement ratio of knowledge in an unfair way.

An immediate way out is the combination of reinforcement learning with some prior, mainly the MDL principle, essayed under the name of 'incremental self-improvement' [Schmidhuber et al. 1997] using syntactic minimality to restrict the appearance of these inventions.

Notwithstanding, our approach also avoids 'fantastic' concepts but it is based exclusively on reinforcement. Consequently, compression turns out to be an 'a posteriori' consequence of a well-established reinforcement, instead of an 'arbitrary' assumption.

5.3 Reinforcement with respect to the Theory Use

For the study of reinforcement we need to introduce some basics for the kind of representation languages to which it can be applied. A 'pattern' of languages is defined as a set of *chunks* or rules *r* which are composed of a head (or consequence) and a body (or set of conditions). Each rule is denoted in the following way $r \equiv \{ h \text{ :- } t_1, t_2, \ldots t_s \}$. A theory is simply a set of rules: $T = \{r_1, r_2, \ldots, r_m\}$.

Since no restriction of how h and t_i can be (there may be variables, equations, Boolean operators...), this definition can be specialised to propositional languages, Horn theories, full logical theories, functional languages, some kind of grammars, and even higher-order languages. In the following, the semantics of the representations will be left unspecified and we will just say that e is a consequence of P, denoted $P \models e$ (in other words, there is a proof for e in P, or, simply, P covers e).

Given the slight semantical and syntactical restriction of the previous paragraphs, we introduce some useful and simple constructions which will shape our framework with more determination.

Definition 5.35 A rule r_i is said to be necessary with respect to T for an example e iff

$$T \models e \text{ and } T - \{r_i\} \not\models e$$

From here,

Definition 5.36 A theory T is reduced for an example e iff

$$T \models e \text{ and } \neg\exists\, r_i \in T \text{ such that } r_i \text{ is not necessary for } e$$

For the rest of the chapter, we consider a proof as a set of rules, independently of their order of combination, the applied substitutions or number of times that each rule is used. This unusual (and incomplete) conception of proof allows us to work without considering the concrete semantics while maintaining an appropriate degree of detail. This makes the following definition possible:

Definition 5.37 We say that S_1 and S_2 are alternative proofs for an example e in the theory T iff

$$S_1 \subset T,\ S_2 \subset T,\ S_1 \neq S_2 \text{ and } S_1 \text{ and } S_2 \text{ are reduced for } e$$

We denote with $Proof(e,T)$ the set of alternative proofs for an example e with respect to a theory T. Finally, we can define $Proof_r(e,T)$ as the set of alternative proofs which contain r. More formally,

Definition 5.38

$$Proof_r(e,T) = \{ S : S \subset Proof(e,T) \text{ and } r \in S \}$$

With these naive constructions, we are able to introduce our first measurement of reinforcement.

We present the first intuitive way to compute the reinforcement map for a given theory, depending on past observations.

Definition 5.39 The pure reinforcement $\rho\rho(r)$ of a rule r from a theory T with respect to a given evidence $E = \{e_1, e_2, \ldots, e_n\}$ is defined as:

$$\rho\rho(r) = \Sigma_{i=1..n}\, card(Proof_r(e_i,T))$$

In other words, $\rho\rho(r)$ is computed as the number of proofs of e_i where r is used. If there are more than one proof for a given e_i, *all* of them are reckoned, but in the same proof, a rule is computed only once.

Definition 5.40 The (normalised) reinforcement is defined as:

$$\rho(r) = 1 - 2^{-\rho\rho(r)}.$$

Definition 5.40 is motivated by the convenience of maintaining reinforcement between 0 and 1. However, its computation is easy, as the following elementary lemma shows:

Lemma 5.16 Suppose a new example is added to the evidence and it is covered by the theory. For each rule r that is used for it, the new $\rho'(r)$ can be easily obtained from the old $\rho(r)$ by:

$$\rho'(r) = [\, \rho(r) + 1 \,] / 2$$

PROOF. The new $\rho\rho'(r)$ is incremented by one, i.e. $\rho\rho'(r) = \rho\rho(r) + 1$. From here, $\rho'(r) = 1 - 2^{-\rho\rho'(r)} = 1 - 2^{-\rho\rho(r) - 1} = 1 - 2^{-\rho\rho(r)}/2 = ½ \cdot [2 - 2^{-\rho\rho(r)}] = ½ \cdot [\, 1 + 1 - 2^{-\rho\rho(r)}\,] = ½ \cdot [\, 1 + \rho(r)]$. □

Corollary 5.17 If an example is removed from the evidence, for each rule r that was used for it, the new $\rho'(r)$ can be easily obtained from the old $\rho(r)$ by:

$$\rho'(r) = 2 \cdot \rho(r) - 1$$

Hence, if a rule r covers a single example we have $\rho(r) = 0.5$ and if the rule becomes not necessary, then $\rho'(r) = 0$.

Definition 5.41 The mean reinforcement ratio $m\rho(T)$ is defined as

$$m\rho(T) = \Sigma_{r\in T}\, \rho(r)/m,$$

with m being the number of rules.

From these definitions one can verify that, in general, the most (*mean*) reinforced theory is not the shortest one as the following example shows:

Example 5.1

Given the evidence e_1, e_2, e_3, consider a theory $T_a = \{r_1, r_2, r_3\}$ where $\{r_1\}$ covers $\{e_1\}$, $\{r_2\}$ covers $\{e_2\}$ and $\{r_3\}$ covers $\{e_3\}$ and a theory $T_b = \{r_1, r_2, r_3, r_4\}$ where $\{r_1, r_4\}$ cover $\{e_1\}$, $\{ r_2, r_4 \}$ cover $\{e_2\}$ and $\{r_3, r_4 \}$ cover $\{e_3\}$.

From here, T_a is less reinforced than T_b.

In the first case we have $\rho\rho_{a,1}= \rho\rho_{a,2}= \rho\rho_{a,3}= 1$ and $m\rho(T_a) = 0.5$. For T_b we have $\rho\rho_{b,1}= \rho\rho_{b,2}= \rho\rho_{b,3}= 1$, $\rho\rho_{b,4}= 3$ and $m\rho(T_b) = 0.5938$.

In addition, redundancy does not imply a loss of mean reinforcement ratio (e.g. just add twice the same rule).

However, measuring reinforcement of the *theory* presents problems of *fantastic* (unreal) concepts:

Theorem 5.18 Consider a program P composed of rules r_i of the form $\{ h \text{ :- } t_1, t_2, .. \, t_s \}$, which covers n examples $E = \{ e_1, e_2, \ldots \; e_n \}$. If the mean reinforcement ratio $m\rho < 1 - 2^{-n}$ then it can always be increased.

PROOF. A *fantastic* rule r_f can be added to the program by modifying all the rules of the program in the following way $r_i = \{ h \text{ :- } t_1, t_2, .. \, t_s , r_f \}$. Obviously, all the other rules maintain the same reinforcement but r_f is now reinforced with $\rho\rho(r_f) = n$. Since $\rho(r_f) > m\rho$ then the new $m\rho'$ must be greater than $m\rho$. □

One can argue that these *fantastic* rules could be checked out and eliminated. However, there are many ways to 'hide' a fantastic rule; in fact, cryptography relies on this fact.

5.4 Reinforcement with respect to the Evidence

It can be derived from this problem that reinforcement must be combined with a simplicity criterion in order to work (maybe neural networks theory is the field where this avoidance of overfitting, ensured by simplicity, has been more thoroughly studied in combination with reinforcement).

However, there is solution without explicitly making use of simplicity. The idea is to measure the validation *with respect to the evidence.*

Definition 5.42 The course $\chi_T(f)$ of a given fact f with respect to a theory T is defined as:

$$\chi_T(f) = max_{S\subset Proof(f, T)} \{ \Pi_{r\in S}\, \rho(r) \}$$

More constructively, $\chi_T(f)$ is computed as the product of all the reinforcements $\rho(r)$ of all the rules r of T that are used in the proof of f. If a rule is used more than once, it is computed once. If f has more than one proof, we select the greatest course.

Whereas Definition 5.39 eased theory use, this must be restricted somehow, in order to find a compromise between interconnectivity and complexity. Due the multiplicative character of Definition 5.42, long proofs are also penalised, and this compromise is attained.

The way reinforcements are calculated makes very complex programs to be avoided, but redundancy is possible. However now there is no risk of fantastic concepts. As said before, for any program P composed of rules r_i of the form $\{ h$:- $t_1, t_2, .. t_s \}$, which covers m examples $E = \{ e_1, e_2, ... \ e_n \}$ and their reinforcements ρ_i, a *fantastic* rule r_f could be added to the program and all the rules could be modified in the following way $r_i = \{ h$:- $t_1, t_2, .. t_s , r_f \}$. The following theorem shows that now it is not reinforced over the original one:

Theorem 5.19 The course of any example cannot be increased by the use of *fantastic* concepts.

PROOF. Since the *fantastic* concept r_f now appears in all the proofs of the n examples, the reinforcement of r_f is exactly $1 - 2^{-n}$ and the reinforcements of all the r_i remain the same. Hence, the course of all the n examples is modified to $\chi'(e_j) = \chi(e_j) \cdot r_f = \chi(e_j) - \chi(e_j) \cdot 2^{-n}$. Since n is finite, for all $e_j \in E$, $\chi'(e_j)$ can never be greater than $\chi(e_j)$. □

5.5 Evaluation of Inductive Theories

Now it is time to start to use the previous measure to evaluate inductive theories. The first idea is to use the greatest *mean* of the courses of all the data presented so far, defined as:

Definition 5.43 The mean course $m\chi(T, E)$ of a theory T with respect to an evidence E is defined as:

$$m\chi(T, E) = \Sigma_{e\in E}\, \chi_T(e)/n$$

with $n = card(E)$.

In order to obtain a more compensated theory, a geometric mean can be used instead, which we will denote by $\mu\chi$.

For every theory T, we will say that it is *worthy* for E iff $m\chi(T, E) \geq 0.5$. If the representation language is expressible enough, it is easy to show that for every evidence E there is at least a theory worthy for it (just choose a theory with an extensional rule for covering each example). The same holds for $\mu\chi$.

5.5.1 Knowledge Construction, Revision and Abduction

The use of these simple values can be seen in the following long example, in order to show the use of this *new* criterion for knowledge construction:

Example 5.2

Using Horn theories as representation (Prolog), suppose we have an incremental learning session as follows:

⌧ Given the background theory B = { s(a,b), s(b,c), s(c,d) } we observe the evidence

E = { e^+_1: r(a,b,c), e^+_2: r(b,c,d), e^+_3: r(a,c,d), e^-_1: ¬r(b,a,c), e^-_2: ¬r(c,a,c) }:

The following programs could be induced, with their corresponding reinforcements and courses:

P_1 = {r(X,Y,Z) :- s(Y,Z) : ρ = 0.875}

$\chi(e^+_1)= \chi(e^+_2)= \chi(e^+_3)= 0.875$

P_2 = {r(X,c,Z) : ρ = 0.75
r(a,Y,Z) : ρ = 0.75}

$\chi(e^+_1)= \chi(e^+_2)= \chi(e^+_3)= 0.75$

P_3 = {r(X,Y,Z) :- s(X,Y) : ρ = 0.75
r(X,Y,Z) :- s(Y,Z) : ρ = 0.875}

$\chi(e^+_1)= \chi(e^+_2)= \chi(e^+_3)= 0.875$

P_4 = {r(X,Y,Z) :- t(X,Y), t(Y,Z) : ρ = 0.875
t(X,Y) :- s(X,Y) : ρ = 0.875
t(X,Y) :- s(X,Z), t(Z,Y) : ρ = 0.5}

$\chi(e^+_1)= \chi(e^+_2)= 0.7656, \chi(e^+_3)= 0.3828$

P_5 = {r(X,Y,Z) :- t(X,Y) : ρ = 0.875
t(X,Y) :- s(X,Y) : ρ = 0.875
t(X,Y) :- s(X,Z), t(Z,Y) : ρ = 0.5}

$\chi(e^+_1)= \chi(e^+_2)= 0.7656, \chi(e^+_3)= 0.3828$

At this moment, P_1 and P_3 are the best options and P_4 and P_5 seem ‘risky’ theories according to the evidence.

⌧ e^+_4 = r(a,b,d) is observed.

P_1 does not cover e_4^+ and it is patched:

P_{1a}’ = {r(X,Y,Z) :- s(Y,Z) : ρ = 0.875
r(a,b,d) : ρ = 0.5}

$\chi(e^+_1)= \chi(e^+_2)= \chi(e^+_3) = 0.875, \chi(e^+_4) = 0.5$

$m\chi = 0.78, \mu\chi = 0.76$

P_{1b}' = {r(X,Y,Z) :- s(Y,Z) : $\rho = 0.875$
r(X,Y,d) : $\rho = 0.875$ }

$\chi(e^+_1) = \chi(e^+_2) = \chi(e^+_3) = \chi(e^+_4) = 0.875$

P_2' is reinforced = {r(X,c,Z) : $\rho = 0.75$.
r(a,Y,Z) : $\rho = 0.875$}

$\chi(e^+_1) = 0.875, \chi(e^+_2) = 0.75, \chi(e^+_3) = \chi(e^+_4) = 0.875$

P_3' is reinforced = {r(X,Y,Z) :- s(X,Y) : $\rho = 0.875$.
r(X,Y,Z) :- s(Y,Z) : $\rho = 0.875$}

$\chi(e^+_1) = \chi(e^+_2) = \chi(e^+_3) = \chi(e^+_4) = 0.875$

P_4' is reinforced = { r(X,Y,Z):-t(X,Y), t(Y,Z): $\rho = 0.9375$
t(X,Y) :- s(X,Y) : $\rho = 0.9375$
t(X,Y) :- s(X,Z), t(Z,Y) : $\rho = 0.75$}

$\chi(e^+_1) = \chi(e^+_2) = 0.8789, \chi(e^+_3) = \chi(e^+_4) = 0.6592$

$m\chi = 0.77, \mu\chi = 0.76$

P_5' is slightly reinforced

P_5' = { r(X,Y,Z) :- t(X,Y) : $\rho = 0.9375$.
t(X,Y) :- s(X,Y) : $\rho = 0.9375$
t(X,Y) :- s(X,Z), t(Z,Y) : $\rho = 0.5$}

$\chi(e^+_1) = \chi(e^+_2) = 0.8789, \chi(e^+_3) = 0.4395, \chi(e^+_4) = 0.8789$

$m\chi = 0.77, \mu\chi = 0.74$

At this moment, P_{1b}' and P_3' are the best options. Now P_4' and P_5' seem more grounded.

⌧ We add $e^{-}_3 = \neg$r(a,d,d)

P_{1a}' remains the same and P_{1b}' and P_2' are inconsistent, motivating the following 'patches' for them:

P_{2a}' = {r(X,c,Z) : $\rho = 0.75$.
r(X,b,Z) : $\rho = 0.75$}

$\chi(e^+_1) = \chi(e^+_2) = \chi(e^+_3) = \chi(e^+_4) = 0.75$

P_{2b}' = {r(X,Y,Z) :- e(Y) : $\rho = 0.9375$.
e(b) : $\rho = 0.75$
e(c) : $\rho = 0.75$}

$\chi(e^+_1) = \chi(e^+_2) = \chi(e^+_3) = \chi(e^+_4) = 0.7031$

P_3' and P_4' remain the same. P_5' becomes inconsistent.

⌦ We add e^+_5 = r(a,d,e)

P_{1a}', P_{2a}', P_{2b}' can only be patched with e^+_5 as an exception because abduction is not possible.

P_3' has abduction as a better option.

P_3'' = {s(d,e) : ρ = 0.5
r(X,Y,Z) :- s(X,Y) : ρ = 0.875
r(X,Y,Z) :- s(Y,Z) : ρ = 0.9375}

$\chi(e^+_1)=\chi(e^+_2)=\chi(e^+_3)=0.9375, \chi(e^+_4)=0.875, \chi(e^+_5)=0.4688$

$m\chi = 0.831$, $\mu\chi = 0.805$

P_4' makes the same abduction

P_4'' = { s(d,e) : ρ = 0.5
r(X,Y,Z):-t(X,Y),t(Y,Z): ρ=0.96875
t(X,Y) :- s(X,Y) : ρ = 0.96875
t(X,Y) :- s(X,Z), t(Z,Y): ρ = 0.875}

$\chi(e^+_1)=\chi(e^+_2)=0.939, \chi(e^+_3)=\chi(e^+_4)=0.82, \chi(e^+_5)=0.41$

$m\chi = 0.786$, $\mu\chi = 0.754$

At this moment, P_3'' and P_4'' are the best options.

Further examples of the theory would be required to distinguish with more reliability which is the 'intended' one.

The example illustrates that, in general, and by using this new reckoning of reinforcement, the shortest theories are not the best ones. More importantly, the weak parts are detected by a low value of reinforcement, and revision, if necessary, should be done to these parts of the theory. On the other hand, as soon as a theory gains some solidity, in terms of increase of reinforcement, abduction can be applied. Another advantage of this approach is that a 'rated' ontology can be derived directly from the theory.

5.5.2 Consilience can be precisely defined

The idea of 'consilience', introduced by Whewell in the XIXth century [Whewell 1847], and other related concepts, such as Reichenbach's principle of common cause, Thagard's coherence [Thagard 1978], all share the common idea of giving a conciliating theory for all the data, i.e., all the evidence must be accounted by the same explanation or by very closely related explanations.

In the context of reinforcement, it is easy to define consilience:

Definition 5.44 A theory T is partitionable with respect to an evidence E iff $\exists T_1, T_2 : T_1 \subset T, T_2 \subset T$ and $T_1 \neq T_2$ such that $\forall e \in E : T_1 \models e \vee T_2 \models e$. We define $E_1 = \{ e \in E : T_1 \models e \}$ and $E_2 = \{ e \in E : T_2 \models e \}$ and $E_{12} = E_1 \cap E_2$. Finally, we will use the term $S\chi(T_1 \oplus T_2, E)$ to denote the expression $m\chi(T_1, E_1) \cdot [\ card(E_1) - card(E_{12})/2\ ] + m\chi(T_2, E_2) \cdot [\ card(E_2) - card(E_{12})/2\ ]$.

Definition 5.45 A theory T is consilient with respect to an evidence E iff there does not exist a partition T_1, T_2 such that $S\chi(T_1 \oplus T_2, E) \geq m\chi(T, E) \cdot card(E)$.

In other words, a theory T is consilient with respect to an evidence E iff there does not exist a bi-partition $P \in \wp(T)$, such that every example of the evidence E is still covered separately and there is no loss of reinforcement.

Example 5.3

Given the following evidence (in Prolog):

$E = \{$ p(a), p(b), p(e), q(a), q(b), q(e), q(f) $\}$

The following program could be induced, with its corresponding reinforcements and courses:

$P = \{$ p(X) : $\rho = 0.875$

q(X) : $\rho = 0.9375\}$

$m\chi(E, P)= 0.9107$

The following partition:

$P_1 = \{$ p(X) : $\rho = 0.875\ \}$

$m\chi(E_1, P_1)= 0.875$

$P_2 = \{$ q(X) : $\rho = 0.9375\}$

$m\chi(E_2, P_2)= 0.9375$

In this case it is obvious that $m\chi(E_1, P_1) \cdot 3 + m\chi(E_2, P_2) \cdot 4 = 0.9107 \cdot 7$, so, as expected, P is not consilient.

The next example shows that consilience is a delicate notion:

Example 5.4 (using Horn theories)

Consider the following extensional theory $T= \{$ p. q. $\}$ for the following simple theory $E= \{$ p, q $\}$. As expected, $m\chi(T, E) = (0.5 + 0.5) / 2 = 0.5$ and by using the partition $T_1= \{$ p. $\}$, $T_2= \{$ q. $\}$ is easy to show that it is not consilient.

The trick is again the addition of a new fantastic rule f in the following way: $T'= \{$ p:- f. q:- f. f $\}$. As we have said, the mean course is robust to this kind of tricks, and it is clearly lower: $m\chi(T', E) = (0.5 \cdot 0.75 + 0.5 \cdot 0.75) / 2 = 0.375$. However, the only partition which is now possible, $T'_1= \{$ p:- f. f $\}$, $T'_2= \{$ q:-f. f. $\}$ gives that $S\chi(T'_1 \oplus T'_2, E)= 0.25 \cdot [1 - ½ \cdot 0] + 0.25 \cdot [1 - ½ \cdot 0] = 0.5 < m\chi(T', E) \cdot 2$. The result is that T' is consilient!

This example can be interpreted in two ways. If one has T and tries to make it consilient by using a fantastic concept, she would get an important decrease in $m\chi(T', E)$ enough for discarding T'. On the other hand, if one considers T' from scratch (without knowing T), she could be cheated by the illusion that T' is a good consilient theory if these invented concepts were difficult to detect.

It is important to realise that Definition 5.45 is reliable; independently from whether the unifying concept would be fantastic or not, the theory is properly consilient.

The aftermath harmonises with the classical rationale of the plausibility of a theory: it depends on the intuition, intelligence or whatever other ability to unveil fantasies by comparing the current theory with other competing theories. The advantage of these measures of mean course and consilience based on reinforcement is that the first one avoids fantastic concepts, so giving an approximation to plausibility, which must be weighed up with consilience.

The following example shows the use of mχ and consilience in the context of abduction and background knowledge. In this case, invented concepts are more difficult to introduce if the background knowledge cannot be modified by adding a fantastic rule.

Example 5.5 (using extended logic theories)

Let us suppose that in the nineteenth century a biologist has the following incomplete but fully validated background knowledge B, ($\forall r \in B\ \rho(r) = 1$).

B= { r_{b1}: Vertebrate(X) :- Fish(X)

r_{b2}: Vertebrate(X) :- Reptile(X)

r_{b3}: Vertebrate(X) :- Bird(X)

r_{b4}: Vertebrate(X) :- Mammal(X)

r_{b5}: Has-wings(X) $\lor$ Has-fins(X) :- Bird(X)

r_{b6}: Has-wings(X) $\lor$ Has-fins(X) :- Echo-locates(X),Mammal(X)

r_{b7}: Hasn't-mandibule(X) :- Agnate(X)

r_{b8}: Creeps(X) :- Reptile(X)

r_{b9}: Marine(X) :- Fish(X)

r_{b10}: Marine(X) :- Cephalopod(X) }

After performing some observations and dissections to a sample of animals from the Pacific Ocean, some hypotheses can be abduced:

E_1 = { e_1: Vertebrate(a), e_2: Creeps(a) }

$h_1 = E_1$ $m\chi(B+h_1, E_1) = 0.5$

h_2 = { Reptile(a). } $m\chi(B+h_2, E_1) = 0.75$

Moreover h_2 is consilient with respect to E_1.

E_2 = { e_3: Vertebrate(*b*), e_4: Marine(*b*) }

$h_3 = E_2$ $m\chi(B+h_3, E_2) = 0.5$

h_4 = { Fish(*b*). } $m\chi(B+h_4, E_2) = 0.75$

h_5 = { Cephalopode(*b*). Vertebrate(*b*). } $m\chi(B+h_5, E_2) = 0.5$

Only h_4 is consilient. Note that, according to B, h_5 is *consistent*.

E_3 = { e_5: Vertebrate(*c*), e_6: Has-wings(*c*) }

$h_6 = E_4$ $m\chi(B+h_6, E_3) = 0.5$

h_7 = { Bird(*c*). } $m\chi(B+h_7, E_3) = 0.75$

h_8 = { Echo-locates(*c*). Mammal(*c*). } $m\chi(B+h_8, E_3) = 0.625$

Both h_7 and h_8 are consilient.

E_4 = { e_7: Vertebrate (*d*), e_8: Hasn't-mandibule(*d*) }

$h_9 = E_4$ $m\chi(B+h_9, E_4) = 0.5$

h_{10} = { Agnate(*d*). Vertebrate(*d*). } $m\chi(B+h_{10}, E_4) = 0.5$

h_{11} = { Agnate(*d*). Vertebrate(X):-Agnate(X). } $m\chi(B+h_{11}, E_4) = 0.625$

In this last case, only h_{11} is consilient, and it shows that an extension can be made to B with new rules in order to cover the evidence in a consilient way.

However, the example shows that in many cases $m\chi$ is positively related to consilience, so it is a good criterion to guide knowledge creation and revision. Abduction has been naturally incorporated as a special case of explanatory induction, where, in general, the hypotheses are *factual* (although in the examples h_{11} includes non-factual ones and it can also be considered an abduction). It is remarkable to see that the hypotheses would be more accurate if B would be not completely validated $\exists r \in B\ \rho(r) < 1$ or, even better, if a separate measure of frequency were added to B, so reflecting the frequency of previous animal samples. Moreover, r_{b5} and r_{b6} should split their heads in order to compute independently their reinforcement. This all is more related with probabilistic abduction, which falls out of the topic of this chapter.

Finally, Definition 5.45 can be parameterised by introducing a consilience factor:

Definition 5.46

The degree of consilience of a theory T with respect to an evidence E is defined as the minimum real number k such that there exists a partition T_1, T_2 such that: $k \cdot S\chi(T_1 \oplus T_2, E) \geq m\chi(T, E) \cdot card(E)$.

From the computational point of view, both $m\chi$ and consilience degree should be computed jointly, in order to reduce the number of partitions which are to be examined.

5.5.3 Intrinsic Exceptions, Consilience and Noise

Using reinforcement, an intrinsic exception or extensional patch can be easily defined as a rule r with $\rho = 0.5$, i.e. it just covers one example e, or, in other words, it is necessary for only one example. However we must distinguish between completely extensional exceptions, when r does not use any rule from the theory to cover e, and partially extensional exceptions when r uses other rules to describe e .

The following theorem shows that completely extensional exceptions should be avoided to obtain consilient programs.

Theorem 5.20 If a worthy theory T for an evidence E has a rule r with $\rho = 0.5$, and completely extensional, then T is not consilient.

PROOF. Just choose the partition $T_1 = T - r$ and $T_2 = T$. Since $\rho = 0.5$ then r is only used by one example e_r. Since it is a completely extensional exception, we have that r does not use any rule from T_1 to cover e_r, so $\rho'(r_i) = \rho(r_i)$ for all $r_i \in T_1$. Let n be the number of the examples of the evidence E. Hence, $m\chi(T_1, E_1) = [m\chi(T, E) \cdot n - \chi(e_r,T)\,] / (n-1) = [m\chi(T, E) \cdot n - ½\,] / (n-1) = [m\chi(T, E) \cdot n + m\chi(T, E) - m\chi(T, E) - ½\,] / (n-1) = m\chi(T, E) + [m\chi(T, E) - ½] / (n-1)$.

From Definition 5.44, the disequality simplifies as follows:

$$S\chi(T_1 \oplus T_2, E) =$$

$$m\chi(T_1, E_1) \cdot [\text{card}(E_1) - \text{card}(E_{12})/2]+ m\chi(T_2, E_2) \cdot [\text{card}(E_2) - \text{card}(E_{12})/2] =$$

$$[m\chi(T, E) + [m\chi(T, E) - ½] / (n-1)\,] \cdot [(n-1) - (n-1)/2]+ m\chi(T, E) \cdot [n - (n-1)/2] =$$

$$m\chi(T, E)\cdot[(n-1) - (n-1)/2 + n - (n-1)/2]+[m\chi(T, E) - ½] \cdot [(n-1) - (n-1)/2] / (n-1) =$$

$$m\chi(T, E) \cdot [\,n\,] + [m\chi(T, E) - ½] / 2$$

Since T is worthy, then $m\chi(T, E) \geq 0.5.$, the left hand side

$$S\chi(T_1 \oplus T_2, E) \geq m\chi(T, E) \cdot n = m\chi(T, E) \cdot \text{card(E)}. \;\Box$$

In the same way, partially extensional exceptions are not convenient for consilience, but a limit would depend on how many rules are been used by the exception, because the separation would make the reinforcement of these rules decrease as follows $\rho'(r_i) = 2\cdot\rho(r_i) - 1$, by the corollary of Lemma 5.16.

In any case, not only intensionality (avoidance of exceptions) but consilience are both very strict requirements in the presence of noise, because any piece of data which is left as noise would be tried to be 'conciliated' with the rest of the theory, sometimes in an artificial way.

However, if used correctly, reinforcement is a very powerful tool to control the level of noise in a theory. This means that if we have any information or hint about the expected noise ratio, we can adjust the percentage of examples covered by extensional rules.

5.5.4 Reinforcement, Intensionality and Cross-Validation

Although the next chapter is devoted to the notion of intensionality, seen as avoidance of exceptions, we advance some results for reinforcement. The idea of intensionality is useful to distinguish between explanatory views of induction (and abduction as a particular case) and non-explanatory induction. In the latter case the goal is to describe compactly the evidence, but not to explain it. Moreover, there is a strong relation between intensionality (or avoidance of exceptions) and hypothesis stability.

In this section we will make the connection between intensionality (i.e. avoidance of exceptions, as they were defined in the previous section) and cross-validation. There are many variants of cross-validation (training-test split, leave-one-out or deleted estimate or *k*-fold). The relation between leave-one-out cross-validation and hypothesis stability was established by Devroye and Wagner [Devroye and Wagner 1979].

We will work with many-fold split, that is to say, we will take into account all the possible splits in all the possible orders, to see the influence of intrinsic exceptions in the theory. Let us denote with n_e the number of rules r that just cover one example e. In other words, if the example e had not appeared, the rule r would be useless. We will make the following reasonable assumption: a natural learning algorithm is a learning algorithm that does not add useless rules to the theory.

Let us define $P(A,T,E,k)$ as the probability that the algorithm A gives the theory T with the first k examples of the evidence E, considering all possible orderings of E.

Theorem 5.21 For any natural learning algorithm A,

$$P(A,T,E,k) \leq 1 - [(n-n_e)^{n-k} / n^{n-k}]$$

with $n = \text{card}(E)$.

PROOF. Let us denote with E^w the examples from E that are covered by a rule with $\rho = 0.5$. Let $w = \text{card}(E^w)$, $E^b = E - E^w$ and $b = n-w$. Obviously, $w \leq n_e$ since there can be examples covered by more than one exception rule. We denote with $E^{1..k}$ and $E^{k+1..n}$ the set of the first k examples and the rest of the n examples of a given ordering of E, in other words, a split at position k. We define $P^w(E,k)$ as the probability of $E^w \cap E^{k+1..n} \neq \emptyset$, i.e., the probability of having one exception example in the second part of the split. By a simple combinatorial analysis, by removing from the whole probability the probability of having all $E^{k+1..n}$ from E^b, this probability is:

$$P^w(E,k) = [(w+b)^{n-k} - b^{n-k}] / n^{n-k} = 1 - [b^{n-k} / n^{n-k}]$$

Since $b = n-w$, we have

$$P^w(E,k) = 1 - [(n-w)^{n-k} / n^{n-k}]$$

and $w \leq n_e$, then

$$P^{w}(E,k) \leq 1 - [(n - n_e)^{n-k} \ / \ n^{n-k}]$$

but $P(A,T,E,k) \leq P^{e}(E,k)$ because A is natural. □

The result can be understood that one should avoid exceptions, in order to have $P^{w}(E,k) = 1$. For instance, given a theory with 3 exception rules for an evidence of 100 examples, we have that the probability that the theory could be found with eighty examples is $P^{w}(E, 80) \leq 1 - 97^{20}/100^{20} = 0.46$.

The ideas of intensionality have been used in an incremental learning system [Hernández-Orallo and Ramírez-Quintana 1998] using Curry as a representation language (a logic functional programming language based on narrowing with some higher-order constructs). The results demonstrate that the *intended* hypothesis is found sooner than when using the MDL principle, because the latter allows the introduction of patches (exceptions) in an incremental session.

A deeper reflection on Theorem 5.21 shows that stability of the whole theory is a very strict requirement. If it is substituted by partial stability, i.e., how many rules of the theory can be obtained in early learning steps, the result may be quite different. Moreover, the connection between mean course and cross-validation would be more enlightening, although more difficult to obtain.

In the end, Theorem 5.21 is just an example of the connections that could be established between model selection methods for constructive languages, using reinforcement as a measure in a very differently way that other comparisons based on error estimation and attribute complexities [Kearns et al 1999]. In this section it has been done with a particular variant of cross-validation. In section 5.7 we will address the relation with the MDL principle.

5.6 Analogy, Consilience and Reinforcement

Although induction and abduction are recognised as the basic processes in scientific discovery, there is an inference process that is the fundamental mechanism for obtaining consilient theories, analogy. The reason is simple: as we commented in chapter 2, analogy extracts a common superstructure between two situations, and this 'shared' superstructure is reinforced by both situations.

If we restrict analogy under the following scheme:

Analogy:

Background Knowledge: b entails E_1 and c entails E_2.

Evidence: E_1 and E_2.

Process: Extract similarities between b and c into a new superstructure a in order to obtain a consilient theory composed of a, b' and c'.

We can state that analogy favours consilience.

Theorem 5.22 If b entails E_1, c entails E_2, b does not entail E_1 and c does not entail E_2, the new theory $T = \{ b', c', a \}$ such that $T_1 = \{ b', a \} \models E_1$ and $T_2 = \{c', a \} \models E_2$, and no other proper subset of T covers any example, is consilient.

PROOF. Since no other proper subset of T covers any example but T_1 and T_2, then there is only one possible partition to study consilience $\{T_1, T_2\}$. Since E_1 and E_2 are non-empty, then $m\chi(a, E_1) < m\chi(a, E_1 \cup E_2) > m\chi(a, E_2)$, and then $S\chi(T_1 \oplus T_2, E) < m\chi(T, E_1 \cup E_2) \cdot card(E_1 \cup E_2)$. From Definition 5.45, T is consilient. □

Once again, analogy, as it has defined, allows the introduction of fantastic concepts. In order to talk about a 'real' analogy, some information must be shared between b and c and moved into a. In other words, b' and c' should be simplified with respect to b and c. This can be related with reinforcement and extended from simple components such as b and c to sub-theories composed of many rules or components.

Definition 5.47 Non-fictitious Analogy:

Consider a theory T covering E, i.e., $\forall e \in E$, $T \models e$, which contains two sub-theories T_1 and T_2, which cover $E_1 \subset E$ and $E_2 \subset E$, respectively. A non-fictitious analogy is the addition to T of a new super-theory A, and the modification of T_1 and T_2 into T'_1 and T'_2 such that $T' = ((T / T_1) / T_2) \cup A \cup T'_1 \cup T'_2$ covers E, i.e. $\forall e \in E$, $T' \models e$, with the additional conditions that $m\chi(T', E) \geq m\chi(T, E)$ and T' must be consilient with respect to E_1 and E_2.

Note that if $T'_1 = T_1$ and $T'_2 = T_2$ there cannot be analogy. This definition is more accordant with classical computational approaches to analogy [Kling 1971] [Winston 1992].

5.7 Extended and Balanced Reinforcement

With the final measure introduced in section 5.4 there is still a tricky way of increasing reinforcement: joining rules. If a high-level representation language allows very expressive rules, larger rules can be made in order to stand for the same that was expressed with separated rules, with the advantage of increasing reinforcement and mean course:

Example 5.6

For instance, the following extended functional programs are equivalent:

```
Ta = {        r1 = { f(X,a)  →    g(b) },
              r2 = { f(X,c)  →    i(d) } }
Tb = {        r  = { f(X,Y)  →    if (Y=a) then g(b)
                                  if (Y=c) else i(d) } }
```

but T_b would be more reinforced than T_a.

In order to maintain the granularity of the theory there are two options: (1) the introduction of a factor directly related with the number of rules, and (2) the introduction of a factor inversely related with the syntactical length of each rule. We will choose this second option to clarify that this modification still makes our measure very different from a prior distribution such as the MDL principle.

With $\mathrm{length}(r)$ we will denote the length of a rule r for the concrete language which would be used. The only restriction for length is that for all r, $\mathrm{length}(r) \geq 1$. Thus we extend the definitions of section 5.4:

Definition 5.48 The extended pure reinforcement is defined as:

$$\rho\rho^*(r) = \rho\rho(r) / \mathrm{length}(r).$$

The extended normalised reinforcement $\rho^*(r)$ and the extended course $\chi^*(e)$ are defined in the obvious way by using $\rho\rho^*(r)$ and $\rho^*(r)$, respectively.

With this extension, it is easy to show that —in the limit— that compression is an excellent principle for increasing reinforcement:

Theorem 5.23 If the data E are infinite and a theory T is finite, the mean course $m\chi^*(T, E) = 1$.

PROOF. Given some infinite data as evidence $E = \{ e_1, ..., e_n \}$, without loss of generality, consider that T can be exclusively composed of two rules: r_1, which covers all E except e_i and, *independently*, r_2, which covers e_i. The reinforcements are $\rho^*(r_1) = (1-2^{(1-n)/\mathrm{length}(r_1)})$ and $\rho^*(r_2) = (1-2^{-1/\mathrm{length}(r_2)})$ and the mean course $m\chi^*(T, E) = [(n-1) \cdot (1-2^{(1-n)/\mathrm{length}(r_1)}) + (1-2^{-1/\mathrm{length}(r_2)})] / n$. For infinite data, we have that $lim_{n\to\infty}$ $m\chi^*(T, E) = 1$. □

The result is independent of the last extension given by Definition 5.48. In general, the theorem shows that maximum reinforcement matches with maximum compression in the limit (simply because both are saturated). However, when the data are finite we have many cases where they differ (significantly when the evidence is incompressible). The most blatant case occurs when some exception is covered extensionally (as r_2, which covers d_i in the proof of Theorem 5.23) and there is an important loss of reinforcement vs. a slight loss of compression. The following example illustrates this point:

Example 5.7

Consider the following evidence e_1–e_{10}:

E = { e_1: e(4) → true, e_2: e(12) → true,

e_3: e(3) → false, e_4: e(2) → true,

e_5: e(7) → false, e_6: e(7) → false,

e_7: e(20) → true,e_8: e(0) → true,

e_9: o(3) → true, e_{10}: o(2) → false }

where natural numbers are represented by using the functor *s* as the symbol for successor, e.g. s(s(s(0))) means 3. The length (denoted *l*) of a rule is computed as $1+n_f+n_v$, where n_f means the number of functors (including constants as functors with arity 0) and n_v being the number of variables.

From here, the following theories are evaluated:

		: *l*	ρρ	ρρ*	ρ*
T_a=	{e(s(s(X)) → e(X)	: 7	7	1	0.5
	e(0) → true	: 4	5	1.2	0.5647
	e(s(0)) → false	: 5	3	0.6	0.3402
	o(s(s(s(0)))) → true	: 7	1	0.1429	0.0943
	o(s(s(0))) → false	: 6	1	0.1667	0.1091}

The extended courses are $\chi^*(e_1, e_2, e_4, e_7, e_8) = 0.5 \cdot 0.5647 = 0.28235$, $\chi^*(e_3, e_5, e_6) = 0.5 \cdot 0.3402 = 0.1701$, $\chi^*(e_9) = 0.0943$ and $\chi^*(e_{10}) = 0.1091$.

The mean extended course $m\chi^*$' is 0.2125.

		: *l*	ρρ	ρρ*	ρ*
T_b=	{e(s(s(X)) → e(X)	: 7	7	1	0.5
	e(0) → true	: 4	5	1.2	0.5647
	e(s(0)) → false	: 5	3	0.6	0.3402
	o(s(s(X)) → o(X)	: 7	2	0.2857	0.1797
	o(0) → false	: 4	1	0.25	0.1591
	o(s(0)) → true	: 5	1	0.2	0.1294}

The extended courses are $\chi^*(e_1, e_2, e_4, e_7, e_8) = 0.5 \cdot 0.5647 = 0.28235$, $\chi^*(e_3, e_5, e_6) = 0.5 \cdot 0.3402 = 0.1701$, $\chi^*(e_9) = 0.1797 \cdot 0.1294 = 0.02325$ and $\chi^*(e_{10}) = 0.1797 \cdot 0.1591 = 0.02859$.

The mean extended course $m\chi^{*}$' is 0.1974.

	: l	$\rho\rho$	$\rho\rho^*$	ρ^*
T_c= {e(s(s(X)) → e(X)	: 7	9	1.2857	0.5898
e(0) → true	: 4	6	1.5	0.6464
e(s(0)) → false	: 5	4	0.8	0.4257
o(X) → not(e(X))	: 6	2	0.3333	0.2063
not(true) → false	: 4	1	0.25	0.1591
not(false) → true	: 4	1	0.25	0.1591}

The extended courses are $\chi^*(e_1, e_2, e_4, e_7, e_8) = 0.5898 \cdot 0.6464 = 0.3813$, $\chi^*(e_3, e_5, e_6) = 0.5898 \cdot 0.4257 = 0.2511$, $\chi^*(e_9) = 0.2063 \cdot 0.5898 \cdot 0.4257 \cdot 0.1591 = 0.00824$ and $\chi^*(e_{10}) = 0.2063 \cdot 0.5898 \cdot 0.6464 \cdot 0.1591 = 0.0125$.

The mean extended course $m\chi^{*}$' is 0.2681.

Note that the lengths ($l(T_a)$=29, $l(T_b)$=32, $l(T_c)$ = 30) would not give many hints about which theory to select.

The example also shows the advantages of this approach for explanation-based learning. Since all the data must be explained, if a part is left in an extensional way (or unrelated with the rest), it is penalised. On the other hand, we have seen in the preceding sections that *fantastic* concepts are also avoided, so it results to be a *balanced* criterion for a more reasonable degree of intensionality of theories, without falling into fantasy.

Regarding T_c of Example 5.7, this measure can be adapted to situations where a more compensated theory is required, by using a *geometric mean* instead of an *arithmetic mean*. In addition, and concerning T_a, if exceptions (extensional parts) are not admitted at all, any theory where a fact has a course value less than the mean divided by a constant can be discarded. More formally,

Definition 5.49 A theory T is k-balanced with respect to an evidence E if:

$$\neg\exists\, e \in E : \chi^*(T, e) < k \cdot m\chi^*(T, E)$$

where k is a value between 0 a 1. If k=1 all the evidence must have exactly the same course, and if k=0 every theory is balanced.

The use of a intermediate value (e.g. 0.5) suggests the triggering of theory revision in an incremental framework in order to integrate (or reconcile) the example that has low course with the theory.

5.8 Rewarded Reinforcement

Up to this moment we have only dealt with positive (and absolute) reinforcement. An example is covered or not by the theory. In reinforcement learning, though, it is usually assumed that the learner receives different reward and penalty values for its actions. In other words, prediction hits can receive different degrees of reward and prediction errors (including novelties and anomalies) can receive different degrees of penalty (or negative reward).

Usually, this broader view of reinforcement is suitable for frameworks where reasoning about action is necessary. The rewards are assigned depending on the actions that the agent performs for each situation. Apart from Markov decision processes [Kearns and Singh 1999], other more expressible temporal languages are used for representation, such as event calculus or situation calculus [Kowalski and Sachi 1997]. The important issue here is that our model selection measures can be used for these high-level representations. The value of reinforcement can be understood as the prediction reliability of the following situation s_{n+1} after every possible action that can be performed in a certain situation s_n. The task of the system seems to be to select the one with the greatest reward. With this first approach, in the case the result of the action matches with the evidence, a positive hit happens with the predicted reward. However, in the case a prediction error occurs, the action might have no awful consequences (no penalty), but, in some cases, it may be fatal. The question is how ontology and 'hedonism' must be combined. It is commonly accepted in psychology the claim that hedonism motivates ontology, and this is stronger the earlier the stage of development of a cognitive system. In my opinion, this motivation does not imply that they must be mixed. Moreover, rewards should also be learned because they may change.

Hence, the choice of the best action must take into account both the reliability of the prediction (i.e. the reinforcement) weighed with the reward, not the action with the best reward alone (because it may be a very weak guess).

Summing up, the decision of which action should be taken would depend on:

- the reliability of recognising the situation where the agent is actually embedded.
- the reliability of predicting the consequence of a given action in that situation.
- the reward (or penalty) of the consequence.

This implies degrees of reliability in the evidence. This degree may come from different reliabilities of the sensors of the system or from intermediate recognition or sensor pre-processing subsystems. We will take this into account in the following way: every fact of the evidence is assigned a real number as a reliability degree, $-1 \leq d_f \leq 1$. In this framework, the completely reliable positive examples are assigned a value of $d_f = 1$ and the completely reliable negative examples are assigned a value of $d_f = -1$.

Definition 5.50 The *'grounded' course* $\chi'(f)$ of a given fact f with respect to a theory is computed as the normal course $\chi(f)$ multiplied by the reliability degree of f. More formally, $\chi'(f) = \chi(f) \cdot d_f$.

In the previous section we considered the length of rules. Another straightforward extension to our approach is to consider the length of the examples, too. This can also be incorporated in the same way as the reliability degree.

Finally, Definition 5.50 introduces negative values for d_f. Nonetheless, they are still considered in a positive way for computing ρ. The next section discusses how to incorporate this negative evidence into the theory.

5.9 External Inconsistencies. Negative Reinforcement

Hitherto we have only considered theories which are completely consistent with the evidence. Whenever an uncovered or inconsistent example was found, the theory was remade (patched) in order to cover the new positive example (novelty) or uncover the new negative example (anomaly).

In some contexts, an anomaly, if not patched, should make the theory be rejected, as it has been commented in the previous sections, where structural patches should also be avoided. But in other contexts (approximate learning, noisy data, etc.) a single anomaly should not force the revision of the whole theory.

A first idea to handle anomalies is to compute the course of positive examples and compute the course of negative examples that are covered. An optimality criterion for the theory could be given by the positive ${}_m\chi$ minus the negative ${}_m\chi$. This could be made by using Definition 5.50. Concretely,

Definition 5.51 The (+/–) *course* $\chi(f)$ of a given fact f with respect to to a theory is computed as the normal course $\chi(f)$ multiplied by 1 if $f \in E^+$ and -1 if $f \in E^-$.

However, this measurement would not allow to know which rules are being affected. Morever, it is somehow paradoxical because anomalies that occur through few very reinforced rules are much more taken into account than anomalies that occur through many reinforced rules or non-reinforced rules. In some way, Definition 5.51 measures the *hardness* for conciliating the anomaly, i.e., the difficulty to revise the theory to account for the new evidence without reorganising the rest of the theory. Or, seen in other way, how plausible is it to expect that f is noise, or, in other words, a measure of **surprise**, from values close to 0 (no surprise) to values close to -1 (very surprising).

If the theory is not to be remade or we want to detect which rules are affected, we must propagate negative reinforcement into the rules, too. A much more insightful extension is to re-consider the reinforcement of each rule in the following way.

Definition 5.52 The positive pure reinforcement $\rho\rho^{+}(r)$ of a rule r from a theory T with respect to some given positive evidence $E^{+} = \{e^{+}_{1}, e^{+}_{2}, ..., e^{+}_{n}\}$ is defined as:

$$\rho\rho^{+}(r) = \Sigma_{i=1..n}\, card(Proof_{r}(e^{+}_{i},T))$$

Definition 5.53 The negative pure reinforcement $\rho\rho^{+}(r)$ of a rule r from a theory T with respect to some given negative evidence $E^{-} = \{e^{-}_{1}, e^{-}_{2}, ..., e^{-}_{n}\}$ is defined as:

$$\rho\rho^{-}(r) = \Sigma_{i=1..n}\, card(Proof_{r}(e^{-}_{i},T))$$

Both measures are identical to Definition 5.39. The question is how to weigh positive and negative pure reinforcement in a single value. One option is to normalise and then to weigh them, formalised in the following way:

Definition 5.54 The (normalised) positive reinforcement is defined as:

$$\rho^{+}(r) = 1 - 2^{-\rho\rho+(r)}$$

Definition 5.55 The (normalised) negative reinforcement is defined as:

$$\rho^{-}(r) = 1 - 2^{-\rho\rho-(r)}.$$

And finally,

Definition 5.56 The (normalised) reinforcement is defined as:

$$\rho_{1}(r) = \rho^{+}(r) - \rho^{-}(r).$$

It is obvious that this measure matches with Definition 5.40 if no negative evidence is given. This is a very strict way of considering negative evidence because if a rule has a single negative example covered by it, its reinforcement is necessary less than 0.5, so every positive evidence which is covered by it should be more optimally covered by an extensional patch.

Moreover, there can be 'independent' properties such as "X + Y = Y + X" that can be used for negative evidences, and they would be highly penalised by that kind of measurement unless a smoothing factor is considered. In some cases, an 'auxiliary' rule could be distinguished for the 'guilty' rule because auxiliary rules often do not cover any evidence alone. However, this heuristic is not valid in general.

Another option is to subtract them and then normalise them. More formally,

Definition 5.57 The pure reinforcement is defined as:

$$\rho\rho_{2}(r) = \rho\rho^{+}(r) - \rho\rho^{-}(r).$$

And normalised reinforcement $\rho_{2}(r)$ is obtained, by normalising, in the initial way. This minimises the problem but, in the end, it is just a question of how to weigh positive and negative reinforcement.

In fact, the best solution is to consider separately each (+/−) *course* $\chi(f)$, and both variants of $\rho^{+}(r)$ and $\rho^{-}(r)$, in order to know whether and where the theory should be revised. Another derived measure of course could be given as:

Definition 5.58 The course $\chi^0(f)$ of a given fact f with respect to a theory T is defined as:

$$\chi^0(f) = \begin{cases} max_{S\subset Proof(f,T)} \{ \Pi_{r\in S}\, \rho^+(r) \} & \text{if } f\in E^+ \\ -max_{S\subset Proof(f,T)} \{ \Pi_{r\in S}\, \rho^-(r) \} & \text{if } f\in E^- \end{cases}$$

And the mean $m\chi^0(f)$ is computed as $\sum_{f\in E}\chi^0(f)$ / card(E^-)

Let us show the use of these new measures for the same positive and negative evidence than Example 5.2:

Example 5.8

Consider the the background theory B = { s(a,b), s(b,c), s(c,d) } and the evidence

E = { e^+_1: r(a,b,c), e^+_2: r(b,c,d), e^+_3: r(a,c,d), e^+_4: r(a,b,d), e^+_5: r(a,d,c), e^-_1: ¬r(b,a,c), e^-_2: ¬r(c,a,c) , e^-_2: ¬r(a,d,d) }:

T_1 = { r_1 = r(X,Y,Z) }: $\rho^+(r_1) = 1 - 2^{-5} = 0.96875$

$\rho^-(r_1) = 1 - 2^{-3} = 0.875$

$\rho_1(r_1) = \rho^+(r_1) - \rho^-(r_1) = 0.09375$

$\rho_2(r_1) = 1 - 2^{-(5-3)} = 0.75$

$m\chi^0(T_1, E) = (0.96875 \cdot 5 \; - \; 0.875 \cdot 3) / 5 = 0.44$

$m\chi^1(T_1, E) = (0.09375 \cdot 5) / 5 = 0.09375$

$m\chi^2(T_1, E) = (0.75 \cdot 5) / 5 = 0.75$

T_2 = { r_2 = r(X,c,Z) $\rho^+(r_2) = 1 - 2^{-2} = 0.75$

r_3 = r(a, Y,Z) $\rho^-(r_2) = 0$

$\rho_1(r_2) = \rho^+(r_2) - \rho^-(r_2) = 0.75$

$\rho_2(r_2) = 1 - 2^{-(2-0)} = 0.75$

$\rho^+(r_3) = 1 - 2^{-4} = 0.9375$

$\rho^-(r_3) = 1 - 2^{-1} = 0.5$

$\rho_1(r_3) = \rho^+(r_3) - \rho^-(r_3) = 0.4375$

$\rho_2(r_3) = 1 - 2^{-(4-1)} = 0.875$

$m\chi^0(T_2, E) = (0.9375 \cdot 4 + 0.75 \cdot 1 - \; 0.5 \cdot 1) / 5 = 0.8$

$m\chi^1(T_2, E) = (0.4375 \cdot 3 + 0.75 \cdot 2) / 5 = 0.5625$

$m\chi^2(T_2, E) = (0.875 \cdot 4 + 0.75 \cdot 1) / 5 = 0.85$

The first theory is more positively reinforced but has 3 anomalies whereas the second one is more extensional but it has only 1 anomaly. The three different measures give better results for T_2 than for T_1. $m\chi^1$ is very strict with anomalies, $m\chi^2$ very lax and

$m\chi^0$ finds a compromise between the other two. For revision purposes, though, the measure ρ_1 is more illustrative to see which rule should be addressed first.

5.10 Reinforcement and Deduction

Reinforcement has almost always been related with induction in the literature, at least always with learning algorithms. However, a deductive inference can establish new connections and, consequently, the theory's ontology and robustness, are also increased. There are two different cases to consider here, omniscient and non-omniscient systems. The existence of new deductive connections in non-omniscient systems is intuitive and necessary, as we saw in the previous chapter. However, for omniscient systems, are there any new connections to establish? The answer is no, but this does not mean that the 'explicitation' of deductive consequences could not increase the whole reinforcement in both omniscient and non-omniscient systems. Let us first see this case:

5.10.1 Derived Rules Explicitation

There is an important trait of reinforcement propagation that has been latent in the character of the theories we are dealing with. Theories are usually convenient and reduced representations of the evidence they cover, and reinforcement is distributed among these explicit rules and not all their derivable consequences.

This situation is especially appropriate for understanding jointly the function of both induction and deduction in the construction of an ontology. As well as induction adds new rules to a theory, deduction must be used to derive new theorems from a theory, which can act as properties or rules that are adjoined explicitly to it.

Consequently, new derivations can increase the reinforcement of the theory with respect to the same evidence. For instance, if a rule *a* entails E_1 and *b* entails E_2. A new rule *c* may be derived from *a* and *b* such that *c* entails most of E_1 and E_2.

Note that it is independent to whether the rule is derivable from *a* and *b* under the same or different semantics where the theory operates. The relevance is that *c* is left explicit, i.e., it composes the theory.

Consider the following example:

Example 5.9

Given the evidence

$E = \{ e^+_1$: p(a, c), e^+_2: p(a,b), e^+_3: p(c,a), e^+_4: r(c,c) $\}$

and the following Horn theory

$T_1 = \{$ r_1 = p(X,Y) :- r(X,X). $\rho(r_1) = 1 - 2^{-3} = 0.875$

r_2 = r(a,a). $\rho(r_2) = 1 - 2^{-2} = 0.75$

r_3 = r(a,b). $\rho(r_3) = 0$

r_4 = r(b,c). $\rho(r_4) = 0$

r_5 = r(c,c). } $\rho(r_5) = 1 - 2^{-2} = 0.75$

the theory is specialised by the following derivations (resolution)

$$\frac{\begin{array}{l}\text{p(X,Y) :- r(X,X)}\\ \text{r(a,a)}\end{array}}{\text{p(a,Y)}}$$

and

$$\frac{\begin{array}{l}\text{p(X,Y) :- r(X,X)}\\ \text{r(c,c)}\end{array}}{\text{p(c,Y)}}$$

and added to the theory:

T'_1 = { r_1 = p(X,Y) :- r(X,X). $\rho(r_1) = 1 - 2^{-3} = 0.875$

r_2 = r(a,a). $\rho(r_2) = 1 - 2^{-2} = 0.75$

r_3 = r(a,b). $\rho(r_3) = 0$

r_4 = r(b,c). $\rho(r_4) = 0$

r_5 = r(c,c). $\rho(r_5) = 1 - 2^{-2} = 0.75$

r_6 = p(a,Y). $\rho(r_6) = 1 - 2^{-2} = 0.75$

r_7 = p(c,Y). } $\rho(r_7) = 1 - 2^{-1} = 0.5$

Note that the reinforcement of the original rules are not modified by these derivations. However, the courses of some examples are increased due to the use of r_6 alone for e^+_1 and e^+_2. This happens because r_1 is somehow fantastic for the evidence, an artificial way to conciliate apparently sparse examples.

A question that is suggested by the preceding example, especially the case of r_7, is whether it is intuitive to assign to r_7 a value of reinforcement less than the rules where it derives from. In other words, if r_1 has $\rho(r_1) = 0.875$ and r_2 has $\rho(r_2) = 0.75$ we have that $r_1, r_2 \models r_7$ but $\rho(r_7) = 0.7$. This is contrary to what is given by Carnap calculus.

This apparent paradox is dissipated if one only understand the whole mean course as a measure of plausibility and nothing more. In other words, the reinforcement of a rule is only a measure of its use and it cannot be seen as a measure of independent plausibility and even less of its probability.

There are some ways to give a measure of plausibility, different from reinforcement. A first idea is to conceive course as plausibility, given the following equation:

Definition 5.59 For every rule s such that $T \models s$ we define its plausibility as:

$$P_1(s) = max_{S \subset Proof(s,T)} \{ \Pi_{r \in S}\, \rho(r) \}$$

which is like Definition 5.42 but applied to rules as well as evidence. Or alternatively,

Definition 5.60 For every rule s such that $T \models s$ we define its plausibility as:

$$P_2(s) = max_{S \subset Proof(s,T)} \{ min_{r \in S}\, \rho(r) \}$$

Definition 5.60 matches with many works about logics with uncertainty values. For instance, if C represents the certainty of a given fact, $C(p \wedge q) = \min (C(p), C(q))$ and $C(p \vee q) = \max (C(p), C(q))$. On the contrary, Definition 5.59 resembles some other popular theories of uncertainty when C is measured between 0 and 1, $C(p \wedge q) = C(p) \cdot C(q)$ and $C(p \vee q) = 1 - (1 - C(p)) \cdot (1 - C(q))$

These annotated plausibility can have a partial use to guide revisions of the theory, and it is more important as long as the theory is getting larger, and parts of the theory are instances or specialisations of more general theories that must be explicitly stated for the sake of efficiency and shortening of proofs. We will get back on its use in the following subsection.

However, for the whole theory, there is no need for a new whole plausibility criterion, because $m\chi$ is robust to deduction. This is even more justified because deduction cannot decrease the whole course of the theory, as the following theorem shows:

Theorem 5.24 Given any theory T and any evidence E, if any rule r is added such that $T \models r$, then $m\chi(T, E) \leq m\chi(T \cup \{ r \}, E)$.

PROOF. By the definition of $\chi(T, e) = max_{S \subset Proof(e,T)} \{ \Pi_{r \in S}\, \rho(r) \}$it is obvious that $\chi(T, e) \leq \chi(T \cup \{ r \}, e)$. □

The things are different if we consider negative reinforcement, at least for $m\chi^0(T, E)$, which can be reduced by a derivation which jointly covers more negative examples than its premises. For, $m\chi^1(T_2, E)$ and $m\chi^2(T_2, E)$ the previous theorem also holds. In fact, in these latter two cases, it is usual than a specialisation (such as Example 5.8) highly increases reinforcement, something that it is more accordant with Carnap's Calculus.

However, the question of which representation is the best for a given evidence, discussed in the previous chapter, can be complemented by the use of these new measures.

In the previous chapter we discussed about minimising $LT(e)$ for any example e of the evidence (i.e. shortening proofs), giving a measure of optimality as $opt_{DS}(T| E) = argmin_T(\Sigma_{e \in E}\, Cost(e|T))$. In the context of reinforcement, we have seen that the best theory was $argmax_T(m\chi(T, E))$. Given a non-optimal theory constructed by induction, we can still consider deduction for improving it, because we can express the same theory with more derived rules or a specialised version of it. Since induction is usually a harder problem than deduction, it seems logical to profit any increase that could be obtained by deduction before *revising* the theory by inductive methods.

According to reinforcement, we can define different variants of optimal specialisations of a theory and representations by using exclusively deductive inference:

Definition 5.61 Optimal Specialisation of a Theory:

A theory T is an optimal specialisation for E if there does not exist another T' such that $T \models T'$ such that $m\chi(T, E) < m\chi(T', E)$.

Definition 5.62 Optimal Strict Specialisation of a Theory:

A theory T is an optimal strict specialisation for E if there does not exist another T' such that $\forall r \in T'$, $T \models r$ and $T \subset T'$ such that $m\chi(T, E) < m\chi(T', E)$.

Definition 5.63 Optimal Representation of a Theory:

A theory T is an optimal representation for E if $T \models E$ and there does not exist another T' such that $\forall e\ T \models e \leftrightarrow T' \models e$ and $m\chi(T, E) < m\chi(T', E)$.

Definition 5.64 Optimal Strict Representation of a Theory:

A theory T is an optimal strict specialisation for E if $T \models E$ and there does not exist another T' such that $\forall e\ T \models e \leftrightarrow T' \models e$ and $\forall r \in T'$, $T \models r$ and $T \subset T'$ such that $m\chi(T, E) < m\chi(T', E)$.

The difference between non-strict and strict variants is important. For instance, T = { p(X,a,b). p(c,Y,c). r(b). r(c). } implies (under the close-world assumption) T' = { p(X,a,b). p(c,Y,c). p(c,a,Z) :- r(Z). r(b). r(c). } but $T \not\models$ { p(c,a,Z) :- r(Z). }.

In many cases, these optimisations could be automatised. In fact, some inductive algorithms work in the previous way (top-down). They construct the most general theory and specialise it by deduction in order to arrive to the optimal (or an acceptable) one.

5.10.2 Non-Omniscient Deduction

Inside non-omniscient systems, the relevance of deduction and the possibilities of increasing reinforcement are much more important than in the case of omniscient theories.

Imagine the situation of an axiomatic theory T composed of one part (or set of rules) T_1 such that entails E_1, and a second part T_2 that entails E_2. No relation is still established between T_1 and T_2. Imagine that a new theorem is found that establishes that T_1 entails T_2, or, in other words, T_2 is a special case of T_1.

Many rules of T_1 can be reinforced now by E_2 too by, according to the formula $\rho\rho(r) = \Sigma_{i=1..n}\, card(Proof_r(e_i,T))$, and, consequently, the plausibility of the whole theory increases, according to $\chi_T(f) = max_{\,S\subset Proof(e,T)}\ \{\ \Pi_{r\in S}\ \rho(r)\ \}$. This situation is usual in physics, whenever two unrelated explained phenomena are connected by a theory.

The interpretation of new connections is different depending on the course of each separate theory. For instance, if T_1 is much more reinforced than T_2, the connection can be interpreted as an explanation for T_2, which originally lacked the support that T_1 provides. In this case, the measures of plausibility which were presented in the previous subsection (P_1 and P_2) may be useful to propagate reinforcement downwards, to recognise the reinforcement that T_2 deserves.

On the contrary, if T_2 is much more reinforced than T_1, the connection is then understood as further support for T_1.

Both cases are independent to the use of T_2 in an explicit way after the connection. This would depend on the time complexity of both T_1 and T_2. In many cases, T_2 could be much more efficient than T_1 or even can give better value for χ than T_1, as in Example 5.9.

Additionally, a new established connection can 'consiliate' a theory that was initially not consilient, because the evidence was explained by separate theories which are now related, as we saw in section 5.6 and 5.7.

Finally, a new connection can be made with previously independent evidence. For instance, consider the case where T_1 entails E_1, and T_2 entails E_2 and E_3. A new deductive connection establishes that T_1 also entails E_2. This must force the revision of the theory, because there may be a better theory than T_2 for covering E_3. More extremely, consider than it is also found that T_2 also entails E_3. Then T_2 should only be maintained if it covers better (according to some optimality criterion) the evidence E_2 and E_3. In many other cases, it should be removed because it is no longer necessary. It is necessary to detect the cases when, following the example, $m\chi(T_1, E_1 \cup E_2 \cup E_3) \geq m\chi(T_1 \cup T_2, E_1 \cup E_2 \cup E_3)$ and consider whether T_2 should be preserved. This process would be similar to garbage collection, and finally removes from *explicit* memory the properties and rules that are no longer used.

5.10.3 Reinforcement, Consilience and Interestingness in Mathematics

Finally, there are some questions about mathematical utility and interest that can be enlightened by the use of reinforcement, more specifically, by the use of consilience. Although consilience was introduced by Whewell for scientific theories, we can see that this notion applies as well for mathematical theories.

For instance, Bundy et al. propose a system for inventing mathematical definitions and conjectures [Bundy et al. 1998]. They use a weighed measure of parsimony and clarity, but it is not sufficient to distinguish those concepts that the authors consider interesting. For example, they find attractive the concept of 're-factorable numbers', found by the system, which is defined as "*those integers for which the number of factors is itself a factor of the integer*". This can be easily understood as a very reinforced and consilient concept, since the same rules are repeatedly used in the same definition.

The idea is not far from any theory that is originally constructed from an evidence but, after a time, it finally becomes a very theoretical system, where deduction is also as important as experimentation and induction. The question Hintikka raised open in the seventies [Hintikka 1970a] is still relevant today:

> [...] But even within deductive systematization there are many other important properties of theories which we will not directly deal with here. For example, it is claimed that theoretical concepts may in some pragmatic sense give us deeper understanding and better explanation of the phenomenon under investigation that do statements in the vocabulary of λ only. We certainly agree with this. Moreover, it may (or may not) be the case that theories using auxiliary concepts are heuristically more fruitful, more manageable and suggestive, and also simpler (in some sense) than purely observational statements. Again, there may be strong ontological concepts like fields or unconscious wishes, for example, because we believe that there are such entities and that why can be studied for their own sake.
>
> Despite the restricted scope of our discussion, we believe that we have already brought out something interesting. There are two main aims which auxiliary terms are typically supposed to serve. They are (observational) *richness* and *economy*. However, in the literature of philosophy of science we find next to no insights as to *how* they serve these purposes, and not even much indication that they in fact succeed in doing so.

I think that both the information gain measure of the previous chapters and the reinforcement theory of this one, give many results about *how* auxiliary concepts serve *these purposes*, and a support that they in fact succeed in doing so.

5.11 Reinforcement as a Theory of Confirmation

In chapter 2, the two different approaches from two philosophers and logicians from the Wiener Kreis were discussed. A quantitative concept of degree of confirmation, as a value between 0 and 1 for a hypothesis given an evidence, was developed by Carnap, who associated it, as seen, with a notion of probability. On the contrary, Hempel introduced a qualitative concept of confirmation, i.e., a Boolean relation between hypothesis and evidence, in the way that E confirms H or E does not confirm H.

In the previous section we have included two different functions P_1 and P_2 which, along with ρ and χ, could be used as a quantitative measure of confirmation, somehow in between Carnap and Hempel.

In particular, they are compliant with some of Hempel's adequacy conditions, but in a quantitative way:

(H1) *Entailment condition*: any sentence which is entailed by an observation report is confirmed by it.

(H1.1) Any observation report is confirmed by itself.

In other words, evidence is not questioned by the theory of reinforcement. The plausibility of e is maximum if $e \in E^+$.

(H2) *Consequence condition*: if an observation report confirms every one of a class K of sentences, then it also confirms any sentence which is a logical consequence of K.

(H2.1) *Special consequence condition*: if an observation report confirms a hypothesis H, then it also confirms every consequence of H.

(H2.2) *Equivalence condition*: if an observation report confirms a hypothesis H, then it also confirms every hypothesis which is logically equivalent with H.

(H2.3) *Conjunction condition*: if an observation report confirms each of two hypotheses, then it also confirms their conjunction.

Reinforcement is partially accordant with H2 and H2.1*, although this depends of how many elements there are in K and their particular reinforcement values.* Obviously it also depends of which of the two functions (P_1 or P_2) is chosen. On the contrary, H2.2 is not compatible with the theory developed in this chapter because the *form* (explicit representation) is important for reinforcement. Finally H2.3 is partially followed, provided both are consistent, although in any case the value is somehow minimised by the conjunction, depending on P_1 (product) or P_2 (min).

(H3) *Consistency condition*: every logically consistent observation report is logically compatible with the class of all the hypotheses which it confirms.

(H3.1) Unless an observation report is self-contradictory, it does not confirm any hypothesis with which it is not logically compatible.

(H3.2) Unless an observation report is self-contradictory, it does not confirm any hypotheses which contradict each other.

This adequacy condition has been included (in a quantitative way) by the fact that only positive evidence distributes positive reinforcement and the view that negative evidence distributes negative reinforcement.

(H4) *Equivalent condition for observations*: if an observation report B confirms a hypothesis H, then any observation report logically equivalent with B also confirms H.

This adequacy condition is completely fulfilled by the theory of reinforcement.

(H5) *Converse consequence condition*: if an observation report confirms a hypothesis *H*, then it also confirms every formula logically entailing *H*.

This is the base for reinforcement propagation, although this confirmation is quantitative and greater as long as more evidence confirms each part of the theory.

The following classical paradoxes (from [Holland et al. 1986]) are also avoided by the theory of reinforcement. The first one can be stated by the fact that the proposition "All ravens are black" is confirmed by observations of ravens that are black. But the statement "All ravens are black" is formally equivalent to the statement "All nonblack things are nonravens" (Adequacy condition H2.2). The latter proposition would be confirmed by a white shoe. Nonetheless, this does not happen for reinforcement, since the form of the theory is important and more rules are necessary for expressing "all nonblack things are nonravens", apart from being of little use for covering the evidence.

The other paradox is the famous "grue paradox" [Goodman 1965]. Define "grue" as "green before time *t* and blue otherwise". Then observing a green emerald seems to confirm equally well both "All emeralds are green" *and* "All emeralds are grue" (assuming *t* is still in the future). According to Goodman, from a syntactic perspective it is hard to see why "All emeralds are green" is the most attractive conclusion. However, there *are* syntactic criteria that solve this problem. A simplicity criterion gives more plausibility to "All emeralds are green" than to "All emeralds are grue" because the first one is shorter to describe (the second one must include the definiton of grue). In the same way, the theory of reinforcement (although is partially a semantic criterion) is also free from this paradox.

In my opinion, a quantitative (but not probabilistic) way is the only way to include both H2 (top-down) and H5 (bottom-up). As a result, reinforcement is a theory between the MDL principle and Popper's informativeness, which is also useful for deduction, induction, analogy and abduction.

5.12 Reinforcement and Information Gain

As we have seen, different measures based on reinforcement (especially mean course) can act as plausibility criteria. On the contrary, Information Gain was a measure of effort, of resource investment, that was only partially related to plausibility. It is precisely the combination of a plausibility criterion and a gain criterion which fully exploits the possibilities of both theories.

Let us first study the relationship between reinforcement and information gain and then their combination.

5.12.1 Reinforcement vs. Gain

For the case of induction, when we analyse the information gain of a theory with respect to an evidence, there are two special cases where both measurements are positively related. The first case is given when the theory which has been induced is completely extensional, namely, $T = E$. In this case, $G(T \mid E) = 0$ and $m\chi(T, E) = 0.5$, both being the minimum value of both measures.

A quite different case is when T highly compresses the evidence. As we saw in the previous chapter, $G(T \mid E)$ is usually high. In the same way, as we have seen in this chapter, $m\chi(T, E)$ tends to 1 as the compression ratio increases.

In the rest of cases, there is no clear relation about the effort or gain for obtaining the theory and its mean course, because, as we have said, both concepts represent different dimensions.

For the case of deduction, we studied how to optimise (in this case, minimise) the value of $G(E \mid T)$, because a good theory should ease the extraction of its consequences (maybe by the use of intermediate, valuable properties, which will be discussed in the following subsections). By the way χ is computed, complex proofs are avoided, because the reinforcement of each rule is multiplied, and, hence, quickly lowered. However, this does not mean that the *time* complexity of obtaining a single evidence cannot be high, because a rule can be used many times. Hence, the idea for conciliating efficiency and reinforcement is to use a lot of rules few times, without loops.

For both $G(E|T)$ and $V(E|T)$, it is also *relative* descriptional space which should be minimised. In this case, a conciliation is much more difficult, according to the formula $\rho\rho(r) = \Sigma_{i=1..n}\ card(Proof_r(e_i,T))$. A good way to increase the reinforcement of a rule is that it participates in many proofs, so finally we have that each example could have many different proofs. This makes that, given an example or a set of example, more information is required to select which proof has been used and, consequently, $K(E|\ T)$ augments.

5.12.2 Combination of Gain and Reinforcement

In the previous chapter we talked about how intermediate information makes a system better with respect to some evidence, by using $G(E|\ T)$ as a measure of optimality.

Consider again the same equational theory T for addition and product as before:

$X + s(Y) = s(X + Y)$

$X + 0 = X$

$0 \times X = 0$

$$sX \times Y = X \times Y + Y$$

For instance, a rule of commutativity $r_1 = (X + Y = Y + X)$ is a property which can be derived from the previous theory that allows to shorten many derivations in arithmetic. However, the property $r_2 = (X + X + 3 = (X + 1) \cdot 2 + 1)$ is also derivable from it but not so useful in arithmetic. However, both have a high value for G, but only one of them would be worth maintaining explicitly, to lower $G(E| T)$, as it has been discussed in Section 5.10.1.

By using reinforcement, this is even more clear. It is sufficient to select a set of examples of arithmetic practice to ascertain the utility of both rules.

Information gain jointly with reinforcement can also be used to know whether an inductive or deductive effort has been useful. For instance, a new rule can be obtained by induction from the evidence such that $G(r | T) \approx 1$, i.e. with high effort. Despite all the effort, if r does not help to increase the χ of the whole evidence, it has been a vain effort. In the same way, a new rule can be obtained by deduction from the theory such that $G(r | E) \approx 1$, i.e., it has been a hard derivation. Despite all this effort, if r does not help to increase the χ of the whole evidence, it has been again a vain effort.

From here, once we have information about the effort of obtaining r and its utility, we have the possibility to ascertain if r should be maintained or withdrawn, according to the memory limitations of the system.

5.12.3 Forgetting Highly Reinforced Parts

Let us recall the oblivion criterion that was presented in the previous chapter. Given a plausibility criteria $PC(h \mid d)$, and a learner with alternative hypotheses and limited memory resources, its memory politics can be ruled by the following oblivion criterion:

$$OC(h \mid d) = G(h \mid d) \cdot PC(h \mid d)$$

The hypotheses with lower OC should be forgotten.

By selecting $PC(h \mid d) = m\chi(T, E)$, and we have an oblivion criterion. This criterion can be used for rules, since not all the rules which are derived by induction or deduction can be maintained explicitly, and they must be erased from time to time.

Moreover, as long as a theory is constructed from an upcoming evidence $E_{1..n}$, this evidence must be memorised, because if further evidence E_{n+1} refutes the hypothesis, $E_{1..n}$, could be useful to revise and remake a new theory. However, this evidence needs memory, and real systems do not have infinite capacities, so sooner or later some evidence should be forgotten. A first naive idea is to forget the oldest data.

In another way, negative refuting evidence should also be maintained, something suggested by Levi [Levi 1980]. He ascribed *epistemic utilities* to competing hypotheses. The information value of a datum was defined to be the sum of the epistemic utilities of hypotheses disproved by the datum.

A better solution is to realise that as long as more evidence is perceived, some hypotheses are being validated. The examples that have a greater course have less probability that the theory that covers them would be revised. Note also that Levi's ideas are also included, because negative evidence decreases reinforcement in any of the ways seen in section 5.9.

In this way, we can define an oblivion criterion for the evidence:

Definition 5.65 Oblivion Criterion for an Evidence:

Given a theory T and an evidence E, the elements of E should be removed according to:

$$OC(e \mid \langle T,E \rangle) = m\chi(T, e)$$

The original reinforcement of the rules should not be affected by this elimination. This establishes a new situation, because the reinforcement should be incrementally computed and they cannot be recomputed. This introduces an additional complexity in how to treat the cases when two rules r_1 and r_2 cover an example that it is removed. Both r_1 and r_2 should annotate the number of forgotten examples that have been removed. The problem appears if a new evidence makes r_1 inconsistent and it is removed. In this case, r_2 does not deserve the additional reinforcement of the forgotten examples, because they were proven with the help of r_1, which has been removed. An easy solution may be to annotate the rules that are used to prove the examples that are forgotten. This is somehow similar to MOBAL's inference engine IM2 [Emde 1992] [Morik et al. 1993], which maintains derivation information for each statement (fact) in the theory. However, this solution requires a great amount of memory, which is precisely the problem that is to be solved.

This more complex propagation of reinforcement will be left as future work, although some ideas are outlined in an application for software maintenance, which is presented in chapter 9.

5.13 Reinforcement and Theory Understandability

According to many aspects, theories with high and balanced course are reinforced theories, which also explain *all* the data. Consequently, in terms of plausibility, reinforcement turns out to be a good criterion. Methodologically, it usually provides short theories and, due to the way reinforcement is computed, the derivations cannot be much too complicated or involve too many rules. However, there are other methodological criteria whose behaviour with respect to reinforcement deserve to be studied. The most important one is understandability.

[Sommer 1995b] precisely addresses this problem. His intuitions about understandability of theories are the following ones:

1. *Intermediate concepts, sparingly introduced, are a Good Thing.*
2. *Similarly, a deep inferential [hierarchical] structure is more understandable and easier to maintain and modify, than a flat one, because it is more modular.*
3. *The more rules define a concept, the harder it is to grasp.*
4. *Long rules are harder to understand than short rules.*
5. *It is probably not a Good Thing if the encoding of a theory costs more (in some information-theoretic sense) than like encoding of the instances it covers/derives/explains.*
6. *The more variables made reference to in a rule, the harder it is to understand.*
7. *The more non-head variables in a rule, the harder it is to understand.*
8. *The more constants appear in a rule, the less general value it has.*
9. *Non-generative rules are not a Good Thing.*
10. *The less instances a rule covers/derives/explains, the less inclined we will be to accept it (and invest effort in understanding it).*
11. *The more instances are multiply (redundantly) covered/derived/explained, the less inclined we will be to accept the theory (and invest effort in understanding it).*

Intuition 1 is fully recognised by the theory of reinforcement and is also discussed in section 5.10. Intuition 3 is also fulfilled by the character of reinforcement. Intuition 4 is observed due to the extended reinforcement modification made in section 5.7. Intuition 5 has been shown by the connection between reinforcement and compression. Intuitions 6, 7, 8, 9 are concerned with the arguments of clause literals (rule premises) of first-order definite clauses and their correspondence is not applicable here. Intuition 10 is almost exactly the same as the motivation of reinforcement learning.

On the contrary, intuitions 2 and 11 seem to be incompatible with theories with high mean course. At first sight, intuition 2 contradicts the property of reinforcement in which derivations cannot be much too complicated or involve too many rules. In the subsequent reading of [Sommer 1995b], it is realised that Sommer means hierarchical and the word 'deep' is used in order to make a difference with respect to flat theories, which, as we know, are usually little reinforced. Thus, this initial contradiction vanishes. However, there is certainly a problem with highly reinforced theories but it is found in their modularity, because flat theories are usually avoided by reinforcement. Note that Sommer asserts that flat theories are not modular, which, in a strict sense, is false. I am inclined then to understand he means they are not reusable but, in the end, I think that intuition 2 is rather ambiguous. In the end, the problem, in my opinion, must be studied with respect to the topology structure of knowledge, because a hierarchical structure can be either a tree, an inverse tree or any kind of lattice, and these topologies can give very different results of maintainability and modifiability. An approach for studying this topologies is essayed in chapter 9, especially in section 9.2, where reinforcement is applied for different topologies, and the issues about modularity, modification and maintainability are clarified.

Finally, intuition 11, although Sommer recognises that lacks backward reference in the literature (i.e., it is just Sommer's intuition), is indeed contrary to the way reinforcement is computed, because all the alternative proofs are reckoned in the measurement of reinforcement.

If we would like to modify reinforcement to comply with Sommer's latter intuition, we should modify it to penalise the alternative proofs for each example. However, we can do it in several ways. A first idea is to penalise this in the definition of $\rho\rho(r)$ in the following way:

Definition 5.66 The *exclusive* pure reinforcement $\rho\rho^e(r)$ of a rule r from a theory T with respect to some given evidence $E = \{e_1, e_2, ..., e_n\}$ is defined as:

$$\rho\rho^e(r) = \Sigma_{i=1..n}\, card(Proof_r(e_i,T)) \,/\, card(Proof(e_i,T))$$

In other words, $\rho\rho(r)$ is computed as the number of proofs of e_i where r is used divided by the number of proofs of e_i.

However, I think it would be better to include the penalisation in the definition of course, because a rule cannot be punished for being useful. For instance, it would be like penalising the commutativity property because it is used for covering many different evidences.

Another option is to include the penalisation in a redefinition of the course:

Definition 5.67 The *exclusive* course $\chi^e_T(f)$ of a given fact f with respect to a theory T is defined as:

$$\chi^e_T(f) = max_{S\subset Proof(f,\,T)}\ \{\ \Pi_{r\in S}\, \rho(r)\ \}\ /\ card(Proof(f,T)) = \chi_T(f) \,/\, card(Proof(f,T))$$

Finally, another solution may be the use of a measure called "redundancy index" [Sommer 1995b], which is defined in the following way:

Definition 5.68 The redundancy index of a theory T with respect to an evidence E is defined as: **[Sommer 1995b]**

$$Red(E,T) = 1 - card(E) \,/\, \Sigma_{r=1..m}\, \Sigma_{i=1..n}\, nonempty(Proof_r(e_i,T))$$

where for any set S, nonempty(S) = 0 iff $S = \varnothing$ and nonempty(S) =1 otherwise.

In Sommer's words, this measure gives a value of the redundancy of the theory in the following sense: "*ideally, each instance should be covered by only one rule in the theory; if this is the case, [...], Red = 0. The more instances are multiply covered, the [more] Red → 1*". By using this measure as a factor for the course, one could also penalise $\chi_T(f)$.

Although both Definition 5.67 and the use of the redundancy index would comply with Sommer's intuitions, they cannot be used in a careless way. In my opinion, this is a question that is usually addressed heedlessly. One cannot penalise alternative explanations for a given example in a blind way, because this would ban the generation of alternative explanations. In other words, one is surer of a theory when there are not alternative theories for explaining the evidence, but not when she has been unable to (or has penalised) finding alternative plausible theories. Obviously, the certainty of a theory is only increased when the intelligence of the agent is high and he has sought hard for alternative theories. Only in this case, the agent can approximate the certainty of knowing that there are not alternative explanations. This notion of unquestionability (and note that it is not only a methodological but a plausibility issue) will be discussed in following chapter.

5.14 Computing Reinforcement

We have not dealt anywhere about how the theory could be constructed from the evidence (this will be briefly discussed in chapter 7). On the contrary, this chapter has presented a setting for constructive reinforcement learning based on a measurement that allows a detailed study of the relation between the theory and the evidence, for assisting the evaluation, the selection, and the revision of theories.

However, the *measurement* needs to be computed. A general method of computing positive reinforcement is just as it has been used in all the examples which have appeared throughout the chapter:

General Method:

Consider the theory T, with m rules $r_1..r_m$, and the evidence E, with n examples $e_1..e_n$, such that $T \vDash E$. First we must *prove* all the examples and compute $\rho\rho^*$ and ρ^* for each rule. In a second stage, we *prove* again the n examples, computing χ^* from the ρ^* obtained in the first stage.

The complexity of the previous method *seems* to be, in the worst case, in $O(m \cdot n)$. However it is not so, because we have not stated any restriction about the computational cost of the theory, and each proof has its own cost, and there may be more than one proof for each example.

Nonetheless, it would be more realistic to consider the reckoning of reinforcement in an incremental setting:

Incremental Method:

We will use four arrays: $l_{1..m}$, $\rho\rho^*_{1..m}$, $\rho^*_{1..m}$, $\chi^*_{1..n}$ for the lengths, the pure and normalised reinforcements and the courses, respectively. An additional Boolean bidimensional array $U_{1..m,1..\,n}$ assigns *true* to $U_{j,i}$ iff e_i uses r_m in its proof and *false* otherwise.

For each new example e_{n+1} that is received we have different possibilities:

1. If it is a *hit*, we remake $\rho\rho^*_{1..m}$, $\rho^*_{1..m}$, according to the proof of e_{n+1}, U is extended to $U_{\cdot,n+1}$ and $\chi^*_{1..n+1}$ is updated using U.
2. If it is a *novelty* and no revision is made to T, only an extension $T' = T \cup \{r_{m+1}, ..., r_{m+k}\}$, the steps are very similar to the previous case, except that the arrays must be extended to $m+k$.
3. Finally, if it is a *novelty* or an *anomaly* and the theory is revised in some rules $\{r_1, ..., r_p\}$ and extended in others $\{r_{m+1}, ..., r_{m+k}\}$, only the $U_{\cdot,j}$ which does not use any rule from $\{r_1, ..., r_p\}$ can be preserved. The rest must be remade.

The previous method ignores two exceptional cases: that a *hit* would trigger a revision of the theory to readjust reinforcements, and that case 2 may produce alternative proofs for previous examples.

Further optimisation could come from a deeper study of the static dependences (i.e. some rule always depends on others) and the topology of dependences that the theory generates. On the other hand, an appropriate approximation could also be used. Even more, as we have seen before, part of the past evidence can be 'forgotten' if it is covered by very reinforced rules, so avoiding part of the future computations.

However, in the case that an inductive learning method uses reinforcement for evaluating the theories it is constructing, the complexity of these methods would surely be very modest compared to the usual huge costs of machine learning algorithms.

Moreover, reinforcement measures are a very adequate tool to guide a learning algorithm. For instance, in a learning algorithm for logic functional languages based on genetic programming [Hernández-Orallo and Ramírez-Quintana 1998], the examples and rules with low reinforcement are mixed first in order to 'conciliate' them and to obtain more compact and reinforced theories.

5.15 Summary and Contributions of This Chapter

We have presented a framework to distribute or propagate reinforcement into a theory depending on the observation (or evidence). The advantage of this approach is that it makes no assumptions about the prior distribution. Also in this framework, knowledge can have alternative descriptions, without reducing the evidence's courses. Moreover, "deduction in the knowledge" can affect positively to reinforcement, something that the MDL principle or other syntactic priors avoid because the theory cannot change its syntax or representation without changing its a posteriori probability.

These characteristics are related to *reinforcement learning* and some reinforcement or credit propagation systems, such as ANN. Section 2 discusses why the idea of reinforcement has not been applied for constructive languages, i.e., languages with the ability of redescription. Conscious about these difficulties, in Section 3 a first adaptation of reinforcement is presented to realise the problems of 'fantastic' concepts. Section 4 remakes the approach and introduces the idea of 'course' to measure reinforcement, which is shown to be robust to these problems.

Section 5 applies the use of course for the evaluation of inductive theories, and the role of induction, abduction are illustrated. Some other relevant criteria of the inductive literature, such as cross-validation and the MDL principle are shown and related to reinforcement in this section. Consilience, informally referred since Whewell introduced it, is also defined in the context of reinforcement.

Analogy is another inference process that can be studied with the help of reinforcement. Its connection with consilience is established in section 6. Section 7 discusses a more balanced reinforcement criterion which is suitable for explanation, and tries to exclude those parts of the theories which are exceptions, i.e., poorly reinforced. This section serves as an introduction and motivation for the following chapter.

Section 8 discusses the extension of these ideas to wider notions of reinforcement with the presence of reward and penalties, more accordingly to the traditional use of reinforcement learning.

The case of negative evidence is analysed and the framework is conveniently extended in section 9, establishing which properties still hold and which do not. Deduction is addressed in section 10, showing that a deductive inference cannot decrease the course of a theory, quite differently from Carnap's probabilistic calculus. Furthermore, deduction can increase information. The role of intermediate information is highlighted and some questions about plausibility must be reconsidered. Section 11 distinguishes reinforcement as a quantitative (but non-probabilistic) theory of confirmation, somehow between Hempel's and Carnap's. Section 12 relates and combines information gain with reinforcement. The oblivion criterion is adapted to be used with mean course as a plausibility criterion and extended for forgetting the evidence which has been explained, in order to optimise memory resources. Section 13 discusses the relationship between a methodological criterion like understandability and reinforcement.

Finally, some issues about how to compute reinforcement in practice and its complexity are examined in section 14, with the conviction that although it entails an additional cost for any inductive algorithm, it is then counteracted by the advantage of its use as a guide for theory revision, hypotheses rating, evidence management, etc.

One of the most important results of this chapter is that the way we distribute reinforcement into knowledge results in a *rated* ontology, which allows the evaluation of the whole theory or any part of it. In this way, one of the most difficult dilemmas of inductive learning, the choice of a prior distribution, disappears. In other words, it is not necessary to work with probabilities to know the whole and the detailed plausibility of each rule of the theory and each fact that is derived from it.

After this summary, among the relevant contributions of this chapter, we highlight the following:

- Reinforcement allows a more detailed treatment of exceptions and provides different ratings for different parts of a theory, not the single probability value given by priors which is assigned to the whole theory.
- Different predictions or assumptions are provided with different reliability values.
- Reinforcement behaves appropriately for different inference processes such as induction, abduction, analogy and deduction, which are involved in theory construction.
- Some *vague* notions such as consilience and explanatory induction are easily formalised under this framework.
- Intermediate information is shown to be valuable for increasing the course of the theory.
- In the case of non-omniscient systems, new deductively established connections can increase significantly the reinforcement of the whole theory.
- Gain and Reinforcement act as a perfect team to discern which rules should be left explicitly in the representation of a theory.
- An oblivion criterion is derived and extended to manage past and explained evidence.
- By the use of reinforcement as a quantitative measure of confirmation, top-down (deductive) and bottom-up (inductive and abductive) propagation is possible.

Definitely, it is obvious the relation of this framework with the distribution of reinforcement in neural networks, and the problems of overfitting and underfitting in the learning of linear functions (it even resembles some popular algorithms, such as back-propagation). In my opinion, this use of reinforcement for different processes such as induction, abduction and deduction for knowledge acquisition, revision and construction is portable even from expert systems and diagnostic systems to neural networks (training = induction, recognition = abduction). Some applications will be outlined in chapter 9.

However, although the approach which has been presented in this chapter is applicable to constructive theories (for the first time for reinforcement), it would be interesting to extend the results of this chapter to any representational language, not only rule-based languages. The following chapter addresses this problem.

6. Intensionality and Explanation

The great tragedy of Science
—the slaying of a beautiful hypothesis by an ugly fact
T.H. Huxley, 1825-1895

Abstract: *this chapter addresses the problem of formally distinguishing between an extensional definition (or by extension) and an intensional one (or by comprehension). After some approaches of formalisations for logical theories as avoidance of exceptions and in the context of reinforcement, the solution to grasp intensionality for finite concepts is based once again on different concepts based on descriptional complexity. The notions of projectible descriptions and stable descriptions are introduced to account more easily for the notion of intensionality in general. The final approach allows the definition of an explanatory variant of Kolmogorov Complexity, which corresponds to an explanatory counterpart to the MDL principle. Some connections are also established. First, intensionality is closely related to information gain, since extensional descriptions are never informative. Secondly, explanation is also related to the notion of unquestionability, which is given when there are not alternative explanations, a notion that will be necessary for chapter 8.*

Keywords: Extension, Intension, Exceptions, Explanation, Kolmogorov Complexity, Meaning, Information Gain and Intension, Subprogram, General Reinforcement, Plato's Problem.

6.1 Introduction

Classically, logic and philosophy of language have distinguished between definitions by comprehension and extension. Comprehension means the connotation of a term (opposed to its denotation or extension), i.e. the intension, the set of its characteristic properties. The correspondence between the words comprehension and intension is not casual. An *extensional* description (by enumeration) has no connotation and consequently entails no comprehension at all. On the contrary, an *intensional* description (by comprehension), obtained for a given concept, may have not discovered the right meaning or *real* mechanism of the evidence, but still has a chance of it.

In a similar way, methodology establishes [Bochenski 1965], in general, four laws of definition:

1. The definition must be clearer than the thing defined.
2. The definition must refer to the defined thing and only to the defined thing.
3. The defined thing cannot appear in the definition.
4. If possible, it must be affirmative, and not negative.

If we denote by x the definition and by y the defined thing, the first property is accounted by the notion of representational optimality seen in chapters 4 and 5, or, alternatively, by the measure of explicitness which is represented by $G(y|x)$. The second and fourth properties are easily fulfilled by the use of minimality and by the nature of the descriptions that are made by computational systems.

Property 3 has been partially addressed by the theory of reinforcement of the previous chapter, which avoided rules that are extensional patches of the evidence. This was specifically accounted by a balanced evaluation criterion, which strongly penalised extensional rules. However, this measure has only been defined for rule-based languages and cannot be used for this property in general.

It would then be very appropriate to distinguish formally and generically those descriptions that follow this comprehension requirement: "*the defined thing cannot appear in the definition*". This slogan is firmly observed in dictionaries and used by teachers when asking to their pupils, in order to know whether they have comprehended a concept.

Additionally, traditional use in mathematics distinguishes an extensional definition from an intensional definition (or by comprehension). However, this distinction is completely intuitive and a scarce interest has been shown for formalising it, because for infinite sets, frequent in mathematics, every definition must be intensional (or by comprehension). Nonetheless, for *finite* sets, there is still no formal difference between an intensional description and an extensional one. For instance, the set S= { 3, 12, 21, 30, 102, 111, 120, 201, 210, 300 } has infinite many descriptions, e.g. D_1 = S, i.e., the extensional description of S, D_2 = "Start with number 3. The following three numbers are obtained by adding 9 to the preceding one. Continue with number 102. The following two numbers are obtained by adding 9 to the preceding one. Continue with number 201 and add 9 to the next number. Finally, include number 300" and D_3 = "natural numbers of 3 digits or less whose digits in decimal representation amounts to 3". Intuitively, D_3 seems more intensional than D_2 and D_2 more intensional than D_1.

At first sight, Kolmogorov or Descriptional Complexity seems sufficient to distinguish extensional descriptions from intensional ones. However, if we consider the compression ratio, there can be cases where $CR_\phi(p_x) < 1$, i.e., no compression has taken place, and the description p_x can still be intensional (in the previous case D_3 would usually be larger than D_1 if addition has to be coded as well in the definition). More severely, for cases where $CR_\phi(p_x) > 1$, i.e., compression has taken place, it is impossible that the description p_x would be completely extensional (some pattern must have been discovered in order to compress) but it is still possible that the description would be partially extensional (such as D_2).

As a result, the MDL principle, which chooses the shortest description for a given concept *x*, does not ensure that the description is intensional. In the vast majority of cases, the data is not compressible, and the MDL principle will give the data itself, which is the most extensional description, providing no hint about the comprehension of that data. Even in the rare cases where the data is compressible, a short description does not ensure that all the data is described intensionally; there could be a part that could be highly compressed and another part that could be quoted as an exception.

The question is then more conspicuous: is there any way to distinguish pattern from exceptions, program from data? An answer to this question is necessary for a theory of intensionality, which then brings light to many related questions, from the distinction between explanatory induction and descriptional induction to the problem of meaning.

The solution to this question is closely related to some of the notions presented in the previous chapters, as we will show. Chapters 3 and 4 dealt with information gain. In some way, this gain increased when extensionality decreased. It was shown that compression is a good heuristic to attain reduction of extensionality, but, as we have just commented, it can leave several parts in an extensional way. In a different way, chapter 5 introduced a measure of detailed reinforcement or apportionment of credit for representational languages that are composed of rules. The question is whether this idea can be extended to any descriptional mechanism (e.g. Turing machines) where rules, components, parts or subprograms are difficult to distinguish. As we will see, this will require the clarification of the notion of subprogram.

For the rest of this chapter, I will forget the other *aspect* of intensionality more related with meaning until Section 6.6 (where some philosophical issues of meaning and sense will be discussed), and I will deal exclusively about the mathematical notion of intensional description, which I will try to formalise. Let us begin with the idea that gives the name to the term.

6.2 Extensional and Intensional Definitions

It is well known that sets can be described extensionally or intensionally. A finite set A can be described in these two classical ways:

EXTENSIONAL DESCRIPTION: the elements of the set A are enumerated.

$A = \{ a_1, a_2, \ldots, a_n \}$

INTENSIONAL DESCRIPTION: the elements of A are those elements which follow a given property (or predicate function) *p*.

$A = \{ a \mid p(a) \}$

It is obvious that an infinite set can only be described in an intensional way[27].

In general mathematical practice the nature of *p* is left out of discussion, except in intuitionistic (or constructive) mathematics.

A constructive way of seeing the property *p* is to assume that it has to be a computable (or effective) function, i.e., there exists an algorithm that outputs consecutively all the elements of A, halting after them if A is finite. The set is more commonly known as *recursively enumerable*. Also obviously, every finite set is *recursively enumerable* because we can construct the algorithm *from* an extensional description of all the elements in A.

Once at this point, we pose the central question of this chapter. Is there any way of distinguishing whether we have a *pure intensional description*?

First of all, we will have to clarify, at least informally, what we mean with *pure intensional description*, an idea that has only been sketched in the introduction. Let see it with an example.

The set A_1 = { 0, 1, 4, 9, 16, 25, 36, 49, 64, 81} can be easily described intensionally as:

$A_1 = \{ a \mid p_1(a) \}$ being $p_1(a)$ = " $a \in \mathbf{N} \wedge \exists x \in \mathbf{N}: a = x^2 \wedge a < 100$ ".

[27] Note that dots (...) assure that the reader would make a mental intensional model of the series that is intended.

We intuitively say that p_1 is intensional because we see no extensional enumeration in p_1.

However, the set $A_2 = \{ 0, 1, 3, 4, 5, 9, 16, 19, 25, 36, 49, 64, 81\}$ poses serious problems for an *easy* description. A practical description for the preceding set could be:

$$A_2 = A_1 \cup \{ 3, 5, 19 \}$$

And now, we clearly see that A_2 is not purely intensional, some of its elements are described extensionally.

Intuitively, we say that some function is not purely intensional iff it has exceptions, i.e. if we have to "quote" part of the elements extensionally. But this idea is not sufficient. For instance, if we have the set $A_3 = \{ 0, 1, 2, 4, 8, 9, 16, 25, 32, 36, 49, 64, 81\}$ we have alternative descriptions:

$$A_3 = A_1 \cup \{ 2, 8, 32 \}$$

or:

$$A_3 = A_1 \cup \{ a \mid a \in \boldsymbol{N} \wedge \exists\, x \in \boldsymbol{N}: a = 2^x \wedge a < 100 \}$$

In the following we will consider that both descriptions used for A_3 are not pure intensional. However, the detection of the last definition as not intensional will pose more problems. The idea of separability will have to be taken into account, and it must be robust to other kinds of representation such as:

$$A_3 = \{ a \mid a \in \boldsymbol{N} \wedge (\exists\, x \in \boldsymbol{N}: a = x^2 \vee \exists\, y \in \boldsymbol{N}: a = 2^y) \wedge a < 100 \}$$

because it still can be split up without "effort".

Finally, there are many ways to 'camouflage' the extensionality of some function. Also, it is difficult in this context to take the step to the intensionality or extensionality *degree* or *characterisation* of a set (instead of the description), since there are many possible functions to describe a set.

At this moment it is very reasonable for the reader to think that the characterisation that we pretend is so arbitrary, so slippery, that it is not formalisable, just because it *is* subjective. In the end, it is broadly believed that there is no *objective* way to select a description or mathematical concept over another, i.e. only experience and applications can judge their utility. We expect that previous chapters have helped to change that idea, and we will show that descriptional complexity can be used to make the notion of intensionality more objective.

6.3 Exception-Free Descriptions

If we regard intensionality as avoidance of exceptions, then the question is to distinguish what an exception is. Let us see how difficult it is.

Example 6.10

Given the sequence:

x = 1,2,3,5,7

we can guess some short hypotheses or *descriptions* for *x*, simply "the sequence 1,2,3,5,7", or "the first four odd numbers and the number two, ordered", or "the first three natural numbers and the number 5 and 7", or "begin with numbers 1,2. The following two are the sum of their preceding two, and the last is number 7" or "the first five numbers that are only dividable by 1 and itself". Almost everybody would select the last hypothesis as the more explanatory and would predict the number 11 as the next element.

An immediate critique to this example is that the probability of 11 as being the 'correct answer' is the same (or less according to the MDL principle) and it is only the assumption that the concept of prime is well known by most of human beings what increases the value of the answer 11. However there are objective reasons to prefer the intensional description. The shortest description "the first four odd numbers and the number 2" has an exception (the number 2) which "perverts" the hypothesis. In general, the MDL principle has been used successful because the strings are long enough or the bias does, intrinsically, not allow *exceptions*.

Exceptions are useful to memorise, to *describe*, to learn something in noisy situations, but they are not suitable for a *robust explanation*. But how can we eliminate exceptions? Going back to our initial claim, how can we be sure that we have a pure intensional description?

The first idea can be stated informally as, "*an exception is something we can take apart from a description, so leaving it much simpler with respect to the magnitude of the evidence removed or not covered*". More concretely, a description is exception-free if it does not exist a subdescription that produces almost all the data, i.e., there is not a reduction in the description that could be greater that the corresponding reduction in the described data. Let us formalise this.

Definition 6.69 A description p_x for the data x is c-exception-free (denoted $\Delta_c(p_x) = 0$) iff there does not exist a subprogram p_y of p_x, p_y being a program for y and $y \subset x$, such that $K(p_x) - K(p_y) \geq [K(x) - K(y)] / c$. Note that in the case it exists, $p_x - p_y$ is the exception (and p_y the main rule).

The parameter c can be tuned depending on the deductive framework and the approximation for computing K, which can be Kt or simply the length function. In the following, c is assumed to be 1.

Obviously, a formalisation of subprogram is necessary in the deductive framework that would be chosen. As we will see, in the case of logical theories, this question is trivial but, in other cases, it can be very arduous. I will introduce the notion of subprogram in section 6.4.

For the moment, let us see how Definition 6.69 works:

Example 6.11

Consider the facts $F = \{ f_1, f_2, \ldots, f_{10} \}$ and a theory $T_a = \{ t_1, t_2 \}$ that covers these facts in the following way: t_1 covers f_1 to f_9 and, separately, t_2 covers f_{10}. Since t_1 and t_2 are separable, we can check the condition simply as $K(t_2) \geq K(f_{10})$. If it is the case, we say that f_{10} is an exception with respect to T_a. In contrast, we may find a theory $T_b = \{t_1, t_2, t_3)$ longer than T_a that covers the facts in the following way t_1 covers f_1 to f_4, t_2 covers f_5, f_6, f_7 and t_3 covers f_8, f_9, f_{10}. For Definition 6.69, it would be exception-free. It is said that this theory is 'balanced' if $K(t_1) \approx K(t_2) \approx K(t_3)$. Finally, we can consider another theory $T_c = \{ t_1 \}$ longer than T_b which is not only balanced, but t_1 cannot be split up to cover separately subsets of F. That is to say, T_c conciliates F. In this case, it would also be exception-free.

6.3.1 Exception-Free Logic Programs

The preceding definition of exception-free description is general enough to be adapted to any descriptional language. Nevertheless, this generality renders the comparison with other related notions difficult and it cannot be made operative easily without a definition of subprogram.

The advantage of logic programs (and any other rule-based language) is that the notion of subprogram is direct. We just require a proper notion of partition:

Definition 6.70

Consider a program P as a set of Horn clauses with its minimal Herbrand model $M^+(P)$ equal to the set of ground literals L_i such that $P \models L_i$.

P is n-separable into the partition of *different* programs $\Pi = \{ P_1, P_2, \ldots, P_n \}$ iff

$$M^+(P) = \bigcup_{i=1..n} M^+(P_i) \text{ and}$$

$$\forall_{i=1..n} \left(M^+(P_i) \neq \emptyset\right)$$

Definition 6.71

P is *non-subset* n-separable into the partition $\Pi = \{ P_1, P_2, \ldots, P_n \}$ iff it is n-separable into Π and

$$\forall_{i,\, j=1..n} \left(P_i \subseteq P_j \text{ implies } i = j\right).$$

The existence of a non-subset 2-separation can be regarded as a condition to detect exceptions. However, this exception-free condition would be so strict that it would ban any *modularity* in programs. Let us introduce three other variants:

Definition 6.72

P is *disjoint* n-separable into the partition $\Pi = \{ P_1, P_2, \dots , P_n \}$ iff it is n-separable into Π and

$$\forall_{i,\, j=1..n} \left(P_i \cap P_j = \varnothing \right)$$

Definition 6.73

P is *non-subset model n-separable* into the partition $\Pi = \{ P_1, P_2, \dots , P_n \}$ iff it is n-separable into Π and

$$\forall_{i,\, j=1..n} \left(M^+(P_i) \subseteq M^+(P_j) \text{ implies } i = j \right).$$

Definition 6.74

P is *disjoint model n-separable* into the partition $\Pi = \{ P_1, P_2, \dots , P_n \}$ iff it is n-separable into Π and

$$\forall_{i,\, j=1..n} \left(M^+(P_i) \cap M^+(P_j) = \varnothing \right)$$

To show how they differ, we give some examples:

Example 6.12

Given the following program P_1= { p(a). q(X) :- r(X). r(a). } it is separable for all the definitions we have given into the partition Π = {{p(a)} , {q(X) :- r(X). r(a)}}.

The program P_2= { q(X) :- r(X). r(a). } is not separable for any of the definitions we have given.

The program P_3= { q(X) :- r(X). p(X) :- r(X). r(a). } is non-subset (model) separable into Π = {{ q(X) :- r(X). r(a)}, {p(X) :- r(X). r(a). }} but it is not disjoint (model) separable.

The program P_4= { q(a). p(X) :- q(X). p(a) } is non-subset (model) and disjoint separable into Π = {{ q(a). p(X) :- q(X). }, {p(a)}} but it is not disjoint model separable.

The program P_5= { s(X):- p(X), q(b). p(X) :- q(X). t(X):-p(X),q(a) } is non-subset (model) and disjoint separable model into Π = {{ s(X) :- p(X), q(b). p(X) :- q(X) }, { p(X) :- q(X), t(X) :- p(X), q(a) } but it is not disjoint separable.

Moreover, it is trivial to show the following theorems:

Theorem 6.25

If a program P is *disjoint separable* then it is *non-subset separable.*

Theorem 6.26

If a program P is *disjoint model separable* then it is *non-subset model separable.*

At this point, different notions of exception can be given by using Definition 6.70 (single partition), Definition 6.71 (non-subset partition), Definition 6.72 (disjoint partition), Definition 6.73 (non-subset model partition), Definition 6.74 (disjoint model partition) that I will dub *modes.*

Now, the informal definition that was given in general: "an exception is something we can take apart from a program so leaving the program much simpler with respect to the magnitude of the length of the elements removed" can be specialised to Horn logic programs.

Definition 6.75

A program P has e = card($M^+(P_E)$) *c*-exceptions, denoted $\Delta_c(P) = e$, generated from P_E, iff there is a partition $P = \{ P_R, P_E \}$ such that:

$$l(P) - l(P_R) \geq \left[l(M^+(P)) - l(M^+(P_R)) \right] / c$$

Definition 6.75 means that what is reduced in the length (l) of the program is greater than what is reduced in the consequences, but it would be slightly different depending on which of Definition 6.70-Definition 6.74 is used.

The greatest value of c that still makes a program exception-free (i.e., $\Delta_c(P) = 0$) is known as its *consilience* level. On the other hand, when not indicated it is assumed to be 1, and, therefore, a program will be exception-free if its consilience level = 1. Finally, there are many ways to estimate the length of logic programs $l(P)$, but, customarily, a syntactical measure such as those given in chapter 4 can be used.

Let us illustrate the difference between explanatory induction and descriptional induction in an example:

Example 6.13

Given the facts F = { even(0). even(s(s(0)). even(s(s(s(s(0)))), ¬even(s(0)) } the following programs can be induced:

P_1 = { even(0). even(s(s(X)) }, which is the shortest one but it is separable in all cases and *even*(0) is an exception.

P_2 = { even(0). even(s(s(X)) :- even(X) }, which is not separable in any case and logically it has no exceptions.

P'_1 = { even(0) :- *fant.* even(s(s(X)) :- *fant. fant.* }, which is non-subset (model) separable, but it is not disjoint (model) separable. Therefore it has exceptions for the two first modes.

The last program from Example 6.13 shows that a 'fantastic' concept can make a program non-separable for some modes, *hiding* exceptions. It is easy to prove that any separable program in the disjoint modes can be extended to a non-separable program by using a fantastic concept. We say that the concept is not fantastic (it is really consilient) when it must reduce the size of the conciliated part, in a similar way as, in the previous chapter, it should increase reinforcement. This implies that it is impossible to make every program exception-free, i.e., intensional.

Although the non-subset mode alone is too strict and the disjoint mode easy to cheat, the *non-subset* mode *combined* with the value of c=1 for exceptions are appropriate to distinguish a consilient program for most applications. Indeed, different modes and values for c can be combined for various degrees of desired explanatory induction.

The main problem of the definition of exception-free is that it must be computed with respect to the given data (facts), because all the possible consequences can be infinite. Consequently, intensionality is defined relatively to the given evidence, and not in an absolute way.

It is outside of this thesis to take into account the presence of noise, but a degree or ratio of exceptions could be fitted to the expected ratio ε making $\Delta_c(p) = \varepsilon$.

6.4 Subprograms and General Reinforcement

The previous approach illustrates the application of the idea of exception to logic programs. However, it would be interesting to be able to apply it to any descriptional mechanism. In the same way, the previous chapter gave a measure of reinforcement for languages where the notion of rule and subprogram was clear, because it was easy to recognise which parts were responsible for covering each example.

Let us generalise the detailed utility criterion represented by reinforcement and the idea of partition for general descriptional languages (e.g. Turing machines). For this, as we will see, we require a notion of subprogram.

6.4.1 Subpart and partitions

Definition 6.76. Subpart

The object y is a subpart of an object x in β, denoted by $y \subseteq_\beta x$, iff:

$$K_\beta(y|x) < \log K_\beta(y)$$

It is interesting to compare the definition of subpart with the notion of subset. For instance, it is easy to show that the empty string is never a subpart of any non-empty string and that most objects (but not all) are subparts of themselves. It is more intuitive to see the idea of subpart as a cognitive notion, such a subpicture.

Definition 6.77. Proper Subpart

The object y is a proper subpart of an object x in β, denoted by $y \subset_\beta x$, iff $y \subseteq_\beta x$ but $x \subsetneq_\beta y$.

Finally, we define also a partition from Definition 6.76:

Definition 6.78. Partition

A set of objects $Y = \{y_1, y_2, ..., y_m\}$ is a partition of an object x in β iff

$\forall y_i$: $1 \leq i \leq m$: $y_i \subseteq_\beta x$ and there exists an ordering o_j of Y such that

$x \subseteq_\beta y_{o1} \cdot y_{o2} \cdot ... \cdot y_{om}$, where $\cdot$ represents the composition of objects.

Note that the second condition is that x can be reconstructed from the partition.

Definition 6.79. Reduced Partition

The set of objects Y is a reduced partition of an object x in β iff it is a partition of x in β and $\neg\exists Y' \subset Y$ such that Y' is a partition of x.

Definition 6.80. Proper Partition

A set of objects $Y = \{y_1, y_2, ..., y_m\}$ is a proper partition of an object x in β iff

$\forall y_i$: $1 \leq i \leq m$: $y_i \subset_\beta x$ and there exists an ordering o_j of Y such that

$x \subseteq_\beta y_{o1} \cdot y_{o2} \cdot ... \cdot y_{om}$, where $\cdot$ represents the composition of objects.

6.4.2 Subprogram

The idea of subprogram is derived from Definition 6.76:

Definition 6.81. Subprogram (or subtheory)

The object y is a subprogram of an object x in β iff

$$y \subseteq_\beta x \text{ and } \phi(y) \subseteq_\beta \phi(x)$$

Definition 6.82. Proper Subprogram (or subtheory)

The object y is a proper subprogram of an object x in β iff

$$y \subset_\beta x \text{ and } \phi(y) \subseteq_\beta \phi(x)$$

Definition 6.83. Proper Program Partition

A set of objects $Y = \{y_1, y_2, ..., y_m\}$ is a program partition of an object x in β iff Y is a proper partition of x and $Y' = \{\phi(y_1), \phi(y_2), ..., \phi(y_m)\}$ is a proper partition of $\phi(x)$.

From here we could redefine the notion of exception-free by using the notion of subprogram. In other words, Definition 6.69 is now fully formalised.

6.4.3 General Reinforcement

The theory of reinforcement, as it was presented in the previous chapter, was based on the notion of rule necessity for the evidence. A program was composed of rules and the evidence was divided into atomic facts. In the general case, however, the only indivisible part is a bit.

Given a program in any descriptional mechanism (a string p) and the evidence it covers (a string e), a first idea is to change a bit of p and see how many bits of e are changed. This may give an approximation of how useful is each bit of p for covering the evidence, but this solution is not valid for any descriptional mechanism. Apart from the problem that a bit change in the program may make the program not computable, the worst problem is that a bit change in the program may make the program *incorrect* with respect to the representational mechanism and this could assign reinforcement to bits that do not deserve it. For instance, a descriptional mechanism with CRC (Cyclic Redundancy Code) would not allow this kind of measure, because it would be impossible to assign detailed reinforcement values to the theory.

The solution, once again, must be based on a descriptional notion of necessity. In a first approach, we could say that the bit i is necessary for E iff $K(E|p) \neq K(E|p_{\neg i})$, where $p_{\neg i}$ represent the complement of bit i. However, this would usually be the case for every bit of p if p has no redundant information. A refinement of this idea could be to compare $K(E|p_{\neg i})$ - $K(E|p)$ with $K(E_{\neg j}|p_{\neg i})$ - $K(E_{\neg j}|p)$ but this idea gives problem precisely when the program quotes extensionally the evidence.

The final solution is finally based on the notion of subprogram:

Definition 6.84. Bit Subprogram Independence

Given a program p and a subprogram s of it, the bit i of p is independent to s iff

$$K(s|p) \geq K(s|p_{\neg i})$$

In a similar way,

Definition 6.85. Bit Evidence Independence

Given a program p, its evidence e, and a subprogram s of p, the bit j of e is independent to s iff

$$K(e|s) \geq K(e_{\neg j}|s)$$

These two definitions define two dependence arrays, $VP^s(1..m)$ and $VE^s(1..n)$, defined in the following way:

Definition 6.86. Independence Arrays

$VP^s i$ = 0 iff the bit i of p is independent to s.

= 1 otherwise.

$VE^s j$ = 0 iff the bit j of e is independent to s.

= 1 otherwise.

From here, we can finally obtain a correspondence matrix between the bits of the evidence and the bits of the program in the following way:

Definition 6.87. Reinforcement Matrix

$$\mathrm{MPEi,j} = \textstyle\sum_{s \text{ is a subprogram of } p}[\mathrm{VP}^s\mathrm{i} \cdot \mathrm{VE}^s\mathrm{j}] \;/\; \textstyle\sum_{s \text{ is a subprogram of } p} 1$$

The major problem of the preceding definition is that the number of possible subprograms given by Definition 6.81 may be too high. However, the obtaining of this matrix allows directly the definition of general notions of reinforcement and course:

Definition 6.88. General Reinforcement

The reinforcement of each bit *i* of a program *p* is obtained as:

$$\rho(i) = \textstyle\sum_{j=1..n} \mathrm{MPEi,j} \;/\; n$$

Definition 6.89. General Course

The course of each bit *j* of an evidence *e* with respect to a program *p* is obtained as:

$$\chi(j) = \textstyle\sum_{i=1..\,m} \mathrm{MPEi,j} \cdot \rho(i)$$

For instance, consider an evidence composed of 5 bits and a program of 3 bits with the following matrix MPEi,j = { { 0.1, 0.4, 0.7 }, { 0.3, 0.2, 0.8 }, { 0.2, 0.3, 0.7 }, { 0.7, 0.2, 0.4 }, { 0.8, 0.1, 0.7 } }. This gives, for instance $\rho(i)$ = { 0.42, 0.24, 0.66 } and $\chi(1)$ = { 0.42 · 0.1 + 0.24 · 0.4 + 0.66 · 0.7 = 0.6 }.

The following definitions are direct adaptations of the notions that were seen in the previous chapter:

Definition 6.90. Mean General Course

The mean course of an evidence *e* with respect to a program *p* is obtained as:

$$m\chi(e) = \textstyle\sum_{j=1..n} \chi(j) \;/\; n$$

Definition 6.91. Intensionality based on General Reinforcement

There cannot be a bit *j* of *e* such that $\chi(j) < c$, this value depending on the descriptional mechanism.

Definition 6.92. Balanced Description based on General Reinforcement

There cannot be a bit *j* of *e* such that $\chi(j) < c \cdot m\chi(e)$, this value *c* being between 0 and 1 and depending on the descriptional mechanism.

If *Kt* is used instead of *K*, the definitions are effective, although, in general, these definitions are difficult to apply if the descriptional mechanism does not clearly recognise the notion of subprogram. For the problem of reinforcement for general descriptional languages not much can be done in an efficient way.

Let us see, though, another more practical and still general approach. However, this approach is not valid for general reinforcement, only for the idea of intensionality.

6.5 Projectible Descriptions and 'Pattern'

Fortunately, there is another approach for the notion of exception without requiring the definition of subprogram. It is based on the recognition of the structure or projectible part of a description.

First of all, we must formalise what is a projectible description, i.e., a description that can predict future evidence.

Definition 6.93. k-Projectible Description

A *k*-projectible description for a string x is a program p on a descriptional mechanism ϕ such that:

$$\phi(p) = y, \text{ and } \exists w\ l(w) = k : y = xw \text{ (i.e. } x = y_{0..l(x)})$$

w is known as the *prediction* of p.

The compression ratio of an infinite projectible description with respect to its prediction is always infinite. For this reason we must define the relative compression ratio of a projectible description p for a string x with respect to this string x as $CR_\phi(p\,|\,x) = l(x) / l(p)$.

According to the MDL principle, given any sequence x, the optimal model in ϕ for it is x^*. If x^* is projectible, i.e. it allows to predict the subsequent symbols of the sequence x, then $\phi(x^*)_{n+1}$ will be the most plausible prediction according to Occam's razor, "the best model of the world x". However, if x^* is not projectible, this prediction cannot be done. For this reason, we define an ideal MDL principle based on a projectible variant of Kolmogorov Complexity.

Definition 6.94 k-Projectible Kolmogorov Complexity

The *k-Projectible Kolmogorov Complexity* of an object x given y on a descriptional mechanism (or bias) β is defined as:

$$K'_\beta(x|y) = \min \{ l_\beta(p) : \exists w\ l(w) = k \text{ such that } \phi_\beta(<p,y>) = xw) \}$$

where p denotes any "prefix-free" β-program, and $\phi_\beta(<p,y>)$ denotes the result of executing p using input y.

The literature has used Kolmogorov Complexity and not its projectible variant for prediction due to the following theorem:

Theorem 6.27

For every string x, $K'(x) <^+ K(x)$.

PROOF. Every non-projectible program p can be transformed into a projectible program p' = "execute p and then print 1 forever. Let us denote by c the length of this extra coding of "and then print 1 forever". Hence there exists a constant $k=c+1$ such that $l(p') < l(p) + k$, i.e. $l(p') <^+ l(p)$. This can be extended to the definitions of K' and K, thus the theorem is proven. □

The contrary relationship ($K(x) <^+ K'(x)$) does not hold. Consider the string x = "1,2,3, ..., n". The projectible program p' = "print the natural numbers, ordered" has constant size, say $l(p') = c$. On the contrary, the non-projectible program p = "print the first n natural numbers, ordered" is, in the general case, not smaller than $c' + \log n$.

Another question is the projectible extension of Kt complexity. To extend LT-Complexity to projectible descriptions, we must measure Cost(p) in an asymptotical way. Consider a machine ϕ such that the output tape cannot be rectified (or simply it cannot go back). Cost(p)[..n] is defined as the time or machine steps such that the first n symbols of the definite output are placed at the beginning of the output tape. We will also use the following notation: Cost(p)[$n..m$] = Cost(p)[..m] − Cost(p)[..n]. From here we define $LT_\beta(p_x)[n..m] = l(p_x) + \log \text{Cost}(p_x)[n..m]$ and $LT_\beta(p_x)[..n] = l(p_x) + \log \text{Cost}(p_x)[..n]$:

Definition 6.95 k-Projectible Length-Time Complexity

The *k-Projectible Length-Time Complexity* of an object x given y on a descriptional mechanism β is defined as:

$Kt'_\beta(x|y) = \min \{ LT_\beta(<p,y>)[..l(x)]-l(y) : \exists w\ l(w) = k \text{ such that } \phi_\beta(<p,y>) = xw) \}$

Since $LT(<p, y>)$ considers the length of y (the background knowledge which is given), this must be corrected by the term $-l(y)$.

Before using this definition for formalising the idea of exception, we must first recall some approaches in the literature. The idea of projectible description was also addressed by Koppel for a very similar reason to the one of this chapter, the aim of distinguishing pattern from data. More precisely, Koppel introduced the notion of sophistication with the goal of distinguish the structural part of an object [Koppel 1988] from its data or non-compressible part of it. Sophistication is measured by the use of a special kind of Turing Machines ϕ', which separate program from data. Sophistication is then measured as $Soph(x) = \min\{l(p) : \exists d \text{ s.t. } \phi'(p,d) = x\}$ with the restriction that p must be total, i.e., defined for all d. This last restriction precludes that the whole description is passed to the part of data, by maintaining an interpreter i of the data $d' = <p,d>$. According to Koppel [Koppel 1987], "*the sophistication of an object is the size of that part of the most concise description of that object which describes its structure, i.e., the aggregate of its projectible properties. For example, the sophistication of a string which is random except that each bit is doubled (e.g. 00110000110011....) is the size of the part of the description which represents the doubling of the bits*". In our opinion, this interpretation is not exact. In general, for complex objects, sophistication represents the size of too much general programs. For instance, we could use a functional interpreter with syntactical verifications of termination, to interpret the data as functional programs.

This would make $Soph(x) \leq l(i)$ for a great majority of complex objects for which there is a program under this syntactical restrictions, and would leave most of the structure of these objects in the data. As a result, sophistication does not represent the idea of pattern or structure of an object.

Similarly logical depth, as defined by Bennett [Bennett 1988] does not represent the idea of the structure or real complexity of an object. Motivated by the fact that Kolmogorov Complexity gives high values for random strings, which have no pattern and structurally are simple, Bennett introduced the notion of depth, which measures the amount of time required for a string to be generated from its minimal description. But precisely, Koppel showed [Koppel 1987] that "sophistication" and "depth" were equivalent up to a constant. Hence, the previous rationale can be applied to logical depth as well.

Consequently, we need a different approach to distinguish whether any description has exceptions (partially or totally extensional) or it is composed exclusively of pattern (it is all structure or totally intensional). The idea is to compare the part that is used for all the data (to the limit), which is the structure, with the part that is only used in some portion of the data (the exception).

Definition 6.96 Equivalence in the Limit

A description p' is (n,k)-equivalent in the limit to a description p iff

$$\exists n \in \mathrm{N},\ n > 0 \text{ and } \exists k \in \mathrm{Z} \text{ such that } \phi(p')_{n+k..} = \phi(p)_{n..}$$

Informally, two descriptions are equivalent in the limit if there is a point from which their predictions always match. If both descriptions are k-projectible with k finite they are always equivalent in the limit, if only one of both is ∞-projectible then they cannot be equivalent in the limit. Hence, the definition applies when both descriptions are ∞-projectible descriptions.

Definition 6.97 Fully Projectible Description

A description p is a fully projectible description of x given y iff $<p,y>$ is an ∞-projectible description of x and $\neg\exists p'$ such that

1. $<p',y>$ is (n,k)-equivalent in the limit to $<p,y>$,
2. $<p',y>$ not extensionally equivalent to $<p,y>$ and,
3. $LT(<p',y>)[n+k..n+k+l(x)] < LT(<p,y>)[n..n+l(x)]$.

The second condition that p' is not extensionally equivalent to p is for avoiding that given two or more equivalent descriptions, only the shortest one would be fully projectible[28]. The third condition measures that this p' is simpler than p. Note that LT (and only applied to the first chunk of length $l(x)$ where p' and p begin to be

[28] This condition could be removed or bounded (as well as equivalence in the limit) if one wants to make the definition computable.

equivalent) is used instead of *l*. Let us recall our previous example with the evidence as “3, 12, 21, 30, 102, 111, 120”, we can consider several projectible descriptions. For instance, D'_1 = “3, 12, 21, 30, 102, 111, 120 and 1 forever” is not fully projectible because there exists a shorter description “1 forever” which is equivalent in the limit. In the same way, D'_2 = “Start with number 3. The following three numbers are obtained by adding 9 to the preceding one. Continue with number 102. The following numbers are obtained by adding 9 to the preceding one” is not fully projectible because there exists a shorter description “Start with number 3. The following numbers are obtained by adding 9 to the preceding one” which is equivalent in the limit. On the contrary, the description D'_3 = “numbers whose digits in decimal representation amounts to 3 ordered” is fully projectible. Similarly, the description D'_4 = “repeat 3, 12, 21, 30, 102, 111, 120 for ever” is fully projectible. Finally, the following description is also fully projectible D'_5 = “the y values of a polynomial $y = P(x)$ taking for x the natural numbers” where P is a polynomial such that P(1) = 3, P(2) = 12, ..., P(7) = 120.

The last two descriptions seem counterintuitive but, in some way, this is something logical, since a fully projectible description formalises the idea of explanation (and not the comprehension requirement): it describes the evidence, it accounts for all of it (there are no exceptions because it is fully projectible) and it can be related to others (because of the use of LT). And D'_4, whether we like it or not, is an *explanation* for the evidence.

6.6 Intensionality, Informativeness and Explanatory Induction

Kolmogorov Complexity has been used as “a perfect theory of induction” [Solomonoff 1968]. However, the problems of the MDL principle for explanation are notorious, as they were seen in chapter 2.

We could now define variants of Kolmogorov Complexity based on the previous notion of exception-free description:

Definition 6.98 The *Intensional Complexity* of a string x on a bias β, denoted $E\beta(x)$, is defined as follows:

$$E_\beta(x) = \min \{ l_\beta(p_x) : \Delta(p_x)= 0\}$$

i.e., the shortest program for x without intrinsic exceptions. $l_\beta(p_x)$ denotes the length of p_x in β.

There can be short intensional descriptions whose computational cost would be so high that they are of little use as theories. In addition, Definition 6.98 turns out to be non-computable (such as $K(x)$). An explanatory variant of intensional complexity can be defined in the following way:

Definition 6.99 The *Explanatory Complexity* of a string x on a bias β, denoted $Et\beta(x)$, is defined as follows:

$$Et_\beta(x) = \min \{ LT_\beta(p) : \Delta(p) = 0\}$$

There are good reasons to choose a time-weighted definition of the best explanation. The intuitive view of explanation entails that the hypothesis can be *explained* to others. At the moment a system has to *tell* or communicate the explanation to other system (or internally work with it), there are two important topics: the space of the discourse and the time the system will need to relate it. Moreover, people and Science expect that nature has underlying mechanisms that emerge 'quickly' in our observations, simply because nature is not a reliable computer for executing long programs.

The previous definition has the problem of detecting subprograms (in order to obtain $\Delta(p)$), a thing which has been shown to be extremely difficult and language dependent. Fortunately, according to the notion of projectibility we can give an alternative definition:

Definition 6.100 Explanatory Complexity (Projectible Version)

The *Explanatory Complexity* of an object x given y on a descriptional mechanism β is defined as:

$$Et_\beta(x \lfloor y) = \min \{ LT_\beta(<p,y>)[..l(x)] - l(y) \text{ s.t. } <p,y> \text{ is fully projectible } \}$$

The string y, which we have supposed empty in the previous example, represents the context or previous knowledge where the explanation must be applied. In the following, β will be omitted. In the same way it is done with K and the MDL principle, we can denote with SED($x \lfloor y$) the Shortest Explanatory Description for x given y, i.e. the first shortest fully projectible (in lexicographic order) description for x given y. Logically, $l(SED(x \lfloor y)) = Et(x \lfloor y)$.

Note that due to the effect of this easy projectibility shown by description D'$_4$ of the previous example we have that $Et(x) <^+ l(x) + \log l(x)$, something that also held for $Kt(x)$. However, in general $Et(x)$ and $Kt(x)$ may differ significantly, because although there are many ways to hide extensional data by using an intricately coding (in order to feign an intensional description), this must take some space and/or time.

However, we still have that for most strings, SED(x) will be just the *rote* description "repeat x forever" which does not entails any comprehension. A first idea to avoid this phenomenon is to force the description to be shorter than the data and to say that the data has no explanation if this is not the case. However, most of everyday data is not compressible and it is still comprehended.

Another approach is to exclude the descriptions that are generated by *rote* learning, i.e. the extensional repetition of part or all the data. This idea is not new and two evaluation criteria such as reinforcement and cross-validation are inspired in it.

For instance, if we remove the last element of the previous series, i.e. "3, 12, 21, 30, 102, 111", it is not much expectable that D'_4 and D'_5 would be produced but D'_3 can still be generated. In general[29],

Definition 6.101 Stability on the Right

A string x is *m-stable on the right* in the descriptional system β iff

$$\forall d,\ 1 \leq d \leq m : \mathrm{SED}_{\beta}(x_{-d}) \text{ is extensionally equivalent to } \mathrm{SED}_{\beta}(x)$$

In other words, a string x is *m-stable on the right* if taking m elements from the right, it still has the same best explanation. These m elements, if given a posteriori, are considered reinforcement or confirmation of the explanation, and, if given a priori, are considered redundancy or hints to help to find the explanation.

Consequently, although rote learning can be trickily used to make an extensional description fully projectible, reinforcement or cross-validation is shown to be a methodological criterion to avoid this phenomenon.

There is still another reason to support the previous notion of comprehension/intensionality as an ontological principle. Why must we avoid rote learning? Why must we anticipate? Why do children find more complex patterns? [Marcus et al. 1999] Why are we genetically programmed to open any black box we are presented? This search for more informative hypotheses instead of the easiest ones may lead to fantasy, but this is not dangerous as the system can interact with the world in order to refute the hypotheses.

This informativeness or investment in the hypotheses was advocated by Popper for the scientific method, and as we have seen, it is equally applicable for cognition. Even if we make the very strong assumption of Occam's razor, i.e., things in nature are not complex unnecessarily, the previous rationale is justified by the fact that, as well as every incompressible string has compressible substrings, *most* compressible strings have incompressible substrings by their own, because the shorter the less worthy that is to compress. If the evidence is presented incrementally, it is better to invest in more informative or general hypotheses instead of finding the optimal one for each chunk that finally will turn out to be not part of the whole description of the whole evidence. This rationale is further justified by the following theorem:

Theorem 6.28 Anticipation

For every descriptional mechanism β, there exists a constant c which depends exclusively on β such that for every string x of length n with $\mathrm{SED}(x) = x^*$ and $l(x^*) = m$ s.t. $m < n$, then any partition $x = yz$, $l(y) < m - c$ such that $\mathrm{SED}(y)$ is not equivalent in the limit with x^*.

[29] This definition is particularised to sequences. Hence, the stability is measured with respect to the last symbols. In the general case of cross-validation, a subset of elements is removed for obtaining the hypotheses and then validated with the rest.

Proof

Consider any string x and SED(x) = x^* with $l(x^*) = m$ s.t. $m < n$. Take ANY prefix y such that $l(y) < m - c$. It has a projectible description p_y = "print y for ever" with $l(p_y) = l(y) + c' < m - c + c'$, this constant c' being the space which is required for coding "print .. for ever". Since the computational cost of p_y is linear, say $k' \cdot l(x)$, it is sufficient to choose $c \geq c' + \log k'$ to ensure that the description p_y is shorter than x^*, and $LT_\beta(p_y)[..l(x)] < LT_\beta(x^*)[..l(x)]$ because $\log k' \cdot l(x) = \log k' + \log l(x)$. Moreover, p_y and x^* cannot be equivalent in the limit because x^* is fully projectible and, by definition, there does not exist a description with less LT equivalent in the limit. □

It is clear that the idea of stability or cross-validation is supported by the previous theorem. In fact, it is an innate *aesthetic* preference in the explanations that human beings generate. Why is it more pleasant to give the answer 23 to the series "3,7,11,15,19,..." than to give the answer 3?

The definition of stability and the previous theorem serve as a formalisation and justification of intensionality, respectively. However, the projective character of Definition 6.101 and its avoidance of rote learning, make it a first criterion to detect when *comprehension* has taken place.

As a result of this section, stable objects give SED descriptions where comprehension has taken place, i.e., comprehensive descriptions.

6.6.1 Descriptive vs. Explanatory Induction

As it was seen in chapter 2, the principle of simplicity, represented by Occam's razor, selects the shortest hypothesis as the most plausible one.

It is remarkable (and often forgotten) than Kolmogorov Complexity just gives consistency to this theory of induction, but Occam's razor is *assumed*[30] but not proven. Nonetheless, some justifications have been given in the context of physics, reliability and entropy, but, in my opinion, it is the notion of *reinforcement* (or cross validation) which justifies the MDL principle in a more natural way. As we saw, the higher the mean compression ratio the higher the mean reinforcement ratio.

The problem of the MDL principle for explanation is that for the sake of maximum mean compression, some part of the hypothesis cannot be compressed at all, resulting in a very compressed part plus some additional extensional cases. This extensional part is not validated, making the whole theory weak.

[30] Furthermore, in the case the universal distribution $2^{-K(x)}$ is assumed, giving a priori predilection of short programs, the a posteriori optimality of the MDL principle is proven, supposing the *randomness of the hypothesis to the data* [Vitányi & Li 1997]. But precisely in explanatory prediction, if the hypothesis is random to the data, it cannot be the cause!

Summing up, the MDL principle says that, in absence of any other knowledge about the hypotheses distribution, we should select the prior $P(h) = 2^{-K(\mathrm{h})}$. For explanatory induction I propose to use $P(h) = 2^{-Et(\mathrm{h})}$ instead. This principle has been dubbed the shortest explanatory description (SED). In this way, priority is given to the avoidance of extensionality over simplicity. This complies with Chaitin's view of the scientific method: "*Scientists consider the simplest theory to be the best one, and that if a theory is too "ad hoc", it is useless*" [Chaitin 1974]. A compromise between both thing is represented by SED.

There are other approaches to finding intensional theories. Wexler claimed that the subset principle was an intensional principle [Wexler 1992], for the case of positive data only. The subset principle (also known as Least General Generalisation (lgg) by Plotkin [Plotkin 1970]) means that if two theories explain some positive data, we should select the more specific one, because it is the more informative (and the more falsifiable). The problem of the subset principle is that it must be combined with some simplicity criterion, because, if not, the more specific hypothesis is the data themselves, which is completely extensional.

6.6.2 Unquestionability

It has been frequently argued in philosophy of Science and induction that the plausibility and unquestionability of a theory or explanation not only depends on the intrinsic characteristics of the explanation but also to the ability of finding alternative explanations.

Let us make formal and objective this idea. At first sight it seems that stability avoids this but, if we restrict to stable descriptions, we can still modify any explanation p with the addendum "Execute p but print a '1' every hundred symbols are printed" which would be comprehensive for the data but would differ from p in the limit, and would be only a little longer.

For this reason, we must extend the previous notion of stability and apply it to descriptions:

Definition 6.102 Plausibility on the Right

A fully projectible description p for a string x is *(c,m)-plausible on the right* in the descriptional system β iff

$$\forall d,\ 0 \leq d \leq m : LT_\beta(\mathrm{SED}_\beta(x_{-d}))[..l(x_{-d})] + c > LT_\beta(p)[..l(x_{-d})].$$

Intuitively, a description is plausible if it is one of the c-best explanations for x and this holds even if we remove up to m elements from the right of x.

Once the notion of stability has been extended, we can face unquestionability in the following way:

Definition 6.103 Unquestionability

A fully projectible description p for x is *(c,m)-unquestionable* in the descriptional system β iff it is *(c,m)-plausible* and there does not exist another *(c,m)-plausible* description p' for x.

This is a more restrictive condition as c and m are greater. In order to augment these two parameters and still have some unquestionable descriptions we must make the strings larger. For instance, if the length of the portion "print a '1' every hundred symbols are printed" is c', then, in order to obtain a (c,m)-unquestionable description with $c > c'$ we would have to increase the length of x over 100 symbols.

6.7 Information Gain and Intensionality

One may think that informativeness, in the sense of Popper, and intensionality are quite the same thing because both avoid extensional descriptions. However, computational information gain is quite different from intensionality. G implies that the theory is creative or informative. On the contrary, intensionality avoids patches and extensional exceptions in the theory. As we have said, though, the construction of the n–1 order polynomial for n points of data is a systematic method, so there is always an 'easy' intensional description for any evidence. The major coincidence between intensionality and a high value of $G(x|y)$ is that extensional quoting is avoided, as the next theorem shows:

Theorem 6.29

Given an efficient description x for a long data y, such that x contains a sequential quoting Q of a random sequence q from y of significant size, namely, $l(q)=e > \log^2 l(y)$, then x is not intensional and $G(x|y) < 1-e/l(x)$.

For instance, 1,000 bits of data with a description of length 200 bits that contains a sequential quoting of 120 bits is not intensional and $G(x|y) < 0.4$.

PROOF. Since Q is a quoting such as "Print $y_k, y_{k+1}, ..., y_{k+e-1}$" then $CR(Q) = e / \{mf_q \cdot e + af_q\} \leq 1$. The first assertion, x is not intensional, is obvious by choosing Q as the exception and the rest of T removing Q as the general rule G.

Then, since $n > 1$, the compression of the whole theory $CR(T) > 1$, then $CR_{\phi(T)}(G) \geq CR(T)$ because $CR(Q) \leq 1$, and $l(\phi(T) - \phi(G)) / \{mf_q \cdot (l(T) - l(G)) + af_q\} \leq 1$, because the first term is precisely $CR(Q)$.

For the second assertion, we begin from the definition of gain, $G_\beta(x\,|\,y) = Kt(x\,|\,y) / Kt(x)$. Since there is a part of x which is exactly in y, it can be recognised from the input y by only selecting the beginning of the sequence in y and the length e. Coding this information $Kt(q\,|\,y)$, in any case, cannot be greater in length than $\log(l(y)) + c_l$, because a position can be coded by a usual digital notation and it cannot be greater in time than $l(y) + c_t$, to traverse the sequence y. Jointly, we have that $Kt(q \mid y) \leq \log(l(y)) + c_l + \log(l(y) + c_t) = 2 \cdot \log(l(y)) + c_{lt}$. Since y is long, c_{lt} can be ignored.

Since q is random, $Kt(q) \geq l(q) + \log l(q) = e + \log e \geq e$. The term $Kt(x)$ can be decomposed into the cost of describing q and the code of describing the rest, say g, namely, $Kt(x) \cong Kt(g) + Kt(q)$. However, $Kt(x \mid y)$ is exactly $Kt(g \mid y) + Kt(q \mid y)$. Since $Kt(g \mid y)$ is always less or equal than $Kt(g)$ and we have stated that $Kt(q \mid y) \leq 2 \cdot \log(l(y))$ then $Kt(x \mid y) \leq Kt(g) + 2 \cdot \log(l(y))$. From here, $G(x \mid y) = Kt(x \mid y) / Kt(x) \leq \{Kt(g) + 2 \cdot \log(l(y))\} / \{Kt(g) + Kt(q)\} \leq \{Kt(g) + 2 \cdot \log(l(y))\} / \{Kt(g) + e\} = \{Kt(g) + 2 \cdot \log(l(y)) + e - e\} / \{Kt(g) + e\} = 1 - \{e - 2 \cdot \log(l(y))\} / \{Kt(g) + e\}$. Since $e > \log^2(l(y))$ and l(y) is long enough we can ignore the term $\log(l(y))$, giving $G(x \mid y) \leq 1 - e / \{Kt(g) + e\}$

Since $Kt(g) + Kt(q) = Kt(x)$, by using again the value of $Kt(q)$, then we have that $Kt(g) \leq Kt(x) - e$ and we finally have that $G(x \mid y) \leq 1 - e / \{Kt(x) - e + e\} = 1 - e / Kt(x)$ and since $\mathit{log}\, l(x) \leq Kt(\mathrm{x}) \leq l(x) + \mathit{log}\, l(x) \approx l(x)$ then $G(x \mid y) \leq 1 - e / l(x)$. $\square$

Apart from these commonalties between gain and intensionality, they express quite different but compatible notions, which are worth combining. The idea is to obtain explanatory descriptions and to preserve those which are valuable in terms of computation gain. In other words, free computational resources (time and space) should be invested in informative hypotheses.

As a result we are able to counter two assertions from the advocators of the MDL principle. Their first claim is: "*a model that is much too complex is worthless, while a model that is much too simple can still be useful.*" [Grünwald 1999]. My response is that a model that is evident or extensional is worthless, while a surprising model or intensional can still be useful. In the same line, Grünwald presents "*another way of looking at Occam's Razor*" as "*If your overfit, you think you know a lot but you do not. If you underfit, you do not know much but you know that you do not know much. In this sense, underfitting is relatively harmless while overfitting is dangerous*". However, since *most* of data sequences are incompressible, the MDL principle gives no knowledge at all, in *general*. Maybe not knowing, i.e., ignorance, is relative harmless, but it is also useless.

In conclusion, the MDL principle works well in those environments where the bias does not allow extensional descriptions or where the data is huge and from statistical or imperfect sources. But, when faced to a concrete learning problem or in scientific discovery, we have to tune length, computational time, intensionality and informativeness of descriptions according to the expectation we have about the source of knowledge. In our view, Occam's Razor should be understood in this non-autistic way.

6.8 Intensionality, Learning, and Meaning

Chomsky [Chomsky 1986a] states that grammatical principles are *intensional*, not *extensional*. He says they are part of the *I*-language, which includes a "universal grammar" and some other properties acquired by experience. He even argues that there may not even be a coherent notion of *E*-language.

The question then was centred in discovering how children acquired these *I*-language from an evidence that is mainly extensional. This is just the problem of learning human language, which it is just like any other learning problem with some particular traits. Concretely, it is especially troublesome the so-called "Plato's problem" or what Chomsky called [Chomsky 1966], following Descartes, the problem of the "poverty of stimulus". This is one of the main questions of linguistics and learning, because it has been long accepted that children receive little negative evidence [Brown and Hanlon 1970], [Wexler and Culicover 1980], and, it has been shown that learning from positive data only is much harder than learning from positive and negative evidence [Angluin 1980]. This led Chomsky to think that children had innate principles of grammars, a *universal grammar*, because, if not, it would be impossible for them to learn human language.

There have been several proposals to address "Plato's problem" or the case of learning from positive data only [Muggleton 1984], the MDL principle also being among them. Another approach has been the subset principle: "every structure that is grammatical under *A* is also grammatical under *B*, then choose option *A* if it is consistent with the input", i.e., one would select the most specific theory in order to avoid overgeneralisation. In [Wexler 1993] it is even argued that "the subset principle is an intensional principle". However, this approach has had many problems when formalised because the most specific theory is always the data itself.

I propose a solution to this problem as a combination of the notions of intensionality in this chapter with Wexler's notion of 'intensionality', understood as avoidance of generality. A good principle for positive evidence only would be to select the most specific theory that is intensional, i.e., does not have many exceptions. This solution will be essayed in the next chapter, although not for natural language problems.

Fortunately, recent results [Marcus et al. 1999] have shown that children are able to learn virtually any pattern, which discredits the theory of innatism. Moreover, it shows that children usually overgeneralise, and find more intensional (without exceptions) descriptions, such as the over-regular formation of past tenses in English.

Although in inductive inference is not fair to select a selection criterion as we have commented, this could be different for the problem of learning human language. Quite possibly there is not a universal grammar as Chomsky postulated, but it may exist a concrete and innate selection criterion which is applied by children in order to learn human language.

In our opinion, this criterion could be closely related to intensionality, to the idea of comprehending the whole evidence, the whole sentence. As Hofstadter pointed out [Hofstadter 1979]: "*It would be nice if we could define intelligence in some other way than "that which gets the same meaning out of a sequence of symbols as we do". [...] This in turn would support the idea of meaning being an inherent property.*"

This, indeed, is not only applicable for the first years of learning human language but it is necessary for understanding context, sense and intention for the rest of life. Chapter 8 will inquire on this idea of understanding and the relationship between intensionality and the notions of explicit and implicit, based on information gain. At the end of chapter 9 other issues about language and communication will be discussed.

Finally, there is another use of the terms intension and extension which is more related with philosophy of language, following classical definitions of these terms [Cohen and Nagel 1993]: "*A term [an element of a proposition] may be viewed in two ways, either as a class of objects (which may have only one member), or as a set of attributes or characteristics which determine the objects. The first phase or aspect is called the denotation or extension of the term, while the second is called the connotation or intension.*"

There have been many philosophical approaches (Kripke, Carnap, Montague and others) to formulate logics that could account for assertions and other expressions which abound in natural language whose meaning depends on an implicit context or index, such as time or spatial position. These logics are called intensional (or indexical logics). One of the most famous formalisms is the (Kripke-style) indexical semantics, in which context sensitive expressions are interpreted as denoting values that vary over a space of "possible worlds".

A concrete approach in this line is the so-called "intensional logic" [van Bentheme 1988] which studies such 'intensional' phenomena in human reasoning as modality, knowledge, or flow of time. These all require richer semantics than standard truth values in one static environment, including modal logic, tense logic, and conditional logic, all of which illustrate motivations coming from philosophy and linguistics.

This engages with the important connection between intension and intentionality, for many semantical systems, such as extensional model theory, which are limited to extensions, and cannot provide plausible accounts of the language of intentionality. However, an account of intention falls out the scope of this thesis.

6.9 Summary and Contributions of This Chapter

The notion of intensionality is closely related with that of information gain and reinforcement. A completely extensional theory has no information gain and it is not reinforced. However, the question is more complicated when it is realised that there can be partial extensionalities in the theory, i.e., exceptions. How to detect and measure them has centred the interest of this chapter. The idea of compression is related to the notion of intensionality but it still allows partial extensionalities, so they differ significantly in general. Nonetheless, it has been shown that it is impossible to determine the notion of intensionality without the idea of simplicity or descriptional complexity.

Section 2 presents in an informal way the distinction between intensional definitions and extensional ones. Section 3 formalises the notion of exception and consequently the idea of exception-free description. The notion is particularised for logical theories (or any model and rule based language).

In section 4, the previous notion is generalised for any descriptional language. In order to do this, it is necessary to formalise previously some blurred notions such as subpart and subprogram. This generalisation allows to define a measure of general reinforcement, too.

Section 5 introduces a different approach for distinguishing between pattern (or structure) and data, under several refinements over the idea of projectible descriptions.

From the initial notion of exception-free description and that based on projectible description, several variants of descriptional complexity are introduced in section 6, such as intensional complexity and explanatory complexity. The distinction between descriptive induction and explanatory induction is illustrated and an anticipation theorem is proved, suggesting the seek for more informative theories instead of the shortest ones, something that was already advocated in chapter 5.

Section 7 establishes the connection between the notion of information gain of chapters 4 and 5 with the notion of intensionality.

Section 8 examines the relationship between the notion of intensionality presented in this chapter and the original sense of the word intension, more related with language and meaning.

In the end, this chapter does not only contributes with a theory of intensionality but many other interesting concepts have been introduced along the way:

- The idea of intensionality is formalised in terms of avoidance of exceptions, these seen as extensional or non-validated parts of a theory.
- The idea is directly applied to logical theories.
- The porting of intensionality and reinforcement to any descriptional language is essayed, based on a formal and general definition of subprogram.
- Different concepts based on descriptional complexity are introduced, such as projectible descriptions and stable descriptions, which will be useful to grasp the implications of the term 'comprehension'.

- The definition of an explanatory variant of Kolmogorov Complexity allows to define an explanatory counterpart to the MDL principle.
- It is formally shown that intensionality is closely related to information gain, since extensional descriptions are not intensional nor informative.
- Explanation is also related to the notion of unquestionability, given when there are not alternative explanations.

This chapter closes the introduction of new measures and notions, as well as their theoretical comparison, which began in chapter 3. The following chapters apply these measures for very different purposes and compare them in a more practical way.

Within the specific applications of intensionality, we will see in chapter 9 the application of intensional / extensional parts of a database.

7. Evaluation and Generation of Inductive Hypotheses

Proporció és bellesa en l'ordre de les mesures
Ramon Llull, Proverbis, II, CLIX, 1.

Abstract: *this chapter shows the application in practice of most of the measures that have been developed in the previous chapters. The most classical criteria for the evaluation of Logic Programs, especially two variants of the MDL principle, are compared with reinforcement, intensionality and gain. In terms of plausibility, reinforcement is shown to be manifestly better than the MDL principle, either for whole positive evidence, partial positive evidence and partial positive and negative evidence. Intensionality shows in which degree the data is 'conciliated' by the theory, and in some cases it could be seen as a prerequisite (abduction, explanatory reasoning, etc.). Finally, for the case of evaluation, gain is used to know when a real learning has taken place, i.e., the theory is original with respect to the data. Apart from evaluation, the question of how reinforcement and gain can be combined for guiding a machine learning algorithm is discussed. First, it is shown that both enumeration algorithms and randomised data-driven approaches are compatible with an increase of gain. Secondly, a data-driven approach can be constructed with the help of genetic programming, where the selection criterion (oblivion criterion) is a combination of the optimality of the program (the individual) and the gain (unusual or rich genotype).*

Keywords: Logic Programming, ILP, Machine Learning, Inductive Algorithms, MDL principle, Genetic Programming, Enumeration Algorithms, Space Limitations, Generality, Learning from Positive Evidence.

7.1 Introduction

The previous chapters have been introducing different concepts and ideas about informativeness of theories, reinforcement, and intensionality. Many of the applications have already been presented when these concepts were developed. Among these applications, we saw the difference between explicit and implicit, the recognition of what is to discover, the notions of authentic learning and hints about the problem of creativity, the evaluation of deductive systems, especially logical theories, a measure of detailed reinforcement, for measurement or ontological purposes and, finally, a clarification of the notion of explanation and comprehension based on intensionality. Nonetheless, it would be strange that a work on fundamental issues of inference processes would not have even more applications, and, desirably, of more practical character.

This chapter and the following two present a series of instrumentalisations and applications of the most relevant notions and constructions seen in this work. Concretely, this chapter centres on *evaluation*, which, logically, is the first direct application of every measure. We compare different measures for the evaluation of inductive theories, expressed in a logical framework. The classical notion of generality, size complexity, the MDL principle based on model complexity and the MDL principle based on proof complexity, which have been studied [Conklin and Witten 1994] [Sommer 1995b] and applied [Muggleton et al. 1992], [Muggleton and Page 1994]) for ILP, are compared with the measures which have been introduced in this work: information gain, reinforcement, and, intensionality. We will consider the case of learning from whole positive evidence, partial positive evidence and partial positive and negative evidence. The results are quite revealing; reinforcement (conveniently weighed with generality) is the best criterion by far, and information gain and intensionality are also interesting measures to evaluate hypotheses. Moreover, in the case of noisy evidences, where the MDL principle has behaved more successfully, it will be seen that this is precisely because of its blindness. Instead, a measure such as reinforcement that can give an error ratio should be used to obtain more accurate results.

Evaluation criteria can be considered ontological/epistemological or methodological. Some epistemological criteria are useful as methodical criteria, just as some methodological criteria turn out to be epistemological as well. For instance, the MDL principle says that the shorter the more probable but simplicity is also convenient for methodological purposes. It is precisely the methodological part which still shows more advantages for our evaluation criteria. In section 3 we discuss how the search for hypothesis must be tackled by the use of the criteria shown in the previous section. Informativeness (the effort invested) is used when the theories are to be pruned because of space necessity. Consilience can be used to join rules in a learning algorithm. Finally, reinforcement gives a detailed evaluation for each rule of the theory, and thus allows discerning which rules are to be revised, as we saw in chapter 5. As it was also seen, some evidence can be forgotten without problems, depending on how well covered it is.

7.2 Evaluation of Inductive Logical Theories

In chapter 4, when dealing with information gain and deduction, we introduced some measurements to evaluate a logical theory, such as the proof complexity and the model complexity. We used them to evaluate the best axiomatic theory. In this section we will use some of them and others to evaluate inductive logical theories.

In the paper "*Complexity-based Induction*" [Conklin and Witten 1994], a slanted (in my opinion) comparison of evaluation criteria is presented, concretely, between the MDL principle based on model complexity and the MDL principle based on proof complexity. We will counteract the results of that article with much more evaluation criteria and the same examples, although we will consider more theories for them, theories that were not considered by Conklin and Witten because their conclusions would have been less conspicuous.

In the following, I will illustrate the application of the measures introduced by Conklin and Witten, some other measures not considered by them (but clearly better) and the different measures that have been presented in this work.

The measures we are going to consider are: generality degree of the theory, the length of the theory (or Covering MDL), Descriptional MDL based on Model Complexity, Descriptional MDL based on Proof Complexity, Reinforcement, Intensionality and Information Gain. Our distinction between covering MDL and descriptional MDL is that of Conklin and Witten and it is necessary to clarify when the theory can cover more facts than those given by the evidence (i.e. the theory generalises the evidence). This is desirable up to an extent. By Covering MDL it is meant the shortest theory which covers the evidence. By Descriptional MDL it is meant the shortest way to code the evidence, i.e. to transmit it, and overgeneralisations are penalised.

The comparison of all these criteria will be empirical. Although, in the following, we will consider only one example, the reachability relationship, in [Hernández-Orallo and García-Varea 1999] a specific theoretical discussion about the non-informativeness of the MDL principle can be found.

Before making the comparison we must introduce some of the measurements that have not been presented yet: Generality, Descriptional MDL based on Model Complexity and Descriptional MDL based on Proof Complexity.

7.2.1 Generality Measures: $GD(T|E)$ and $g(H)$

The first method that was used for induction of logic programs was the relative least general generalisation (*rlgg*) introduced in the late 1960s by Reynolds and Plotkin [Plotkin 1970]. The reason for using the most specific generalisation is to avoid overgeneralisation. As we saw in the previous chapter, the *subset principle* [Wexler 1992] is simply this avoidance of overgeneralisation, i.e., if two hypotheses cover the data, we select the most specific one.

The most general hypothesis is represented by $\top$ and the most specific hypothesis is represented by $\perp$. Note that the subset principle (without restrictions such as the rlgg, which only allows one clause per predicate) always give $\perp + E$. However, the generality measure is useful when combined with other criteria. Let us first introduce a measure of generality for logic programs:

Definition 7.104. The Generalisation Degree of a logic program P with respect to a set of ground literals E, denoted $GD(P|E)$, is defined as follows:

$$GD(P|E) = \text{card}\, M^+(P) \,/\, \text{card}(E^+)$$

$M^+(P)$ being the model of P. If $GD(P|E) < 1$, we have that the program does not cover all the samples, so some exception should be added. If $GD(P|E) > 1$, which is the general case, the idea is adjusting to $GD(P|E)$ = card(*Total Positive Possible Examples*) / card(*Presented Positive Examples*) but, obviously, the total of positive possible samples is not known a priori.

The previous measure, however, is not applicable in many situations, since card $M^+(P)$ can be infinite. For this reason, there is better measure for the generality of a hypothesis:

Definition 7.105. [Muggleton 1995] Let H be a wff (well-formed formula) and D be a probability distribution over a (possibly infinite) set of wffs X. The generality g of H is defined as:

$$g(H) = \sum_{x \in X,\, H \models x} D(x)$$

$g(H)$ is just the probability that an instance drawn randomly from D will be entailed by H. According to the Central Limit Theorem, this measure can be approximated given a sufficiently large random sample S from D, the proportion of S entailed by H being an arbitrarily good estimate of g(H):

7.2.2 The MDL principle based on Model Complexity

We saw in chapter 4 a way to compute the model complexity of an evidence with respect to a program $MC(E|T)$. This is based on the length of coding the rules of the program in the following way [Conklin and Witten 1994].

$l(P) = 1 + \log(v + 1) + 2$ bits per literal + the size of each literal computing the size of each literal as *size*(l) = $a \log (v + c)$, a being the arity of the predicate of literal, c the number of constants in the program and v being the number of variables of the rule with more variables.

The MDL principle for logic programs can be defined in terms of Model Complexity (MC). If the theory T covers exactly the evidence E, i.e, $GD(T|E) = 1$, we have that $MC(T|E) = L(T)$, and we talk indistinctly about descriptions and theories.

But in the case that T does not cover all the examples, i.e, $GD(T|E) < 1$, we have two options, we can *augment* it with the exceptions or quote them separately. In both cases the measure should be the same, so we remake the definition accordingly: $MC(T|E) = L(T) + L(E - M^+(T))$.

Finally, in the case that $GD(T|E) > 1$, more elements than the positive evidence can be deduced, so we need to remove the extra consequences which are not from the evidence, $MC(T|E) = L(T) + L(M^+(T) - E)$. This makes the final definition of $MC(T|E)$:

$$MC(T|E) = L(T) + L(M^+(T) - E) + L(E - M^+(T))$$

In [Conklin and Witten 1994] the term $L(M^+(T) - E)$ is substituted by $L(E)$ when $L(E) < L(M^+(T) - E)$, but, in my opinion, this is not fair, because in that case, T would be useless to describe the data. Moreover, this would require an extra bit to distinguish between both situations.

The most efficient way of measuring this $L(M^+(T) - E)$ is given by the following formula:

$$L(E \mid T) = \log \binom{l(T)}{l(E)}$$

And finally the MDL principle (a first version) is defined as (supposing that the theory is always augmented to cover all the evidence):

$$MDL_1(T \mid E) = L(T) + \log \binom{l(T)}{l(E)}$$

7.2.3 The MDL principle based on Proof Complexity

The proof complexity measure $PC(E|T)$ [Muggleton et al. 1988] [Muggleton et al. 1992] was introduced in chapter 4 (Definition 4.29) as $L_{PC}(E|T)$. We just present the variant of the MDL principle based on this variant:

$$\mathrm{MDL}_2(T|E) = L(T) + PC(E|T)$$

7.2.4 Information Gain revisited: G (T|*E*)

In chapter 4 we estimated $G(E|T)$ for logical theories. The estimation of $G(T|E)$, the explicitness of a theory with respect to the evidence is more complex, because it depends on the inductive method which would be used.

However, there are cases that are always the same for every inductive algorithm. The theory $T = E$, i.e. $T = \perp + E$, is generally the easiest one to find, the most explicit one. In the case of logic programs, a rule $p(X,Y)$ when there is a fact in the evidence $p(a,b)$ is also easy to find and short to describe from the evidence (take example $p(a,b)$ and generalise it, denoted by *cgen*(p(a,b))). On the contrary, invented predicates are difficult to find. According to these ideas, a rough approximation to the explicitness of a theory can be defined as follows:

Let us consider the background knowledge B jointly with the evidence E and the theory T. Then we have to consider p= number of predicates of $<E,B,T>$, px = number of invented predicates (new predicates of T), n = number of facts of E, c = number of constants and different variables of $<E,B,T>$, cx = number of constants which appear as new in T with respect to $<E,B>$, and v = number of different variables of $<E,B,T>$.

Definition 7.106. Approximation of Gain for Logical Theories:

$G(T \mid <E,B>) =$ $\log (v + 1) + \log px + \log cx +$

for each literal s of T: $+ 1 +$

if $s \in E$ then add:	$1 + \log n$
if $t \in E$ and $s = cgen(t)$	$1 + 1 + \log p$
else	$1 + 1 + \log p + a \cdot \log (c+v)$

A final bit is reckoned to indicate the case that the theory covers all the evidence (i.e. $GD(T|E) > 1$), and in this case the extensional facts of the theory exactly equal to the evidence are not taken into account (cost = 0), and, therefore, need not be transmitted.

7.2.5 Reinforcement Revisited

For positive evidence, the mean course $m\chi(E|T)$ is a very convenient way of evaluate the theory. However, we must compensate the cases where the theory is much too general.

On the other hand, we must also consider the negative evidence. We will select $m\chi^0$ which was seen in chapter 5, because it weighed independently the values of ρ^+ and ρ^-.

Another measure that will be used is derived from the generality degree seen before and this measure of mean course. Namely,

$$m\chi' = m\chi \cdot (1 - 0.5f + f \cdot 2^{-\mathrm{GD}})$$

where $f = (n^+ - n^-) / (n^+ + n^-)$, with n^+ being the number of positive examples and n^- being the number of negative examples.

It is important to note that this formula can give a value of $m\chi'$ slightly greater than 1. If $f > 0$ (more positive examples than negative ones) then generality is penalised because it is easier. Contrariwise, if $f < 0$ (more negative examples than positive ones) then generality is favoured because it is more difficult. Logically, if GD = 1 then $m\chi' = m\chi$.

Now, given these new measurements and those seen in the previous chapters we will compare them in different situations.

7.2.6 Example

First, we are going to use one of the most classical examples in ILP, also revisited by [Conklin and Witten 1994], which is originally discussed by [Quinlan 1990] and describes the connection or "reachability" relation in a network. The signature comprises two binary predicates *reach* and *linked*, along with $c = 9$ constants {0,…,8}. The background theory B is composed of 10 extensional facts:

B = { linked(0,1), linked(0,3), linked(1,2), linked(3,2), linked(3,4), linked(4,5), linked(4,6), linked(6,8), linked(7,6), linked(7,8) }

This background is represented in the following figure:

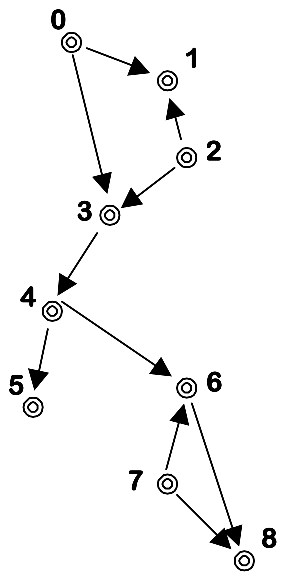


Figure 7.1. *Graph of the Reachability Example*

This background knowledge has the following length:

$l(B) = \log(0+1) + 1 + 10 \cdot (\log 1 + 1) + 20 \log (9) = 11 + 20 \log 9 = 74{,}4$

7.2.6.1 Inducing from Complete Evidence: All the Positive Samples

This is the easiest case, because the closed world assumption is right here: we have all the positive samples and all the rest are negative.

In this case, the evidence E is a *complete* specification of the predicate *reach* composed of 19 facts out of the possible 72 combinations:

E = { reach(0,1). reach(0,2). reach(0,3). reach(0,4). reach(0,5). reach(0,6). reach(0,8). reach(1,2). reach(3,2). reach(3,4). reach(3,5). reach(3,6). reach(3,8). reach(4,5). reach(4,6). reach(4,8). reach(6,8). reach(7,6). reach(7,8) }

with $l(E) = \log(0+1) + 1 + 19 \cdot (\log 1 + 2) + 38 \log (9) = 39 + 38 \log 9 = 159.5$

The theories shown in table 1 might be induced [Conklin and Witten 1994]:

Theory	Program	Comment
T_1	reach(X,Y)	$T_1 = \top$
T_2	reach(0,1). reach(0,2). reach(0,3). reach(0,4). reach(0,5). reach(0,6). reach(0,8). reach(1,2). reach(3,2). reach(3,4). reach(3,5). reach(3,6). reach(3,8). reach(4,5). reach(4,6). reach(4,8). reach(6,8). reach(7,6). reach(7,8)	$T_2 = \perp + E$
T'_2	reach(0,X). : $\rho = 1 - 2^{-7} \approx 0.992$ reach(3,X). : $\rho = 1 - 2^{-5} \approx 0.969$ reach(X,8). : $\rho = 1 - 2^{-5} \approx 0.969$ reach(1,2). reach(4,5). reach(4,6). reach(7,6). : $\rho = 0.5$	T'_2 = simple generalisation when there are more than 5 facts.
T_3	reach(X,Y) :- linked(X,Y). reach(0,2). reach(0,4). reach(0,5). reach(0,6). reach(0,8). reach(3,5). reach(3,6). reach(3,8). reach(4,8).	
T_4	reach(X,Y) :- linked(X,Y). reach(X,Y) :- linked(X,Z). (T'_4)	The second clause subsumes the first one.
T_5	reach(X,Y) :- linked(X,Y). reach(X,Y) :- linked(X,Z), linked(Z,Y). reach(0,5). reach(0,6). reach(0,8). reach(3,8).	
T_6	reach(X,Y) :- linked(X,Y). reach(X,Y) :- linked(X,Z), reach (Z,Y).	The intended one

Table 7.1. *Theories for the "reachability" relation.*

With the following lengths:

$L(T1) = 1 + 1.58 + 1(0+2) + (2 \cdot \log 11) = 11.5$ (2 variables)

$L(T2) = 1 + 0 + 19(0+2) + 19 \cdot (2 \cdot (\log 9)) = 159.5$

$L(T'2) = 1 + 1 + 7(0+2) + 7 \cdot (2 \cdot (\log 9)) = 60.3$ (1 variables)

$L(T3) = 1 + 1.58 + 11 \cdot (\log 2 + 2) + 11 \cdot (2 \cdot \log 11) = 111.7$ (2 variables, 2 predicates)

L(T4) = 1 + 2 + 4(log 2 + 2) + 4 · 2 · log 12 = 43.7 (3 variables)

L(T'4) = 1 + 2 + 2(log 2 + 2) + 2 · 2 · log 12 = 23.3 (3 variables)

L(T5) = 1 + 2 + 9(log 2 +2) + 9 · (2 · log 12) = 94.5 (3 variables, 2 predicates)

L(T6) = 1 + 2 + 5(log 2 +2) + 5 · (2 · log 12) = 53.8 (3 variables, 2 predicates)

and the corresponding *L(T | <E,P>)*:

L(T1 | <E,P>) = 1 + log 3 + 2 + log 2 + 1 = 6.58

L(T1) = 11.5 G(T1 | <E,P>) = 0.57

L(T2 | <E,P>) = 1 + log 3 = 2.58

L(T2) = 159.5 G(T2 | <E,P>) = 0.02

L(T'2 | <E,P>) = 1 + log 3 + 3 · (2 + log 2 + 2 · log (9 +2) + 1) = 35.3

L(T'2) = 60.3 G(T2 | <E,P>) = 0.59

L(T3 | <E,P>) = 1 + log 3 + 2 · (2 + log 2 + 1) = 10.58

L(T3) = 111.7 G(T3 | <E,P>) = 0.09

L(T4 | <E,P>) = 1 + log 3 + 3 · (2 + log 2 + 1) + 1 · (2 + log 2 + 2 · log (9 +2) + 1) = 25.5

L(T4) = 43.7 G(T4 | <E,P>) = 0.58

L(T'4 | <E,P>) = 1 + log 3 + 1 · (2 + log 2 + 1) + 1 · (2 + log 2 + 2 · log (9 +2) + 1) = 17.5

L(T'4) = 23.3 G(T'4 | <E,P>) = 0.75

L(T5 | <E,P>) = 1 + log 3 + 3 · (2 + log 2 + 1) + 2 · (2 + log 2 + 2 · log (9 +2) + 1) = 36.42

L(T5) = 94.5 G(T5 | <E,P>) = 0.39

L(T6 | <E,P>) = 1 + log 3 + 3 · (2 + log 2 + 1) + 2 · (2 + log 2 + 2 · log (9 +2) + 1) = 36.42

L(T6) = 53.8 G(T6 | <E,P>) = 0.68

The reinforcements are computed as follows:

reinforcements	Mean Course
reach(X,Y) : $\rho = 1 - 2^{-19} \approx 1$	$m\chi \approx 1$
reach(0,1). reach(0,2). reach(0,3). reach(0,4). reach(0,5). reach(0,6). reach(0,8). reach(1,2). reach(3,2). reach(3,4). reach(3,5). reach(3,6). reach(3,8). reach(4,5). reach(4,6). reach(4,8). reach(6,8). reach(7,6). reach(7,8) : $\rho = 0.5$	$m\chi = 0.5$
reach(0,X). : $\rho = 1 - 2^{-7} \approx 0.992$ reach(3,X). : $\rho = 1 - 2^{-5} \approx 0.969$ reach(X,8). : $\rho = 1 - 2^{-5} \approx 0.969$ reach(1,2). reach(4,5). reach(4,6). reach(7,6). : $\rho = 0.5$	$m\chi = 0.88$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-10} \approx 0.999$ reach(0,2). reach(0,4). reach(0,5). reach(0,6). reach(0,8). reach(3,5). reach(3,6). reach(3,8). reach(4,8). : $\rho = 0.5$	$m\chi \approx 0.76$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-10} \approx 0.999$ reach(X,Y) :- linked(X,Z). : $\rho = 1 - 2^{-19} \approx 1$	$m\chi \approx 1$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-10} \approx 0.999$ reach(X,Y) :- linked(X,Z), linked(Z,Y). : $\rho = 1 - 2^{-5} \approx 0.969$ reach(0,5). reach(0,6). reach(0,8). reach(3,8). : $\rho = 0.5$	$m\chi \approx 0.886$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-19} \approx 1$ reach(X,Y) :- linked(X,Z), reach (Z,Y). : $\rho = 1 - 2^{-9} \approx 0.998$	$m\chi \approx (1 \cdot 10 + 0.998 \cdot 1 \cdot 5 + 0.998 \cdot 0.998 \cdot 1 \cdot 3 + 0.998 \cdot 0.998 \cdot 0.998 \cdot 1 \cdot 1\ ) / 19 = 0.9985$

From here we can give the values shown in table 2:

T	L(T)[31]	GD	Consilient (no excepts.)	Gain	Mean Reinf. ($m\chi$)	Spec. ($m'\chi$)	L(E|T)	**MDL$_1$**	PC(E|T)	**MDL$_2$**
T_1	11.5	3.8	Yes	0.57	≈ 1	0.57	56.7	68.2	120.5	**132.0**
T_2	159.5	1	No	0.02	= 0.5	0.5	0	159.5	80.7	240.2
T'_2	60.3	1.52	No	0.59	0.88	0.75	24.3	84.6	100.9	161.2
T_3	111.7	1	No	0.09	0.76	0.76	0	111.7	96.3	208.0
T_4	43.7	2,53	No	0.58	≈ 1	0.67	43.4	87.1	110.6	154.3
T'_4	23.3	2,53	Yes	0.75	≈ 1	0.67	43.4	66.7	123.3	133.9
T_5	94.5	1	No	0.39	0.886	0.89	0	94.5	101.9	196.5
T_6	53.8	1	Yes	0.68	0.999	**0.999**	0	**53.8**	106.1	160.0

Table 7.2. *Values for the different criteria studied in this section.*

[31] In [Conklin & Witten 1994] the complexities are reckoned in a different way, because the predicates from the background theory *B* not used in *T* are taken into account. The results are 12.5, 178.5, 111.7, 43.7, 94.5, 53.8. The differences are not important (*l* bits, being *l* the number of literals), however, but we have preferred to apply the measure strictly as it is defined. Also, if the background knowledge is great, [Conklin & Witten 1994]'s measurement would differ considerably from ours.

According to the results of Table 2, [Conklin and Witten 1994] concludes that the MDL_1 is the best way to select the right hypothesis (in this case MDL_2 does not select T_6). Without considering the new evaluation criteria, we think that this conclusion is somehow overstated. Note that, from the three theories which are closer to T_6 (which are T'_4, T_1 and T'_2), T'_2 and T'_4 are not considered by Conklin and Witten (the difference is reduced to only 13 bits.) and T_1 is not computed properly by them (in their paper log(81 over 19) = 60.4 is computed instead of log(72 over 19) = 56.7).

According to specialised reinforcement ($m'\chi$), however, the optimality of T_6 is manifest (0.999), after taking into account generality and pure mean course ($m\chi$).

Moreover, in this case, it is just sufficient to combine generality (in this finite case we have used $GD(P|E)$) and exception-free (intensional or consilient) to also select T_6 as the most specific intensional hypothesis.

Gain also provides useful information about the theories. According to the Oblivion Criterion seen in chapter 4 as the product of gain and a plausibility criterion, if we are not sure of which hypothesis is the best, and we want to store some of the theories, but we only have space for 3, we would choose (by using $m'\chi$ as plausibility criterion):

T	Gain	$m'\chi$	OC
T_1	0.57	0.57	0.32
T_2	0.02	0.5	0.01
T'_2	0.59	0.75	**0.44**
T_3	0.09	0.76	0.07
T_4	0.58	0.67	0.38
T'_4	0.75	0.67	**0.50**
T_5	0.39	0.89	0.35
T_6	0.68	0.999	**0.68**

As we said in chapter 4 this would allow to preserve the effort which has been invested in the search of plausible and difficult hypotheses while still controlling the size of memory.

Finally, let us consider the case of T_4' and T_4

T_4' and T_4	r_a:	reach(X,Y) :- linked(X,Z). : $\rho = 1 - 2^{-19} \approx 1$
T_4	r_b:	reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-10} \approx 1$

Note that r_b is a specialisation of r_a, i.e., it is a deductive consequence of r_a. The measure of reinforcement is logically not affected by a deductive consequence that is added to the theory, which in this case does not improve the reinforcement of the whole theory. However, both measurements of the MDL principles are affected, when T_4' and T_4 are, semantically, exactly the same theory. This problem of the MDL principle to work consistently with deduction has been pointed out several times in work.

7.2.6.2 Inducing from Partial Evidence: Partial Positive Sample

In this case we are going to study the most usual case when learning from positive evidence: only a part of the evidence is shown. The example is modified to the same background knowledge B but a different evidence E, which has 12 examples, instead of 19:

E = { reach(0,3). reach(0,4). reach(0,5). reach(0,8). reach(3,2). reach(3,4). reach(3,5). reach(3,8). reach(4,6). reach(4,8). reach(6,8). reach(7,8) }

The theories we are going to consider and their reinforcements are shown in Table 3:

reinforcements	Mean Course
reach(X,Y) : $\rho = 1 - 2^{-12} \approx 1$	$m\chi \approx 1$
reach(0,3). reach(0,4). reach(0,5). reach(0,8). reach(3,2). reach(3,4). reach(3,5). reach(3,8). reach(4,6). reach(4,8). reach(6,8). reach(7,8): $\rho = 0.5$	$m\chi = 0.5$
reach(0,X). : $\rho = 1 - 2^{-4} \approx 0.9375$ reach(3,X). : $\rho = 1 - 2^{-4} \approx 0.9375$ reach(X,8). : $\rho = 1 - 2^{-5} \approx 0.969$ reach(4,6). : $\rho = 0.5$	$m\chi = 0.91$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-6} \approx 0.984$ reach(0,4). reach(0,5). reach(0,8). reach(3,5). reach(3,8). reach(4,8).: ρ=0.5	$m\chi \approx 0.742$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-6} \approx 0.984$ reach(X,Y) :- linked(X,Z). : $\rho = 1 - 2^{-12} \approx 1$	$m\chi \approx 1$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-6} \approx 0.984$ reach(X,Y) :- linked(X,Z), linked(Z,Y). : $\rho = 1 - 2^{-3} \approx 0.875$ reach(0,5). reach(0,8). reach(3,8). : $\rho = 0.5$	$m\chi \approx 0.836$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-12} \approx 0.9998$ reach(X,Y) :- linked(X,Z), reach (Z,Y). : $\rho = 1 - 2^{-6} \approx 0.984$	$m\chi \approx (0.9998 \cdot 6 + 0.9998 \cdot 0.984 \cdot 3 + 0.9998 \cdot 0.984 \cdot 0.984 \cdot 2 + 0.9998 \cdot 0.984 \cdot 0.984 \cdot 0.984 \cdot 1) / 12 = 0.987$

Table 7.3. *New theories for the "reachability" relation and their corresponding* $m\chi$.

The lengths and gains are computed again:

L(T1 | <E,P>) = 1 + log 3 + 2 + log 2 + 1 = 6.58

$L(T1) = 11.5$ $G(T1 \mid <E,P>) = 0.57$

$L(T2 \mid <E,P>) = 1 + \log 3 = 2.58$

$L(T2) = 101.1$ $G(T2 \mid <E,P>) = 0.03$

$L(T'2 \mid <E,P>) = 1 + \log 3 + 3 \cdot (2 + \log 2 + 2 \cdot \log (9 +2) + 1) = 35.3$

$L(T'2) = 35.4$ $G(T2 \mid <E,P>) = 0.998$

$L(T3 \mid <E,P>) = 1 + \log 3 + 2 \cdot (2 + \log 2 + 1) = 10.58$

$L(T3) = 81.9$ $G(T3 \mid <E,P>) = 0.13$

$L(T4 \mid <E,P>) = 1 + \log 3 + 3 \cdot (2 + \log 2 + 1) + 1 \cdot (2 + \log 2 + 2 \cdot \log (9 +2) + 1) = 25.5$

$L(T4) = 43.7$ $G(T4 \mid <E,P>) = 0.58$

$L(T'4 \mid <E,P>) = 1 + \log 3 + 1 \cdot (2 + \log 2 + 1) + 1 \cdot (2 + \log 2 + 2 \cdot \log (9 +2) + 1) = 17.5$

$L(T'4) = 23.3$ $G(T'4 \mid <E,P>) = 0.75$

$L(T5 \mid <E,P>) = 1 + \log 3 + 3 \cdot (2 + \log 2 + 1) + 2 \cdot (2 + \log 2 + 2 \cdot \log (9 +2) + 1) = 36.42$

$L(T5) = 84.5$ $G(T5 \mid <E,P>) = 0.43$

$L(T6 \mid <E,P>) = 1 + \log 3 + 3 \cdot (2 + \log 2 + 1) + 2 \cdot (2 + \log 2 + 2 \cdot \log (9 +2) + 1) = 36.42$

$L(T6) = 53.8$ $G(T6 \mid <E,P>) = 0.68$

And finally, we construct again the table with all the measurements:

T	L(T)	GD	Consilient (no excepts.)	Gain	Mean Reinf. ($m\chi$)	Spec. ($m'\chi$)	L(E|T)	**MDL$_1$**	PC(E|T)	**MDL$_2$**
T_1	**11.5**	6	Yes	0.57	≈ 1	0.52	43.8	**55.3**	76.1	**87.6**
T_2	101.1	1	No	0.02	= 0.5	0.5	0	101.1	43.0	144.1
T'_2	35.4	2.17	No	≈ 1	0.91	0.66	23.2	58.6	58.9	94.3
T_3	81.9	1.33	No	0.13	0.74	0.66	10.8	92.7	94.1	176.0
T_4	43.7	4	No	0.58	≈ 1	0.56	36.0	79.7	70.9	114.6
T'_4	23.3	4	Yes	0.75	≈ 1	0.56	36.0	59.3	77.9	101.2
T_5	84.5	1.25	No	0.43	0.836	0.77	8.83	93.3	70.3	154.8
T_6	53.8	1.58	Yes	0.68	0.987	**0.82**	15.6	69.4	81.9	135.7

Table 7.4. *Values for the different criteria studied in this section.*

We can observe that when the evidence is reduced, both variants of the MDL principle leave behind the theory T_6 and promote other theories. On the contrary, reinforcement still selects it as the best theory.

7.2.6.3 Inducing from Partial Evidence: Positive and Negative Evidence

Finally, let us introduce negative evidence. Since learning from positive and negative evidence is much easier than from positive evidence only, it is expected to obtain better results than the previous cases.

We have the same positive evidence:

E^+ = { reach(0,3). reach(0,4). reach(0,5). reach(0,8). reach(3,2). reach(3,4). reach(3,5). reach(3,8). reach(4,6). reach(4,8). reach(6,8). reach(7,8) }

and a negative evidence

E^- = { reach(8,3). reach(5,4). reach(0,7). }

We are going to consider the same theories and, obviously, the lengths and gain will be the same.

Positive reinforcement is maintained, and there is no negative reinforcement for T_2, T_3, T_5 and T_6. However, for the other theories, reinforcement (the $m\chi^0$ version) must be computed again:

reinforcements	Mean Course
reach(X,Y) : $\rho^+ = 1 - 2^{-12} \approx 1$, $\rho^- = 1 - 2^{-3} = 0.875$	$m\chi \approx 1$, $m\chi^0 = (12 \cdot 1 - 3 \cdot 0.875) / 12 = 0.78$
reach(0,3). reach(0,4). reach(0,5). reach(0,8). reach(3,2). reach(3,4). reach(3,5). reach(3,8). reach(4,6). reach(4,8). reach(6,8). reach(7,8) : $\rho = 0.5$	$m\chi = 0.5 = m\chi^0$
reach(0,X). : $\rho = 1 - 2^{-4} \approx 0.9375$, $\rho^- = 1 - 2^{-1} = 0.5$ reach(3,X). : $\rho = 1 - 2^{-4} \approx 0.9375$ reach(X,8). : $\rho = 1 - 2^{-5} \approx 0.969$ reach(4,6). : $\rho = 0.5$	$m\chi = 0.91$, $m\chi^0 = (5 \cdot 0.969 + 6 \cdot 0.9375 + 1 \cdot 0.5 - 1 \cdot 0.5) / 12 = 0.87$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-6} \approx 0.984$ reach(0,4). reach(0,5). reach(0,8). reach(3,5). reach(3,8). reach(4,8). : $\rho = 0.5$	$m\chi \approx 0.742 = m\chi^0$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-6} \approx 0.984$ reach(X,Y) :- linked(X,Z). : $\rho^+ = 1 - 2^{-12} \approx 1$, $\rho^- = 1 - 2^{-1} = 0.5$	$m\chi \approx 1$, $m\chi^0 = (12 \cdot 0.984 - 1 \cdot 0.5) / 12 = 0.94$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-6} \approx 0.984$ reach(X,Y) :- linked(X,Z), linked(Z,Y). : $\rho = 1 - 2^{-3} \approx 0.875$ reach(0,5). reach(0,8). reach(3,8). : $\rho = 0.5$	$m\chi \approx 0.836$
reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-12} \approx 0.9998$ reach(X,Y) :- linked(X,Z), reach (Z,Y). : $\rho = 1 - 2^{-6} \approx 0.984$	$m\chi \approx 0.987 = m\chi^0$

In this case $m'\chi$ is computed in the following way:

$f = (12 - 3) / 12 = 0.75$

$m'\chi = m\chi^0 \cdot (1 - 0.5 \cdot 0.75 + 0.75 \cdot 2^{-GD}) = m\chi^0 \cdot (0.625 + 0.75 \cdot 2^{-GD})$

And from here, we have the new results in table 7.5:

T	L(T)	GD	Consilient (no excepts.)	Gain	Mean Reinf. ($m\chi^0$)	Spec. ($m'\chi^0$)	L(E|T)	**MDL₁**	PC(E|T)	**MDL₂**
T_1	11.5	6	Yes	0.57	0.78	0.50	43.8	**55.3**	76.1	**87.6**
T_2	101.1	1	No	0.02	= 0.5	0.5	0	101.1	43.0	144.1
T'_2	35.4	2.17	No	≈ 1	0.87	0.79	23.2	58.6	58.9	94.3
T_3	81.9	1.33	No	0.13	0.74	0.68	10.8	92.7	94.1	176.0
T_4	43.7	4	No	0.58	0.94	0.63	36.0	79.7	70.9	114.6
T'_4	23.3	4	Yes	0.75	0.94	0.63	36.0	59.3	77.9	101.2
T_5	84.5	1.25	No	0.43	0.836	0.79	8.83	93.3	70.3	154.8
T_6	**53.8**	1.58	Yes	0.68	0.987	**0.86**	15.6	69.4	81.9	135.7

Table 7.5. *Values for the different criteria for positive and negative evidence.*

Note that both MDL$_1$ and MDL$_2$ do not change because both ignore errors. This is because $L(E|T)$ must say which facts of $M(T)$ are really in E^+. Since the aim of MDL principle is descriptional, it is not relevant whether part of the information is required for telling that an evidence has not still appeared (*e* in $M(T)$ but *e* not in E^+ and not in E^-) or whether the information is required for the cases where (*e* in $M(T)$ and *e* in E^-).

A partisan of the MDL principle could say that T$_6$ is the shortest one without errors. This is true in this case but this criterion would turn the MDL principle useless mainly for the cases which has been more successful, learning from noisy data. A better idea is to rectify the MDL principle by the proportion of errors:

This gives a MDL$_1$ as:

$$MDL^{+,-}{}_1 = MDL_1 \cdot 2^{\alpha \cdot e(T)}$$

with *e*(*T*) being the error ratio (negative examples covered / positive examples covered). The positive and negative examples covered by each of them are:

T	$e(T)$	**MDL_1**	**$MDL^{+,-}_1$** ($\alpha = 1$)	**$MDL^{+,-}_1$** ($\alpha = 5$)
T_1	0.25	**55.3**	65.8	131.5
T_2	0	101.1	101.1	101.1
T'_2	0.083	58.6	62.1	104.4
T_3	0	92.7	92.7	92.7
T_4	0.083	79.7	84.4	106.4
T'_4	0.083	59.3	**62,8**	79.2
T_5	0	93.3	93.3	93.3
T_6	0	69.4	69.4	**69.4**

Naturally, as greater the value of α, the less robust to errors that the measure would be.

Finally, let us consider the contrary case, 3 positive examples and 12 negative ones (f = -0.75). In this case, the extensional theory T_2 with mean reinforcement = 0.5 and GD = 1 and the same theory T_6 which still gives:

reach(X,Y) :- linked(X,Y). : $\rho = 1 - 2^{-3} = 0.875$

reach(X,Y) :- linked(X,Z), reach (Z,Y). : $\rho = 1 - 2^{-2} = 0.75$

a mean reinforcement $m\chi = (0.875 \cdot 2 + 0.875 \cdot 0.75 \cdot 1) / 3 = 0.8$.

With a $GD = 19 / 3 = 6.33$ we have:

$$m'\chi(E|T_2) = m\chi(E|T_2) \cdot (1 + 0.5 \cdot 0.75 - 0.75 \cdot 2^{-GD}) =$$
$$= 0.5 \cdot (1.375 - 0.75 \cdot 0.5) = 0.5$$
$$m'\chi(E|T_6) = m\chi(E|T_6) \cdot (1 + 0.5 \cdot 0.75 - 0.75 \cdot 2^{-GD}) =$$
$$= 0.8 \cdot (1.375 - 0.75 \cdot 0.01) = 1.09$$

Reasonably, $m'\chi(T_6)$ is increased by the fact that it is deserving that a general theory 'survives' to a mostly negative evidence.

7.2.6.4 Inducing from Noisy Evidence

The MDL principle, as it has been commented, has been successfully applied to noisy data. Since the goal is to compress, some extensional patches can still be added to the theory whereas the whole compression ratio keeps high. However, the MDL principle is blind to the degree of errors in the evidence. The usual solution in a universal description mechanism is to quote the exceptions separately, be they positive or negative, and, consequently, there are no extrinsic exceptions, and they are penalised by the increment of size of the theory. However, in Horn theories, we cannot remove a consequence to patch a theory, we cannot say $M(T) - f$. This represents a non-uniform way of considering excluded positive evidence (just patch them) and included negative evidence. A solution can be a weighing such as $\mathrm{MDL}^{+,-}{}_1$.

However, the MDL principle is blind in another sense. It gives a single value, and one cannot finally know which percentage of the data is covered extensionally, thus it is difficult to know whether they may be a lot of errors in the theory or not.

On the contrary, reinforcement or intensionality are useful to distinguish the error ratio of the theory, and compare it with the expected error ratio of the evidence. This is precisely the most practical result of *explanatory complexity*. Given an evidence x, if we have an expectancy of noise of about 3%, we must only search for descriptions whose extensional part is $\Delta(p_x) \approx l(x) \cdot 0.03$. It is important to realise that the MDL principle gives an *uncontrollable* and *unpredictable* exception ratio, which only depends on the data and usually will underfit (for explanation) or overfit.

7.2.6.5 Conclusions

In the framework of incremental learning, an intensional criterion is less conservative than the MDL principle, and consequently it usually minimises the whole number of 'mind changes' (although these changes are usually more radical) when the data is perfect. Loosely, we should say that the MDL principle complies with Kuhn's philosophy of changing paradigms; when the number of exceptions is too great, the paradigm must be changed. In contrast, an intensional criterion anticipates this necessity since any exception forces the revision of the model. It is more eager in the sense of chapter 4.

Reinforcement is somehow between the two extremes. This compromise has been shown to be a much more reliable criterion that the MDL principle. Although the comparison made here should be applied to much more examples, the results shown here are expected to hold in general, due to the theoretical justifications given in chapter 6.

7.3 Generation of Inductive Hypotheses

The great advantage of reinforcement (or a detailed exception detection) over the MDL principle is not only that is a measure which is more adjustable to the expectations of the source. The great advantage of a detailed measure is that it allows to guide the inductive search, because it detects which parts of the theory are weak and must be revised in order to obtain better theories.

First of all, however, we will show that it is possible to obtain informative hypothesis with efficient methods. This may seem paradoxical at first sight because the information gain of a hypothesis with respect to the data depends on the difficulty of a concrete inductive algorithm to find them. We will show that this is possible in two different ways: non-data driven approaches and randomised approaches. Of the first kind, enumeration approaches can find informative hypotheses because they are not guided by the data.

However, to avoid the computational cost, the common approach is to construct the theory from the examples (either top-down or bottom-up). In this case, creativity is more difficult to obtain, but a theory that covers the evidence is found in less time. In order to allow this informativeness, randomised techniques such as genetic programming can be used. We will use the measurement of reinforcement and the concepts of intensionality and consilience to outline algorithms for finding comprehensive hypotheses.

7.3.1 Information Gain and the Enumeration Approach

There is a well-known powerful and complete algorithm for the induction problem called the enumeration algorithm. The algorithm is the simplest one: Select an ordering P_1, P_2, ... of all possible programs in a given representation mechanism. Once settled this ordering (the most critical question), the algorithm is *just*:

```
DOVETAIL ALGORITHM
Input: the positive and negative evidence $E^+$ and $E^-$. A background theory $B$. These must ensure $B \wedge E^- \not\models \Box$ (prior satisfiability) and also, it is supposed that $B \not\models E^+$ (prior necessity)
Output: a program $P$, such that $B \wedge P \models E^+$ (posterior sufficiency) and $B \wedge P \wedge E^- \not\models \Box$ (posterior satisfiability).
Let $n = 1$.
while $n$ + log exectime(p) < $k$
  Select program $P_n$ and check $E^+$ and $E^-$ on $B \wedge P_n$ up to a time-bound $2^{(k - n)}$.
  if $B \wedge P_n \models E^+$ (posterior sufficiency) and $B \wedge P_n \wedge E^- \not\models \Box$ (posterior satisfiability) and Eval($P_n$) = true then stop.
  else let $n := n + 1$.
endwhile
```

If the evaluation criteria Eval(·) gives always true, it is clear that this algorithm founds the shortest program whose Levin's $Kt(p) = \min (l(p) + \log \text{cost}(p)) < k$ if the enumeration n is selected according to the length ($l(p)$).

Imagine an intended program has length 17. Just in this simple case, a dovetail style algorithm would require the evaluation of 2^{17} (approximate 100,000) programs, which will require less than (log t) · 2^{17} time.

In the view of this example, it must seem fool to even consider this option, but it is important to recall that we are interested in short solutions (theories) even when large amounts of data are given. Suppose we are given $n_p = 20$ positive examples and $n_n = 30$ negative ones, being the solution of 20 bits. An $O(n_p^3 \cdot n_n^2)$ algorithm would be much worse (24,300,000 against 2^{20} = 1,048,576). In this line, we can include many works that claim polynomial-time learnability (with respect to the size of the evidence) for certain restrictions of regular logic programs. For instance, [Krishna-Rao 1997] presents an algorithm that makes polynomial-time learnable some concepts by using *regular* background knowledge. In the case of deriving multiplication from addition as background knowledge and positive evidence, the polynomial is greater than $16n^8$. Obviously, no hints of being implemented or expected execution times are presented. Finally, and more curiously, these polynomial limits are obtained by using length bounds to enumeration algorithms.

The 'folly' idea of enumeration is supported by some additional considerations: a dovetail style algorithm is complete if an effective and short program exists, it can be pruned syntactically and semantically (regarding the data), schemata or priors can be used to change the ordering of programs and, finally, there is no other algorithm more given to parallelisation.

Another relevant feature of this algorithm is that it can find informative and creative hypotheses. Note that since $G(T \mid E) = Kt(T \mid E)/Kt(T)$, it is important to keep $Kt(T)$ small in order to have a chance of making $G(T \mid E)$ close to 1. This was formally shown in chapter 4. But note that a complexity-ordered algorithm does not mean that the selection principle should be the MDL principle, which in this case would be equal to Eval = true, which stops with the first hypothesis that would be found. On the contrary, the algorithm could be run up to a limited space and select a set of optimal hypotheses according to some criterion, e.g. the mean course or an intensional criterion.

Although this complexity driven induction is computationally very hard, some learning systems are beginning to use it profitably, such as some ILP systems such as Progol [Muggleton 1995], other more direct approaches [Giordano 1994], [Minton 1994], [Riddle et al. 1994], [Schmidhuber et al. 1997] or evolutionary variants [Olson 1994] [Wyard 1994] [Conte et al. 1997].

7.3.2 Randomised example-driven Induction, Reinforcement and Gain

As we have said in the introduction of this section, the common approach to induction is to begin from the evidence, and refine the hypothesis according to some evaluation criterion. In ILP it is possible to construct a search space based on subsumption or some other semantic relationship. The algorithms can then begin with the most general theory and refine it by negative evidence or by some specialisation criterion with respect to the positive evidence, or they can grow from the most specific hypothesis. Most ILP systems order their search space by semantic relations. Other approaches, however, are beginning to be explored in ILP, such as genetic programming [Olson 1994] [Charif 1994] [Varsek 1995] [Ichise 1998.

As we have discussed before, genetic programming is a randomised technique and therefore the informativeness of the results can be high since the inductive method is not deterministic.

Another advantage of a genetic programming approach is that the evaluation criterion is flexible and can be changed without remaking the whole structure of the algorithm. In [Hernández-Orallo and Ramírez-Quintana 1998a, 1998b, 1999a] , this approach is considered, but applied to functional logic programs instead of logic programs, by using a narrowing operator instead of resolution.

Roughly, the algorithm generates a population of generalisations of the instances given in the evidences. This constitutes a set *EH*, which is composed of equation. These equations are rated according to some specific criteria (accordingly to the positive and negative facts they cover), which is closely related to the measurement of reinforcement seen in chapter 5. A set *PH*, composed of programs, is initially constructed from all the unary sets that can be constructed from the set *EH*.

The selection criterion of the algorithm is a variant of the intensional or consilient criterion seen in chapter 6, and this is responsible for promoting the combination of rules in order to make final programs not separable, i.e., to cover comprehensively the evidence. Moreover, this combination has the advantage that reinforcement does not need to be recomputed from scratch but the separate reinforcements of each part can be profited to compute the new reinforcement.

One of the characteristics of this method is that every combination of rules that generates a new rule by inverse narrowing (an analogous process to inverse resolution for functional logic programs) adds this new rule to *EH*, so this set is enlarged by new rules which can compose the programs. To maintain all the rules which are being generated is not a real problem with the memory resources and power of current computers. However, the set *PH* is composed of combinations of elements of *EH*, which can be extremely large. The problem is well-known in genetic algorithms and genetic programming: many of the individuals must die to leave place for better individuals. However, must only the criterion of the best individual rule this selection?

In practice, there are two better possibilities. The first one is to recall which programs have been essayed. This, however, has the problem that, in many cases, it also takes a great amount of space. The second option is to take into account the oblivion criterion: easy programs that can be easily generated from *PH* by combinations can be forgotten without hesitation, because they will soon appear again.

The maintenance of this population of hypotheses is even more important for the case of incremental learning, because some earlier hypotheses could be revived by new evidences. In other words, an incremental algorithm cannot only keep the best hypothesis, because any future change would make that all the process starts again.

Moreover, in real situations, the past evidence *can* change, and this is something that is not usually contemplated by any inductive algorithm, because in this case, it should start again from scratch.

On the contrary, with the use of a set of hypotheses, in case of a change of the evidence, only the reinforcements are to be computed again but not all the inductive process. Just if none of these hypotheses gives now a good optimality value, the inductive process is re-started.

7.4 Summary and Contributions of This Chapter

This chapter has shown the first practical applications of most of the measures seen in this work: evaluation of (logical) theories and guiding of learning algorithms.

In Section 2 we have reviewed some of the most classical criteria for the evaluation of Logic Programs used in ILP, especially two variants of the MDL principle. We have compared them with reinforcement, intensionality and gain. In terms of plausibility, reinforcement is manifestly better than the MDL principle, either for whole positive evidence, partial positive evidence and partial positive and negative evidence. Intensionality can be computed to know in which degree the data is 'conciliated' by the theory, and in some cases it can be a prerequisite (abduction, explanatory reasoning, etc.). Finally, for the case of evaluation, gain has only some auxiliary use, mainly for ascertaining when a real learning has taken place, i.e., the theory is original with respect to the data.

The section also shows the importance of combining the generality criterion with reinforcement and the relevance of considering the negative evidence in more detail.

Section 3 shows how reinforcement and intensionality can be combined for guiding a machine learning algorithm. First, it is shown that the enumeration algorithm is compatible with a search for gain, because the data is only used to check the hypotheses. Secondly, a data-driven approach is constructed with the help of genetic programming, where the selection criterion (oblivion criterion) is a combination of the optimality of the program (the individual) and the gain (unusual or rich genotype). Moreover, the randomised character of genetic programming allows the generation of informative hypotheses.

The contributions of this chapter are:

- The comparison of the evaluation measures of this thesis and other classical criteria of logical theories with respect to whole or partial positive evidence, and positive and negative evidence.
- The outstanding results of reinforcement are shown for all these cases.
- A variant of reinforcement that is weighed with the generality degree measure is introduced.

- A variant of the MDL which measures the generality of the positive and negative evidence.
- In the presence of noise, reinforcement and consilience measures allow to obtain the percentage of errors in the theory, which can be compared with the expected noise ratio.
- Hints about hypotheses generation and how this generations should be done in order to obtain informative hypotheses.

In our opinion, the evaluation and generation of hypotheses must be highly interlaced, although if the inductive system is wanted to be applicable to different situations, it must allow the change of the evaluation criterion, because there is not a best criterion for all the situations. Only some paradigms, such as genetic programming, allow the change of this criterion without also forcing the change of the algorithm.

8. Measuring Intellectual Abilities

Encara us dic que port una Art general
que novament és dada per do espiritual,
per qui hom pot saber tota re natural,
segons que enteniment ateny lo sensual.
A Dret e a Medicina e a tot saber val
e a Teologia, la qual m'és mais coral,
e a soure qüestions nul·la art tant no val,
e a destruir errors per raó natural;
tenc-la per perduda car quaix a hom no cal.

Ramon Llull, *Lo desconhort*, VIII.

Abstract: *in this chapter, the ability to comprehend is identified with the main factor of intelligence, derived from the notion of comprehension introduced in chapter 6. However, some technical problems arise when this factor is to be measured, especially unquestionability, and to define an absolute scale of difficulty of comprehension. Both problems are solved in this chapter and the result is a comprehension test, or C-test, exclusively defined in terms of universal descriptional machines. Despite the absolute and non-anthropomorphic character of the test it is equally applicable to both humans and machines. Moreover, it correlates with classical psychometric tests, thus establishing the first firm connection between information theoretic notions and traditional IQ tests. From here, a factorisation is outlined, considering other inductive and deductive factors, thus allowing a theoretical study of their inter-dependence, something that has only been possible to do in an experimental way, as the statistical correlations which have been studied by psychometrics.*

Keywords: Measurement of Intelligence, Psychometrics, IQ tests, Turing Test, Unquestionability, Series Prediction, Comprehension Ability, Deductive Ability.

8.1 Introduction

In the previous chapter, the measurements were applied to hypothesis evaluation and generation. However, a measure for inference processes can also be applied to systems. For instance, in the case of deduction, we measured in chapter 5 the complexity and gain of a given conclusion from its premises. If we devise a test to measure the time that a certain system takes to find the solution, we know that it is able to solve problems of a given complexity. In a more intricate way, we can find induction problems whose answer is unquestionable (we will see that this is possible to some extent) and devise a set of questions to evaluate the inductive ability of a given system.

Although the measurement of inductive abilities is more difficult than the measurement of deductive abilities, we will centre mainly on the measurement of inductive abilities. The reason-why is that any process of induction needs deductive abilities to check the hypotheses with respect to the evidence. This explains why the measurement of induction abilities correlates with g, the main factor of intelligence.

This will allow identifying the main factor of intelligence as the ability to comprehend, derived from the notion of comprehension introduced in chapter 6. The result is a comprehension test, or C-test, exclusively defined in terms of universal descriptional machines (e.g. universal Turing machines). Despite the absolute and non-anthropomorphic character of the test it is equally applicable to both humans and machines. Moreover, we will see that it correlates with classical psychometric tests, thus establishing the first firm connection between information theoretic notions and traditional IQ tests.

8.2 Requirements and Technical Problems

AI has striven to imitate human behaviour in many tasks, under the slogan "*Artificial intelligence is that thing that if made by humans would require intelligence*", which has frequently promoted the view that "human intelligence subsumes machine intelligence" [Bradford and Wollowski 1995] instead of the open and more realistic view that "robots will be more intelligent than we humans are" [Moravec 1998]. Finally, the Turing Test (TT) has usually been understood also as an effective test (and not only as a philosophical exercise). This misinterpretation, jointly with his celebrity, has motivated that there has not been the necessary effort for designing new alternative intelligence tests. The TT has even eclipsed such well-reputed proposals as Simon's early works on the relation between IQ tests and AI [Simon and Kotovsky 1963], on some heuristic approaches to solve analogy problems from

IQ tests [Evans 1963] and Chaitin's suggestion "*develop formal definitions of intelligence and measures of its various components*" [Chaitin 1982]. Even Johnson's call "*Needed: A New Test of Intelligence*" [Johnson 1992] has been responded by formalisations of the TT [Bradford and Wollowski 1995] instead of devising new proposals.

As any other discipline, AI requires an effective measure of its major issue, a gradual and detailed measure of intelligence. A scientific measure of intelligence should follow these requirements:

- Non-Boolean: intelligence is not an absolute attribute. From Darwin's "mental continuity" to infant psychology, there is an unquestionable certainty that intelligence is a gradual aptitude. Any discretisation of the TT as a function of the time of the test or the score of the judges shows the inappropriateness of the TT to measure intelligence in a gradual way, i.e., to give a continuous value of intelligence. The reason is obvious: the TT is a test of humanity [Fostel 1993], and the idea of being more or less human makes no sense.
- Factorial: intelligence is multi-dimensional. It is quite unbelievable that there is a concrete ability that would be optimal for every context or world. Nonetheless, it is conceivable that a concrete ability would be almost optimal for most contexts, and intelligence has sometimes been defined as "second-best in everything". This may justify that a wide context as everyday life, which includes many other contexts, can distinguish a special kind of ability: "*human beings rank animals [and people] using a distinctively human concept of intelligence, the primitive concept found in everyday life, and these rankings correlate with g*" [Flynn 1987].
- Non-anthropomorphic: Maybe the major problem of AI is that the only reference of intelligent behaviour is human intelligence. Only recently, AI researchers have paid attention to other 'intelligences': ants, rats, etc. in order to scale up the problem towards human intelligence. However, the reference or the goal is always human intelligence. Nowadays the reason is not only pride and anthropocentrism but also necessity, because "*there is not yet a solid definition of intelligence that doesn't depend on relating it to human intelligence*" [McCarthy 1998].
- Computationally based: According to the Church-Turing thesis, there is no reason to think that intelligent systems cannot be implemented in current computers. The only question is whether they have the necessary speed and memory for it. I share the opinion that "*computers of 30 years ago were fast enough if only we knew how to program them*" [McCarthy 1998], and the AI problem is *just* to discover what makes a program intelligent, or, in other words "*What kind of Information Processing is Intelligence?*" [Chandrasekaran 1990]. Consequently, it is extremely significant to be able to state the problem computationally, in other words, to give the specification of the problem in computational terms, in order to solve the problem with AI means, which are exclusively machines and programs.

- Meaningful: intelligence is not an ethereal ability. It cannot be exclusively defined as "*intelligence is what is measured by intelligence tests*". Intelligence must be expressed from its original meaning, the ability to comprehend, and this ability is what should be measured: "*the only way to know if our machines, or people, are intelligent, is to make them explain how they did what they did... people can attempt to give rational explanations, and ultimately that is how we must measure intelligence*" [Schank 1986].

Psychometrics has developed measurements of intelligence according to the first two requirements. Since Spearman founded the field [Spearman 1904], psychometrics has been more and more characterised by the scientific method: systematic experimentation and statistical rigour. "*Despite the many shortcomings of an IQ score, no other measure has been found to be related to so many other behaviors of theoretical or practical significance*" [Zigler and Seitz 1982]. However, psychometrics has neglected, or failed, to incorporate the three last requirements, which, in fact, are highly related. Psychometrics is anthropomorphic by definition since it is the science of measuring human intelligence, although there have been adaptations and essays with chimpanzees, dolphins and other animals. Psychometrics is an experimental science that has used the Homo Sapiens Sapiens as both target and reference. The frequently demanded theoretical foundation of psychometrics depends on the change of the point of reference, closely connected to the last three requirements.

On the contrary, Computational Learning Theory is non-anthropomorphic and computational. The question "What is to learn?" has been assimilated to the formal notion of identification in the limit [Gold 1967], which can be sketched as follows: given a pattern or concept C to learn, and an evidence e_1, e_2, ..., as a sample from C, a learner L identifies C if there exists a finite number k such that for every example e_n, $n > k$, the learner predicts all of them correctly.

After some discouraging results on the learnability of very simple languages, the complexity of learning has been studied for other paradigms, mainly PAC learning [Valiant 1984] and Query Learning [Angluin 1988].

However, these theoretical results have not been used to develop rigorous measurements of learning ability. Contrarily, some contests and comparisons have been held for practical systems, but as collections of arbitrary examples extracted from the literature, without many justifications of the theoretical complexity of each of them.

The reason is that it is difficult to establish the complexity of an instance, when the results of computational learning theory apply to classes of concepts, and most results are asymptotical. Moreover, the paradigm of identification in the limit is not applicable for finite instances, because they can be identified by themselves. The philosophical problem of any measurement of learning ability is the same as the philosophical problem of series prediction: given any finite set of examples, there are infinite many concepts that are consistent with them and diverge in their predictions.

This is exactly the 'subjectivity objection' of IQ tests: there may be controversy about the *correct* answer.

With these requirements in mind, let us start with Chaitin's proposal "*develop formal definitions of intelligence and measures of its various components*" [Chaitin 1982] by using descriptional complexity and the ideas of learning as compression. At first sight it seems to be easily applicable. However, there are many technical reasons that explain that such an intriguing proposal (and made from such an eminent source) has not been addressed yet[32].

For instance, the following "compression test" can be recklessly defined:

Definition 8.107 Compression Test. Give an arbitrary string x of size n to a subject and ask for the following symbol s according to the shortest projective description. Mark the subject's answer as a *hit* if $\phi(x^*)_{n+1} = s$.

This test, however, has many technical and fundamental riddles:

- $K(x)$ is not computable, so there is no effective way to know which is the 'correct' answer and, consequently, to know whether the subject's answer is a hit.
- There are different equally alternative plausible descriptions: x^* is just the first one in lexicographic order of all the shortest descriptions.
- Despite the invariance theorem that states that x^* depends on ϕ only up to a constant, this constant is relevant if n is small, and there is no reason to prefer one descriptional system over another. The test suffers the subjectivity objection.
- The test intends to measure the ability of compression, but this does not match exactly[33] with the ability of comprehension, i.e., intelligence.

The first problem has a practical solution. $K(\cdot|\cdot)$ does not reflect a cognitive view of information nor a cognitive view of simplicity, because for some strings the shortest description could be extremely time consuming and, consequently, not valid as an explanation because it cannot be related to others ("*if you want to understand a concept, try explaining it to someone else*" [Winston 1982]).

In the previous chapter we defined a fully projectible version of Kolmogorov Complexity based on the Levin variant LT. We have defined explanatory complexity from it, and the shortest explanatory description (SED) has also been defined. Finally, stable (on the right) objects give SED descriptions where comprehension has taken place, i.e., comprehensive descriptions.

[32] At least to the author's knowledge and as Chaitin himself has recognised (Chaitin 1998, personal communication).

[33] "*I just see how Kolmogorov Complexity and Intelligence could be well related but I don't think it would be 'exactly' so*". (Hofstadter 1997, personal communication)

Theoretically, there are two ways to know whether a system's operation is compliant with some requirements: by inspecting its code (or program) or by testing its behaviour. In software engineering it has been finally accepted that the verification of a specification with respect to a program (formal verification) is only feasible for small systems. In general, for complex systems, verification is experimental, made by means of sets of tests. It is an open and extremely hard problem to devise a complete specification of intelligence, mainly because it depends on a consensus on the *functionalities* that an intelligent system must be able to perform. However, and this is precisely what I claim in this chapter, it is possible to distinguish some functionalities that are fundamental for intelligence. A verification of intelligence behaviour should begin with these fundamental traits, and gradually add more diverse test cases in order to make the test set more robust.

Comprehending is the most important trait of intelligence, and we have formalised it in a computational framework. This would be the major difference between psychometrics and the intended measurement of this chapter. The test's exercises are not selected experimentally but theoretically, so, finally, we will know what is to be measured.

However, if we intend to measure comprehensibility there are still two questions to solve. First, we must design unquestionable exercises, in order to avoid the 'subjectivity objection' of IQ tests. Secondly, we require an absolute referent of comprehension difficulty in order to give a non-Boolean score independent to the mean ability of the subjects or society who have made the test before.

The following sections are devoted to ensure that the descriptions would give the same meaning out of a sequence of symbols as we do [Hofstadter 1985], and only other intelligent beings would do (because the sequence is unquestionable) and how to measure their complexity.

8.3 Unquestionability

Psychometrics has striven to show that it is not absurd to measure the 'correct' solution. Its answer is that if the great majority matches with some solution is *because there are not alternative solutions of similar complexity*, and, consequently, it is the most plausible. However, this assertion is made from a very subjective and informal point of view.

As we saw in chapter 6, we introduced the notion of stability to avoid rote or repetitive descriptions. However, there was a need to avoid extra patterns that could be invented to make the prediction differ.

For this reason, we extended the notion of stability and applied it to descriptions. Let us recall Definition 6.102:

Definition 8.108 Plausibility on the Right. A fully projectible description p for a string x is (c,m)*-plausible on the right* in the descriptional system β iff

$$\forall d,\ 0 \leq d \leq m : LT_\beta(\mathrm{SED}_\beta(x_{-d}))[..l(x_{-d})] + c > LT_\beta(p)[..l(x_{-d})].$$

Intuitively, a description is plausible if it is one of the c-best explanations for x and this holds even if we remove up to m elements from the right of x.

From here, in chapter 6, we also introduced unquestionability in the following way (Definition 6.103):

Definition 8.109 Unquestionability. A fully projectible description p for x is (c,m)*-unquestionable* in the descriptional system β iff it is (c,m)*-plausible* and there does not exist another (c,m)*-plausible* description p' for x.

As we will see later, if c and m are tuned conveniently for a concrete descriptional mechanism, the tests can still be composed of short strings x as exercises such that its $\mathrm{SED}_\beta(x)$ is (c,m)-unquestionable.

This restriction to unquestionable descriptions not only preserves the goal of the test but even strengthens it, in ontological terms. As we saw the plausibility and unquestionability of a theory or explanation not only depends on the intrinsic characteristics of the explanation but also on the ability of finding alternative explanations. In this sense we can see intelligence as the most important means to augment the plausibility and confidence of explanations, and, consequently, the ontology of an 'intelligent' system.

8.4 Establishing Absolute Difficulty

Once we are able to obtain strings whose $\mathrm{SED}_\beta(x)$ is (c,m)-unquestionable for the test, we should ascertain the complexity or difficulty of each problem, in order to be able to give a test set of exercises of different complexity. The first idea is to relate this complexity with explanatory complexity (Et):

Definition 8.110 Comprehensibility (first approach). A string x is k*-hard (*or k*-incomprehensible*) given y in a descriptional system β iff k is the least positive integer number such that:

$$Et_\beta(x|y) \leq k \cdot \log l(x)$$

The use of the factor $\log l(x)$ is to compensate the fact that x must be printed and, therefore, for all x $Et(x) \geq \log l(x)$. Consequently, for all x, $k \geq 1$. As an example, consider a string x of length 256, with $Et(x)= 50$. The comprehensibility of x is k=7.

If Et is substituted by Kt we would have the definition of potential which measures "*the time that needs to be pumped into a number by a computation that finds it*" [Li and Vitányi 1997]. Accordingly, the preceding definition does not measure exactly the complexity of finding an explanation for a string x. Consider for instance the string x = "*wwwwww...*" where w is a string with $l(w) = m$, it has as SED the stable description r = "repeat w for ever" with $Et = m + c + \log c' \cdot l(x)$, which gives $k = (m + c) / \log c' \cdot l(x)$. Consider just $m = 80$ and $l(x) = 256$, and we have that the string can be more than 10-hard, which is quite high for a string whose SED is extremely simple!

To correct this problem we must first realise the reason why we consider a string like r simple. The rationale is that it is easy to give the description if we have the data, i.e., it is obvious or explicit in it. Enquiring this line leads to the definition of information gain we gave in chapter 4.

From here, we can give a definite and corrected version of comprehensibility in the following way:

Definition 8.111 Comprehensibility (Corrected Version). A string x is *k-hard (*or *k-incomprehensible)* given y, denoted by $incomp(x \lfloor y)$, in a descriptional system β iff k is the least positive integer number such that:

$$Et_\beta(x \lfloor y) \cdot G(\mathrm{SED}(x \lfloor y) \mid <x,y>) \leq k \cdot \log l(x)$$

This weighing finally measures the real complexity of finding SED(x) from x, because descriptions of the form "repeat x for ever" which have Kt high (to quote x) are corrected by G, but the length of x is still important.

8.5 The Test

Now we are prepared to construct a generic test of ability of comprehension by generating a series of strings of gradual comprehensibility. However, as it has been said, it is important that the answer is unquestionable, because if not, the answer would be an arbitrary choice from the examiner. An easy way is to give redundant information to make the answer unquestionable. However, we cannot exceed this redundancy too much, because if not, the problems would be much too long. For instance, given the series "a, c, c, a, c, c, c, a, c, c, c, c, a, ..." it seems logical to expect that it would follow "c, c, c, c, c, a, c, ...", so it looks redundant to present more than the necessary symbols.

The measurement that is to be presented below requires the collaboration of the subject, which must employ all its resources to perform the test. It is not relevant whether the subject understands the aim of the test or whether it is programmed to do it. It is even more impartial if the subject does not know that it is an intelligence test. The subject must only know the language and questions the test is composed of.

With these clarifications about the nature of the test, we can define the intelligence of a given system *S* as the value that results from applying the following test to it:

> **Definition 8.112 C-Test.** Let us select a descriptional system β sufficiently expressive and impartial, composed of an alphabet of symbols Ω_β and a set of operations Θ_β to manipulate these symbols, and their corresponding *cost* (or length). We provide (or programme) to *S* the alphabet, operations and cost.
>
> Depending on the expected intelligence of a system we select a sufficiently wide range 1..*K* of difficulty. For each $k = 1..K$ we choose randomly *p* sequences $x^{k,p}$, being *k-incomprehensible*, *c-plausible*, *c-m-unquestionable* and *d-stable* with $d \geq r$, *r* being the number of redundant symbols (or hints) of each exercise.
>
> We measure the intelligence of a pretended intelligent system *S* in the following way:
>
> The questions are the $K{\cdot}p$ sequences without their $d - r$ elements ($x^{k,p}{}_{-(d+r)}$). We give them to *S* and we ask for the following element according to the best explanation that is able to construct with Ω_β and Θ_β. We leave *S* a fixed time *t* and we record its answers: *guess*(*S*, $x^k{}_{-d+r+1}$).
>
> The result of this test of comprehensibility (or C-test) is measured as:
>
> $$I(S) = \sum_{k=1..K} k^e \cdot \sum_{i=1..p} hit\left[x^{k,i}_{-d+r+1}, guess(S, x^{k,i}_{-d+r+1})\right]$$
>
> the function *hit* is usually measured as $hit(a,b) = 1$ if $a = b$ and 0 otherwise (negative values can be used to penalise errors). The value *e* is simply for weighing the difficult questions (if we choose $e = 0$ all the questions have the same value).

In an informal way, "the test measures the ability of finding the best explanation (a fully-projectible description with no alternative fully-projectible descriptions of comparable complexity) for sequences of increasing comprehensibility in a fixed time".

One relevant feature of the test is that, although the subject is supposed to be a particular universal descriptional system ϕ_s with a particular background knowledge (life experience) B_s, it is given a descriptional system β over it, which highly minimises the influence of the difference between the computations performed by ϕ_s and other subject ϕ_t, i.e. the difference between $Et_s(x\,|\,<B_s, \beta>)$ and $Et_t(x\,|\,<B_t, \beta>)$. This makes it possible for the notions of plausibility and unquestionability to be similar for both subjects. Nonetheless, we will see in section 7 how this test can be extended to measure this influence.

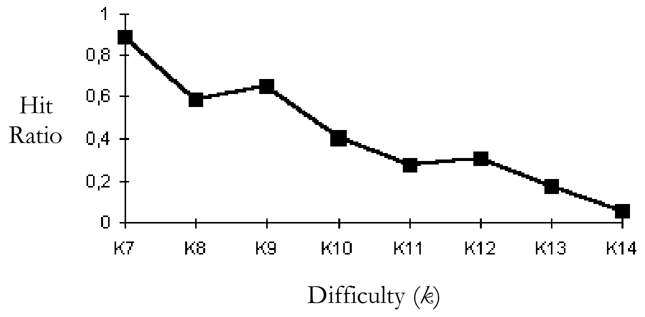
8.6 Measurement of Pretended Intelligent Systems

The preceding test is applicable to any system whose degree of intelligence is questioned. Appropriately selecting the descriptional system and the rest of parameters of the test, it can be used for humans, animals, computer systems, extraterrestrial beings and any collection of the preceding working jointly.

Although Definition 8.112 evaluates a single ability, there are still many ways to realise a specific test from the definition. In [Hernández-Orallo and Minaya-Collado 1998] the test was implemented by using an abstract machine quite similar to a state machine, a simplified version of a *Reduced Instruction Set Computer* (RISC). From here, a variety of strings of different *comprehensibility* in that machine were generated. Although the set of *k*-potent numbers of length at most *n* can be computed in polynomial time in *n* (see a proof in [Li and Vitányi 1997]), the cost of $O(n^k)$ forces to use some heuristics for this. In the same way, *G* was approximated. Finally, a sieve was applied for obtaining only *c-plausible*, *c-m-unquestionable* and *d-stable* sequences. More details are shown in the appendix.

The same work presents the results of applying the test to 65 subjects from species Homo Sapiens Sapiens aged between 14 and 32 years, jointly with a classic test of intelligence, the *European IQ Test*. The correlation between both tests was 0.77. This value only justifies a further more exhaustive study over greater groups and several variations derived from Definition 8.112. For the moment, this psychometric evidence is of vital relevance for a formal theory of measurement of intelligence, since, according to Brand [Brand 1996], the correlation with IQ tests is a necessary condition (but not sufficient) for a good measurement of intelligence.

Another remarkable experimental result shown in the figure below is that the relation between hit ratio and *k*-incomprehensibility is straight, which suggests that comprehensibility really estimates the difficulty of each string:



Logically, it is little interesting to know that the average Homo Sapiens is able to 'understand' sequences of incomprehensibility = 10 in a reasonable time. Similarly, it is not expectable (for the moment) that contrasted and widely used IQ tests were substituted by these C-tests. Nonetheless, this could entail a milestone in the theoretical foundation of psychometrics because it is the first measurement of an intelligence factor that is based theoretically and not using the Homo Sapiens as a reference.

However, it is not human intelligence but non-human what is urgent to measure. A formal declaration of what is expected from an intelligent system should allow two important things: to derive more intelligent systems from a more concrete specification and, secondly, to evaluate them. Definition 8.112 provides a first step for both things, a detailed scale for measuring the progress (in one intelligence factor) of generic systems in AI. As any other field of science, a great advance in a discipline happens when one of its fundamental topics can be measured in an effective and justified way. Artificial Intelligence, as a science, requires measurements of intelligence, or at least, measurements of their different factors.

Modern AI systems are much more functional than systems from the sixties or the seventies. They solve problems in an automated way that before required human intervention. However, these complex problems are solved because a methodical solution is found by the system's designers, not because current systems are more intelligent than preceding ones. No one cares about measuring how functional these systems are for other kinds of problems, since "*it's easier to evaluate systems that do specific things than it is to evaluate systems whose tasks are more general*" [Nilsson 1995]. The current oblivion of general problem solvers may be technologically correct with an applications demanding discipline but not fair with A.I. foundational name. "*It is time to begin to distinguish between general, intelligent programs and the special performance systems*" [Nilsson 1995].

This initial aim of being more general is nowadays still represented by two subfields of artificial intelligence: automated reasoning and machine learning. Automated theorem provers are able to solve complex problems from different fields of mathematics. The great advance of the last two decades is mainly caused by the existence of sets of problems to compare different systems. Even these sets have evolved and grown to huge and complete libraries of theorem proving problems, such as TPTP [Suttner and Sutcliffe 1996]. Machine learning is also taking a more experimental character and different systems (from different paradigms) are evaluated according to classical problems in the literature.

However, as we discussed in the introduction, there is no theoretical (nor empirical) measurement about the complexity of the problems that compose the sets of problems. By selecting a proper representational language we could compute this complexity. For instance, if we select first-order logic as a universal descriptional mechanism, we can measure the complexity of the evidence, hypothesis (explanation) and background knowledge in the same way by using, for instance, a measure of the length of a logic program, like those seen in the previous chapter. This would allow, for instance, to give the theoretical complexity of the problems that are usually passed to ILP systems and discover how intelligent they are.

8.7 Factorisation

The previous test measures one factor, which could empirically be identified with the g factor or liquid intelligence. During the XXth century, psychometrics and zoology have striven for differentiating between evolutionary-acquired knowledge, life-acquired-knowledge and '*liquid intelligence*' (or individual adaptability). While the first one is the most important one in the great majority of lower animals, the latter two characteristics distinguish the animals with the ability to learn from dull animals. However, the higher the animal scale the more difficult to distinguish life-acquired-knowledge from 'liquid intelligence'. The case is extreme for human intelligence. Accordingly, exercises from IQ tests are strictly selected to avoid the influence of background knowledge. As a result, the scores obtained in the tests of liquid intelligence (*g*) are almost constant from the age of 14 until 60, independently from the education and knowledge that can be acquired during all this time.

A complete test of intelligence should only measure these knowledge-independent abilities to still reflect the possibility of "idiots savants", i.e., systems with little intelligence but a lot of embedded knowledge. Even with liquid intelligence, there are many knowledge-independent abilities (or factors) to measure. For instance, memory or 'memo-isation ability' is a factor that is knowledge-independent that can be easily measured. However, this factor is not very interesting for AI nowadays, because it is not a technical problem to make systems with large memories (and it is not correlated much with *g* either).

Let us review which inductive and deductive factors are feasible and interesting to measure:

8.7.1 Inductive Factors

There are partially independent factors that could be measured by using extensions of the framework presented in the previous section. For instance, knowledge applicability, contextualisation and knowledge construction ability can be measured in the following way:

- Knowledge Applicability: we provide a background knowledge B and we give a set of sequences x_i such that $incomp(x_i | B) = incomp(x_i) - u$ but still $\text{SED}(x_i | B) = \text{SED}(x_i)$ and are unquestionable with or without B. We can compare the difference of performance between cases with B and without B. This test would actually measure the application of the background knowledge depending on two parameters: the complexity of B (i.e. $Kt(B)$) and the necessity or usefulness of *B*, measured by *u*. Cattell called this *crystallized intelligence* (*gc*) which correlated with fluid intelligence (*gf*).

- Contextualisation: it is measured in a similar way as knowledge applicability but providing different contexts $B_1, B_2, ..., B_T$ with different sequences $x_{i,t}$ such that $incomp(x_{i,t} | B_t) = incomp(x_{i,t}) - u$. This multiplicity of background knowledge (a new parameter *T*) differentiates this factor from the previous one. Analogy tests generally resemble this type of exercises, as it was shown in [Hernández-Orallo and Minaya-Collado 1998].
- Knowledge Construction: we provide a set of sequences x_i such that exists a common knowledge or context *B* and a constant *u* such that for $incomp(x_i | B) \leq incomp(x_i) - u$. A significant increase of performance must take place between the first sequence and the later sequences. The parameters are the same as the first case, the complexity of *B* and the constant *u*. This learning from precedents has also been studied in AI (see e.g. [Winston 1982]).

Other factors are more related with *intentionality* than general intelligence (and *intensionality*). These are reactivity, pro-activity, interactivity and the recently elsewhere vindicated emotional abilities. Most of them can be measured adopting notions from Query Learning paradigms [Angluin and Smith 1983] [Angluin 1988] formalised using interactive Turing machines. Others are much more related with the idea of congruence or coherence and could be measured as constraint satisfaction [Thagard 1989].

8.7.2 Deductive Abilities

Deductive abilities are much easier to measure, because there is no question about selection criterion and, consequently, there is no possible subjectivity in the correct answer; given the premises and the way to operate with them, only one plausible answer is possible. We must adapt the measurement of gain introduced in chapter 4 for different deductive systems to give a version of difficulty of deduction (similar to comprehensibility for inductive problems).

To avoid the subjectivity objection, we must just give a set of premises *x*, and only a possible conclusion. This in fact, represents a deductive problem or calculation. Note that any deductive system can be converted in this form if we add to the premises the needed restrictions in order to allow just one possible conclusion. The exercise must evaluate the ability of the subject to find this unique conclusion *c*.

Consequently, it is only needed to ascertain the difficulty of each instance in a similar way as we did with comprehensibility.

With $x \vdash_p c$ we denote that *p* is a proof for *c* in *x*. First of all, we must evaluate the complexity of this *p* as:

Definition 8.113 Deductive Effort. The effort of the deductive inference of the proof p over the system x is given by:

$$LT^{ded}(p) = l(p) + \log Cost(\text{apply } p \text{ to } x)$$

Logically, we can define the best proof as the proof with less effort:

Definition 8.114 Best Proof. The best proof of a conclusion c with respect to a set of premises x is:

$$BestProof(c \mid x) = \text{argmin}_p \; \{ \, LT^{ded}(p) \text{ s.t. } x \vdash_p c \, \}$$

A first approximation of the deductive complexity can be given by:

Definition 8.115 Deductive Complexity. The deductive complexity of a problem x whose only solution is c is given by:

$$Et^{ded}(x) = LT^{ded}(BestProof(c \mid x))$$

Analogously as we did with comprehensibility we can define:

Definition 8.116 Solvability (First Version). A deductive problem x is *k-solvable* iff k is the least positive integer number such that:

$$Et^{ded}(x) \cdot G(BestProof(c \mid x) \mid x) \leq k \cdot \log l(x)$$

Note that G is computed in the classical way: $Kt\,(p \mid x) \;/\; Kt\,(p)$.

However, in this case we have not eliminated the possibility of finding alternative proofs, and this can affect the difficulty to find the answer. For instance, consider a problem with just one proof (way) to the solution with solvability 10, and another problem with many proofs (ways) to the solution with solvability 11. Which one is easier?

To correct this problem we must consider all the alternative proofs.

Definition 8.117 Proof Distribution. The proof distribution of a deductive problem x with respect to a conclusion c is given by:

$$\delta(x,c) = \log \sum_{x \vdash_p c} 2^{-(LT_{ded}(p) \cdot G(p \mid x))}$$

From here we can introduce a final version of solvability:

Definition 8.118 Solvability (Corrected Version). A deductive problem x with solution c is *k-solvable* iff k is the least positive integer number such that:

$$\delta(x,c) \leq k \cdot \log l(x)$$

Once again, there are partially independent factors that can be measured from here. In this case, however, they turn to be closely related and they turn to be different presentations of the same problem:

- Calculus Ability: Given an x, only one c is possible (a problem). The subject must obtain c. However, the rules can only be applied in a few ways, so it is only a mechanical application of them. That is to say, there are few ways to act, and only it is required that the subject applies correctly the rules, but they are quite clearly determined. Obviously, the difference between $\delta(x,c)$ and $Et^{ded}(x) \cdot G(BestProof(c|x) \mid x)$ is small.

- Problem Solving Ability: The problem is similar to the previous case but, in this cases, there are many possible ways to apply the rule which do not lead to any solution.
- Accepter Ability: Given an x, some c's are provided. Only some of them are consequences of x. The subject must discern which ones (this is a generalisation of the calculus ability).
- Derivational Ability: Given an x, the subject must obtain the greater number of correct derivations as it can in a limited time. For this case we should extend the previous definition for multiple conclusions.

It is not expected that these deductive abilities would be independent to the previous inductive abilities (as it has been shown by psychometrics for the Homo Sapiens). The reason, however, has theoretical roots. As we commented, any inductive process requires deduction to check the hypotheses, thus, obviously, inductive ability is influenced by deductive ability.

Nonetheless, deductive ability is also influenced by inductive ability as long as the problems get harder. Some lemmata or rules can be generated by an intelligent subject in order to help to shorten an ease the proof from the premises to the conclusion. This may explain why artificial problem solvers without inductive abilities have not been able to solve complex problems, and this is especially clear in Automatic Theorem Proving.

8.7.3 Other factors

Other factors usually found in psychological tests are 'verbal ability', 'visual ability', 'calculation / deductive ability', etc. Some of them depend on background knowledge and are difficult to measure if the system does not have a base or some important perception abilities.

Many other factors could be measured justifiably by information-theoretic means to different kind of systems: animals, A.I. systems, machine learners, etc. However, not every factor is meaningful for intelligence. Factors such as "playing chess well" are much too specific to be robust to background knowledge. Other factors will result in being highly correlated to other more distinct factors. This correlation cannot only be established experimentally like in psychometrics but theoretically, as deduction ability can be shown to be correlated with knowledge applicability, or some learners have been shown to be formally equivalent to interactive proof systems.

The influence of the descriptional mechanism should also be studied for each factor. In the same way, some variants in the test could be made by using middle gaps instead of sequence predictions. For the comprehensibility factor, this change of presentation (such as an abduction problem) was studied also in [Hernández-Orallo and Minaya-Collado 1998] and no significant difference was perceived with respect to to the (inductive) previous presentation.

In the end, the question is to refine and extend all the previous ideas in order to make different and founded tests of intelligence, knowing exactly what is measured. I think that this is an urgent and fascinating task for artificial intelligence.

8.8 The C-test and The Turing Test

The imitation game was conceived by Turing to dissipate the doubts about possibly non-human intelligent beings. He left no place for human's exclusivity: intelligence can be evaluated by an exclusively behavioural test; the rest of details (nature, introspection, ...) are irrelevant. Unfortunately, instead of recognising this his most important contribution, the test was and is still understood as 'a goal' in AI. Nonetheless, this view has been responded by many authors, which criticise that the TT does provide little information about what intelligence is; it is just a test of humanity [Fostel 1993], that, in fact, if applied to human beings, gives many paradoxes. The result of applying it to ourselves is a recursive trap (for self-evaluation) which is unable to answer the question of whether we are intelligent or not, or more precisely, how intelligent the Homo Sapiens is.

Some authors have tried unsuccessfully to correct the two main problems of the Turing Test for measuring intelligence: its informal character and its anthropocentrism. In some cases, this has led to disparate proposals, as the so-called formalisation of the Turing Test [Bradford and Wollowski 1995], sustained from the, in my opinion folly, assumption that we are non-deterministic machines able to solve NP-complete problems in polynomial time. As [McCarthy 1998] clarifies: "*humans often solve problems in NP-complete domains in times much shorter than is guaranteed by the general algorithms, but can't solve them quickly in general*".

There is still a third problem, which is the necessity of several intelligent 'judges' and a 'referent' to make the test. The self-reference question arises again: Who is the first intelligent being to start the game? These and other problems are incarnated in the Loebner Prize, which usually awards the participant who has devised the system more able to cheat the judges, because "*humans are surprisingly bad at distinguishing humans from computers*" [Johnson 1992]. In the end, there is no way of knowing who is cheating, the system or its designer.

However, if fairly played, the imitation game is a hard examination for any pretended intelligent system. It is extremely difficult to behave like an average human being of this epoch (it is even difficult for some human beings). For a non-human-contextualised being, it would be required to comprehend the complex behaviour of human beings of these times, their evolution-acquired traits, their language, their culture, their limitations, etc. It is much easier then to try to cheat the judges. In fact, the judges "*are especially fooled into reading structure into chaos, reading meaning into nonsense. (...) Sensitivity to subtle patterns in our environment is extremely important to our ability to perceive, learn and communicate*" [Shieber 1994].

Curiously, it is precisely this 'lack' of the judges, reading structure everywhere, what the C-tests measure. In fact, the C-tests are difficult to cheat, they are not anthropomorphic, do not require any judge which must previously be determined as intelligent and give an independent and possibly multi-dimensional value (and not a Boolean answer). However, the C-tests, as they have been presented, are necessary (at least to obtain a minimum value of $I(S)$) but not sufficient (other important factors should be measured as well). It has been already suggested that both kind of tests (TT and factorial) could be combined in order to give a more accurate test of intelligence: "*it is this posing of puzzles in arbitrary domains that is the hardest part of the Turing Test, and a part that no program has yet passed*" [Shapiro 1992]. This idea, however, would ultimately turn the TT into a lightweight and less rigorous version of the C-Test.

In my opinion the TT should be celebrated as an extremely valuable philosophical exercise about the behavioural character of intelligence. However, in practice, it should be substituted by progressively more accurate computational and factorial tests of different cognitive abilities.

8.9 Summary and Contributions of This Chapter

We have taken an important step for the formal measurement of intellectual abilities. Different measures of cognitive abilities are presented, in special a measure of comprehension ability, which finally correlates with the classical *g* factor.

Section 2 has presented the requirements that are needed for a proper measurement of intelligence, and, in general, of any intellectual ability: non-Boolean, factorial, non-anthropomorphic, computational and meaningful. Some technical difficulties are immediately found when this is tried to made directly from descriptional complexity. Once the notion of comprehensibility is recovered from chapter 6, section 3 is devoted to solve the 'subjectivity objection' under the notion of unquestionability, also presented in chapter 6. Another important question is to order reasonably the difficulty of instances, which is solved in section 4. Consequently, the construction of a comprehension test (C-test) is then presented in section 5. Section 6 presents the results of applying it to humans and compares it with psychometrical tests. The applicability to artificial intelligence is discussed. Section 7 studies the measurement of other factors, inductive (knowledge applicability, contextualisation, knowledge construction) or deductive (calculus ability, problem solving ability, derivation ability) under the same conditions that the C-test has been devised with.

After the previous results and auspices, the Turing Test is re-examined in section 8 and reduced to its original philosophical and even metaphorical character. Compared with the C-Test, the significance of the TT is recognised, as well as the acute deficiencies of its misinterpretation and incarnations, such as the Loebner Prize.

As a result, the main contributions of this chapter are:

- A non-anthropomorphic test of intelligence, which is based on computational and information-theoretic notions, which can make an important advance in the evaluation of AI progress.
- Different fields of AI can adapt these measures to evaluate automated induction / deduction / reasoning systems. For instance, the difficulty of problems that are classically used by the ML community can be classified by their comprehensibility in order to know which ML algorithms are better. In the same way, automated theorem provers can be evaluated in a less experimental and arbitrary way than by using large collections of test sets (such as the TPTP library [Suttner and Sutcliffe 1996]) whose intrinsical difficulty is not known.
- Psychometrics finds its long-awaited theoretical foundation in information theory and computation, and opens the door for more experimental and theoretical research.
- In special, factor independence can be studied theoretically and not only experimentally, as psychometrics has been doing during the last fifty years. If these inter-dependences are clarified, shorter and more precise tests could be devised in the future.

The idea of the Turing Test as a practical test of intelligence should be left behind, and substituted by computational and factorial tests of different cognitive abilities, a much more useful approach for artificial intelligence progress and for many other intriguing questions that are presented and that now it is feasible to answer.

8.10 Appendix. An Example of C-Test

This appendix appeared just as it is in [Hernández-Orallo and Minaya-Collado 1998] and it is included here to show how a test can be implemented and some of its results:

The problem of selecting a good bias for generating *k-hard* strings depends on many factors. The objective is to maintain expressiveness, to ease the problem of finding explanatory descriptions and to limit the combinatorial *explosion*. The final choice we present is an oversimplified abstract machine that is easily extensible to work as a Turing machine.

8.10.1 A Toy Memory-less Abstract Machine

Due to the current technology of the computers we can use, we have chosen an extremely abridged emulation of the machine that will *effectively* run the programs, instead of more proper languages, such as λ-calculus (or LISP). We have adapted the "toy RISC" machine of [Hernández-Orallo and Hernández-Orallo 1993] with two remarkable features inherited from its object-oriented coding in C++: it is easily tunable for our needs, and it is *efficient*. We have made it even more reduced, removing any operand in the instruction set, even for the loop operations. We have only three registers, which are AX (the accumulator), BX and CX. The operations Θ_β we have used for our experiment are in Table 1:

LOOPTOP	Decrements CX. If it is not equal to the first element jump to the program top.
LOOPS	Same as LOOPTOP but it jumps n (for the tests n=4) instructions backward.
LOOPM	Same as LOOPTOP but it jumps m (for the tests m=7) instructions backward.
SUCC	Increments the accumulator.
PRED	Decrements the accumulator.
WRITE	Writes into the output and moves fwd.
BREAD[2]	Moves back and reads from the output.
FREAD[2]	Moves fwd and reads from the output.
MOV A,B[1]	Copy register BX into AX
MOV B,A[1]	Copy register AX into BX
MOV A,C	Copy register CX into AX
MOV C,A	Copy register AX into CX
ROTR[3]	Rotates 45° to the right.
ROTL[3]	Rotates 45° to the left.

Table 1. Instruction Set

The operations with no superscript are present in all the subsets. Operations marked with (1) are present in the 'professional' version of the machine, the operations with (2) are present in the *Turing-like* version and those with (3) are present in the *Logo* version where the output is bidimensional. This sparseness of only 10 operations will be clearly justified later. We have essayed with many different alphabets but for this test we will use the professional version and a circular alphabet $\Omega_\beta = \{a,b,c,d,...,z\}$, i.e., incrementing 'z' yields 'a' and decrementing 'a' yields 'z'. Since the first element is an inflexion point for the loops, it is presented to the subjects as "*a critical element*".

This configuration *still* produces many programs that are not robust (intensional) because programs can be often split into subprograms. The solution for these cases comes from another restriction: the programs must be comprised wholly inside a loop. This leaves a good approximation to explanatory programs. The rest to do is to avoid repetitions of patterns such as "abcabcabcabc" (for sake of gain and plausibility) and take apart the strings where an important part is explained by a shorter program (for sake of intensionality). We think that the bias is not all the expressible we would like but it allows the generation of strings of certain complexity. Also we think it is *fair* because it does not relate on arithmetic (such as *cryptarithmetic* tests) or any other preceding knowledge, except the order of the alphabet.

8.10.2 The Generation of *k*-Hard Strings

The algorithm we have used to generate a set of different *k*-incomprehensible strings is very similar to the one we presented in section 5.4 (of [Hernández-Orallo and Minaya-Collado 1998]). Having 10 operations, we have that usually only about a 20% of the programs of any size are explanatory. This means that trying to know whether a randomly generated program of, say, size 15, is valid, will need the checking of more than 2,222,222,222,222 programs. And this is the case if the computational cost of x^* is slow, contrariwise (if x^* is a *costly* program) we will have to check longer programs.

We have used some optimisations and heuristics in order to make the great amount of programs to check more tractable. Some examples of questions are:

Prediction style:

*k*9:	a, d, g, j, …	Answer: 'm'
*k*12	a, a, z, c, y, e, x, …	Answer: 'g'
*k*14:	c, a, b, d, b, c, c, e, c, d, …	Answer: 'd'

Abduction style:

*k*8:	a, _, a, z, a, y, a, …	Answer: 'a'
*k*10:	a, x, _, v, w, t, u, …	Answer: 'y'
*k*13:	a, y, w, _, w, u, w, u, s, …	Answer: 'y'

8.10.3 The Tests

Four tests were devised to measure prediction, abduction, g-factor and similarity. The prediction test is composed of 19 exercises generated with the following *k*-hardness distribution (2 *k*7, 1 *k*8, 2 *k*9, 3 *k*10, 3 *k*11, 3 *k*12, 2 *k*13 and 1 *k*14), redundancy $r = 2$ and the less 'akin' as possible. The abduction test is composed of 15 exercises using the same generator and redundancy. The distribution was (2 *k*7, 2 *k*8, 1 *k*9, 2 *k*10, 1 *k*11, 3 *k*12 and 4 *k*13). In these two tests, the incorrect options were generated randomly but relative near to the solution and the letters appearing in the string. The

IQ test we used was the European IQ test simply because it is a culture-fair test, devised for 20 minutes, ensuring a reasonable range (75-174) of values and available on the Internet. The similarity test is composed of 8 exercises generated with binary strings of different length and different levels of edit errors (insertion, deletion or change). The strings were generated and checked by dynamic programming to ensure that they did not have a better correction path. The purpose of this test was to measure the ability of compression by *trivial* pattern matching.

8.10.4 Subjects and Administration

Subjects were selected from two different groups: the first group was composed by 48 high-school students with ages comprised between 14 and 18 years. The second group was composed by 17 subjects of a mixed sample of undergraduate and postgraduate university students with ages comprised between 22 and 32 years.

All the tests were passed in the same session. The times were, without including instructions, 10 min. for the prediction test, 5 min. for the abduction test, 5 min. of break, 20 min. for the IQ test and 3 min. for the similarity test.

8.10.5 Results

We evaluated the test without penalising the errors, i.e., the function *hit* evaluated the same for blanks than for mistakes. We chose e=0, i.e. all questions with the same value. IQ-correlations are illustrated in Table 2.

	Pred.	Abd.	Induct.	Simil.
High-School	0.31	0.38	0.42	0.39
University	0.51	0.42	0.56	0.35
Both Groups	0.73	0.68	**0.77**	0.50

Table 2. Correlations with EIQ test

The correlation for induction (prediction + abduction) is of the same order as the usual correlation for induction tests made by psychologists. The correlation between the abduction and prediction tests was 0.61, less than expected, which suggests that even problems constructed by the same generator can be more or less difficult depending on its presentation (abductive or predictive). The correlation between induction and similarity was 0.51, which supports the thesis that "the ability of compression" is different from "the ability of comprehension". Finally, we think that an analogy test based on our theory would surely round off the study.

With these data and our amateur methods we are not in conditions to assert more things about the relation between C-tests and IQ-tests. There is only a thing that has no discussion in the light of the results, the k-hardness matches fairly well with the difficulty people found on them, as it is seen in Figure 1:

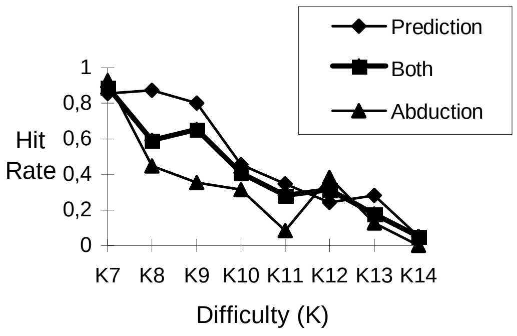


Figure 1. Hit Rate per Difficulty

9. Prospective Applications

I never waste memory on things that can easily be stored and retrieved from elsewhere

Albert Einstein (1879-1955)

Abstract: *this chapter includes some proposals in different fields. The first application discusses the optimal representation for deductive databases, according to the optimal representation seen in chapters 3 and 4, in order to improve the performance of a database depending on which operations are more frequent and the degree of regularity of the data. Another application is the study of validation and maintenance characteristics of software systems under the analogy between software science and philosophy of science or, more precisely, between software construction and machine learning. Reinforcement measures from chapter 5 are adapted to define a measure of software 'predictiveness', which is identified with software validation, to represent the stability of a system. An inversely related measure, the probability of modification, is also obtained for each component and for the whole system. Some models of maintenance are considered, and different software arrangement topologies are studied theoretically under them. Finally, some other applications are outlined, especially related with meaning and language, and their applications to agents communication.*

Keywords: Databases, Data Mining, Data Quality, Software Engineering and Maintenance, Software Topologies, Knowledge-Based Systems, Meaning and Language.

9.1 Introduction

In this chapter some other *prospective* applications are presented. This includes new theoretical tools and adaptations of the concepts that have been presented in this work to quite different fields. Thus the term 'prospective' indicates that there is not a second stage of experimentation of the theoretical items and models which are advocated here, which will judge the goodness of these proposals in the end.

In Section 2, we will study the applications of information gain for *information systems*. After a brief description of predictive data mining, which is an application of machine learning (ML) techniques for obtaining knowledge from databases, we study the possibility and usefulness of non-predictive data mining. More concretely, according to the optimal representation measure seen in chapters 3 and 4, we will discuss which would be the optimal representation for deductive databases, in order to improve the performance of database operations depending on which operations are more frequent and the degree of regularity of the data. Once the physical level is separated from the logical question, the intensional relationships which are found in a database on a higher level are much more important for the data quality of a system, in order to control consistency and redundancy of the data. Finally, both deductive and inductive processes (and their integration) will be increasingly more important in future databases, which will be better known as knowledge bases or knowledge systems.

In Section 3, validation and maintenance characteristics of software systems are reconsidered under the analogy between software science and philosophy of science or, more precisely, between software construction and machine learning (ML). From this outset, many classical techniques from ML can be used. In particular, the reinforcement measures from chapter 5 are adapted to define a measure of software 'predictiveness', which is identified with software validation, to represent the stability of a system. An inversely related measure, the probability of modification, is also obtained for each component and for the whole system. The application in practice of these measurements is discussed. From here, some models of maintenance cost are presented, based on a detailed combination of predictiveness and modifiability. Different software arrangement topologies are studied theoretically. Hierarchical topologies, especially downward confluent ones such as trees and lattices involve less maintenance costs. Moreover, some intuitive expectations are confirmed, namely that compressed systems and coherent models (without patches or exceptions) are manifestly more maintainable.

In Section 4 some other applications are outlined, especially related with interaction and mutual understanding, some questions related with meaning and language, and their applications to agents communication.

9.2 Representational Data-Mining and Data Quality

It is usually said that traditional databases are extensional. This assertion is increasingly less true as long as database technology has been advancing. In a relational database, constraints are intensional definitions, which can be expressed as first-order formulae. More importantly, views are derived relations expressed from the base relations and other derived relations.

It is not strange that, from a theoretical point of view, databases have been seen as deductive systems where a great proportion of the data is in an extensional way. Concretely, the most widely used model, the relational model, understands a database as a first-order theory, where each relation is seen as a predicate.

Recently, there is an interest for using other database models that allow expressing more intensional properties intrinsically. For instance, object-oriented databases can include any inclusion property of the world as a concrete inheritance relationship in the database. The result, however, is the same; an intensional definition is taking place in the database. A classical example of this is a relation such as "person(X) :- employee(X)".

Almost any modern data model allows intensional definitions (in the worst case, a model must allow the definition of views, which are intensional definitions). In the end, it is necessary an elicitation of which parts must be left in an intensional way and which parts must be left in an extensional way. In fact, this is one of the most problematic questions in database design. As [Blockeel and De Raedt 1995] point out "*When designing databases, the designer has to determine the structure of the database by determining the extensional and intensional predicates, and by providing definitions for each of the intensional predicates. (...). The design ultimately determines the quality of the database*".

I completely share this view, however, I do not share their criterion: "*the better* (sic) *database is the more* (sic) *compact one, i.e., the one that requires less memory*". In my opinion, things are much more complex. The best database should be measured according to the representational optimality seen in chapter 4, namely:

Definition 9.119 The representational optimality of a database is given by:

$$\mathrm{Opt}(db|\, E) = \alpha \cdot l(db) + \beta \cdot Cost(E|\, db)$$

with *db* being the database and *E* the evidence or the whole of data that the database must accumulate.

However, a database is not a stored picture, which is static and has only a view. There are many operations and partial recovers (queries) that are to be performed over it. Consequently, *Cost*(*E*| *db*) must measure the cost of the most typical queries, the cost of updates, the access to partial information, etc.

It is clear that traditional compression techniques such as Lempel-Ziv algorithm [Lempel and Ziv 1977], require uncompressing a large portion of the file even if only a small part of that file is required. Consequently, there have been some proposals of compressing algorithms specifically design for databases [Moffat and Zobel 1992] [Goldstein et al. 1998].

A Database Management System (DBMS) tries to obtain the compromise that is represented by Definition 9.119. However, this depends on many factors which are independent from the data itself, well-known in database literature: secondary memory is slower than primary memory, the size of buffer blocks and buffer pool, the primary memory which is available [Elmasri and Navathe 1994]. In other words, Definition 9.119. is only of theoretical interest. In practice, well-studied structures have been implemented: indexes, B-trees, R-trees, snapshots... [Date 1995][Elmasri and Navathe 1994]

The discussion about the degree of intensionality of a database is then more centred at a higher level, namely, at the conceptual level, as it has been studied by Hull's paper on the information in a relational database schema [Hull 1984]. The main question is avoidance of redundancy, but it is important to realise that the degree of compression at the physical level is independent to the degree of redundancy at the logical and conceptual levels. A DBMS, which compresses the tables of a database, can still suffer from redundancy if functional dependences exist in the schema of the database or derived information is maintained explicitly in the database.

Let us see first of all how this redundant information can be detected in databases and then let us study in which cases it is convenient to eliminate this redundancy.

9.2.1 Knowledge Discovery in Databases (KDD)

Theoretically, the design of a small schema can be done in order to minimise the redundancy of the data that it may hold. However, as long as more data is available and it is automatically added to existing databases, the control of this redundancy must also be partially automated. A new emerging field is tackling this problem: data mining, or, more generically, knowledge discovery in databases (KDD).

More precisely, data mining is just a part of this complex process as it is shown in Figure 9.1; KDD includes Data Preparation, Data Mining itself, Interpretation/Evaluation and sophisticated Visualisation tools. The whole process transforms data into knowledge, because the input of the process is extensional information from databases and the output is intensional information in a comprehensible language.

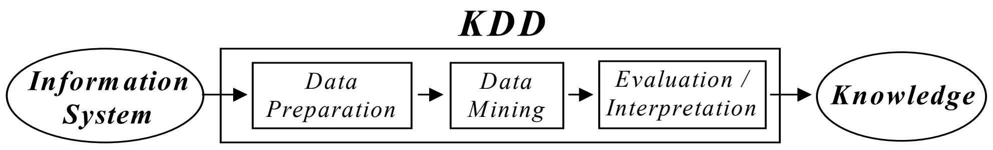


Figure 9.1. KDD process

Note that the formalisation of the transformation from "data into knowledge" must depend on an evaluation of the knowledge (or theory) with respect to the data (or evidence) in a similar way as it has been done throughout this work. Nonetheless, KDD can be informally defined as: "*the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data*" [Fayyad et al. 1996b].

Let us examine the properties that these patterns which are discovered in the data must follow and let us relate them with important concepts that have been introduced in this work:

- *valid*: it is the most important issue. A pattern or property holds when an important part of the data is compliant with it. This is directly related to the notions of plausibility. An ideal measurement for *validity* can be reinforcement, especially, as it has been applied in chapter 7 to logic programs.
- *novel*: it is not a prerequisite but a desirable property. As we said in chapter 4, to talk about a real discovery, the pattern cannot be explicitly in the data. The more a KDD discovers, the more implicit the patterns and properties were in the data.
- *useful*: this is logically measured a posteriori. If a valid and novel pattern has been discovered, it may or may not be used for other/future data to detect inconsistencies or predict new information. As we saw in chapter 5, reinforcement could be used as well to now the usefulness of a concept, by measuring to how many other cases it is applicable.
- *understandable*: this is not a strict requirement, either. A difficult pattern can be discovered by a KDD system and used internally. However, if a human expert wants to apply the knowledge that has been obtained to make a decision, to explain some fact, etc., the concept cannot be a hard numerical formula but a comprehensive concept. In this case, a measure of comprehensibility, as introduced in chapter 6 and used in 8 can be used profitably.

After these desirable properties for a discovered pattern, let us see which kinds of patterns can be identified from a database. Theoretically, any complex property can be discovered. However, there are specific patterns which are more useful (and common) to obtain from a database. Concretely, functional dependences [Mannila and Räihä 1994], are properties, relational in nature, which do not accept exceptions. An example of these could be the relation of surname between father and child. Other systems are based on association rules [Agrawal and Srikant 1994] [Mannila et al. 1994] that are probabilistic and hence allow exceptions. Associations are propositional, such as the relation between foggy weather and flight delays. Other systems allow exceptions in relational dependences, under the name of partial determinations [Pfahringer and Kramer 1995] or probabilistic functional dependences [Akutsu and Takasu 1994].

More expressive approaches are represented by ILP application to data mining [Brockhausen and Morik 1997] [Dzeroski 1996] [Morik 1997] [De Raedt and Dehaspe 1997]. ILP allows the generation of richer concepts (more comprehensive). Theoretically, any first-order property or constraint can be expressed and hence obtained by ILP.

The final question is whether these findings of relationships, properties or patterns should affect the schema of the database or they should only be used as knowledge to be interpreted by an expert. The following section claims that this decision must be taken according to 'data quality' criteria.

9.2.2 Relationship between Intensionality and Data Quality

Data Quality is defined as the accuracy to which an information system reflects the reality. All the features that are relevant are reflected by the database (completeness) and all the data of the database is true in reality (correctness).

The first relation between data quality and data mining has been established in the cleansing phase of a KDD system. Namely, if the information system does not reflect accurately the reality, this verification would surely fail, independently of how good the KDD system could be. In [Cortés et al. 1995], it is formally shown how "*random errors and insufficiencies in databases limit the performance of any classifier trained from and applied to the database*".

Since it is impossible in general to validate completely an information system with reality, an effective and accurate measurement of the data quality of the database could be extremely profitable for KDD. However, this measurement is more useful all along the KDD process, before, during and after the data mining process. This approach is taken in [Hernandez-Orallo and Alamagnac 1999].

For the case of data quality, it is more important to see also the other way: data mining can improve data quality. The relation between the environment and the database is bidirectional, i.e., there is a lot of acquisition from the reality to the database system and there are many outputs (or answered queries) from the database which are returned to reality. This process can detect acquisition (or operation) errors, which provoke an important feedback from reality to the database, as it is shown in the left-hand side figure 2.

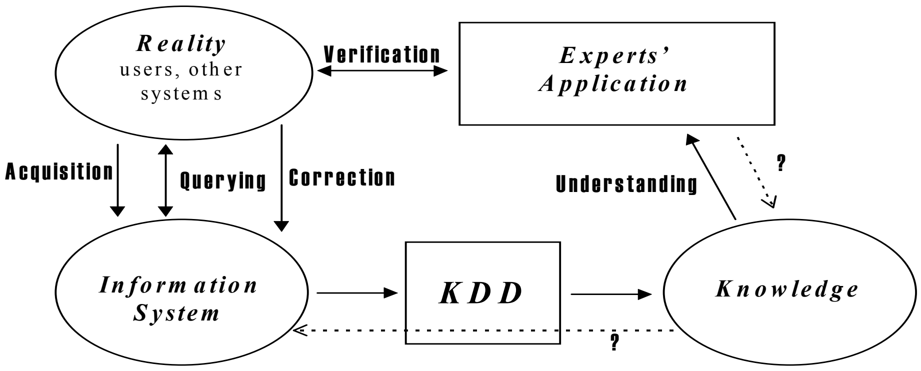


Figure 9.2. KDD and validation with reality

However, if a KDD produces knowledge, this knowledge can be used to contrast with reality or to correct some other parts of the information system, in a similar way as cognitive or scientific experimentation, validation and refutation.

In this way, information system should be increasingly *smarter*, in the way that both induction and deduction should be used to maintain the data consistent, internally and externally.

Let us see an example where a data-mining algorithm can find new constraints that can be useful to correct errors in a database, i.e., to increase its quality.

Example 9.14

Consider a small hospital database with 20,000 case histories. A data-mining system discovers the following functional dependency:

P1: patient(X, _, ..., _, male) :- illness(X, _, ..., _, colour-blindness)

With 150 cases of colour-blindness in the hospital, from which 144 were men, 1 women and the other 5 do not have any value for the 'sex' attribute. The generality degree, as it was introduced in chapter 7, would be *GD*= 149/144 = 1.03.

Due to one error, this is considered a partial dependency. However, if a user or an expert is informed of this property, she may know that colour-blindness only affects men and she concludes that there is an error in a database, since colour-blindness cannot affect women. Moreover, the rule can help to complete the record of the other 5 people, which, necessarily, should be men.

Another question that suggests the previous example is whether this property should affect the schema of the database. In this case, the most reasonable change is the addition of *P1* as a restriction, but the frequencies of men, women and colour-blindness do not suggest a change in the definition of the tables.

Example 9.15

In the same database as Example 9.14 consider the following table:

Patient			
SSN	...	...	*Penicillin_Allergic*
52677328	...	...	false
...			...
...			...
67616274	...	...	false

where only 4 patients from 20,000 are allergic to penicillin. A data-mining system discovers this situations and suggests the following change to the database:

Base_Patient		
SSN	...	...
52677328	...	...
...		
...		
67616274	...	...

Penicillin_Allergic
SSN
24254151
76327675
5254151
61616166

and Patient is defined as a view, namely:

```
CREATE VIEW patient' AS
SELECT base_patient.*, false FROM base_patient
WHERE SSN not in (SELECT * FROM penicillin_allergic)
UNION
SELECT base_patient.*, true FROM base_patient B, penicillin_allergic P
WHERE B.SSN= P.SSN;
```

This would save space in the database, (theoretically, 20,000 bits - 32 · 4 = 19,872 bits). However, the schema would be more complicated and there could appear some problems of updates or integrity with the view that could finally advise against the change.

Only in some special cases, where classical design rules of relational databases are not observed (normal forms, many-to-many relationships, etc), the schema should be changed.

Many other factors even justify the existence of redundancy in a database. For instance, derived attributes (such as the number of surgical operations) may be useful for obtaining frequent information which must be looked up in other tables, mainly historical tables. Derived attributes do not affect information quality if they are congruently and automatically maintained.

This is especially useful in modern data-warehouses [Chaudhuri and Dayal 1997], where there is a trend to de-normalise the relations in order to have a much quicker access to relevant information. This is possible in these systems because data-warehouses have always derived and historical information which is used to obtain statistics, knowledge, trends that would be very difficult to see in the normalised databases.

As a result, there are many important applications and open questions for non-predictive uses of data mining in order to improve the data quality of an information system. There would be more tasks that an information system should do in the future. In Wagner's words:

"*The evolution of information system concepts can be roughly described by the sequence of hierarchical and network databases, relational databases, object-oriented, deductive and active databases, their resp. Enhancements by special-purpose features, e.g. for temporal, spatial, or uncertain, resp. Fuzzy information, and their 'globalization' for distributed, mobile and cooperative information processing. This ongoing evolution will lead to knowledge systems capable of processing various kinds of higher-level information, such as uncertain, disjunctive, and negative information, and in addition various kinds of knowledge, such as (deductive and active) rule knowledge, and 'social' knowledge on how to cooperate with global networks*" [Wagner 1998].

In this context, the consistent conjunction of different reasoning systems and the evaluation of the inference they perform must be essential in order to maintain the quality and integrity of the information and knowledge which will be maintained by this knowledge systems.

9.3 Software Topologies and Reinforcement

The analogy between programs and scientific theories outlined in [Fetzer 1991], and the modern view of software as an experimental science [Basili et al. 1986] [Basili 1993] has left behind the previous unsuccessful analogue between software entities such as specification, program and verification, and mathematical entities such as problem, theorem and proof.

Philosophy of science provides a much more enlightening paradigm for software construction, by explicitly recognising that software engineering is an experimental science but also that the development of an actual software system requires more inductive techniques [Partridge 1997] than deductive ones.

More concretely, machine learning (ML) is a more precise and practical framework for this new paradigm for software engineering. A software system is then regarded as a learning system. More concretely, traditional software systems are viewed as eager learning systems, where the system is an intensional and operative expression of the requirements, which behaves correctly in a certain environment. By using the ML terminology, requirements can be identified with the training data, the software system is just a working hypothesis and correction is more properly viewed as predictive accuracy.

The new wave of software paradigms, under very different banners such as intelligent software [Maes 1995], smart software and software agents [Genesereth and Katchpel 1994] [Nwana 1996], interactive software [Wegner 1996] or adaptive software [Lieberherr 1996] rounds off the analogy with ML, because they can be seen as more reactive or interactive learning systems, and many results and techniques, especially from lazy methods [Aha 1997], can be applied to them from the sub-fields of query learning, case-based reasoning, knowledge acquisition and revision, etc.

In our opinion, it is somehow short-sighted to try to develop intelligent, smart, interactive or adaptive software from scratch, without regarding more than thirty years of theoretical and experimental results from ML. Even in the case of 'traditional' software, it is worthy adapting some constructions, techniques, methods and theoretical results to better understand the development and nature of software systems.

In this way, this section 'reuses' for software development the theory introduced in chapter 5. Thus, the use of reinforcement as a tool for the study of the *validation* and *revision* of an inductive theory is translated into its use for the *validation* and *maintainability* issues of software systems.

9.3.1 Adapting the ML framework

First, the sample data for constructing a software system is composed of experience from other software systems, software repositories and requirements information. The experience and software repositories can be well formalised under the usual "background knowledge" in ML, which can be expressed in an intensional way and is supposed to be validated. However, the information that is usually gathered up for requirement elicitation is not composed mostly of extensional data such as input-output pairs or positive and negative examples. On the contrary, this information provided for the construction of a software system is composed of base cases, scenarios, interviews and a great amount of intensional knowledge.

Secondly, once the system is in operation or in the implantation stage, the validation cannot come exclusively from its use, it necessarily must be combined with the user's satisfaction about the product, by extending reinforcement with rewards and penalties.

Thirdly, to study maintainability, we must study two different things: the predictiveness of software, i.e., the expectancy of future modifications, which directly depends on the reinforcement which has been distributed upwards, and the consequences that each change may have in other components, which determines a downward flow. However, the first topology is dynamic while the second one is usually static.

9.3.2 Sample data. Training set

It is essential to discern what will constitute the examples or sample data from which reinforcement originates. In ML, these cases are usually facts, correspondences, pairs of input-outputs, etc. Classically, it is said that the behaviour of any system can be described in terms of input-output pairs, i.e. a function, expressed under a proper codification. However, theoretically, it has been proved that most complex systems cannot be identified by a finite data set of input-outputs [Gold 1967]. Part of the data must be given in an intensional way or there is a need of interaction. Finally, even if this important fact is ignored, in practice, the effort to convert a software system in terms of binary input-output is not sensible nowadays.

Contrarily, it is more practical to extend the notion of example. Apart from input-output pairs, we can identify many sorts of examples in the training phase (or requirements elicitation). They may be extensional, such as a use case, a scenario, a row in a correspondence table, a query and answer, etc., or an intensional concept. Even each sentence from the specification in natural language can be used as an example.

As we will see, any of these sorts of examples can be used for reckoning reinforcement. The only requirement is the definition of a proper notion of 'accordance', i.e., that a system is in accordance with some example. For instance, in the case of input-output pairs, the idea of accordance is extremely simple; if the system receives the input and returns the output as a result, the system is in accordance with the example. However, it may be more difficult to define 'accordance' for other kind of examples. In any case, it is important not to measure the *different* kinds of examples with the same value, because some of them are incomparable. Hence, the idea is to study reinforcement in a separate way for each sort and then try to put all that information together.

9.3.3 Granularity of propagation

One of the objectives of this study is the detection of which parts are being more reinforced than others, in order to know which are more predictive to future situations, or in other words, are less expectable to be modified in the future.

The first thing to do is to recognise the entities or components where we are going to centre the reinforcement measure. Although in the following the focus will lay on software components, the idea of component that will be used in the following will always be broader than that "component software" [Szyperski 1998] and easily extensible to any other system component, either physical (hardware) or logical (software).

The most minute choices, such as an instruction, show that reinforcement must not be distributed by the execution trace. For instance, some instruction can appear in a loop, being unfairly reinforced. On the other hand, the choice of large components provides wider views of how the software is being used. However, this higher level presents some other problems. For example, a module *A* can make use of another module *B* for just one functionality whereas it can use a module *C* for many functionalities. In some way, *C* should be more reinforced than *B*, but this granularity does not allow this appreciation. Although some of these problems are solved later on, once again, the idea is to measure reinforcement for the greater number of granularities as possible and then try to understand all the information jointly.

9.3.4 Validation data. User's accordance

The idea of accordance for the training set is clear. All the examples are usually labelled with positive and negative tags such as "the system should behave as the following scenario describes" or "the following situation should not happen".

However, when the system is in use, most of the situations are not like the training set, so they must be accompanied by the tag "this behaviour has been correct" or "this has been a system error". This feedback can be given by the environment, an external system or, more consciously, the user. In the case the behaviour is 'correct', the system is reinforced, as when a new example is predicted by a theory. On the other hand, when the behaviour is detected as 'incorrect', we have a *prediction error* of the system. A simpler assumption could be that things are going well until some feedback states the contrary. In this way, software is being reinforced as time passes by and no modification has been necessary. However, it has been shown in most ML paradigms that learning from positive data only is much harder, so the feedback from the user is also essential for software quality.

For semantic-based representational languages, there is usually a notion of proof or positive covering, i.e., a theory covers an example iff the example can be derived from the theory. This results in a Boolean notion of accordance. Examples of these languages are propositional languages, Horn theories, full logical theories, functional languages, some kind of grammars, and even higher-order languages. However, with our generalisation of example and with general software systems, one cannot assign a true or false label to the behaviour of a system with respect to some case. It is more accurate to talk about a degree of correctness, from absolute correctness to full malfunctioning.

Definition 9.120 We denote the accordance of a given example *e* with respect to a system *S* by $S \sqsupset^{\alpha} e$:

For convenience, $-1 \leq \alpha \leq 1$. In the following, we will refer to *e* as a positive example of *S* when α is 1 and a negative example if α is -1.

In the simplest case, when a system can be specified by a function $F \subset I \times O$, a positive example is just any element of $e \in F$. If we define $F^{neg} = \{ (i,o) \in I \times O$ such that $\exists e' = (i,o') \in F$ and $e' \neq e \}$ then we have that a negative example is just any element from F^{neg}. If *F* is complete, there are only three situations: positive hits, not covered cases and errors. Hence, $\alpha \in \{1, 0, -1\}$. Positive and negative samples are just subsets of *F* and F^{neg}. respectively. In concept learning or ILP, we have that $O= \{$ *false, true* $\}$ and it is said that the theory or system *S* covers the examples iff $F \subset S$. In more complex cases, the user or other client systems are responsible for providing the value of α for each example.

9.3.5 Software and reinforcement

In the case of software, reinforcement could be applied at almost any granularity. For instance, a *component* can be a rule, a procedure or a function, a class, a method, a variable, a module or any other higher division. A system will be just a set of components.

For a correct apportionment of credit, we need to discern which parts are justifiably responsible for the system to behave correctly for a particular example. In other words, if a component can be removed without affecting the functionality of the system with respect to some example, it is clear that this example should not be reinforced. The following definitions try to formalise and extend this idea:

Definition 9.121 A component r_i is said to be β-necessary with respect to *S* for an example *e* iff:

$$S \sqsupset^{\alpha} e \quad \text{and} \quad S - \{r_i\} \sqsupset^{\alpha'} e \quad \text{and} \quad \beta = \alpha - \alpha'$$

In general, if $\beta = 0$ we say that r_i is not necessary. On the contrary, if $\beta = 1$ we simply say that r_i is necessary. For instance, if we consider a system *S* composed of modules, we can have that the system covers an example *e* and without a module m_i, the system does not cover the example, so $S \sqsupset^1 e$, $S - \{m_i\} \sqsupset^0 e$ and $\beta=1$.

Definition 9.122 A system *S* is reduced for an example *e* iff:

$$S \sqsupset^{\alpha} e \quad \text{and} \quad \neg\exists\, r_i \in S \text{ such that } r_i \text{ is not necessary for } e$$

Definition 9.123 Reduced Set: $RS(e, S) = \{ S_i \subset S, S_i \text{ is reduced for } e \}$

This excludes subsystems with components that are not useful for increasing the accordance of the system with respect to the example.

However, it is not clear how to assign a credit to each component, as the following example shows:

Example 9.16

Suppose a system S with four components $\{ r_1, r_2, r_3, r_4 \}$ with $S = \{ r_1, r_2, r_3, r_4 \} \supset^1 e$, $S_1 = \{ r_1, r_2, r_3 \} \supset^1 e$, $S_2 = \{ r_1, r_2 \} \supset^{0.5} e$, $S_3 = \{ r_2, r_3 \} \supset^{0.9} e$ and for any other subset of S we have $\alpha = 0$.

We have that $RS(e, S) = \{ S_1, S_2, S_3 \}$.

We can particularise a different set for each component.

Definition 9.124 $RS_r(e,S) = \{ S_i : S_i \subset RS(e,S) \text{ and } r \in S_i \}$

And from here we compute the credit of each rule in the following way:

Definition 9.125 $credit\,(r, e) = \{ \Sigma_{S' \in RS_r(e, S)} \alpha \mid S' \supset^{\alpha} e \} \,/\, card(RS(e, S))$

For the previous example, Definition 9.125 gives these reasonable values: $credit(r_1, e) = 0.5$, $credit(r_2, e) = 0.8$, $credit(r_3, e) = 0.63$, and $credit(r_4, e) = 0$.

However, in the case of software, the influence of the different components is not additive. Very important modules, classes or functions do not perform anything valuable on their own, whereas interface components are much more visible to the user. Hence, we will only consider the 'saturated' subsystems:

Definition 9.126 A subsystem S' of S is saturated for an example e iff $\neg\exists\, r_i \in S$ such that:

$$S' \supset^{\alpha} e \text{ and } S' \cup \{r_i\} \supset^{\alpha'} e \text{ and } \alpha' - \alpha > 0$$

Theorem 9.30 A subsystem S' of S is saturated for an example e iff $\neg\exists\, r_i \in S$ such that $S'' = S' \cup \{r_i\}$ and r_i is β-necessary with respect to S'' for e with $\beta > 0$.

PROOF. If r_i is β-necessary with respect to S'' for an example e, we have by Definition 9.121 that $S'' \supset^{\alpha''} e$ and $S'' - \{r_i\} \supset^{\alpha'''} e$ and $\beta = \alpha'' - \alpha'''$. Since $S'' = S' \cup \{r_i\}$ then $\alpha'' = \alpha'$ and $\alpha = \alpha'''$. Since $\beta > 0$, we have that $\alpha'' - \alpha''' > 0$ and $\alpha' - \alpha > 0$. □

Definition 9.127 $SS(e,S) = \{ S_i \subset S,\ S_i$ is both reduced and saturated with respect to S for $e \}$.

We will refer to the elements of SS as *alternative* subsystems. For the previous example we have that $SS(e, S) = \{ S_1 \}$. Finally, we can define the set of alternative subsystems that contain r as,

Definition 9.128 $SS_r(e,S) = \{ S_i : S_i \subset SS(e,S) \text{ and } r \in S_i \}$.

One of the first results of these definitions is that a subsystem is a set of components. That is to say, it is independent of the trace, of how many times a component is used for a given example. This ultimately allows the following definitions, more similar to those of chapter 5.

Definition 9.129 The pure reinforcement $\rho\rho(r)$ of a component r from a system S with respect to some example e is defined as: $\rho\rho(r, e) = \Sigma_{S' \in SS_r(e, S)} \{\alpha : S' \supset^{\alpha} e \}$.

In other words, $\rho\rho(r)$ is computed as the sum of 'accordances' from the alternative subsystems for e where r is used. If there are more than one alternative subsystem for a given *e, all* of them are reckoned, but, as we have said, for the same subsystem, a component is computed only once.

The proportion of examples from a given evidence E where r is used, can be computed as

Definition 9.130 The probability of r being used for a given example from evidence $E= \{e_1, e_2, ..., e_n\}$ can be approximated by: $P_{use}(r) = \Sigma_{e \in E} \{\text{if } \rho\rho(r, e) > 0 \text{ then } 1 \text{ else } 0 \} / \text{card}(E)$.

For a set of examples, i.e., an evidence E, we extend Definition 9.129 in the obvious way:

Definition 9.131 The pure reinforcement $\rho\rho(r, E)$ of a component r from a system S with respect to some given evidence $E = \{e_1, e_2, ..., e_n\}$ is defined as: $\rho\rho(r, E) = \Sigma_{i=1..n}\, \rho\rho(r, e_i)$.

Definition 9.132 The (normalised) reinforcement is defined as: $\rho(r, E) = 1 - 2^{-\rho\rho(r, E)}$.

In the following, we will omit E when there is no possible confusion. Definition 9.132 is justified by the convenience of maintaining reinforcement between 0 and 1, as it was shown in chapter 5, while rendering easy the computation of reinforcement because each time a new example is covered by a system, the reinforcement of the components that have been used are incremented by $\rho'(r) = (\rho(r) + 1)/2$.

Definition 9.133 The mean reinforced ratio $m\rho(S)$ of a system S with m components is defined as:

$$m\rho(S) = \Sigma_{r \in S}\, \rho(r)/m$$

Finally, the validation *with respect to the evidence* is measured in the following way.

Definition 9.134 The course $\chi_S(e)$ of a given example e with respect to a system S is defined as:

$$\chi_S(e) = max_{S' \subset SS(e, S)} \{ \Pi_{r \in S'}\, \rho(r) \}$$

More constructively, $\chi_S(e)$ is computed as the product of all the reinforcements $\rho(r)$ of all the components r of S used in an alternative subsystem of e. If a component is used more than once, it is computed once. If f has more than one alternative subsystem, the greatest course is selected.

9.3.6 Validation propagation by reinforcement

As it was discussed in chapter 2, in the ML and philosophy of Science literature, there is a variety of evaluation criteria to select the most plausible hypothesis, i.e., the one with less prediction errors [Merhav and Feder 1998]. From this variety, the MDL (Minimum Description Length) principle [Rissanen 1978, 1996] [Barron et al. 1998] and the MLE (Maximum Likelihood Estimator) method have been thoroughly studied and inter-related [Kearns et al. 1999]. Associated with them are some popular validation methods such as cross-validation, which is also connected with different notions of hypothesis stability and reinforcement.

Intuitively, a theory that has been reinforced by the past evidence is more likely to behave properly for the future evidence. Differently from other evaluation criteria, the previous subsections have presented measures of reinforcement for each component, and not a unique value for the whole system. In order to estimate the predictive accuracy (or predictiveness) of a system, a single value is used instead. The most natural idea is the mean of all the courses of all the examples in the evidence:

Definition 9.135 The mean course $m\chi(S, E)$ of a system S with respect to an evidence E, with $n = \text{card}(E)$, is defined as: $m\chi(S, E) = \Sigma_{e \in E}\, \chi_S(e)/n$.

In chapter 5, the maximisation of $m\chi(S, E)$ and the MDL principle have been theoretically related. Logically, the shorter the theory the more probability that reinforcement would be more concentrated. In the same paper there are some examples that show that $m\chi(S, E)$ is a more compensated criterion than the MDL principle. Finally, it is possible to formalise the concept of intensionality by using reinforcement. A system is intensional if there are not examples covered by some component with low reinforcement value. Intensionality was shown to be closely related to cross-validation. In other words, systems with components added to the system to cover some exceptional examples, i.e. patches, have less stability. These extensional parts are not validated and it is highly unlikely that new examples will not be covered by these parts, so the system will probably be revised.

In the same way, we can translate these rationales to software systems. Hypothesis stability in ML is converted into system stability, i.e., the system endurance to requirement changes.

Predictiveness is thus distinguished as an actual software quality factor, inversely related to the number of modifications for evolving requirements in the same context. Reinforcement can be used as a very appropriate measure to estimate the probability of modification. More concretely, the probability of modification of a component can be directly specified from the reinforcement value.

Definition 9.136 The isolated probability of modification is: $P_{\text{mod}}(r) = 1 - \rho(r) = 2^{-\rho\rho(r)}$.

It is obvious that this defines a probability, since $0 \leq P_{mod}(r) \leq 1$. The term 'isolated' is motivated by the aim that Definition 9.136 should only measure the probability of modification that originates from each component r, not that other components could occasion the modification of r.

From here, it is straightforward to obtain the probability of modification of the whole system:

Theorem 9.31 If we consider independent the isolated modification of each rule of a system S, the isolated probability of modification of a system S is: $P_{mod}(S) = 1 - \Pi_{r \in S}\, \rho(r)$.

PROOF. Since the modification of each component is an independent fact, and S is defined as the set of rules, the probability of modification of one or more element of this set is obtained in the classical way: $P_{mod}(S) = 1 - \Pi_{r \in S} \left(\overline{P}_{mod}(r) \right) = 1 - \Pi_{r \in S} (1 - P_{mod}(r)) = 1 - \Pi_{r \in S}\, \rho(r)$. □

The absolute stability of a system can be defined as $\sigma(S) = 1 - P_{mod}(S) = \Pi_{r \in S}\, \rho(r)$, i.e., the probability that a system is not to be modified at all. This stability *of the whole theory* is a very strict requirement, so we will define another notion of stability later.

Up to here, we have been given probabilities of modification throughout the whole life cycle of the system. However, it would be interesting to measure the probability of modification just for the following k examples. Let us consider the probability of use of one component for one example $P_{use}(r)$, given by Definition 9.130. By a simple combinatorial analysis, the probability that one or more of the following k examples would use r is $1 - (\overline{P}_{use}(r))^k$. Then

Definition 9.137 The isolated probability of modification of component r before example k is:

$$P_{mod}(r, k) = P_{mod}(r) \cdot \{ 1 - (\overline{P}_{use}(r))^k \}$$

Theorem 9.32 Given a component r from system S and an evidence $E = \{e_1, e_2, ..., e_n\}$, such that $\exists e \in E\ \rho\rho(r, e) > 0$ (i.e., it is a useful component), then, as k grows, we have that $P_{mod}(r, k)$ approximates $P_{mod}(r)$.

PROOF. From Definition 9.137, we have that $\lim_{k\to\infty} \{ P_{mod}(r, k) \} = \lim_{k\to\infty} \{ P_{mod}(r) \cdot \{ 1 - (\overline{P}_{use}(r))^k \}\} = P_{mod}(r) \cdot \{ 1 - \lim_{k\to\infty} (\overline{P}_{use}(r))^k \}$. Since there exists an example e such that $\rho\rho(r, e) > 0$, then, by Definition 9.130, we have that $P_{use}(r) > 0$, or consequently $\overline{P}_{use}(r) < 1$. Hence, $\lim_{k\to\infty} (\overline{P}_{use}(r))^k = 0$, and this yields: $\lim_{k\to\infty} \{ P_{mod}(r, k) \} = P_{mod}(r) \cdot \{ 1 - 0 \} = P_{mod}(r)$. □

In the following, we will suppose there are no useless components, and we will use Definition 9.136 and Theorem 9.31 to work with long-term life cycles. However, Definition 9.137 would be useful to compute short-term modification probabilities and maintenance costs. It even can be modified to consider the last n' examples instead of the whole evidence. For instance, if a module has not been used in the last 4 months, it is not likely that a modification would affect this module.

9.3.7 Measurement in practice

Definition 9.136 and Theorem 9.31 provide a means to evaluate the predictiveness of a system or, in some way, how much validated it is. However, as we said, there are still some details to resolve in order to make these measurements applicable for software systems: (1) one cannot measure the different examples with the same value, and (2) reinforcement can be measured for different granularities of components.

Weighing the evidence

In ML, examples are usually regularised to the same significance. However, in software, it is difficult to balance some kinds of examples, such as an input-output pair with a scenario. In addition, some examples are used to describe exceptional behaviours, with low frequency of use whereas other examples are introduced to represent the main part of a system or frequent operation. The following extension is useful if one can assign a significance degree d_e. to the examples which conform the evidence E, and it is exactly the same as Definition 5.50.

Definition 9.138 The '*grounded*' *course* $\chi'(e)$ of a given example e with respect to a system is computed as the normal course $\chi(e)$ multiplied by the significance degree of e. More formally, $\chi'(e)=\chi(e)\cdot d_e$.

Another approach is the repetition of the examples that are more significant. This is exactly equivalent to the use of the previous definition, by repeating each example e in the evidence d_e. times.

Weighing components

The same approach is not valid with components. We cannot compare the reinforcement of a module with the reinforcement of a function. However, if one uses modules as components for obtaining a mean course $m\chi(S,E)$ and an absolute stability $1-P_{mod}(S)$, it is possible to make the same thing for another granularity, e.g. functions, to obtain a different mean course $m\chi'(S, E)$ and absolute stability $1-P'_{mod}(S)$. If one wants to combine both measurements, a major problem arises. In general, the grosser the granularity the greater the mean course and absolute stability. The reason is quite simple. For the same system, the finer the granularity the greater the number of components and reinforcement must be scattered. In the extreme case, if we consider only a component, the system itself, we have the maximum value for $m\chi(S, E) = \Sigma_{e\in E}\, \chi_S(e)/n = \Sigma_{e\in E}\, \rho(S)/n = \Sigma_{e\in E}\, (1 - 2^{-n})\, /n$, which converges quickly to 1 if $n = card(E)$ increases.

To equilibrate the matter there are two options: (1) the introduction of a factor directly related to the number of components, and (2) the introduction of a factor inversely related to the size of each component. The first one may propitiate the pseudo-repetition of components, i.e., components that are always needed in conjunction. Hence, we will choose this second option, which is also the same as Definition 5.48.

With size(r) we will denote the size of a component r, with the only restriction for size that for all r, size$(r) \geq 1$. We extend the definitions in the following way:

Definition 9.139 The extended pure reinforcement is defined as: $\rho\rho^*(r) = \rho\rho(r)$ / size(r).

Likewise we could define the extended normalised reinforcement $\rho^*(r)$ and the extended course $\chi^*(r)$.

Finally, with this modification, reinforcement can be associated with the idea of reusability. Inside a single system, a module or component is reinforced if it is used for many cases or examples. Moreover, the last modification favours granularity, which also eases reusability. At the level of different applications, and by considering the evidence as the set of all the examples for these different applications, a highly reinforced module is reused to cover many groups of examples.

9.3.8 Modification propagation

We have dealt about predictiveness, as the ability of a system to behave correctly for evolving requirements. This gives an isolated probability of modification $P_{mod}(S)$ whatever the part of the system. However, to estimate maintenance costs it is important to know the consequence of each change, i.e., how many components are to be modified and how difficult these modifications are.

The following figure shows the two main factors that affect maintenance: the probability of modification which is inversely related to the validation or predictiveness characteristic, and the modifiability of the components which are more likely to be revised.

However, in the literature of software modifiability, there is usually no detailed relationship between the probability of modification of each component and the modifiability of each component. In general, the relation is between the validation of the whole system and the modifiability of the whole system. Figure 9.3. shows the difference of accuracy between the classical approach and the detailed approach.

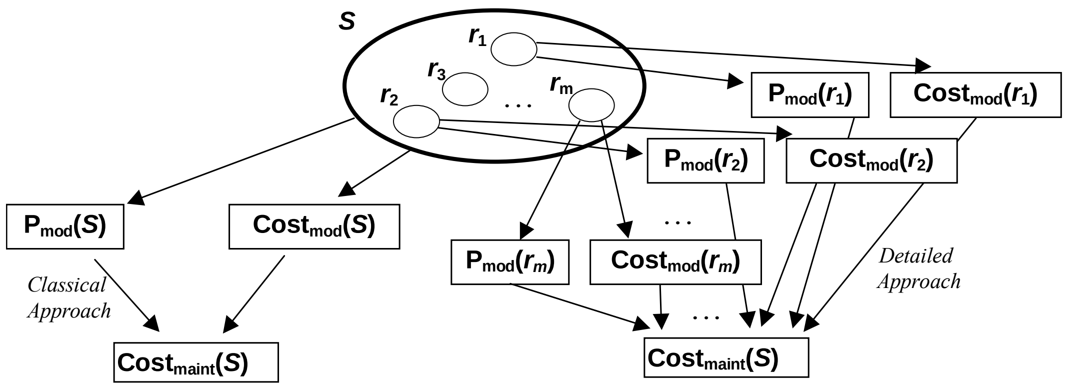


Fig 9.3. Two different ways to estimate the Maintenance Cost

In what follows, different particularised models for approximating this maintenance cost are presented, that we denote by $\text{Cost}_{\text{maint}}(S)$.

Model 0

The easiest (but less realistic) model for modifiability is the assumption that every component modification is independent to the rest of components. In this case, it is only necessary to know that each component has a modification cost, a real number that we denote by $\text{Cost}_{\text{mod}}(r)$:

Definition 9.140 The *isolated* cost of maintenance is defined as:

$$\text{Cost}^0_{\text{maint}}(S) = \Sigma_{r \in S}\, \text{P}_{\text{mod}}(r) \cdot \text{Cost}_{\text{mod}}(r)$$

Although more detailed than the classical $\text{P}_{\text{mod}}(S) \cdot \text{Cost}_{\text{mod}}(S)$, this last definition has been computed according to the isolated probability of modification.

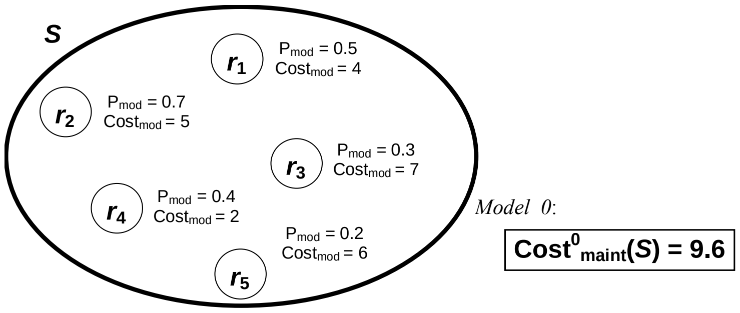

S
r1 Pmod = 0.5 Costmod = 4
r2 Pmod = 0.7 Costmod = 5
r3 Pmod = 0.3 Costmod = 7
r4 Pmod = 0.4 Costmod = 2
r5 Pmod = 0.2 Costmod = 6
Model 0:
Cost0maint(S) = 9.6

Fig 9.4. Example of estimation using Model 0

For instance, given the system illustrated in figure 9.4, the classical approach takes $\text{P}_{\text{mod}}(S) = 0.9496$ to obtain $\text{Cost}_{\text{maint}}(S)$. For instance, if it is estimated that each modification would affect 1.5 components on the average, we can compute $\text{Cost}_{\text{mod}}(S)=1.5\cdot\Sigma_{r\in S}\ \text{Cost}_{\text{mod}}(r)/\ \mathbf{card}(S)\ =1.5\cdot 24/5=\ 7.2$. Finally, $\text{Cost}_{\text{maint}}(S) =\text{P}_{\text{mod}}(S)\ \cdot\ \text{Cost}_{\text{mod}}(S)\ =0.9496\cdot 7.2=\ 6.84$, which is different from the one given by Definition 9.140.

In this example, both values, $\text{Cost}_{\text{maint}}(S)$ and $\text{Cost}^0_{\text{maint}}(S)$ are not too despair, but, in general, they can differ a great deal. Despite the fact that model 0 is more detailed that the classical one, it is still very simple because it ignores the relationships between components where modification propagation flows.

9.3.9 Modification dependences

Given a representation language, there are different notions of dependence. There are functional dependences, where execution (and semantics) propagates (usually bottom-up) and static dependences, where modification propagates (usually top-down).

It is difficult to establish exactly which are the modification dependences of a given system. It depends on the degree of encapsulation of the components, their coupling and other semantic or syntactical considerations. Moreover, these factors are highly reliant on the granularity of components. For instance, if a module or class is modified in its declaration, then it is easier to detect the modules or classes that are expected to be modified than if a single line of a program is modified.

Once these questions are solved for a particular system, the modification dependences can be formalised by the term "r depends on t_i" that we write $r \hookleftarrow t_i$. For all the dependences of a single component we will also use the following notation $r \hookleftarrow t_1, t_2, .. t_n$. This dependence relation does not need to be reflexive or transitive.

Definition 9.141 The direct ascendant set of a component r is defined as:

$$DAsc(r)=\{r' : r \hookleftarrow r'\}$$

Definition 9.142 The direct descendant set of a component r is:

$$DDes(r) = \{ r' : r' \hookleftarrow r \}$$

We define the relation $\hookleftarrow^*$ as the transitive and reflexive closure of the dependency relation $\hookleftarrow$. Formally,

Definition 9.143 For any two components r_a , r_b, we have that $r_a \hookleftarrow^* r_b$ holds iff $r_a = r_b$ or $r_a \hookleftarrow r_b$ or there exists another r_c such that $r_a \hookleftarrow^* r_c$ and $r_c \hookleftarrow^* r_b$.

Definition 9.144 The ascendant set of a component r is defined as:

$$Asc(r) = \{ r' : r \hookleftarrow^* r' \}.$$

Definition 9.145 The descendant set of a component r is:

$$Des(r) = \{ r': r' \hookleftarrow^* r \}.$$

These dependences are more or less difficult to establish depending on the granularity chosen for the examples. For instance, in a procedural language, suppose that a function f uses functions g and h in its definition, h uses function i in its definition, and function i uses g. The resulting components and dependences are: $f \hookleftarrow$

g, *f* ↲ *h*, *h* ↲ *i*, and *i* ↲ *g*. By the transitivity closure, ↲* extends ↲ to *h* ↲* *g*, *f* ↲* *i* , *f* ↲* *g* and all the reflexive relations.

In the same way, one can extend dependences to sets of functions, or modules. In this case, the 'uses' or 'includes' directives are a good overestimation to modification dependences. How much these dependences overestimate modification dependences relies on the kind of modification (in behaviour or declaration) and the encapsulation of each module.

Finally, to give a much more present and realistic view, in some modelling stages or languages, dependences are very heterogeneous, as the following simple object model illustrates:

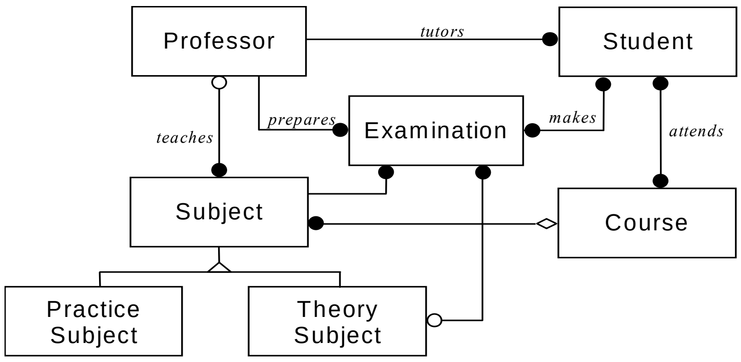


Fig 9.5. Example of heterogeneous dependences

If we identify classes with components, in many cases we could identify modelling relationships (associations, aggregations and inheritance) in one or both ways, according to external information to the model (or design model information). In any case, the dependences that can be extracted are barely representative of the modification dependences between classes. A better approximation can be made by studying the methods and other relationships between classes.

In short, it is possible to define ↲ for any granularity and any representational language, but the accuracy to which ↲ represents modification dependences is heavily contingent on this granularity and any other experience or information which may be used to refine the estimate.

Besides, not any relation ↲ can be used. There is an important property that this relation must hold, acyclicness, i.e., ↲* must be a partial order relation. This limitation is not very restrictive because, although *static* functional dependences are frequently cyclic and *static* modelling dependences are sometimes cyclic (like figure 9.3), the instantiated dependences of an effective program are acyclic.

This hierarchisation was advocated long ago by [Dijkstra 1968] and [Parnas 1972]: "We have a hierarchical structure if a certain relation may be defined between the modules or programs and that relation is a partial ordering. The relation we are concerned is "uses" or "depends upon"".

However, if the dependency relation is cyclic, with two components r_1 and r_2 such that r_1↲$^*r_2$ and r_2↲$^*r_1$, then a fictitious component r_f must be inserted to make r_1↲$^*r_f$ and r_2↲$^*r_f$ and the cycle is removed. Obviously, the costs and probabilities of modification should be readjusted among r_1, r_2, r_f and other components involved.

Model 1

Once relation ↲ is approximated, we can remake the effective probability of modification introduced in the previous section. We can define a new measure that weighs the isolated probability of modification and the scope of each modification (its propagation), assuming $P_{mod}(r)$ independent.

Definition 9.146 Given the acyclic relation ↲ for modification dependences, the related probability of modification P^*_{mod} of a single component is defined as:

$$P^*_{mod}(r) = 1 - \overline{P}_{mod}(r) \cdot \Pi\; a_{i \in Dasc(r)} \left(-\overline{P}^*_{mod}(a_i) \right)$$

where Π is defined to be 1 if it has no factors.

Finally, model 1 can be introduced accordingly:

Definition 9.147 $Cost^1_{maint}(S) = \Sigma_{r \in S}\, P^*_{mod}(r) \cdot Cost_{mod}(r)$

Let us extend the example of figure 9.4 with some dependences. The new model applied in fig. 9.6 shows that the cost of maintenance increases, due to the consideration of these modification propagations that were not taken into account in model 0.

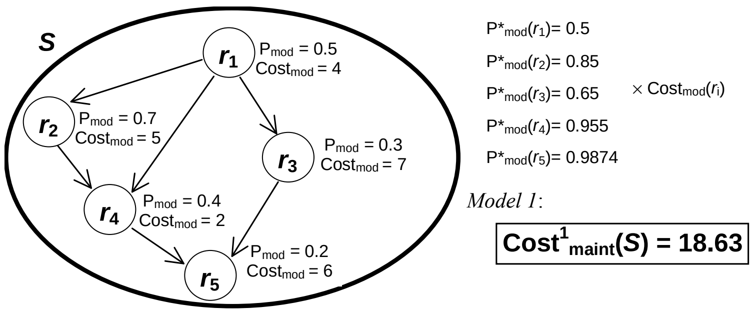


Fig 9.6. Example of Estimation using Model 1

However, this model presents some problems. It over-propagates modification, because it modifies the probabilities through all the paths that dependences draw. For instance, in Fig. 9.6, we can observe that $P^*_{mod}(r_4)$ takes into account $P^*_{mod}(r_1)$ twice: from the path $r_1 \rightarrow r_2 \rightarrow r_4$ and from the path $r_1 \rightarrow r_4$ directly. The same happens with $P^*_{mod}(r_5)$.

Model 1b

The correction must only consider each dependency once by using the set of ascendants instead of a recursive reckoning of the dependences.

Definition 9.148 Given the acyclic relation $\lrcorner$ for modification dependences, the corrected related probability of modification P^{*b}_{mod} of a single component is defined:

$$P^{*b}_{mod}(r) = 1 - \Pi_{a \in Asc(r)} (\overline{P}_{mod}(a))$$

and we redefine the cost of maintenance

Definition 9.149 $Cost^{1b}_{maint}(S) = \Sigma_{r \in S} P^{*b}_{mod}(r) \cdot Cost_{mod}(r)$.

The previous example is corrected to $P^{*b}(r_1)=0.5$, $P^{*b}(r_2)=0.85$, $P^{*b}(r_3)=0.65$, $P^{*b}(r_4)=0.91$, $P^{*b}(r_5)=0.9496$, which gives $Cost^{1b}_{maint}(S) = 18.32$.

Finally, we could use this model to define the detailed stability of a system.

Definition 9.150 The stability of a system S is defined as $\sigma(S) = 1 - \Pi_{r \in S} P^{*b}_{mod}(r)$.

Model 2

Although model 1b is useful to define stability, it does not proceed in an additive way with the modification costs. For instance, it is more intuitive to proceed bottom-up as follows: if we have a modification at component r, we have to *add* the cost of all the components that depend on it, as follows:

Definition 9.151 The *accumulate* cost of a component r is defined as:

$$AcCost_{mod}(r) = \Sigma_{a \in Desc(r)} Cost_{mod}(a)$$

And once again, the cost of maintenance of a system S could be defined as:

Definition 9.152 $Cost^{2}_{maint}(S) = \Sigma_{r \in S} P_{mod}(r) \cdot AcCost_{mod}(r)$

And the results are quite different in this case: $AcCost_{mod}(r_1)=24$, $AcCost_{mod}(r_2)=13$, $AcCost_{mod}(r_3)=13$, $AcCost_{mod}(r_4)=8$, $AcCost_{mod}(r_5)=6$, that gives $Cost^{2}_{maint}(S) = 29.4$.

9.3.10 System topologies and maintenance cost

A validation (or predictiveness) measure for a software system can be obtained by using reinforcement propagation. This is inversely related to the modification probability. We have also introduced a dependency relation where modification propagates. As it has been said, the major problem of this dependency relation is that it is difficult to obtain. In general, when a component r is modified, the set of components that are to be modified depends mostly on the utilisation rate from the other components. This use rate is precisely what determines reinforcement. This insight motivates the following assignment:

> **Assumption 9.1.** The modification dependency graph, determined by relation ↵, usually top-down, matches *reversely* with the validation reinforcement graph, usually bottom-up.

Although this assumption is controvertible, it has many advantages as a working approximation,

- modification dependences can be determined by the course of reinforcement.
- conversely, the course of reinforcement, which is extremely variable and uncertain for static models, can be approximated by the graph of modification dependences.

On the other hand, this approximation has also some inconveniences. Not all granularities admit this matching. Moreover if one tries to mix up different granularity in both ways, for instance, using a procedural granularity to assign reinforcement and using an object-oriented granularity for modification dependences, the results may be useless.

The final justification of this assignment is that it allows a theoretical study of the trade-off between validation (or predictiveness) and modifiability. More concretely, in order to obtain a validated (reinforced) system, a component should be used in the greater number of cases (and other modules) as possible. However, this would compromise modifiability, because any simple modification would propagate to an enormous number of other components.

There is a long debate about the convenience of high fan-in and slow fan-out and vice-versa. The slogan of reusability is keep fan-out high and keep fan-in low. The slogan of modification in inheritance is to avoid a great number of children. This discussion is somehow paradoxical because for every dependency that goes out from a component, there is another component where it arrives to. In other words, mean fan-in is always equal to mean fan-out. So, it is more sensible to talk about high or low connectivity or, more meaningfully, to talk about topologies.

Intuitively, a hierarchical arrangement of dependences eases the modification of the leaves situated at the bottom without the modification of the leaves at the top. This was recognised by [Parnas 1972] long ago: "The partial ordering gives us two additional benefits. First, parts of the system are benefited (simplified) because they use the service of [upper] levels. Second, we are able to cut off the [lower] levels and still have a usable and useful product. [...]. The existence of the hierarchical structure assures us that we can "prune" off the [lower] levels of the tree and start a new [reversed] tree on the old trunk. If we had designed a system in with the "[high] level" modules made some use of the "[low] level" modules, we would not have the hierarchy, we would find it much harder to remove portions of the systems.". However, one can wonder if the shape of this graph should be tree-like or root-like, the latter implicitly advocated by Parnas. This motivates a more detailed analysis of configurations of a given program P.

Definition 9.153 The Bottom or Minimal Set of a program P, denoted Bot_P, is composed of every component $b \in P$ such that $\neg\exists c \in P$, $c \neq b$, such that $c \lrcorner b$. In other words, $Bot_P = \{ b : Des(b) = \{ b \}\}$.

Definition 9.154 The Top or Maximal Set of a program P, denoted Top_P, is composed of every component $t \in P$ such that $\neg\exists c \in P$, $c \neq t$, such that $t \lrcorner c$. In other words, $Top_P = \{ t : Asc(t) = \{ t \}\}$.

According to the previous characteristics, we are going to study five different *topologies*:

- Topology 1: "Horizontal": No dependences at all. Obviously, $P = Bot_P = Top_P$. We will consider the following extreme cases:

  a) Compensated. $\forall i\ \rho\rho(c_i) = n / m$.

  b) With exceptions. $\exists j\ \rho\rho(c_j) = n - m + 1$ and the rest are exceptions (or patches) $\rho\rho(c_i) = 1$, $i \neq j$.

- Topology 2: "Vertical": The dependency relation $\lrcorner$ obeys a full order relation $<$, $\forall\ c_1, c_2 \in P$, $c_1 \neq c_2$, then $\neg(c_2 < c_1) \leftrightarrow c_1 < c_2$. There is a unique top component $t \in Top_P$, i.e., card(Top_P) = 1. There is a unique bottom component $b \in Bot_P$, i.e., card(Bot_P) = 1. From here, the following properties hold, $\forall\ c \in P$, $c \neq t$, then $c < t$ and $\forall\ c \in P$, $c \neq b$, then $b < c$.

- Topology 3: Lattice: The dependency relation $\lrcorner$ obeys a partial order relation $<$. There is a unique top component t such that $\forall\ c \in P$, $c \neq t$, then $c < t$ and a unique bottom component b such that $\forall\ c \in P$, $c \neq b$, then $b < c$. We will consider three extreme cases:

  a) A unary lattice which corresponds to topology 2.

  b) Wide lattice with depth = 3, where the middle level has m–2 components.

  c) Binary lattice. We assume $m = 2^k - 1 + 2^{k-1} - 1 = 3 \cdot (2^{k-1}) - 2$, being k a natural number.

- Topology 4: Tree: The dependency relation obeys a partial order relation $<$ with no unique top element ($card$(Top$_P$) ≥ 1) and $\forall\ a,b : a \notin$ Des(b) $\wedge\ b \notin$ Des(a) $\rightarrow$ Asc(a) $\cap$ Asc(b) $= \emptyset$. There is a unique bottom component b such that $\forall\ c \in P$, $c \neq b$, then $b < c$. We will consider three prototypical cases:

  a) One vertical branch (i.e. $card$(Top$_P$) = 1). Equivalent to topology 2.

  b) Wide tree with depth = 2, where the top level has m–1 components.

  c) Binary tree. We assume $m = 2^k - 1$, being k a natural number.

- Topology 5: Root (inverse tree): The dependency relation obeys a partial order relation < with no unique bottom component ($card(Bot_P) \geq 1$) and $\forall\ a,b : a \notin \text{Asc}(b) \wedge b \notin \text{Asc}(a) \rightarrow \text{Des}(a) \cap \text{Des}(b) = \emptyset$. There is a top component t such that $\forall\ c \in P, c \neq t$, then $c < t$. We will consider three prototypical cases:

    a) One vertical branch (i.e. $card(Bot_P) = 1$). Equivalent to topology 2.

    b) Wide root with depth = 2, where the bottom level has m–1 components..

    c) Inverse binary tree. We assume $m = 2^k - 1$, being k a natural number.

  Cases b) and c) will be studied in two ways: compensated and with exceptions.

We will assume that all components have the same size and that all examples have the same significance. From here,

Theorem 9.33 Given n examples $e_1, e_2, ..., e_n$, a program of m components arranged under topologies 2, 3 or 4 has the following properties:

- For every component c from P, the pure reinforcement $\rho\rho(c)$ is n, and the normalised reinforcement $\rho(c) = 1 - 2^{-n}$.
- For every example e_i the course $\chi(e_i) = (1-2^{-n})^m$. Hence, the mean course is $m\chi(E) = (1-2^{-n})^m$.
- For every component c, the isolated probability of modification $P_{\text{mod}}(c)$ is 2^{-n} and $P_{\text{mod}}(P) = 1 - (1-2^{-n})^m = 1 - m\chi(E)$.

PROOF. Topologies 2, 3 and 4 have a unique bottom b, and obviously, all the examples are covered by this bottom component b. Hence, $\rho\rho(b) = n$, and $\rho(b) = 1$. Since we consider static dependences, and all the components are required by b, because $\forall c \in P, c \neq b$, then $b \dashv c$, they all have the same pure reinforcement $\rho\rho(c) = n$. The rest of properties follow from here by applying previous definitions. □

Topologies 1 and 5 may have an overlap in the coverings of the bottom set, i.e., $\forall\ b_i \in Bot_P, \Sigma\rho\rho(b_i) > n$. This kind of redundancy is usually eliminated in software systems (except when a voting method is used to increase reliability), so we will consider $\forall\ b_i \in Bot_P, \Sigma\ \rho\rho(b_i) = n$.

As a result, given n examples, it is shown in the appendix of this chapter that a program P of m components with $n \gg m$ such that $\forall r \in P$, $\text{Cost}_{\text{mod}}(r) = U_{cost}$, it brings forward the following maintenance costs (under model 2):

Topology	*Maintenance cost*
1a) Horizontal compensated	$O(2^{-n/m} \cdot m)$
1b) Horizontal with Exceptions	$O(m)$
2) Vertical	$O(2^{-n} \cdot m^2)$
3b) Lattice with depth = 3	$O(2^{-n} \cdot m)$
3c) Binary Lattice	$O(2^{-n} \cdot m \cdot \log_2 m)$
4b) Tree with depth = 2	$O(2^{-n} \cdot m)$
4c) Binary Tree	$O(2^{-n} \cdot m \cdot \log_2 m)$
5b-i) Inverse Tree with depth = 2 and compensated	$O(2^{-n/m} \cdot m)$
5c-i) Binary Inverse Tree and compensated	$O(2^{-n/m} \cdot m \cdot \log_2 m)$
5b-ii) Inverse Tree with depth = 2 with exceptions	$O(m)$
5c-ii) Binary Inverse Tree with exceptions	$O(m \cdot \log_2 m)$

Fig 9.7. Results of Maintenance Costs for Different Topologies

Having in mind the assumptions and approximations that have led us to use $\text{Cost}^2_{\text{maint}}(S)$ for approximating the maintainability of a software system, we can extract some conclusions:

First of all, when the software system has exceptions or patches, which are used for few examples (topologies 1b, 5b-ii and 5c-ii), they have not been validated and the maintenance cost depends almost exclusively on them, in the way that the cost is asymptotically independent from *n*, the factor that reduces the cost.

From all the rest of compensated topologies, where reinforcement is distributed uniformly, the results are not so despair. However there is a great asymptotic difference between a wide tree or a lattice and a binary inverse tree. This remarks that topologies should be confluent or 'conciliated' at the bottom, much more like a tree than like a root. In other words, components at the bottom should behave in a broad way and not in a specialised way, something that may be interpreted very differently depending on what one could think of a component. Finally, other more intuitive consequence is that wide topologies are better than thin ones, because of modification propagation.

The strongest result derivable from figure 9.7 is that compression, i.e. increasing *n* over *m*, is an excellent way to reduce maintenance cost. In relation to the same sample, simple systems are more reinforced because the ratio of examples by piece of software is greater, so validation is higher. In the other way, modification is much easier. Although this is well known since long ago, recently, however, there have been claims about considering software engineering as compression [Wolff 1994], supported by the idea of learning as compression. However, a very compressed model can be spoilt by some patches, something that it is plainly seen in topologies 1b, 5b-ii and 5c-ii.

Finally, as we have said, more things can be inferred from this study if the components are particularised to real objects: classes, functions, modules, etc. For instance, if the components are classes, one can identify these results with four OO software quality metrics such as "lack of cohesion of methods" (increase granularity when possible) , "coupling between objects" (decrease granularity when possible), "depth of inheritance tree" and "number of children". Other interpretations are at first sight less intuitive or even contradictory. Inheritance, which is widely accepted, determines a topology of type 5c-i). However, the dependency relation is not only conditioned by inheritance relationship but also by associations, aggregations, etc. Moreover, multiple inheritance helps to change the topology to types 3c) or 4c). Ultimately, polymorphism represents the previous idea of confluence or avoidance of specialisation at the bottom. In some way, polymorphism tries to 'pump up' reinforcement.

9.4 Other Applications

Here I include some other areas where some other concepts of this work may be useful, although a deeper examinations would be necessary to know up to which degree and success.

9.4.1 Meaning and Language Understanding

In chapter 6 we dealt about some questions of the learnability of natural language that were related with the notion of intensionality. In the same way we pointed out the close relationship between intention and intension, although they are different terms. However, there are many other traits of the act of comprehending that are related to other notions seen in chapters 4 and 7.

For instance, the distinction between explicit and implicit, as seen in chapters 3 and 4, is essential for the communication between two or more individuals. in the case of anaphora or ellipsis, natural language usually uses ellipsis of what is clearly implicit. An example of ellipsis is "Mary eats and savours an apple", known as the "shallow structure" which is made explicit into its "deep structure": "Mary eats an apple and Mary savours that apple". An example of anaphora is "It is the kind of apples that she likes" which has as deep structure "The apple is the kind of apples that Mary likes". It is impossible to devise rules to account for all cases, as some natural language processing systems have implemented with partial success. It is necessary that the systems would have explanatory reasoning and ability to understand to address these problems in general, to face ambiguity, etc. In all these cases, apart from the intention of the speaker, the explicitness and plausibility of the different interpretations should be evaluated.

Another question where the measures introduced in this work may be useful is the measurement of understandability. This problem has been usually addressed by linguistics and psychology [Hörmann 1981][Just and Carpenter 1987] [Rayner and Pollatsek 1989], but never formalised. Some representative results can be found in [Sommer 1995b].

These results have been translated into formal languages like logic theories but, since understandability (or comprehensibility) has never been defined in a formal way, the translation has never been made in the other way. The notion of comprehensibility introduced in chapter 6 has been applied to symbolic problems like those that appear in IQ tests. However, the same measures could be applied to logical programs and then generate equivalent (although maybe inelegant) sentences from them, in order to measure understandability in a verbal context.

Nonetheless, the measures could also be used in Natural Language Processing (NLP). The combination of existing techniques in NLP, mainly based on an ad-hoc coding of grammars and rules through Lisp or Prolog programs should be accompanied by inductive methods, and, in this case, inductive logic programming seems the most appropriate one, in what has been called the triple *L*, namely, "*language, learning and logic*" (LLL), as it has been advocated by [Muggleton 1998].

Natural language learning based on statistical approaches (e.g. n-gram language modelling) has been successful, but it is well known that such linguistically impoverished approaches have severe limitations. In contrast, the flexibility and expressivity of logical representations make them highly suitable for natural language analysis. Consequently there is a growing interest in applying Inductive Logic Programming techniques to linguistic learning problems.

From the NLP point of view the promise of ILP is that it will be able to steer a mid-course between the two alternatives of large scale, but shallow levels of analysis, and small scale, but deep and precise analyses, and produce a better ratio between breadth of coverage and depth of analysis. Many measures introduced by this thesis could be useful in this endeavour.

However, the adaptation of the different measures seen in chapter 7 for logic programs (especially for ILP) to address different interpretations of a given sentence and other problems of NLP could surely be the topic of another thesis.

9.4.2 Agents Communication

There has been a recent interest in defining standard languages for the communication between agents in a virtual environment. This can be a solution (or a necessity) for *compatible* agents, but this finally turns to be a restriction. Any language extends, changes and particularises as long as one knows the knowledge and intelligence of the other peer, which progressively becomes more ambiguous and less understandable for other persons.

The communication between two intelligent beings that do not share a single word of their languages has always been a relevant question in philosophy of language. The evidence is that they finally find a compromise (a dominant, mixed or invented language) to communicate each other. The question is harder in the case of virtual world or textual worlds, mainly because there are few external references and analogies, such as the person who can point a tree to another person and say "árbol" where the other sees an "arbre".

These contacts will be precisely given by communication between agents in different multiagent systems, like any computer network and, especially, the Internet. Imagine the following situation. An agent establishes communication with other agent, both being stranger to each other. In the case that each agent could essay with different languages to find one in common, this would be the best solution, but, in general, if both say that they speak English, there are still different levels up to they can speak English. Even one or two of them can be humans with few knowledge about English and the communication has to take this into account. This is a classical law of communication: both peers must synchronise their code in order to communicate effectively. As long as the agents are more intelligent and the languages richer, this is more difficult because *exact* protocols are difficult or impossible to implement. It is necessary that both peers use their *common sense* in order to adapt to the peer's limitations.

It is important to note that such a role has been assumed by humans and computers since the beginning of computer science and, nowadays, it is a crucial aspect of computer-human interaction. Currently, human beings degrade their language to be able to communicate with computers, because, nowadays, computers do not understand analogies, or jokes, and they do not solve most of the ambiguities of human communication and, consequently, the expressivity and flexibility of the language which is finally used must be reduced to a lower level.

It is then necessary to evaluate these abilities in order to improve the communication between peers that do not share the same language and the same level of intelligence and knowledge. Different protocols and tests should be devised in the future for this task, more or less akin to that of chapter 8.

9.5 Summary

Very different applications have been presented, each of them more or less related to one or more of the previous chapters. As expected, prospective applications and combinations raise more questions than they settle, but this is precisely what is intended by a prospect, to open new fields.

Section 2 discusses optimal representation of databases (understood as extremely lazy knowledge systems) for optimising accesses. Soon it is realised that this question has been addressed at the physical level and it is independent from the degree of intensionality of the model or logical schema of the database. However, it is also recognised that intensional properties are useful in this level, due to different reasons. Apart from the fact that data quality and data cleansing can affect positively to a data-mining process, it is realised that the other way it is also very significant to detect redundancies, inconsistencies and other properties which can help to improve the data quality of an information system. This will be more important for future information systems that will adopt more reasoning power, both inductive and deductive, in what is beginning to be called *knowledge systems*.

Section 3 applies reinforcement to study maintainability issues for programs (seen as eager knowledge systems). From the results, we can highlight that a great number of characteristics of software can be theoretically studied using ML analogies and techniques. In our case, reinforcement is used to obtain a probability of modification, where many other measures are derived from, such as system stability and maintainability measures for different topologies.

Section 4 presents other applications, mainly related with language and communication. Fixed languages have limited expressiveness and it would be rather surprising that a standard language could be used by most of the systems that may interrelate in a real world or in a virtual world. Different languages and *changing* languages must be used to communicate with different agents with divergent *intelligence* and *knowledge*, and the communication must adapt to these situations. The difference between explicit and implicit, formalised and clarified in this work could have fruitful applications in the area of natural language processing. The formal notion of comprehensibility introduced in chapter 6 has also been highlighted as an important trait to be exploited for language understanding.

This latter issue, comprehensibility, may also be applicable to knowledge systems, because understandability is one of the factors that affect the acceptance and popularisation of these systems, since many of them are completely cryptic for the user [Kodratoff 1994].

Databases and software systems are both extremes of knowledge systems. The classical knowledge-based systems, the natural language processing systems and the new trend of information agents or software agents are tracing an increasingly more diverse spectrum of use of inductive, deductive, analogical and abductive reasoning methods. Moreover, eager and lazy techniques are also beginning to be combined. In this panorama, the applications of the measures and other different notions presented in this work seem even more promising in the near future.

9.6 Appendix

This appendix includes the results of maintainability for the different topologies presented in section 3.

Topology 1. card(Bot_P) = card(P). So $\Sigma_{i=1..m}\ \rho\rho(c_i) = n$.

- For the case a) we have that $\rho(c) = 1 - 2^{-n/m}$ and for every example e_i the course $\chi(e_i) = (1-2^{-n/m})$. The isolated and related probabilities of modification are the same, exactly, $P_{\text{mod}}(c) = P{*}^{b}_{\text{mod}}(c) = 2^{-n/m}$. From here, $\text{P}_{\text{mod}}(P) = 1 - (1 - 2^{-n/m})^m$.
  $\forall i$ $\text{AcCost}_{\text{mod}}(c_i) = \Sigma_{a \in Des(r)} \text{Cost}_{\text{mod}}(a) = \text{Cost}_{\text{mod}}(a) = \text{U}_{\text{mod}}$ and $\text{Cost}^2_{\text{maint}}(P) = \Sigma_{r \in S} \text{P}_{\text{mod}}(r) \cdot \text{AcCost}_{\text{mod}}(r) = m \cdot 2^{-n/m} \cdot \text{U}_{\text{mod}} \in \mathbf{O}(\boldsymbol{m \cdot 2^{-n/m}})$.
- For the case b) we have that $\rho(c_j) = 1 - 2^{-(n-m+1)}$ and $\rho(c_i) = 0.5$ iff $i \neq j$. We have the course $\chi(e_j) = (1-2^{-(n-m+1)})$ and $\chi(e_i) = 0.5$ iff $i \neq j$. The mean course is $[(m-1)/2 + (n-m+1) \cdot (1-2^{-(n-m+1)})\ ] / n = \ [(1-m)/2 + n + (m-n-1) \cdot (2^{-(n-m+1)})\ ] / n$. If $n >> m$ this is approximately $(1-2^{-n})$. The isolated probability of modification is, $P_{\text{mod}}(c_j) = 2^{-(n-m+1)}$ and $P_{\text{mod}}(c_i) = 0.5$ iff $i \neq j$. From here, $\text{P}_{\text{mod}}(P) = 1 - (1 - 2^{-(n-m+1)})^{n-m+1} \cdot (1/2)^{m-1} = 1 - (1 - 2^{-(n-m+1)})^{n-m+1} \cdot 2^{1-m}$. Again, since $n >> m$ and n is great then $\text{P}_{\text{mod}}(P) \cong 1 - 2^{1-m}$.
  $\forall i$ $\text{AcCost}_{\text{mod}}(c_i) = \Sigma_{a \in Desc(r)} \text{Cost}_{\text{mod}}(a) = \text{Cost}_{\text{mod}}(a) = \text{U}_{\text{mod}}$ and $\text{Cost}^2_{\text{maint}}(P) = \Sigma_{r \in S} \text{P}_{\text{mod}}(r) \cdot \text{AcCost}_{\text{mod}}(r) = ((n-m+1) \cdot 2^{-(n-m+1)} + (m-1) \cdot 1/2\ ) \cdot \text{U}_{\text{mod}}$. Since the first term $(n-m+1) \cdot 2^{-(n-m+1)}$ is always ≤ 1 if $n > m$, then $\text{Cost}^2_{\text{maint}}(P) \in \mathbf{O}(\boldsymbol{m})$

Topology 2. From Theorem 9.33, $\rho\rho(c) = n$, and $\rho(c) = 1 - 2^{-n}$, the course $\chi(\ e_i) = (1-2^{-n})^m$ for all e_i. For all c_i, the isolated probability of modification $P_{\text{mod}}(c_i)$ is 2^{-n} and $P_{\text{mod}}(P) = 1 - (1 - 2^{-n})^m$.

- Without loss of generality in this topology, $c_1 = b$ and $c_m = t$, with $c_i < c_{i+1}$ $\forall i$ $1 \leq i < m$.

$\forall i$ $\text{AcCost}_{\text{mod}}(c_i) = \Sigma_{a \in Des(c_i)} \text{Cost}_{\text{mod}}(a) = i \cdot \text{U}_{\text{mod}}$ and $\text{Cost}^2_{\text{maint}}(P) = \Sigma_{r \in S} \text{P}_{\text{mod}}(r) \cdot \text{AcCost}_{\text{mod}}(r) = 2^{-n} \cdot \Sigma_{i=1..m}\ i \cdot \text{U}_{\text{mod}} = 2^{-n} \cdot m \cdot (m+1)/2 \cdot \text{U}_{\text{mod}} \in \mathbf{O}(2^{-n} \cdot m^2)$

Topology 3. From Theorem 9.33, $\rho\rho(c) = n$, and $\rho(c) = 1 - 2^{-n}$, the course $\chi(\ e_i) = (1-2^{-n})^m$ for all e_i. For all c_i, the isolated probability of modification $P_{\text{mod}}(c_i)$ is 2^{-n} and $P_{\text{mod}}(\text{P}) = 1 - (1 - 2^{-n})^m$.

- For the case b) $\text{AcCost}_{\text{mod}}(b) = \text{U}_{\text{mod}}$ and $\text{AcCost}_{\text{mod}}(t) = m{\cdot}\text{U}_{\text{mod}}$ and $\text{AcCost}_{\text{mod}}(r) = 2{\cdot}\text{U}_{\text{mod}}$ iff $r \neq t$ and $r \neq b$. So, $\text{Cost}^2_{\text{maint}}(P) = \Sigma_{r \in S} \text{P}_{\text{mod}}(r) \cdot \text{AcCost}_{\text{mod}}(r) = 2^{-n} \cdot (1 + m + (m-2){\cdot}2\ ) {\cdot}\text{U}_{\text{mod}} \cong 2^{-n} \cdot 3m \cdot \text{U}_{\text{mod}} \in \mathbf{O}(\boldsymbol{2^{-n} \cdot m})$
- For the case c) it is obvious that $\text{AcCost}_{\text{mod}}(t) = m{\cdot}\text{U}_{\text{mod}}$. Since $m = 2^k - 1 + 2^{k-1} - 1$, we have $2k-1$ levels: k levels with increasing width and $k-1$ levels with decreasing width. For the first k levels, j-numbered top-down from 1 to k, we have 2^{j-1} components on each level, and $\text{AcCost}_{\text{mod}}(r^j) = (2^{k-j+1} - 1 + 2^{k-1-j+1} - 1 + (j-1)) {\cdot}\text{U}_{\text{mod}} = (3 \cdot 2^{k-1-j+1} - 3 + j) {\cdot}\text{U}_{\text{mod}}$
  For the next $k-1$ levels, i-numbered top-down from $k-1$ to 1, we have 2^{i-1} components on each level, and $\text{AcCost}_{\text{mod}}(r^i) = i {\cdot}\text{U}_{\text{mod}}$

Finally, $\text{Cost}^2_{\text{maint}}(P) = \Sigma_{r \in S}\, \text{P}_{\text{mod}}(r) \cdot \text{AcCost}_{\text{mod}}(r) = 2^{-n} \cdot (\Sigma_{j=1..k}\, 2^{j-1} \cdot [3 \cdot 2^{k-1-j+1} - 3 + j] + \Sigma_{i=1..k-1}\, 2^{i-1} \cdot i\,) \cdot \text{U}_{\text{mod}} = 2^{-n} \cdot (\Sigma_{j=1..k}\, [3 \cdot 2^{k-1} - 3 \cdot 2^{j-1} + j \cdot 2^{j-1})] + \Sigma_{i=1..k-1}\, 2^{i-1} \cdot i\,) \cdot \text{U}_{\text{mod}} = 2^{-n} \cdot ([k \cdot 3 \cdot 2^{k-1} - 3 \cdot 2^k + 2 + \Sigma_{i=1..k-1} j \cdot 2^{j-1})] + \Sigma_{i=1..k-1}\, 2^{i-1} \cdot i\,) \cdot \text{U}_{\text{mod}}$

By using the approximation $\Sigma_{l=1..p}\, 2^{l-1} \cdot l \cong p \cdot 2^p$., we have: $\text{Cost}^2_{\text{maint}}(P) \cong 2^{-n} \cdot ([k \cdot 3 \cdot 2^{k-1} - 6 \cdot 2^{k-1} + 2 + (k-1) \cdot 2^{k-1})] + (k-1) \cdot 2^{k-1}\,) \cdot \text{U}_{\text{mod}} = 2^{-n} \cdot ((k \cdot 3 + 2k - 2 - 6) \cdot 2^{k-1} + 2) \cdot \text{U}_{\text{mod}} = 2^{-n} \cdot ((5k - 8) \cdot 2^{k-1} + 2) \cdot \text{U}_{\text{mod}}$

Since $m = 2^k - 1 + 2^{k-1} - 1$, then $2^{k-1} = (m+2)/3$ and $k = \log_2 [(m+2)/3 + 1]$ then, $\text{Cost}^2_{\text{maint}}(P) \cong 2^{-n} \cdot ((5 \log_2 [(m+2)/3 + 1] - 8) \cdot (m+2)/3 + 2) \cdot \text{U}_{\text{mod}} \cong 2^{-n} \cdot 5/3 \cdot m \cdot \log_2 (m/3) \cdot \text{U}_{\text{mod}} \in \mathbf{O(2^{-n} \cdot m \cdot \log_2 m)}$

Topology 4. From Theorem 9.33, $\rho\rho(c) = n$, and $\rho(c) = 1 - 2^{-n}$, the course $\chi(\,e_i) = (1-2^{-n})^m$ for all e_i. For all c_i, the isolated probability of modification $P_{\text{mod}}(c_i)$ is 2^{-n} and $P_{\text{mod}}(\text{P}) = 1 - (1 - 2^{-n})^m$.

- For the case b) $\text{AcCost}_{\text{mod}}(b) = \text{U}_{\text{mod}}$ and $\text{AcCost}_{\text{mod}}(t) = 2 \cdot \text{U}_{\text{mod}}$ for all $t \in Top_P$. So, $\text{Cost}^2_{\text{maint}}(P) = \Sigma_{r \in S}\, \text{P}_{\text{mod}}(r) \cdot \text{AcCost}_{\text{mod}}(r) = 2^{-n} \cdot (1 + (m-1) \cdot 2) \cdot \text{U}_{\text{mod}} \cong 2^{-n} \cdot 2m \cdot \text{U}_{\text{mod}} \in \mathbf{O(2^{-n} \cdot m)}$.
- For the case c) we directly have that $\text{AcCost}_{\text{mod}}(t) = k \cdot \text{U}_{\text{mod}}$. Since $m=2^k-1$, there are k levels with decreasing width, i-numbered top-down from k to 1, and 2^{i-1} components on each level, so $\text{AcCost}_{\text{mod}}(r^i) = i \cdot \text{U}_{\text{mod}}$
  Finally, $\text{Cost}^2_{\text{maint}}(P) = \Sigma_{r \in S}\, \text{P}_{\text{mod}}(r) \cdot \text{AcCost}_{\text{mod}}(r) = 2^{-n} \cdot (\Sigma_{i=1..k}\, 2^{i-1} \cdot i\,) \cdot \text{U}_{\text{mod}}$
  By using again the approximation $\Sigma_{l=1..p}\, 2^{l-1} \cdot l \cong p \cdot 2^p$., we have: $\text{Cost}^2_{\text{maint}}(P) \cong 2^{-n} \cdot (k \cdot 2^k\,) \cdot \text{U}_{\text{mod}}$
  Since $m = 2^k - 1$, then $2^k = m+1$ and $k = \log_2 [m+1]$ then,
  $\text{Cost}^2_{\text{maint}}(P) \cong 2^{-n} \cdot \log_2 [m+1] \cdot (m+1) \cdot \text{U}_{\text{mod}} \in \mathbf{O(2^{-n} \cdot m \cdot \log_2 m)}$

Topology 5. Cases b) and c) will be studied with these two extreme conditions:

i) Compensated: $\forall c_i \in \text{Bot}_\text{P}$ then $\rho\rho(c_i) = n\, /\, Card(Bot_P)$.

ii) With exceptions: $\exists j \in \text{Bot}_\text{P}\ \rho\rho(c_j) = n - Card(Bot_P) + 1$ and the rest $c_i \in \text{Bot}_\text{P}$ have $\rho\rho(c_i) = 1$, $\text{i} \neq \text{j}$.

- For the case b)-i) we have that $Card(Bot_P) = m\text{-}1$, so $\rho(t) = 1 - 2^{-n}$ for t and $\rho(r) = 1 - 2^{-n/(m-1)}$ iff $r \neq t$. For every example e_i the course $\chi(e_i) = (1-2^{-n/(m-1)}) \cdot (1 - 2^{-n})$. The isolated probabilities are $P_{\text{mod}}(t) = 2^{-n}$ and $P_{\text{mod}}(r) = 2^{-n/(m-1)}$ iff $r \neq t$. $P_{\text{mod}}(P) = 1 - (1 - 2^{-n})^{\text{m}-1} \cdot (1 - 2^{-n/(m-1)})$.
  On the other hand, $\text{AcCost}_{\text{mod}}(t) = 2 \cdot \text{U}_{\text{mod}}$ and $\text{AcCost}_{\text{mod}}(b) = \text{U}_{\text{mod}}$ iff $r \neq t$
  So, $\text{Cost}^2_{\text{maint}}(P) = \Sigma_{r \in S}\, \text{P}_{\text{mod}}(r) \cdot \text{AcCost}_{\text{mod}}(r) = (2^{-n} \cdot 2 + (m-2) \cdot 2^{-n/(m-1)}\,) \cdot \text{U}_{\text{mod}} \in \mathbf{O(m \cdot 2^{-n/m})}$
- For the case c)-i) we have that $\text{Card}(Bot_P) = (m + 1)\, /\, 2$. We have $\rho(t) = 1 - 2^{-n}$ for t and for the k levels of the inverse tree, numbered top-down from 1 to k, we have $\rho(r^j) = 1 - 2^{-n/(2^\wedge(j-1))}$. For every example e_i the course $\chi(e_i) = \Pi_{j=1..k}\, (1 - 2^{-n/(2^\wedge(j-1))}) \leq 1 - 2^{-n/(2^\wedge(k-1))} = 1 - 2^{-2n/(m-1)}$.
  We have that the probabilities of modification are $P_{\text{mod}}(t) = 2^{-n}$. Since $m = 2^k - 1$, we have k levels with increasing width, numbered top-down from 1 to k, we have

2^{j-1} components on each level, and $P_{\mathrm{mod}}(r^j) = 2^{-n/(2^\wedge(j-1))}$. We have $P_{\mathrm{mod}}(P) = (1 - \Pi_{j=1..k} \cdot (1 - 2^{-n/(2^\wedge(j-1))})^{2^\wedge(j-1)})$.
It is obvious that $\mathrm{AcCost}_{\mathrm{mod}}(t) = m \cdot \mathrm{U}_{\mathrm{mod}}$. Since $m = 2^k - 1$, we have $2k-1$ levels: k levels with increasing width, j-numbered top-down from 1 to k, we have 2^{j-1} components on each level, and $\mathrm{AcCost}_{\mathrm{mod}}(r^j) = (2^{k-j+1} - 1 + 2^{k-1-j+1} - 1 + (j-1)) \cdot \mathrm{U}_{\mathrm{mod}} = (3 \cdot 2^{k-1-j+1} - 3 + j) \cdot \mathrm{U}_{\mathrm{mod}}$
Finally, $\mathrm{Cost}^2_{\mathrm{maint}}(P) = \Sigma_{r \in S} \mathrm{P}_{\mathrm{mod}}(r) \cdot \mathrm{AcCost}_{\mathrm{mod}}(r) = \Sigma_{j=1..k} 2^{j-1} \cdot 2^{-n/(2^\wedge(j-1))} \cdot [3 \cdot 2^{k-1-j+1} - 3 + j] \cdot \mathrm{U}_{\mathrm{mod}} = \Sigma_{j=1..k} 2^{-n/(2^\wedge(j-1))} \cdot [3 \cdot 2^{k-1} - 3 \cdot 2^{j-1} + j \cdot 2^{j-1}] \cdot \mathrm{U}_{\mathrm{mod}}$
Since both factors increase very quickly with j, and using the approximation $\Sigma_{l=1..p} 2^{l-1} \cdot l \cong p \cdot 2^p$, we have that we can roughly approximate to: $\mathrm{Cost}^2_{\mathrm{maint}}(P) \cong 2^{-n/(2^\wedge k)} \cdot k \cdot 2^k \cdot \mathrm{U}_{\mathrm{mod}}$
Since $m=2^k-1$, then $2^k=m+1$ and $k=\log_2(m+1)$, so $\mathrm{Cost}^2_{\mathrm{maint}}(P) \cong 2^{-n/m} \cdot m \cdot \log_2 m \cdot \mathrm{U}_{\mathrm{mod}} \in \mathbf{O(2^{-n/m} \cdot m \cdot \log_2 m)}$

- For the case b)-ii) we have that $\mathrm{Card}(Bot_P) = m-1$, so $\rho(t) = 1-2^{-n}$ for t and $\exists j \in \mathrm{Bot_P}\ \rho\rho(c_j) = n - \mathrm{Card}(\mathrm{Bot_P}) + 1 = n-m+2$ and the rest $m-2$ components $c_i \in \mathrm{Bot_P}$ have $\rho\rho(c_i)=1$, $i \neq j$. There are $m-2$ examples with course $\chi(e) = (1-2^{-1}) \cdot (1 - 2^{-n})$. The rest $n-m+2$ examples have $\chi(e) = (1-2^{-(n-m+2)}) \cdot (1-2^{-n})$.
  The isolated probabilities are $P_{\mathrm{mod}}(t) = 2^{-n}$ and $\exists j \in \mathrm{Bot_P}\ P_{\mathrm{mod}}(r) = 2^{-(n-m+2)}$ and the rest $m-2$ components $c_i \in \mathrm{Bot_P}$ have $P_{\mathrm{mod}}(r) = 0.5$. We have $P_{\mathrm{mod}}(P) = (1 - (1-2^{-n}) \cdot (1-2^{-(n-m+2)}) \cdot (1-0.5)^{(m-2)})$.
  On the other hand, $\mathrm{AcCost}_{\mathrm{mod}}(t) = 2 \cdot \mathrm{U}_{\mathrm{mod}}$ and $\mathrm{AcCost}_{\mathrm{mod}}(b) = \mathrm{U}_{\mathrm{mod}}$ iff $r \neq t$
  So, $\mathrm{Cost}^2_{\mathrm{maint}}(P) = \Sigma_{r \in S} \mathrm{P}_{\mathrm{mod}}(r) \cdot \mathrm{AcCost}_{\mathrm{mod}}(r) = (2^{-n} \cdot 2 + 2^{-(n-m+2)} + (m-2) \cdot 0.5 \cdot 2) \cdot \mathrm{U}_{\mathrm{mod}}$. Since the first two terms are always ≤ 1 if $n > m$ and n great, then $\mathrm{Cost}^2_{\mathrm{maint}}(P) \in \mathbf{O(m)}$.
- Case c)-ii) would be lengthy to study directly. However, the results are very similar to the case of considering a vertical propagation like topology 2 on one side and the biggest subtree of topology c)- i) on the other side. This latter part, composed of $((m+1) / 2) -1$ nodes, will dominate the whole result because it is reinforced by $(m+1)/4$ examples only, one for each component of Bot_P. (The former part is $\mathrm{O}(2^{-n} \cdot m^2)$).
  Using the results of topology c)- i) and changing n by $(m+1)/4$ we have: $\mathrm{Cost}^2_{\mathrm{maint}}(P) \cong 2^{-(m+1)/4m} \cdot m \cdot \log_2 m \cdot \mathrm{U}_{\mathrm{mod}} \in \mathbf{O(m \cdot \log_2 m)}$.

10. Conclusions

Learning without thinking is useless.
Thinking without learning, dangerous.
Confucius, 551-479 BC

Abstract: *this chapter discusses the results that have been fulfilled in relation to the expectations and possibilities that were initially aroused from the ideas and methodological tools that motivated the goals of this thesis. Their accomplishment is shown by the main contributions of this work, as well as the open questions and the future work. The chapter ends with a broader view of the overall results and an appraisal of the practical possibilities and philosophical implications of this work.*

Keywords: Evaluation Measures, Inference Processes, Induction, Deduction, Semantic Information, Kolmogorov Complexity, Utility, Reasoning, Inference Paradox, Information Gain, Reinforcement, Intensionality, Measurement of Cognitive Abilities, Evaluation of Logical Theories.

10.1 Introduction

The determination of approaching inference processes from a non-strictly semantical point of view seemed risky at first sight. How much and how well could be addressed with measures constructed on descriptional, computational and numerical tools was a question at the beginning of this work. However, some semantic or logical approaches had unsuccessfully essayed to understand different inference processes in a joint and consistent way, under some kind of unified logical framework. Nonetheless, there were no valid attempts for ascertaining the result of the different inference processes. Hence, in my opinion, an approach based on evaluation (in this case syntactical approaches also deserved to be studied.

The results have been, fortunately, quite satisfactory, according to the expectations at the beginning of this work. Such important traits of inference processes such as explicitness, implicitness, novelty, intermediate information, representational optimality, informativeness, extensionality, intensionality, plausibility and confirmation have been accounted by these measures. Moreover, the main tools have been basic definitions and properties of descriptional (Kolmogorov) complexity on one hand, and a quite simple theory of reinforcement on the other. In other words, the measures which have been presented (except from the difficult account of the concept of intensionality) have been kept without entering in intricate mathematics. The reason may be found in my conviction of keeping the things as simple as possible and, of course, my declared inability to use some powerful and sophisticated mathematical techniques that possibly could have been useful in some occasions. In particular, some results of descriptive set theory, topology, model theory, and complex probability distributions should also be applied for accounting, in a joint way, for different inference processes. The use of some of them would have given the thesis another scope and perhaps would have given some interesting results at a higher level. Be it a voluntary outcome or not, I think that the fact that most of the measures have been kept quite intuitive and comprehensible must be seen as a positive payoff rather than a negative feature.

Furthermore, in my opinion, the methodology and tools must follow the needs that are generated by the goals of any work. Obviously, these goals can only remain if some ideas are found that supports in advance the subsequent research. In my case, these have been the following:

- The idea of re-connecting the intuitive notion of information with resource consumption or computational/reasoning effort. In particular, the weighing of space and time as given by the function LT seemed to me an appropriate measure of effort.

- The idea of a non-probabilistic but quantitative account for confirmation under the following simple assumption: the value of confirmation that a deductive connection provides must be much greater than the one a hypothetical connection provides.
- The idea that hypotheses and theories must not be accompanied by a single value of plausibility. Any useful account of confirmation must be detailed for any part of the theory. In particular, most of the differences between consilient/intensional theories and descriptional/(partially extensional) theories are originated by the degree of uniformity of the distribution of this confirmation, a question that has been neglected in the machine learning literature.
- The idea that reasoning abilities can be evaluated by the use of formally and computationally derived measures. Moreover, the conviction that there is no need of an external and initially predetermined reference in order to scale the intelligence of an agent, a person, an animal or any cognitive system.

Let us see in which extent the preceding methodology and ideas have been fruitful:

10.2 Main Contributions

The main contributions of this thesis are:

1. A measure of time-ignoring information gain $V(x|y)$ which represents the degree of information of x which is *implicitly* in y. ***A new effective measure of computational information gain*** $G(x|y)$ which depends on the computational effort (time and space) and can be used to measure the proportion of x which can be easily obtained on the help of y. In other words, the degree of information of x that is explicitly in y is given exactly by $1 - G(x|y)$.

2. ***Representation Gain and Representational Optimality Measures*** are also defined from computational information gain. A general notion of simplification and the definition of a representational optimality criterion are introduced, as well as general measures of System Optimisation and Systematic Power.

3. ***Uniform account for induction and deduction***. In induction, Popper's idea of informativeness is perfectly grasped by the use of G. Deduction can also be informative and different measures are introduced for several deductive paradigms, especially first-order theories. A new notion of authentic learning is introduced, ensuring that learning has taken place, independently of how compressible the evidence is. Moreover, this notion is applicable to deduction, showing that learning deals not only with induction

but also with deduction. A comparison with Hintikka's ideas is performed, establishing the relationship between G and Surface Information, and between V and Depth Information.

4. ***A new measure of reinforcement that quantifies confirmation propagation inside a theory***. Reinforcement allows a more detailed treatment of exceptions and provides different ratings for different parts of a theory, not the single probability value for the whole theory usually given by prior distributions.

5. ***Reinforcement behaves appropriately as a measure of confirmation for different inference processes*** such as induction, abduction, analogy and deduction, which are involved in theory construction. Some previously *vague* notions such as Whewell's 'consilience' and explanatory induction are easily formalised under this framework.

6. Gain and Reinforcement act as a perfect team to discern which rules should be left explicitly in the representation of a theory. From here, ***the need of intermediate information is formally realised and an oblivion criterion is derived*** and extended to manage past and explained evidence. In this context, the difference between eager and lazy methods is clarified and a degree of laziness is defined.

7. ***The idea of intensionality is formalised*** in terms of avoidance of exceptions, these seen as extensional or non-validated parts of a theory. It is directly applied to logical theories and then extended to any descriptional language, based on a formal and general definition of subprogram. This idea is also connected with 'consilience'.

8. Motivated by the problems of the MDL principle, different concepts based on descriptional complexity are introduced, such as projectible descriptions and stable descriptions. The definition of an explanatory variant of Kolmogorov Complexity allows to define ***an explanatory counterpart to the MDL principle.***

9. ***A non-anthropomorphic test of intelligence***, which is based on computational and information-theoretic notions, that can make an important advance in the evaluation of AI progress. Several particularisations for different inductive and deductive factors are also introduced. Moreover, psychometrics finds its long-awaited theoretical foundation in information theory and computation.

10. ***Application of the measurements to different kinds of logical and knowledge-based systems***, such as Horn theories, databases and software systems. Some issues of *novel* knowledge discovering in databases and the

need of intermediate information can be better understood in terms of information gain. In the case of software systems, maintainability can be studied under the theory of reinforcement, since the more reinforced (used) a software component is, the less probable it should be modified in the future.

Although the main measurements, computational information gain, reinforcement and intensionality, are defined independently, they (alone or combined) are useful to formalise or better comprehend different concepts which have been traditionally rather ambiguous: novelty, explicitness/implicitness, informativeness, surprise, interestingness, aesthetics, comprehensibility, consilience, utility, unquestionability, ...

Naturally, the relationships between these measures and other classical evaluation measures are analysed. Information gain is analogous to Quinlan's gain ratio for induction and to Hintikka's surface and depth information for deduction. Intensionality is closely related to information gain, since extensional descriptions are never informative. Comprehension is also related to the notion of unquestionability, given when there are not alternative explanations. Reinforcement and the MDL principle are also positively related but reinforcement is more robust to random evidences, giving more informative hypotheses. Some of these results are obtained in general and others are particularised for first-order logical theories.

Part of these contributions have appeared in some journals and conferences, with the aim of disseminating this work. Concretely, contributions 1 and 3 appeared in Kurt Gödel Colloquium / Barcelona Logic Meeting (KGS'99), contribution 4 will appear in the International Journal of Intelligent Systems (IJIS), contributions 7 and 9 were partially included (joint work with N. Minaya) in the International Symposium of Engineering of Intelligent Systems (EIS'98) and an article version of them is submitted for a special issue on "Alan Turing and Artificial Intelligence" of the Journal of Language, Logic and Information (JoLLI), contribution 7 appeared in Model-Based Reasoning in Scientific Discovery (MBR'98) and will be published in *Philosophica*, contribution 8 was presented in MBR'98 and is accepted for *Foundations of Science* (in collaboration with I. García); finally, part of the ideas of contribution 10 on databases are used in a paper to be presented in European congress on Systems Science (ESS'99, joint work with F. Alamagnac).

As we will see in the next section this research has also opened many other questions to investigate, and I hope that they will show in much more extent the usefulness of the preceding contributions.

10.3 Open Questions and Future Work

There are two kinds of questions raised by this thesis: technical ones, which in some sense highlight the limitations or unsolved problems left by this thesis, and new theoretical ones, which are, in the end, expected and desired at the end of any scientific task which also looks for new fertile fields to explore.

Among the technical open questions, one main source of uneasiness is found about the computational cost of the measures that have been introduced. As it has been shown, the gain function *G* is computable but intractable. We have argued, however, that it is not strange to be this way, because, if it were efficient, it could be used for guiding inductive and deductive processes, precisely impairing what is measured, the effort of a process of inference.

Nonetheless, some lower approximations can be obtained from some other measures that have been shown to be partially and positive related with it, such as intensionality. I say lower approximations because they are useful to discard many non-informative hypotheses, but not all. For instance, the avoidance of extensionalities (or explicitness) can be a good heuristic to obtain informative results. Nonetheless, reinforcement must always be used in order to avoid informative but fantastic outcomes.

Fortunately, reinforcement is computationally feasible, since the algorithm that has been presented can be adapted to re-calculate only the parts of the theory that change after a revision takes place. Moreover, the oblivion criterion can be used to optimise the use of space resources, since the evidence is usually so huge that part of it must be forgotten. The theory of reinforcement allows to know which are to be forgotten. Nonetheless, a study of other validation propagation algorithms that have been introduced in the last decades in the area of artificial neural networks (e.g. backpropagation) could inspire some improvements or new algorithms for computing reinforcement.

There are still, of course, many old problems to solve (or to accept), such as the intractability of induction. This intractability (well-known since Gold's results) is even backed by some results of this thesis, since it has been shown that polynomial algorithms that work exclusively from the data cannot find *informative* hypotheses in general. This justifies the avoidance of data-driven inductive methods and the use of more 'randomised' techniques, such as genetical algorithms, which may be good inducers *on the average*. The use of detailed evaluation measures such as reinforcement may be a perfect optimality criterion to guide the selection mechanism of this kind of algorithms. This is beginning to be essayed in [Hernández-Orallo and Ramírez-Quintana, 1998a, 1998b, 1999a]. In the future, it is envisaged the combination of deduction and induction for incomplete and inefficient deductive systems, especially present in higher-order logic. A deductive strategy could be largely benefited from inductive techniques.

Among the new open questions, I would expressly highlight the implications of the theoretically-based measurement of cognitive abilities which have been introduced in chapter 8. Many fascinating questions are open from the correlation of classical psychometrics tests (IQ tests) and a C-test which is exclusively generated from the information theoretic notions of comprehensibility and unquestionability. I think the most transcendental question that could be solved in the future would be "*How intelligent are we?*" from a non-anthropomorphic point of view. Moreover, psychometrics could start a theoretical study (helped by theoretically computer science) in order to study formally (and not only experimentally) the independence or dependence of several cognitive factors. The fact that deductive abilities correlate with inductive ones supports the position that induction and deduction are not inverse processes in any way.

In the mid term it seems to me that the idea of the Turing Test as a practical test of intelligence should be left behind, and substituted by computational and factorial tests of different cognitive abilities, a much more useful approach for artificial intelligence progress.

Finally, an issue that has only partially been addressed in this thesis is the implications of the measures that have been introduced in philosophy of Science. Although some notions originally introduced in this context have been formalised, such as Whewell's consilience, Popper's informativeness, a notion of discovery, and the distinction between explanatory and descriptive induction, the results could be further exploited into a more comprehensive account of informativeness and confirmation in the context of philosophy of Science. The same applies, in more or less extent, to philosophy of mathematics, as has also been sketched in chapter 4.

In the end, there is a lot of theoretical work to be done in the previous technical and theoretical questions. For instance, the connection of information gain with cryptography could unveil further insights about the behaviour of G. Its relationship with PAC-learning may suggest the definition of a probabilistic version of information gain, although KT partially accounts for this because it does weight space and time by taking all the possible combinations. Another line of future theoretical research would be the relationship and combination of the theory of reinforcement learning with fuzzy logic, or even with neuro-fuzzy approaches, which, in my opinion, may be the closest approach to the way reinforcement propagation has been defined.

Moreover, there is even more experimental work to do for exploiting the possible applications of the measures that have been introduced in this thesis. The recent overwhelming popularity of field of rational agents has not been accompanied by successful and general combinations of different inference processes, partially due to the lack of measures that could be consistently applied for all of them.

The most immediate applications can appear from the use of the measurements for first-order logic in chapter 7, which can be used directly for the ILP community. The previous chapter is also a suggestion of more or less prospective applications for databases, software systems and natural language, where the ideas of informativeness and intensionality can be crucial for communication and understanding.

As long as more fields dealing with information or knowledge systems, artificial intelligence, natural language processing and machine learning would be integrating more inference techniques, much more applications are envisaged of the measurements and other concepts which have been presented by this thesis.

10.4 Concluding Remarks

According to the results and open questions that have been discussed in the previous sections, it seems to me that the main goal of the thesis as discussed in chapter 1: "*the formal study of concept synthesis usefulness and aftermath in terms of information gain and reinforcement inside inference systems, consistently and equally applicable for both deductive and inductive inference*" has substantially been met. The set of measures which have been introduced allow a detailed analysis of the value of the output of any inference process with respect to the input and the context (background knowledge or axiomatic system), both in terms of informativeness and confirmation.

Appendix A. A Brief Review of Kolmogorov Complexity

Dila —dijo don Quijote—, y sé breve en tus razonamientos; que ninguno hay gustoso si es largo.

Miguel de Cervantes Saavedra, 1547-1616, Don Quijote de La Mancha

Abstract: this chapter is devoted to give a quick and comprehensive review on Kolmogorov Complexity and some of the properties and related concepts which are used from chapter 3 to chapter 8.

Keywords: Information, Entropy, Turing Machines, Compression, Randomness, Inductive Inference, MDL Principle, Occam's Razor, Universal Distribution, Information Distance, Logical Depth.

A.1 Introduction

Descriptional Complexity or Intrinsic Information (also referred as Kolmogorov Complexity or Algorithmic Complexity) was independently introduced (with initial different reasons and directions too) by R.J. Solomonoff, A.N. Kolmogorov and G.J. Chaitin. The theory underlying Descriptional Complexity contains deep and sophisticated mathematics. Fortunately we only need some of its definitions and results. In this appendix we include definitions and properties that may be used or may be useful in some way for the rest of this dissertation. We refer to [Chaitin 1974] [Chaitin 1992] [Li and Vitányi 1997] (some extracts have been directly taken from them) for a much formal and precise extended treatment of the matter.

The departure of this fascinating theory is incredibly simple:

The Minimal Length Encoding MLE(x) of a finite string x is simply the length of the shortest program, in a Turing machine without any output, which prints x. The choice of another programming language instead a Turing machine would result in another value with an irrelevant constant difference. Therefore the complexity is defined independently of the algorithmic language or machine that may be used.

For instance, the MLE(s) of a string s= "11111...." with infinite length is a constant k because in any universal descriptional mechanism there is a program of the form "REPEAT FOREVER PRINT '1'". The MLE(s') of a string s'= "101010....10" of length n would be, in general, (log $n/2 + k'$) because we have to express the length in the program in order to make it stop. Then the program would be something like "REPEAT n/2 TIMES PRINT '10'".

A.2 Mathematical Definition and Properties

The term Algorithmic Complexity is generally associated with its 'purest' variant, based on the MLE or the Minimal Description Length (MDL), i.e., the shortest string that, taken as an algorithm, produces the original string. Formally,

Definition A.155. Plain Complexity

$$C_f(x) = \min \{ l(p) : f(p) = x \}$$

In computer science terminology, p would be the program, f would be the computer. We will often refer to p as the *compressed* string and x to the *original* or *plain* string.

The following invariance theorem is proved elsewhere [Li and Vitányi 1997]:

Theorem A.34 *Descriptional Complexity is an invariant, universal and objective property of each string (upto a constant value) independently of the computer, function or whatever descriptional method may be used.*

Since any descriptional method can be simulated by a Universal Turing Machine, a machine M (or a universal partial recursive function) is selected arbitrarily as reference. For this reason, plain complexity will be referred simply as $C(x)$.

There is a conditional version of descriptional complexity which is defined as follows:

Definition A.156. Conditional Plain Complexity

$$C_f(x \lfloor y) = \min \{\, l(p) : f(p,y) = x \,\}$$

where $f(p,y)$ means the execution of program p in f with input y.

It is easy to see that[34] $C(x) = C(x|\varepsilon)$, $C(x|x) = O(1)$ and $C(x\lfloor y) \leq C(x) + O(1)$. This definition will be of great importance to clarify some intrinsic features of interdependence between some concepts or parts of a given theory.

The importance of the invariance theorem is crucial for the recent popularity and wide use of Kolmogorov Complexity. Regarded as a measure of information that contains a string x, it differs from the other classical views of information (Shannon, Hintikka, ...) where the information of an object depends on the number of objects of the alphabet or world under consideration. [Li and Vitányi 1997] show clearly this distinction:

> We are interested in a measure of information content of an individual finite object, and in the information conveyed about an individual finite object by another individual finite object. Here, we want the information content of an object x to be an attribute of x alone, and not to depend on, for instance, the means chosen to describe this information content. Making the natural restriction that the description method should be effective, the information content of an object should be a recursively invariant property among the different description systems. Pursuing this thought leads straightforwardly to Kolmogorov complexity.

Another fundamental (and trivial) property is the following one:

Theorem A.35. Upper limits

$$C(x) \leq l(x) + O(1).$$

since we can always describe a string x as a program more or less similar to "PRINT x" depending to the descriptional scheme used. We will refer this program as the *trivial mention description*.

However, as [Chaitin 1974] points out:

[34] O(1) represents any constant that is independent of x and y. In the introduction the notation $a <^+ b$ iff $a \leq b + O(1)$ was given which is commonly used in the rest of this thesis. In this appendix it will only eventually used.

> the complexity of the great majority of strings of length *n* is approximately *n*, and very few strings of length *n* are of complexity much less than *n*. The reason is simply that there are much fewer programs of length appreciably less than *n* than strings of length *n*. More exactly, there are 2^n strings of length *n*, and less than 2^{n-k} programs of length less than *n* - *k*. Thus the number of strings of length *n* and complexity less than *n* - *k* decreases exponentially as *k* increases.

Moreover, the *incompressibility theorem* says that "every finite set *A* of cardinality *m* has at least $m(1-2^{-c}) + 1$ elements *x* with $C(x) \geq \log m - c$".

It is precisely for those strings that are so irregular and cannot be compressed, that the shortest description is the trivial mention description. These strings are called *incompressible*. A formal generalisation of this is given by the following definition:

Definition A.157 A string *x* is *c-incompressible* if $C(x) \geq l(x) - c$.

Incompressible strings have an important feature, they pass all tests of statistical randomness (the contrary is not the case). As Chaitin states again [Chaitin 1974]:

> These considerations have revealed the basic fact that the great majority of strings of length *n* are of complexity very close to *n*. Therefore, if one generates a binary string of length *n* by tossing a fair coin *n* times and noting whether each toss gives head or tail, it is highly probable that the complexity of this string will be very close to *n*. In 1965 Kolmogorov proposed calling random those strings of length *n* whose complexity is approximately *n*.

A non-intuitive property of incompressible strings is that it may have compressible substrings, corresponding to the known fact that a random sequence *must* contain long runs of zeros.

Another non-intuitive property is that *C*(*x*) is nonmonotonic on prefixes, i.e. the complexity of a part can turn out to be bigger than the complexity of the whole[35].

Another important property is expressed as the non-additive character of plain descriptional complexity:

$$C(x,y) \leq C(x) + C(y) + O(\log(\min(C(x), C(y))))$$

The complexity of two strings (the length of the shortest program that can print them *separatedly*) is less (or equal) than the sum of the complexities of both string plus a term that is necessary, in the worst case, to separate one from another.

[35] For instance, we may have a very short program *p* to print the first 65536 (2^{16}) primes. Modifying that program to print the first 65530 (2^{16}-6) primes will probably be a little larger (to express the −6 difference).

This ugly term can be eliminated by using the Algorithmic Prefix Complexity, defined in a similar way:

Definition A.158. Prefix-free Complexity

$$K(x) = \min \{ l(p) : \phi(p) = x \}$$

but forcing ϕ to be a *partial recursive prefix function*[36], i.e., if $\phi(x) < \infty$ and $\phi(y) < \infty$, then x is not a proper prefix of y. This makes K additive:

$$K(x,y) \leq K(x) + K(y) + O(1)$$

$K(x)$ has some additional advantages over $C(x)$, and therefore is often considered the *standard* algorithmic complexity or Kolmogorov Complexity.

There are several properties that are useful when working with Kolmogorov Complexity:

Some Properties:

$K(x^* | x) \leq \log l(x) + O(1)$

$K(x | x^*) \leq O(1)$

$C(C(x)) \leq \log l(x) + 1$

$C(x^*) = C(x) + O(1)$

The proofs can be found in [Li and Vitányi 1997].

A.3 Mutual Information and Information Distance

The algorithmic information about y contained in x is defined as:

Definition A.159. Common Information

$$I_c(x : y) = K(y) - K(y | x)$$

and the mutual information:

Definition A.160. Mutual Information

$$I(x : y) = K(x) + K(y) - K(x, y)$$

Although both definitions can differ by a great constant, there is an important asymptotical result which states:

Theorem A.36.

$$I_c(x : y) \overset{+}{=} I_c(\mathrm{y} : \mathrm{x}) \overset{+}{=} I(x : y)$$

The notions of common and mutual information allow the definition of a very interesting concept: universal distance [Bennett et al. 1997]:

[36] This kind of coding is well-known in the theory of communication. One message can be separated from the following one without any information of the position or the length.

> While Kolmogorov Complexity is the accepted absolute measure of information content in an individual object, a similarly absolute notion is needed for the information distance between two individual objects, for example, two pictures.

However, several definitions of Information Distance have been presented.

Definition A.161. Variants of Information Distance

Universal Effective Information Distance:

$$E_0(x,y) = \min(l(p) : U(p,x) = y, U(p,y) = x$$

where U is a universal machine.

The Max Distance:

$$E_1(x,y) = \max(K(x\lfloor y), K(y|x))$$

Reversible Distance:

$$E_2(x,y) = KR(y|x) = \min\{l(p) : UR(p,x) = y\}$$

where UR is a universal reversible machine.

Sum Distance:

$$E_3(x,y) = K(x\lfloor y) + K(y|x) + O(\log K(x,y))$$

From these definitions, and in order to select the most appropriate one, [Bennett et al. 1997] show the following relations:

Theorem A.37. Relation between Distances

$$E_1(x,y) = \max\{K(y|x), K(x\lfloor y)\} =^{\log}$$
$$E_2(x,y) = KR(y|x) =^{+}$$
$$E_0(x,y) = \min\{l(p): U(p,x) = y, U(p,y)=x\} <^{\log}$$
$$K(x\lfloor y) + K(y|x) =^{\log} E_3(x,y) <^{\log}$$
$$2E_1(x,y)$$

Finally, E_1 is selected as the cognitive distance between two objects, by using the following trivial modification:

Definition A.162. Universal Information Distance

$$E(x,y) = E_1(x,y) + c \text{ iff } x \neq y, \text{ and}$$
$$E(x,y) = 0 \text{ iff } x = y$$

and it satisfies the triangle inequality, it is 0 if and only if $x = y$, it is symmetric and it is upper semicomputable and normalised, i.e., it is an admissible distance. Moreover it is a universal distance, because for every admissible distance $D'(x,y)$ we have $E(x,y) <^{+} D'(x,y)$. [Bennett et al. 1997]

A.4 Algorithmic Probability and Inductive Reasoning

If we are talking about *K(x)* as a complexity measure is because it provides an objective value of the complexity of a string *x*. This first correspondence shows both counter-intuitive and intuitive results: incompressible (random) strings are "complex" and very regular strings are simple.

The other correspondence is more accurate and it is established between *K(x)* and the information content of a string. Information and complexity get connected through the notion of probability [Li and Vitányi 1997]:

> This gives an objective and absolute definition of 'simplicity' as 'low Kolmogorov complexity'. Consequently, one obtains an objective and absolute version of the classic maxim of William of Ockham (1290?-1349?), known as Occam's razor: "*if there are alternative explanations for a phenomenon, then , all other things being equal, we should select the simplest one*". One identifies 'simplicity of an object' with 'an object having a short effective description. In other words, *a priori* we consider objects with short descriptions more likely than objects with only long descriptions. That is, objects with low complexity have high probability while objects with high complexity have low probability. Pursuing this idea leads to the remarkable probability distribution $2^{-K(x)}$ below.

Thus we can formally define the probability that a string *x* was algorithmically produced by a random sequence taken as a program:

$$R(x) = 2^{-K(x)}$$

Occam's razor has always been a recurrent theme in philosophy of science and induction, but now it assumes a main role. Although Karl Popper said that there is no such a objective criterion, *K(x)* is indeed an objective criterion for simplicity. This is precisely what R.J. Solomonoff and G. Chaitin proposed as a 'perfect' theory of induction, in [Li and Vitányi 1997] words. The same [Chaitin 1974] explains:

> Solomonoff and the author proposed that the concept of complexity might make it possible to precisely formulate the situation that a scientist faces when he has made observations and wishes to understand them and make predictions. In order to do this the scientist searches for a theory that is in agreement with all his observations. We consider his observations to be represented by a binary string, and a theory to be a program that calculates this string. Scientists consider the simplest theory to be the best one, and that if a theory is too "ad hoc", it is useless. How can we formulate these intuitions about the scientific method in a precise fashion? The simplicity of a theory is inversely proportional to the length of the program that constitutes it. That is to say, the best program for understanding or predicting observations is the

shortest one that reproduces what the scientist has observed up to that moment.

If we regard theories as descriptional sequences of some facts, by using Occam's Razor Principle we will choose the one that is shorter. In other words, Occam's Razor is just the assumption of the probability distribution $2^{-K(x)}$. This leads to the definition of the Minimum Description Length (MDL) principle, which is discussed in chapter 2. Note that Chaitin's claim "*Scientists consider the simplest theory to be the best one, and that if a theory is too "ad hoc", it is useless*" is only partially fulfilled by the MDL principle, since for random evidences, the MDL principle gives the most "ad hoc" theory, the evidence itself.

Although the MDL principle can be seen as a plausibility criterion, it can also be used as a methodological one. In other words, a selection criteria is always directly applicable to the sifting of theories. But most of the times, we do not have a 'given' range of theories to select. Precisely, we would like the contrary: to obtain a good (read shorter) theory from facts.

Given a sequence of facts, say x, we may essay the following algorithm.

Universal Enumeration Algorithm (UEA): From i=1 upto l(x) take the 2^i possible programs of length i and run them to check if one of them produces x. If it does, then stop. If found, this is the shortest description. If not, "PRINT x" is the shortest description.

We have presented a universal algorithm to find the simplest theory to explain some facts! Unfortunately, this algorithm is uncomputable, because some programs never end (the undecidability of the halting problem). There are some ways of facing this problem (such as executing the programs up to a limit number of steps), but even if we make it computable, there are some drawbacks:

- The UEA will be still $O(2^{l(x)+1})$ in the worst case.
- Due to the nonmonotonicity on prefixes of $K(x)$, if we are given a new example, the minimum descriptional theory may be shorter. Obviously, only one example does not justify a re-execution of the algorithm, because the theory may be shorter only up to a laconic constant. It is better to patch the new example.

A.5 Resource-bounded Complexity and Universal Search

The problem with $K(x)$ and the MDL principle is that they are not computable unless the description mechanism is restricted drastically. Instead of this undesired restriction, there is a way of maintaining the description mechanism powerful (such as Turing Machines, grammar and general-purpose computers, i.e. universal partial recursive functions) without falling in incomputability. This can be accomplished by limiting a resource, e.g. time, of the strings that we would be used as description. As we will see below this is sufficient to ensure that the universal enumeration algorithm (UEA) ends.

The other reason to introduce resource bounds is more intuitive. We are considering the shortest description as the best one. But "decompressing" the description may consume so many resources (e.g. time) that we would prefer a description that is a little bit longer but easier to decompress. [Chaitin 1974] puts it this way:

> There are less than 2^n strings of complexity less than *n*, but some of them are incredibly long. If one wishes to communicate all of them to someone else, there are two alternatives. The first is to directly show all of them to him. In this case one will have to send him an incredibly long message because some of these strings are incredibly long. The other alternative is to send him a very short message consisting of *n* bits of axioms from which he can deduce which strings are of complexity less than *n*. Although the message is very short in this case, he will have to spend and incredibly long time to deduce form these axioms the strings of complexity less than *n*.

The usual resource-bounded variant of descriptional complexity takes into account both the time and the working space used to decompress the description. Since the required space is generally smaller than the time, we directly introduce the following definition:

Definition A.163. Time-Bounded Kolmogorov Complexity

$$C^t(x) = \min \{ l(p) : \phi^t(p) = x \wedge \text{cost}\ (p) \leq t \}$$

where ϕ is a universal computer which does the computation of translating p into x in at most t steps. Obviously $t > l(x)$ if we want $C^t(x)$ to exist, because printing x just takes $l(x)$ steps. Also obviously, for $t = l(x) + c$, $C^t(x)$ must exist (by using the program "PRINT x"). There is another interesting property:

$$C^{c \cdot t \cdot log(t)}(x) \leq C^t(x) + c$$

i.e., a rise of the time limit allows obtaining shorter descriptions. Although $C^t(x)$ is monotonically decreasing with respect to t, each x will determine a different relation between t and $C^t(x)$.

We have dealt about a *bound* but we are more interested in finding a compromise between length and time.

There are many ways to weigh the length of the description p and the time to decompress it to the original string x, e.g. the product $l(p) \cdot cost(p,x)$, a normalised product $l(p)\cdot cost(p,x)/l(x)$ and others. The expression that has been shown to be the most appropriate[37] one is due to [Levin 1973] and is referred as *Kt* complexity[38]:

Definition A.164. Levin-Solomonoff Complexity

$$Kt(x) = \min \{ LT(p) : \phi(p) = x \}$$

where $LT(p) = l(p) + \log \text{cost}(p)$.

It is straightforward to obtain the limits of Kt:

$$\log l(x) + c \leq Kt(x) \leq l(x) + \log l(x) + c$$

The left-hand side is justified because x must be printed and this takes at least $l(x)$ steps. The right hand side is obtained by using the mention trivial description “PRINT x”.

The great advantage of *Kt* is that it is computable. From here it is easy to see that:

$$K(x^{*LT} \mid \text{x}) \leq O(1) \text{ where } x^{*LT} \text{ is the program for } x \text{ where } \text{LT}(x^{*LT}) = Kt(x)$$

Since we have that *Kt(x)* is effectively computable, we can reformulate our Universal Enumeration Algorithm (UEA):

Time Bounded Enumeration Algorithm (TBEA): Begin with the upper limit L= *$l(x)$ + $\log l(x)$. From $i=1$ to $l(x)$ take the 2^i possible programs p of length i and run them (upto a limited number of computation steps 2^{L-i}) to check if $\phi(p)=x$. If it does, then modify* L *to $(l(p) + \log cost(p))$. If finally found, this is the shortest description. If not, "PRINT x" is the shortest description.*

The worst case of this algorithm is still $O(2^{l(x)+1})$ but for compressible strings it is likely that *L* will be reduced and therefore the average cost of the algorithm. Furthermore, if we know a good initial description we can start with its corresponding *L*, using TBEA as a *theory refinement algorithm*. Moreover, TBEA is fully parallelisable. Also, a slight restriction of the descriptional language may turn the algorithm almost tractable.

The TBEA is a variant of the SEARCH algorithm presented in [Li and Vitányi 1997] which is also a variant of the *universal optimal search procedure* of Levin which shows a worst-case cost of only $O(2^{K(x)+1})$ in [Levin 1973].

[37] Levin showed its optimality for universal search problems. That is to say, an enumeration algorithm, ordered by LT, was optimal in the sense that there is no other search algorithm that can be better for all the possible search problems.

[38] Using the logarithm of the cost instead of the cost or the product of $l(p) \cdot cost(p)$ would allow the consideration of shorter programs that are NP-hard (or exponential), that otherwise would be replaced by the program "PRINT x" that would have less complexity.

A.6 Algorithmic Potential

Before, we have seen that the more time cost we allow the shorter descriptions we expect to obtain. This means that for each x, the length of the description and the time to compute it are inversely related. This relation between length and time suggests the introduction of two borrowed concept from physics. Following [Li and Vitányi 1997], time is a resource identified with 'energy' and:

> Intuitively, we would like to define *potential* as the amount of time that needs to be pumped into a number by a computation that finds it.
>
> Computing a large composite number from two primes costs only a small amount of time. To recover the primes is likely to be difficult and time-consuming. This suggests a notion of potential numbers such that high-potential primes have relatively low-potential products. Such products would be hard to factor, because all methods must take the time to pump the potential back. We will show that if factoring is not in P, then this indeed is the reason why.

Formally, a string x is *k-potent* if k is the least positive integer such that $Kt(x) \leq k \log l(x)$. For instance, the string 1^n is 1-potent because $Kt(x) \approx 1 \cdot \log(n)$ whereas an incompressible string s is *$(l(s)/\log l(s) +1)$-potent* since $Kt(s) = l(s) + \log l(s)$. There is a direct correspondence with potential and some computational complexity theorems. For more details see [Li and Vitányi 1997].

If we regard set of concepts as sequences, a potent sequence can be the formal correspondent to the notion of a hard-to-learn concept.

A.7 Algorithmic (or Logical) Depth and Sophistication

Another intuitive notion of sequences that seems to be formalisable by variants of Kolmogorov Complexity, is depth [Li and Vitányi 1997]:

> From the point of view of an investigator, a sequence is deep if it yields its secrets only slowly: one will be able to discover all significant regularities in it only if one analyzes it long enough.

Taking $Kt(x)$ (the potential) directly results in incompressible strings as being the deepest, which is not intuitively accurate. A better approach can be considered if we recall that a binary string x is *b-compressible* if $K(x) \leq l(x) - b$. From here we can introduce the notion of logical depth as it was introduced by [Bennett 1988]:

Definition A.165 A string x is *(d,b)-deep* if and only if d is the least time required by any b-incompressible program to print x.

We also call $\varepsilon = 2^{-b}$ the significance level. We say a string *x* is *d-deep* iff *x* is *(d,∞)-deep*.

Definition A.166 At any significance level, *x* is *d-shallow* if its depth does not exceed *d*. Any string *x* must take at least *n= l(x)* steps to be printed. If *x* is *n*±O(1)-*shallow* (at all significance levels), then we will simply call *x* shallow.

From both definitions we get intuitive consequences. A random (incompressible) string *x* of length *n* is always shallow, because it can be printed by its shortest program (PRINT 'x') of length *n* ±O (1), in *n* steps. But simple strings such as 1^n are also shallow. So we have that logical depth is different from both algorithmic information and algorithmic potential.

An interesting property (for biology) is the following one [Li and Vitányi 1997]:

> Depth is stable. That is, deep strings cannot be quickly computed from shallow ones. In the genetical sense, organisms evolve relatively slowly. This may be called the Slow Growth Law. There is a mathematical version of such a law. Consider any string *x* and two significance parameters $s_2 > s_1$. A random program generated by coin tossing has probability less than $2^{-(s2-s1)+O(1)}$ of transforming *x* into an excessively deep output, one whose s_2-significance depth exceeds the s_1-significance depth of *x* plus the run time of the transforming program plus O(1).

Koppel introduced the notion of sophistication with the goal of distinguish the structural part of an object [Koppel 1988] from its data or non-compressible part of it. Sophistication is measured by the use of a special kind of Turing Machines ϕ', which separate program from data. Sophistication is then measured as $Soph(x) = min\{l(p) : \exists d \text{ such that } \phi'(p,d) = x\}$ with the restriction that *p* must be total, i.e., defined for all *d*. This last restriction precludes that the whole description is passed to the part of data, by maintaining an interpreter *i* of the data $d' = \langle p,d \rangle$. According to Koppel [Koppel 1987], "*the sophistication of an object is the size of that part of the most concise description of that object which describes its structure, i.e., the aggregate of its projectible properties. For example, the sophistication of a string that is random except that each bit is doubled (e.g. 00110000110011....) is the size of the part of the description that represents the doubling of the bits*".

Koppel showed [Koppel 1987] that "sophistication" and "depth" were equivalent up to a constant. Some problems of both sophistication and depth are discussed in the chapter 6 of this dissertation.

Appendix

B. Publications Generated from this Thesis

Ich habe nichts dagegen wenn Sie langsam denken, Herr Doktor, aber ich habe etwas dagegen wenn Sie rascher publizieren als denken.[39]
Wolfgang Pauli (1900-1958)

Many chapters of this thesis dissertation have been reflected more or less completely in several publications. Their references are included in this appendix. For a complete version of them, please visit the web page: "`http://www.dsic.upv.es/~jorallo/escrits/escritsa.htm`".

Other publications by the author (which can be found at appendix C) are also indirectly related with this work. Most of them can also be found at the same address.

Finally, an electronic version of this dissertation can be found at "`http://www.dsic.upv.es/~jorallo/tesi/tesi.htm`".

[39] I'm not scared when you think slowly, Doctor; I'm really afraid when you publish quicker than think.

Journal Papers:

- Hernández-Orallo, J. "Constructive Reinforcement Learning" *International Journal of Intelligent Systems*, Wiley, to appear.

- Hernández-Orallo, J.; García-Varea, I. "Explanatory and Creative Alternatives to the MDL Principle", *Foundations of Science*, Kluwer, to appear.

- Hernández-Orallo, J.; "A Computational Definition of 'Consilience'", *Philosophica*, to appear.

- Hernández-Orallo, J. "Beyond the Turing Test", *Journal of Logic, Language and Information*, Kluwer, submitted.

Conference Papers:

- Hernandez-Orallo, J. "Unified Information Gain Measures for Inference Processes", *5th Barcelona Logic Meeting and 6th Kurt Gödel Colloquium*, pp. 39-43, Barcelona 1999.

- Hernández-Orallo, J.; Minaya-Collado, N.: "A Formal Definition of Intelligence Based on an Intensional Variant of Kolmogorov Complexity", *Proceedings of the Intl. Symposium of Engineering of Intelligent Systems* (EIS'98), ICSC Press, pp. 146-163, 1998.

- Hernández-Orallo, J.; Garcia-Varea, I. "Distinguishing Abduction and Induction under Intensional Complexity" in Flach, P.; Kakas, A. (eds.) *European Conference of Artificial Intelligence* (ECAI'98), *Workshop on Abduction and Induction in AI*, pp. 41-48, Brighton 1998.

- Hernández-Orallo, J. "Consilience as Basis for Theory Formation", in S.Rini, G.Poletti (eds.) *Proceedings of the 1998 International Conference on Model Based Reasoning* (MBR'98), pp. 25-27, Pavia 1998.

- Hernández-Orallo, J.; Garcia-Varea, I. "On Autistic Interpretations of Occam's Razor", in S.Rini, G.Poletti (eds.) *Proceedings of the 1998 International Conference on Model Based Reasoning* (MBR'98), pp. 25-27, Pavia 1998.

Appendix

C. References

Ordenar bibliotecas es ejercer, de un modo silencioso y modesto, el arte de la crítica

Jorge Luis Borges (1899-1986), Elogio de la sombra, Junio 1968

[A.S. 1992] Artificial Stupidity, The Economist, vol. 324, no. 7770, August 1, p. 14, Editorial, 1992.

[Abe 1997] Abe, N. "Towards Realistic Theories of Learning" New Generation Computing, 15, 1997

[Adé and Denecker 1995] Ade, H.; Denecker, M. "Abductive Inductive Logic Programming" International Joint Conference of Artificial Intelligence, IJCAI-95, 1995.

[Agrawal and Srikant, 1994] Agrawal R. and Srikant R. "Fast Algorithms for Mining Association Rules" *Proc. of the 20th VLDB Conference*, Santiago de Chile, 1994.

[Aha 1997] D.W. Aha, "Lazy Learning. Editorial" Special Issue about "Lazy Learning" AI Review, v.11, 1-5, Feb. (1997).

[Aisbet and Gibbon 1998] Aisbett, J. and Gibbon, G. "Epistemic utility in commonsense reasoning", 3rd Conference in Information-Theoretic Approaches to Logic, Language and Computation, ITALLC'98, Taiwan, 1998.

[Aisbet and Gibbon 1999] Aisbett, J. and Gibbon, G. "A practical definition of the information in a logical theory", J. Experimental and Theoretical Artificial Intelligence 11, 201-217, 1999.

[Aisbet et al. 1997] Aisbett J.; Gibbon, G. And Lear, F. "On the quality of data and its effect on information usage" *Pacific Asia Conference on Information Syst PACIS'97*, 703-711, 1997.

[Akama 1992] Akama, K. "A Theory of Predicate Invention" Inductive Logic Programming'92, Muggleton, S. (ed.), Report {ICOT-TM}-1182, 1992

[Akutsu and Takasu 1994] Akutsu, T.; Takasu, A. "On PAC Learnability of Functional Dependencies" New Generation Computing, 12, 359-374, 1994.

[Alchouron et al. 1985] Alchouron, G.; Gardenfors, O.; Makinson, D. "On the logic of theory change" J. Symbolic Logic, 50, 510-530, 1985.

[Aliseda-Llera 1996] Aliseda-Llera, A. "A Unified Framework for Abductive and Inductive Reasoning in Philosophy and AI" in M. Denecker, L. De Raedt, P. Flach and T. Kakas (eds) Working Notes of the ECAI'96 Workshop on *Abductive and Inductive Reasoning*, pp. 7-9, 1996.

[Aliseda-Llera 1997] Aliseda-Llera, A. "Seeking Explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence" Department of Philosophy, Stanford Univerisity, 1997.

[Alpuente et al. 1998] Alpuente, M.; Falaschi, M.; Vidal, G. "Partial Evaluation of Functional Logic Programs" ACM Transactions on Programming Languages and Systems, 20(4):768-844, 1998.

[Anderson 1990] Anderson, J.R. "The adaptative character of thought" Hillsdale: Erlbaum 1990.

[Angluin 1980] Angluin, D. "Inductive Inference of Formal Languages from Positive Data" Information and Control 45, 117-135, 1980.

[Angluin 1988] Angluin, D., Queries and concept learning, Machine Learning 2, 4:319-342, 1988.

[Angluin and Smith 1983] Angluin, D.; Smith C.H. "Inductive inference: theory and methods, ACM Comput. Surveys 15 (3) (1983) 237-269, 1983.

[Anthony and Frisch 1997] Anthony, Simon; Frisch, Alan M. "Cautious Induction in Inductive Logic Programming" in Lavrac, Nada; Dzeroski, Saso (eds.) "Inductive Logic Programming. 7th International Workshop, ILP-97" Lecture Notes in Artificial Intelligence, Springer 1997.

[Antoniou 1997] Antoniou, G. "Nonmonotonic Reasoning: The Classical Approaches. Cambridge, MA. MIT Press, 1997.

[Arcos and Plaza 1996] Arcos, J.L. and Plaza, E. "Reflection in Noos: An object-centered representation language for knowledge modelling" *Future Generation Computer Systems*, 1996.

[Armengol and Plaza 1994] Armengol, E.; Plaza, E. "Integrateing induction in a case-based reasoner" in J.P. Haton, M. Keane, and M. Manago (eds.) *Advances in Case-Based Reasoning*, number 984 in Lecture Notes in Artificial Intelligence, pp. 3-17, Springer-Verlag, 1994.

[Baader et al. 1992] Baader, F.; Bürckert, H.J.; Heinsohn, J.; Hollunder, B.; Müller, J.; Nebel, B.; Nutt, W.; Profitlich, H. "Terminological knowledge representation: a proposal for a terminological logic", DFKI Report, DFKI, Saarbrücken 1992.

[Balasubramanian 1997] Balasubramanian, V. "Statistical Inference, Occam's Razor, and Statical Mechanics on the Space of Probability Distributions' Neural Computation 9, 349-368, 1997.

[Banerji 1984] Banerji, R.B.: Some insights into automatic programming using a pattern recognition viewpoint in Biermann, A.W.; Guiho, G.; and Kodratoff, Y. (Eds.): *Automatic program construction techniques*, Macmillan, (1984).

[Barendregt 1984] Barendregt, H.P. "The Lambda Calculus. Its syntax and semantics", North Holland, Elsevier Science Publishers B.V. 1984.

[Bar-Hillel and Carnap 1953] Bar-Hillel, Y.; Carnap, R. "Semantic Information" *British Journal for the Philosophy of Science* 4, 1953, 147-157.

[Barker 1957] Barker, S.F. *Induction and Hypothesis* Ithaca, 1957.

[Barron et al. 1998] Barron, A.; Rissanen, J.; Yu, B.: The Minimum Description Length Principle in Coding and Modeling, *IEEE Transactions on Information Theory*, Vol. 44, No. 6, 2743-2760, October (1998).

[Barto et al. 1995] Barto, A.G.; Bradtke, S.J. and Singh, P. "Learning to act using real-time dynamic programming" *Artificial Intelligence,* 72 (1-2): 81-138, 1995.

[Basili 1993] Basili, V.R.: The Experimental Paradigm in Software Engineering, in Rombach, H.D.; Basili, V.R; Selby, R. *Experimental Software Engineering Issues: Critical Assessment and Future Directives*, LNCS 706, Springer-Verlag, August (1993).

[Basili et al. 1986] Basili, V.R.; Selby, W.; Hutchens, D.H.: Experimentation in Software Engineering, *IEEE Transactions on Software Engineering*, vol. SE-12, no.7, pp. 733-743, July (1986).

[Bayes 1764] "Essay towards solving a problem in the doctrine of chances published" Philosophical Transactions of the Royal Society of London, 1764.

[Bennett 1982] Bennett, C.H. "Logical reversibility of computation" IBM J. Res. Develop., 17:525-531, 1973.

[Bennett 1988] Bennett, C.H. "Logical depth and physical complexity", in Herken, R. "The universal Turing machine: a half-century survey" Oxford University Press, 1988, 227-258, 2nd Edition 1994.

[Bennett and Gardner 1979] Bennett, C.H.; Gardner, M., The random number omega bids fair to hold the mysteries of the universe. *Scientific American*, 241:20-34, May 1979.

[Bentehtm et al. 1983] van Benthem, Johan; Doets, Kees 'High-order logic' in D.Gabbay and F.Guenthner (eds.) *Handbook of Philosophical Logic*, ch. 4, pages 275-329. Reidel, Dordrecht, 1983.

[Berry et al. 1998] Berry, D. M.; Lawrence, B.: Requirements Engineering, *IEEE Software*, March/April 1998, pp. 26-29.

[Bertalanffy 1971] Bertalanffy, L.V.: *Teoria generale dei sistemi*, Instituto Librario Internationales, Milano 1971.

[Bibel 1982] Bibel, Wolfgang "A Comparative Study of Several Proof Procedures. Artificial Intelligence 12, 269-293, 1982.

[Bibel 1987] Bibel, Wolfgang "Automated Theorem Proving" Vieweg Verlag, Vraunschweig, 2nd Edition 1987.

[Bibel 1988] Bibel, Wolfgang "Advanced Topics in Automated Deduction' in Nosum, R. (ed.) Advanced Topics in Artificial Intelligence, LNCS 345, 41-59. Berlin, Springer 1988.

[Bibel 1991] Bibel, Wolfgang 'Perspectives on Automated Deduction' in Robert S. Boyer (ed.) 'Automated Reasoning. Essays in Honor of Woody Bledsoe', Kluwer Academic Publishers 1991.

[Bibel 1993] Bibel, Wofgang "Deduction: Automated Logic" Academic Press Limited 1993.

[Biela and Borowczyk 1996] Biela, A.; Borowczyk, J. "A system that looks for axioms", Acta Inf., 33, 759-780, 1996.

[Biermann 1984] Biermann, A.W.; Guiho, G.; and Kodratoff, Y. (Eds.): *Automatic program construction technique*, Macmillan, 1984.

[Bledsoe 1977] Bledsoe, Woodrow W. "Non-resolution Theorem Proving" Artificial Intelligence 9 (1), 1977 1-35.

[Bledsoe 1979] Bledsoe, Woodrow W. "A maximal method for set variables in automatic theorem proving" in Machine Intelligence 9, pp 53-100, Ellis Horwood, Chichester 1979.

[Bledsoe 1983] Bledsoe, Woodrow W. "Using examples to generate instantiations of set variable" in proceedings of IJCAI-83, pp 892-901, Los Altos CA, Kaufmann 1983.

[Blockeel and De Raedt 1995] Blockeel, Hendrik and De Raedt, Luc "Inductive Database Design" TR. Dep. of Computer Science, Katholicke Universiteit Leuven, 1995.

[Blum 1967] Blum, M. "A machine-independent theory of the complexity of recursive functions" J. ACM 14, 4, 322-6, 1967.

[Blum and Blum 1975] Blum, L.; Blum, M. "Towards a mathematical theory of inductive inference" *Inform. and Control* 28, 125-155, 1975.

[Blumer et al. 1987] Blumer, A.; Ehrenfeucht, A.; Haussler, D.; Warmuth, M. K. "Occam's razor" Inf.Proc.Lett. 24, 377-380, 1987.

[Blumer et al. 1989] Blumer, A.; Ehrenfeucht, A.; Haussler, D.; Warmuth, M. "Learnability and the Vapnik-Chervonenkis Dimension" Journal of ACM, 36, pp. 929-965, 1989.

[Board and Pitt 1990] Board, R.; Pitt, L. "On the necessity of Occam algorithms" in Proceedings, 22nd ACM Symposium on Theory of Computing, pp. 54-63, 1990.

[Bochenski 1965] Bochenski, J.M. "The methods of contemporary thought", Dordrecht, D. Reidel 1965, Spanish Translation, "Los métodos actuales del pensamiento", Rialp, Madrid, 1968.

[Boehm 1981] Boehm, B.W.: *Software Engineering Economics,* Prentice Hall, 1981.

[Booch 1994] Booch, G., *Object-oriented Analysis and Design with Applications,* The Benjamin/Cummings Publishing Co., Inc., 1994.

[Boolos and Jeffrey 1989] Boolos, G.S.; Jeffrey, R.S. "Computability and Logic" 3rd Edition, Cambridge University Press, Cambridge 1989.

[Borgida 1996] Borgida, A. "On the relative expressiveness of description logics and predicate logics" Artificial Intelligence 82, 353-367, 1996.

[Bosch 1994] Bosch, van den, Simplicity and Prediction, Master Thesis, dep. of Science, Logic and Epistemology of the Faculty of Philosophy at the Univ. of Groningen, 1994.

[Botilier and Becher 1995] C. Botilier and V. Becher, "Abduction as belief revision" *Artificial Intelligence* 77, 43-94, (1995).

[Bouhoula et al. 1995] Bouhoula, A.; Kounalis, E.; Rusinowitch, M. "Automated Mathematical Induction" J. Logic Computation, Vol. 5, pp. 631-668, 1995.

[Bowers et al. 1997] Bowers, A.F.; Giraud-Carrier, C.; Kennedy, C.; Lloyd, J.W.; MacKinney-Romero "A Framework for High-Order Inductive Machine Learning" in "Representation issues in reasoning and learning" Area Meeting of CompulogNet Area "Computional Logic and Machine Learning" Prague, September 20, 1997.

[Brachman 1977] Brachman, R.J. "A structural paradigm for representing knowledge" Ph.D. Thesis, Division of Engineering and Applied Physics, Harvard University, Cambridge, MA, 1977.

[Brachman and Levesque 1985] Brachman, R.J.; Levesque, H.J. "Readings in Knowledge Representation" Morgan Kaufmann, Los Altos, 1985..

[Bradford and Wollowski 1995] Bradford, P.G.; Wollowski, M., A Formalization of the Turing Test (The Turing Test as an Interactive Proof System), SIGART Bulletin, Vol. 6, No. 4, p. 10, 1995.

[Brand 1996] Brand, C. "The g Factor: General Intelligence and its implications" Wiley 1996

[Bratko and Dzeroski 1995] Bratko, I.; Dzeroski, S. "Engineering Applications of ILP" New Generation Computing, 13, 313-333, 1995.

[Brent 1993] Brent, J. "Charles Sanders Pierce: A life" Indianapolis: Indiana University Press 1993.

[Brockhausen and Morik 1997] Brockhausen, P. and Morik, K. "A multistrategy approach to relational knowledge discovery in databases. *Machine Learning* Journal, Vol. 27 (3), Kluwer, 1997

[Brooks 1995] Brooks, F.P.: No Silver Bullet, in *The Mythical Man-Month: Essays on Software Engineering* (2nd ed.) Addison Wesley Longman, Reading, Mass. 1995, pp. 179-203.

[Brouwer 1907] Brouwer, L.E.J., Over de Grondslagen der Wiskunde, Doctoral Thesis, University of Amsterdam, 1907. Reprinted with additional material (D. van Dalen, ed.) by Matematisch Centrum, Amsterdam, 1981.

[Brown and Hanlon 1970] Brown, R.; Hanlon, C "Derivation Complexity and the Order of Acquisition of Child Speech" in J.R. Hayes (ed.), Cognition and the Development of Language, Wiley, New York, 11-53, 1970.

[Bundy 1983] Bundy, Alan "The Computer Modelling of Mathematical Reasoning" Academic Press, London, 1983.

[Bundy 1990] Bundy, Alan; van Harmelen, F.; Horn, C.; Smaill, A. "The Oyster-Clam system" in M.E.Stickel (ed.), *10th International Conference on Automated Deduction*, pages 647-648. Springer-Verlag, 1990. Lectures Notes in Artificial Intelligence No. 449. Also availible from Edinburgh as DAI Research Paper 507.

[Bundy 1991] Bundy, Alan "A Science of Reasoning" in Lassez, Jean-Louis; Plotkin, Gordon 'Computational Logic' THE MIT Press 1991.

[Bylander et al. 1991] Bylander, T.; Allemang, M.C.; Tanner, M.C.; Josephson, J.R. "The computational complexity of abduction" Artificial Intelligence, 49:25-60, 1991

[Carbonell 1989] Carbonell, Jaime G. "Paradigms for Machine Learning" *Artificial Intelligence*, 40: 1-9, 1989.

[Carnap 1947] Carnap, R. ``Meaning and necessity: A study in semantics and modal logic" Chicago, University of Chicago Press, 1947.

[Carnap 1950] Carnap, R. "Logical Foundations of Probability", Routledge & Kegan Paul, London, 1950.

[Carnap 1952] Carnap, R. "The Continuum of Inductive Methods" The University of Chicago Press, 1952.

[Case and Smith 1983] Case J.; Smith, C. "Comparison of identification criteria for machine inductive inference", *Theoret. Comput. Sci.* 25, 193-220, 1983.

[Chaitin 1969] Chaitin, G.J. "On the length of programs for computing finite binary sequences: statistical considerations" Journal of the ACM, 16:145-159, 1969.

[Chaitin 1974a] Chaitin, G.J., Information-Theoretic Computational Complexity, IEEE Transactions on Information Theory, vol. IT-20, no.1, pp. 10-15, january 1974.

[Chaitin 1974b] Chaitin, G.J. "Information-theoretic limitations of formal systems", Journal of the ACM, 21, 403-424., 1974.

[Chaitin 1982] Chaitin, G.J. "Gödel's Theorem and Information" Int. J. of Theoretical Physics, vol.21, no.12, pp. 941-954, 1982.

[Chaitin 1992] Chaitin, G.J. "Algorithmic Information Theory", fourth printing, Cambridge University Press, 1992.

[Chaitin 1998] Chaitin, G.J. "The limits of mathematics : A course on information theory and the limits of formal reasoning", Singapore, Springer, 1998.

[Chandrasekaran 1990] Chandrasekaran, B. "What kind of Information Processing is Intelligence?" in Partridge,D.; Wilks, Y., *Foundations of AI: A Source Book* Cambridge Univ. Press, 1990.

[Chang 1973] Chang, C.L.; Lee, R.C.T. "Symbolic Logic and Mechanical Theorem Proving" Academic Press, New York, 1973.

[Charif 1994] Charif, A. "Genetic Logic Programming for Natural Language Understanding", MSc thesis, University of Technology, Sydney, 1994.

[Charniak 1978] Charniak, E. "On the use of framed knowledge in language comprehension" Artificial Intelligence, 11 (3), 225-265, 1978.

[Charniak and Shomony 1994] Charniak, E. ; Shomony, S. "Cost-based abduction and MAP explanation" Artificial Intelligence, 66, 345-374, 1994.

[Chaudhuri and Dayal 1997] Chaudhuri, S.; Dayal, U. "An Overview of Data Warehousing and OLAP Technology" in ACM SIGMOD Record, vol. 26, pp. 65-74, 1997.

[Cheeseman 1990] Cheeseman, P. "On finding the most probable model" in Shrager, J. and Langley, P. "Computational models of scientific discovery and theory formation", chapter 3, Morgan Kaufmann.

[Chen 1983] Chen, K. "Tradeoffs in inductive inference of nearly minimal sized programs" *Inform. and Control* 52, 68-86, 1982.

[Chidamber 1994] Chidamber, S.; Kemerer, C.: A Metrics Suite for Object-Oriented Design, *IEEE T. Software Eng.*, June, pp.476-492 (1994).

[Chomsky 1966] Chomsky, N. "Cartesian Linguistics: A Chapter in the History of Rationalist Thought" Harper and Row, New York, 1966.

[Chomsky 1986a] Chomsky, N. "Knowledge of Language: its Nature, Origin, and Use" Praeger, New York, 1986a.

[Chomsky 1986b] Chomsky, N. "Barriers" MIT Press, Cambridge, Massachusetts, 1986b.

[Church 1940] Church, A. "A formulation of the simple theory of types" Journal of Symbolic Logic, 5:56-68, 1940.

[Clark 1991] Clark, K. "Logic Programming Schemes and Their Implementations" in Lassez, J.L. and Plotkin, G. *Computational Logic* The MIT Press 1991.

[Clark and Toribio 1998] Clark, A. and Toribio, J. (eds.) *Cognitive Architectures in Artificial Intelligence. The Evolution of Reserach Programs*, Garland Publishing, Inc., 1998.

[Cohen 1992] Cohen, W.W. "Abductive Explanation-Based Learning. A Solution to the Multiple Inconsistent Explanation Problem" Machine Learning, Vol. 8, pp. 167-219, 1992.

[Cohen 1994] Cohen, W.W. "Incremental Abductive EBL" Machine Learning, 15, pp. 5-24, 1994.

[Cohen and Nagel 1935] Cohen, M.R.; Nagel, E. "Introduction to logic and scientific method" in Spanish translation Cohen, M; Nagel, E. "Introducción a la lógica y al método científico" Amorrortu, Buenos Aires, 1968. Recent Re-edition: An Introduction to Logic. Indianapolis, Indiana: Hackett Publishing Company, 1993._

[Colburn 1991] Colburn, T.R.: Program Verification, Defeasible Reasoning, and Two Views of Computer Science, *Minds & Machines* 1, 97-116, 1991.

[Conklin and Witten 1994] Conklin, D.; Witten, I. H. "Complexity-Based Induction" *Machine Learning*, 16, 203-225, 1994.

[Conte 1986] Conte, S.D.; Dunsmore, H.E.; Shen, V.Y.: *Software Engineering Metrics Models*, The Benjamin/Cummings Pub. Co., 1986.

[Cox 1987] Cox, B.: *Object Oriented Programming, An Evolutionary Approach*, Addison Wesley 1987.

[Cullingford 1978] CullingFord, R. "Script Application: Computer Understanding of Newspaper Stories" Ph. D. thesis, Yale University. Computer Science Department Technical Report 116, 1978.

[Cummins 1986] Cummins, Rob "Inexplicit Information" in *The Representation of Knowledge and Belief*, eds. M.Brand and R.M. Harnish (Tuscon: University of Arizona Press 1986).

[Cussens 1998] Cussens, James "Deduction, Induction and Probabilistic Support" Synthese Journal, 1998.

[Date 1995] Date, C.J. "An Introduction to Database Systems" Addison-Wesley, 6th Edition, 1995.

[De Millo et al. 1979] De Millo, R.; Lipton, R.J, Perlis, A.J.: Social Processes and Proofs of Theorems and Programs, *Communications of the ACM* 22 (5), 271-280, (1979).

[De Raedt 1996] De Raedt, L. (ed.) "Advances in Inductive Logic Programming" IOS Press, 192-205, 1996.

[De Raedt and Bruynooghe 1993] De Raedt, L. and Bruynooghe, M. "A theory of clausal discovery" in Proc. of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93), Morgan and Kaufmann, 1993, pp. 1058-1063, 1993.

[De Raedt and Dehaspe 1997] De Raedt, L.; Dehaspe, L. "Clausal Discovery", Machine Learning , 26:99-146, 1997.

[De Raedt and Dzeroski 1994] De Raedt, L.; Dzeroski, S. "First order jk-clausal theories are PAC-learnable" *Artificial Intelligence*, 70:375-392, 1994

[DeJong 1979] DeJong, G. "Skimming Stories in Real Time: An Experiment in Integrated Understanding" Ph.D. thesis, Yale University. Computer Science Department, Technical Report 158, 1979.

[DeJong and Mooney 1986] DeJong, G.; Mooney, R. "Explanation-based learning: an alternative view" Machine Learning, 1(1), 145-176, 1986.

[DeMillo et al 1979] DeMillo, Richard A.; Lipton, Richard J.; Perlis, Alan J. "Social Processes and Proofs of Theorems and Programs" Communications of the ACM 22 (5) 1979, pp. 271-280.

[Dershowitz and Reddy 1992] Dershowitz,N.;Reddy, U.S."Deductive and Inductive Synthesis of Equat. Programs" 1992

[Derthick 1990] Derthick, M. "The Minimum Description Length Principle Applied to Feature Learning and Analogical Mapping" MCC Technical Report Number ACT-CYC-234-90, 1990.

[Deville and Kung-Kiu 1994] Deville, Y.; Kung-kiu, L. "Logic Program Synthesis" J. Logic Programming, 19,20:321-350, 1994.

[Devlin 1992] Devlin, K. *Information and Logic*, Cambridge University Press, 1992.

[Devroye 1979] L.P. Devroye and T.J. Wagner, "Distribution-free performance bounds for potential function rules" *IEEE Transactions on Information Theory*, IT-25(5):601-604, 1979.

[Dietrich 1990] Dietrich, E. "Programs in the Search for Intelligent Machines: The Mistaken Foundations of AI" in Partridge & Yorick, *Foundations of AI: A Source Book*, Cambridge Univ. Press, 1990.

[Dietterich and Flann 1997] Dietterich, T.G. and Flann, N.S. "Explanation-Based Learning and Reinforcement Learning: A Unified View" *Machine Learning*, 28, 169-210, (1997).

[Dijkstra 1972] Dijkstra, E.W.: Notes on Structured Programming, in O.Dahl *et al.* (eds.) *Structured Programming*, Academic Press 1972.

[Dimopoulous and Kakas 1995a] Dimopoulos, Yannis; Kakas, Antoni Kakas "Abduction and Learning" Departament of Computer Science, University of Cyprus, 1995.

[Dimopoulous and Kakas 1995b] Dimopoulos, Yannis; Kakas, Antoni Kakas "Learning Non-Monotonic Logic Programs: Learning Exceptions" Machine Learning: ECML-95, Proc. European Conf. on Machine Learning, 1995, N. Lavrac and S. Wrobel (eds.), LNCS 911, pp. 107-121, Springer Verlag 1995.

[Dimopoulous and Kakas 1996] Dimopoulos, Yannis; Kakas, Antoni Kakas "Learning Abductive Theories" W. Wahlster (ed.) *ECAI 96, 12th European Conference on AI.*, John Wiley & Sons Ltd. 1996.

[Donini et al. 1997] Donini, F.M.; Lenzerini, M.; Nardi, D.; Schaerf, A. "Reasoning in Description Logics" in U.Gnowho and U.Gnowho-Else (eds) "A Great Collection" CSLI Publications, 1997.

[Dowty et al. 1981] Dowty, D. R.; Wall, R. E.; Peters, S. "Introduction to Montague Semantics", Reidel, Dordrecht, 1981.

[Duc 1997] Duc, H.N. Reasoning about rational, but not logically omniscient, agents, Journal of Logic and Computation, Vol. 7, nº5, pp. 633-648, 1997.

[Duda and Hart 1973] Duda, R.; Hart, Peter "Pattern Classification and Scene Analysis" John Wiley and Sons, New York, 1973.

[Dummett 1973] Dummett, M.A.E. "The Justification of Deduction" in Proceedings of the British Academy, LIX (1973), reprinted in *Truth and Other Enigmas*, London, Duckworth, pp. 290-318, 1978.

[Dzeroski 1996] Dzeroski, S. "Inductive logic programming and knowledge discovery in databases" in Fayyad, U.; Piatetsky-Shapiro, G., Smith, P. and Uthurusamy, R. (eds.) *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, Cambridge Mass., 1996.

[Eberle 1974] Eberle, R. "A Logic of Believing, Knowing and Inferring" *Synthese*, 26, 356-382, 1974.

[Edmonds 1993] Edmonds, J.E.: Interview, FAUW Forum, University of Waterloo, January 1993.

[Eiter et al. 1997] Eiter, T.; Gottlob, G.; Leone, N. "Abduction from logic programs: Semantics and Complexity" Theoretical Computer Science, 189, 129-177, 1997.

[Elmasri and Navathe 1994] Elmasri, R.; Navathe, S. "Fundamentals of Database Systems" Addison-Wesley, 2nd Ed., 1994.

[Emde 1989] Emde, W. "An Inference Engine for Representing Multiple Theories" in K. Morik (ed.), "Knowledge Representation and Organization in Machine Learning", pp. 148-176, Springer, New York, Berlin, Tokyo, 1989.

[Epstein 1992] Epstein, R., Can Machines Think? *AI Magazine*, Vol. 12, No.2, 80-95, 1992.

[Ernis 1968] Ernis, R. "Enumerative Induction and Best Explanation" *The Journal of Philosophy*, LXV (18), 523-529, (1968).

[Etchemendy 1990] Etchemendy, J. *The Concept of Logical Consequence*, Harvard University Press, 1990.

[Evans 1963] Evans, Thomas G. "A Heuristic Program to Solve Geometric Analogy Problems" in *Semantic Information Processing*, edited by Marvin Minsky, MIT Press, Cambridge, MA, 1968, Based on a PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1963.

[Eysenck 1979] Eysenck, H.J. "The Structure and Measurement of Intelligence", Springer-Verlag 1979.

[Fagin et al. 1995] Fagin, R.; Halpern, J.Y. "Belief, awareness and limited reasoning" *Artificial Intelligence*, 34, 39-76, 1988.

[Falkenhainer 1990] Falkenhainer, B. "Abduction as similarity-driven explanation" in O'Rorke, P. (De.) Working Notes of the 1990 Spring Symposium on Anotated Abduction, pp. 135-139. AAAI. Technical Report 90-32. Department of Information and Computer Science, University of California, Irvine, 1990.

[Fallis 1996] Fallis, Don "The Source of Chaitin's Incorrectness" Philosophia Mathematica, 4, 1996.

[Fayyad et al. 1996a] Fayyad, U.; Piatetsky-Shapiro, G., Smith, P. and Uthurusamy, R. (eds.) *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, Cambridge Mass., 1996.

[Fayyad et al. 1996b] Fayyad, U.; Piatetsky-Shapiro, G., and Smith, P. "From data mining to knowledge discovery: An overview" in Fayyad, U.; Piatetsky-Shapiro, G., Smith, P. and Uthurusamy, R. (eds.) *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, Cambridge Mass., 1996.

[Fayyad et al. 1996c] Fayyad, U.; Piatetsky-Shapiro, G., and Smith, P. "The KDD Process for Extracting Useful Knowledge from Volumes of Data" Communications of the ACM, Vol. 39, No. 11, 27-34, November 1996.

[Feldman 1972] Feldman, J. Some decidability results on grammatical inference and complexity. Information and Control, 20:244-262, 1972.

[Fetzer 1988] Fetzer, J.H. "Program Verification: The Very Idea", *Communication of the ACM* 31(9), 1048-1063, (1988).

[Fetzer 1991] Fetzer, J.H.: Philosophical Aspect of Program Verification, *Minds and Machines* 1, 197-216, (1991).

[Finkelstein 1988] Finkelstein, A.: Re-use of formatted requirements specifications, *Software Engineering J. 3*, 3, 186-197, Sept. (1988).

[Flach 1993] Flach, P.A. "Predicate Invention in Inductive Data Engineering" in Pavel B. Brazdil (eds.) *Machine Learning: European Conference on Machine Learning (ECML-93)*, pages 83-94, Lecture Notes in Artificial Intelligence 667, Springer-Verlag 1993.

[Flach 1995a] Flach, Peter "Conjectures. An inquiry concerning the logic of induction", Thesis Dissertation, Proefschrift Katholieke Universiteit Brabant, Tilburg, http://www.cs.bris.ac.uk/~flach/Conjectures/

[Flach 1995b] Flach, Peter "Abduction and Induction: Syllogistic and Inferential Perspectives" INFOLAB, Tilburg University 1995.

[Flach 1996] P. Flach, "Abduction and Induction: Syllogistic and Inferential Perspectives" in M. Denecker, L. De Raedt, P. Flach and T. Kakas (eds) *Working Notes of the ECAI'96 Workshop on Abductive and Inductive Reasoning*, pp. 7-9, 1996.

[Flach and Kakas 1999] Flach, P. and Kakas, A. (eds.), *Abduction and Induction. Essays on their relation and integration*, in press, Kluwer.

[Flann and Dietterich 1989] Flann, N.S.; Dietterich, T.G. "A Study of Explanation-Based Methods for Inductive Learning" Machine Learning, Vol. 4., pp. 187-266, 1989.

[Flynn 1987] James Robert Flynn, "The ontology of intelligence" Routledge and Paul Kegan, 1987.

[Forsyth 1992] Forsyth, R.S. "Ockham's Razor as a Gardening Tool: Simplifying Discrimination Trees by Entropy Min-Max" in M. A. Bramer and R. W. Milne (eds.) Research and Development in Expert Systems, X Cambridge: Cambridge University Press, pp. 183-195, 1992.

[Fostel 1993] Fostel, G., The Turing Test is For the Birds, SIGART Bulletin, Vol. 4, No. 1, 7-8, 1993.

[Fouqué and Matwin 1993] Fouqué, G.; Matwin, S.: A case-based approach to software reuse, *J. Intelligent Inform. Systems*, 2 (2): 165-197, (1993).

[Frege 1884] Frege, G. "Die Grundlagen der Arithmetik" Wilhelm Koebner, Breslau 1884.

[Freivalds 1990] Freivalds, R. "Inductive inference of minimal size programs", in M. Fulk and J. Case (eds) "Proceedings of the third Annual Workshop on Computational Learning Theory", pp. 1-20, Morgran Kaufman, San Mateo, CA, 1990.

[Freivalds et al. 1995] Freivalds, R.; Kinber, E.; Smith, C.H. "On the Intrinsic Complexity of Learning" Inf. and Control 123, 64-71, 1995.

[Freksa 1997] Freksa, C. (de) "Foundations of Computer Science: Potential-Theory-Cognition" Lecture Notes in Computer Science, Springer, 1997.

[Fu and Buchanan 1985] Fu, L.M.; Buchanan, B. "Learning Intermediate Concepts in Constructing a Hierarchical Knowledge Base" in *Proc. 9th International Joint Conference on Artificial Intelligence*, pp. 659-666, San mateo, CA, 1985.

[Furukawa et al. 1997] Furukawa, Koichi; Murakami, Tomoko; Ueno, Ken; Ozaki, Tomonobu; Shimazu, Keiko "On a Sufficient Condition for the Existence of Most Specific Hypothesis in Progol" in Lavrac, Nada; Dzeroski, Saso (eds.) "Inductive Logic Programming. 7th International Workshop, ILP-97" Lecture Notes in Artificial Intelligence, Springer 1997.

[Gabbay et al. 1992] Gabbay, D.M., C. J. Hogger and J. A. Robinson (eds.), Handbook of Logic in Artificial Intelligence and Logic Programming. Vol. 3, Oxford: Clarendon Press, 1992.

[Garrido 1995] Garrido, Manuel "Lógica simbólica" 3rd Edition, Tecnos, 1995.

[Genesereth and Ketchpel 1994] Genesereth, M.; Ketchpel, S.P.: Software Agents. *Communications of the ACM*, 37(7):48-53, (1994).

[Gentner 1983] Gentner, D. "Structure-mapping: A theoretical framework for analogy" Cognitive Science. 7 (2), 1983.

[Gentzen 1935] Gentzen, Gerhard "Untersuchungen über das logische Schileben" Mathematische Zeitschrift, 39: 176-210 and 405-531, 1935.

[Gibbon and Aisbet 1998] Gibbon, G.; Aisbett, J. "Switching between reasoning and search" *Lecture Notes in Artificial Intelligence*; Springer-Verlag 1441, 94-108, 1998.

[Gigerenzer and Goldstein 1996] Gigerenzer, G; Goldstein, J.; "Reasoning the fast and frugal way: Models of bounded Rationality" *Psychological Review*, 103, 4, 650-669, 1996.

[Gilmore 1986] Gilmore, Paul C. 'Natural Deduction based set theories: A new resolution of old paradoxes' Journal of Symbolic Logic, 51:393-411, 1986.

[Girard 1995] Girard, Jean-Yves "Advances in Linear Logic" London Mathematical Society Lecture Note Series, No. 222, 1995.

[Girard et al. 1989] Girard, Jean-Yves; Taylor, Paul; Lafont, Yves "Proofs and Types" Cambridge Univ. Press, 1989.

[Globig et al. 1997] Globig, C.; Jantke, K.P.; Lange, Steffen; Sakakibara, Y. "On Case-Based Learnability of Languages" New Generation Computing, 15, 59-83, 1997.

[Godin and Missaoui 1994] Godin, R.; Missaoui, R. "An Incremental Concept Formation Approach for Learning From Databases", Theoretical Computer Science 133, 387-419, 1994.

[Goebel 1997] Goebel, R.G. "Abduction and its relation to constrained induction" in Peter Flach and Antonis Kakas (eds), Proceedings of the IJCAI'97 Workshop on Abduction and Induction in AI, Nagoya, Japan 1997.

[Gold 1967] Gold, E.M. "Language Identification in the Limit" *Inform. and Control.*, 10, pp. 447-474, 1967.

[Goldblatt 1987] Goldblatt, R. "Logic of Time and Computation" CSLI, Stanford, 1987.

[Goldreich 1997] Goldreich, O. "Probabilistic Proof Systems —A survey" in R. Reischuk and M. Morvan (eds.) *14th Annual Symposium on Theoretical Aspects of Computer Science* (STACS'97) in LNCS vol. 1200, pp. 595-611, Springer-Verlag, 1997.

[Goldstein et al. 1998] Goldstein, J.; Ramakrishnan, R.; Shaft, U. "Compressing Relations and Indexes" in International Conference on Data Engineering, ICDE 1998, 370-379.

[Good 1971] Good, I.J. "Twenty-seven principles of rationality" in *Foundations of Statistical Inference*, V.P. Godambe and D.A. Sprott (eds.), Toronto: Holt, Rinehart and Winston 1971.

[Goodman 1965] Goodman, N. *Fact, fiction and forecast* (2nd Edition), Indianapolis, Bobbs-Merrill, 1965.

[Green 1969] Green, C. "The Application of Theorem-Proving to Question Answering Systems" Ph. D. Thesis. Department of Electrical Engineering, Stanford University 1969

[Grégoire and Saïs 1996] Grégoire, E. and Saïs, L., Inductive reasoning is sometimes deductive, in M. Denecker, L. De Raedt, P. Flach and T. Kakas (eds) Working Notes of the ECAI'96 Workshop on *Abductive and Inductive Reasoning*, , 1996.

[Groth 1998] Groth, R., *Data Mining. A Hands-On Approach for Business Professionals*, Prentice Hall 1998.

[Grünwald 1997] P. Grünwald, "The Minimum Description Length Principle and Non-Deductive Inference" in P. Flach and A. Kakas (eds.), Proceedings of the IJCAI'97 Workshop on *Abduction and Induction in AI*, Nagoya, Japan 1997.

[Gull 1988] Gull, S.F. "Bayesian inductive inference and maximum entropy" in *Maximum Entropy and Bayesian Methods in Science and Engineering, Vol. 1: Foundations*, de. by G.J. Erickson & C.R. Smith, 53-74. Dordrecht: Kluwer 1988.

[Gunetti and Trinchero 1994] Gunetti, D.; Trinchero, U. "Intensional Learning of Logic Programs" in Franceso Bergadano and Luc de Raedt (eds) Machine Learning, Proceedings of the European Conference on Machine Learning (ECML-94), pp. 359-362, Lecture Notes in AI 784, Springer-Verlag 1994.

[Haack 1978] Haack, S., *Philosophy of Logics*, Cambridge, Cambridge University Press 1978.

[Habbermas 1972] Habbermas, J. *Knowledge and Human Interests*, London 1972.

[Hall 1989] Hall, R.P. "Computational approaches to analogical reasoning: A comparative analysis" Artificial Intelligence, 39, 39-120, 1989.

[Halpern 1997] Halpern, JY. A theoy of knowledge and ignorance for many agents, Journal of Logic and Computation, Vol. 7, nº1, pp. 79-108, 1997.

[Harel 1984] Harel, D. "Dynamic Logic" in *Gabbay, D.* and *Guenthner, F.* (eds.) "Handbook of Philosophical Logic", Vol. II., 497-604, Reidel, Dordrecht, 1984.

[Harman 1965] Harman, G. "The inference to the best explanation" *Philosophical Review*, 74, 88-95, 1965.

[Harnad 1992] Harnad, S., The Turing Test Is Not a Trick: Turing Indistinguishability Is A Scientific Criterion, , SIGART Bulletin, Vol. 3, No. 4, 9-10, October 1992,

[Harrison 1992] Harrison W.: An Entropy-Based Measure of Software Complexity, IEEE T. Software Eng. 18, No.11, 1025-34, Nov (1992)

[Harrison et. al. 1982] Harrison, W.; Magel, K.; Kluczney, R.; DeKock, A.: Software Complexity Metrics and their application to maintenance, *IEEE Computer*, pp. 65-79, Sept. 1982.

[Haussler et al. 1994] Haussler, D.; Kearns, M.; Schapire, R. "Bounds on the Sample Complexity of Bayesian Learning Using Information Theory and VC Dimension" *Machine Learning 14, 1,* pp. 88-113, January 1994.

[Heijenoort 1967] Heijenoort, J.V. (de) 'From Frege to Gödel' Harvard University Press, Cambridge Mass., 1967.

[Helft, 1989] Helft, N. "Induction as nonmonotonic inference" in Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning, pp. 149-156, 1989.

[Helmer and Oppenheim 1945] Helmer, O.; Oppeheim, P. "A syntactical definition of probability and of degree of confirmation", *Journal of Symbolic Logic* 10, 25-60, 1945.

[Hempel 1943] Hempel, C.G. "A purely syntactical definition of confirmation" J. Symbolic Logic 6 (4): 122-143; 1943.

[Hempel 1945] Hempel, C.G. "Studies in the logic of confirmation" *Mind* 54 (213): 1-25; 54(214): 97-121, 1945.

[Hempel 1965] Hempel, C.G. *Aspects of Scientific Explanation*, The Free Press, New York, N.Y. 1965.

[Hendricks and Faye 1998] Hendricks, V. F. and Faye, J., 1998, Abducting explanation, extended abstract in: MBR'98 Abstracts, S. Rini and G. Poletti, eds., complete paper, Department of Philosophy, Univ. of Copenhagen.

[Hendrix 1982] G.G.Hendrix "Computational Models of Belief and the Semantics of Belief Sentences" in S. Peters and E.Saarinen (eds.) "Processes, Beliefs, and Questions". D.Reidel Publishing Company, 1982.

[Herken 1988] Herken, R. "The universal Turing machine: a half-century survey" Oxford University Press, 1988, 2nd Edition 1994.

[Hernández-Orallo 1998a] Hernández-Orallo, J. "Reinforcement Learning in Constructive Languages", Proceedings of CCIA'98 , pp. 264-272, Tarragona, 21 - 23 october de 1998

[Hernández-Orallo 1998b] Hernández-Orallo, J. "Formalising Consilience", in S.Rini, G.Poletti (eds.) Proceedings of the 1998 International Conference on Model Based Reasoning (MBR'98), pp. 25-27, Pavia 1998.

[Hernández-Orallo 1999a] Hernández-Orallo, J. "A Computational Definition of 'Consilience'", *Philosophica*, to appear.

[Hernandez-Orallo 1999b] Hernandez-Orallo, J. "Unified Information Gain Measures for Inference Processes", *5th Barcelona Logic Meeting and 6th Kurt Gödel Colloquium*, Barcelona, pp. 39-42, 1999.

[Hernández-Orallo 1999c] Hernandez-Orallo, J., "Universal and Cognitive Notions of 'Part'", to be presented at the *4th European Congress on Systems Science (ESS'99)*, 1999.

[Hernández-Orallo 1999d] Hernández-Orallo, J.: Constructive Reinforcement Learning, *Intl. J. of Intelligent Systems, to appear*. URL: http://www.dsic.upv.es/~jorallo/escritsa/IJISHern.ps.gz

[Hernández-Orallo 1999e] Hernández-Orallo, J, Unified Information Measures for Inference Processes, *Collegium Logicum* - Annals of the Kurt-Gödel-Society Vol. 4, Springer Verlag Wien, to appear.

[Hernández-Orallo 1999f] Hernández-Orallo, J. "Beyond the Turing Test", *Journal of Logic, Language and Information*, submitted.

[Hernández-Orallo and Alamagnac 1999] Hernández-Orallo, J.; Alamagnac, F. "Data Quality for Data-Mining" to be presented at the *4th European Congress on Systems Science (ESS'99)*, 1999.

[Hernández-Orallo and García Varea 1998b] J. Hernández-Orallo and I. García-Varea, "On Autistic Interpretations of Occam's Razor", in S.Rini, G.Poletti (eds.) Proceedings of the 1998 International Conference on Model Based Reasoning (MBR'98), pp. 25-27, Pavia 1998.

[Hernández-Orallo and García-Varea 1998a] J. Hernández-Orallo, J. and García-Varea, I. "Distinguishing Abduction and Induction under Intensional Complexity" in P. Flach and A. Kakas (eds.) *Proc. of the European Conference of Artificial Intelligence (ECAI'98) Ws. on Abduction and Induction in AI*, pp. 41-48, Brighton 1998.

[Hernández-Orallo and García-Varea 1999] Hernández-Orallo, J.; García-Varea, I. "Explanatory and Creative Alternatives to the MDL Principle", *Foundations of Science*, Kluwer, to appear,

[Hernández-Orallo and Hernández-Orallo 1993] Hernández-Orallo, J.; Hernández-Orallo, E.: *Programación en C++*, Paraninfo 1993, 2nd Ed., Intl. Thompson Publishers, 1995.

[Hernández-Orallo and Minaya-Collado 1998] Hernández-Orallo, J. "A Formal Definition of Intelligence based on an Intensional Variant of Algorithmic Complexity" Proceedings of Engineering of Intelligent Systems (EIS98), ICSC Academic Press 1998 .

[Hernández-Orallo and Pinto 1996a] Hernandez-Orallo, J.; Pinto, J. "Viabilidad de un Modelo de Conocimiento Falible en Cálculo de Situaciones" Departamento de Computación, Pontificia Universidad Católica de Chile, August 1996.

[Hernández-Orallo and Pinto 1996b] Hernandez-Orallo, J.; Pinto, J. "Especificación Formal de Protocolos Criptográficos en Cálculo de Situaciones" Departamento de Computación, Pontificia Universidad Católica de Chile, August 1996, to be published in Novatica, to appear 1999.

[Hernández-Orallo and Ramírez-Quintana 1998a] Hernández-Orallo, J. and Ramírez-Quintana, M.J. "Inductive Inference of Functional Logic Programs by Inverse Narrowing" J. Lloyd (ed) *Proc. JICSLP'98 CompulogNet Meeting on Comp. Logic & Machine Learning*, pp. 49-55, 1998.

[Hernández-Orallo and Ramírez-Quintana 1998b] Hernández-Orallo, J.; Ramírez-Quintana, M.J.: Inverse Narrowing for the Inductive Inference of Functional Logic Programs, in Freire-Nistal, J.L.; Falaschi, M.; Vilares-Ferro, M. (eds) Proc. 1998 Joint Conference of Declarative Programming., pp.379-392, 1998.

[Hernández-Orallo and Ramírez-Quintana 1999a] Hernández-Orallo, J. and Ramírez-Quintana, M.J. "A Strong Complete Schema for Inductive Functional Logic Programming", in Flach, P.; and Dzeroski, S. *Inductive Logic Programming'99* (ILP'99), in the Volume 1634 of the Lecture Notes in Artificial Intelligence (LNAI) series, Springer-Verlag 1999.

[Hernández-Orallo and Ramírez-Quintana 1999b] Hernández-Orallo, J. and Ramírez-Quintana, M.J. "Inductive Functional Logic Programming", 8th International Workshop on Functional and Logic Programming (WFLP'99), Grenoble, France, 28-30, June 1999.

[Hesse 1974] Hesse, M., *The Structure of Scientific Inference*, MacMillan, London, 1974.

[Heyting 1930] Heyting, A., "Die formalen Regeln der intuitionistischen Logik", Sitzungsber. Preuss. Akad. Wiss. Berlin, 42-56, 1930.

[Hintikka 1962] Hintikka, J. *Knowledge and Belief.* Cornell University Press, Ithaca, NY, 1962.

[Hintikka 1964] Hintikka, J. Towards a theory of inductive generalization, in Proceedings of the 1964 International Congress for Logic, Methodology and Philosophy of Science, pp. 274-288, 1964.

[Hintikka 1970a] Hintikka, J. "On Semantic Information" in Hintikka, J.; Suppes, P. (eds.) D.Reidel Publishing Company, pp. 3-27, 1970.

[Hintikka 1970b] Hintikka, J. "Surface Information and Depth Information" in Hintikka, J.; Suppes, P. (eds.) D.Reidel Publishing Company, pp. 263-297, 1970.

[Hintikka 1973] Hintikka,J. "Logic, Language-Games and Information" The Calrendon Press, Oxford Univ. 1973.

[Hintikka 1988] Hintikka, J. "Model Minimization - An Alternative to Circumscription" Journal of Automated Reasoning 4(1): 1-13, 1988.

[Hintikka 1996] Hintikka, J. "What is abduction? The fundamental problem about ampliative inference", International Congress on Discovery and Creativity, Ghent (Belgium), May 1998, http://allserv.rug.ac.be/ ~jmeheus/

[Hintikka 1996] Hintikka, J. "Principles Of Mathematics Revisited" Cambridge University Press, 1996.

[Hintikka and Suppes 1970] Hintikka, Jaakko; Suppes, Patrick "Surface Information and Depth Information" in Hintikka, Jaakko; Suppes, Patrick "Information and Inference", D. Reidel Publishing Company 1970.

[Hintikka and Tuomela 1970] Hintikka, J.; Tuomela, R. "Towards a General Theory of Auxiliary Concepts and Definability in First-Order Theories" in Hintikka, J.; Suppes, P. (eds.) D.Reidel Publishing Company, pp. 298-330, 1970.

[Hinton and Zemel 1994] Hinton, G.; Zemel, R. "Autoencoders, minimum description length, and Helmholtz free energy" in Cowan, J., Tesauro, G., and Alspector, J. (eds.) *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann Publishers, San Francisco, CA., 1994.

[Hoaglin et al. 1983] Hoaglin, D., Mosteller, F., and Tukey, J. *Understanding Robust and Exploratory Data Analysis*, Wiley, New York, 1983.

[Hoare 1969] Hoare, C.A.R.: An Axiomatic Basis for Computer Programming, *Communications of the ACM* 12, 576-580, 583, (1969).

[Hobbs et al. 1993] Hobbs, J.; Stickel, M.; Appelt, D.; Martin, P. "Interpretation as abduction" Artificial Intelligence, 63 81-2), 69-143.

[Hofstadter 1979] Hofstadter, D.R. "Gödel, Escher, Bach: An eternal golden braid" New York: Basic Books, 1979.

[Hofstadter 1985] Hoftadter, D.R. "Metamagical Themas. Questing for the Essence of Mind and Pattern" Basic Books, Inc., 1985.

[Hofstadter et al. 1995] Hofstadter, D.R.; Fluid Analogies Research Group "Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought" Basic Books, 1995.

[Holland et al. 1989] Holland, J.H.; Holyoak, K.J.; Nisbett, R.E.; Thagard, P.R. "Induction. Processes of Inference, Learning, and Discovery" The MIT Press 1989.

[Hörmann 1981] Hörmann, H. *To Mean — To Understand: Problems of Psycho-logical Semantics*, Springer-Verlag, 1981.

[Horvitz 1990] Horvitz, E. "Computation and Action Under Bounded Resources" PhD Dissertation, Stanford University, 1990.

[Howard 1966] Howard, R.A. "Information value theory" *IEEE Transactions on Systems Science and Cybernetics*, SSC-2 (1): 22-26, 1966.

[Ichise 1998] Ichise, R. "Synthesizing Inductive Logic Programming and Genetic Programming", in H. Prade (ed.) *13th European Conference on Artificial Intelligence*, John Wiley & Sons, pp. 465-468, 1998.

[Ifrah 1994] Ifrah, G. "Histoire Universelle des Chiffres" Editions Robert Laffont, S.A., Paris, 1994.

[Indurkhya 1991] Indurkhya, B. "On the role of Interpretative Analogy in Learning" New Generation Computing, 8, 385-402, 1991.

[Ireland et al. 1996] Ireland, Andrew; Bundy, Alan 'Productive Use of Failure in Inductive Proof' Journal of Automated Reasoning 16: 79-111, 1996.

[Jacobson 1995] Jacobson, I.: The Use-Case Construct in Object-Oriented Software Engineering, *Scenario-Based Design: Envisioning Work and Technology in System Development*, J.Carroll, ed., John Wiley & Sons, New York, 1995, pp. 309-336.

[Jaeger 1998] Jaeger, M. "Probability Logics" course notes for "Formal Systems for Probabilistic Inference, Part 1", ESSLLI-98, Saarbrücken, 1998.

[Jain 1995] Jain, Sanjay "On a question about learning minimal programs" Information Processing Letters 53, 1-4, 1995.

[Jeffrey 1984] Jeffrey, R.C. "The Impossibility of Inductive Probability" Nature 310, 434, 1984.

[Jevons 1874] Jevons, W.S. "The Principles of Science: A Treatise on Logic and Scientific Method" Macmillian, London, 1874.

[Johnson 1992] Johnson, W.L., Needed: A New Test of Intelligence, SIGART Bulletin, Vol. 3, No. 4, 7-9, October 1992, Editorial and Commentary.

[Josephson and Josephson 1994] Josephson, John R.; Josephson, Susan G. "Abductive Inference. Computation, Philosophy, Technology" Cambridge University Press, New York 1994.

[Juedes et. al 1993] Juedess, D.W.; Lathrop; J.I.; Lutz, J.H. "Computational depth and reducibility" Proc. 20th Int. Colloq. Automata, Languagess, Prog., LNCS, Springer-Verlag, 1993.

[Just and Carpenter 1987] Just, M.A.; Carpenter, P.A. *The psychology of reading and language comprehension*, Allyn & Bacon, Boston, 1987.

[Kaelbling et al. 1996] Kaelbling, L.; Littman, M.; Moore, A.: Reinforcement Learning: A survey, *J. of AI Research*, 4, 237-285, (1996).

[Kakas and Mancarella 1990] A.C. Kakas and P. Mancarella "On the relation between truth maintenance and abduction" In Proceedings of the "nd Pacific Rim International Conference on Artificial Intelligence, 1990.

[Kakas et al. 1993] Kakas, A.C.; Kowalski, A.; Toni, F. "Abductive Logic Programming" *J. of Logic and Comp.*, 2 (6): 719-770, 1993.

[Kantola et al. 1992]Kantola, M.; Mannila, H.; Räihä, K.J.; and Siirtola, H. "Discovering Functional and Inclusion Dependencies in Relational Databases" *International Journal of Intelligent Systems*, 7, pp. 591-607, 1992

[Karmiloff-Smith 1992] Karmiloff-Smith, A., Beyond Modularity: A Developmental Prespective on Cognitive Science, The MIT Press 1992.

[Kass 1986] Kass, A. "Modifying explanations to understand stories" in Proceedings of the Eighth Annual Conference of the Cognitive Science Society, pp. 691-696, Amherst, MA. Cognitive Science Society.1986.

[Kass 1990] Kass, A. "Developing Creative Hypotheses in Adapting Explanations" Ph. D. thesis, Yale University, Northwestern University Institute for the Learning Sciences, Technical Report 6, 1990.

[Kearns 1990] Kearns, Michael J. "The Computational Complexity of Machine Learning". The MIT Press 1990.

[Kearns 1992] Kearns "Oblivious PAC Learning of Concept Hierarchies" AAAI 1992 nº10.

[Kearns and Singh 1999?] M. Kearns and S. Singh, "Near-Optimal Performance for Reinforcement Learning in Polynomial Time" URL: http://www.research.att.com/~mkearns/

[Kearns et al. 1999] Kearns, M.; Mansour, Y.; Ng, A.Y.; Ron, D.: An experimental and theoretical comparison of model selection methods, *Machine Learning*, to appear. URL: http://www.research.att.com/~mkearns/

[Kemeny 1953] Kemeny, J., "A logical measure function" *Journal of Symbolic Logic*, 18, 289-308, 1953.

[Keynes 1921] Keynes, J.M., *A Treatise on Probability*, Macmillan, London 1921.

[Kieffer and Yang 1996] Kieffer, J.C.; Yang, E. "Sequential Codes, Lossless Compression of Individual Sequences, and Kolmogorov Complexity" IEEE Trans. on Inf. Theory, vol. 42, no. 1, Jan. 1996.

[King et al. 1995] King, R.D.; Sternberg, J.F.; Srinivasan, A. "Relating Chemical Activity to Structure: An Examination of ILP Successes" New Generation Computing, 13, 411-433, 1995.

[Kintsch and Keenan 1973] Kintsch, W.; Keenan, J. "Reading rate and retention as a function of the number of propositions in the base structure of sentences" *Cognitive Psychology*, 5:257-274, 1973.

[Kirsh 1990] Kirsh, David "When Is Information Explicitly Represented?" in *Information, Language, and Cognition*, edited by Philip P. Hanson, Vancouver, University of British Columbia Press, 1990, 340-65.

[Kline 1980] Kline, M "Mathematics. The loss of certainty" New York, Oxford University Press 1980.

[Kling 1971] Kling, R.E "A paradigm for reasoning by analogy" Artificial Intelligence, 2:147-178, 1971.

[Kneale and Kneale 1984] Kneale, W. and Kneale, M. "The Development of Logic" Clarendon Press, Oxford, 1984.

[Kodratoff 1994] Kodratoff, Y. Guest Editor's Introduction "The Comprehensibility Manifesto" *AI Communications*, 7(2): 83-85, 1994.

[Kolmogorov 1965] Kolmogorov, A.N. "Three Approaches to the Quantitative Definition of Information" Problems Inform. Transmission, 1(1):1-7, 1965.

[Kolmogorov 1968] Kolmogorov, A.N., Logical basis for information theory and probability theory, *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 662-664, sept. 1968.

[Konolige 1986] Konolige, K. "A Deduction Model of Belief" Pitman Publishing, London, 1986.

[Konolige 1991] Konolige, K. "Abduction versus closure in causal theories" Artificial Intelligence, 52:255-72, 1991.

[Konolige 1992] Konolige, K "Autoepistemic Logic" in [Gabbay et. al. 1992], pp. 217-295.

[Koppel 1987] Koppel, M., Complexity, Depth, and Sophistication, *Complex Systems* 1, 1087-1091, 1987.

[Koppel 1988] Koppel, M. "Structure", in Herken, R. "The universal Turing machine: a half-century survey" Oxford University Press, 1988, pp. 435-452, 2nd Edition 1994.

[Kotovsky and Simon 1990] Kotovsky, K.; Simon, H.A. "Why are some problems really hard: explorations in the problem space of difficulty". Cognitive Psychology, 22, 143-183 1990.

[Kowalski 1974] Kowalski, R.A. 'Predicate Logic as a Programming Language', Proc. of the 6th IFIP Congress, 569-574, North-Holland, 1974.

[Kowalski 1975] Kowalski, R.A. 'A Proof Procedure using Connection Graphs', Journal of ACM 22, 4, 572-595, 1975.

[Kowalski 1979] Kowalski, Robert A. 'Logic for Problem Solving', North-Holland, New York, 1979.

[Kowalski and Sachi 1997] R. Kowalski and F. Sachi "Reconciling the Event Calculus with the Situation Calculus" *J.Logic Prog.* 31(1-3), 39-58, (1997).

[Kowalski et al 1971] Kowalski, R.A., Kuehner, D. 'Linear resolution with selection function' Artificial Intelligence 2, 227-260, 1971.

[Koza 1993] Koza, J.R. "Genetic Programming on the programming of computers by means of natural selection" The MIT Press, Cambridge MA, 1993.

[Kramer 1995] Kramer, S. "Predicate Invention: A Comprehensive View". Report TR-95-32, Austrian Research Institute for Artificial Intelligence, Vienna, 1995.

[Kripke 1963] Kripke, S. "A Semantical Analysis of Modal Logic I: nomral modal prop. calculi" Zeitschrift für Math, Logik und Grundlagen Mathematik 9, pp. 67-96, 1963.

[Kuhn 1970] Kuhn, T.S.: *The Structure of Scientific Revolution*, University of Chicago 1970.

[Lakatos 1976] Lakatos,I. "Proofs and Refutations. The Logic of Mathematical Discovery" Cambridge. Univ. Prs., 1976.

[Lakatos 1979] Lakatos, I. "What Does a Mathematical Proof Prove?" in "Mathematics, Science and Epistemology" Cambridge University Press, 1979, pp.540-551.

[Lange and Zeugmann 1995] Lange, S.; Zeugmann, Thomas "Refined Incremental Learning" 1995.

[Langley and Simon 1995] Langley, P., and Simon, H.A. "Applications of machine learning and rule induction" *Commun. ACM 38*, 11 (Nov. 1995), 55-64, 1995.

[Larsson 1993] Larsson, J.E., The Turing Test Misunderstood, SIGART Bulletin, Vol. 4, No. 4, p. 10, 1993.

[Lavrac and Dzeroski 1994] Lavrac, N.; Dzeroski, S. "Inductive Logic Programming: Techniques and Applications" Ellis Horwood, 1994.

[Lavrac and Dzeroski 1997] Lavrac, Nada; Dzeroski, Saso (eds.) Inductive Logic Programming. 7th International Workshop, ILP-97" Lecture Notes in Artificial Intelligence, Springer 1997

[Leake 1992] Leake, David B. "Evaluating Explanation: A Content Theory" Lawrence Erlbaum Associates, Hillsdale, NJ, 1992.

[Leake 1993] Leake, David B. "Focusing construction and selection of abductive hypotheses" in Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp. 24-29, Chambery, France, IJCAI, 1993.

[Leake 1994a] Leake, David B. "ACCEPTER: evaluating explanations" in Schank, R.; Riesbeck, C.; Kass, A. "Inside Case-Based Explanation" chap. 6, pp. 167-207. Lawrence Erlbaum Associates 1994.

[Leake 1995] Leake, David B "Abduction, Experience, and Goals: A Model of Everyday Abductive Explanation" The Journal of Experimental and Theoretical Artificial Intelligence 1995.

[Leake and Owens 1986] Leake, David B.; Owens, C. "Organizing Memory for Explanations" in Proceedings of the Eighth Annual Conference of the Cognitive Science Society, pp. 710-715, Amherst, M.A. Cognitive Science Society, 1986.

[Lebowitz 1980] Lebowitz, M. "Generalisation and Memory in an Integrated Understanding System" Ph. D. thesis, Yale University. Computer Science Department. Technical Report 186, 1980.

[Leivant 1994] Leivant, Daniel 'Higher Order Logic' in Gabbay, Hogger & Robinson (eds.) Handbook of Logic in AI and LP, Volume 2, Deduction Methodologies, Clarendon Press, Oxford 1994.

[Lempel and Ziv 1977] Lempel, A.; Ziv, J. "A Universal Algorithm for Sequential Data Compression" in *IEEE Transactions on Information Theory*, Vol. 31, No. 3, pp. 337-343, 1977.

[Lesgold 1972] Lesgold, A.M. "Pronominalization: a Device for Unifying Sentences in Memory" *Jouranl of Verbal Learning and Verbal Behavior*, 11:316-323, 1972.

[Levesque 1984] Levesque, Hector J. "A logic of implicit and explicit belief" in *Proceedings AAAI-84*, pp. 198-202, Austin, TX, 1984.

[Levesque 1996] Levesque, Hector J. "What is planning in the presence of sensing" American Association for Artificial Intelligence 1996.

[Levi 1973] Levi, I. *Gambling with truth* MIT Press Paperback Edition, 1973.

[Levi 1980] Levi, I. *The Enterprise of Knowledge* MIT Press, 1980.

[Levin 1973] Levin, L.A. "Universal search problems" *Problems Information Transmission*, 9:265-266, 1973.

[Levinson 1973] R. Levinson "General game-playing and reinforcement learning" *Computational Intelligence*, 12(1): 155-176, (1996).

[Li and Vitányi 1996] Li, M. and Vitányi, P. "Reversibility and adiabatic computation: trading time and space for energy" Proc. Royal Society of London, Series A, 452 (1996), 760-789.

[Li and Vitányi 1997] Li, M.; Vitányi, P. "An Introduction to Kolmogorov Complexity and its Applications" 2nd Ed. Springer-Verlag 1997.

[Lieberherr 1996] Lieberherr K.J.: *Adaptive Object-Oriented Software: The Demeter Method with Propagation Patterns*, PWS Pub. Co, 1996.

[Ling 1991] Ling, Charles. "Inventing Necessary Theoretical Terms in Scientific Discovery and Inductive Logic Programming" TR 302, Department of Computer Science, University of Western Ontario, 1991.

[Ling and Narayan 91] Ling, Charles; Narayan M.A. "A Critical Comparison of Various Methods Based on Inverse Resolution, in Birnbaum L.A. & Collins G.C. (eds.), Machine Learning: Proceedings of the Eighth International Workshop (ML91), Morgan Kaufmann, San Mateo, CA, pp. 168-172, 1991.

[Lloyd 1987] Lloyd, J. "Foundations of logic programming" Springer-Verlag, 2nd edition, 1987.

[Lloyd 1995] Lloyd, J.W. "Declarative programming in Escher" Technical REport CSTR-95-013, Department of Computer Science, University of Bristol, 1995. Also available at http://www.cs.bris.ac.uk/

[López de Mántaras and Armengol 1998] R. López de Mántaras and E. Armengol, "Machine Learning from examples: Inductive and Lazy Methods" *Data and Knowledge Engineering* 25, 99-123, (1998).

[Lorenz and Kidd 1994] Lorenz, M.; Kidd, J.: *Object-Oriented Software Metrics*, Prentice Hall Publishing, 1994.

[Loucopoulos and Karakostas 1995] Loucopoulos, P.; Karakostas, V.: *System Requirements Engineering*, McGraw-Hill, New York, 1995.

[Loveland 1978] Loveland Donald W. 'Automated Theorem Proving: A Logical Basis, North-Holland, Amsterdam, 1978, 405 pp.

[Loveland 1984] Loveland, D.W. 'Automated Theorem Proving: A Quarter Century Review' Contemporary Mathematics, Vol. 29, 1984, pp. 1-45.

[Lozinskii 1994] Lozinskii, E. "Information and Evidence in Logic Systems" *Journal of Experimental and Theoretical Artificial Intelligence* 6, 163-193, 1994.

[Lucas 1962] Lucas, J.R. "Minds, Machines, and Gödel" Reprinted in Anderson, A. (ed) "Minds and Machines", Prentice Hall, 1962.

[Lukasiewicz 1970] L. Borowski, editor. Selected Works of Jan Lukasiewicz. North-Holland Publishing Co., Warsaw, 1970.

[Mackay 1969] Mackay, D. *Information, mechanism and meaning* (MIT Press), 1969.

[MacKenzie 1995] MacKenzie, Donald 'The Automation of Proof: A Historical and Sociological Exploration' IEEE Annals of the History of Computing 1995.

[Maddox 1993] Maddox, H. *Theory of knowledge*, Cambridge University Press 1993.

[Maes 1995] Maes, P.: Intelligent Software, *Scientific American* 273 (3), September, (1995).

[Mandler 1967] Mandler, G. "Organisation and Memory" in K.W. Spence and J.T. Spence (eds.) *The Psychology of Learning and Motivation*, vol. 2, pp. 189-196, Academic Press, New York, 1967.

[Mannila and Räihä 1994] Mannila H.; Räihä, K.J. "Algorithms for Inferring Functional Dependencies From Relations" *Data & Knowledge Engineering*, 12, 83-99, 1994.

[Mannila et al. 1994] Mannila H.; Toivonen H., and Verkamo A.I. "Efficient Algorithms for Discovering Association Rules" Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases, 1994.

[Marcus et al. 1999] Marcus, G.F.; Vijayan, S.; Bandi Rao, S.; Vishton, P.M. "Rule Learning by Seven-Month-Old Infants" Science, January 1999, pp. 77-80

[Marek and Truszczynski 1993] Marek, V.; Truszczynski, M. "Nonmonotonic Logic", Springer-Verlag, 1993.

[Martin 1977] Martin, D.A. "Descriptive Set Theory: Projective Sets" in J.Barwise (ed.) *Handbook of mathematical logic*, Elsevier Science Publishers, 1977.

[Martín 1998] Martín, M. "Reinforcement learning for embedded agents facing complex tasks" Thesis Dissertation, Universitat Politècnica de Catalunya.

[Martinich 1985] Martinich, A.P. "The philosophy of language", Oxford University Press, New York and London, 1985.

[Martin-Löf 1982] Martin-Löf, Per 'Constructive mathematics and computer programming' in Logtic, Methodology and Philoshophy of Scinece, vol. IV, pages 153-175. North-Holland, Amsterdam, 1982.

[Martin-Löf 1984] Martin-Löf, Per 'Intuitionistic Type Theory' Studies in Proof Theory Lecture Notes. Bibliopolis, Napoli, 1984.

[Maslov 1987] Maslov, S. Yu "Theory of Deductive System and Its Applications" The MIT Press 1987

[Matheus and Rendell 1989] Matheus C.J., Rendell L.A.: "Constructive Induction On Decision Trees" in Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), Morgan Kaufmann; Los Altos, CA, 645-650, 1989.

[McCarthy 1968] J. McCarthy "Situations, actions and causal laws" TR, Stanford University 1963. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pp. 410-417.

[Merhav and Feder 1998] Merhav, N.; Feder, M.: Universal Prediction, *IEEE Transactions on Information Theory*, Vol. 44, No. 6, 2124-2147, October (1998).

[Michalski 1987] Michalski, R.S. "Concept Learning" in S.C. Shapiro (ed). "Encyclopedia of Artificial Intelligence" 185-194, John Wiley, Chicester, 1987.

[Michalski 1993] Michalski, R.S. "Inferential Theory of Learning as a Conceptual Basis for Multistrategy Learning" Machine Learning, 11, 111-151, 1993.

[Mill 1843] Mill, J.S. *A System of Logic*, 1843.

[Millican and Clark 1996] Millican, P.J.R. and Clark, A. (eds.) *Machines and Thought. The Legacy of Alan Turing*, Vol. I, Clarendon Press, Oxford, 1996.

[Minsky 1975] Minsky, M. "A framework for representing knowledge" in Winston P. (ed) The Psychology of Computer Vision, chap. 6, pp. 211-277. McGraw-Hill, New York, 1975.

[Mitchell 1997] Mitchell, T.M.: *Machine Learning*, McGraw-Hill Series in Computer Science, 1997.

[Mitchell and Kedar-Cabelli 1986] Mitchell, T. & Kedar-Cabelli, S. "Explanation-based Learning. A Unifying View" Machine Learning, Vol. 1., pp. 47-80, 1986.

[Mitchell et al. 1991] Mitchell, T.M.; Allen, J, Chalasani, P.; Cheng, J. Etzioni, O.; Ringuette, M; Schlimmer, J. "THEO: A framework for self-improving systems" in Kurt VanLehn, editor, Architectures for Intelligence, pages 323-356, Lawrence Erlaum Associates, 1991.

[Mizoguchi and Ohwada 1995] Mizoguchi, F.; Ohwada, H. "Constrained Relative Least General Generalization for Inducing Constraint Logic Programs" New Generation Computing, 13, 335-368, 1995.

[Moffat and Zobel 1992] Moffat, A.; Zobel, J. "Compression and Fast Indexing for Multi-Gigabyte Text Databases" First Australian Workshop on Information Retrieval, Monash University, November 1992.

[Moody 1993] Moody, T.C., *Philosophy and Artificial Intelligence*, Englewood Cliffs, NJ, Prentice Hall, 1993.

[Mooney 1990] Mooney, R. "A General Explanation-based Learning Mechanism and its Application to Narrative Understanding" Morgan Kaufmann Publishers, Inc., San Mateo. 1990

[Mooney 1997] Mooney, R.J. "Integrating Abduction and Induction in Machine Learning" in Peter Flach and Antonis Kakas (eds), *Proceedings of the IJCAI'97 Workshop on Abduction and Induction in AI*, Nagoya, Japan 1997.

[Moore 1984] Moore, Robert C. "Possible-World Semantics for Autoepistemic Logic" Proceedings Non-Monotoning Reasoning Workshop, New Paltz, New York, 1984.

[Moore 1985a] Moore, Robert C. "A formal theory of knowledge and action" in Hobbs, J.R. and Moore, R.C. editors 1985, *Formal Theories of the Commonsense World* Ablex, Norwood, NJ. 319-358.

[Moore 1985b] Moore, Robert C. "Semantical Considerations on Nonmonotonic Logic" Artificial Intelligence, vol. 25, no.1, 1985.

[Moore 1993] Moore, Robert C. "Autoepistemic Logic Revisited" Artificial Intelligence, vol. 59, nos.1-2, 1993.

[Moravec 1998] Moravec, H., *ROBOT: Mere Machine to Transcendent Mind*, Oxford Univ. Press, 1998.

[Moreno 1998] Moreno, A. "Avoiding logical omniscience and perfect reasoning: a survey" AI Communications 2, 1998.

[Moreno and Sales 1997] Moreno, A., Sales, T. "Dynamic belief analysis" in Intelligent Agents III: Agent theories, architectures and languages, LNAI 1193, Springer Verlag, 87-102, 1997.

[Morik 1997] Morik, K. "Knowledge Discovery in Databases – An Inductive Logic Programming Approach" in C. Freksa, M. Jantzen, R. Valk (eds.) *Foundations of Computer Science*: *Potential-Theory-Cognition*, LNCS 1337, Springer, 1997.

[Morik et al. 1993] Morik, K.; Wrobel, S.; Kietz, J.; Emde, W. *Knowledge Acquisition and Machine Learning: Theory, Methods and Applications*, Academic Press, 1993.

[Muggleton and Buntime 1988] Muggleton, S. and Buntime, W. "Machine invention of first-order predicates by inverting resolution" *Fifth International Conference on Machine Learning.* Morgan Kaufmann, 1988.

[Muggleton 1984] Muggleton, S. "Induction of Regular Languages from Positive Examples" Tech. Rep, Turing Institute Research Memoranda, Glasgow, 1984.

[Muggleton 1991] Muggleton, S. "Inductive Logic Programming" *New Generation Computing, 8, 4*, pp. 295-318, 1991.

[Muggleton 1992] Muggleton, S. "A Theoretical Framework for predicate invention", The Turing Institute 1992.

[Muggleton 1994a] Muggleton, S. "Predicate Invention and Utility" *Journal of Experimental and theoretical Artificial Intelligence* 6 (1): 127-130, 1994.

[Muggleton 1994b] Muggleton, S. "Bayesian Inductive Logic Programming" in Cohen, W; Hirsh, H. (eds.) *Proceedings of the Eleventh International Machine Learning Conference* , San Mateo, CA, Morgan-Kaufmann, 371-379, 1994.

[Muggleton 1995a] Muggleton, S. "Inverse Entailment and Progol" New Generation Computing Journal 13:245:286, 1995.

[Muggleton 1995b] Muggleton, S. "Mode-Directed Inverse Resolution" in Kurukawa, K.; Michie, D.; Muggleton, S. (eds.) Machine Intelligence 14, Oxford University Press, 1995.

[Muggleton 1998] Muggleton, S., "Inductive logic programming: issues, results and the LLL challenge" in H. Prade, editor, Proceedings of ECAI98, page 697. John Wiley, 1998.

[Muggleton and De Raedt 1994] Muggleton, S. & De Raedt L. "Inductive Logic Programming — theory and methods" Journal of Logic Programming, 19-20:629-679, 1994.

[Muggleton and Feng 1990] Muggleton, S.; Feng, C. "Efficient Induction of Logic Programs" in Arikawa, S.; Goto, S.; Ohsuga, S.; Yokmori, T. (eds.) Proc. 1st Conf. on Algorithmic Learning Theory, Japanese Society for Artificial Intelligence, Tokyo, 1990.

[Muggleton and Page 1994] Muggleton, S.; Page, D. "A Learnability Model for Universal Representations" *Technical Report*, PRG-TR-3-94, Oxford University Computing Laboratory, Oxford 1994. URL: http://www.cs.york.ac.uk/~stephen/jnl.html

[Muggleton et al. 1992] Muggleton, S.; Srinivasan, A.; Bain, M. "Compression, significance and accuracy" in D. Sleeman and P. Edwards (eds.) *Machine Learning: Proceedings of the Ninth International Conference (ML92]*, pages 523-527, Wiley 1992.

[Muggleton et al. 1995] Muggleton, S.; Mizoguchi, F.; Furukawa, K. "Special Issue on Inductive Logic Programming" New Generation Computing, 13, 243-244, 1995.

[Mura 1990] Mura, A. "When Probabilistic Support is Inductive" Philosophy of Science 57, 278-289, 1990.

[Myhill 1958] Myhill, J. "Problems arising in the formalization of intensional logic" *Logique et Analyse*, vol. 1, pp.75-83, 1958..

[Nake 1974] Nake F. "*Ästhetik als Informationsverarbeitung*" Springer 1974.

[Neisser et al. 1996] Neisser, U.; Boodoo, G.; Bouchard, T.J., Jr., Boykin, A.W., Brody, N., Ceci, S.J. Halpem, D.F., Lochlin, J.C., Perloff, R., Sternberg, R.J., Urbina, S. "Intelligence: Knowns and unknowns" *American Psychologist,* 51, 77-101, 1996.

[Newell 1990] Newell, A. *Unified Theories of Cognition*, Cambridge, Mass.: Harvard University Press, 1990.

[Ng and Mooney 1990] Ng.H.; Mooney, R. "On the role of coherence in abductive explanation" in *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 337-342 Boston, MA. AAAI, 1990.

[Niblett 1988] Niblett, T. "A study of generalisation in logic programs", In *Proc. European Working Sessions on Learning EWSL'88*, d. Sleeman (ED), Pitman, 131-138, 1988.

[Niblett 1993] Niblett, T "A Note on Refinement Operators" in Pavel B. Brazdil (Ed.): Machine Learning: ECML-93, European Conference on Machine Learning, Vienna, Austria, April 5-7, 1993, Proceedings. Lecture Notes in Computer Science, Vol. 667, Springer, 1993, pp. 329-335

[Nienhuys-Cheng and de Wolf 1997] Nienhuys-Cheng, S.H.; de Wolf, R. "Foundations of Inductive Logic Programming" Springer-Verlag 1997.

[Nienhuys-Cheng and Polman 1993] Nienhuys-Cheng & Polman, M. "Complexity Dimensions and Learnabiity" in Brazdil, P.B. *European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence 667, pp. 348-353, 1993.

[Nilsson 1986] Nilsson, N. "Probabilistic logic" Artificial Intelligence, 28, 71-88, 1986.

[Nilsson 1995] Nilsson, Nils J. "Eye on the Prize" AI Magazine, July 1995, also at http://robotics.stanford.edu/~nilsson/

[Nishida et al. 1991] Nishida, F.; Takamatsu, S.; Fujita, Y.; and Tani, T.: Semi-automatic program construction from specifications using library modules, *IEEE Trans. on Software Eng.*, 17, (9), pp. 853-871, (1991).

[Núñez et al. 1995] Núñez, G.; Cortés, U.; Larrosa, J., Non-Monotonic Characterization of Induction and Its Application to Inductive Learning, International Journal of Intelligent Systems, Vol. 10, 895-927, 1995.

[Nute 1988] Nute, D. "Defeasible reasoning: a philosophical analysis in Prolog." In James Fetzer (ed.), Aspects of Artificial Intelligence, Studies in Cognitive Systems, Kluwer Academic Publishers, Boston, 251-288, 1988.

[Nute 1992] Nute, D. "Basic Defeasible Logic" in L. Farinas_del Cerro and M. Penttonen (eds) "Intensional Logics for Programming", Clarendon Press, Oxford, 125-154, 1992.

[Nute 1994] Nute, D. "Defeasible logic." In D. Gabbay and C. Hogger (eds.), Handbook of Logic for Artificial Intelligence and Logic Programming, Vol. III, Oxford University Press, 1994:353-395.

[Nwana 1996] Nwana, H.: Software Agents: an overview, *Knowledge Engineering Review*, 11(3), pp.1-40, Sept. (1996).

[O'Rorke 1989] O'Rorke, P. "Coherence and abduction" *The Behavioural and Brain Sciences*, 12 (3), 484, 1989.

[Olson 1994] Olson, R. "Inductive Functional Programming Using Incremantal Program Transformation" Thesis Dissertation, University of Oslo, Dep. of Computer Science, 1994.

[Olson 1995] Olson, R. "Inductive functional programming using incremental program transformation", *Artificial Intelligence*, v. 74, n. 1, 1995.

[O'Rorke 1994] O'Rorke, P. "Abduction and Explanation-Based Learning: Case Studies in Diverse Domains" Computational Intelligence, Vol. 10, pp. 295-330, 1994.

[Page and Cohen 1995] Page, C.D.; Cohen, W.W. "Polynomial Learnability and Inductive Logic Programming: Methods and Results " New Generation Computing, 13, 369-409, 1995.

[Paris 1994] Paris, J.B. "The Uncertain Reasoner's Companion" Cambridge University Press, 1994.

[Parnas 1971] Parnas, D.L.: Information distribution of design methodology, Tech. Rept., Depart. Computer Science, Carnegie Mellon U., Pittsburgh, Pa., 1971. Also presented at the IFIP Congress 1971, Ljubljana, Yugoslavia.

[Parnas 1972] Parnas, D.L.: On the Criteria To Be Used in Decomposing Systems into Modules, *Communication of the ACM*, Vol. 15, no. 12, December 1972, pp. 1053-1058. http://www.acm.org/classics/may96.html

[Parsons 1982] Parsons, C. "Intensional logic in extensional language" *Journal of Symbolic Logic*, vol. 47, pp. 289-328, 1982.

[Partridge 1997] Partridge, D.: The Case for Inductive Programming: *IEEE* Computer, January, pp. 36-41, (1997).

[Patel-Schneider and Swartout 1994] Patel-Schneider, P.F.; Swartout, B. "Description logic knowledge representation system specification from the KRSS group of the ARPA knowledge sharing effort", AT&T Bell Laboratories Report, Murray Hill, NJ, 1994.

[Paul 1993] Paul, G. "Approaches to abductive reasoning: an overview" Artificial Intelligence Review, 7, 109-152, 1993.

[Paulson, L. 1987] Paulson, Lawrence C. 'Logic and Computation: Interactive Proof with Cambridge LCF' *Cambridge Tracts in Theoretical Computer Science 2*, Cambridge University Press, 1987.

[Paulson, L. 1994] Paulson, Lawrence C. 'Isabelle: A Generic Theorem Prover' with contributions by Tobias Nipkow. Springer Lecture Notes in Computer Science 828. XVII+321 pages, 1994.

[Pazzani and Kibler 1992] Pazzani, M.; Kibler, D. "The Utility of Knowlege in Inductive Learning" Machine Learning, Vol. 9., pp. 57-94, 1992.

[Pearl 1988] Pearl, J. "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference" Morgan Kaufmann, San Mateo, 1988.

[Pearl 1993] Pearl, J. "Belief Networks Revisited" Artificial Intelligence (59), 45-56, 1993.

[Pednault 1991] Pednault, E.P.D. "Minimal Length Encoding and Inductive Inference" in B. Piatetsky-Shapiro and W.J. Frawley (eds.) Knowledge Discovery in Databases, Cambridge, Mass.: MIT Press, pp. 71-92, 1991.

[Peirce 1867/1960] Peirce, C.S. "Collected papers of Charles Sanders Peirce" Cambridge. Harvard University Press 1960.

[Peng 1990] Peng, Y. and Reggia, J. "Abductive Inference Models for Diagnostic Problem Solving" Springer Verlag, New York, 1990.

[Peng and Reggia 1987] Peng, Y.; Reggia, J.A. "Abductive Inference Models for Diagnostic Problem-Solving, Symbolic Computation Series. Springer-Verlag, 1987.

[Pettorossi and Proietti 1990] Proietti, M.; Pettorossi, A. "Synthesis of eureka predicates for developing logic programs" in *Proceedings of ESOP '90* (Copenhagen), *Lecture Notes in Computer Science 432*, Springer Verlag, 306-325, 1990.

[Pettorossi and Proietti 1996a] Pettorossi, A.; Proietti, Maurizio "Rules and Strategies for Transforming Functional and Logic Programs" *ACM Computing Surveys,* Vol. 28, no. 2, June 1996.

[Pettorossi and Proietti 1996b] Pettorossi, A.; Proietti, Maurizio "Developing Correct and Efficient Logic Programs by Transformation" *Knowledge Engineering Review*, Vol. 11, No. 4, December 1996.

[Pfahringer 1994] Pfahringer, Bernhard "Controlling Constructive Induction in CiPF: An MDL Approach" in F. Bergadano and L. de Raedt (eds) Machine Learning, Proceedings of the European Conference on Machine Learning (ECML-94), pp. 242-256, Lecture Notes in AI 784, Springer-Verlag 1994.

[Pfahringer and Kramer 1995] Pfahringer, B. and Kramer, S. "Compression-Based Evaluation of Partial Determinations" Austrian Research Institute for AI, ai.univie.ac.at, 1995.

[Piatetsky-Shapiro and Frawley 1991] Piatetsky-Shapiro, G.; Frawley, J. "Knowledge Discovery in Databases" The AAAI Press, Menlo Park, 1991.

[Plaisted 1980] Plaisted, David A. 'Abstractions mappings in mechanical theorem proving' in W.Bibel and R. Kowalsi, (eds.) *5th International Conference on Automated Deduction*, pages 264-280, Lecture Notes in Computer Science, Vol. 87. Berlin, Springer, 1980.

[Plaza 1992] Plaza, E. "Reflection for analogy: Inference-level reflection in an architecture for analogical reasoning" in *Proc. IMSA'92 Workshop on Reflection and Metalevel Architectures*; pp. 166-171, 1992.

[Plaza and Arcos 1993] Plaza, E. and Arcos, J.L. "Flexible Integration of Multiple Learning Methods into a Problem Solving Architecture" Report de Recerca IIIA 93/16 Octubre 1993, also appeared in Proceedings of the European Workshop on Knowledge Acquisition in 1994.

[Plaza and Arcos 1996] Plaza, E. and Arcos, J.L. "Overview of Noos v.1.0. Draft" IIIA, "http://www.iiia.csic.es/ Project/Noos.html" 1996.

[Plotkin 1970] Plotkin G. "A note on inductive generalization" Machine Intelligence, Vol. 6, Edinburgh University Press, Edinburgh 1970.

[Polya 1968a] Polya, G. Patterns of Plausible Inference, Princeton University Press, Princeton, NJ. 1968.

[Polya 1968b] Polya, G. Induction and Analogy in Mathematics: Mathematics and Plausible Reasoning; Vol. I, Oxford University Press, London, 1968.

[Polya 1968c] Polya, G. Patterns of Plausible Inference, Mathematics and Plausible Reasoning; Vol. II, Oxford University Press, London, 1968.

[Polya 1969] Polya, G. Mathematical Discovery, Vols. I and II, Wiley, New York, 1969.

[Poole 1985] Poole, D. "On the Comparison of Theories: Preferring the Most Specific Explanation" in *IJCAI'85*, pages 144-147, 1985.

[Poole 1989] Poole, D. "Explanation and Prediction: An Architecutre for Default and Abductive Reasoning" Computational Intelligence 5(2), 97-110, 1989.

[Poole 1997] Poole, D. "Who chooses the assumptions?" in P. O'Rorke (ed) *Abduction*, AAAI/MIT Press, 1997.

[Popper 1935] Popper, K.R. "Logik der Forschung" Julius Springer, Wien. Engl. Transl. "The Logic of Scientifica Discovery" Hutchinson, London, 1959.

[Popper 1962] Popper, K.R. *Conjectures and Refutations: The Growth of Scientific Knowledge*, Basic Books, 1962.

[Popper 1963] Popper, K.R. *Conjectures and Refutations: The Growth of Scientific Knowledge*, Routledge and Kegan Paul, London, 1963.

[Popper 1969] Popper, K.R. "Conjectures and Refutations", London 1969.

[Popper and Miller 1983] Popper, K.R.; Miller, D. "A Proof of the Impossibility of Inductive Probability" Nature 302, 687-688.

[Popper and Miller 1987] Popper, K.R.; Miller, D. "Why Probabilistic Support is not Inductive" Philosophical Transactions of the Royal Society of London, A, 321, 569-591, 1987.

[Prawitz 1960] Prawitz, Dag "An improved proof procedure' Theoria, 26:102-139, 1960.

[Prawitz 1965] Prawitz, Dag "Natural Deduction" Alinquist and Wiksell, Stockholm 1965.

[Pressman 1992] Pressman, R.S.: *Software Engineering: A practitioner's Approach*, McGraw-Hill, 1992.

[Preston 1991] Preston, B. 1991. AI, anthropocentrism, and the evolution of "intelligence.". Minds and Machines 1:259-277.

[Quine 1953] Quine, W.V.O. "Two Dogmas of empiricism. From a Logical Point of View", Harvard University Press, Cambridge, Massachusetts, 1953.

[Quinlan 1986] Quinlan, J.R. "Induction of Decision Trees" Machine Learning, 1:81-206, 1986.

[Quinlan 1990] Quinlan, J.R. "Learning Logical Definitions from Relations" *Machine Learning*, 5 (3): 239-266, 1990.

[Quinlan and Cameron-Jones 1995] Quinlan, J.R.; Cameron-Jones, R.M. "Induction of Logic Programs: FOIL and Related Systems" New Generation Computing, 13, 287-312, 1995.

[Quinlan and Rivest 1989] Quinlan, J.; Rivest. R. "Inferring decision trees using the minimum description length principle" Information and Computation, vol. 80, 227-248., 1989.

[Quinlan, 1993] Quinlan, J.R. "C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, C.A., 1993.

[Rayner and Pollatsek 1989] Rayner, K.; Pollatsek, A. *The psychology of reading*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[Reddy 1990] Reddy, U.S. 'Term rewriting induction' in *Proceedings of the Tenth International Conference on Automated Deduction, Kaiserslautern*, PAGES 162-177. Springer LNAI vol 449, 1990,

[Reggia 1983] Reggia, J. "Diagnostic expert systems based on a set-covering model" International Journal of Man-Machine Studies, 19(5), 437-460, 1989.

[Reichenbach 1956] Reichenbach, H. "The Direction of Time" University of California Press, Berkeley and Los Angeles, 1956.

[Reuland and Abraham 1993] Reuland, E. and Abraham, W. (eds.) "Knowledge and Language", Vol. I, From Orwell's Problem to Plato's Problem, Kluwer Academic Publishers, 1993.

[Rissanen 1978] Rissanen, J.: Modelling by the shortest data description, *Automatica-J.IFAC*, 14, 465-471, 1978.

[Rissanen 1986] Rissanen, J., Stochastic complexity and modeling, Annals Statist. 14:1080-1100, 1986.

[Rissanen 1996] Rissanen, J.: Fisher Information and Stochastic Complexity, *IEEE Trans. Inf. Theory*, 1(42): 40-47, (1996).

[Rivest and Sloan 1994] Rivest, R.L.; Sloan, R. “A Formal Model of Hierarchical Concept Learning” Inf. and Comp. 114, 88-114, 1994.

[Robinson 65a] Robinson, J.A. ‘A machine-oriented logic based on Resolution Principle’, Journal of ACM 12, 23-41, 1965.

[Robinson 65b] Robinson, J.A. ‘Automated Deduction with Hyper-resolution’, Int. J. Comp. Math. 1, 227-234, 1965.

[Rumbaugh 1994] Rumbaugh, J.: Getting Started: Using Use Cases to Capture Requirements, *J. Object-Oriented Prog.*., Sept. p.8, (1994).

[Russell and Norvig 1995] Russell, S.; Norvig, P. “Artificial Intelligence: A Modern Approach” Prentice Hall 1995.

[Russell and Subramian 1993] Russell, S.J.; Subramanian, D. "Probably bounded optimal agents" in Proc. of IJCAI-93, Chanbery, France, Morgan Kaufmann 1993

[Russell and Wefald 1991] Russell, S.J. and Wefald, E.H. “Do the Right Thing: Studies in limited rationality” Cambridge, Massachusetts, MIT Press, 1991.

[Russell et al. 1993] Russell, S.J., Subramanian, D.; Parr, R. “Provably bounded optimal agents” *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 338-344, Chambery, France, 1993.

[Sagan 1973] Sagan, C. (ed). “Communication with Extraterrestrial Intelligence” Cambridge, Mass.: MIT Press, 1973.

[Sagan and Shklovskii 1971] Sagan, C. and Shklovskii, *Intelligent Life in the Universe*, 1971.

[Sakakibara 1997] Sakakibara, Y. *Recent advances of grammatical inference*, Theoretical Computer Science 185, pp. 15-45, 1997.

[Schaffer 1994] Schaffer, C. “A conservation law for generalization performance” in *Proceedings of the Eleventh International Conference on Machine Learning*, pages 259-265, Morgan Kaufmann, 1994.

[Schank 1982] Schank, R. "Dynamic Memory. A Theory of Learning in Computerss and People" Cambridge University Press, Cambridge, England 1982.

[Schank 1986a] Schank R.C., ``What is AI, Anyway?" in AI Magazine, 8, 59-65, 1987 and in R.J. Sternberg & D.K. Detterman, What is Intelligence? contemporary viewpoints on its nature and definition Norwood, NJ. : Ablex, 1986.

[Schank 1986b] Schank, R. "Explanation Patterns. Understanding Mechanically and Creatively" Lawrence Erlbaum Associatess, Hillsdale, NJ. 1986.

[Schank and Abelson 1977] Schank, R.; Abelson, R. "Scripts, Plans, Goals and Understanding" Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.

[Schank et al. 1994] Schank, R.; Riesbeck, C.; Kass, A. (Eds.) "Inside Case-Based Explanation" Lawrence Erlbaum Associates, Hillsdale, NJ., 1994.

[Schlechta 1995] Schlechta, Karl "Logic, Topology and Integration" J. of Automated Reasoning, 14:353-381, 1995.

[Schmidhuber 1997] Schmidhuber, Jürgen "What's Interesting?" In Abstract Collection of SNOWBIRD: Machines That Learn. Utah, April 1998 based on TR IDSIA-35-97, Lugano, Switzerland 1997.

[Schmidhuber et al. 1997a] J. Schmidhuber, J. Zhao and M. Wiering, "Shifting Inductive Bias with Success-Story Algorithm, Adaptive Levin Search, and Incremental Self-Improvement" *Machine Learning*, 28, 105-132, (1997).

[Schmidhuber et al. 1997b] J. Schmidhuber, J. Zhao, N. Schraudolph. Reinforcement learning with self-modifying policies. In S. Thrun and L. Pratt, eds., Learning to learn, Kluwer, pages 293-309, 1997.

[Sebag and Rouveirol 1997] Sebag, M.; Rouveirol, C. "Tractable induction and classification in FOL" in Proceedings of IJCAI-97, 888-892, Morgan Kaufmann.

[Seidenfeld 1986] Seidenfeld, T. "Entropy and uncertainty" *Philosophy of Science*, 53, 467-491, 1986.

[Selman and Levesque 1990] Selman, B.; Levesque, H.J. "Abductive and default reasoning: a computational core" In Proceeding of the Eighth National Conference on Artificial Intelligence, pp. 343-348 Boston, MA. AAAI. 1990.

[Sempere 1998] Sempere, José M. "Some Learning Systems are Interactive Proof Systems" Learning 98, Getafe, September 1998.

[Shanahan 1989] Shanahan, M. "Prediction is Deduction but Explanation is Abduction" in Proc. IJCAI'89, p.1055-1060, 1989.

[Shanahan 1993] Shanahan, M. "Explanation in the Situation Calculus" Proceedings of IJCAI'93, pp. 160-165, 1993.

[Shanahan 1997] Shanahan, M. "Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia", The MIT Press 1997.

[Shannon 1948] Shannon, C.E. "The mathematical theory of communication" *Bell System Tech. J.*, 27:379-423, 623-656, 1948.

[Shapiro 1981] Shapiro, E. "Inductive Inference of Theories form Facts" RR 192, D. Comp. Science, Yale Univ., 1981, reprinted in Lassez, J.; Plotking, G. (eds.) "Computational Logic" The MIT Press 1991.

[Shapiro 1981] Shapiro, E. "Inductive Inference of Theories form Facts" RR 192, D. Comp. Science, Yale Univ., 1981, in Lassez, J.; Plotking, G. (eds.) "Computational Logic" The MIT Press 1991.

[Shapiro 1984] Shapiro, E.Y. "Alternation and the computational complexity of logic programs" *J. Logic Programming*, 1, 19-33, 1984.

[Shapiro 1992] Shapiro, S.C., The Turing Test and The Economist, SIGART Bulletin, Vol. 3, No. 4, 10-11, October 1992.

[Sharger and Langley 1990] Sharger, J. and Langley, P., 1990, Computational Models of Scientific Discovery and Theory Formation, Morgan Kaufmman, 1990.

[Shieber 1994] Shieber, S.M. "Lessons from a Restricted Turing Test" Communications of the ACM, June 1994, Vol. 37, No. 6.

[Shinohara 1991] Shinohara, T. "Inductive Inference of Monotonic Formal Systems from Positive Data" New Generation Computing 8, 371-384, 1991.

[Simon 1982] Simon, H. "Models of Bounded Rationality" Cambridge, MIT Press 1982.

[Simon and Kotovsky 1963] Simon, H.; Kotovsky, K. "Human acquisition of concepts for sequential patterns" Psych. Review 70, 534-46, 1963.

[Solomonoff 1964] Solomonoff, R.J. "A formal theory of inductive inference" Inform. Contr. vol. 7, pp. 1-22, Mar. 1964; also, pp. 224-254, June 1964.

[Solomonoff 1978] Solomonoff, R.J. "Complexity-based induction systems: comparisons and convegence theorems" *IEEE Trans. Inform. Theory*, IT-24:422-432, 1978.

[Solomonoff 1986] Solomonoff, R.J. "The Application of Algorithmic Probability to Problems in AI" in L.N. Karnal; J.F. Lemmer(eds) *Uncertainty in AI*, Elsevier Science, pp.473-91, 1986.

[Sommer 1995a] Sommer, E.: "Fender: An approach to theory restructuring" in Proc of the European Conference on Machine Learning (ECML-95), 1995.

[Sommer 1995b] Sommer, E.: "An Approach to Quantifying the Quality of Induced Theories" in C. Nedellec (ed.), Proc. IJCAI'95 Workshop on Machine Learning and Comprehensibility, 1995.

[Sommer et al. 1995] Sommer, E.; Emde, W.; Kietz, J.U.; Wrobel, S. "Mobal 3 User Guide" Arbeitspapiere der gmd, GMD, 1995. Also at http://natham.gmd.de/projects/ml/home.html"

[Sommerville 1992] Sommerville, I.: *Software Engineering. Fourth Edition*, Addison-Wesley, 1992.

[Spearman 1904] Spearman, C. "'General Intelligence' objectively determined and measured" Amer. J. of Psych-, 15, 201-293, 1904.

[Spirtes et al. 1993] Spirtes, P., Glymour, C., and Scheines, R. *Causation, Prediction, and Search*, Springer-Verlag, New York, 1993.

[Srinivasan and Camacho 1997] A. Srinivasan and R.C. Camacho "Experiments in numerical reasoning with ILP" Journal of LP (accepted), >=1997.

[Srinivasan et al. 1994] Srinivasan, A.; Muggleton, S.H.; King, R.D.; Sternberg, M.J.E. "Mutagenesis; ILP Experiments in a Non-Determinate Biological Domain" in Wrobel, S. (ed.) Proceedings of the Fourth International Inductive Logic Programming Workshop, Gesellschaft für Mathematik und Datenverarbeitung MBH, 1994, GMD-Studien Nr 237, 1994.

[Stahl 1994] Stahl, I. "On the utility of Predicate Invention in Inductive Logic Programming" in Franceso Bergadano and Luc de Raedt (eds) Machine Learning, Proceedings of the European Conference on Machine Learning (ECML-94), pp. 272-286, Lecture Notes in AI 784, Springer-Verlag 1994.

[Stahl 1994] Stahl, Irene "On the Utility of Predicate Invention in Inductive Logic Programming" ECML94, 272-286. Begadano, F and De Radetyl 1994.

[Stahl 1995] Stahl, I. "The appropiateness of predicate invention as bias shift operation in ILP" *Machine Learning*, 20:95-117, 1995.

[Sternberg 1977] Sternberg, R.J. "Intelligence, Information Processing, and Analogical Reasoning" John Wiley & Sons 1977

[Stickel 1990] Stickel, M. E. "Rationale and Methods for Abductive Reasoning in Natural-Language Interpretation" in R. Studer (ed.) "Natural Language and Logic" Lecture Notes in AI 459, pp. 233-252, Springer-Verlag 1990.

[Stolcke and Omohundro 1994] Stolcke, A.; Omohundro, S. "Inducing Probabilistic Grammars by Bayesian Model Merging" in R.C. Carrasco and J. Oncina (Eds.) Grammatical Inference and Applications, Lecture Notes in Artificial Intelligence, 862, pp. 106-118, Springer-Verlag 1994.

[Stolcke and Omohundro 1994] Stolcke, A.; Omohundro, S. "Inducing Probabilistic Grammars by Bayesian Model Merging" in R.C. Carrasco and J. Oncina (Eds.) *Grammatical Inference and Applications*, Lecture Notes in Artificial Intelligence, 862, pp. 106-118, Springer-Verlag 1994.

[Stonier 1992] Stonier, T. "Beyond Information. The Natural History of Intelligence" Springer-Verlag 1992.

[Storer 1988] Storer, J.A. "Data Compression: Methods and Theory" Computer Science Press, 1988.

[Suttner and Sutcliffe 1996] Suttner, C.B.; Sutchliffe, G. "The TPTP Problem Library", Tech. Univ. Munich, Germany, 1996

[Sutton 1991] R.S. Sutton, "Special issue on reinforcement learning" *Machine Learning*, 1991.

[Szyperski 1998] Szyperski, C.: *Component Software – Beyond Object-Oriented Programming*, Addison Wesley Longman Limited, 1998.

[Takeuti 1987] Takeuti, G. "Proof Theory" Second Edition, North Holland 1987.

[Tarski 1936] Tarski, A. "On the concept of logical consequence" in *Logic, semantics, metamathematics* (J.H. Woodger translator), Oxford at the Clarendon Press, Oxford, 1956.

[Tarski 1956a] Tarski, A. "The concept of truth in formalized languages" in *Logic, semantics, metamathematics* (J.H. Woodger translator), Oxford at the Clarendon Press, Oxford, 1956.

[Tarski 1956b] Tarski, A. "Some methodological investigations on the definability of concepts" in *Logic, semantics, metamathematics* (J.H. Woodger translator), Oxford at the Clarendon Press, Oxford, 1956.

[Thagard and Nowak 1990] Thagard P., Nowak G. "The Conceptual Structure of the Geological Revolution, in Shrager J., Langley P. (eds.): Computational Models of Discovery and Theory Formation, Morgan Kaufmann, San Mateo, CA, 1990.

[Thagard 1978] Thagard, P. "The best explanation: Criteria for theory choice" Journal of Philosophy, 75, 76-92, 1978.

[Thagard 1986] Thagard, P., The emergence of meaning: An escape from Searle's Chinese Room. Behaviorism 14:139-46, 1986.

[Thagard 1989] Thagard, P. "Explanatory coherence" *The Behavioural and Brain Sciences*, 12 (3), 435-502, 1989.

[Thagard 1992] Thagard, P., 1992, Conceptual Revolutions, Princeton Univ. Pr, Princeton, N.Y.

[Thagard 1998] Thagard, P. "Probabilistic networks and explanatory coherence" in P. O'Rorke & J. Josephson (eds), Automated Abduction: Inference to the best explanation, Menlo Park, AAAI Press 1998.

[Thagard and Shelley 1997] Thagard, P. and Shelley, C., 1997, Abductive reasoning: Logic, visual thinking, and coherence. URL: http://cogsci.uwaterloo.ca/Articles/Pages/Coherence.html.

[Thagard and Verbeurgt 1997] Thagard, P. and Verbeurgt, K., 1997, Coherence as Constraint Satisfaction, Cognitive Science, forthcoming, 1997.

[Tour et al. 1987] de la Tour, T. Boy; Caferra, R. 'Proof analogy in interactive theorem proving: A method to express and use it via second order pattern matching' in Proceedings AAAI-87, pages 95-99, San Mateo CA, Morgan Kaufmann, 1987.

[Tukey 1977] Tukey, J. *Exploratory Data Analysis*, Rading, MA, Addison Wesley, 1977.

[Turing 1936] Turing, A.M. "On computable numbers with an application to the Entscheidungsproblem" *Proc. London Math. Soc.*, series 2, 42:230-265, 1936. Correction, Ibid, 43:544-546, 1937.

[Turing 1950] Turing, A.M. "Computing Machinery and Intelligence" Mind 59: 433-460, 1950.

[Tymoczko 1986] Tymoczko, Thomas ``New Directions in the Philosophy of Mathematics. An Antology" Birkhäuser Boston, Inc. 1986.

[Ungar 1992] Ungar, A.M "Normalization, Cut-Elimination and the Theory of Proofs", CSLI Lecture Notes 28, 1992.

[Uspensky 1992] Uspensky "Kolmogorov and Mathematical Logic" Journal of Symbolic Logic, 57, 2, pp. 385-412, 1992

[Valiant 1984] Valiant, L. "A theory of the learnable". Communication of the ACM 27(11), 1134-1142, 1984.

[van Benthem 1988] van Benthem, Johan "A Manual of Intensional Logic" CSLI Lecture Notes 72, 1988, Second Edition.

[van den Bosch 1994] van den Bosch, *Simplicity and Prediction*, Master Thesis, Dep. of Science, Logic & Epistemology of the Fac. of Philosophy at the Univ. of Groningen, 1994.

[Van der Laag and Nienhuys-Cheng 1998] Van der Laag, P.R.J.; Nienhuys-Cheng, S. "Completeness and Properness of Refinement Operators in Inductive Logic Programming" *J. of Logic Programming*, 201-225, March 1998.

[Varsek 1993] Varsek, A. "Genetic Inductive Logic Programming", Doctoral Dissertation, University of Ljubljana, Slovenia, 1993.

[Varsek 1999] Varsek, A. *Personal Communication*, 1999.

[Vitányi and Li 1996] Vitányi, P.; Li, M. "Minimum Description Length Induction, Bayesianism, and Kolmogorov Complexity"

[Vitányi and Li 1997] Vitányi, P.; Li, M. "On Prediction by Data Compression", *Proc. 9th European Conference on Machine Learning*, Lecture Notes in AI, Vol. 1224, Springer-Verlag, 14-30, 1997.

[von Wright 1951] von Wright "Deontic Logic", Mind 1951.

[Wagner 1998] Wagner, G. "Foundations of Knowledge Systems. With Applications to Databases and Agents" Institut für Informatik, Universität Leipzig, http://www.informatik.uni-leipzig.de/~gwagner, 1998.

[Wallace and Boulton 1968] Wallace, C.S; Boulton, D.M "An information measure for classification" *Computer Journal* 11 (1968) 185-195.

[Walther 1994] Walther, C. "Mathematical Induction" in Gabbay, Hogger & Robinson (eds.) Handbook of Logic in AI and LPI, vol 2, deduction methodologies, Clarendon Press, Oxford 1994.

[Watanabe 1972] Watanabe, S. "Pattern Recognition as Information Compression" in Watanabe (ed.) Frontiers of Pattern Recognition New York: Academic Press, 1972.

[Watanabe 1992] Watanabe, O. (ed.) "Kolmogorov complexity and computational complexity" Monographs on TCS, Springer 1992.

[Wegner 1996] Wegner, P.: Interactive Software Technology, *Handbook of Computer Science and Engineering.*, CRC Press, Dec. 1996. URL: http://www.cs.brown.edu/~pw/

[Wegner 1998] Wegner, P.: "Interactive Foundations of Computing", *Theoretical Computer Science*, Feb. 1998. URL: http://www.cs.brown.edu/~pw/

[Weidenhaupt 1998] Weidenhaupt, K.; Pohl, K.; Matthias, J.; Haumer, P.: Scenarios in System Development: Current Practice, *IEEE Software*, March-April, 34-45, (1998).

[Weiss and Indurkhya 1997] Weiss, S. M.; Indurkhya, N. "Predictive Data-Mining: A Practical Guide" Morgan Kaufmann Publishers, San Franciso, 1997.

[Wexler 1992] Wexler, K. "The Subset principle is an intensional principle" in *Knowledge and Language: Issues and Representation and Acquisition* (E. Reuland and W. Abrahamson, eds.), Kluwer Academic Publishers, 1992

[Wexler 1993] Wexler, K. "The subset principle is an intensional principle" in Eric Reuland and Werner Abraham (eds.) "Knowledge and Language", Vol. I, From Orwell's Problem to Plato's Problem, 217-239, Kluwer Academic Publishers, 1993.

[Wexler and Culicover 1980] Wexler, K. and Culicover, P. "Formal Principles of Language Acquisition" MIT Press, Cambridge 1980.

[Weyhrauch 1980] Weyhrauch, R.W. "Prolegomena to a theory of mechanized formal reasoning", Artificial Intelligence 3(1), 1980, pp. 133-170. Also in Brachman, Ronald J.; Levesque, Hector J. "Readings in Knowledge Representation" Morgan Kaufmann Publishers, Inc. 1985.

[Weyhrauch et al. 1996] Weyhrauch, Richard W.; Talcott, Carolyn L. 'Towards a Theory of Solving Problems', Department of Computer Science, Stanford University, Stanford, February 1996.

[Weyuker 1988] Weyuker, E.: Evaluating software complexity measures, *IEEE Trans Software Eng.*, vol. 14, pp. 1357-1365, Sept. (1988).

[Whewell 1847] Whewell, W. "The philosophy of the inductive sciences" New York, Johnson Reprint Corp, 1847.

[Winograd 1986] Winograd, T. "Thinking Machines: Can There Be? Are We?" in Winograd & Fernando Flores "Understanding Computers and Cognition: A New Foundation for Design" Norwood, 1986.

[Winston 1980] Winston, Patrick, H. 'Learning and reasoning by analogy' Communications of the ACM, 23(12):689-703, 1980.

[Winston 1982] Winston, Patrick Henry "Learning New Principles from Precedents and Exercises" *Artificial Intelligence*, vol. 19, no. 3, 1982.

[Winston 1992] Winston, P.H. "Artificial Intelligence" Third edition, Addison-Wesley Pusblishing Company 1992

[Wittgenstein 1922] Wittgenstein, L. *Tractatus Logico-Philosophicus* 1922

[Wolff 1991] Wolff, J.Gerard Towards a Theory of Cognition and Computing, Chichester: Ellis Horwood, 1991.

[Wolff 1992] Wolff, "Information Compression and Logic" New Generation Computing, 13, 187-214, 1995, OHMSHA, LTD. and Springer-Verlag 1995.

[Wolff 1994] Wolff, J.G.: Towards a new concept of software, *Software Engineering Journal*, IEE, January (1994).

[Wolff 1995] Wolff, J.G. "Computing as Compression: An Overview of the SP Theory and System" New Gen. Computing 13, 187-214, 1995.

[Wolpert 1992] Wolpert, D. "On the connection between in-sample testing and generalization error" *Complex Systems*, 6:47-94m 1992.

[Woods 1975] Woods, W.A. "What's in a Link: Foundations for semantic Networks" in *Representation and Understanding: Studies in Cognitive Science*, D.G Bobrow and A.M. Collins (ed), 35-82, Academic Press, Republished in Brachman and Levesque 1985.

[Wooldridge and Jennings 1995] Wooldridge, M.; Jennings, N.: Intelligent Agents:Theory and Practice, *Knowledge Eng. Review* 10 (2), 115-152, (1995).

[Wos 1996] Wos, Larry 'The Resonance Strategy' Mathematics and Computer Science Division, Argonne National Laboratory. Source: *http://www.mcs.anl.gov/people/wos/index.html*

[Wos et al. 1992] Wos, L.; Overbeek, R.; Lusk, E.; Boyle, J. "Automated Reasoning: Introduction and Application", 2nd. ed., McGraw-Hill, New York 1992.

[Wos et al. 1994] Wos, Larry; Veroff, Robert 'Logical Basis for the Automation of Reasoning: Case Studies' in Gabbay, Hogger & Robinson (eds.) Handbook of Logic in AI and LP, Volume 2, Deduction Methodologies, Clarendon Press, Oxford 1994.

[Wrobel 1995] Wrobel, S. "First Order Theory Refinement" ILP Project (ESPRIT III) Common Deliverable for Workpackage 2, 1995.

[Yu 1994] Yu, C. H. "Abduction? Deduction? Induction? Is there a Logic of Exploratory Data Analysis?" Annual Meeting of American Educational Research Associations, New Orleans, 1994.

[Zadeh 1965] Zadeh, L.A. "Fuzzy Sets," Information and Control, No 8, pp. 338-353, June 1965.

[Zadeh 1972] Zadeh, L.A. "A Fuzzy-Set-Theoretic Interpretation of Linguistic Hedges". Journal of Cybernetics, Vol.2, 1972.

[Zadeh 1983] Zadeh, L.A. "A Computational Approach to Fuzzy Quantifiers in Natural Language". Comp. and Maths with Appls, Vol.9, 1983.

[Zalta 1998] Zalta, Edward N. "Intentional Logic and the Metaphysics of Intentionality" 1998

[Zemel 1993] Zemel, R "A minimum description length framwork for unsupervised learning" Ph. D. Thesis, Dept. of Computer Science, University of Toronto, Toronto, Canada, 1993.

[Zemel 1993] Zemel, R. "A minimum description length framework for unsupervised learning", Ph. D. Thesis, Dept. of Computer Science, University of Toronto, Toronto, Canda, 1993.

[Zeugmann and Lange 1995] Lange, S.; Zeugmann, Thomas "A Guided Tour Across the Boundaries of Learning Recursive Languages" in Jantke, K.P.; Lange, S. (eds) Algorithmic Learning for Knowledge-Based Systems, Lecture Notes in Artificial Intelligence, Vol. 961, Springer, Berlin, pp. 193-262, 1995.

[Zilberstein 1995] Zilberstein, S. "Models of Bounded Rationality. A concept paper" AAAI Fall Symposium on Rational Agency, Cambridge, Massachusetss, November 1995.

[Zilberstein 1996] Zilberstein, S. "Resource-bounded reasoning in intelligent systems" Computing Survey 28(4), 1996.

[Zilberstein 1999] S. Zilberstein and the Resource-Bounded Reasoning Lab. "What is resource-bounded reasoning?" http://anytime.cs.umass.edu/Home.html

[Zurek 1989a] Zurek, W.H. "Thermodynamic cost of computation, algorithmic complexity and the information metric" Nature, 341:119-124, 1989.

[Zurek 1989b] Zurek, W.H. "Algorithmic randomness and physical entropy" Phys. Rev., A40:4731-4751, 1989.

[Zuse 1991] Zuse, H.: *Software Complexity: Measures and Metrics*, Berlin, Germany: Walter de Gruyter, 1991.

[Zytkow 1993] Zytkow, J.M. "Introduction: Cognitive Autonomy in Machine Discovery" in Special Issue on Machine Discovery, *Machine Learning*, 12 (1-3), 1993.

Appendix

D. Acronyms

Entre deux mots il faut choisir le moindre.
Paul Valéry (1871 - 1945)

Many acronyms appear in this dissertation. The intention has been to give the full meaning the first time the acronym appears in the text but, in some cases, there can be a long space between the first appearance and the second one. Therefore, a listing of all them is included here with their correspondence or a brief explanation.

AC: Computational Accepter.

AI: Artificial Intelligence

AILP: Abductive Inductive Logic Programming.

ALP: Abductive Logic Programming [Kakas et al. 1993].

ANN: Artificial Neural Networks.

ATP: Automatic Theorem Proving.

B: Generally, the *Background Knowledge*.

CBR: Case-Based Reasoning.

CRC: Cyclic Redundancy Code.

DBMS: Database Management System.

DS: Computational Deterministic Derivational (or simply Deductive) System.

E^-: Generally, the *Negative Evidence*.

E: Generally, the *Evidence*.

E^+: Generally, the *Positive Evidence*.

EBL: Explanation-Based Learning.

$G(x|y)$: Computational Information Gain from *y* to *x*.

GD: Generalisation Degree.

H: Generally, the *Hypothesis*.

ID3: successful machine learning algorithm [Quinlan 1986, 1990].

ILP: Inductive Logic Programming.

IQ: Intelligence Quotient.

$K(x|y)$: Relative Kolmogorov Complexity of *x* given *y*.

KDD: Knowledge Discovery in Databases.

$Kt(x|y)$: Relative Levin Complexity of *x* given *y*.

LGG: Least General Generalization.

LLL: A new field called "*language, learning and logic*" which combines logic, ML and NLP.

$LT(\cdot)$: Function weighing the length of a program with the log. of its computational cost.

MBR: Model Based Reasoning.

MC: Model Complexity.

MDL: Minimum Description Length.

ML: Machine Learning.

MLE: Maximum Likelihood Estimator. Also, Minimal Length Encoding.

MML: Minimum Message Length.

NLP: Natural Language Processing.

NP: Non Polynomial.

P: Generally, a *Program*. Also Polynomial (computable in polynomial time).

PAC-learning: Probably Approximate Correct learning.

PC: Proof Complexity.

RL: Reinforcement Learning.

RLGG: Relative Least General Generalization.

SAT problem: to decide whether a Boolean formula in conj. normal form is satisfiable.

SED: Shortest Explanatory Description.

SLD: Selective linear resolution for definite clauses.

SLDNF: SLD with Negation as Failure.

T: Generally, a *Theory*.

***TG*(*x*|*y*)**: True Information Gain from *y* to *x*.

TP: Computational Theorem Prover.

TT: Turing Test. An Imitation Game, for ascertaining humanity.

***V*(*x*|*y*)**: Absolute (Time Ignoring) Information Gain from *y* to *x*.

Appendix

E. Index