UNIVERSIDAD POLITÉCNICA DE VALENCIA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

RESOLUCIÓN SL*:

UN PARADIGMA BASADO EN RESOLUCIÓN LINEAL PARA LA DEMOSTRACIÓN AUTOMÁTICA

Tesis doctoral presentada por:
Juan C. Casamayor Ródenas

Dirigida por:
Dr. Isidro Ramos Salavert
Dr. Hendrik Decker

Tesis Doctoral

RESOLUCIÓN SL*:

UN PARADIGMA BASADO EN RESOLUCIÓN LINEAL PARA LA DEMOSTRACIÓN AUTOMÁTICA

Memoria presentada por Juan C. Casamayor Ródenas, licenciado en Ciencias Físicas por la Universidad de Valencia, para la obtención del grado de doctor en Informática por la Universidad Politécnica de Valencia.

DIRECTORES:

Dr. Isidro Ramos Salavert (Universidad Politécnica de Valencia)

Dr. Hendrik Decker (Siemens AG, Munich)

TRIBUNAL:

Mario Rodríguez Artalejo (Universidad Complutense de Madrid)

José Cuena Bartolomé (Universidad Politécnica de Madrid)

Juan José Moreno Navarro (Universidad Politécnica de Madrid)

María Alpuente Frasnedo (Universidad Politécnica de Valencia)

Matilde Celma Giménez (Universidad Politécnica de Valencia)

AGRADECIMIENTOS

En primer lugar me gustaría agradecer el apoyo recibido por los miembros del grupo de investigación de Programación Declarativa y en particular a los del subgrupo de Bases de Datos Deductivas. Quiero también expresar mi agradecimiento a Matilde Celma y Laura Mota por su ayuda, que ha hecho posible en muchas ocasiones que pudiera dedicar mi tiempo y esfuerzo a finalizar el presente trabajo. También me gustaría agradecer a Francisco Marqués, su interés en el tema de la demostración automática que me ha estimulado a continuar en este campo, y por su colaboración en numeroso aspectos, tanto teóricos como de implementación. Por último, dar las gracias a los directores de la presente tesis, Isidro Ramos y Hendrik Decker, por su interés y apoyo, que ha sido clave en su desarrollo y consecución.

PRÓLOGO

El trabajo incluido en la presente tesis se enmarca dentro del campo de la demostración automática de teoremas y consiste en la estudio, definición y desarrollo de un paradigma de resolución lineal, denominado *Resolución SL**. La razón para utilizar la denominación de paradigma reside en el hecho de que en sí misma resolución SL* no es un procedimiento, sino que se puede entender como una forma de razonamiento con ciertos parámetros cuya instanciación da lugar a diferentes procedimientos que son adecuados para el tratamiento de distintos tipos de problemas. Por otro lado, se le ha dado el nombre de resolución SL* porque, como posteriormente se explicará, está muy cercano a Eliminación de Modelos y a resolución SL (de ahí la primera parte del nombre). El asterisco final quiere denotar su parametrización, de forma que los procedimientos instancias de resolución SL* serán denominados con una letra más en vez del asterisco, como posteriormente se verá.

La tesis ha sido dividida en cuatro capítulos que se describen brevemente a continuación.

En el primero se realiza una breve introducción histórica a la demostración automática, que va desde los orígenes de la lógica con el uso de las primeras notaciones matemáticas formales en el siglo XVI hasta la aparición de los resultados más importantes de la lógica descubiertos por Herbrand, Gödel, Church, etc. Se hace un especial hincapié en este capítulo en la demostración automática realizando un recorrido desde sus orígenes a finales del siglo XVIII hasta el momento actual, en el cual es posible ver cuál ha sido la evolución de este campo y qué descubrimientos y resultados se pueden presentar como los principales puntos de inflexión.

En el segundo capítulo se presentan la resolución lineal y algunos de sus principales refinamientos, ya que resolución SL* es un variación de resolución SL y por tanto de resolución lineal. Para ello se introduce el principio de resolución, viendo los problemas de su mecanización, y posteriormente se ven dos refinamientos de resolución: resolución semántica y resolución lineal. Para concluir se estudian los principales refinamientos de resolución lineal: resolución de entrada, resolución lineal con fusión, resolución lineal con subsumción, resolución lineal ordenada, resolución MTOSS y TOSS, Eliminación de Modelos, resolución SL y el sistema MESON.

En el tercer capítulo se presentan y estudian con profundidad las principales aportaciones al campo de la demostración automática que se han producido en los últimos años y que están cercanas a la aproximación del presente trabajo. Se han incluido los siguientes trabajos: el demostrador PTTP de Stickel, el sistema MESON basado en secuencias de Plaisted, el

demostrador SATCHMO de Manthey y Bry, los procedimientos Near-Horn Prolog de Loveland y otros autores y, por último, el demostrador SETHEO de Bibel y otros autores. Obviamente no se han incluido todos los demostradores y procedimientos, pero sí aquellos que se han considerado como los más interesantes y cercanos a resolución SL* de manera que sea posible realizar comparaciones, de forma que queden patentes las aportaciones realizadas.

En el cuarto capítulo se presenta resolución SL*. Se da la definición formal de la misma y se introduce el concepto fundamental de *elección de ancestros*. La elección de ancestros es el mecanismo que permite controlar la aplicación de la resolución de ancestro haciendo posible una reducción del coste de su aplicación y una adecuación de resolución SL* al tipo de problema a tratar. Posteriormente se ven las principales instancias de resolución SL*, los procedimientos SLT y SLP. En este capítulo se hace un especial hincapié en la elección de ancestros, ya que es la principal aportación de resolución SL*, analizando tanto las ventajas que aporta asociadas al incremento de la eficiencia como el hecho de dotar a resolución SL* la capacidad de adaptarse a los problemas que trata. También en este capítulo se presenta una implementación de resolución SL*, en particular del procedimiento SLT, y se incluyen resultados sobre un conjunto extenso de problemas del campo de la demostración automática. En la última sección de este capítulo se realiza una comparación de resolución SL* con los demostradores y sistemas más cercanos, tanto a nivel de caractarísticas como de resultados.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1:

1. LÓGICA Y RAZONAMIENTO AUTOMÁTICO

1.1. BREVE HISTORIA DE LA LÓGICA

Fue Frege en 1879 el que abrió un nuevo periodo en la lógica que dio como resultado la lógica moderna. No obstante, previas a la aportación de Frege, cabe destacar un buen número de valiosas ideas que han sobrevivido, ya sea en el propio sistema formal de Frege o por sus propios méritos.

El uso de notaciones matemáticas proviene del siglo XVI. Estas notaciones matemáticas permitieron la aparición del *cálculo simbólico* que puede ser utilizado para la resolución de problemas, tanto científicos como prácticos. El *cálculo simbólico* puede ser entendido como el procedimiento que nos permite, a partir de unas fórmulas escritas en un papel, su manipulación y transformación en otras fórmulas, pudiendo así llegar a descubrir a partir de estas últimas hechos o ideas que permanecían ocultos. Así, se puede considerar que el formalismo matemático que Issac Newton usó para hallar y demostrar sus descubrimientos en el campo de la Física, el *cálculo diferencial e integral*, es un cálculo simbólico de reescritura y manipulación de fórmulas.

Fue el filósofo Thomas Hobbes el que consideró en un principio como lógica a la computación simbólica o manipulación de fórmulas en 1655 y ya presentó la idea de que el razonamiento podía ser reducido o entendido como una clase de cálculo. Creía Hobbes que debía ser posible desarrollar la forma en la cual, aplicando reglas analíticas, símbolos complejos fueran descompuestos en sus simples constituyentes y que estos últimos eran reensamblados en otros símbolos complejos a base de utilizar reglas sintéticas. Él llegó a escribir, "no debemos pensar que la computación, que es raciocinación, tenga lugar sólo con números".

El filósofo y matemático G.W. Leibnitz tenía sólo ocho año cuando Hobbes escribió esas ideas. Más adelante cuando tenía diecinueve, y habiendo recibido la influencia de este último, puso de manifiesto su objetivo: desarrollar una lengua lógica universal, *lingua characteristica universalis*, y un cálculo simbólico, *calculus racionator*, en los cuales la computación conceptual imaginada por Hobbes pudiera ser realizada, con tanta fluidez y mecanización como el cálculo numérico. Leibnitz no logró su objetivo esencialmente por la restringida noción de forma lógica que tomó como base de su análisis y que provenía del sistema silogístico aristotélico. Así supuso que la conjunción era el único modo de combinación de dos conceptos o proposiciones.

Durante los doscientos años que pasaron desde la propuesta de Leibnitz hasta el sistema de Frege hubo otros intentos, pero todos ellos usaron la asunción de que los conceptos lógicos debían de algún modo ser modelados por conceptos numéricos y algebraicos, en otras palabras por nociones y modelos ya presentes en varios sistemas numéricos y sus operaciones. Este fue en todos los casos un error, el mismo que cometió Leibnitz, que les impedía llegar al objetivo buscado. De entre estos intentos cabe destacar el Álgebra de Boole en 1852 como uno de los más destacados.

En 1879 Frege encontró el modo de formalizar lo que actualmente se conoce como Lógica. Frege deliberadamente evitó la tentación de buscar contrapartidas numéricas y analogías para cada característica de las estructuras y procesos conceptuales que aparecen en el pensamiento lógico. Así, se puede considerar que él fue el primero en apreciar la sabiduría del comentario de Hobbes: "no debemos pensar que la computación tiene lugar sólo en los números".

Frege comprendió que no era porque los números fueran *números* por lo que se puede aplicar sobre ellos las matemáticas y computar con ellos; sino que ellos y sus propiedades pueden ser asociados con elementos de una estructura simbólica organizada, definida por reglas precisas de formación y transformación. La computación numérica y la manipulación algebraica tradicionales eran simplemente *razonamiento deductivo*. Frege asimiló totalmente estas ideas y llegó a la conclusión de que él podía diseñar un cálculo simbólico completo usando las más altas normas de precisión en sus reglas de formación y transformación, que no eran en absoluto

"numéricas". Era una notación simbólica neutra, de propósito general, totalmente expresiva para representar la forma lógica y para manipular estas representaciones. El nombre que el dio a su notación – *begriffsschrift* – significa "ideografía" o "escritura de conceptos". Esta notación sirve para la expresión del "pensamiento puro" y ninguna materia en particular está asociada con ella. Esta presentación hecha por Frege se corresponde, en esencia, con el cálculo de predicados tal y como hoy lo conocemos, aunque el conjunto de símbolos usados sea diferente del que se usa en la actualidad.

La historia del cálculo de predicados desde 1879 ha sido rica y llena de intensa investigación en la cual la teoría del sistema formal ha sido promovida a su estado presente altamente desarrollada por una serie de descubrimientos matemáticos fundamentales. Durante la primera década del presente siglo la lógica estuvo fuertemente convulsionada por las llamadas "paradojas lógicas", debidas, no ya a su incorrecta formalización, sino al incompleto conocimiento de los objetos matemáticos (por ejemplo, *conjunto*, *función*, *secuencia*, *número*, etc.) que se pretendían axiomatizar con la lógica. Una de la más famosas puede que sea la paradoja propuesta por Bertrand Russell (1902) sobre la teoría de conjuntos.

David Hilbert intentó mantener la lógica libre de las dudas y ansiedades de las paradojas "lógicas" desarrollando lo que se denominó "programa de Hilbert" que consistió en una disciplina de austeridad intelectual que impidiera asumir asunciones dudosas. Así, según la propuesta de Hilbert, el formalismo lógico debía ser construido y estudiado desde un punto de vista estrictamente finito en el cual los métodos de demostración usados para establecer las propiedades del formalismo no estarían abiertos a cuestiones sobre su fiabilidad. Así Hilbert, junto con Ackerman, delimitan en 1928 de un modo claro por primera vez la lógica de primer orden, presentan un cálculo deductivo para ella y formulan la pregunta de si dicho cálculo era "semánticamente suficiente" (habría que esperar hasta 1930 para que Gödel respondiera a esta pregunta). Con la seguridad de las herramientas lógicas firmemente establecidas, Hilbert se movió hacia los problemas de la formalización de las teorías matemáticas (por ejemplo de la aritmética, la teoría de conjuntos y el análisis real) y la demostración de su consistencia.

La austera disciplina presentada por Hilbert molestó hasta cierto punto a algunos teóricos que deseaban fortalecer y ensanchar la teoría del cálculo de predicados. Alfred Tarsky (1936) creyó necesario el empleo de toda la teoría general de conjuntos y el sistema de números transfinitos de Cantor (1895, 1897) para la proposición de la correcta formalización de la semántica del sistema formal representando así la totalidad de todos los *modelos* de un cálculo puro dado. Las ideas de Tarsky son aceptadas hoy en día como el análisis adecuado de las nociones intuitivas de *verdad*, *consistencia*, *consecuencia lógica*, etc.

Dos de los más grandes teóricos de la lógica, Herbrand (1930) y Gentzen (1936), encontraron importantes descubrimientos usando metodologías finitistas y nociones puramente

sintácticas, sin entrar, por tanto, en la formalización de Tarsky que ellos veían como extravagante.

Se llegó así a una distinción entre dos ramas de la teoría de la lógica: la teoría de la demostración y la teoría de los modelos. La última concernía con materias tales como modelos, universos (o dominios) de estos modelos, que eran conjuntos no vacíos, *corrección* de los sistemas deductivos, *certeza*, *falsedad*, etc. La teoría de la demostración rehuía estos conceptos y se concentraba en las propiedades "intrínsecas" de las fórmulas y secuentes considerados como objetos en sí mismos, y en su derivabilidad de acuerdo con unos principios de inferencia o con unos procedimientos computacionales dados.

Fue en 1930 cuando Gödel respondió a la pregunta formulada años atrás por Hilbert y Ackerman. En su tesis doctoral demostró la *completitud* del cálculo de predicados, esto es, toda fórmula cierta puede ser probada. De su tesis también se deduce el teorema de Skolem-Löwenheim, que se conoció con anterioridad como resultado de las investigaciones de Löwenheim (1915) y Skolem (1920). El concepto de la *compacidad* fue también establecido por Gödel (1930) para evidenciar que dado un conjunto inconsistente infinito de fórmulas se puede *probar* que lo es. En 1931 Gödel demostró un teorema sorprendente que echó por tierra uno de las aspiraciones del programa de Hilbert, el objetivo de formular, y posteriormente probar su consistencia y completitud, la teoría de los números naturales ("la aritmética"). Gödel mostró que cualquier teoría formalizada de la aritmética debe ser siempre defectuosa en una de las siguientes formas: o es *incompleta*, en el sentido de que una fórmula *cierta* no puede ser *probada*; o es *inconsistente*, en el sentido de que *todas* las fórmulas, ciertas y falsas, pueden ser probadas.

También hay que destacar el trabajo de Gentzen (1936) que investigó profundamente tanto en el campo de la lógica como en el de los fundamentos de las matemáticas. Descubrió la noción de *secuente* y el hecho de que se puede probar que los secuentes son ciertos en demostraciones libres de podas.

El problema de la *decisión*, propuesto inicialmente por Hilbert, quedó resuelto por Church (1936)[1] cuando publicó sus resultados. Estos resultados, que se pueden considerar unos efectos laterales de los obtenidos por Gödel, demuestran que dado un conjunto de fórmulas de la lógica de primer orden (sin identidad) no es decidible el problema de demostrar su consistencia, esto es, no existe un algoritmo que determine en un número finito de pasos si el conjunto de fórmulas es consistente o no. Este resultado conmocionó el mundo de la lógica y su aplicación en las matemáticas.

[1] De forma independiente, A. Turing también obtuvo el mismo resultado.

A partir de este punto en la historia de la lógica los estudios y aplicaciones continuaron hasta hoy en día. En el siguiente capítulo se presentará la rama que apareció posteriormente a esta época y que derivó en lo que actualmente se conoce como el campo del razonamiento automático y, más específicamente, en la demostración automática.

1.2. RAZONAMIENTO AUTOMÁTICO

El primer intento de utilizar el razonamiento automático aplicado a la lógica aristotélica se debe a Ramón Llull y su *Ars Magna* en el siglo XIII [Gardner, 1985]. Esta consistió en un dispositivo mecánico que facilitaba el manejo de sistemas lógicos. Posteriormente, se puede resaltar el intento de la mecanización del cómputo de la máquina aritmética de Pascal y la máquina analítica de Babbage. Ya en la revolución industrial, se pueden encontrar artilugios como el del conde de Stanhope (1753-1816) para resolver silogismos de la lógica tradicional, el piano lógico de W. Jevons o la máquina de Marquarnd de 1882 [Gardner, 1985]. Todas estos dispositivos se pueden relacionar relativamente con la lógica de proposiciones.

Ya más centrado en la lógica de primer orden y durante la década de los cincuenta, con la aparición de los primeros ordenadores digitales, se desarrollaron los primeros programas de demostración automática. Cabe citar los de Gelernter dentro del campo de la lógica geométrica elemental o los de Newell, Shaw y Simon para la lógica de proposiciones. Estos programas se basaban en métodos heurísticos a partir del formato de reducción de problemas [Loveland, 1978] e intentaban modelar la forma de raciocinación humana al enfrentarse a un problema de demostración [Nilson, 1987] [Robinson, 1992].

Hacia 1960 Gilmore, Prawitz y posteriormente Davis y Putman intentaron conseguir procedimientos de demostración para el cálculo de predicados basándose en el teorema de Herbrand. Su método consistía, básicamente, en ir obteniendo de manera progresiva las instancias base de las cláusulas iniciales usando los términos del universo de Herbrand de este conjunto de cláusulas. Cada una de estas instancias se usaba para comprobar su inconsistencia mediante un procedimiento de demostración para lógica proposicional. A pesar, de que el teorema de Herbrand asegura la existencia de un procedimiento, las propuestas de estos investigadores basadas en esta aproximación chocaron frontalmente con lo que se denominó la “explosión combinacional”. Este problema, a pesar de no poner en peligro en teoría la completitud de los procedimientos, conllevaba que la demostración buscada debería ser encontrada después de utilizar miles de millones de combinaciones de instanciaciones, haciendo en la práctica imposible su computación. Así la propuesta inicial de Gilmore se mostró inviable incluso en la demostración de los problemas más sencillos. Davis y Putman introdujeron algunas mejoras incorporando algunos criterios de poda para evitar la generación de información redundante. A estos métodos se los denominó de forma un tanto despreciativa “métodos del

Museo Británico" por la forma que tenían de generar y probar ciega e indiscriminadamente todos las posibles combinaciones esperando encontrar la deseada.

La idea de, en vez de probar con todas las instanciaciones posibles sobre el universo de Herbrand, buscar aquellas que dan lugar a la demostración deseada a base de utilizar el denominado *algoritmo de unificación* ya había sido propuesta escuetamente por Herbrand en su tesis doctoral aunque pasó inadvertida hasta que Prawitz (1960) volvió a redescubrirla. El procedimiento propuesto por Prawitz fue implementado por A. Robinson (1962) y también por M. Davis y sus colegas (1963). Fue Robinson quien finalmente propuso un procedimiento más cercano a la mecanización que denominó *principio de resolución* en cuyo núcleo principal se puede encontrar el algoritmo de unificación. Según sus propias palabras en [Robinson, 1965], "*el único enfoque posible era buscar la manera de restringir el procedimiento de prueba todo lo posible, simplificando la forma de las expresiones en el cálculo de predicados y empaquetando más potencia en las reglas de inferencia, se trataba pues de hacer más eficiente el proceso de búsqueda*". Su método se alejaba del teorema de Herbrand permitiendo deducir consecuencias universales de enunciados universales, evitando de esta forma el problema de la explosión combinacional. Por otro lado, el principio de resolución se concibió como una regla de inferencia más adecuada a la mecanización (ya que subsumía varias reglas de inferencia conocidas como *modus ponens* e *instanciación universal*) que los métodos propuestos anteriormente que eran próximos a la forma deductiva humana, con necesidad de realizar pasos deductivos simples y elementales, que implicaban demostraciones largas y tediosas.

En 1964, L. Wos, G. Robinson y D. Carson programaron en los Argonne Laboratories el primer demostrador basado en el principio de resolución, introduciendo ciertas mejoras en la estrategia de búsqueda (conjunto soporte) que limitaban el espacio de búsqueda. Durante ese mismo año, A. Robinson propuso hiperresolución en su intento de encontrar reglas de inferencia más compactas y próximas a la deducción mecanizada [Robinson, 1965b].

Hacia finales de los sesenta y principios de los setenta aparecieron gran número de resultados sobre procedimientos basados en resolución. En esa época en Edimburgo se formó una escuela con numerosos jóvenes investigadores, como R. Kowalski, P. Hayes, D. Kuehner, G. Plotkin, R. Boyer, J. Moore, D. H. Warren, M. van Emden, R. Hill, etc. Estos investigadores junto con A. Colmerauer, que por la misma época trabaja en Marsella, fueron los iniciadores de lo que posteriormente se denominó la *programación lógica*. Las propuestas de estas dos escuelas resolvieron la dicotomía procedural-declarativa que existía con los investigadores americanos, principalmente representados por J. Mccarthy, M. Minsky y S. Papert del M.I.T. utilizando un tipo especial de cláusulas, las cláusulas de Horn, que podían ser interpretadas de forma procedural y declarativa [Kowalski, 1974] [Robinson, 1992]. Además se demostró que este subconjunto del cálculo de predicados, las cláusulas Horn, (asociado a un demostrador correcto y completo) era un mecanismo de computación de potencia idéntica a

otros paradigmas ya conocidos (la máquina de Turing o λ-cálculo) [Lloyd, 1987]. Este hecho dotaba al cálculo de predicados de mayor relevancia, si cabe, al impedir considerarlo un mero formalismo de representación del conocimiento. En un intento de finalizar con la lucha entre la visión procedural y declarativa la escuela de Edimburgo desarrolló un sistema de resolución lineal para el caso restringido de cláusulas Horn altamente eficiente que se denominó LUSH (posteriormente se le dio el muy conocido nombre de SLD). Hay que destacar que los trabajos de los grupos de Edimburgo y Marsella con la importancia que tienen en la historia de la lógica computacional por proporcionar por primera vez y de forma eficiente una aproximación a la programación declarativa, no se pueden considerar como muy adecuados en el campo de la demostración automática, ya que, por su propia simplicidad, el paradigma de la programación lógica presenta numerosos inconvenientes para el tratamiento de problemas complejos, cuya solución comporta el recorrido de espacios de búsqueda muy grandes.

Por la misma época D. Loveland, que trabajaba junto con M. Davis en New York University desarrollando el método de unificación de este último, concibió de manera independiente su sistema de *Eliminación de Modelos* [Loveland, 1968], un método de razonamiento lineal completamente similar a los sistemas de *resolución lineal* desarrollados por el grupo de Edimburgo y por D. Luckham en Stanford.

Mientras tanto en los Argonne Laboratories, L. Wos y G. Robinson extendieron la aplicabilidad de la unificación a base de aumentar resolución con más reglas de inferencia que estaban especializadas en el razonamiento de la igualdad (*modulación* y *paramodulación*) que implicaron una mejora considerable en la eficiencia de los demostradores que manejaban teorías con igualdad. Estas técnicas junto con otras como el uso de las subsumción [Wos y Overbeek, 1993], el tratamiento de la información intermedia o métodos cuyo poder deductivo pudiera ser variable (*linked-resolution*) se han mostrado imprescindibles para el tratamiento de problemas de complejidad mediana y grande que conllevan espacios de búsquedas de crecimiento exponencial muy rápido y/o demostraciones de profundidad elevada.

También hay que destacar la demostración de lema SAM (acrónimo de *semiauthomated mathematics*) como la primera contribución de la demostración automática a la matemática. El programa usado fue desarrollado por Guard en 1969. Posteriormente se ha seguido utilizando la demostración automática en cuestiones abiertas de las matemáticas como reducciones de conjunto completos.

Basados en las ideas antes apuntadas existen varios demostradores generales para teorías clausales que se han mostrado muy potentes y eficientes. Cabe destacar el Karl Markgraf Refutation (basado en la idea del grafo de conexión [Kowalski, 1975]), el sistema OTTER desarrollado en los Argonne Laboratories [McCune, 1990], SETHEO [Letz, Schuman, Bayerl y Bibel, 1992] basado en la utilización del método de *tableaux*, etc. Por otro lado, han aparecido

últimamente demostradores más cercanos a la programación lógica (PTTP de Stickel [Stickel, 1988], SATCHMO de Bry y Manthey [Manthey y Bry, 1988], nH-Prolog de Loveland [Loveland, 1991], etc.) que son razonablemente eficientes en el tratamiento de problemas de complejidad sencilla o media y que serán ampliamente discutidos en esta tesis.

Por otro lado existen también demostradores basados en técnicas inductivas, como el propuesto por Boyer y Moore, o demostradores de lógicas que no son de primer orden: lógicas modales, temporales, etc. (aunque en el contexto formal de estos últimos hay que destacar que incluso el problema de la unificación sintáctica de términos es indecidible para lógicas de segundo orden o superiores [Knight, 1989]).

Los límites computacionales del razonamiento automático están bastante delimitados. Ciñéndonos a la lógica de primer orden este límite viene dado por la indecidibilidad de la consistencia de un conjunto de fórmulas y de clases importantes de axiomas que formalizan teorías específicas. Hamilton [Hamilton, 1981] menciona que los siguientes conjuntos de axiomas son indecidibles:

Teoría de grupos de primer orden.
- Teoría de anillos de primer orden.
- Teoría de cuerpos de primer orden.
- Teoría de semigrupos de primer orden.
- El sistema ZF (axiomas conjuntistas de Zaermelo-Franklin).

Por otra parte también destaca algunos que son decidibles:

Teoría de grupos abelianos de primer orden.
- Aritmética de primer orden sin multiplicación.

Otro límite, mucho más frecuente, viene dado por la complejidad computacional de los problemas y algoritmos utilizados en su resolución. Por un lado, el crecimiento exponencial inexorable del espacio de búsqueda, por otro el coste temporal de algoritmos intrínsecos al proceso de la demostración: unificación, subsumción, podas en el espacio de búsqueda, etc. Como ejemplo, sólo citar que el problema de determinar de forma general si una fórmula del cálculo proposicional es una tautología es coNP-completo [Garey y Johnson, 1979].

1.3. APROXIMACIONES AL RAZONAMIENTO AUTOMÁTICO

Desde el inicio del razonamiento automático hasta su posterior desarrollo actual han existido distintas aproximaciones al mismo. Estas aproximaciones se pueden agrupar en dos grandes líneas: por un lado, aquéllas que intentan simular la forma de raciocinio del ser humano, llevando sus mecanismos de deducción a los sistemas y métodos desarrollados por éste; por otro

lado están aquéllas que, rechazando los mecanismos humanos por simplistas y poco adecuados para su automatización, promueven la invención y desarrollo de mecanismos específicos que aunque lejanos y poco comprensibles al raciocinio humano se adapten perfectamente a la automatización.

Otra característica distinta que también diferencia unos demostradores de otros es la cantidad de información intermedia retenida. Por un lado, están los que no almacenan ningún tipo de información intermedia, intentando ganar eficiencia y evitar un crecimiento muy rápido del espacio de búsqueda; por otro lado están los que sí almacenan información intermedia. Como un ejemplo de los primeros está resolución SLD donde no se almacena información intermedia (en general las aproximaciones basadas en resolución lineal intentan minimizar la cantidad de información derivada retenida) y en el extremo opuesto esta resolución binaria de Robinson donde toda la información derivada es almacenada como información intermedia. En un punto medio podría esta resolución semántica que sólo retiene información derivada que cumple determinados requisitos.

1.3.1. IMITACIÓN DEL RAZONAMIENTO HUMANO VS. DEMOSTRADORES BASADOS EN RESOLUCIÓN

La obtención de mecanismos, programas, o métodos para el razonamiento automático que imiten la forma del razonamiento humano es la aproximación tomada por la Inteligencia Artificial clásica. Esta aproximación pretende estudiar las formas del razonamiento humano y, posteriormente, imitarlas de forma automática obteniendo de esta forma sistemas que puedan abordar el tratamiento automático de problemas. Esta propuesta, como destaca Wos en [Wos y Overbeek, 1993] cae en el error de tratar de emular el razonamiento humano en vez de estudiar cómo las personas razonan para encontrar claves que permitan incrementar la eficiencia de los programas de razonamiento automático. Una primera razón del porqué no es válida la aproximación de la imitación del razonamiento humano es que las personas y los computadores son entes totalmente distintos, y por tanto la forma de raciocinio de unos no se puede aplicar directamente a los otros, ni viceversa. Un ejemplo de todo esto se puede observar en la regla de inferencia *instanciación*. Si se pretende obtener sistemas que imiten el razonamiento humano, éstos deberían hacer un uso intenso de esta regla de inferencia, muy aplicada por las personas, que como se sabe tiene un poder de deducción relativamente pequeño. La aplicación de esta regla produce la aparición extremadamente rápida del problema de la explosión combinacional, como ya se apuntó anteriormente en este capítulo.

Por otro lado, los sistemas basados en resolución son un excelente ejemplo de la necesidad de enfatizar la obtención de conclusiones generales a partir de premisas generales, aplicando lo menos posible el uso de la instanciación. Así, resolución puede ser vista como una

regla de inferencia potente y bien adecuada a las características de los computadores pero que está alejada de las formas de razonamiento de las personas. Existen otras reglas de inferencia mucho más potentes, como hiperresolución [Robinson, 1965b], ultrarresolucion [Robinson, 1992], modulación, paramodulación [Wos, Robinson, Carson y Shalla, 1967], etc. que son una muestra mucho más extrema de este mismo hecho.

No obstante, como apunta Wos en [Wos y Overbeek, 1993], no hay que desdeñar las formas de razonamiento humano, ya que su estudio puede conducir a encontrar ciertas claves y métodos que aplicados, después de una correcta adecuación, pueden ser esenciales para la obtención de demostradores que pueden abordar con mucha mayor eficiencia problemas de complejidad elevada.

1.3.2. RETENCIÓN DE INFORMACIÓN INTERMEDIA

Como se mencionó anteriormente una característica importante de los sistemas de razonamiento automático es la cantidad de información intermedia que retienen durante el proceso de cómputo. Esta característica es muy importante ya que supone en un extremo el almacenamiento de toda la información derivada intermedia, con la consiguiente explosión del número de expresiones a tratar, y en el otro extremo la ausencia de información intermedia retenida con la consiguiente pérdida de la potencia deductiva del sistema. Ya A. Robinson, descubrió que este problema estaba presente en la aplicación inmediata de la resolución binaria, en lo que se conoce como estrategia de saturación de nivel [Robinson, 1965] [Chang y Lee, 1973] [Loveland, 1978], ya que implicaba la aparición de un número excesivo de información intermedia irrelevante que implicaba un crecimiento desmesurado del espacio de búsqueda. Robinson, que se percató de que la razón de este hecho era que resolución comprendía pasos deductivos muy cortos que permitían la aparición de demasiada información intermedia, desarrolló hiperresolución (una forma especial de resolución semántica) [Chang y Lee, 1973] en la que los pasos deductivos eran mucho más grandes, disminuyendo de esta forma la cantidad de información intermedia retenida.

Si se piensa en la demostración automática en términos algorítmicos, se puede reducir el problema a la exploración de un árbol de estados generados a partir de las diferentes reglas de inferencia. Este hecho deja claro el carácter exponencial del problema de la demostración automática. La eficiencia de un demostrador dependerá fundamentalmente del tamaño del árbol de estados que generan las reglas de inferencia que aplica y los diferentes criterios de poda que utiliza para disminuir su tamaño, preservando evidentemente la corrección y completitud. Estos criterios de poda claramente tienen un coste temporal que se ha de contrapesar con la disminución del espacio que conlleva su aplicación. Generalmente la aplicación de estos criterios de poda supone la retención de información intermedia que permita detectar la aparición de

información irrelevante durante el proceso de cómputo. Del equilibrio entre los costes espaciales del árbol de estados y los costes temporales de los criterios de poda dependerá, posiblemente, el éxito en demostrar un cierto problema. Por otro lado no hay que olvidar que también el problema a resolver tiene su influencia, ya que la complejidad del mismo puede implicar el utilizar ciertos métodos en vez de otros.

Atendiendo al problema de la información intermedia, dos casos extremos son claramente discernibles: por un lado, resolución binaria (estrategia de saturación de nivel) donde toda la información intermedia es retenida y que, como ya se mencionó anteriormente, se mostró inadecuada para tratar incluso problemas triviales; por otro lado, el razonamiento automático basado en la programación lógica que esencialmente comprende el uso de resolución lineal (o más específicamente, variaciones de las misma como, resolución de entrada, Eliminación de Modelos, resolución SLD, etc.) como mecanismo fundamental de deducción. Las ventajas de este último caso recaen en la ausencia de retención de información intermedia con la consiguiente mejora del coste espacial, así como un buen comportamiento temporal al no ser necesario aplicar ciertos criterios de poda como subsumción o canonización de términos iguales. El uso de la programación lógica en el razonamiento automático ha supuesto por otra parte la necesidad de salvar algunos de los problemas que impiden la completitud de la misma, como son la aplicación de una unificación correcta y el uso de otros métodos de exploración del árbol de estados. Por otro lado el uso de otros mecanismos, como la precompilación de los problemas a tratar, ha permitido a algunos investigadores como Stickel [Stickel, 1988] y Loveland [Loveland, 1991] alcanzar logros considerables, al menos en el tratamiento de problemas de complejidad sencilla o media.

El enfoque de la programación lógica ha sido el utilizado para el desarrollo del trabajo de la presente tesis. A pesar de las justas críticas de Wos en [Wos y Overbeek, 1993] es necesario destacar algunas ventajas de este enfoque que justifican su uso:

- Complementa de una forma interesante la programación lógica, corrigiendo ciertos defectos, desde el punto de vista de la demostración de teoremas, que ésta presenta.

- Dada las altas tasas de inferencia, permite abordar problemas de complejidad moderada con costes temporales razonables.

- Como no se retiene información intermedia es sencillo de implementar al no ser necesario el desarrollo de funciones de poda.

- Permite utilizar como herramienta de implementación el lenguaje Prolog, consiguiendo unos desarrollos muy rápidos y fácilmente modificables.

CAPÍTULO 2:

2. RESOLUCIÓN LINEAL Y SUS REFINAMIENTOS

2.1. INTRODUCCIÓN

El trabajo de la presente tesis se circunscribe al marco de la resolución lineal, que apareció como un refinamiento de los demostradores que se basaban en la estrategia de saturación de nivel del principio de resolución. Actualmente existen numerosas propuestas que se basan en la programación lógica que esencialmente se pueden entender como refinamientos de resolución lineal. En el presente capítulo se realizará una exposición detallada de resolución lineal y de sus refinamientos. Previamente a esta presentación se recordará la definición del principio de resolución y de los primeros refinamientos del mismo que existieron.

Todos los demostradores basados en resolución proceden por *refutación*, que se fundamenta en el conocido método de demostración matemático de *reducción al absurdo*. En esencia, estos demostradores realizan la deducción de una fórmula F a partir de un conjunto de premisas S, probando la inconsistencia del conjunto formado por la negación de F y S, esto es, $\{\neg F\} \cup S$. Los principios y características de resolución y sus refinamientos pueden estudiarse en cualquier libro de fundamentos de lógica o de demostración automática como [Chang y Lee, 1973] [Loveland, 1978] [Hamilton, 1981] [Cuena, 1985].

2.2. RESOLUCIÓN

En el año 1960 Gilmore desarrolló un procedimiento de demostración por refutación para teorías clausales basado en el teorema de Herbrand. Este procedimiento se mostró muy ineficiente incluso para problemas muy sencillos. Al poco tiempo Davis y Putman introdujeron un conjunto de refinamientos al procedimientos de Gilmore que esencialmente eliminaban la información redundante de un conjunto de cláusulas libres de variables. A pesar de estas mejoras, estos procedimientos basados directamente en el teorema de Herbrand siguieron siendo muy ineficientes al adolecer del defecto, intrínseco a su forma de funcionamiento, de generar de forma ordenada y sucesiva los conjuntos de instancias base de las cláusulas a partir de los términos del universo de Herbrand del lenguaje de primer orden subyacente. El tamaño de los conjuntos de cláusulas instanciadas base tiene un crecimiento exponencial, por lo que a estos procedimientos se les denominó, de una forma un tanto despectiva, "métodos del Museo Británico" [Robinson, 1979] por la inmensa vastedad del espacio de búsqueda que generaban.

En el año 1963, y basándose en un artículo de Prawitz, A. Robinson concibió la idea de buscar las instancias que permitían obtener el resultado final, en vez utilizar todas las instancias posibles, para encontrar la demostración deseada, evitando así la explosión combinacional asociada al uso de todas las instancias. Esta idea, que ya aparece de manera marginal en la tesis de Herbrand, está subyacente al algoritmo de unificación y fue Prawitz quien la redescubrió posteriormente. Robinson introdujo una nueva regla de inferencia, que denominó *resolución*, que elegantemente captura y generaliza *modus ponens* y el silogismo, permitiendo de esta forma, obtener consecuencias universales a partir de premisas universales. Esta regla, que posiblemente queda un tanto lejana a la forma de razonamiento humano, era muy adecuada para su mecanización en los computadores. La adecuación y compacidad de esta regla queda demostrada por el hecho de que es posible definir un sistema formal sin axiomas, correcto y completo, para cláusulas libres de variables en la que la única regla de inferencia sea la proporcionada por el principio de resolución. Para cláusulas con variables además de resolución es necesario introducir la regla de inferencia *factorización*.

A continuación se va a proceder a dar la definición de la resolución binaria. Esta definición supone que las cláusulas son conjuntos de literales[2] y que un literal está formado por un átomo (negado o no) que a su vez se compone de un símbolo de predicado *n*-ario donde sus *n* argumentos son términos del lenguaje de primer orden subyacente.

[2] La definición de las cláusulas como conjuntos de literales, convencional en la literatura clásica, puede llevar a confusiones al no hacer explícitas ciertas operaciones propias de los conjuntos.

Definición 2.1 (Resolvente) [Loveland, 1978]

Sean A y B dos cláusulas (denominadas *cláusulas padres*), L_1 un literal de A, L_2 un literal de B, η una substitución de renombramiento para A o B y σ un unificador más general (umg) para L_1 y $\neg L_2\eta$, entonces la cláusula

$$R(A, B, L_1, L_2) = (A\sigma - \{L_1\sigma\}) \cup (B\eta\sigma - \{L_2\eta\sigma\})$$

se denomina un resolvente de A y B sobre L_1 y L_2. Si no existe un unificador más general para L_1 y $\neg L_2\eta$, el resolvente no está definido. R se denomina operador de resolución, su aplicación es la operación de resolución. En el caso de que A y B sean base, los literales de éstas sobre los que no se resuelve son denominados *literales padres* de las ocurrencias de los mismos literales en el resolvente.

Teorema 2.1 (Corrección del operador resolución) [Chang y Lee, 1973]

Sean A, B y C tres cláusulas, L_1 un literal de A y L_2 un literal de B, entonces

$$R(A, B, L_1, L_2) = C \quad \Rightarrow \quad \{A, B\} \models C$$

Definición 2.2 (Factor) [Loveland, 1978]

Sea A una cláusula, L_1, L_2 un par de literales de A y σ un unificador para L_1 y L_2, entonces la cláusula $F(A, L_1, L_2) = A\sigma$ es una factor de A.

A F se le denomina el operador de factorización. Se destaca que de acuerdo con esta definición la propia cláusula A es una factor de ella misma.

Teorema 2.2 (Corrección del operador de factorización)

Sea A una cláusula, L_1 y L_2 un par de literales de A y σ un unificador para L_1 y L_2, entonces:

$$F(A, L_1, L_2) = A\sigma \quad \Rightarrow \quad A \models A\sigma$$

Dadas ya las definiciones de los operadores de resolución y factorización es posible definir el *sistema lógico resolución*, así como el concepto de deducción y refutación en este sistema.

Definición 2.3 (Sistema **R**) [Loveland, 1978]

El sistema lógico resolución, sistema **R**, queda definido por los siguientes tres elementos:

1) Lenguaje:
   a) Alfabeto: símbolos de variables, constantes, funciones y predicados; símbolo de las conectivas lógicas ($\neg$, $\vee$); el símbolo cláusula vacía $\square$; paréntesis, llaves y otros signos de puntuación.
   b) Fórmulas: cláusulas son su notación habitual y la cláusula vacía.

2) Axiomas lógicos: ninguno.
3) Reglas de inferencia: resolución y factorización.

Definición 2.4 (R-deducción y R-refutación) [Loveland, 1978]

Dado un conjunto S de cláusulas y una cláusula C, una R-deducción de C, a partir de S, denotada por $S \vdash_R C$, es una secuencia finita de cláusulas $C_1, C_2, \ldots, C_n$ tales que:

1) C_n es C
2) Para cualquier i, $1 \leq i \leq n$, se cumple uno de los siguientes casos:
   a) C_i es un miembro de S
   b) C_i es un factor de C_j, con $j < i$
   c) C_i es un resolvente de C_j y C_k, para $j < i$ y $k < i$

Una R-refutación de S es una R-deducción de $\square$ a partir de S.

La corrección y completitud del principio de Resolución de Robinson quedan demostrados por el siguiente resultado.

Teorema 2.3 (Corrección y completitud del principio de Resolución) [Chang y Lee, 1973] [Loveland, 1978] [Robinson, 1965]

Un conjunto S de cláusulas es inconsistente si y sólo si $S \vdash_R \square$.

La aplicación inmediata del principio de Resolución de Robinson al razonamiento automático dio lugar a los métodos de saturación de nivel. A continuación se presenta una formulación de estos métodos de forma compacta. La corrección y completitud de estos métodos se puede demostrar fácilmente a partir de los teoremas 2.1 al 2.3.

Definición 2.5 (Jerarquía de conjuntos de cláusulas) [Loveland, 1978]

Sea S un conjunto de cláusulas, se define $R^0(S) = S$ y $R^{n+1}(S) = R^n(S) \cup \boldsymbol{R}(\boldsymbol{F}(R^n(S)))$, donde $\boldsymbol{R}(S)$ es el conjunto de todos los resolventes de miembros de S y $\boldsymbol{F}(S)$ es el conjunto de todos los factores de miembros de S.

Cualquier cláusula perteneciente a $R^{n+1}(S) - R^n(S)$ está en el nivel n+1, asignándose a los miembros de S el nivel 0. Obsérvese que $\boldsymbol{F}(S)$ incluye las cláusulas de S, ya que, por la definición de factorización, todo cláusula es un factor de sí misma.

Dada esta definición de jerarquía de conjuntos de cláusulas, es sencillo definir el método de saturación de nivel que comprueba la inconsistencia de un conjunto de cláusulas. Una de las posibles formas de definir este método es la presentada en la siguiente definición.

Definición 2.6 (Método de saturación de nivel) [Chang y Lee, 1973] [Loveland, 1978]

Sea S un conjunto inconsistente de cláusulas. El método de saturación de nivel consiste en computar $R^n(S)$ sucesivamente para n igual a 0, 1, … hasta que la cláusula vacía sea miembro del último conjunto obtenido.

Esta definición se corresponde con una búsqueda en anchura en el árbol de estados dados por los resolventes de los distintos conjuntos. Así definido, es fácil demostrar que el método de saturación de nivel es completo para determinar la inconsistencia de un conjunto de cláusulas.

2.3. REFINAMIENTOS DE RESOLUCIÓN

El método de saturación de nivel, desde un punto de vista experimental, resulta muy ineficiente, lo que le lleva en la práctica a ser incapaz de resolver problemas de complejidad sencilla o media. El principal problema de este método reside en la obtención y almacenamiento de una enorme cantidad de información intermedia que es irrelevante o redundante. La aplicación de las criterios de borrado de Davis y Putman, que se pueden entender como podas del árbol de estados, a los sucesivos conjuntos de cláusulas obtenidos en la jerarquía permiten eliminar parte de la información irrelevante, pero como se apunta en [Chang y Lee, 1973], la propia generación y posterior eliminación suponen de antemano un consumo temporal considerable. A pesar de estos criterios, el crecimiento exponencial de los conjuntos de cláusulas hace, en la práctica, inaplicable este método a problemas de considerable complejidad. Existen números refinamientos de resolución (en realidad, del método de saturación de nivel) conducentes a evitar el aumento excesivo del espacio de búsqueda del método de saturación de nivel. Como acertadamente introduce Loveland en [Loveland, 1978], un refinamiento P’ de un procedimiento de demostración P puede ser o bien una restricción del mismo (en el sentido que genera un subconjunto de las deducciones obtenidas por P) o una estrategia (si se limita a reordenar el orden de desarrollo de las deducciones de P). Esencialmente, se pueden distinguir tres tipos de restricciones de resolución: restricciones por ordenamiento de predicados, literales o cláusulas, restricciones que usan información semántica y restricciones lineales. Dentro de la primera restricción se puede encontrar “*lock resolution*” introducida por Boyer en 1971 [Chang y Lee, 1973]. La aplicación de la segunda restricción da lugar a la *resolución semántica*,

propuesta por Slage en 1967 [Chang y Lee, 1973] y asociada a ella se pueden encontrar la *hiperresolución* de A. Robinson [Robinson, 1965b] y la *estrategia del conjunto soporte* de Wos, Carson y G.A. Robinson [Wos, Robinson, Carson y Shalla, 1967]. La tercera restricción se encuentra en la *resolución lineal*, propuesta de manera separada por Loveland [Loveland, 1968] y Luckham [Luckham, 1970], comprende *resolución de entrada* y *Eliminación de Modelos* entre otras. Hay que destacar que la restricción por ordenamiento se aplica también frecuentemente junto con las otras dos restricciones, por lo que habitualmente no se suele presentar separadamente. Una clasificación de los diferentes refinamientos de resolución se puede encontrar en [Chang y Lee, 1973] [Loveland, 1978].

Aunque el trabajo de la presente tesis se enmarca en la resolución lineal, es interesante exponer, de forma breve, resolución semántica, ya que, primero, algunos refinamientos de resolución lineal se basan en ella; segundo, en algunos mecanismos y criterios de demostradores actuales se puede observar su conexión; y tercero, algunas ideas del presente trabajo se enraízan en las ideas que subyacen a la resolución semántica.

2.3.1. RESOLUCIÓN SEMÁNTICA

La presentación de resolución semántica (ordenada) que se va exponer es informal[3], ya que sólo se pretende dejar claras sus ideas principales de forma sencilla e intuitiva. La idea principal que aparece en la resolución semántica para disminuir el tamaño del espacio de búsqueda es reducir el número de posibles resolventes que se pueden obtener. Para ello se limita el conjunto de cláusulas que pueden ser utilizadas para generar nuevos resolventes a partir de una cláusula. Esta limitación se consigue aplicando dos restricciones:

- *Restricción semántica*: dividiendo el conjunto inicial de cláusulas en dos subconjuntos mediante una interpretación, de forma que las cláusulas ciertas en la interpretación pertenecen a un subconjunto y las falsas al otro. Así, dada una cláusula que pertenece a uno de los dos conjuntos, sólo pueden utilizarse cláusulas del otro conjunto para aplicar la operación de resolución. Como apunta Wos en [Wos y Overbeek, 1993] la elección de la interpretación puede ser crucial, aunque habitualmente se eligen las formadas por todos los literales negativos o positivos ya que la condición de certeza en ellas es fácilmente implementable. Esta restricción es la que da nombre a la resolución semántica.

- *Restricción por ordenamiento*: ordenando los símbolos de predicado del lenguaje subyacente, de forma que posteriormente, cuando se aplique la operación de resolución

[3] Una definición formal de resolución semántica se puede encontrar en [Chang y Lee, 1973].

entre dos cláusulas C_1 y C_2, el símbolo del literal sobre el que se resuelve de C_1 sea mayor que el del literal de C_2 , según la ordenación establecida.

Estos dos restricciones establecen la validez de los resolventes que se pueden generar en la resolución semántica (ordenada). Dicho con otras palabras, solamente aquellos resolventes que cumplan estas dos condiciones van a ser generados y retenidos como información intermedia. Una idea similar a la que se ha expuesto como la restricción semántica es la que Robinson propuso al definir hiperresolución en 1965. En aquellos días Robinson perseguía el objetivo de definir una regla de inferencia más potente que resolución binaria, al englobar en cada aplicación un conjunto de aplicaciones de la última. De esta forma, al ser el número de (hiper)resolventes menor, se conseguía una reducción de la cantidad de información intermedia retenida [Robinson, 1965b].

Los distintos refinamientos y estrategias de resolución semántica pueden ser estudiados en [Chang y Lee, 1973], así como los resultados de corrección y completitud de los mismos.

2.4. RESOLUCIÓN LINEAL

La resolución lineal básica fue introducida en 1970, de forma separada, por Loveland que propuso el término *resolución lineal* [Loveland, 1968] y por Luckham que la denominó *resolución por filtrado de ancestro* [Luckham, 1970].

La restricción que se utiliza en resolución lineal consiste esencialmente en únicamente derivar la cláusula siguiente en una deducción a partir de la que le precede, de ahí su carácter lineal. La cláusula siguiente se puede obtener de la cláusulas que le precede aplicando o factorización o resolución junto con una cláusula que puede pertenecer al conjunto inicial o una cláusula anterior en la deducción. La idea básica es transformar siempre la última cláusula en la deducción para obtener otra más próxima a la meta buscada. Esta forma de proceder, a base de cadenas de pensamientos enlazados, es más próxima a la forma de razonamiento de los humanos. A continuación se presenta la definición de resolución lineal.

Definición 2.7 (Deducción lineal, refutación lineal) [Chang y Lee, 1973] [Loveland, 1978] [Kowalski y Kuehner, 1971]

Sea S un conjunto de cláusulas y C una cláusula. Una deducción lineal de C a partir de S es una secuencia finita de cláusulas $C_1, C_2, \ldots, C_n$ tal que:

1) $C_n = C$
2) $C_1 \in S$, denominándose cláusula raíz o inicial de la deducción.
3) Para i, $1 < i \leq n$, se cumple uno de los siguientes puntos:
   a) C_i es un factor de C_{i-1}, o
   b) C_i es un resolvente de C_{i-1} y un factor de una cláusula de S

c) C_i es un resolvente de C_{i-1} y un factor de C_j, tal que $1 \leq j < i$

Una refutación lineal de S es una deducción lineal de $\square$ a partir de S.

Dada una cláusula C_i , para $i > 1$, que es resolvente de las cláusulas C_{i-1} y una cláusula C (que pertenece a S o es una cláusula anterior en la deducción), a C_{i-1} se le denomina *padre cercano* de C_i y a C se le denomina *padre lejano* de C_i. En el caso a) del punto 3) C_{i+1} se obtiene por resolución de entrada y a la cláusula padre lejano se la denomina *cláusula padre de entrada*. En el caso b) del punto 3) C_{i+1} se obtiene por resolución de ancestro y a la cláusula padre lejano se la denomina *ancestro*.

Uno de los problemas que aparece con el uso de resolución lineal es la necesidad de focalizar la comprobación del problema. Esto es, es necesario que exista una cláusula raíz a partir de la cual se construya el árbol de estados. Para ello se ha de exigir que el conjunto de cláusulas que formaliza el problema a resolver sea mínimamente inconsistente. El siguiente ejemplo permite observar este hecho.

Ejemplo 2.1 (Resolución lineal y conjunto de cláusulas mínimamente inconsistente)

Sea S el siguiente conjunto de cláusulas $\{ p \vee q, \neg p, \neg q, r\}$. Claramente, S es inconsistente y, sin embargo, no existe una refutación lineal de S con r como cláusula raíz.

La siguiente definición presenta el concepto de conjunto de cláusulas mínimamente inconsistente.

Definición 2.8 (Conjunto de cláusulas mínimamente inconsistente) [Loveland, 1978]

Sea S un conjunto de cláusulas. S es mínimamente inconsistente si y sólo si es inconsistente y ningún subconjunto propio de S es inconsistente.

Dada esta definición ya es posible enunciar el resultado de completitud de resolución lineal que se presenta en el siguiente teorema.

Teorema 2.4 (Completitud de resolución lineal) [Chang y Lee, 1973] [Loveland, 1978]

Sea S un conjunto de cláusulas. Si S es mínimamente inconsistente entonces, para todo cláusula C de S, existe una refutación lineal de S en la que C es la cláusula raíz.

Existe otra enunciado de la completitud de la resolución lineal que, a pesar de ser menos compacta, es más habitual.

Teorema 2.5 (Completitud de resolución lineal) [Chang y Lee, 1973]

Sea S un conjunto consistente de cláusulas y C una cláusula. Si $S \cup \{C\}$ es inconsistente, entonces existe una refutación lineal de S en la que C es la cláusula raíz.

El siguiente ejemplo presenta una refutación lineal.

Ejemplo 2.2 (Refutación lineal)

Sea S el siguiente conjunto de cláusulas mínimamente inconsistente:

$C_1 = p \vee q$
$C_2 = \neg p \vee q$
$C_3 = p \vee \neg q$
$C_4 = \neg p \vee \neg q$

La figura siguiente muestra una refutación lineal de S:

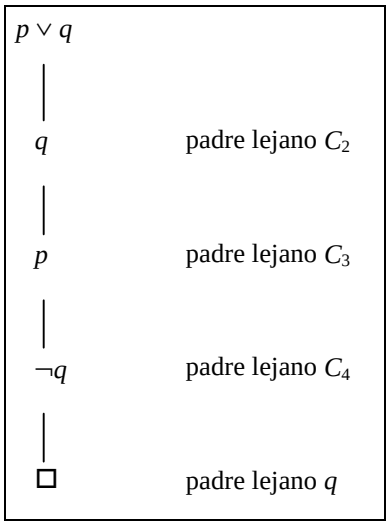


Figura 2.1: Refutación lineal para el Ejemplo 2.2

La cláusula raíz de esta refutación de S es C_1. Es fácil comprobar que también existe refutaciones de S en las que C_2, C_3 y C_4 son las cláusulas raíces.

El principal problema que presenta resolución lineal [Kowalski y Kuehner, 1971] es la redundancia de las demostraciones que aparecen en el espacio de búsqueda, aumentando así su tamaño enormemente. Este problema aparece porque una cláusula $C = L_1 \vee L_2 \vee \ldots \vee L_n$ en una deducción de resolución, que corresponde a un nodo en el árbol de estados, tiene como nodos hijos a todo resolvente de C y cualquier cláusula C', bien del conjunto inicial o bien que sea un ancestro en la demostración, sobre cualquiera de los literales de C. Esto, evidentemente, implica un factor de ramificación del árbol de estados muy alto que provoca, como ya se ha mencionado, un tamaño enorme del espacio de búsqueda. Lo peor de este fenómeno es que esta multiplicidad de resolventes conlleva la aparición de multitud de derivaciones redundantes.

2.5. REFINAMIENTOS DE RESOLUCIÓN LINEAL

Los refinamientos de resolución lineal, que se presentan a continuación, intentaron solucionar los problemas mencionados anteriormente e incrementar la eficiencia mediante la aplicación de diversas restricciones y estrategias. Numerosos refinamientos de estos surgieron durante la década de los 70, cuando se desarrollan refinamientos de cualquier procedimiento completo buscando una mejora de la eficiencia siempre se pretende que éstos también sean completos. A pesar de ello, existen refinamientos de resolución lineal que, aunque incompletos, son suficientemente eficientes y potentes como para demostrar una, o varias, clases de teoremas. En este caso se encuentra dos de los refinamientos que se van a presentar: resolución de entrada y resolución unitaria.

2.5.1. RESOLUCIÓN DE ENTRADA

Resolución de entrada es uno de los refinamientos de resolución lineal más conocidos, al ser el procedimiento SLD [Lloyd, 1987], base del lenguaje Prolog, una restricción del mismo. En esencia resolución de entrada es una restricción de resolución lineal que consiste en utilizar para resolver con cláusulas de las derivaciones únicamente las cláusulas del conjunto inicial. A este conjunto inicial se le denomina conjunto de entrada.

Definición 2.9 (Deducción de entrada, refutación de entrada) [Chang y Lee, 1973] [Loveland, 1978]

Sea S un conjunto de cláusulas y C una cláusula. Una deducción de entrada de C a partir de S es una secuencia finita $C_1, C_2, \ldots, C_n$ tal que:

1) $C_n = C$
2) $C_1 \in S$, denominándose cláusula raíz o inicial
3) Para i, $1 < i \leq n$, C_i es uno de los siguientes:
   a) Un factor de C_{i-1}
   b) Un resolvente de C_{i-1} y un factor de una cláusula de S

Una refutación de entrada de S es una deducción de entrada de $\square$ a partir de S.

Por la equivalencia en lo tocante a los resultados que existe entre la resolución de entrada y la resolución unitaria, es interesante presentar brevemente esta última. Hay que indicar que a pesar de que resolución unitaria no es un refinamiento de resolución lineal, se considera interesante su estudio por la equivalencia antes citada y porque es un claro ejemplo de cómo incrementar la eficiencia de un procedimiento restringiendo la aplicaciones de la resolución binaria y conduciendo las deducciones por el uso de un tipo de cláusulas.

Una *cláusula unitaria* es la que sólo contiene un literal. Un factor de una cláusula que es una cláusula unitaria se denomina *factor unitario*. Introducidos estos dos conceptos, a continuación se presenta la definición de resolución unitaria.

Definición 2.10 (Resolvente unitario, deducción y refutación unitaria) [Chang y Lee, 1973] [Loveland, 1978]

Un resolvente es unitario si al menos una de las dos cláusulas padres es unitaria o un factor de las mismas es unitario.

Sea *S* un conjunto de cláusulas y *C* una cláusula. Una deducción unitaria de *C* a partir de *S* es una R-deducción de *C* a partir de *S* en la que todos los resolventes de la deducción son unitarios. Una refutación unitaria de *S* es una deducción unitaria de □ a partir de *S*.

Resolución unitaria se puede ver como una extensión de la regla de un literal de Davis y Putman para la eliminación de información redundante en el método de saturación de nivel. El siguiente resultado enuncia la equivalencia entre resolución de entrada y resolución unitaria.

Teorema 2.6 (Equivalencia entre resolución de entrada y unitaria) [Chang y Lee, 1973]

Sea *S* un conjunto de cláusulas. Existe una refutación unitaria de *S* si y sólo si existe una refutación de entrada de *S*.

Como se ha comentado anteriormente resolución de entrada y, por el resultado anterior, resolución unitaria no son completas (para el conjunto de cláusulas del ejemplo 2.1 no existe una refutación unitaria o de entrada). Sin embargo, hay que destacar, que estos dos refinamientos de resolución son relativamente eficientes (Np-Hard [Garey y Johnson, 1979]). Por otra parte, es interesante destacar que para la clase de cláusulas Horn ambos refinamientos son completos.

La importancia de resolución de entrada en el campo del razonamiento automático y, posteriormente, como precursora de la programación lógica viene dada por diferentes motivos:

- Es completa para teorías clausales Horn [Loveland, 1978].
- Admite una semántica procedural muy simple, como la dada en [Kowalski, 1974].
- Es sencilla de implementar debido a que no se retiene información intermedia.

Hay que indicar que el problema principal de resolución lineal, la multiplicidad de resolventes de la cláusula en curso en la deducción, se sigue dando en la resolución de entrada, aunque en menor medida ya que las cláusulas que se pueden usar para obtener estos resolventes se limita al conjunto de entrada.

2.5.2. RESOLUCIÓN LINEAL CON FUSIÓN

Este tipo de procedimiento proviene de un refinamiento no lineal de resolución conocido como *resolución con fusión* que posteriormente se verá que se combina bien con resolución lineal.

Definición 2.11 (Resolvente y factor con fusión) [Loveland, 1978] [Nilson, 1987]

Un resolvente con fusión es un R-resolvente que tiene un literal que proviene de la fusión de literales (literal fusionado) con al menos un literal de éstos en cada cláusula padre.

Un factor con fusión es el factor de un resolvente (que no es necesario que sea con fusión) que contiene un literal fusionado con literales padres en una de las dos cláusulas padres.

Claramente de la definición se puede deducir que un resolvente con fusión es también un factor con fusión. Veamos un ejemplo de esta definición.

Ejemplo 2.3 (Resolvente con fusión, factor con fusión)

Sean las siguientes tres cláusulas:

$$C_1 = p \vee q \vee r(x)$$
$$C_2 = \neg p \vee q$$
$$C_3 = \neg p \vee r(a)$$

La cláusula $q \vee r(x)$ es un resolvente con fusión de C_1 y C_2, siendo q el literal fusionado. También, la cláusula $q \vee r(a)$ es un factor con fusión de C_1 y C_3 , siendo $r(a)$ el literal fusionado.

Conocidos ya los concepto de resolución y factor con fusión, se está en disposición de dar la definición de resolución lineal con fusión.

Definición 2.12 (Deducción y refutación lineal con fusión) [Loveland, 1978] [Nilson, 1987]

Sea S un conjunto de cláusulas y C una cláusula. Una deducción lineal con fusión de C a partir de S es una secuencia finita $C_1, C_2, \ldots, C_n$ de cláusulas tal que:

1) $C_n = C$
2) $C_1 \in S$, denominándose cláusula raíz o inicial de la deducción
3) Para i, $1 < i \leq n$, se cumple uno de los siguientes puntos:
   a) C_i es un factor de C_{i-1}
   b) C_i es un resolvente de C_{i-1} y un factor de una cláusula de S
   c) C_i es un resolvente de C_{i-1} y un factor con fusión C_j , tal que $1 \leq j < i$

Una refutación lineal con fusión de S es una deducción lineal con fusión de $\square$ a partir de S.

De la definición se puede observar que la resolución lineal con fusión es una restricción de resolución lineal que consiste en limitar la aplicación de la resolución de ancestro. Así, únicamente los ancestros que son factores con fusión pueden ser usados para realizar esta

operación. Evidentemente, esta restricción no se puede considerar muy significativa ya que la reducción del espacio de búsqueda que provoca no es considerable.

El siguiente teorema establece la completitud de resolución lineal con fusión que se puede encontrar como corolario de un teorema más general en [Loveland, 1978].

Teorema 2.7 (Completitud de resolución lineal con fusión)

Sea S un conjunto de cláusulas y C una cláusula cualquiera de S. Si S es mínimamente inconsistente, entonces existe una refutación lineal con fusión de S cuya cláusula raíz es C.

Como se indica en [Loveland, 1978] la restricción de la resolución lineal con fusión puede ser reforzada exigiendo que el literal sobre el que se resuelve del ancestro sea el literal fusionado.

Ejemplo 2.4 (Refutación lineal con fusión)

Sea S el conjunto mínimamente inconsistente formado por las siguientes cláusulas:

$C_1 = p(x) \vee \neg r(y)$ $\qquad$ $C_4 = \neg p(x) \vee \neg t(y)$

$C_2 = \neg q(x)$ $\qquad$ $C_5 = p(x) \vee q(y) \vee r(z)$

$C_3 = \neg p(x) \vee t(y)$

La siguiente es una refutación lineal con fusión de S:

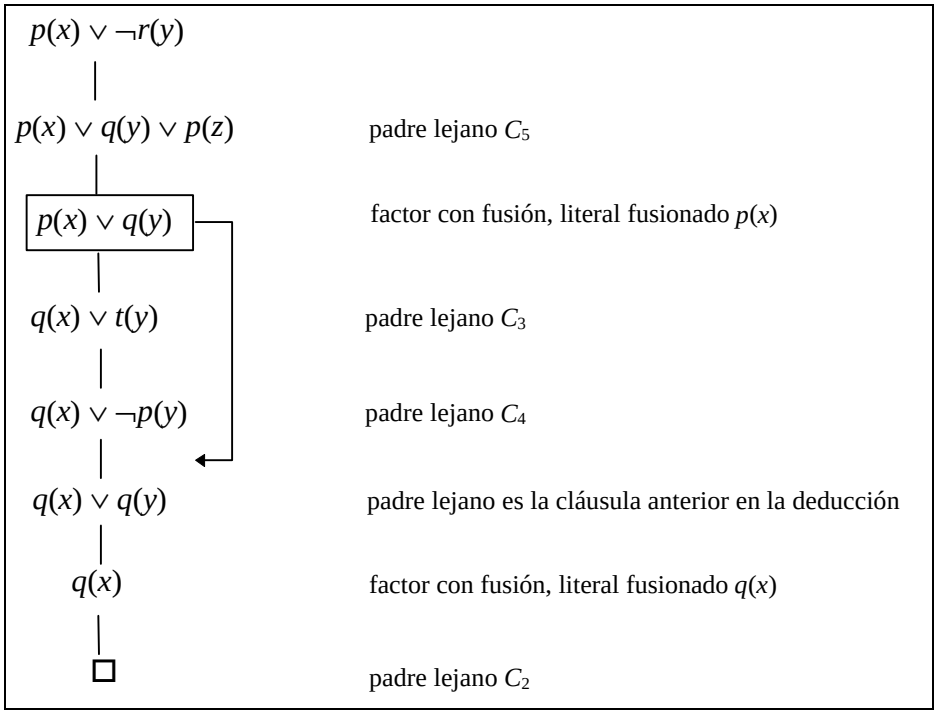


Figura 2.2: Refutación lineal con fusión para el Ejemplo 2.4

Obsérvese que $p(x)$ en la tercera cláusula en la refutación es un literal fusionado de un factor de la segunda cláusula. Esto permite que esta tercera cláusula sea usada como ancestro de la cláusula sexta. Además el literal sobre el que se resuelve es precisamente $p(x)$.

Como se indica en [Marqués, 1992], el uso de la resolución lineal con fusión reduce la ramificación del árbol de estados y generalmente el espacio de búsqueda. Sin embargo, esto no siempre lleva consigo un incremento en la eficiencia, ya que en algunos casos la profundidad de la refutación lineal con fusión que se puede localizar con una cierta cláusula raíz es mayor que la correspondiente refutación lineal. Este problema se ve en el siguiente ejemplo.

Ejemplo 2.5 (Refutación lineal frente a refutación lineal con fusión)

Sea *S* el mismo conjunto de cláusulas que el del Ejemplo 2.4. La siguiente figura muestra una refutación lineal del mismo.

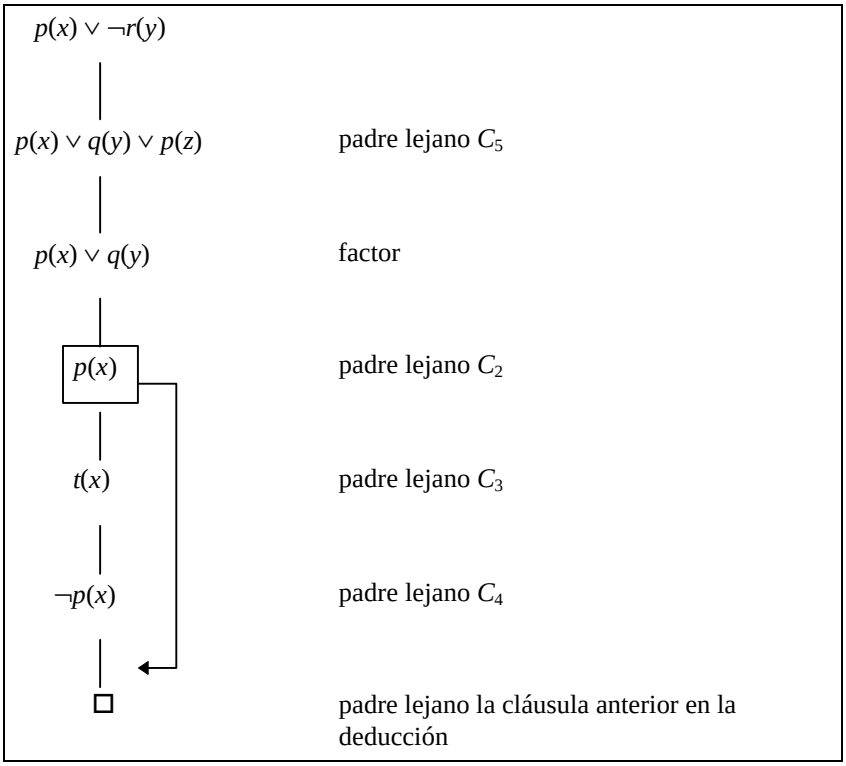


Figura 2.3: Refutación lineal con fusión para el Ejemplo 2.5

Nótese que la cláusula vacía en el último paso de la refutación se obtiene aplicando una resolución de ancestro usando como ancestro la cuarta cláusula en la refutación, $p(x)$, que no es un factor con fusión. Por tanto resolución con fusión no podría aplicar este paso, alargándose la refutación con fusión que se obtiene. Otros ejemplos similares se pueden encontrar en [Loveland, 1978].

2.5.3. RESOLUCIÓN LINEAL CON SUBSUMCIÓN

La resolución lineal son subsumción se abrevia habitualmente como s-resolución lineal. De igual forma se denotará la deducción y refutación lineal con subsumción como s-deducción y s-

refutación lineal. Antes de dar la definición de la misma es necesario introducir el concepto de *subsumción* y *q-subsumción*.

Definición 2.13 (Subsumción) [Loveland, 1978]

Sean A y B dos cláusulas. Se dice que A subsume a B si y sólo si $\forall A \rightarrow \forall B$ es una fórmula válida.

Ya que la subsumción es indecidible, se suele restringir la subsumción a la denominada θ-subsumción que si es decidible.

Definición 2.14 (θ-subsumción) [Loveland, 1978]

Sean A y B dos cláusulas. Se dice que A θ-subsume a B si existe una sustitución θ tal que $A\theta \subseteq B$.

Obsérvese que la θ-subsumción implica la subsumción pero no al contrario. Por ejemplo dos tautologías cualesquiera se subsumen pero no se θ-subsumen para ninguna sustitución θ. Así mismo, es interesante observar que, como consecuencia inmediata de la definición anterior y de la Definición 2.2, una cláusula siempre θ-subsume a todo factor de ella misma.

Definición 2.15 (S-resolvente) [Loveland, 1978]

Un s-resolvente C de cláusula padre cercano C_1 y cláusula padre lejano C_2 es un factor de un resolvente de C_1 y C_2 tal que C θ-subsume a C_1.

Ejemplo 2.6 (S-resolvente)

Sean C_1 y C_2 las siguientes dos cláusulas:

$$C_1 = p(a) \vee \neg q(x)$$
$$C_2 = p(x) \vee p(y) \vee q(x)$$

entonces las cláusulas $p(a)$ y $p(a) \vee p(x)$ son dos s-resolventes de C_1 y C_2.

Definición 2.16 (S-deducción lineal y s-refutación lineal) [Loveland, 1978]

Sea S un conjunto de cláusulas y C una cláusula. Una s-deducción lineal de C a partir de S es una secuencia $C_1, C_2, \ldots, C_n$ de cláusulas tal que:

1) $C_n = C$
2) $C_1 \in S$, denominándose cláusula raíz o inicial de la deducción
3) Para i, $1 < i \leq n$, C_i es uno de los siguientes:
   a) Un factor de C_{i-1}
   b) Un resolvente de C_{i-1} y un factor de una cláusula de S
   c) Un s-resolvente de C_{i-1} y un factor C_j, tal que $1 \leq j < i$

Una s-refutación lineal de *S* es una s-deducción lineal de □ a partir de *S*.

De la definición dada de s-resolución lineal se puede observar que el propósito de la misma, al igual que el de la resolución lineal con fusión, es restringir el número de resolventes que se pueden obtener usando una cláusula anterior de la deducción. El número de resolventes que se evitan de esta forma puede ser elevado [Loveland, 1978], con lo cual el espacio de búsqueda también se reducirá. La idea que subyace a esta restricción es que los resolventes que no son s-resolventes son información intermedia irrelevante ya que son sustituidos por los s-resolventes posteriores en la deducción. Sin embargo el problema asociado a esta resolución, en particular con los s-resolventes, es el alto coste asociado a la obtención de los mismos.

La propiedad de completitud de la s-resolución lineal se enuncia en el siguiente teorema.

Teorema 2.8 (Completitud de la s-resolución) [Loveland, 1978]

Sea *S* un conjunto de cláusulas y *C* una cláusula de *S*. Si *S* es mínimamente inconsistente, entonces existe una s-refutación de *S* cuya cláusula raíz es *C*.

La resolución lineal con fusión y la s-resolución lineal se pueden combinar dando lugar a la s-resolución lineal con fusión a base de restringir la aplicación de resolución de ancestro únicamente a cláusulas que sean un factor con fusión y el resolvente obtenido sea un s-resolvente.

2.5.4. RESOLUCIÓN LINEAL ORDENADA

Como se ha comentado anteriormente, tanto resolución lineal con fusión como s-resolución lineal son dos restricciones de resolución lineal que limitan el uso de la resolución de ancestro. Esto claramente reduce el número de resolventes posibles, decreciendo el factor de ramificación del árbol de estados y reduciéndose, en general, el tamaño del espacio de búsqueda. El refinamiento que se va a presentar a continuación hace uso de la otra restricción que se le puede aplicar a resolución lineal para limitar el número de resolventes posibles de la cláusula en curso en una deducción. Esta restricción consiste en el ordenamiento de las cláusulas, de forma que sólo se puede elegir un literal, el que cumpla determinado criterio en términos del ordenamiento, en cada cláusula en curso sobre el cual se aplicara las operaciones de resolución. Así el número de resolventes que se pueden obtener se reduce drásticamente. El ordenamiento de alguno de los objetos que intervienen en el proceso de demostración, símbolos de predicados, literales o cláusulas, es una restricción habitual en los refinamientos de resolución. En el caso de la resolución lineal con cláusulas ordenadas, la ordenación se impone sobre los literales de las cláusulas. A continuación se introducen los conceptos de cláusula y factor ordenado necesarios para la definición de esta resolución.

Definición 2.17 (Cláusula ordenada) [Chang y Lee, 1973] [Loveland, 1978]

Una cláusula ordenada (de forma abreviada O-cláusula) es una secuencia de literales acorde con una determinada regla de ordenación *O*.

Similarmente, una O-cláusula se puede entender como una aplicación de sus literales sobre los números naturales de forma que se cumple para todo par de literales l_i y l_j que si $l_i < l_j$ entonces $i < j$. Habitualmente esta aplicación se realiza escribiendo los literales más pequeños más a la izquierda de la cláusula. De esta forma se dice que l_i está *más a la izquierda* que l_j o que l_j está *más a la derecha* que l_i .

Las definiciones que se pueden encontrar en la literatura de factor (abreviadamente O-factor) y de resolvente de cláusulas ordenadas difieren de unos autores a otros [Chang y Lee, 1973] [Loveland, 1978]. Una consecuencia de ello es que la resolución ordenada puede aparecer en algunos libros como un procedimiento completo y en otros no. En esta presentación se va a seguir la definición de [Loveland, 1978] ya que posteriormente se utilizará para introducir otros procedimientos.

Definición 2.18 (Factor ordenado)

Dada una O-cláusula *C*, un O-factor de *C* es un factor de *C* tal que uno de los literales unificados es el literal más a la derecha de *C*.

De la definición se puede concluir que un O-factor de un O-factor de una cláusula *C* es también un O-factor de *C*. Así mismo, *C* es un O-factor de sí misma.

Definición 2.19 (Resolvente de O-cláusulas) [Loveland, 1978]

Dadas dos O-cláusulas *A* y *B*, su resolvente $R(A, B, l_i, l_j)$ está definido sólo si l_i y l_j son los literales más a la derecha de las O-cláusulas *A* y *B*.

Hay que tener cuidado con la obtención de resolventes ordenados, ya que, cuando se obtiene una ocurrencia de una O-cláusula los literales iguales son fusionados sobre el que está más a la izquierda, preservando de está forma el orden establecido.

Presentados los anteriores conceptos ya se está en disposición de introducir la resolución con cláusulas ordenadas, dando la definición de deducción y refutación ordenada (de forma abreviada, O-deducción y O-refutación respectivamente), que posteriormente se utilizará en la definición de resolución lineal ordenada.

Definición 2.20 (O-deducción y O-refutación) [Loveland, 1978]

Sea *S* un conjunto de cláusulas y *C* una cláusula. Una O-deducción de *C* a partir de *S* es una R-deducción de *C* a partir de todas las O-cláusulas de *S* permisibles bajo la ordenación *O*,

en la que los factores utilizados son O-factores y los resolventes obtenidos son O-cláusulas de resolventes de O-cláusulas.

Una O-refutación de *S* es una O-deducción de □ a partir de *S*.

La resolución ordenada en general no es completa. Sin embargo, existen algunos órdenes para los cuales la resolución ordenada correspondiente sí que es completa (por ejemplo, la denominada *Lock resolution* de Boyer y Moore [Chang y Lee, 1973] [Loveland, 1978]).

A continuación se presenta la resolución lineal ordenada, dando la definición de deducción y refutación lineal ordenada (de forma abreviada, O-deducción y O-refutación lineal).

Definición 2.21 (O-deducción y O-refutación lineal) [Loveland, 1978]

Sea *S* un conjunto de cláusulas y *C* una cláusula. Una O-deducción lineal de *C* a partir de *S* es al mismo tiempo una deducción lineal y una O-deducción de *C* a partir de *S*.

Evidentemente, ya que la resolución ordenada en general no era completa, tampoco lo es la resolución lineal ordenada. El siguiente ejemplo ilustra este hecho.

Ejemplo 2.7 (Incompletitud de resolución lineal ordenada)

Sea *S* el conjunto de las siguientes cláusulas:

$C_1 = p \vee q$
$C_2 = \neg p \vee q$
$C_3 = p \vee \neg q$
$C_4 = \neg p \vee \neg q$

siendo la ordenación O la dado por el orden de los literales de las cláusulas de izquierda a derecha. Entonces, no existe ninguna O-refutación lineal de *S*. Sin embargo, a continuación se puede observar una O-refutación de *S*.

1.	p	resolvente de C_1 y C_3
2.	$\neg p$	resolvente de C_2 y C_4
3.	□	resolvente de 1. y 2.

2.5.5. RESOLUCIÓN MTOSS Y TOSS

Los procedimientos MTOSS y TOSS son refinamientos de resolución lineal que combinan características de resolución con fusión, s-resolución y resolución ordenada, alterando ligeramente algunos de sus mecanismos. Para el estudio de estos dos procedimientos es necesario introducir algunos conceptos previos.

Definición 2.22 (Reglas de ordenación a izquierdas) [Loveland, 1978]

Una regla de ordenación a izquierda es una regla de ordenación que cumple que los literales de cada resolvente de una derivación lineal que son descendientes del padre cercano están a la izquierda y entre ellos ordenados como lo estaban en la cláusula padre cercano. Los literales descendientes exclusivamente del padre lejano aparecen a la derecha y ordenados entre ellos como disponga la propia ordenación.

Para garantizar la completitud de estos procedimientos con regla de ordenación a izquierdas es necesario extender el conjunto inicial de cláusulas incluyendo, por cada factor *F* de una cláusula *C* de éste y por cada literal *l* de *F*, una O-cláusula cuyo literal más a la derecha sea *l*. Esto es equivalente a considerar el conjunto de todas las contrapositivas del conjunto *S* y de sus posibles factores. La necesidad de incluir todos los factores en el conjunto de cláusulas de entrada se puede observar en el siguiente ejemplo de [Loveland, 1978].

Ejemplo 2.8 (Incompletitud de la O-resolución lineal con contrapositivas sin factores)

Sea *S* el conjunto inconsistente formado por las siguientes cláusulas:

$C_1 = \neg p(x, x)$
$C_2 = p(x, y) \vee q(y, z) \vee p(y, x) \vee q(z, y)$
$C_3 = \neg q(x, x)$

y *O* cualquier regla de ordenación a izquierdas que preserve la alternancia de literales con símbolos de predicados distintos. Se puede comprobar claramente que no existe ninguna O-refutación de *S* si no se incluyen los factores en el conjunto de entrada. Claramente si se utiliza el O-factor de C_2, $p(x, x) \vee q(x, x)$, sí que existe una O-refutación. Evidentemente el problema estriba en la forma en la cual Loveland define el concepto de resolvente de cláusulas ordenadas como ya se comentó anteriormente.

El uso de todas las posibles O-cláusulas que se pueden obtener colocando cada uno de los literales de las O-cláusulas iniciales, esto es de las contrapositivas de las O-cláusulas iniciales, se puede entender en la práctica como si dada una cláusula en curso en la deducción fuera posible usar como cláusula de entrada cualquier cláusula del conjunto de cláusulas de entrada sobre cualquier literal, esté en la posición que esté. De forma distinta se considerarán, sin embargo, la cláusula padre cercano y los ancestros, ya que para éstas si que habrá que aplicar la restricción de la resolución ordenada resolviendo únicamente sobre el literal más a la derecha.

Definición 2.23 (Deducción estricta y débilmente estricta) [Loveland, 1978]

Sea *S* un conjunto de cláusulas y *C* una cláusula. Una R-deducción $C_1, C_2, \ldots, C_n$ de *C* a partir de *S* es estricta si no existe un par de naturales i, j ($1 \leq i < n$, $1 < j \leq n$ e $i < j$) tal que C_i θ-subsume a C_j . Una R-deducción $C_1, C_2, \ldots, C_n$ de *C* a partir de *S* es débilmente estricta si no

existe un par de naturales i, j ($1 \leq i < n$, $1 < j \leq n$ e $i < j$) tal que C_i θ-subsume a C_j y tanto C_i como C_j no contienen variables.

Estas dos propiedades, que son dos restricciones de resolución binaria provenientes de las reglas de simplificación propuestas por Davis y Putman, conducen a la eliminación de información intermedia en el método de saturación de nivel que era irrelevante o redundante. A continuación se van a presentar estas propiedades aplicadas a los procedimientos anteriores.

Definición 2.24 (Reglas de subsumción y subsumción débil) [Loveland, 1978]

Una s-deducción lineal satisface la regla de subsumción si no se obtienen resolventes ordinarios siempre que sea posible obtener un s-resolvente en la deducción. Una s-deducción lineal satisface la regla de subsumción débil si la condición anterior sólo se aplica a cláusulas padres sin variables.

Introducidos estos conceptos ya se está en disposición de presentar la definición de la resolución MTOSS, acrónimo de "Merge Tight Ordered S-lineal deduction with Subsumtion rule".

Definición 2.25 (Deducción y refutación MTOSS) [Loveland, 1978]

Sea S un conjunto de cláusulas y C una cláusula. Una deducción MTOSS de C a partir de S es una s-deducción lineal ordenada de C a partir de S débilmente estricta que satisface la regla de subsumción débil. Además en la deducción se utiliza la regla de ordenación a izquierdas y para la aplicación de la s-resolución cumple que el literal sobre el que se resuelve es un literal descendiente de un literal proveniente de una fusión.

Una refutación MTOSS de S es una deducción MTOSS de □ a partir de S.

Obsérvese que la condición de resolución con fusión se encuentra debilitada en la resolución MTOSS, ya que se permiten resolventes intermedios entre la aparición de la cláusula con el literal fusionado y su uso posterior.

Como se mencionó anteriormente la restricción asociada a la resolución con fusión conduce, con alguna frecuencia, a inhibir ciertos resolventes que podrían acortar la longitud de la deducción. Este hecho lleva a definir la resolución TOSS, acrónimo de "Tight Ordered S-linear deduction with Subsumtion rule", que no incluye la anterior restricción.

Definición 2.26 (Deducción y refutación TOSS) [Loveland, 1978]

Sea S un conjunto de cláusulas y C una cláusula. Una deducción TOSS de C a partir de S es una s-deducción lineal ordenada de C a partir de S débilmente estricta que satisface la regla de subsumción débil en la cual se utiliza la regla de ordenación a izquierdas.

Una refutación TOSS de S es una deducción TOSS de $\square$ a partir de S.

En el siguiente ejemplo se puede observar una refutación TOSS.

Ejemplo 2.9 (Refutación TOSS)

Sea S el conjunto inconsistente formado por las siguientes cláusulas:

$C_1 = \neg p(a)$ $\quad$ $C_4 = \neg p(b) \vee p(c) \vee \neg p(d)$

$C_2 = \neg p(c)$ $\quad$ $C_5 = p(a) \vee p(b) \vee p(c)$

$C_3 = \neg p(b) \vee p(d)$

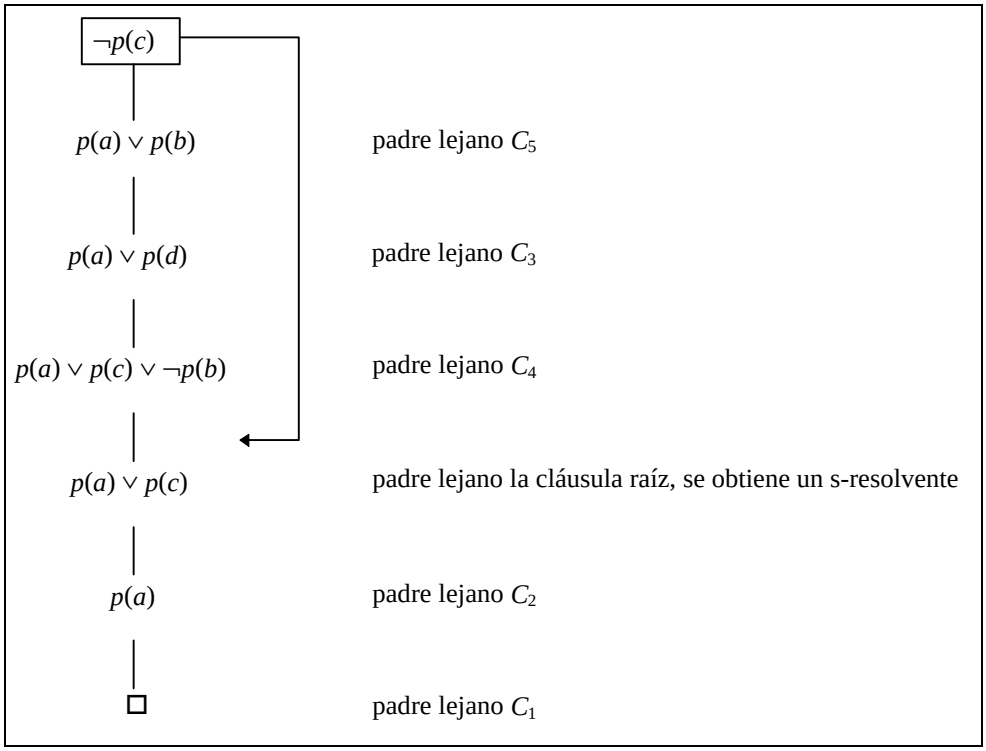


Figura 2.4: Refutación TOSS para el Ejemplo 2.9

La completitud de las resoluciones TOSS y MTOSS se enuncia en el teorema siguiente.

Teorema 2.9 (Completitud de la resolución (M)TOSS) [Loveland, 1978]

Sea S un conjunto de cláusulas, C una cláusula de S y O una ordenación a izquierdas para S. Si S es mínimamente inconsistente entonces existe una refutación (M)TOSS de S cuya cláusula raíz es C usando la regla O.

2.5.6. ELIMINACIÓN DE MODELOS

Tanto la resolución TOSS como la MTOSS han conseguido el objetivo de reducir drásticamente el espacio de búsqueda de resolución lineal. Para ello, se han aplicado

restricciones para limitar la ramificación del árbol de estados que impiden obtener el enorme número de resolventes restringiendo el uso de cláusulas padres lejanos tanto del conjunto de cláusulas de entrada inicial como ancestros. El principal problema asociado a estos dos refinamientos es el elevado coste de la regla de subsumción asociado a la comprobación de si un determinado resolvente es un s-resolvente o no. Este problema lo describe perfectamente D. Loveland [Loveland, 1978]: "sería muy deseable encontrar una forma de modificar la resolución TOSS, manteniendo la condición de subsumción pero evitando al máximo el coste asociado a la comprobación de dicha regla". Con este objetivo Loveland propuso un nuevo procedimiento, basada en resolución lineal, denominado *Eliminación de Modelos* (abreviadamente, ME). Como se verá posteriormente este procedimiento es análogo a otros como son el procedimiento MESON también de Loveland, la resolución SL desarrollada posteriormente por Kowalski y Kuehner, y resolución SL* presentado en el presente trabajo.

El procedimiento de Eliminación de Modelos es equivalente en cierto sentido, que posteriormente se describirá, a la resolución TOSS, aunque el formato de definición sea totalmente distinto ya que Eliminación de Modelos no sigue el esquema de definición de resolución general que se ha dado hasta este momento. La primera diferencia surge con el objeto fundamental de computación. En Eliminación de Modelos no se usan cláusulas sino secuencias de literales de dos tipos: literales enmarcados y sin enmarcar. A estas secuencias de literales se las denomina *cadenas*.

Definición 2.27 (Cadena) [Loveland, 1978]

Una cadena es una secuencia de literales ordenados de izquierda a derecha y estos literales pueden ser de dos tipos: A-literales y B-literales, estando los primeros enmarcados.

Las cadenas que pertenecen al conjunto de entrada no contienen A-literales y se denotan de la forma habitual, esto es siguiendo la notación de las cláusulas aunque a veces por brevedad no se incluyen los símbolos de disyunción. La cadena vacía se representa por □.

Definición 2.28 (Cadena aceptable e inaceptable) [Loveland, 1978]

Una cadena es aceptable si no contiene dos literales del mismo o distinto tipo iguales o complementarios y el de más a la derecha es un B-literal. En caso contrario, una cadena es inaceptable.

Una cadena inaceptable no se ha de entender como irrelevante o redundante, sino como inválida para su tratamiento en Eliminación de Modelos. Existen algunas operaciones que pueden dar lugar a cadenas inaceptables pero éstas a su vez pueden ser transformadas en aceptables. Esta transformación se define como sigue.

Definición 2.29 (Transformación de aceptación) [Loveland, 1978]

Una transformación de aceptación es una aplicación que asocia a una cadena *K* una cadena *K'* de la forma siguiente:

a) Si *K* tiene un B-literal a la derecha de un B-literal idéntico, entonces *K'* no contiene el B-literal más a la derecha.
b) Si *K* tiene un B-literal a la derecha de un A-literal complementario, entonces *K'* no contiene el B-literal más a la derecha.
c) Si *K* tiene un A-literal a la derecha del B-literal más a la derecha y distinto de cualquier otro literal en la cadena o de su complementario, entonces *K'* no contiene ese A-literal.

Ejemplo 2.10 (Transformación de aceptación)

$T(\ p(a)\boxed{p(x)}\ p(a)\ ) = \ p(a)\ \boxed{p(x)}$	por la regla a)
$T(\ \boxed{q(x)}\ p(a)\ \neg q(x)\ ) = \boxed{q(x)}\ p(a)$	por la regla b)
$T(\ \boxed{p(a)}\ \boxed{q(x)}\ ) = \ \square$	por la regla c)
$T(\ \boxed{p(a)}\ \boxed{\neg p(a)}\ )$	inaceptable

Nótese que los símbolos de disyunción no se han incluido.

En el procedimiento de Eliminación de Modelos se definen tres reglas de inferencia que tienen relación con las vistas hasta el momento. La primera, denominada *extensión*, es similar a la operación de resolución usando cláusulas ordenadas, pero además se mantiene enmarcado en el resolvente el literal sobre el cual se aplicó la operación. La segunda, denominada *reducción*, corresponde al colapso de dos literales uno enmarcado y otro no que son unificables y complementarios. Está operación esta relacionada con el uso de un s-resolvente. La tercera es la factorización. Las tres reglas hacen uso de una cadena padre, la cláusula en curso en terminología de resolución lineal, y la primera además también utiliza una cadena del conjunto auxiliar obtenido a partir del conjunto inicial que se denotará por S_E. Este conjunto S_E corresponde al conjunto de todas las contrapositivas de los factores de las cadenas del conjunto inicial (posteriormente se aclarará más la necesidad de este conjunto auxiliar de cadenas). Estas tres reglas se definen a continuación.

Definición 2.30 (Reglas de inferencia de Eliminación de Modelos) [Loveland, 1978]

a) *Extensión*: Sea K_1 una cadena aceptable y K_2 una cadena del conjunto S_E que no comparten variables. Si los literales más a la derecha de K_1 y K_2 son complementarios y existe una sustitución σ que unifica sus átomos, entonces la regla de extensión genera una nueva cadena $T(K_3)$ tal que K_3 se construye tomando $T(K_1\sigma)$, haciendo que el

literal más a la derecha de $T(K_1)$ sea un A-literal, borrando el literal más a la derecha de $T(K_2\sigma)$ y colocando el resto de literales de $T(K_2\sigma)$ a la derecha del nuevo A-literal de acuerdo con alguna regla de ordenación dada. Si $T(K_1\sigma)$ o $T(K_2\sigma)$ no son aceptables o si T modifica alguno de los literales más a la derecha de alguna de las dos cadenas, de forma que ya no son unificables, entonces no existe cadena derivada.

b) *Reducción*: Sea una cadena aceptable K. Si su literal más a la derecha es complementario de un A-literal y existe para los átomos de ambos una sustitución σ, entonces la regla de reducción genera la cadena $T(K\sigma)$.

c) *Factorización*: Sea K una cadena aceptable. Si su literal más a la derecha puede unificarse con otro B-literal de K mediante la sustitución σ, entonces la regla de factorización genera la cadena $T(K\sigma)$.

El procedimiento de Eliminación de Modelos se distingue de los otros refinamientos que se han visto hasta ahora en que usa cadenas como objetos de su computación, en la restricción de cadena aceptable (Definición 2.28), en la transformación de aceptación (Definición 2.29) y en las reglas de inferencia que aplica (Definición 2.30). Tanto la condición de cadena aceptable como la transformación de aceptación esconden en su definición la aplicación de determinadas reglas como fusión, subsumción, obtención de s-resolventes, etc. De esta forma Eliminación de Modelos incorpora los mecanismos vistos anteriormente en los otros refinamientos para mejorar la eficiencia y reducir el espacio de búsqueda. De la definición dada de las reglas de inferencia se puede comprobar que hacen un uso intenso de la condición de ser aceptable una cadena y de la transformación de aceptación. Además posteriormente se presentará y discutirá el refinamiento de Eliminación de Modelos, denominado *Eliminación de Modelos débil* propuesta también por Loveland [Loveland, 1978] que está basado en la modificación de estos conceptos. Por todo ello es interesante profundizar más en el análisis de la propiedad de ser aceptable una cadena y en la transformación de aceptabilidad.

Primero se analizarán cada uno de los casos de cadenas que ni son aceptables ni se les puede aplicar la transformación de aceptabilidad para obtener una cadena aceptable. Estas cadenas siempre son las últimas en las deducciones, esto es, cuando una de estas cadenas aparece, la deducción termina al ser imposible la aplicación de ninguna regla de inferencia. A continuación se presentan estos casos:

1) Cadenas que contienen B-literales complementarios. Este caso corresponde a la simplificación de la regla de tautología cuya aplicación es necesaria en algunos refinamientos de resolución.

2) Cadenas que contienen A-literales idénticos. Este caso corresponde a la aplicación de una regla de eliminación de información intermedia por subsumción.

3) Cadenas que contienen A-literales complementarios. Este tipo de cadenas no pueden darse en las deducciones ya que la aplicación de la transformación de aceptabilidad elimina estos literales.

4) Cadenas que contienen un B-literal a la izquierda de un A-literal y ambos son idénticos. Este tipo de cadenas no puede aparecer en las deducciones por el mismo motivo que el caso anterior.

5) Cadenas que contienen un B-literal a la izquierda de un A-literal complementarios entre sí. Este caso, similar al 1), corresponde a la eliminación de tautología.

6) Cadenas que contienen un B-literal a la derecha de un A-literal y ambos son idénticos. Este caso, que corresponde a la aplicación de la regla de subsumción, es similar al 2) pero se aplica anticipadamente.

A continuación se presentan los casos de cadenas inaceptables pero a las que se les puede aplicar la transformación de aceptabilidad para obtener cadenas aceptables.

1) Cadenas que contienen dos B-literales idénticos. La transformación de aceptabilidad elimina el que se encuentra más a la derecha. Esto se corresponde con la operación de fusión.

2) Cadenas que contienen un B-literal a la derecha de un A-literal y ambos son complementarios. En este caso la aplicación de la transformación de aceptabilidad elimina el B-literal. Esta aplicación es un caso particular de la regla de reducción que posiblemente se propone para mejorar la eficiencia.

3) Cadenas que contienen un A-literal que está a la derecha del B-literal más a la derecha y que es distinto de cualquier otro literal de la cadena o su complementario (si no se cumple esto último se está en los casos 4) o 5) anteriores). La transformación de aceptabilidad lo elimina ya que es información intermedia irrelevante.

Para dar correctamente la definición del procedimiento de Eliminación de Modelos primero hace falta precisar las características de la regla de ordenación que se va a utilizar y del conjunto auxiliar S_E que se introdujo intuitivamente con anterioridad. La regla de ordenación que se puede utilizar en Eliminación de Modelos es cualquier regla de ordenación a izquierdas, que ya se presentó en las resoluciones MTOSS y TOSS (Definición 2.25). El conjunto de cadenas auxiliar S_E es habitualmente el conjunto de todos las cadenas que se pueden obtener a partir de los factores de las cláusulas del conjunto S (equivalente a las cláusulas de entrada), de modo que para cada literal de cada uno de estos factores, existe una cadena en S_E cuyo literal más a la derecha es este literal. Como ya se ha mencionado anteriormente, la construcción de S_E

es equivalente al conjunto de contrapositivas (véase [Loveland, 1978]) de los distintos factores del conjunto de cláusulas de entrada. Así definido este conjunto inicial, se demuestra que el procedimiento de Eliminación de Modelos es completo como posteriormente se enunciará. A continuación se presenta la definición de deducción de Eliminación de Modelos (abreviadamente, deducción ME).

Definición 2.31 (Deducción ME y refutación ME) [Loveland, 1978]

Una deducción ME de una cadena K a partir de un conjunto de cláusulas S, dada una regla de ordenación O, es una secuencia $K_1, K_2, \ldots, K_n$ de cadenas aceptables, tal que:

1) $K_n = K$
2) $K_1 \in S$, denominándose cadena raíz o inicial de la deducción
3) Para i, $1 < i \leq n$, K_i es una cadena derivada a partir de K_{i-1} utilizando una de las siguientes reglas de inferencia:
   a) Una extensión, utilizando una cadena auxiliar de S_E
   b) Una reducción
   c) Una factorización

Una refutación ME de S es una deducción ME de $\square$ a partir de S.

Ejemplo 2.11 (Refutación ME)

Sea S el mismo conjunto de cláusulas que el del Ejemplo 2.9. A continuación se presenta una refutación ME para dicho ejemplo.

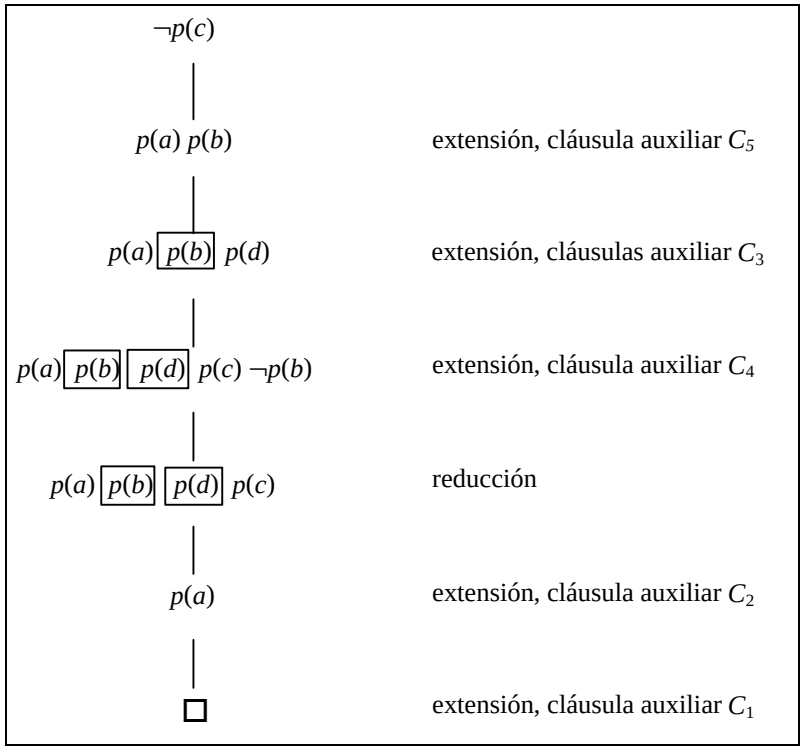


Figura 2.5: Refutación ME para el Ejemplo 2.11

Si se comparan la refutación TOSS del Ejemplo 2.9 y la refutación ME del Ejemplo 2.11 se puede observar que, eliminando los literales enmarcados de la refutación ME, son equivalentes. Las aplicaciones de la regla de extensión se corresponden con la de resolución de entrada. La obtención del s-resolvente en la refutación TOSS equivale a la aplicación de la regla de reducción de la refutación ME.

En general las deducciones TOSS y ME no son equivalentes. Sin embargo, para el caso libre de variables se demuestra la equivalencia de ambos procedimientos. A continuación se estudia dicha equivalencia y para ello, primero, es necesario definir una aplicación, que se denotará *M*, que dada una cadena devuelva la O-cláusula correspondiente a base de eliminar los A-literales de la cadena, respetando el resto de B-literales en el orden en el que se encuentra. Supóngase una cierta regla de ordenación a izquierdas *O*, entonces:

a) *Aplicación de la regla de extensión*: sean dos cadenas aceptables y libres de variables K_1 y K_2. Si K_3 es la cadena resultado de aplicar la regla de extensión usando K_1 y K_2, entonces $M(K_3)$ corresponde al resolvente ordenado de las cláusulas ordenadas $M(K_1)$ y $M(K_2)$. Fácilmente se puede comprobar este hecho a partir de las definición de reglas de ordenación a izquierdas y por la forma de las propias operaciones implicadas.

b) *Aplicación de la regla de reducción*: sea K_1 una cadena aceptable y libre de variables de la forma siguiente,

$$L_1\, L_2\, \ldots\, [L_i]\, \ldots\, L_n$$

siendo L_i y L_n dos literales complementarios y sus átomos unificables. El resto de literales de la cadena son de tipo A o B. Después de la aplicar la regla de deducción la cadena que se obtiene es $L_1\, L_2 \ldots [L_i] \ldots L_{n-1}$ que se denominará K_3. Por las características de la deducciones ME y de la regla de ordenación *O* se cumple que ha de existir una cadena anterior en la deducción K_2 libre de variables cuya forma es $L_1\ L_2 \ldots L_i$, de forma que esta cadena K_2 coincide con la cadena K_1 en sus i-1 primeros literales y el i-ésimo literal de la cadena K_2 es de tipo B e igual al i-ésimo literal de la cadena K_1 que es tipo A.

Si se considera las O-cláusulas $M(K_1)$ y $M(K_2)$, es evidente que un resolvente ordenado de las mismas con padre cercano $M(K_1)$ y padre lejano $M(K_2)$ es precisamente $M(K_3)$ donde se produce la fusión de los i-1 primeros literales de ambos antecesores. Además $M(K_3)$ es un s-resolvente ya que subsume a $M(K_1)$.

c) *Aplicación de factorización*: está regla de inferencia es idéntica en ambos procedimientos por tanto se puede fácilmente comprobar que su aplicación preserva la equivalencia entre ME y TOSS.

Una demostración completa de que para toda deducción ME con cadenas libres de variables existe una deducción TOSS equivalente se puede realizar por inducción en la longitud de las cadenas de ME. En los puntos anteriores se ha presentado, de manera ligeramente informal, la forma de transformar las reglas de inferencia de ME a las operaciones de la resolución TOSS. Sin embargo, existen aplicaciones de las operaciones de la resolución TOSS para las cuales no se puede encontrar una aplicación equivalente de las reglas de inferencia de ME. En particular, existen pasos en la resolución TOSS en los que se obtiene un s-resolvente aplicando resolución de entrada, lo cual no supone la existencia de una reducción en ME. El siguiente ejemplo presenta este caso.

Ejemplo 2.12 (Obtención de s-resolvente TOSS que no es equivalente a la aplicación de reducción de ME)

Dadas las siguientes dos O-cláusulas:

$$C_1 = p(a) \vee p(b) \vee p(c)$$
$$C_2 = p(a) \vee p(b) \vee \neg p(c)$$

la segunda cláusula en la deducción TOSS con cláusula raíz tanto C_1 como C_2 es el s-resolvente $p(a) \vee p(b)$. Sin embargo, en la deducciones ME no se aplica ninguna reducción.

Sin embargo, los pasos de la resolución TOSS en los que se obtiene un s-resolvente mediante resolución de ancestro sí que implican la existencia de una reducción equivalente, en cierta forma. A continuación se analiza con más detalle este caso.

Supóngase una deducción TOSS formada por la secuencia C_1, C_2, …, C_r, de forma que aparece un s-resolvente C_k cuyo padre lejano es el ancestro C_i, como se puede ver gráficamente a continuación:

$C_i = L_1 \vee \ldots \vee L_m$	ancestro
$C_{i+1} = L_1 \vee \ldots \vee L_{m-1} \vee L_{m+1} \vee \ldots \vee L_p$	siguiente cláusula en la derivación
…	
…	pasos de resolución usando
…	cláusulas del conjunto de entrada
…	
$C_{k-1} = L'_1 \vee \ldots \vee L'_n$	padre cercano

como C_k es un s-resolvente cuyo ancestro es C_i los literales L_m y L'_n han de ser complementarios y sus átomos unificables. Además C_k tiene que θ-subsumir a su padre cercano C_{k-1} por tanto, ya que las cláusulas están libres de variables, los literales $L'_1,\ldots, L'_{n-1}$ se han de fusionar con literales de $L_1,\ldots, L_{m-1}$. Por tanto C_k ha de tener la forma $L_1 \vee \ldots \vee L_{m-1}$.

Ahora es posible construir la correspondiente deducción ME:

$K_i = L_1 \ldots L_m$
$K_{i+1} = L_1 \ldots L_{m-1}\, [L_m]\, L_{m+1} \ldots L_p$
…
… extensiones usando las mismas
… cláusulas de entrada
…
$K_{k-1} = L'_1 \ldots L'_n$

de manera que en la cadena K_{k-1} el A-literal $[L_m]$ ha de aparecer entre los n-1 primeros literales. Como los literales L_m y L'_n son complementarios y unificables es posible aplicar una reducción obteniendo la cadena K_k equivalente al s-resolvente C_k de la deducción TOSS.

Si para el caso de cláusulas y cadenas libres de variables se ha demostrado informalmente la equivalencia entre la resolución TOSS y el procedimiento ME, esto no se cumple para el caso en el cual aparecen variables. El siguiente ejemplo tomado de [Loveland, 1978] muestra este hecho.

Ejemplo 2.13 (Deducción TOSS que no es equivalente a deducción ME)

Dado el siguiente conjunto de O-cláusulas:

$C_1 = \neg p(x) \vee \neg q(x)$ $\quad$ $C_3 = \neg p(x) \vee q(f(x))$
$C_2 = p(x) \vee \neg q(x)$ $\quad$ $C_4 = p(x) \vee q(x)$

a continuación se puede observar una refutación TOSS del mismo con cláusula raíz C_4

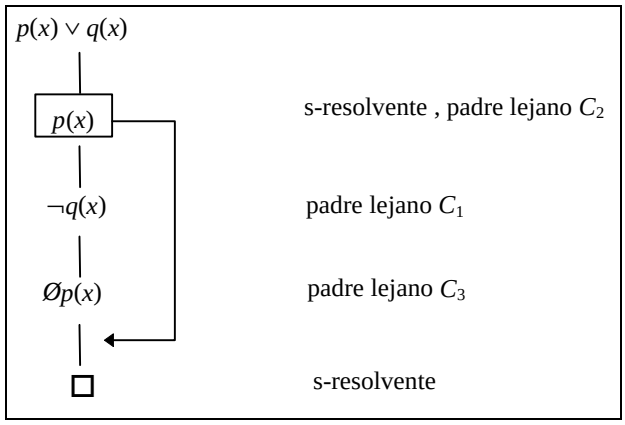


Figura 2.6: Refutación TOSS para el Ejemplo 2.13

Usando el mismo conjunto de cláusulas y partiendo de la misma cláusula inicial, la refutación ME que se obtiene es la siguiente:

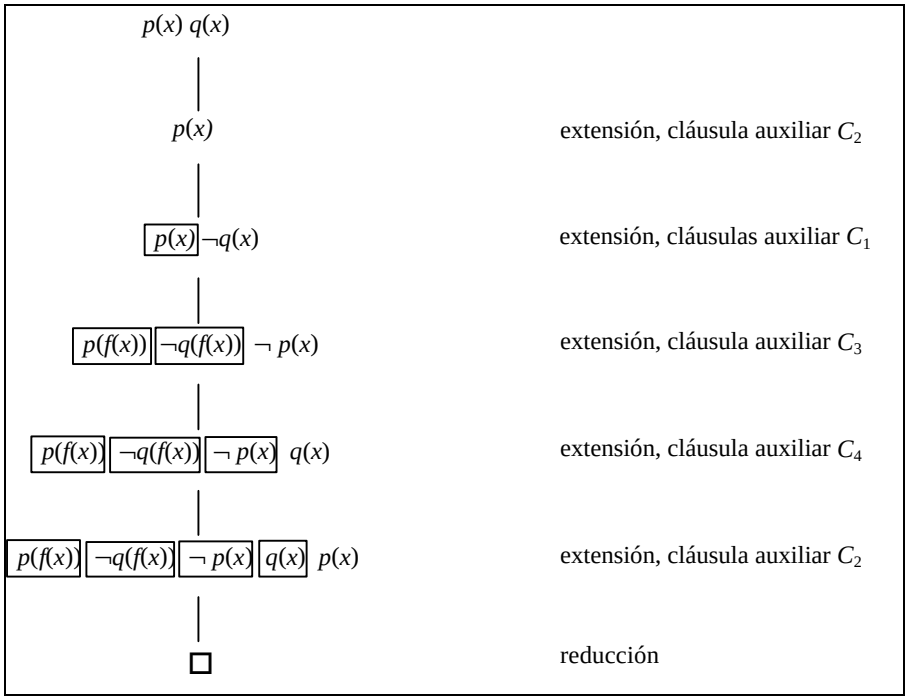


Figura 2.7: Refutación ME para el Ejemplo 2.13

La diferencia entre ambas refutaciones estriba en el uso del s-resolvente que se obtiene en el segundo paso de la refutación TOSS y que posteriormente es usado en el quinto y último paso. Debido a la instanciación de las variables durante la deducción ME se puede observar que no es posible aplicar una reducción a la cuarta cadena, impidiendo de esta forma la obtención de una deducción equivalente a la TOSS.

Los dos teoremas siguientes enuncian la corrección y completitud del procedimiento ME. El establecer el resultado de corrección es necesario ya que ME no es un refinamiento de resolución lineal.

Teorema 2.10 (Corrección de ME) [Loveland, 1978]

Sea *S* un conjunto de cláusulas. Si existe una refutación ME de *S*, entonces *S* es inconsistente.

Teorema 2.11 (Completitud de ME) [Loveland, 1978]

Sea *S* un conjunto de cláusulas, *K* una cadena cualquiera obtenida a partir de una cláusula de *S* y *O* un regla de ordenación a izquierdas para *S*. Si *S* es mínimamente inconsistente entonces existe una refutación ME de *S* cuya cadena raíz es *K*.

El principal problema del procedimiento ME reside en la necesidad del uso de la factorización, tanto del conjunto inicial de cláusulas como de las cadenas intermedias de las derivaciones para lograr la completitud del mismo. Esta necesidad puede conllevar un alto coste temporal, en el cómputo de los posibles factores de las cadenas o cláusulas, así como un alto

coste espacial, al aumentar el número de los posibles cláusulas de entrada a utilizar en los pasos de aplicación de la extensión, incrementando de este modo la ramificación del árbol de prueba. Este problema llevó a D. Loveland a proponer una versión del procedimiento ME denominada *Eliminación de Modelos débil*. Esta versión es completa a pesar de no necesitar la aplicación de la factorización. Este nuevo procedimiento se puede considerar más próximo al formato de resolución de problemas clásico y cuya implementación siguiendo las ideas de ME se denomina *MESON*, procedimiento que será discutido con profundidad más adelante en este capítulo.

El procedimiento de Eliminación de Modelos débil presenta las siguientes diferencias con respecto a ME, visto anteriormente:

1) Las cadenas que aparecen en sus deducciones han de cumplir una condición, distinta de la condición de aceptabilidad del procedimiento ME. Esta condición se presenta a continuación:

Definición 2.32 (Cadena admisible)

Una cadena es admisible si y sólo si:

a) Los B-literales complementarios están separados por un A-literal.
b) Ningún B-literal aparece a la derecha de un A-literal idéntico a él.
c) Ningún A-literal es idéntico o complementario a otro A-literal.
d) El literal más a la derecha es un B-literal.

Las operaciones y conceptos subyacentes a la condición de admisibilidad son similares a las que se discutieron en la condición de aceptabilidad de una cadena. Solamente destacar que, a diferencia de las cadenas aceptables, las cadenas admisibles permiten la presencia de ciertas tautologías (punto a) de la definición anterior) que son necesarias para lograr la completitud del procedimiento. También se puede destacar otra diferencia que consisten en que la condición de admisibilidad permite la existencia de un B-literal a la derecha de un A-literal complementario, hecho por otra parte intranscendente ya que la aplicación de una reducción conduce al mismo resultado que producía la aplicación encubierta de esta reducción hecha por la transformación de aceptabilidad. Quizá la diferencia más importante sea que la condición de admisibilidad permite la existencia de B-literales idénticos. Estos B-literales idénticos desaparecían en el procedimiento ME al aplicar la transformación de aceptabilidad mediante la fusión de los mismos. La posibilidad de la aparición de estos B-literales idénticos es lo que permite al procedimiento de Eliminación de Modelos débil ser completo sin la necesidad de aplicar la factorización, según se destaca en [Loveland, 1978].

2) Se define una aplicación entre cadenas *T*, denominada *transformación de admisibilidad*, que consiste solamente en eliminar los A-literales situados a la derecha del B-literal más a la derecha.

3) Las únicas reglas de inferencia son la extensión y la reducción cuya definición coincide totalmente con las vista para el procedimiento ME.

A continuación se presentan las definiciones de deducción y refutación ME débil.

Definición 2.33 (Deducción y refutación ME débil)

Una deducción ME débil de una cadena K a partir de un conjunto de cláusulas S, dada una regla de ordenación O, es una secuencia $K_1, K_2, \ldots, K_n$ de cadenas aceptables, tal que:

1) $K_n = K$
2) $K_1 \in S$, denominándose cadena raíz o inicial de la deducción
3) Para i, $1 < i \leq n$, K_i es una cadena derivada a partir de K_{i-1} utilizando una de las siguientes reglas de inferencia:
   a) Una extensión, utilizando una cadena auxiliar de una cláusula de S
   b) Una reducción

Una refutación ME débil de S es una deducción ME débil de $\square$ a partir de S.

El siguiente ejemplo, similar al Ejemplo 2.8, muestra como Eliminación de Modelos débil no necesita la factorización al no utilizar la fusión en la aplicación de sus reglas de inferencia, siendo de esta forma un procedimiento completo.

Ejemplo 2.14 (Eliminación de modelos débil y factorización)

Sea S el conjunto inconsistente formado por las siguientes cláusulas:

$$C_1 = \neg p(x, x)$$
$$C_2 = p(x, y) \vee q(y, z) \vee p(y, x) \vee q(z, y)$$
$$C_3 = \neg q(x, x)$$

utilizándose una la regla de ordenación a izquierdas que preserve la alternancia de literales con símbolos de predicados distintos. El procedimiento ME necesita el uso de factores de la segunda cláusula para poder obtener una refutación. Ello es debido a que la aplicación de la transformación de aceptabilidad a la cláusula instancia de C_2 por la substitución correspondiente a la unificación de la resolución realiza una fusión a izquierdas de los B-literales idénticos impidiendo de esta forma la extensión inicial. A continuación puede observarse una refutación ME débil de S.

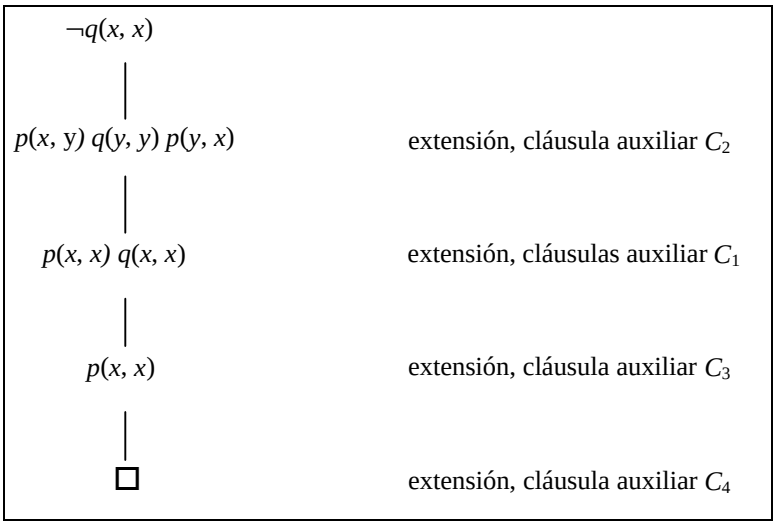


Figura 2.8: Refutación ME débil para el Ejemplo 2.14

Obsérvese que la factorización no ha sido necesaria al no aplicarse la fusión en la primera extensión de la refutación.

El procedimiento de Eliminación de Modelos débil es correcto y completo. Estas propiedades se pueden obtener como corolarios de los teoremas correspondientes del procedimiento MESON que se verá posteriormente, ya que MESON está basado en Eliminación de Modelos débil.

Antes de continuar presentando otros refinamientos es interesante realizar un comentario a cerca del nombre que Loveland dio a su procedimiento. Como él mismo comenta en [Loveland, 1978] estudiando los ejemplos y las demostraciones de ME se puede observar que cada intento de construir una refutación ME conduce a una cadena donde los A-literales forman un conjunto consistente de literales que en su forma maximal da lugar a un modelo del conjunto de cláusulas si éste es consistente. Para un conjunto inconsistente de cláusulas esto obviamente no ocurre; cada serie de aplicaciones consecutivas de extensiones que definen un conjunto consistente de A-literales, debe terminar en la eliminación de A-literales antes de que se construya el modelo completo. El borrado de A-literales, generado a raíz de la aplicación de una reducción, coincide con el reconocimiento de la imposibilidad de construir un modelo (esto es, se *elimina* un modelo posible) incrementando la lista de A-literales. De esta forma, este proceso puede ser visto como una sucesión de intentos y, en el caso de un conjunto inconsistente de cláusulas, *eliminaciones* de construir un modelo. Esta es la motivación que Loveland da para el nombre de su procedimiento.

2.5.7. RESOLUCIÓN LINEAL CON FUNCIÓN DE SELECCIÓN (RESOLUCIÓN SL)

Resolución SL fue introducida por Kowalski y Kuehner en 1970 [Kowalski y Kuehner, 1971] y habitualmente se presenta en literatura como una variante mínima de eliminación de modelos, aunque existen ciertos matices que se han considerado suficientemente interesantes como para realizar su exposición. La terminología y objetos utilizados por Kowalski y Kuehner en la presentación de resolución SL es la misma que la que usó Loveland en su procedimiento Eliminación de Modelos. La idea más novedosa de resolución SL reside en el hecho de utilizar una función de selección que escoge de la cadena en curso el único literal sobre el cual se aplicarán extensiones.

Antes de presentar resolución SL es interesante introducir un refinamiento de resolución lineal denominado t-resolución lineal [Kowalski y Kuehner, 1971], aunque sólo se pueda aplicar a conjunto de cláusulas libres de variables. El interés de este refinamiento reside en la facilidad con que permite ver la ligazón entre s-resolución y resolución SL o Eliminación de Modelos. A continuación se dan unas definiciones previas necesarias.

Definición 2.34 (Descender de, A-ancestro, A-literal) [Kowalski y Kuehner, 1971]

Sea S un conjunto de cláusulas y la secuencia de cláusulas C_1, C_2, …, C_n un derivación lineal a partir de S. Un literal L de C_i desciende del literal L de un ancestro C_j si y sólo si L ocurre en cada cláusula intermedia C_k, $j \leq k \leq I$. Un ancestro C_j de C_i es un A-ancestro de C_i si y sólo si C_{j+1} tiene una cláusula padre de entrada y todos los literales de C_j excepto el literal L sobre el que se resuelve para obtener C_{j+1}, tienen descendientes en C_i. El literal L se denomina el A-literal de C_i proveniente del A-ancestro C_j.

La idea que subyace a estas definiciones es fácilmente explicable: un A-ancestro C' de una cláusula C y el A-literal correspondiente L indican que los literales de C que no se encuentran en C' se han obtenido a partir de una derivación parcial del literal L. A continuación se presenta la definición de t-deducción lineal.

Definición 2.35 (t-deducción lineal, t-refutación lineal) [Kowalski y Kuehner, 1971]

Sea S un conjunto de cláusulas base y C una cláusula base. Una t-deducción lineal de C a partir de S es una secuencia de cláusulas C_0, C_1,…, C_n tal que esta secuencia es una deducción lineal de C partir de S y además se cumplen las siguientes tres restricciones:

1) Si C_{i+1} se obtiene por resolución de ancestro, entonces se obtiene empleando un A-ancestro de C_i.
2) Si C_i contiene un literal complementario de uno de sus A-literales, entonces C_{i+1} se obtiene por resolución de ancestro.
3) Los átomos de los A-literales de C_i que provienen de diferentes A-ancestros son distintos.

Una t-refutación lineal de S es una t-deducción lineal de $\square$ a partir de S.

Es conveniente destacar ciertos aspectos interesantes de t-resolución lineal relativos a su relación con otros refinamientos ya vistos para que posteriormente quede claramente establecido el puente entre s-resolución lineal y Eliminación de Modelos o resolución SL. Estos aspectos son los siguientes:

- La condición 1) tiene como consecuencia que si C_i resuelve con un A-ancestro C_j entonces el literal sobre el que se resuelve es el A-literal de C_i proveniente de C_j. Si no fuera así C_i sería una tautología. De esta forma el resolvente obtenido C_{i+1} subsume a su padre cercano. Este hecho corresponde a la restricción s-lineal de s-resolución lineal de Loveland. La condición 2), que establece la obligatoriedad de efectuar resoluciones de ancestros, evita la obtención de deducciones inútiles.

- Claramente, para una eficiente implementación de t-resolución lineal es muy aconsejable, por no decir imprescindible, que se lleve para cada cláusula de la deducción la lista de sus A-ancestros y sus A-literales. De hecho, únicamente es necesario llevar una lista de A-literales ya que los otros literales de los A-ancestros están contenidos en la cláusula en curso. De esta forma, las restricciones 1) y 2) pueden ser realizadas borrando simplemente cualquier literal de C_i que es complementario de un A-literal de la lista almacenada. Esta forma de recordar los A-ancestros y los A-literales de la cláusula en curso claramente recuerda la forma en la cual se define Eliminación de Modelos.

De lo mencionado anteriormente se puede deducir que esencialmente s-resolución lineal y t-resolución lineal son dos maneras de formalizar la misma restricción que evita la utilización de todos los ancestros reduciendo de esta forma la ramificación del árbol de estados y posiblemente el tamaño del espacio de búsqueda. Al igual que le ocurría a s-resolución lineal, t-resolución lineal es demasiado flexible en la aplicación de resolución de entrada, permitiendo resolver sobre cualquier literal de la cláusula en curso. El siguiente ejemplo muestra este hecho y permite ver la forma de las t-deducciones.

Ejemplo 2.15 (t-deducción y tamaño del espacio de búsqueda de t-resolución lineal)

Sea S el siguiente conjunto de cláusulas:

$C_1 = p \vee q$ $\qquad$ $C_4 = \neg r \vee s$

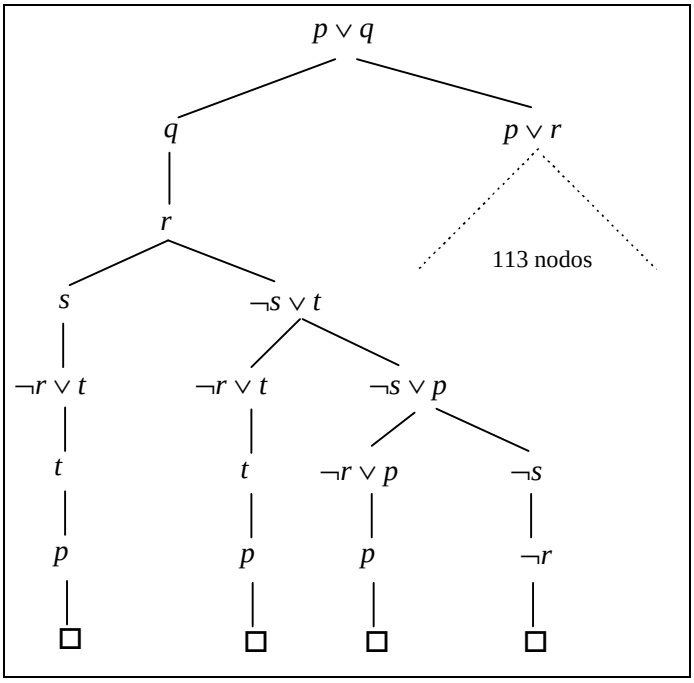
$C_2 = \neg p$ $\qquad$ $C_5 = \neg r \vee \neg s \vee t$

$C_3 = \neg q \vee r$ $\qquad$ $C_6 = p \vee \neg t$

La siguiente figura muestra el árbol de estados de las t-deducciones de C_1 a partir de S.

Figura 2.9: Árbol de estados de las t-deducciones para el Ejemplo 2.15

La t-deducción más a la izquierda en el árbol es la secuencia $p \vee q$, q, r, s, $\neg r \vee t$, t, p, $\square$. En la cláusula t tiene como A-ancestros a las cláusulas q, r, y s siendo los únicos literales de estas cláusulas los A-literales correspondientes. Sin embargo $\neg r \vee t$ no es A-ancestro de t, porque esta última no se obtiene por resolución de entrada. Obsérvese que la cláusula r es un A-ancestro de la cláusula $\neg r \vee t$, siendo r el A-literal, de forma que la siguiente cláusula en la deducción, t, se obtiene obligatoriamente por resolución de ancestro usando el A-ancestro. Fácilmente se puede ver la amplitud del espacio de búsqueda que se obtiene con t-resolución lineal debido a la posibilidad de aplicar resolución de entrada sobre cualquier literal de la



cláusula en curso en la deducción. Es interesante destacar que s-resolución lineal para el caso libre de variables coincide con t-resolución lineal (en el árbol de la Figura 2.9 se puede comprobar que todas las t-deducciones lineal son s-deducciones lineales) con la ventaja añadida en la t-resolución lineal de la económica implementación de las operaciones de resolución de ancestro.

Sin embargo t-resolución lineal presenta el mismo problema que s-resolución lineal: el elevado número de resolventes que se pueden obtener por resolución de entrada de la cláusula en curso en la deducción, al ser posible resolver sobre cualquier literal de ésta. La solución que se aplicaba a este problema en la s-resolución lineal era restringir este amplio número de aplicaciones de la resolución de entrada utilizando la misma técnica que resolución ordenada, de forma que sólo era posible aplicar resolución de entrado sobre un único literal (dado por el orden) de la cláusula en curso. A t-resolución lineal se le impone la misma restricción pero de una forma más flexible, ya que este único literal se selecciona dinámicamente durante la deducción por medio de una función de selección. A este nuevo refinamiento de resolución

lineal se le denomina *resolución SL*. La elección de este literal se restringe a aquellos literales de la cláusula en curso que han aparecido más recientemente. Esta forma de selección asegura que la eliminación del literal seleccionado por ser complementario de un A-literal de la cláusula corresponde a una resolución de ancestro.

Estudiando con más detenimiento este hecho se puede observar que para cada derivación *d* en el árbol de estados obtenido mediante la resolución SL, existe un único literal en la cláusula derivada *C* que es sobre el cual se resuelve para obtener todos los descendientes inmediatos. Si la misma derivación se obtiene mediante t-resolución lineal existen muchos más descendientes obtenidos al resolver sobre los otros literales de *C*. Así, si *C* contiene *m* literales existen, por promedio, *m* veces más descendiente inmediatos de *d* en el árbol de estados de t-resolución lineal que los que hay en el árbol de estados de resolución SL. Si *m* es el número medio de literales en las cláusulas derivadas en la t-deducciones lineales de longitud menor o igual a *n* entonces existen por promedio, m^n más t-deducciones lineales de longitud *n* que deducciones SL de la misma longitud. Esto puede dar una idea de la reducción del espacio de búsqueda que provoca la restricción de resolver sobre un único literal de cada cláusula de la deducción. El siguiente ejemplo muestra el árbol de estados de resolución SL para el ejemplo anterior.

Ejemplo 2.16 (reducción del tamaño del espacio de búsqueda usando resolución SL)

Sea *S* el mismo conjunto de cláusulas que el del Ejemplo 2.15. La siguiente figura muestra el árbol de estados de resolución lineal con función de selección que selecciona el literal cuyo átomo es el mayor alfabéticamente.

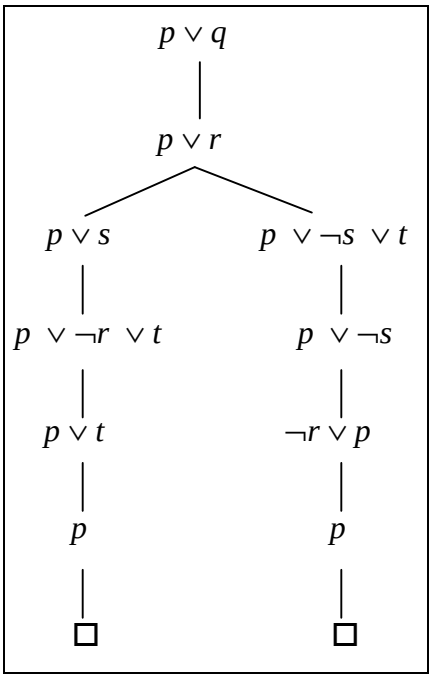


Figura 2.10: Árbol de estados de resolución SL para el Ejemplo 2.16

Como se puede observar la reducción del espacio de búsqueda es espectacular pasándose de un árbol de estados de t-resolución lineal con 134 nodos al de resolución SL con sólo 12 nodos.

Antes de presentar la definición de resolución SL dada en [Kowalski y Kuehner, 1971] es necesario remarcar algunos aspectos interesantes. Primero destacar que la formulación utilizada por estos autores sigue el mismo formato que la de Eliminación de Modelos. Por un lado las cláusulas son transformadas a cadenas en las cuales aparecen dos tipos de literales, los A-literales (con un significado y uso similar al visto en t-resolución lineal) y los B-literales. Por otro lado como cadenas de entrada para resolución de entrada se utilizan todas las contrapositivas de los factores de las cláusulas en el conjunto inicial. Además si C es una cláusula, una cadena obtenida a partir de ella se la denotará por C^*. En las cadenas de entrada que provienen de las cláusulas del conjunto inicial, como ocurría en Eliminación de Modelos, todos los literales son B-literales y además cuando se usan para realizar aplicaciones de resolución de entrada sólo se puede resolver sobre su literal más a la izquierda. Para formalizar correctamente la función de selección estos autores introducen un concepto nuevo denominado *celda*. Así se dice que dos B-literales están en la misma celda si no están separados por un A-literal. De esta forma la cadena $p\ \boxed{q}\boxed{r}\ \neg s\ t$ tiene dos celdas, una contiene el B-literal p y la otra a los literales $\neg s$ y t. Segundo, el literal de la cadena en curso sobre el que se resuelve en la deducción es seleccionado por una función, denominada *función de selección*. A continuación se presenta su definición.

Definición 2.36 (Función de selección) [Kowalski y Kuehner, 1971]

Sea f una función definida sobre cadenas que no son vacías que devuelve cadenas. f es una *función de selección* si y sólo si $f(C^*)$, es C^* o se obtiene a partir de C^* intercambiando el B-literal más a la derecha por otro B-literal que se encuentre en la celda más a la derecha. El literal más a la derecha en $f(C^*)$ se denomina el *literal seleccionado*.

Así, si C^* es la cadena $p\ \boxed{q}\boxed{r}\ \neg s\ t$ entonces $f(C^*)$ puede ser $p\ \boxed{q}\boxed{r}\ \neg s$ o la misma C^*. Esta forma de definir la función de selección aunque puede parecer ligeramente complicada permite dar una definición muy sencilla de la resolución SL. A continuación se presenta esta definición.

Definición 2.37 (Derivación SL y refutación SL) [Kowalski y Kuehner, 1971]

Sea S un conjunto de cláusulas, C una cláusula y ϕ una función de selección. Una *derivación SL a partir C en S* es una secuencia de cadenas $C_1^*, \ldots, C_n^*$ que cumple los puntos 1) a 3) siguientes:

1) $C_1^* = C^*$

2) cada C^*_{i+1} se obtiene a partir de C^*_i aplicando una extensión, reducción o truncamiento.
3) A menos que C^*_{i+1} se obtenga a partir de C^*_i por reducción, entonces no hay dos literales en C^*_i cuyos átomos sean los mismos (*restricción de admisibilidad*).

C^*_{i+1} se obtiene a partir de C^*_i por truncamiento si y sólo si se cumple a) y b):

a) El literal más a la derecha de C^*_i es un A-literal.
b) C^*_{i+1} es la subsecuencia más larga de C^*_i cuyo literal más a la derecha es un B-literal. El estado de cualquier literal de C^*_{i+1} es el mismo tiene en C^*_i.

C^*_{i+1} se obtiene a partir de C^*_i por reducción si y sólo si se cumple de a) a e):

a) El literal más a la derecha de C^*_i es un B-literal.
b) C^*_i no se obtiene de C^*_{i-1} por truncamiento.
c) La celda más a la derecha de C^*_i contiene un B-literal *l* y, o bien
   i) C^*_i contiene un B-literal *K*, que no es el de más a la derecha de C^*_i (factorización básica), o bien
   ii) C^*_i contiene un A-literal *K* complementario a *L*, que no es el A-literal más a la derecha de C^*_i (resolución de ancestro).
d) Los átomos de los literales *L* y *K* son unificables por el umg θ.
e) Sea C^{**}_i la cadena obtenida a base de eliminar la ocurrencia dada de *L* en C^*_i. Entonces C^*_{i+1} es igual a $C^{**}_i\theta$. El estado de cualquier literal en C^*_{i+1} es el mismo que el del literal del cual desciende en C^*_i.

C^*_{i+1} se obtiene a partir de C^*_i por extensión usando una cadena de entrada B^* si y sólo si a)-d):

a) El literal más a la derecha de C^*_i es un B-literal.
b) C^*_i y B^* no comparten variables.
c) El literal seleccionado *L* en C^*_i y el complementario del literal más a la izquierda *K* de B^* son unificables por un umg θ.
d) Sea B^{**} la cadena obtenida eliminando el literal *K* de B^*. Entonces C^*_{i+1} es la cadena $(\phi(C^*_i)\ B^{**})\theta$ obtenida aplicando θ al resultado de concatenar $\phi(C^*_i)$ y B^{**} en ese orden. El literal $L\theta$ en C^*_{i+1} descendiente del literal más a la derecha en $\phi(C^*_i)$ es un A-literal en C^*_{i+1}. Cada uno de los restantes literales en C^*_{i+1} tiene el mismo estado que el del literal en C^*_i o B^{**} del cual desciende.

Es interesante destacar algunos de los puntos de la extensa definición de resolución SL que permiten encontrar algunos nexos con otros procedimientos ya vistos:

- Claramente resolución SL es muy semejante a Eliminación de Modelos, existiendo algunas interesantes diferencias que posteriormente se comentarán.

- Fácilmente se puede comprobar que la restricción de admisibilidad junto con el punto b) de la definición de reducción incorporan las tres restricciones de t-resolución lineal así como la fusión obligatoria y la restricción de inexistencia de tautologías.

- Los autores incorporan a la operación de reducción de manera implícita la factorización, destacando así la importancia que le dan a esta operación. También comentan que la forma de factorización utilizada en resolución SL es el método de factorización menos redundante que genera las cláusulas más cortas de la forma más rápida posible y que no implica un incremento de la complejidad de las deducciones. Por otro parte, las restricciones que le imponen a la operación de reducción es para evitar la aplicación de esta operación superflua con el uso de factores de las cláusulas iniciales.

- Es interesante destacar que la forma de resolución de ancestro definida en resolución SL no corresponde exactamente a la de resolución lineal, ya que cuando se aplica esta operación no se usa una variante de la cláusula ancestro para impedir que comparta alguna variable con la cláusula padre cercano. Esta forma de aplicar resolución de ancestro se puede usar en cualquier sistema basado en resolución lineal en general. Por otra parte esta forma de resolución de ancestro presenta las valiosas ventajas de ser una forma eficiente y restrictiva de resolución de ancestro así como de reflejar de forma más directa la relación existente entre resolución de ancestro y factorización.

Como se puede ver resolución SL está muy cercana, tanto en su definición como en el tipo de representación de las cláusulas, a Eliminación de Modelos. A continuación se presentan las diferencias más importantes entre ambos procedimientos:

a) Para deducciones totalmente instanciadas Eliminación de Modelos no aplica la fusión de literales.

b) Eliminación de Modelos permite la aplicación de resolución de ancestro sobre el A-literal de más a la derecha lo que supone la generación de deducciones redundantes con la existencia de factores de las cláusulas iniciales.

c) La selección de literales mediante una función no aparece en Eliminación de Modelos. Este procedimiento usa la técnica de resolución ordenada para impedir la excesiva ramificación del árbol de estados. Evidentemente, la función de selección se puede incorporar a Eliminación de Modelos fácilmente.

Quizás los dos aspectos más destacables de la presentación de resolución SL en [Kowalski y Kuehner, 1971] sean el estudio de la complejidad de resolución SL y la justificación de la utilidad de la función de selección para obtener reducciones importantes del espacio de búsqueda. Sobre el primero los autores estudian la complejidad de resolución y sus

refinamientos y demuestran que para un conjunto inconsistente de cláusulas existe una refutación SL tan simple como alguna refutación mimimal de dicho conjunto. Sobre el segundo, comentan que el uso de una función de selección que se base en la utilización de árboles de clasificación permite obtener importantes reducciones de la ramificación de los árboles del estados y por tanto un decremento muy importante de los espacios de búsqueda.

Ejemplo 2.17 (Árbol de estados de resolución SL)

Sea S el siguiente conjunto de cláusulas (el mismo que el del Ejemplo 2.15):

$C_1 = p \vee q$ $\quad$ $C_4 = \neg r \vee s$

$C_2 = \neg p$ $\quad$ $C_5 = \neg r \vee \neg s \vee t$

$C_3 = \neg q \vee r$ $\quad$ $C_6 = p \vee \neg t$

La siguiente figura muestra el árbol de estados de las derivaciones SL de C_1 a partir de S. La función de selección elige el literal con el símbolo de predicado mayor alfabéticamente. En la figura los arcos están etiquetados con la operación aplicada para obtener la siguiente cláusula de la derivación (E(n) por extensión donde n es el número de la cláusula de la cual proviene la cadena de entrada usada, R(ra) por Reducción correspondiente a una resolución de ancestro, R(f) por Reducción correspondiente a una factorización y T por Truncamiento).

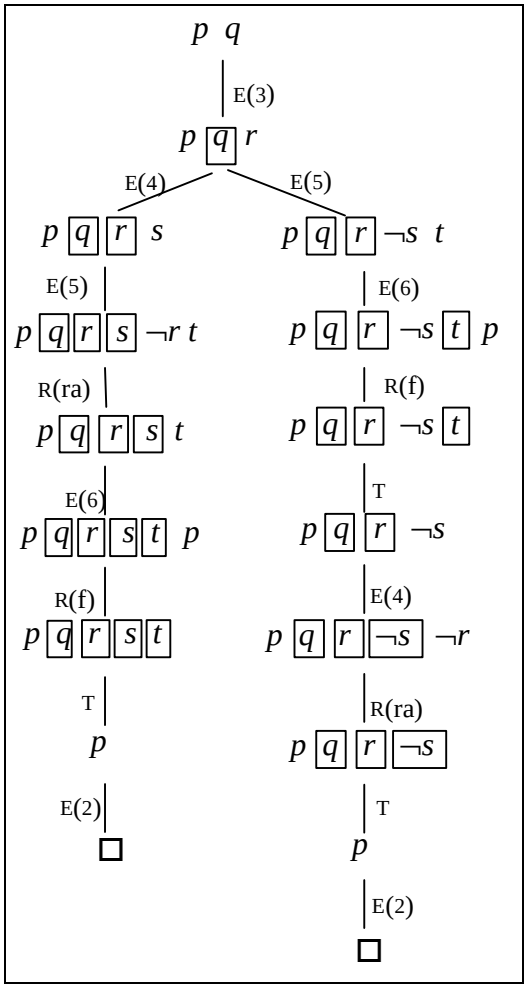


Figura 2.11: Árbol de estados de resolución SL para el Ejemplo 2.17

Como se puede observar en la figura cada cláusula contiene la información suficiente para realizar correctamente la aplicación de resolución de ancestro sin tener que revisar las cláusulas anteriores en la derivación. Así para cada B-literal que contiene los A-ancestros son los A-literales encuentran a su izquierda que son los únicos que se pueden utilizar para aplicar una reducción, realizando así correctamente la resolución de ancestro.

2.5.8. EL FORMATO DE REDUCCIÓN DE PROBLEMAS. EL SISTEMA MESON

La presentación del formato de reducción de problemas y del sistema MESON en mayor grado se considera importante por su proximidad al paradigma de resolución SL* y especialmente a la implementación efectuada de dicho paradigma. Esta presentación aclarará mucho conceptos y mecanismos utilizados en resolución SL*. El formato de reducción de problemas es una forma de organización de los distintos mecanismos de inferencia y búsqueda utilizados en el campo del razonamiento automático e inteligencia artificial clásicos. Este formato tuvo su aparición con anterioridad al principio de resolución de Robinson. Fue utilizado por Newell y Simon en la *Logic Theory Machine* [Simon, 1973], en el procedimiento para la

demostración de teoremas de la geometría elemental de Gelernter y posteriormente en el inacabado sistema Planner desarrollado en la década de los 60 por el MIT.

Antes de describir la relación del formato de reducción de problemas con el procedimiento de Eliminación de Modelos, del cual surge el sistema MESON, se discutirán brevemente las características fundamentales del formato de reducción de problemas clásico. La idea básica de este formato es la de reducir una meta, objetivo o problema a resolver en un conjunto de subproblemas más simples, siguiendo una estrategia descendente, de forma que la resolución de los subproblemas implique la demostración del objetivo inicial. Esta descomposición se realizaba aplicando una serie de reglas que garantizan la corrección, y en algunos casos la completitud del procedimiento, además de considerar las posibles formas de mejorar su eficiencia. Inicialmente el formato de reducción de problemas fue diseñado para aplicarse a problemas formalizados usando la lógica proposicional.

Esencialmente el formato de reducción de problemas hace uso de la notación clausal implicativa para la formalización de los problemas a tratar y en la cual el problema a demostrar o *conclusión* se representa como una proposición G y el conjunto de axiomas o aserciones que representan el problema como *implicaciones* de la forma $A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow C$ (cláusulas en forma implicativa) o *premisas* de la forma P, donde G, P y A_i ($1 \leq i \leq n$) son átomos. En las implicaciones se pueden distinguir los *antecedentes* $A_1, A_2, \ldots, A_n$ y el consecuente C.

El proceso de demostración de la conclusión G, también denominada *meta inicial* o *raíz*, comienza desde la misma intentado comprobar si:

- o bien es ya una de las premisas,
- o bien, es posible demostrar todo antecedente de alguna de las cláusulas que tienen a G como consecuente.

En la práctica la aplicación de esta forma de reducción se realiza construyendo un árbol AND-OR [Nilson, 1987] (o árbol de metas) que permite representar la estructura conjuntiva de los antecedentes de las cláusulas en forma implicativa y la estructura disyuntiva de la utilización de las distintas cláusulas con el mismo consecuente para la demostración de una cierta meta. En este árbol los nodos corresponden a metas (esencialmente, antecedentes de implicaciones) y los arcos entre nodos representan las relaciones implicativas de las premisas. Evidentemente, la aparición de un árbol AND-OR correspondería a una estrategia descendente de demostración a partir de la meta G. Existen otras posibilidades, como son la estrategia ascendente o una estrategia mixta descendente-ascendente, que darían lugar a un grafo AND-OR en vez de un árbol pero que son menos utilizadas. Por lo general los procedimientos basados en este formato construyen un árbol AND-OR cuya raíz es la meta inicial en el cual van etiquetando los nodos según cómo se encuentre su demostración y eliminando aquellos nodos irrelevantes según

ciertos criterios de poda. La demostración termina en el momento en el cual se logra etiquetar la raíz como “probada”.

A continuación se presentan algunos términos utilizados en el formato de reducción de problemas habituales, por otra parte, en los procedimientos de prueba.

Definición 2.38 (Metas padres, sucesoras, ancestros y descendientes) [Loveland, 1978]

Una meta *G* es *meta padre* de los antecedentes de las implicaciones cuyo consecuente es *G*. Estos antecedentes son *metas sucesoras* de *G*. Dadas dos metas *G* y *G’* se dice que *G* es *ancestro* de *G’* si y sólo si o bien *G* es una meta padre de *G’* o bien *G* es una meta padre de un ancestro de *G*, siendo entonces *G’* un *descendiente* de *G*.

Definición 2.39 (Conjunto conjuntivo de submetas y conjunto disyuntivo de una meta) [Loveland, 1978]

Dada una meta G y una implicación de la forma $A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow G$ se denomina *conjunto conjuntivo* de *G* al conjunto $\{A_1, A_2, \ldots, A_n\}$. Se denomina *conjunto disyuntivo* al conjunto de los distintos conjuntos conjuntivos de *G*.

El siguientes ejemplo muestra la manera de construir el árbol AND-OR a partir de las implicaciones.

Ejemplo 2.18 (Construcción del árbol AND-OR en el formato de reducción de problemas)

Dada la meta inicial *g* y el siguiente conjunto de implicaciones:

$g \leftarrow p \wedge q \wedge r$ $\qquad$ $r \leftarrow v \wedge w$

$g \leftarrow s$ $\qquad$ $r \leftarrow x$

$g \leftarrow t \wedge u$ $\qquad$ $r \leftarrow y$

donde *g, p, q, r, s, t, u, v, w, x* e *y* son átomos. La figura siguiente muestra el árbol AND-OR correspondiente a este problema.

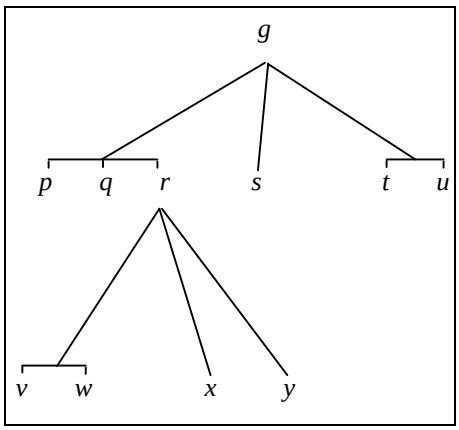


Figura 2.12: Árbol AND-OR para el Ejemplo 2.18

Como se puede ver en la figura anterior, en el contexto del formato de reducción de problemas se construye el árbol AND-OR a partir de la meta inicial, mediante la expansión adecuada de metas en metas sucesoras utilizando las implicaciones del problema a tratar. Además se consideran las relaciones o *interacciones* entre las distintas metas para etiquetar y eliminar adecuadamente los nodos y arcos del árbol AND-OR como mecanismo esencial para la demostración del problema inicial. A continuación se describe la mecánica de la construcción del árbol y las interacciones entre las diferentes metas:

1) Si una meta *G* es idéntica a alguna premisa, entonces *G* se considera la meta probada, constituye una meta terminal, es etiquetada como *premisa* y el arco que lo une con su meta padre es etiquetado como *establecida*.

2) Si todas las submetas de una meta *G* provenientes de una implicación, esto es, un conjunto conjuntivo de *G*, son etiquetadas probadas, entonces el arco que une *G* con el conjunto se etiqueta como establecido y la propia meta *G* se considera probada.

3) Si una meta *G* es idéntica a uno de los ancestros se etiqueta como *meta superior*, se considera meta terminal y no probada, se detiene el proceso de demostración en el conjunto conjuntivo al que pertenece *G* y el arco que une el conjunto conjuntivo con su meta padre se etiqueta como *rechazado*.

4) Si todos los conjuntos conjuntivos de una meta *G* están etiquetados como rechazados, entonces la propia meta *G* se considera rechazada y el arco que la une con su meta padre es a su vez etiquetado como rechazado.

Las anteriores interacciones son las básicas y con ellas se puede proceder a la demostración de problemas. Es interesante destacar los siguientes puntos: para que una meta se considere rechazada todos los conjuntos conjuntivos sucesores de ella deben ser rechazados; en un conjunto conjuntivo si una meta es rechazada el conjunto completo se rechaza; y, para confirmar o establecer una meta únicamente un conjunto conjuntivo sucesor ha de ser probado. Existen otras interacciones que pueden ser utilizadas de forma adicional para aumentar la eficiencia del proceso:

5) Si una meta *G* está establecida y una meta idéntica aparece en el árbol, entonces etiquetarla también como establecida ya que el mismo subárbol que permitió etiquetar como establecida a *G* se puede volver a construir.

6) Considerar como rechazada aquellas metas *G* que no tienen como mínimo una submeta cuyo arco esté etiquetado como establecido.

Este conjunto de reglas suponen, en cierto sentido, la corrección del procedimiento asociado. Desafortunadamente, el formato de reducción de problemas presenta una carencia

expresiva al poder representar únicamente teorías Horn y, por tanto, ser incapaz de tratar problemas más generales. Por otra parte si se extiende la forma de las implicaciones para poder representar cláusulas generales la completitud no está garantizada. Es Loveland [Loveland, 1978] el que propone una extensión del formato para la representación de problemas así como una nueva regla que asegure la completitud del procedimiento en el caso de teorías clausales generales. Las dos definiciones siguientes presentan esta extensión del formato de representación y la nueva regla.

Definición 2.40 (Contrapositivas generales) [Loveland, 1978]

Sean L_i, $1 \leq i \leq n$, y L literales y $L_1 \wedge L_2 \wedge \ldots \wedge L_n \rightarrow L$ una implicación. Una *contrapositiva general* de la dicha implicación es una implicación de la forma $J_1 \wedge J_2 \wedge \ldots \wedge J_n \rightarrow J$, donde J es el literal complementario de L_i y cada J_i es o un L_j o el complementario de L de forma que el complementario de L y cada L_i o su complementario aparecen exactamente una vez en la implicación. La contrapositiva general de la implicación L es $\neg L \rightarrow$.

Ejemplo 2.19 (Contrapositivas generales)

Sea C la siguiente implicación:

$$C = p \wedge q \rightarrow r$$

las siguientes cuatro implicaciones son las contrapositivas generales de C.

$$\neg r \wedge q \rightarrow \neg p$$
$$p \wedge \neg r \rightarrow \neg q$$
$$q \wedge \neg r \rightarrow \neg p$$
$$\neg r \wedge p \rightarrow \neg q$$

La necesidad de las contrapositivas ya se ha comentado ampliamente con anterioridad. Únicamente cabría destacar que en este contexto se hacen necesarias al extender el formato, permitiendo la presencia de literales en las implicaciones. Por este mismo motivo también se hace necesario introducir una nueva regla de interacción que se puede enunciar como sigue: se considera una meta terminal aquella que es complementaria de una de sus metas ancestros. La justificación de esta nueva regla se puede encontrar en el principio de reducción al absurdo: si una meta $\neg A$ tiene un ancestro de la forma A, entonces o bien A o $\neg A$ tienen que ser ciertas, pero no ambas; la asunción de la certeza de $\neg A$ permitiría establecer la certeza de su ancestro A lo cual sería una contradicción, por tanto $\neg A$ deber ser falsa. De esta forma, Loveland que fue el introductor de esta nueva regla, propone considerar como *contradichas* este tipo de metas y considerarlas metas terminales. Explícitamente se evita el considerar la meta como probada porque realmente no lo está y para que así su meta padre (y todas las metas intermedias entre su ancestro complementario y ella misma) no se considere probada ya que ello no sería cierto y

podría llevar a incorrecciones en usos especiales de dichas metas. A las metas intermedias se las considerar *aceptadas* según la siguiente regla: una meta padre de cualquier conjunto conjuntivos de metas se etiqueta como aceptada si cada submeta es o a) una premisa, o b) una submeta etiqueta como contradicha, o c) el padre de un conjunto conjuntivo aceptado.

Por otro lado, y a pesar de lo dicho anteriormente, es posible considerar una meta probada aunque alguno de sus submetas sólo esté aceptada. Una meta *G* etiquetada como aceptada puede ser reetiquetada como probada cuando se cumple que ningún descendiente contradicho tiene como su ancestro complementario a un ancestro de *G*.

Se ha de destacar que el mecanismo para comprobar la posibilidad de contradecir una submeta, que junto con la extensión general del formato de representación consistente en usar contrapositivas se denominará *mecanismo de contradicción*, es un incremento trivial al necesitado para comprobar que la submeta es idéntica a alguno de sus ancestros. Así, dado una nueva submeta se comprueba si es idéntica o complementaria con cada uno de sus ancestro antes de aplicar cualquier otro operación. El siguiente teorema enuncia la completitud del formato de reducción de problemas ampliado con el mencionado mecanismo.

Teorema 2.12 (Suficiencia de formato de reducción de problemas con el mecanismo de contradicción) [Loveland, 1978]

Sea *S* un conjunto de expresiones proposicionales con el formato de reducción de problemas clásico extendido con el del mecanismo de contradicción. Si una meta es consecuencia lógica de *S* y *S* no es contradictorio, entonces el sistema de reducción de problemas ampliado con el mecanismo de contradicción es suficiente para permitir comprobar la meta inicial.

Es interesante destacar que el anterior teorema nos proporciona la seguridad de haber encontrado un mecanismo sencillo y económico cuya incorporación al formato de reducción de problemas permite alcanzar la completitud. Este mecanismo propuesto por el sistema MESON consta de una nueva operación, prueba por contradicción o reducción al absurdo, que permite la demostración de una meta basándose en la información de la derivación en curso. Por otro lado, el formato utilizado por el sistema MESON considera las contrapositivas de las implicaciones de la teoría inicial, ampliando así el formato de reducción de problemas clásico.

Antes de dar la definición del sistema MESON, concretamente una versión restringida ya que la que D. Loveland propone incluye el tratamiento de la igualdad, se va a presentar el formato que usa MESON para la descripción de los problemas, habitualmente denominado *formato meta-aserción*.

Definición 2.41 (Formato meta-aserción) [Loveland, 1978]

Un conjunto de fórmulas está en formato meta-aserción si y sólo si:

1) existe una única fórmula, nominada *meta*, que es una conjunción de literales, y
2) cada fórmula restante, denominada *aserción*, es o bien
   a) un literal (*premisa*), o bien
   b) una *implicación*, de la forma $L_1 \wedge L_2 \wedge \ldots \wedge L_n \rightarrow L_{n+1}$, donde L_i es un literal para $1 \leq i \leq n+1$.

Ejemplo 2.20 (Formato meta-aserción)

Sea S el siguiente conjunto de cláusulas (se podría partir de cualquier conjunto de fórmulas, en general):

$$p(x) \vee q(x)$$
$$\neg p(x) \vee q(x)$$
$$p(x) \vee \neg q(x)$$
$$\neg p(x) \vee \neg q(x)$$

Una expresión del mismo en formato meta-aserción es la siguiente

Meta:	$p(x) \wedge q(x)$	
Aserciones:	$p(x) \rightarrow q(x)$	
	$q(x) \rightarrow p(x)$	
	$\neg q(x) \rightarrow p(x)$	
	$q(x) \rightarrow \neg p(x)$	(negación de la meta)

Obviamente cualquier expresión en la cual se incluyeran contrapositivas de las implicaciones del formato anterior también estaría en formato meta-aserción. La meta corresponde a la negación de la cláusula $\neg p(x) \vee \neg q(x)$, ya que en esencia MESON y el formato de reducción de problemas son sistemas que, en vez de intentar llegar a la contradicción alcanzando la cláusula vacía, demuestran que una instancia de la meta es un teorema del problema original. Evidentemente, las dos filosofías de funcionamiento están interconectadas aunque llevan a distintas consideraciones de formato. Por este motivo la cuantificación implícita de la meta y de las aserciones es distinta. En la primera la cuantificación es existencial y en la segunda universal. Además también es interesante destacar la presencia de la última implicación que corresponde a la negación de la meta, hecho imprescindible para garantizar la completitud del sistema. Este requisito es equivalente a la imposición de la presencia de la cláusula inicial de los sistemas basados en resolución.

La presentación formal del sistema MESON para problemas de primer orden requiere un formato equivalente al visto para el formato de reducción de problemas clásico. Esto es, se ha

de representar la evolución de la deducción a través de las aserciones, la forma en la que una meta se descompone en submetas y las diferentes alternativas que van apareciendo. En el formato de reducción de problemas se utilizaban los árboles AND-OR para realizar esta representación. Con el fin de simplificar ésta para el MESON, Loveland, elige la forma más sencilla representando la porción del árbol que corresponde a las acciones tomadas hasta el momento en curso. Esto se consigue manteniendo separadas las diferentes alternativas de búsqueda para la confirmación de una meta. Técnicamente esto significa tener muchos árboles, cada uno corresponde a una alternativa y uno de ellos es el árbol de la derivación en curso, ninguno de los cuales tiene ramas OR. Estos árboles se denominan árboles AND.

Definición 2.42 (Árbol AND) [Loveland, 1978]

Un *árbol AND* es un árbol de AND-OR en el que cada meta tiene como mucho un conjunto conjuntivo de submetas. Dado un árbol de AND-OR un *subárbol maximal AND* es un árbol AND obtenido a partir del árbol dado eliminando todos los conjuntos de submetas excepto uno de cada una de las metas.

Los árboles AND son suficientes para mostrar cualquier demostración de la meta inicial. Las ramas OR de los árboles AND-OR sirven para mostrar las diferentes alternativas para la confirmación de la meta investigada.

La definición que se va a dar a continuación del sistema MESON corresponde a la versión que no incluye el tratamiento de la igualdad, intentando de esta forma que la esencia del sistema quede más clara. Esencialmente está definición incluye un conjunto de opciones, acciones y reglas de borrado que extienden las vistas en el formato de reducción de problemas. Este conjunto de opciones, acciones y reglas definen el árbol AND que se va a ir generando para la demostración de la meta inicial. De forma más estricta, cada vez que una substitución diferente de la vacía es aplicada o una submeta es añadida, un nuevo árbol AND es generado a partir del anterior aplicando la correspondiente operación. Para investigar caminos alternativos de una posible confirmación para una cierta meta se vuelve al árbol AND donde dicha meta no tiene descendientes y se extiende el árbol; o sea, se genera un nuevo árbol con metas instanciadas, si es necesario, y el nuevo conjunto de submetas es añadido.

Definición 2.43 (El sistema MESON sin igualdad) [Loveland, 1978]

Dado un cierto problema expresado en el formato meta-aserción, las siguientes operaciones, denominadas *opciones*, *acciones* y *reglas de borrado*, permiten construir un árbol AND que demuestra la meta inicial elegida.

Antes de dar la definición del sistema MESON es necesario incluir las siguientes aclaraciones: primero, sea L un literal cualesquiera, por $\widetilde{L}$ se entenderá el literal complementario al mismo; segundo, las sustituciones incluidas en la definiciones son siempre

sustituciones de unificación; y tercero, las operaciones que están entre corchetes no son necesarias para garantizar la completitud aunque se consideran interesantes ya que mejoran la eficiencia del mismo.

a) Opciones:

O1) Sea G una meta que no está marcada y sea B una premisa tal que existe una sustitución s de forma que $Gs=Bs$, entonces Gs es marcada como *premisa* y se considera probada.

O2) Sea G una meta que no está marcada y sea $L \leftarrow L_1 \wedge ¼ \wedge L_n$ una aserción tal que existe una sustitución s de forma que $Gs=Ls$, entonces un conjunto conjuntivo de submetas de Gs es $\{ L_1 s, ¼, L_n s\}$.

O3) Sea G una meta que no está marcada y sea D una meta descendiente de G, si existe una sustitución s tal que $Gs= \widetilde{D}\, s$, entonces la meta Ds es marcada como *contradicha*.

[O4) Sea G una meta que no está marcada y sea D una meta hermana o descendiente de G, si existe una sustitución s tal que $Gs= \widetilde{D}\, s$, entonces la meta Ds es marcada como *desplazada*.]

b) Acciones:

A1) Sea G una meta y S un conjunto conjuntivo de G de la forma $\{L_1, ¼, L_n\}$. Si cada L_i, $1 \leq i \leq n$, está marcada como *premisa* o *probada*, entonces G se marca como *probada*.

A2) Sea G una meta y S un conjunto conjuntivo de G de la forma $\{L_1, ¼, L_n\}$ de forma que cada L_i, $1 \leq i \leq n$, está marcada como *premisa*, *probada*, aceptada, *desplazada* o *contradicha*. Si al menos una submeta L_i no está marcada como *probada* o *premisa* entonces G se marca como *aceptada*.

A3) Si G es una meta marcada como *aceptada* y cada meta descendiente de G que está marcada como *desplazada* o *contradicha* tiene como meta superior asociada a G o a una descendiente de G, entonces G se reetiqueta como *probada*.

c) Reglas de borrado:

R1) Si G es una meta que no está marcada y D es una meta descendiente de G de forma que D es idéntica a G, entonces el conjunto conjuntivo de metas que contiene a D es eliminado.

[R2) Si *G* es una meta que no está marcada y que aún no ha sido expandida, y *D* es una meta descendiente de un hermano de *G* tal que $\tilde{D}$ es idéntica a *G*, entonces el conjunto conjuntivo de metas que contiene a *D* es eliminado.]

R3) Si *G* es una meta que no está marcada y *D* es una meta descendiente de *G* tal que $\tilde{D}$ es idéntica a *G*, entonces el conjunto conjuntivo de metas que contiene a *D* es eliminado.

[R4) Si *G* es una meta que no está marcada y *D* es una meta descendiente de un hermano de *G* tal que *D* es idéntica a *G*, entonces el conjunto conjuntivo de metas que contiene a *D* es eliminado.]

Como afirma Loveland [Loveland, 1978], las opciones de la O1 a la O3 junto con las acciones de la A1 a la A3 y las reglas de borrado R1 y R3 sirven, por sí solas, para definir un sistema completo para el tratamiento de problemas sin igualdad. En la presentación que se ha hecho del sistema MESON se ha decidido no incluir dicho tratamiento para hacer ésta más simple y poder comparar de forma más sencilla este sistema con otros ya vistos.

Si se compara el sistema MESON con el procedimiento ME es posible encontrar diversas analogías en las operaciones que se definen en los mismos. Así, la operación de extensión de ME se encuentra recogida en las opciones O1 y O2 según la cadena de entrada (terminología de ME) corresponda a una premisa (cláusula unitaria) o a una aserción (cláusula que no es unitaria), respectivamente. La operación de reducción de ME se puede asimilar a la opción O3, ya que ambas están dando cuenta de la regla de inferencia *reducción al absurdo*. Claramente, esta correspondencia entre las operaciones de ME y las opciones de MESON es sólo parcial ya que a estas últimas las tienen que acompañar las acciones de A1 a A3 que reetiquetan adecuadamente el árbol AND permitiendo saber qué metas se van probando según las opciones que han tenido lugar sobre ellas o sobre sus descendientes.

La opción O4 junto con la regla de borrado R4, que la complementa, permite el desplazamiento de la demostración de una meta, posponiendo su demostración a la prueba de un ancestro relacionado con ella. Claramente el resultado de aplicar esta opción, que está ligada con factorización, puede ser también conseguido aplicando el uso de lemas en la demostración. Aún así hay que destacar que por un lado su coste se puede considerar bastante elevado cuando haya presencia de variables y, por otra parte, sus beneficios no son demasiados importantes ya que para garantizar la completitud del sistema hay que continuar aplicando otras opciones sobre la meta desplazada. El mismo Loveland señala los problemas de esta opción y aconseja no aplicarla de forma generalizada.

La correspondencia formal entre el sistema MESON y el procedimiento ME queda establecida por Loveland en [Loveland, 1978]. Para ello propone una aplicación que a una

cadena ME le hace corresponder un árbol AND. Por cada A-literal de la cadena aparece una meta en el árbol AND que es el literal complementario del A-literal. Esta meta tiene como conjunto conjuntivo a los B-literales que aparecen en la celda a la derecha del A-literal.

Los siguientes teoremas enuncian las propiedades de corrección y completitud del sistema MESON:

Teorema 2.13 (Corrección del sistema MESON) [Loveland, 1978]

Si un conjunto de fórmulas *M-A* en formato meta-aserción es confirmado por un árbol AND utilizando el sistema MESON, la meta es consecuencia lógica del conjunto de aserciones.

Teorema 2.14 (Completitud del sistema MESON) [Loveland, 1978]

Sea *M-A* un conjunto de fórmulas en formato meta-aserción. Si la cláusula meta es consecuencia lógica del conjunto de aserciones, entonces existe un árbol AND construido usando el sistema MESON para el conjunto *M-A* que confirma la meta.

CAPÍTULO 3:

3. NUEVAS APORTACIONES A LA DEMOSTRACIÓN AUTOMÁTICA BASADAS EN RESOLUCIÓN LINEAL

3.1. INTRODUCCIÓN

En el presente capítulo se van a presentar las últimas aportaciones en el campo de la demostración automática. La mayoría de estas aportaciones se han basado de una forma más o menos directa en la resolución lineal por los motivos de sencillez y eficiencia de la misma expuestos en el capítulo anterior. Se presentan dos aportaciones que no están basadas estrictamente en resolución lineal: el demostrador SATCHMO y los sistemas nH-Prolog. La razón para incluirlos ha sido, su fuerte impacto principalmente en el campo del razonamiento automático y que para su definición y construcción se ha utilizado la programación lógica en concordancia con la relación estrecha de ésta con la demostración automática que ha dado nuevo impulso a la demostración automática. Hay que destacar que la programación lógica desde su aparición en los años setenta hasta su floreciente desarrollo de los últimos años ha estado muy relacionada con diversos aspectos del razonamiento automático, desde los más

clásicos como la demostración automática o los sistemas expertos, hasta los más modernos como el razonamiento hipotético o el razonamiento inductivo. En todos ellos es posible encontrar aproximaciones que se han basado o que han utilizado la programación lógica. La programación lógica se basa en sus fundamentos más básicos en una resolución lineal de entrada. La sencillez de estos fundamentos facilitan por un lado su definición y comprensión y, por otro lado, también provoca que se hayan obtenido implementaciones de procedimientos eficientes de la misma (por ejemplo, el procedimiento SLD). Hay que destacar que la eficiencia lograda en estas implementaciones se basa en gran medida en la ausencia de retención de información intermedia de resolución de entrada. Está ventaja para abordar problemas que no sean excesivamente complejos se convierte en inconveniente cuando los problemas tratados tienen una complejidad elevada ya que, como se comentó en el capítulo anterior la información intermedia es indispensable para aplicar ciertos criterios y métodos necesarios para el tratamiento de este tipo de problemas. Por otra parte, es bien sabido que resolución de entrada es completa únicamente para teorías Horn heredando, por tanto, la programación lógica esta limitación. Esta característica, que algunos investigadores de la demostración automática han apuntado como un importante inconveniente, ha sido destacada por otros como una importante ventaja ya que está demostrado que cualquier problema algorítmicamente resoluble es expresable mediante dichas teorías [Lloyd, 1987].

Recientemente algunos investigadores han hecho uso de la programación lógica para proponer procedimientos y demostradores completos para teorías clausales de primer orden. Evidentemente para conseguir estos procedimientos y demostradores han tenido que incorporar ciertas técnicas y métodos que suplan la incompletitud innata de la programación lógica para este tipo de teorías clausales. La pretensión de estos investigadores es la de conseguir la completitud sin perder por ello la sencillez y eficiencia de la programación lógica. Hay que indicar que relevantes investigadores en el campo de la demostración lógica [Wos y Overbeek, 1993] han apuntado que estas propuestas están limitadas por su propio origen y orientación a tratar problemas de complejidad moderada. Por otro lado, es importante no olvidar que este mismo investigador también destaca el interés de las misma, así como su éxito al abordar este tipo de problemas.

La razón de la coincidencia en el uso de resolución lineal y más concretamente de Eliminación de Modelos o el sistema MESON de las diferentes propuestas se puede encontrar en la siguientes cita de Loveland, creador del procedimiento ME y del sistema MESON [Loveland, 1978]: “La extensión más restrictiva a resolución de entrada que da origen a un procedimiento completo hasta ahora conocida en el procedimiento de Eliminación de Modelos y sus equivalentes”. Esta afirmación hecha en 1978 es, a juicio del autor de la presente tesis y de otros investigadores como Stickel [Stickel, 1988], aún cierta hoy en día.

3.2. EL DEMOSTRADOR PTTP

El demostrador PTTP (acrónimo de "Prolog Technology Theorem Prover") es una propuesta realizada por M. Stickel en 1984 [Stickel, 1984] y que posteriormente ha ido revisando y ampliando [Stickel, 1988] [Stickel, 1992]. Este demostrador se presenta como una extensión de Prolog que es completa para el cálculo de predicados de primer orden. En particular en [Stickel, 1992] se realiza una definición e implementación del mismo en el propio lenguaje Prolog. En realidad, el demostrador PTTP puede ser fácilmente presentado como una implementación del procedimiento Eliminación de Modelos o del sistema MESON. La intención del autor al construir un demostrador siguiendo los fundamentos de Eliminación de Modelos extendiendo el Prolog es la de conseguir un demostrador de teoremas completo de carácter general pero preservando la sencillez y eficiencia del Prolog en lo posible.

A continuación se presentan y discuten las características principales del demostrador PTTP. En [Stickel, 1988] [Stickel, 1992] se pueden encontrar diferentes implementaciones del demostrador y resultados experimentales sobre un conjunto de problemas bastante extenso. En la presentación del PTTP se va utilizar la forma de expresión en Prolog [Stickel, 1992] ya que ésta permite observar mejor las modificaciones que se introducen y es más cercana al propósito original del autor[4].

3.2.1. CARACTERÍSTICAS PRINCIPALES DEL DEMOSTRADOR PTTP

Esencialmente el demostrador PTTP es una implementación del procedimiento de Eliminación de Modelos de D. Loveland usando técnicas de programación lógica. Este aspecto, poco destacado en las publicaciones del M. Stickel, es fundamental para comprender adecuadamente su funcionamiento. Por otra parte, la forma de presentación del procedimiento adoptada por el autor, en la cual presenta las carencias de la programación lógica (en particular del procedimiento SLD y en particular el lenguaje Prolog y los sistemas basados en él) y la solución desarrollada en el procedimiento, es interesante y será la que se adoptará en esta presentación.

El formato que se adopta en el PTTP para la formalización consiste en el uso de las contrapositivas del conjunto de cláusulas que representan el problema a tratar. Las contrapositivas en [Stickel, 1992] se escriben usando la forma habitual del lenguaje Prolog. A continuación se presenta un ejemplo de esta formulación.

[4] En [Stickel 1988] el autor utiliza el Lisp como lenguaje soporte de la implementación del PTTP

Ejemplo 3.1 (Expresión de un problema en formato del PTTP)

Sea *S* el siguiente conjunto de cláusulas:

$C_1 = p(x) \vee q(x)$
$C_2 = \neg\, p(x) \vee q(x)$
$C_3 = p(x) \vee \neg\, q(x)$
$C_4 = \neg\, p(x) \vee \neg\, q(x)$

La representación en el formato empleado en el PTTP sería la siguiente:

$S_{11} = p(X)$:- $not\_q(X)$
$S_{12} = q(X)$:- $not\_p(X)$
$S_{21} = not\_p(X)$:- $not\_q(X)$
$S_{22} = q(X)$:- $p(X)$
$S_{31} = p(X)$:- $q(X)$
$S_{32} = not\_q(X)$:- $not\_p(X)$
$S_{41} = not\_p(X)$:- $q(X)$
$S_{42} = not\_q(X)$:- $p(X)$

Como se puede observar, se han obtenido dos contrapositivas de cada cláusula (al tener todas ellas dos literales). El primer dígito de la numeración de las contrapositivas corresponde al numero del índice de la cláusula de la cual proviene. Es importante destacar que el operador *:-* del Prolog se hará corresponder con la implicación y que el uso de predicados nuevos como *not_p* o *not_q* son necesarios representar la negación lógica y evitar así el uso de la negación como fallo que implementan los sistemas basados en el lenguaje Prolog.

Stickel introduce las principales características del procedimiento PTTP como los remedios a las carencias que impiden que los sistemas basados en el lenguaje Prolog sean completos. Este modo de explicar el procedimiento, que evidentemente resulta muy atractivo para la comunidad de la programación lógica, no facilita ni la clarificación de éste, ni su proximidad a otros procedimientos anteriores, ni las aportaciones que realiza al campo de la demostración automática. A pesar de ello se será fiel al autor y se presentarán estas características siguiendo esta forma a continuación:

UNIFICACIÓN CORRECTA

Es bien conocido que el algoritmo que los sistemas basados en el lenguaje Prolog implementan para realizar la unificación de términos no incluyen el test de ocurrencia en aras de mejorar la eficiencia del mismo. Por otra parte, en los problemas del campo de la programación lógica habitualmente el uso de este tipo de test de ocurrencia no es problemático ya que no es

usual que se realicen unificaciones de términos donde sea necesario. Sin embargo la carencia de este test en el algoritmo de unificación conlleva la pérdida de la completitud de estos sistemas y en el campo de la demostración automática esto es inadmisible. Además es interesante destacar que la obligatoriedad del uso de un algoritmo de unificación correcto, además de imponerlo el propio hecho de mantener la completitud, viene determinada porque en los problemas de este último campo es mucho más habitual que se dé la existencia de unificaciones donde sea necesario la aplicación del test de ocurrencia.

El algoritmo de unificación implementado en el PTTP incluye el test de ocurrencia, siguiendo una propuesta inicial de D. Plaisted. Para disminuir el coste de la aplicación de este algoritmo de unificación, en la fase de compilación del problema a tratar se detecta las cláusulas que obligan al uso de una unificación con test de ocurrencia. Detectadas estas cláusulas se procede a su modificación para que se aplique el algoritmo de unificación correcta cuando sean empleadas en algún paso de la demostración. El resto de cláusulas son tratadas con el algoritmo de unificación sin test de ocurrencia. La forma de saber si una cláusula necesita el algoritmo de unificación correcto es simplemente comprobando si en el literal de la cabeza (se trabaja con contrapositivas) ocurre una misma variable varias veces. El siguiente ejemplo ilustra la forma en la cual se modifica una de estas cláusulas (se supone la existencia de un predicado *unifica*(*X, Y*) que realiza la operación de unificación correcta sobre los términos *X* e *Y*)

Ejemplo 3.2 (Transformación de una cláusula para la aplicación de la unificación correcta)
Sea la siguiente cláusula en formato de entrada PTTP:

$$p(X, Y, f(X, Y)) \text{ :- } q(a, X, Y)$$

el resultado de la transformación es el siguiente:

$$p(X, Y, f(X1, Y1)) \text{ :- } unifica(X, X1), unifica(Y, Y1), q(a, X, Y)$$

Como se puede observar la transformación evita la posibilidad de que se produzca un error al emplear la unificación sin test de ocurrencia sobre el átomo de la cabeza modificando ésta e introduciendo dos nuevos átomos en el cuerpo que realizan de forma retrasada la unificación correcta.

Claramente el uso de la unificación correcta será necesaria tanto en las operaciones de resolución de entrada y en las operaciones de resolución de ancestro. En las operaciones del primer tipo es posible aplicar la técnica mencionada en el ejemplo anterior; sin embargo esto no es aplicable en las operaciones de resolución de ancestro, ya que al ser dinámico el conjunto de ancestros durante las demostraciones no permite que se pueda realizar ningún tipo de test en tiempo de compilación.

SISTEMA DE INFERENCIA COMPLETO

Es bien conocido que los sistemas basados en el lenguaje Prolog al implementar el procedimiento de resolución SLD (que es una restricción de resolución de entrada) no son completos para el tratamiento de teorías clausales generales. Las modificaciones que son necesarias realizar en un sistema basado en el lenguaje Prolog para que sea una sistema de inferencia completo son las siguientes:

- Ampliar el conjunto de cláusulas de entrada con las contrapositivas de cada una de las cláusulas iniciales. Esto es necesario si se va a seguir empleando la forma de aplicar resolución de entrada de los sistemas basados en el Prolog, en los que la cláusula padre cercano es siempre una meta de la forma $\leftarrow L_1 \wedge \ldots \wedge L_n$ y la cláusula padre lejano está en forma implicativa, realizando la resolución entre el literal seleccionado en la cláusula padre cercano y la cabeza de la cláusula padre lejano.

- Tratamiento de las respuestas disyuntivas y reutilización de la cláusula raíz. Claramente en el tratamiento de teorías clausales generales con presencia de disyunción en la cabeza de las cláusulas (en forma implicativa) es necesario incluir en la teoría la propia cláusula raíz que representa al teorema o a la pregunta, tanto para la demostración de teoremas como para la obtención de respuestas. En el formato del PTTP es necesario obtener todas las contrapositivas de la cláusula que corresponde al teorema o pregunta e incluir en la teoría todas las contrapositivas asociadas.

Ejemplo 3.3 (Tratamiento de la disyunción)

Sea el siguiente conjunto inconsistente de cláusulas en formato para el PTTP:

p(*a*) :- *not_p*(*b*), *not_p*(*c*)
p(b) :- *not_p*(a), *not_p*(*c*)
p(*c*) :- *not_p*(a), *not_p*(b)
:- *p*(*X*) Cláusula raíz

En este caso es necesario que la cláusula raíz se incluya en el conjunto de cláusulas de entrada, ya que si no el demostrador no es capaz de refutar este problema.

- Añadir la resolución de ancestro como nueva regla de inferencia. Los sistemas basados en el Prolog incluyen una forma particular de resolución de entrada pero no realizan de ninguna forma la resolución de ancestro, necesaria para obtener un sistema de inferencia completo. El PTTP utiliza una resolución de ancestro idéntica a la del procedimiento de Eliminación de Modelos, esto es, cuando el literal seleccionado en la cláusula en curso en la derivación es complementario de uno de los literales previamente seleccionados (adecuadamente instanciados por las sustituciones aplicadas hasta ese punto) y sus átomos

son unificables, la siguiente cláusula en la derivación se obtiene eliminando el literal seleccionado y aplicando la sustitución de unificación. Stickel propone que se mantenga una lista de los ancestros, literales seleccionados en la derivación en curso, indexada por el símbolo de sus predicados[5]. La forma de generar esta lista y llevarla dinámicamente en las demostraciones es realizando una transformación en tiempo de compilación de las contrapositivas del problema añadiendo en cada uno de los literales que las forman un nuevo argumento que corresponde a la mencionada lista.

ESTRATEGIA DE BÚSQUEDA COMPLETA

Para que un procedimiento de demostración sea completo, en general, es necesario que el recorrido del espacio de búsqueda sea correcto, en el sentido de encontrar permitir alcanzar siempre en un tiempo finito cualquier elemento de dicho espacio. De forma más concreta para los demostradores basados en resolución esto se traduce en que la construcción del árbol de demostración y su recorrido deben permitir que el demostrador compruebe en un tiempo finito cualquier nodo de dicho árbol. Las estrategias de recorrido de árboles se pueden estudiar con detalle en [Horowitz, 1978] [Nilson, 1987]. Los sistemas basados en el Prolog construyen el árbol de demostración seleccionando siempre el literal más a la izquierda de la cláusula en curso y además utilizan la estrategia de recorrido en profundidad, consiguiendo así una eficiencia espacial óptima pero perdiendo la completitud con ello.

El PTTP utiliza una estrategia de búsqueda denominada en profundidad iterada, utilizada originalmente en programas de ajedrez. Esta estrategia se basa en realizar una búsqueda en profundidad acotada iterativamente, de forma que de una pasada a la siguiente se incremente la cota de profundidad. Así, esta estrategia para encontrar la prueba de un cierto problema realiza una búsqueda en profundidad con una cota máxima, por tanto finita, y en caso de no encontrar dicho prueba va considerando cotas de profundidad mayores, hasta localizar la prueba, caso de existir. Obviamente el uso de esta estrategia supone una penalización temporal frente a la búsqueda en profundidad, ya que se repiten exploraciones ya realizadas. Stickel muestra que con esta estrategia se consigue que el procedimiento sea completo manteniendo un coste espacial mínimo. En particular en [Stickel, 1988] compara estos dos costes en función del factor de ramificación del árbol de prueba, diciendo que si este factor es b, entonces el procedimiento que use la estrategia de búsqueda en profundidad iterada efectúa $b(b\text{-}1)$ veces tantas operaciones como efectuaría el mismo procedimiento siguiendo la estrategia de búsqueda en profundidad. Si el factor de ramificación es moderado se puede concluir que este coste no es excesivamente

[5] Esta técnica de indexación sólo la realiza en el artículo [Stickel, 1988], en el cual propone una implementación en Lisp del PTTP En el artículo [Stickel, 1992], al utilizar el Prolog como lenguaje de implementación, no realiza esta indexación para no restarle claridad a la presentación.

grande. Por otra parte es evidente que asintóticamente esta estrategia es óptima tanto temporal como espacialmente dado el crecimiento exponencial del espacio de búsqueda a medida que aumenta la profundidad.

El criterio para establecer la cota de profundidad permite incorporar ciertos aspectos cercanos a la heurística que consiguen mejorar de forma substancial el comportamiento del procedimiento. Esencialmente los criterios fundamentales son los siguientes:

1) Calcular la cota de profundidad mediante el número de pasos de inferencia realizados por la derivación, esto es, sobre la longitud de la derivación.
2) Calcular la cota de profundidad a partir del tamaño del conjunto de ancestros.
3) Calcular la cota de profundidad sobre el número de usos que de cada cláusula se hace en la demostración.

El comportamiento del procedimiento según se use un criterio u otro varía notablemente, como indica Stickel en [Stickel, 1992]. Generalmente el primer criterio es el que da mejores resultados, aunque existen problemas para los que los otros criterios pueden dar lugar resultados llamativamente mejores. La razón por la cual el primer criterio es el más efectivo reside en que el crecimiento del espacio de búsqueda entre iteraciones sucesivas crece lentamente. En los otros dos criterios y cuando la profundidad es elevada, un incremento ligero de la cota puede suponer un crecimiento explosivo del espacio de búsqueda comprendido entre ambas iteraciones. El criterio usado en el PTTP es el 1). Otros demostradores, por ejemplo SETHEO propuesto por Bibel y otros en [Letz, Schuman, Bayerl y Bibel, 1992] con mejores resultados experimentales que el PTTP, utilizan otros criterios para establecer la cota de profundidad. En particular SETHEO, que posteriormente se presentará, utiliza los tres criterios aplicando al mismo problema tres versiones de su demostrador en paralelo. Cada una de estas versiones utiliza un criterio distinto. Como tiempo de cómputo proponen que se tome el triple del tiempo obtenido por la versión del demostrador que encuentra antes la respuesta.

Por otra parte también es interesante destacar que la aplicación de ciertos métodos de corte y poda es dependiente del criterio para establecer la cota de profundidad. Un método simple de poda, usado en el PTTP, cuya aplicación está indisolublemente unida al uso del criterio 1) se basa en el simple hecho de que para resolver la cláusula en curso en una derivación con n literales, es necesario como mínimo la aplicación de n pasos de inferencia. Por tanto, si la cota de profundidad es m y la profundidad del árbol de búsqueda actual es p, la demostración de cualquier cláusula con un número de literales mayor que $m–p$ se poda siguiendo dicho criterio.

Al igual que ocurría en el punto anterior, en el PTTP para implementar de forma eficiente y simple la estrategia de búsqueda completa así como la poda asociada a ella, se realiza una transformación de las contrapositivas de las cláusulas añadiéndoles argumentos y código

adicional que permiten conocer en todo momento en la derivación la profundidad de la misma y realizar la aplicación siempre que sea posible de la poda arriba mencionada. Esta transformación puede estudiarse en profundidad en [Stickel, 1992].

A continuación se presenta un ejemplo de funcionamiento del demostrador PTTP

Ejemplo 3.4 (Demostrador PTTP)

Sea S el siguiente conjunto inconsistente de cláusulas:

$C_1 = p(x) \vee q(x)$ $\quad\quad$ $C_3 = \mathrm{q}(x) \leftarrow p(x)$

$C_2 = p(x) \leftarrow q(x)$ $\quad\quad$ $C_4 = \leftarrow p(a) \wedge q(a)$

cuya transformación al formato PTTP da lugar a las siguientes sentencias:

$S_{11} = p(X)$:- $not\_q(X)$ $\quad\quad$ $S_{31} = q(X)$:- $p(X)$

$S_{12} = q(X)$:- $not\_p(X)$ $\quad\quad$ $S_{32} = not\_p(X)$:- $not\_q(X)$

$S_{21} = p(X)$:- $q(X)$ $\quad\quad$ $S_{41} = not\_p(a)$:- $q(a)$

$S_{22} = not\_q(X)$:- $not\_p(X)$ $\quad\quad$ $S_{42} = not\_q(a)$:- $p(a)$

Fácilmente se puede observar que la numeración asociada permite saber de qué cláusula proviene cada una de las sentencias. A continuación se presenta el árbol de demostración que obtiene el demostrador PTTP desde la cláusula inicial :-$p(a)$, $q(a)$, que corresponde a la sentencias S_{41} y S_{42} pero en formato negativo.

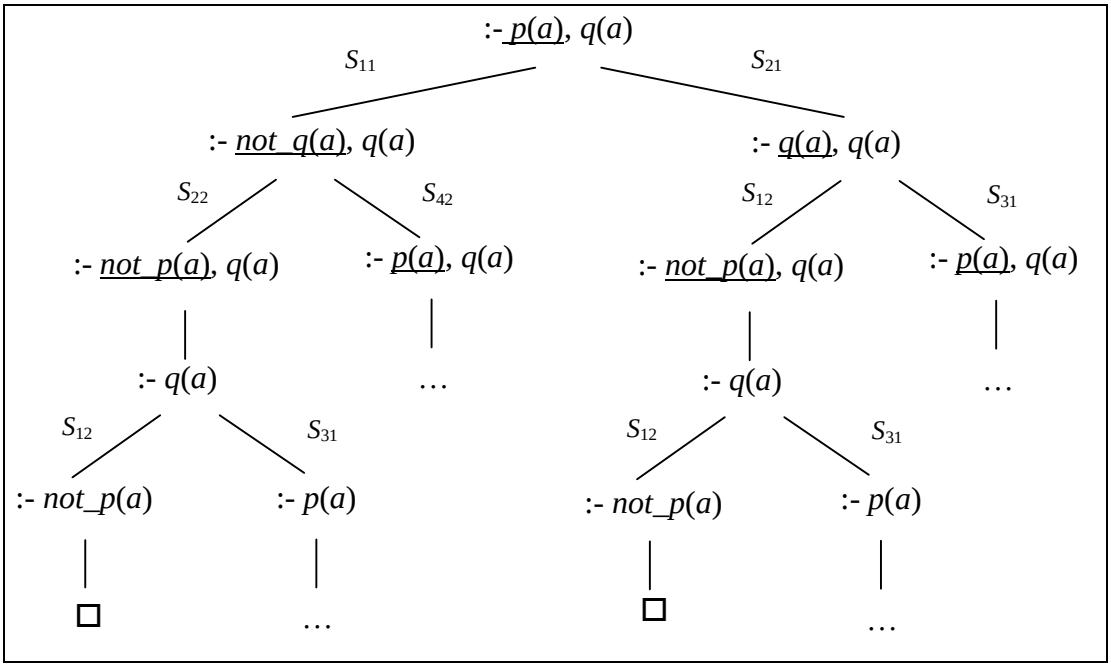


Figura 3.1: Árbol de demostración del PTTP para el Ejemplo 3.4

En la figura anterior los nodos corresponden a cláusulas de la derivación en las cuales se subraya el literal seleccionado (caso de ser necesario). Los arcos se corresponden con un paso de inferencia, que en el caso de realizarse mediante resolución de entrada van etiquetados con la cláusula de entrada utilizada; en caso el caso de resolución de ancestro el arco no va etiquetado. Los ancestros del literal seleccionado en una cierta cláusula *C* del árbol se pueden obtener sin más que revisar los literales seleccionados (se denotan subrayándolos) en las cláusulas que van desde *C* hasta la cláusula raíz. Como se puede observar en el árbol existen varios nodos con el símbolo "…" que indica que el árbol se extiende a partir de ese nodo de manera ilimitada (fácilmente se puede observar que su cláusula padre es una cláusula cuyo literal ya ha sido seleccionado anteriormente).

Las tres características anteriormente presentadas del demostrador PTTP evitan las carencias de los sistemas basados en el Prolog, alcanzando así la completitud. Además Stickel propone un conjunto de refinamientos para mejorar la eficiencia del demostrador. Estos refinamientos van orientados a reducir el espacio de búsqueda mediante podas que esencialmente se relacionan con las ideas básicas en la demostración automática de eliminación de tautologías y utilización de la subsumción. A continuación se presentan estos refinamientos:

*R*1. *Poda por igualdad con un ancestro*

Si el literal seleccionado en la meta en curso es idéntico a uno de sus ancestros, la derivación falla en ese punto.

*R*2. *Reducción del espacio de búsqueda por utilización de la respuesta más general*

a) Si el literal seleccionado en la meta en curso es complementario a uno de sus ancestros y además los átomos de ambos son idénticos, entonces realizar la operación de resolución de ancestro pero no realizar sobre esta meta ninguna aplicación de otra regla de inferencia.
b) Si el literal seleccionado en la meta en curso es subsumido por una cláusula unitaria de la teoría, entonces realizar una operación de resolución de entrada pero no realizar sobre esta meta ninguna aplicación de otra regla de inferencia.

El primer refinamiento *R*1 equivale a la poda de subsumción limitada a la derivación en curso, como ya se mencionó en la exposición del sistema MESON. En programación lógica este refinamiento ha sido utilizado en diversos procedimientos para la detección de bucles y la consecución de terminación [Vielle, 1986]. A continuación se presentan algunos ejemplos que ilustran estos refinamientos:

Ejemplo 3.5 (Utilización del refinamiento *R*1: poda por igualdad con un ancestro)
Sea el siguiente conjunto de cláusulas en formato de entrada PTTP:

S_1 = :- $p(X)$, $r(X)$ cláusula raíz
S_2 = $p(X)$:- $p(X)$, $q(Y)$

La figura de abajo muestra el árbol de demostración PTTP

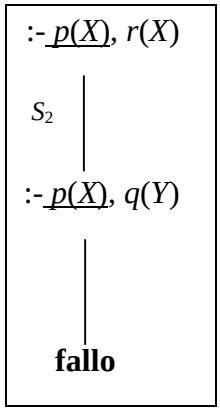


Figura 3.2: Árbol del Ejemplo 3.1

Como se puede observar en la figura el literal seleccionado en la segunda cláusula de la única derivación del árbol, esto es $p(X)$, tiene como único ancestro a $p(X)$, pudiéndose aplicar el refinamiento R1 que permita realizar la poda y concluir de forma finita dicho árbol. Así para este problema el demostrador sería capaz de concluir la consistencia del problema planteado. Si este refinamiento no se aplica el árbol de demostración sería infinito y el demostrador no sería capaz de dar una respuesta al problema planteado. Es sencillo observar que este refinamiento corresponde a una poda por subsumción del resolvente por una cláusula de la teoría. El

resolvente de las cláusulas S_1 y S_2 (en formato de entrada PTTP) es la cláusula :- $r(X)$, $p(X)$, $q(Y)$ que claramente es subsumida por S_1.

El siguiente ejemplo quiere ilustrar la pérdida de la completitud si se permite realizar la poda anterior cuando el literal de la cláusula en curso es subsumida por uno de sus ancestros (se llama la atención sobre el hecho de que el refinamiento exige la igualdad sintáctica del literal y su ancestro).

Ejemplo 3.6 (Pérdida de la completitud por la aplicación del refinamiento R1 cuando se produce la subsumción del literal de la cláusula en curso por uno de sus ancestros)

Considérese la teoría formada por las cláusulas siguientes:

S_1 = :- $p(X)$, $r(X)$ cláusula raíz
S_2 = $p(X)$:- $p(Y)$
S_3 = $r(a)$
S_4 = $p(b)$

entonces la figura de abajo muestra el árbol de demostración PTTP con el refinamiento modificado.

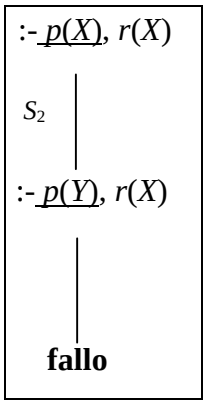


Figura 3.3: Árbol del Ejemplo 3.1

El literal seleccionado en la segunda cláusula es $p(Y)$ y su único ancestro es $p(X)$ que claramente lo subsume. Por tanto se aplicaría la poda y la derivación fallaría. Sin embargo es sencillo comprobar que la teoría es inconsistente y que existe una refutación PTTP para la misma sin aplicar dicho refinamiento. Haciendo uso de la analogía de este refinamiento con la poda por subsumción, la aplicación incorrecta de este refinamiento correspondería, en este ejemplo, a suponer que la cláusula :- $r(X)$, $p(Y)$, resolvente de S_1 y S_2, es subsumida por S_1, esto es :- $p(X)$, $r(X)$, que claramente se ve que no es correcto.

Ejemplo 3.7 (Poda aplicada por el refinamiento *R*2 a), reducción por respuesta más general)

Sea el siguiente conjunto de cláusulas en formato de entrada PTTP:

$S_1 =$:- $p(X)$, $t(X)$	cláusula raíz
$S_2 =$ $p(X)$:- $q(X)$	
$S_3 =$ $q(X)$:- $not\_p(X)$	
$S_4 =$ $not\_p(X)$:- ***grande***	(para el literal *grande* existe un árbol de búsqueda extenso)
S_5 …	

La figura siguiente muestra dos árboles de demostración PTTP para este problema. En el de la derecha no se hace uso del refinamiento R2 a) y en el de la izquierda sí.

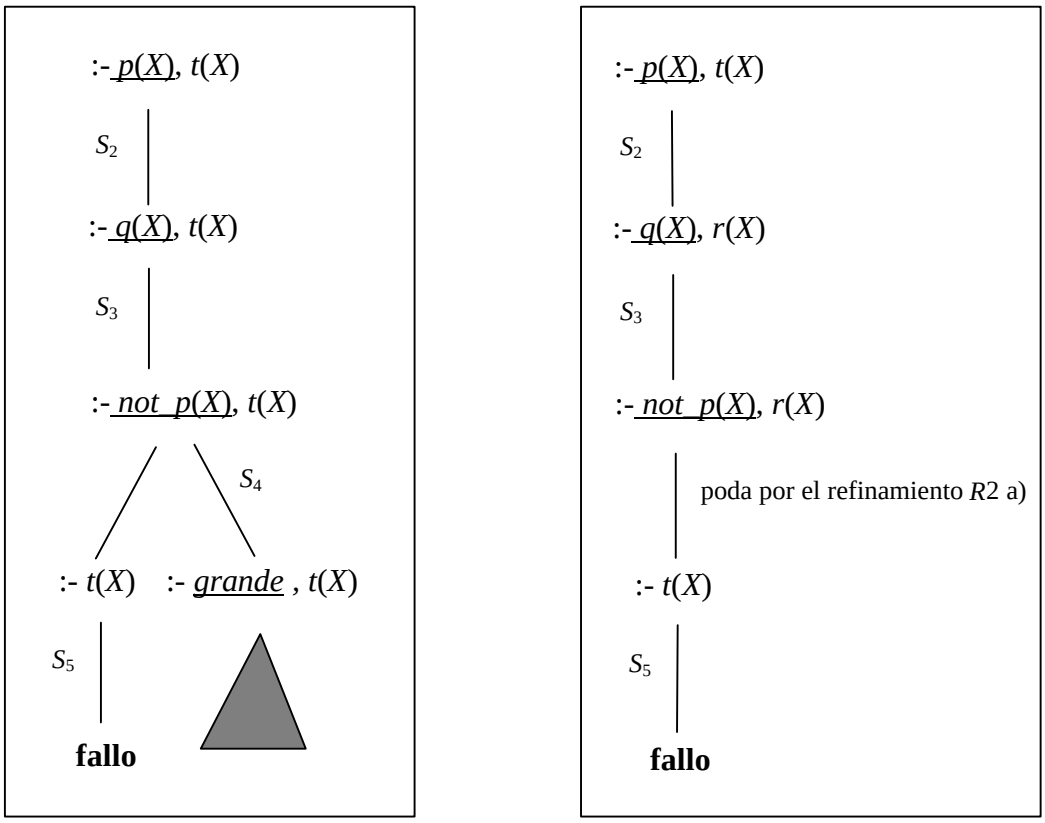


Figura 3.4: Árboles de demostración PTTP para el Ejemplo 3.7

Como se puede observar en la figura de la izquierda, el árbol de demostración PTTP tienen un tamaño que puede ser considerable si el subárbol que se genera para la meta *grande* lo es. Este subárbol aparece por la aplicación de un paso de resolución de entrada a partir de la meta $not\_p(X)$ usando la cláusula S_4. Debido a que también es posible aplicar un paso de resolución de ancestro empleando el ancestro $p(X)$ se genera otro nodo hermano del anterior.

En el árbol de demostración PTTP de la derecha, la meta obtenida por la aplicación de un paso de resolución de entrada a partir de la meta $not\_p(X)$ se poda por la aplicación del

refinamiento *R*2 a), ya que también se realiza sobre ella un paso de resolución de ancestro y un ancestro complementario. Esto provoca una reducción del tamaño del espacio de búsqueda que en algunos casos puede ser muy considerable.

A continuación se presenta un ejemplo para ilustrar que la aplicación del refinamiento *R*1 b) se ha de realizar cuando los átomos de los literales son idénticos, ya que si no se pierde la completitud del procedimiento.

Ejemplo 3.8 (Aplicación incorrecta del refinamiento *R*2 a))

Sea el siguiente conjunto de cláusulas en formato de entrada PTTP:

S_1 = :- $p(a)$ cláusula raíz

S_2 = $p(a)$:- $p(b)$

S_3 = $not\_p(b)$:- $not\_p(a)$ S_4 = $p(b)$:- q

S_5 = q :- $not\_p(X)$, $r(X)$ S_6 = $r(b)$

La figura siguiente muestra el árbol de demostración PTTP para este ejemplo:

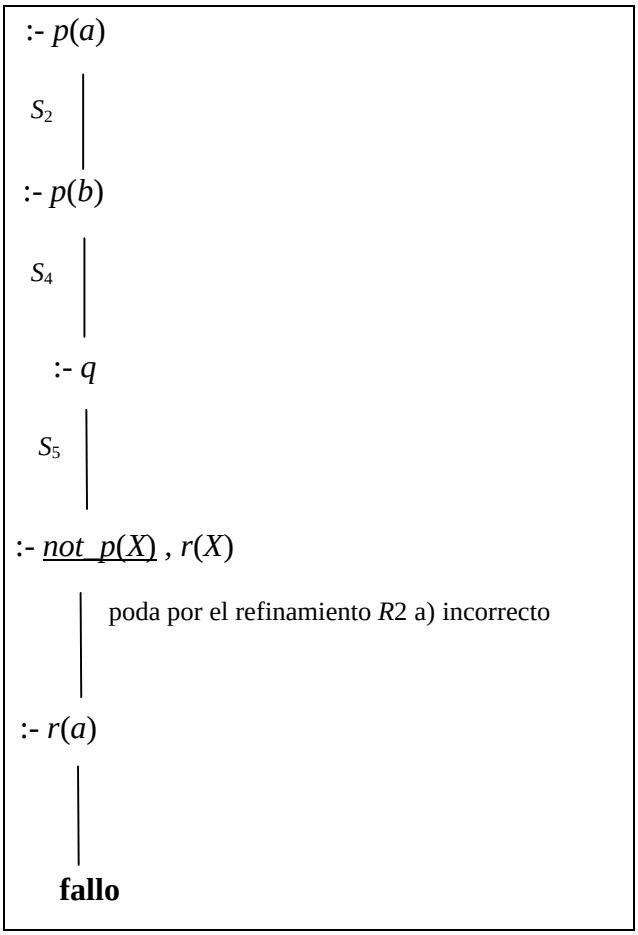


Figura 3.5: Árbol de demostración para el Ejemplo 3.8

La aplicación incorrecta del refinamiento *R*2 a), ya que la meta en esa cláusula en curso es *not_p*(*X*) y el ancestro usado es *p*(*a*), poda el subárbol resultante de aplicar un paso de resolución de entrada a dicha meta usando como cláusula de entrada S_3. En este caso esta poda es dramática ya que en el espacio eliminado se encuentran todas las refutaciones asociadas al problema. Este ejemplo ilustra un caso en el cual el procedimiento resultante del uso de este refinamiento inválido pierde incluso la corrección. Como se puede observar el árbol obtenido es finito y sus ramas son falladas por lo que se concluiría que el conjunto de cláusulas es consistente, cuando realmente no lo es.

Del refinamiento *R*2 b) no se incluye un ejemplo, ya que es sencillo construir uno. Fácilmente, se podrá observar que la aplicación de este refinamiento se corresponde con el hecho de que la respuesta obtenida es la más general posible al refutar el literal de la cláusula en curso cuando se dan las circunstancias de dicho refinamiento, esto es cuando el literal es subsumido por una cláusula unitaria.

Otros refinamientos mencionados en [Stickel, 1988] [Stickel, 1992] que no se encuentran implementados en el demostrador PTTP son los siguientes:

*R*3. *Uso de criterios de poda más precisos en función del estado de la demostración*

El criterio que se emplea en el PTTP es uno de los más simples: se poda la derivación si los literales en la cláusula en curso superan la cota de profundidad. Existen otros criterios más sofisticados que permiten realizar una poda mayor sobre el espacio de búsqueda. Estos criterios estiman de una forma más precisa el número de pasos de inferencia necesarios para finalizar una derivación, apoyándose en un análisis sintáctico de las cláusulas que puede realizarse en tiempo de compilación o de forma dinámica.

*R*4. *Poda de derivaciones hermanas por obtención de la respuesta más general*

Si una meta es resuelta sin instanciación de sus variables se puede podar toda derivación hermana de la refutación que ha dado lugar a esta respuesta.

*R*5. *Estudio del coste temporal de algunos refinamientos*

El coste temporal de algunos refinamientos puede ser elevado en comparación con el ahorro espacial obtenido. Puede ser interesante aplicar algunos de estos refinamientos al principio del recorrido del árbol de demostración, que es cuando se puede realizar las mayores podas, y desactivarlos a partir de una cierta profundidad.

*R*6. *Refinamiento positivo*

Basándose en una propuesta de D. Plaisted [Plaisted, 1990], que posteriormente se presentará en este capítulo, Stickel propone la posibilidad de limitar el número de ancestros

requeridos. Así solamente los literales negativos seleccionados previamente son considerados ancestros, de forma que únicamente sobre los literales positivos de las cláusulas en curso se puede realizar resoluciones de ancestros.

*R*7. *Inclusión de tratamiento de la igualdad*

Es bien conocido en el campo de la demostración automática que el tratamiento de la igualdad requiere de un tratamiento especial que optimice su manipulación [Loveland, 1978] [Chang y Lee, 1973]. Demodulación, paramodulación y otros son métodos clásicos en la literatura para el tratamiento de este aspecto. Stickel comenta que sería posible incorporar al PTTP la regla de inferencia de paramodulación, de forma similar a como se realiza en el MESON, pero que esto supone un incremento del factor de ramificación del árbol de demostración que resulta excesivo para el PTTP y que penaliza demasiado su prestaciones.

3.2.2. CONCLUSIONES SOBRE EL DEMOSTRADOR PTTP

El demostrador PTTP es una buena implementación tanto de Eliminación de Modelos como del sistema MESON dentro del marco de la programación lógica. Las principales aportaciones que ha dado este trabajo son de tipo tecnológico y están relacionadas con la búsqueda de soluciones a la incompletitud del paradigma de la programación lógica. Cabe destacar entre ellas el uso de la búsqueda en profundidad iterada para evitar los problemas de la búsqueda en profundidad y la incorporación de una fase de precompilación que realiza una transformación de la cláusulas originales a sentencias Prolog. Estas sentencias corresponden al conjunto de contrapositivas e incluyen el código necesario para el tratamiento de diferentes cuestiones (unificación correcta, resolución de ancestro, etc.). También hay que destacar que el demostrador PTTP, a pesar de que obtiene unos resultados muy buenos para un conjunto de problemas elevado [Stickel, 1988], no tiene la capacidad de tratar problemas de una complejidad alta, como destaca Wos en [Wos y Overbeek, 1993]. Ello es debido a que, como menciona el anterior autor, el almacenamiento de información intermedia y el uso de ésta para evitar el crecimiento desmesurado del espacio de búsqueda no es demasiado elevado (hecho absolutamente normal ya que este demostrador está basado en Eliminación de Modelos, o más primitivamente, en resolución lineal, y por tanto, intenta reducir al mínimo la información intermedia almacenada sin perder la completitud).

3.3. EL SISTEMA MESON DEFINIDO COMO UN SISTEMA DEDUCTIVO BASADO EN SECUENCIAS

El sistema que se va a presentar a continuación corresponde a una propuesta de David Plaisted [Plaisted, 1990] en la cual realiza una redefinición de Eliminación de Modelos utilizando para ello un sistema deductivo basado en el uso de secuencias. Plaisted propone en [Plaisted, 1988] una posible implementación de dicho sistema basado en el lenguaje Prolog que hay que considerar esencialmente de tipo académico. Su sistema se puede entender como una alternativa al sistema MESON en la cual el formato de las cláusulas y las reglas de inferencias se formalizan sobre el concepto de secuencia. Además también introduce un refinamiento de la propuesta inicial que consiste en una modificación del tratamiento de la resolución de ancestro de forma que únicamente se consideran como ancestros en un cierto punto de la demostración a los literales positivos anteriormente seleccionados. Esta forma simplificada de resolución de ancestro, denominada por Plaisted refinamiento positivo, será estudiada con profundidad.

A continuación se introducen algunas definiciones previas necesarias para la presentación adecuada del sistema.

Definición 3.1 (Secuencia) [Plaisted, 1990]

Una secuencia es una sentencia de la forma $G \rightarrow L$, donde G es una lista (conjunto) de literales y L es un literal. Si G está vacío entonces la secuencia se puede denotar por $\rightarrow L$ o simplemente L.

El significado de una sentencia $G \rightarrow L$ es que la conjunción de los literales de G implica a L.

Definición 3.2 (Sistema deductivo basado en secuencias) [Plaisted, 1990]

Un sistema deductivo basado en secuencias es un sistema deductivo en el cual todas las reglas de inferencia son de la forma:

$$\frac{\Gamma_1 \rightarrow L_1, \Gamma_2 \rightarrow L_2, \ldots, \Gamma_n \rightarrow L_n}{\Gamma \rightarrow L}$$

Cada una de estas reglas se interpreta de la manera siguiente: si se demuestra cada secuencia $\Gamma_i \rightarrow L_i$ para $1 \leq i \leq n$, entonces se infiere la secuencia $\Gamma \rightarrow L$. Expresado esto en términos del formato de reducción de problemas, cada una de las secuencias $\Gamma_i \rightarrow L_i$ se pueden considerar como las submetas de la meta $\Gamma \rightarrow L$.

3.3.1. EL SISTEMA MESON DEFINIDO COMO UN SISTEMA DEDUCTIVO BASADO EN SECUENCIAS

La definición del sistema MESON, en su versión simple que no incluye el tratamiento de la igualdad, en la propuesta inicial de Plaisted es de carácter proposicional, aunque fácilmente se puede extender a la aplicación de teorías de primer orden. El formato de las cláusulas iniciales se basa en el lenguaje Prolog y es similar al utilizado por el demostrador PTTP

Definición 3.3 (El sistema MESON proposicional basado en secuencias)

Sea S un conjunto de cláusulas. Las siguientes reglas forman el sistema MESON proposicional para S:

1. Sea C una cláusula de S, expresada de la forma L:- L_1, L_2, …, L_n, entonces para C y para cada una de sus contrapositivas existe una regla en el sistema deductivo de la forma siguiente:

$$\frac{\left[\Gamma_1, \neg L_1 \rightarrow L_1\right], \left[\Gamma_1, \neg L_2 \rightarrow L_2\right], \ldots, \left[\Gamma_n, \neg L_n \rightarrow L_n\right]}{\Gamma \rightarrow L}$$

Obsérvese que existirán para la cláusula C, n+1 reglas como la anterior que corresponden a las n+1 contrapositivas de C. Si la cláusula C es unitaria, por ejemplo contiene a un único literal L, la regla anterior queda de la forma $\Gamma \rightarrow L$.

2. Para la cláusula inicial o soporte elegida, que en el formato usado se representa en Prolog como *falso*:- L_1, L_2, …, L_n, se incluye adicionalmente a las n reglas correspondientes a las contrapositivas la regla siguiente:

$$\frac{\left[\Gamma_1, \neg L_1 \rightarrow L_1\right], \left[\Gamma_1, \neg L_2 \rightarrow L_2\right], \ldots, \left[\Gamma_n, \neg L_n \rightarrow L_n\right]}{\Gamma \rightarrow \mathit{falso}}$$

De esta forma para la cláusula inicial o soporte con n literales existirán n+1 reglas en el sistema deductivo.

3. Además el sistema deductivo contiene el denominado *axioma de asunción*, que se formaliza como sigue:

$$\Gamma \rightarrow L, \text{ si } L \in \Gamma$$

El sistema formado por todas las reglas construidas a partir de las cláusulas de S, junto con el axioma de asunción, es denotado por $\vdash_S$. Para demostrar que un conjunto de cláusulas S es inconsistente hay que probar la secuencia $\varnothing \rightarrow \mathit{falso}$ o abreviadamente *falso*.

Si se compara la definición del sistema MESON proposicional basado en secuencias anterior con el sistema MESON original, o con el procedimiento de Eliminación de Modelos, es sencillo observar que el antecedente de las secuencias contendrá el conjunto de ancestros usados para la demostración de las distintas metas que corresponden a los consecuentes de las secuencias. A continuación se presenta un sencillo ejemplo para clarificar el funcionamiento del sistema.

Ejemplo 3.9 (Sistema MESON proposicional basado en secuencias)

Sea *S* el siguiente conjunto de cláusulas en formato Prolog.

S_1 = :- p, q cláusula raíz
S_2 = p :- r
S_3 = r.
S_4 = q :- $\neg m$
S_5 = q :- m

El conjunto de reglas deductiva que se obtienen a partir de estas cláusulas es el siguiente:

$$R_{11}\ \frac{[\Gamma, \neg p \to p], [\Gamma, \neg q \to q]}{\Gamma \to \mathit{falso}} \qquad R_{12}\ \frac{\Gamma, \neg q \to q}{\Gamma \to \neg p}$$

$$R_{13}\ \frac{\Gamma, \neg p \to p}{\Gamma \to \neg q}$$

$$R_{21}\ \frac{\Gamma, \neg r \to r}{\Gamma \to p} \qquad R_{22}\ \frac{\Gamma, p \to \neg p}{\Gamma \to \neg r}$$

$$R_{31}\ \Gamma \to r$$

$$R_{41}\ \frac{\Gamma, m \to \neg m}{\Gamma \to q} \qquad R_{42}\ \frac{\Gamma, q \to \neg q}{\Gamma \to m}$$

$$R_{51}\ \frac{\Gamma, \neg m \to m}{\Gamma \to q} \qquad R_{52}\ \frac{\Gamma, q \to \neg q}{\Gamma \to \neg m}$$

A continuación se muestra la secuencia de aplicaciones de las reglas que permiten demostrar que el conjunto *S* es inconsistente:

$$\frac{\dfrac{\neg p, \neg r \to r}{\neg p \to p} \qquad \dfrac{\dfrac{\neg q, \neg m, q \to \neg q}{\neg q, \neg m \to m}}{\neg q \to q}}{\mathit{falso}}$$

La demostración se puede entender de la forma siguiente: se demuestra la inconsistencia de *S* (o sea, *falso*) si se puede demostrar *p* y *q* (R_{11}); *p* se demuestra si se prueba *r* (R_{21}) y esto es inmediato ya que *r* es una cláusula unitaria (R_{31}); por otro lado *q* se demuestra si se prueba $\neg m$ (R_{41}), $\neg m$ se demuestra si se prueba $\neg q$ (R_{42}), y $\neg q$ se demuestra por el axioma de asunción. Obsérvese que esta última operación se corresponde claramente con una reducción, ya que el antecedente de la secuencia es $\neg q, m, q$ que contiene dos literales opuestos. Claramente cada aplicación del axioma de asunción es equivalente a una reducción de Eliminación de Modelos. Las otra reglas aplicadas, provenientes de contrapositivas de cláusulas iniciales, corresponde sencillamente a extensiones.

Para extender el sistema anterior de forma que se puedan tratar teorías de primer orden es necesario reconsiderar las reglas deductivas de manera que se refleje la búsqueda de instancias adecuadas de las reglas realizando unificaciones entre literales adecuados. Para ello se denotara la instancia de una secuencia $\Gamma \rightarrow L$, obtenida aplicando una cierta sustitución σ a cada uno de los elementos de la secuencia, por $[\Gamma \rightarrow L]\sigma$. A continuación se presentan las modificaciones para definir el sistema MESON para teorías de primer orden basado en secuencias:

1. El axioma de asunción se redefine de la forma siguiente:

   $[\Gamma \rightarrow L]\sigma$ si existe un unificador más general σ para el literal *L* y un literal *L'* del antecedente Γ.

2. Las reglas que provienen de las cláusulas iniciales han de tener en cuenta la presencia de variables. Para ello se puede definir las reglas de la siguiente forma: supóngase que L:- $L_1, L_2, \ldots, L_n$ es una cláusula en formato Prolog; entonces para cada contrapositiva de está cláusula y para cada substitución σ existe regla deductiva de la forma,

$$\frac{\left[\Gamma_1, \neg L_1 \rightarrow L_1\right]s, \left[\Gamma_1, \neg L_2 \rightarrow L_2\right]s, \ldots, \left[\Gamma_n, \neg L_n \rightarrow L_n\right]s}{[\Gamma \rightarrow L]s}$$

Así cuando una meta en curso puede unificar con la cabeza de una cierta contrapositiva existe una regla deductiva que permite su descomposición en las submetas correspondientes a los literales del cuerpo de la contrapositiva. A continuación se presenta un ejemplo que ilustra el funcionamiento del sistema MESON.

Ejemplo 3.10 (Sistema MESON basado en secuencias)

Sea *S* el siguiente conjunto inconsistente de cláusulas:

$S_1 = p(x) \vee q(x)$ $\quad$ $S_3 = p(x) \vee \neg q(a)$

$S_2 = \neg p(x) \vee q(x)$ $\quad$ $S_4 = \neg p(x) \vee \neg q(x)$

El conjunto de contrapositivas en formato Prolog es el siguiente:

$S_{11} = p(X)$:- $\neg\, q(X)$ $\qquad$ $S_{12} = q(X)$:- $\neg\, p(X)$

$S_{21} = \neg\, p(X)$:- $\neg\, q(X)$ $\qquad$ $S_{22} = q(X)$:- $p(X)$

$S_{31} = p(X)$:- $q(a)$ $\qquad$ $S_{32} = \neg\, q(a)$:- $\neg\, p(X)$

$S_{41} = \neg\, p(X)$:- $q(X)$ $\qquad$ $S_{42} = \neg\, q(X)$:- $p(X)$

S_{43} = :- $p(X), q(X)$ $\qquad$ (cláusula soporte o inicial)

El conjunto de reglas del sistema MESON basado en secuencias para este problema sería el siguiente:

$$R_{11}\ \frac{\Gamma, q(X) \rightarrow \neg\, q(X)}{\Gamma \rightarrow p(X)} \qquad R_{12}\ \frac{\Gamma, p(X) \rightarrow \neg\, p(X)}{\Gamma \rightarrow q(X)}$$

$$R_{21}\ \frac{\Gamma, q(X) \rightarrow \neg\, q(X)}{\Gamma \rightarrow \neg\, p(X)} \qquad R_{22}\ \frac{\left[\Gamma, \neg\, p(X) \rightarrow p(X)\right]}{\Gamma \rightarrow q(X)}$$

$$R_{31}\ \frac{\Gamma, \neg\, q(a) \rightarrow q(a)}{\Gamma \rightarrow p(X)} \qquad R_{32}\ \frac{\Gamma, p(X) \rightarrow \neg\, p(X)}{\Gamma \rightarrow \neg\, q(a)}$$

$$R_{41}\ \frac{\Gamma, \neg\, q(X) \rightarrow q(X)}{\Gamma \rightarrow \neg\, p(X)} \qquad R_{42}\ \frac{\Gamma, p(X) \rightarrow \neg\, p(X)}{\Gamma \rightarrow \neg\, q(X)}$$

$$R_{43}\ \frac{\left[\Gamma, \neg\, p(X) \rightarrow p(X)\right], \left[\Gamma, \neg\, q(X) \rightarrow q(X)\right]}{\Gamma \rightarrow \mathit{falso}}$$

Una demostración de la inconsistencia del conjunto *S* se puede ver a continuación

$$\dfrac{\dfrac{\dfrac{\neg\, p(X), \neg\, q(a), p(a) \rightarrow \neg\, p(a)}{\neg\, p(X), \neg\, q(a) \rightarrow q(a)}}{\neg\, p(X) \rightarrow p(X)} \qquad \dfrac{\dfrac{\dfrac{\neg\, q(X), p(X), q(X) \rightarrow \neg\, q(X)}{\neg\, q(X), p(X) \rightarrow \neg\, p(X)}}{\neg\, q(X) \rightarrow q(X)}}{\mathit{falso}}$$

En la demostración se puede observar cómo se descomponen las metas en submetas y que a las dos secuencias de la parte superior se les aplica el axioma de asunción que es equivalente a una operación de reducción, en terminología del sistema MESON.

3.3.2. REFINAMIENTO POSITIVO DEL SISTEMA MESON BASADO EN SECUENCIAS

Plaisted propone en [Plaisted, 1990] un refinamiento del sistema MESON que permite una reducción del número de ancestros y que mantiene la completitud del sistema. En la terminología de secuencias, este refinamiento consiste en restringir los literales que se incorporan al antecedente de la secuencia cuando se aplica un regla proveniente de una cláusula

de la teoría a los literales positivos. Esto permite reducir el coste asociado al almacenamiento de los ancestros a lo largo de las demostraciones incrementando, en teoría, la eficiencia del sistema. La desventaja asociada a este refinamiento es que la reducción del conjunto de ancestros almacenados en las demostraciones puede implicar que algunas reducciones no se lleven a cabo y que por tanto sea necesario realizar un número mayor de extensiones para lograr demostrar el problema.

El refinamiento positivo que ya fue mencionado en el apartado de la presentación del demostrador PTTP, se puede reformular fácilmente para el procedimiento de Eliminación de Modelos. En esencia consistiría en únicamente aplicar operaciones de reducción cuando el B-literal más a la derecha de la cadena sea negativo, realizándose por tanto esta operación sobre A-literales positivos. Así sólo sería necesario incorporar A-literales positivos a las cadenas. A continuación se presenta la definición del refinamiento positivo del sistema MESON proposicional basado en secuencias.

Definición 3.4 (El refinamiento positivo del sistema MESON proposicional basado en secuencias) [Plaisted, 1990]

Sea *S* un conjunto de cláusulas en formato Prolog. El sistema deductivo para *S* consta de las siguientes reglas:

1. Sea *C* una cláusula de *S* de la forma L:- L_1, L_2, …, L_n. Entonces para cada una de las contrapositivas de *C* existe una regla en el sistema deductivo de la forma siguiente:

$$\frac{\left[\Gamma_1, \neg L_1 \rightarrow L_1\right], \left[\Gamma_1, \neg L_2 \rightarrow L_2\right], \ldots, \left[\Gamma_n, \neg L_n \rightarrow L_n\right]}{\Gamma \rightarrow L}$$

   donde Γ_i es Γ si L_i es un literal positivo o (Γ, A_i) si L_i es un literal negativo de la forma $\neg A_i$. Al igual que en la Definición 3.3 existirán n+1 reglas para las otras tantas contrapositivas de *C*. Las cláusulas unitarias se formalizan de forma equivalente a la mencionada definición.

2. Para la cláusula inicial o soporte elegida, que en el formato usado se representa en Prolog como *falso*:- L_1, L_2, …, L_n, se incluye la regla siguiente, además de las n reglas correspondientes a las contrapositivas:

$$\frac{\left[\Gamma_1, \neg L_1 \rightarrow L_1\right], \left[\Gamma_1, \neg L_2 \rightarrow L_2\right], \ldots, \left[\Gamma_n, \neg L_n \rightarrow L_n\right]}{\Gamma \rightarrow \mathit{falso}}$$

   donde al igual que en el punto anterior Γ_i es Γ si L_i es un literal positivo o (Γ, A_i) si L_i es un literal negativo de la forma $\neg A_i$.

3. Además el sistema deductivo contiene el denominado *axioma de asunción*, que se formaliza como sigue:

$$\Gamma \rightarrow L, \text{ si } L \in \Gamma$$

De forma similar a como se presenta en la Definición 3.3 se puede realizar el levantamiento a primer orden, obteniendo así el refinamiento positivo del sistema MESON para el caso general.

A continuación se presenta un ejemplo para ilustrar el funcionamiento del refinamiento positivo del sistema MESON basado en secuencias.

Ejemplo 3.11 (Refinamiento positivo del sistema MESON basado en secuencias)

Sea S el mismo conjunto de cláusulas que en Ejemplo 3.9. Realizando las transformaciones adecuadas, el sistema deductivo correspondiente al refinamiento positivo para este conjunto de cláusulas es el siguiente:

$$R_{11}\ \frac{[\Gamma \rightarrow p], [\Gamma \rightarrow q]}{\Gamma \rightarrow \mathit{falso}} \qquad R_{12}\ \frac{\Gamma \rightarrow q}{\Gamma \rightarrow \neg p}$$

$$R_{13}\ \frac{\Gamma \rightarrow p}{\Gamma \rightarrow \neg q}$$

$$R_{21}\ \frac{\Gamma \rightarrow r}{\Gamma \rightarrow p} \qquad R_{22}\ \frac{\Gamma, p \rightarrow \neg p}{\Gamma \rightarrow \neg r}$$

$$R_{31}\ \Gamma \rightarrow r$$

$$R_{41}\ \frac{\Gamma, m \rightarrow \neg m}{\Gamma \rightarrow q} \qquad R_{42}\ \frac{\Gamma, q \rightarrow \neg q}{\Gamma \rightarrow m}$$

$$R_{51}\ \frac{\Gamma \rightarrow m}{\Gamma \rightarrow q} \qquad R_{52}\ \frac{\Gamma, q \rightarrow \neg q}{\Gamma \rightarrow \neg m}$$

A continuación se muestra la secuencia de aplicaciones de las reglas que permiten demostrar que el conjunto S es inconsistente:

$$\dfrac{\dfrac{r}{p} \qquad \dfrac{\dfrac{\dfrac{\dfrac{q \rightarrow r}{q}}{q \rightarrow p}}{q \rightarrow \neg q}}{m}}{q}$$
$$\mathit{falso}$$

Como se puede observar en la demostración, no se ha realizado ninguna reducción y, comparándola con la del Ejemplo 3.9, el número de literales almacenados en los antecedentes de las secuencias es mucho menor. Por otra parte la longitud de la demostración ha crecido, exactamente se realizan dos pasos más, por lo que en el árbol de estados el espacio de búsqueda asociado crecería de forma considerable. Plaisted en sus trabajos no hace referencia a estos dos aspectos contrapuestos en lo que se refiere a la eficiencia del sistema. Tampoco da presenta una argumentación clara, ni teórica ni práctica, a favor del uso del refinamiento positivo. En opinión del autor de la presente tesis este refinamiento se puede considerar aceptable si el número de operaciones de resolución de ancestro que se pierden al aplicarlo no es excesivo o si no provocan un aumento considerable de las demostraciones. En el siguiente capítulo se harán reflexiones más profundas y técnicas sobre este asunto.

En [Plaisted, 1990] se encuentra enunciados y demostrados los siguientes teoremas que establecen la corrección y completitud del sistema MESON basado en secuencias y del refinamiento positivo de este sistema.

Teorema 3.1 (Corrección y completitud del sistema MESON basado en secuencias) [Plaisted, 1990]

Sea *S* un conjunto de cláusulas,

a) Si $\vdash_S$ *falso*, entonces *S* es inconsistente.
b) Si *S* inconsistente, entonces existe una cláusula soporte en *S* tal que $\vdash_S$ *falso*.

Teorema 3.2 (Corrección y completitud del refinamiento positivo) [Plaisted, 1990]

Sea *S* un conjunto de cláusulas,

a) Si existe una demostración de *falso* usando el refinamiento positivo del sistema MESON basado en secuencias, entonces *S* es inconsistente.
b) Si *S* inconsistente, entonces existe una cláusula soporte en *S* tal que a partir de ella se puede demostrar *falso* usando el refinamiento positivo del sistema MESON basado en secuencias.

Como ya se mencionó anteriormente es posible generalizar el refinamiento positivo de forma que en lugar de considerar los literales seleccionados positivos como los únicos que provocan la incorporación de un nuevo ancestro, se consideren los literales seleccionados que cumplan una determinada propiedad. Por ejemplo se podrían utilizar, de forma simétrica, los literales negativos, o aquellos literales ciertos en una cierta interpretación, o aquellos literales que pertenecen a un cierto conjunto predefinido, etc. Sin embargo, de la completitud de estos refinamientos, su utilidad, las propiedades de los sistemas a que dan lugar, así como de comparaciones de eficiencia entre ellos, no se indica nada en los trabajos de D. Plaisted

[Plaisted, 1988] [Plaisted, 1990]. Los refinamientos de este tipo, sobre el conjunto de ancestros a considerar, presentan dos características fundamentales:

1) El coste asociado a realizar las operaciones de resolución de ancestro es menor, al reducirse el tamaño del conjunto de ancestros almacenado a lo largo de las demostraciones. Adicionalmente, esto suele conllevar un aumento de las operaciones de resolución de entrada, con el consiguiente incremento de la longitud de las demostraciones y, por tanto, del espacio de búsqueda. Debido a esta dicotomía, Stickel menciona en [Stickel, 1988] que experimentalmente las ventajas de este tipo de refinamientos no están claras y que deberían de ser probadas suficientemente.

2) Estos refinamientos tienen la posibilidad de adaptarse al tipo de problema a tratar. Por ejemplo, el refinamiento positivo es más adecuado para el tratamiento de problemas que sean Horn o casi Horn. En el primer caso el refinamiento positivo es equivalente a resolución de entrada (o el procedimiento SLD) al no almacenar ningún ancestro. En el segundo caso, el número de ancestros almacenados será mínimo y por tanto éstos no serán una carga para la eficiencia del sistema si se tiene en cuenta que la posibilidad de realizar una resolución de ancestro en este tipo de problemas es pequeña. Esta similitud de funcionamiento con resolución de entrada o el procedimiento SLD hace que el refinamiento positivo sea especialmente atractivo.

3.3.3. IMPLEMENTACIÓN DEL SISTEMA MESON BASADO EN SECUENCIAS

La propuesta realizada en [Plaisted, 1990] consiste en utilizar el Prolog como lenguaje soporte y realizar un representación directa de las reglas deductivas de manera directa. De manera que es el Prolog el encargado de realizar tanto la aplicación de dichas reglas como la búsqueda en el espacio de búsqueda. Plaisted no entra demasiado en detalles en los aspectos de implementación, que por otro lado sería necesario revisar, como unificación, recorrido y construcción del espacio de búsqueda, podas, etc. A continuación se presenta la regla R_{11} del Ejemplo 3.9 como una sentencia Prolog:

secuen(Γ, *falso*) :- *secuen*([¬*p*(*X*)| Γ], *p*(*X*)), *secuen*([¬*q*(*X*)| Γ],*q*(*X*)).

Adicionalmente a las sentencias que representan a las reglas deductivas provenientes de las cláusulas iniciales, existirá otra sentencia que representará el axioma de asunción. Este sentencia y las correspondientes a las cláusulas unitarias permitirán probar una cierta secuencia.

3.3.4. CONCLUSIONES AL SISTEMA MESON BASADO EN SECUENCIAS

Aunque la reformulación de este sistema mediante el uso de secuencias es interesante y puede servir para verter más luz sobre él, la aportación más significativa de D. Plaisted consiste en presentar un refinamiento positivo del sistema MESON. Dicho refinamiento positivo utiliza conjuntos de ancestros en las demostraciones más reducidos y sigue siendo completo para teorías de primer orden. También es muy interesante, aunque no está suficientemente destacado en sus trabajos, la proximidad de funcionamiento del refinamiento positivo con respecto a resolución de entrada o el procedimiento SLD cuando se trabajan con teorías Horn. Por otra parte, como aspectos negativos, hay que destacar la poca profundización que realiza sobre los aspectos de eficiencia e implementación, así como la falta de mención de la posibilidad de utilizar otros refinamientos diferentes al positivo.

3.4. EL DEMOSTRADOR SATCHMO

El demostrador SATCHMO, acrónimo de "SATisfiability CHecking by MOdel generation", es un demostrador de teoremas para teorías clausales propuesto por R. Manthey y F. Bry cuyo funcionamiento está basado en la construcción de un modelo para la teoría investigada. Esta idea también se encuentra en la base del procedimiento de Eliminación de Modelos, aunque con un enfoque bastante distinto. SATCHMO utiliza una estrategia de saturación de nivel para la construcción del posible modelo y su método de demostración es una combinación de razonamiento hacia detrás y hacia delante que permite obtener en algunas ocasiones unos resultados temporales excelentes, especialmente para un tipo de teorías denominadas *teorías de rango restringido*. Hay que destacar que SATCHMO no es un refinamiento de resolución lineal, y que durante las demostraciones se almacena más información que la correspondiente a los ancestros directos de la meta en curso. Una discusión informal de las características del demostrador junto con algunos resultados experimentales obtenidos se pueden encontrar en [Manthey y Bry, 1988]. También en esta sección se va a introducir la última variante de este demostrador denominada SATCHMORE que ha sido presentada en [Loveland, Reed y Wilson, 1995] y en la cual se corrigen algunos de los principales defectos del SATCHMO.

Una de las principales críticas que se le han achacado al demostrador SATCHMO por varios investigadores ha sido la falta de formalización teórica del mismo. El demostrador es presentado por sus creadores como una colección de instrucciones Prolog que son aplicables a teorías clausales expresadas de una forma determinada. El hecho de que no exista una definición formal del demostrador, como se ha comentado anteriormente, es una carencia , que el autor de la presente tesis considera grave. A pesar de ello, la metodología e ideas que guían la

construcción de este demostrador contienen aspectos que han sido destacados por mucho investigadores como interesantes.

Existen tres versiones implementadas del demostrador SATCHMO. Las tres versiones sólo son aplicables a teorías de rango restringido. Esta limitación es superada por los autores proponiendo una transformación de teorías que no son de rango restringido a otra que sí lo son que preserva la consistencia. La primera versión es la original y la más sencilla que permite comprender fácilmente el funcionamiento básico del demostrador. La forma de razonamiento de la primera versión es hacia delante para ir construyendo las diferentes interpretaciones de la teoría inicial en busca de un modelo. La segunda versión consigue aplicar un razonamiento mixto hacia delante y atrás, que aprovecha las características del Prolog y que incrementa la eficiencia de forma notable. La manera de conseguir este razonamiento mixto es realizando una transformación de las cláusulas iniciales al formato SATCHMO distinta en la que las cláusulas Horn son transformadas a formato Prolog. Estas dos versiones son incompletas debido a la presencia de cierto tipo de recursividad y la forma en la cual se realiza la construcción de la interpretación. La tercera versión salva esta dificultad utilizando un mecanismo de saturación de nivel que va generando la interpretación de forma iterativa por niveles. Estas tres versiones serán discutidas a continuación en la exposición siguiente.

Primero se presentará el formato de entrada para el demostrador SATCHMO. Este formato es implicativo y en él se pueden observar características de metaprogramación. En esencia la idea consiste en declarar un operador que corresponda con el razonamiento hacia delante y que puede ser equivalente al operador consecuencia lógica de la Programación Lógica [Lloyd, 1987].

Definición 3.5 (Formato de entrada del SATCHMO) [Manthey y Bry, 1988]

Sea S un conjunto de cláusulas. El formato de entrada del SATCHMO para dicho conjunto se obtiene aplicando a cada cláusula C perteneciente a S las siguientes reglas:

a) si C es una *cláusulas negativas*, de la forma $\neg L_1 \vee \neg L_2 \vee \ldots \vee \neg L_n$, entonces se transforma en al siguiente sentencia:

$$L_1, L_2, \ \ldots, L_n \Rightarrow \mathit{false}$$

b) si C es una *cláusula positiva*, de la forma $L_1 \vee L_2 \vee \ldots \vee L_n$, entonces se transforma en la siguiente sentencia:

$$\mathit{true} \Rightarrow L_1 \ ; L_2 \ ; \ \ldots \ ; L_n$$

c) si C es Horn y no es ni positiva ni negativa, de la forma $L_1 \vee \neg L_2 \vee \ldots \vee \neg L_n$ donde $n>1$, o sea tiene exactamente un literal positivo y al menos un literal negativo, entonces se representa mediante el formato Prolog:

$$L_2, \ldots, L_n \Rightarrow L_1$$

d) si C no es Horn y no es positiva ni negativa, esto es C tiene más de un literal positivo y al menos uno negativo, de la forma $L_1 \vee L_2 \vee \ldots \vee L_n \vee L'_1 \vee L'_2 \ldots \vee L'_m$ donde n>1 y m≥1, entonces se transforma en la siguiente sentencia:

$$L'_1, L'_2, L'_m \Rightarrow L_1 ; L_2 ; \ldots ; L_n$$

Como se puede ver en estas transformaciones el propósito es transformar el conjunto de cláusulas de manera que se diferencien las cláusulas Horn de las que no lo son. Claramente, ya que el demostrador SATCHMO se implementa en Prolog, las cláusulas Horn se representan directamente en formato Prolog mientras que las que no lo son se representan en un formato distinto, más concretamente se utiliza la metaprogramación incluyendo dos metapredicados "⇒" y ";".

A continuación se introduce propiedad de *rango restringido* ya que una de las versiones del demostrador está limitada al uso de teorías clausales de rango restringido.

Definición 3.6 (Cláusula de rango restringido) [Manthey y Bry, 1988]

Una cláusula C es de rango restringido si y sólo si toda variable de la cláusula ocurre por lo menos en un literal negativo de la cláusula. Un conjunto de cláusulas S es de rango restringido si y sólo si cada una de las cláusulas de S es de rango restringido.

La propiedad de rango restringido es muy habitual en el campo de las Bases de Datos Deductivas [Decker, 1988] [Demolombe, 1994]. De forma intuitiva la propiedad de rango restringido asegura que el conocimiento que se puede deducir de la base de datos siempre va asociado a algún individuo que aparece en la base de datos. Formalmente se ha demostrado que las respuestas lógicas y computadas [Lloyd, 1987] de teorías de rango restringido son siempre base, esto es todo elemento de la respuesta instancia una variable por un término base sin variables [Shepherdson, 1991] [Decker, 1991]. A continuación se presenta la versión del demostrador SATCHMO para teorías de rango restringido.

3.4.1. EL DEMOSTRADOR SATCHMO PARA TEORÍAS DE RANGO RESTRINGIDO

Como ya se mencionó anteriormente, la idea fundamental en la que se basa el demostrador es probar la consistencia de un conjunto de cláusulas construyendo un modelo de éste. Si la construcción de este modelo falla entonces se demuestra la inconsistencia del conjunto de

cláusulas. Cualquier modelo de un conjunto de cláusulas puede ser representado por un conjunto *M* de átomos base de forma que los átomos que pertenecen al modelo son ciertos en él y los que no pertenecen son falsos. Así se puede comprobar que dada una cláusula en formato implicativo $A \leftarrow B$, ésta es cierta en *M* si y sólo si para cada sustitución σ tal que $B\sigma$ es cierto en *M* entonces $A\sigma$ es también cierto en *M*. Esto, que es general par cualquier conjunto de cláusulas, en el caso de un conjunto de cláusulas de rango restringido además implica que $A\sigma$ y $B\sigma$ son base. El mecanismo en el que se basa el demostrador es en la construcción iterada de este modelo. El tipo de razonamiento que aplica el demostrador es hacia delante, similar al operador consecuencia lógica de la Programación Lógica [Lloyd, 1987], pero en el cual se tiene en cuenta la presencia de disyunción en las cabezas de algunas cláusulas. A continuación se presenta una definición formal del SATCHMO.

Definición 3.7 (El demostrador SATCHMO con razonamiento hacia delante)

Sea *S* un conjunto de cláusulas de rango restringido que se encuentran en el formato mencionado en la Definición 3.5, el método para construir un modelo *M* de *S* es el siguiente:

1) $M = \{true\}$, esto es *M* está inicialmente vacío.

2) Para cada cláusula *C* de *S*, si *C* es falsa en *M*, entonces aplicar el punto 3) o 4). Si no existe una cláusula *C* de *S* que sea falsa en *M* entonces parar; *M* es un modelo de *S*. Obsérvese que para el conjunto inicial, {*true*}, las cláusulas que son falsas son las cláusulas del tipo b), esto es, las cláusulas positivas.

3) Para cada cláusula proveniente del punto 2) que no sea Horn de la forma $L'_1, L'_2, L'_m \Rightarrow L_1 ; L_2 ; \ldots ; L_n$ y para cada sustitución σ tal que $(L'_2 \wedge \ldots \wedge L'_m)\sigma$ elegir un literal L_i de forma indeterminista, añadir $L_i\sigma$ a *M* y establecer un *punto de elección.*

4) Si *false* pertenece a *M* quiere decir que una instancia de alguna de las cláusulas negativas no es cierta en *M*, por tanto *M* o cualquier supraconjunto de *M* no pueden ser modelo de *S*, entonces realizar si es posible una vuelta atrás al último punto de elección de 3) (que puede pertenecer a la iteración en curso o a un iteración anterior) y seleccionar un literal L_j distinto al seleccionado anteriormente. Si esto no es posible entonces *parar* ya que el conjunto *S* es inconsistente.

5) Volver al punto 2).

Si el conjunto *S* es inconsistente el demostrador termina en el punto 4). Si por el contrario es consistente el demostrador puede acabar en el punto 2) o no hacerlo. Analizando con detalle el punto 4) se puede entender que el demostrador utiliza las disyunciones de las cláusulas de la teoría para ir generando modelos alternativos de manera sucesiva. Si la elección efectuada al elegir un cierto literal de la disyunción no conduce a un modelo, el demostrador probará utilizar

otro literal para generar un modelo alternativo. Es interesante destacar que la propiedad de rango restringido asegura en el punto 3) que las sustituciones serán base y por tanto los literales añadidos a *M* también lo serán.

Hay que indicar que la definición presentada anteriormente ha sido deducida de la presentación informal del demostrador y de la líneas de programa Prolog que lo implementa y que se pueden encontrar en [Manthey y Bry, 1988]. Con esto el autor quiere expresar que coincide con ciertos investigadores que han achacado a este demostrador la falta de rigor y formalización.

A continuación se puede observar el programa Prolog incluido en [Manthey y Bry, 1988] que estos autores presentan como formalización del SATCHMO.

```
consistente:- se_viola(C), !,
              satisfacer(C),
              consistente.
consistente.

satisfacer(C):- componente(X, C),
                assert(X),
                vuelta_atrás(retract(X)),
                not false.

se_viola(C):- (A ⇒ C),
              A, not C.

componente(X, (Y; Z)):- !,(X=Y ; componente(X, Z)).
componente(X, X).

vuelta_atrás(X).
vuelta_atrás(X):- X, !, fail.
```

El siguiente ejemplo sirve para ilustrar el funcionamiento del demostrador presentado.

Ejemplo 3.12 (Funcionamiento del demostrador SATCHMO para rango restringido)

Sea *S* el siguiente conjunto inconsistente de cláusulas de rango restringido expresado en el formado de la Definición 3.5:

C_1. $p(X), q(X) \Rightarrow false$ C_3. $p(X) \Rightarrow q(X)$

C_2. $q(X) \Rightarrow p(X)$ C_4. $true \Rightarrow p(a); q(b)$

Siguiendo el método de la Definición 3.7 se construye iteradamente el conjunto *M* de átomos siguiente:

$M = \{true\}$	
$M = \{p(a)\}$	Punto de elección en la cláusula S_4, eligiendo $p(a)$
$M = \{p(a), q(a)\}$	
$M = \{p(a), q(a), false\}$	El conjunto M, ni ningún supraconjunto suyo, puede ser modelo de *S*. Vuelta atrás al último punto de elección

$M = \{q(b)\}$	Punto de elección en la cláusula C_4, eligiendo $q(b)$
$M = \{q(b), p(b)\}$	
$M = \{q(b), p(b), false\}$	El conjunto M, ni ningún supraconjunto suyo, puede ser modelo de S. No existe otra alternativa. Parar. El conjunto inicial es inconsistente.

Los autores de este trabajo proponen una representación muy sencilla y visual del funcionamiento del demostrador y las tentativas que éste realiza en la búsqueda de un modelo del problema a tratar. Esta representación consiste en un árbol en el cual cada nodo corresponde a un paso de razonamiento hacia delante con la consiguiente introducción de un átomo que se ha introducido en la interpretación parcial. Los nodos del árbol son etiquetados con los átomos introducidos. La figura siguiente corresponde al árbol de aserciones en la interpretación parcial siguiente:

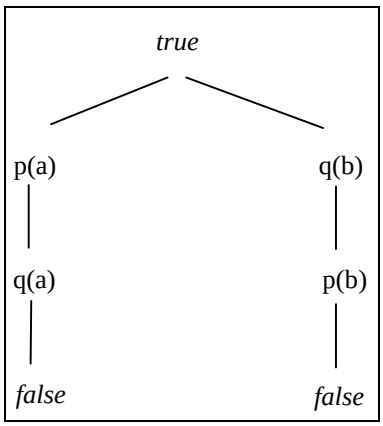


Figura 3.6: Árbol de la construcción de la interpretación parcial del Ejemplo 3.12

Como se puede observar en la figura, el nodo *true* que tiene dos hijos corresponde al punto de elección que proviene de la cláusula disyuntiva de la teoría. La selección de cada uno de ellos da lugar a diferentes interpretaciones. La interpretación parcial corresponde al conjunto de átomos que se encuentran en una rama del árbol desde la raíz hasta un cierto nodo. Si una rama termina en el nodo con *false* quiere decir que la búsqueda del modelo siguiendo esa deducción falla, ya que alguna cláusula es falsa en la interpretación parcial correspondiente. Si una rama del árbol termina en un átomo distinto de *false*, la interpretación parcial de esa rama es un modelo de la teoría. Si todas las rama terminan en *false*, entonces la teoría es inconsistente.

El demostrador SATCHMO propuesto en [Manthey y Bry, 1988] es un programa Prolog basado en la Definición 3.7 anteriormente presentada con las siguientes características:

a) La estrategia de generación del modelo puede ser o bien en profundidad, con lo que se pierde la completitud del método, o bien mediante saturación de nivel, siguiendo una implementación en Prolog literal al método descrito anteriormente.

b) Para implementar el conjunto *M*, que representa la interpretación que se está intentando convertir en modelo, el SATCHMO utiliza la propia base de datos Prolog mediante la instrucciones *assert* para introducir nuevos elementos en *M* y que posteriormente, si no conducen a la construcción del modelo deseado, son eliminados mediante un hábil uso de la vuelta atrás del Prolog y la operación *retract*.

A continuación se va a presentar la primera variante que consiste esencialmente en combinar el razonamiento adelante del demostrador SATCHMO ya presentado y que permite ir construyendo de forma dinámica el modelo con el razonamiento hacia atrás. La forma de introducir el razonamiento hacia atrás es la siguiente: las cláusulas Horn en vez de transformadas al formato SATCHMO se dejan en formato Prolog y la comprobación de la validez de la interpretación en curso se realiza mediante la búsqueda de una refutación en Prolog. La definición siguiente formaliza esta versión del demostrador SATCHMO.

Definición 3.8 (El demostrador SATCHMO con razonamiento hacia atrás)

Sea *S* un conjunto de cláusulas de rango restringido de forma que las cláusulas disyuntivas se encuentran en el formato mencionado en la Definición 3.5 y las cláusulas negativas de la forma $\neg L_1 \vee \neg L_2 \vee \ldots \vee \neg L_n$ se transforman al formato Prolog, obteniendo la sentencia *false*:- $L_1, L_2, \ldots, L_n$. El método para construir un modelo *M* de *S* es el siguiente:

0) Comprobar que el conjunto de cláusulas Horn es consistente. Esto se consigue lanzando una computación desde :-*false* y comprobando que no exista una refutación.

1) $M = \{true\}$, esto es, *M* está inicialmente vacío.

2) Para cada cláusula *C* de *S* que no está en formato Prolog, si *C* es falsa en *M*, entonces aplicar el punto 3) o 4). Si no existe una cláusula *C* de *S* que sea falsa en *M* entonces parar; *M* es un modelo de *S*. Obsérvese que para el conjunto inicial, {*true*}, las cláusulas que son falsas son las cláusulas del tipo b), esto es las cláusulas positivas disyuntivas.

3) Para cada cláusula proveniente del punto 2) de la forma $L'_1, L'_2, L'_m \Rightarrow L_1 ; L_2 ;\ldots; L_n$ y para cada sustitución σ tal que $(L'_2 \wedge \ldots \wedge L'_m)\sigma$ elegir un literal L_i de forma indeterminista, añadir $L_i\sigma$ a *M* y establecer un *punto de elección.*

4) Comprobar que la interpretación *M* no satisface alguna cláusula negativa, y por tanto que no puede ser modelo de la teoría. Esto se realiza comprobando la existencia de una refutación para :-*false*. Si existe una refutación para :-*false*, entonces realizar si es posible una vuelta atrás al último punto de elección de 3) (que puede pertenecer a la iteración en curso o a un iteración anterior) y seleccionar un literal L_j distinto al

seleccionado anteriormente. Si esto no es posible entonces *parar* ya que el conjunto *S* es inconsistente.

5) Volver al punto 2).

Como se puede observar en la definición de esta versión del demostrador SATCHMO, la principal modificación reside en el hecho de utilizar el Prolog para probar la información derivable de la cláusulas Horn de la teoría inicial. Desde el punto de vista de la construcción del modelo, se puede entender que esta versión utiliza el Prolog y las cláusulas Horn para probar que un cierto átomo pertenece al modelo mínimo de estas cláusulas, y sigue usando el mecanismo del demostrador SATCHMO para la búsqueda de los átomos que, no estando en el modelo mínimo de la parte Horn de la teoría, forman parte de un cierto modelo de ésta. El principal problema de esta versión reside, como los mismos autores destacan, en los problemas intrínsecos del Prolog. Es bien conocido que la mayoría de los sistemas basados en el Prolog no incorporan una unificación con el test de ocurrencia y que realizan el recorrido del árbol de estados en profundidad. Por estos dos motivos estos sistemas no son completos para la demostración de problemas Horn y por tanto tampoco lo es esta versión del demostrador SATCHMO. El ejemplo siguiente muestra este hecho.

Ejemplo 3.13 (Incompletitud del demostrador SATCHMO con razonamiento hacia atrás)
Sea *S* el siguiente conjunto inconsistente de cláusulas:

$C_1 = \leftarrow p(x)$
$C_2 = p(x) \leftarrow p(x)$
$C_3 = p(a)$

La transformación de este conjunto de cláusulas a sentencias en formato de entrada para el SATCHMO con razonamiento hacia atrás según la Definición 3.8 da lugar a:

$S_1 =$ *false*:- $p(X)$
$S_2 = p(X)$:- $p(X)$
$S_3 = p(a)$

Claramente en el paso 0) de la Definición 3.8, en el que se comprueba si el conjunto de cláusulas Horn de la teoría es inconsistente, el demostrador puede que no termine ya que los sistemas basados en Prolog no son completos. En particular en este ejemplo el problema reside en la búsqueda en profundidad que realizan estos sistemas.

Evidentemente, existen soluciones para los problemas que presenta el demostrador SATCHMO con razonamiento hacia atrás. Su incompletitud se resolvería si en vez de utilizar un sistema basado en Prolog para realizar el razonamiento hacia atrás se usara algún demostrador

que fuera completo (esencialmente que resuelva los problemas de la unificación y de la búsqueda en profundidad).

Como se ha mencionado anteriormente las dos versiones presentadas del demostrador SATCHMO sólo funcionan adecuadamente cuando la teoría que estudian es de rango restringido. El siguiente ejemplo muestra la incorrección del demostrador para teorías de rango restringido cuando se enfrenta a una que no lo es.

Ejemplo 3.14 (Incorrección del demostrador SATCHMO para rango restringido)

Sea S el siguiente conjunto consistente de cláusulas:

$$C_1 = \leftarrow p(a)$$
$$C_2 = \leftarrow q(b)$$
$$C_3 = p(x) \vee\ q(x) \leftarrow$$

La transformación al formato de entrada para el demostrador SATCHMO según la Definición 3.5 da lugar a las siguientes sentencias:

$$S_1 = p(a) \Rightarrow false$$
$$S_2 = q(b) \Rightarrow false$$
$$S_3 = true \Rightarrow p(X); q(X)$$

El problema viene motivado porque la sentencia S_3 no es de rango restringido. Si se aplica el método de la Definición 3.7 el problema aparece con la manera de tratar cualquiera de los dos literales de la cabeza de esta cláusula. Al no ser ninguno de ellos base no está claro la forma de introducirlos en el conjunto que representa la interpretación parcial. Si se introduce el literal con variables, queriendo indicar que se incorpora cualquier instancia del mismo, el demostrador responde que la teoría es inconsistente cuando no lo es.

La solución que los dos autores proponen en [Manthey y Bry, 1988] es transformar la teoría inicial que no es de rango restringido a una que sí lo sea y que, desde el punto de vista de la consistencia, sea equivalente. A continuación se presenta la versión de SATCHMO para teorías que no son de rango restringido.

3.4.2. EL DEMOSTRADOR SATCHMO PARA TEORÍAS QUE NO SON DE RANGO RESTRINGIDO

Como se ha mencionado anteriormente la idea que esencial para poder trabajar con teorías que no son de rango restringido es transformar estas teorías en otras que sí lo sean de forma que las dos sean equivalentes desde el punto de vista de la consistencia. La transformación se basa en el método que propusieron Davis y Putman para comprobar la inconsistencia de un conjunto

de cláusulas. Brevemente este método consistía en ir obteniendo las instancias bases de las cláusulas usando los elementos del universo de Herbrand de la teoría y con ellas intentar comprobar la consistencia. La propuesta de Manthey y Bry no es tan ineficiente como la de Davis y Putman ya que ellos proponen obtener de aquellas cláusulas que no sean de rango restringido, instancias que sí lo sean. Esta transformación se define a continuación.

Definición 3.9 (Transformación de una teoría que no es de rango restringido en una que sea equivalente desde el punto de vista de la consistencia) [Manthey y Bry, 1988])

Sea *S* un conjunto de cláusulas en formato implicativo que no es de rango restringido. teoría. La teoría *S'* obtenida a partir de *S* aplicando las siguientes reglas es de rango restringido:

1) Sea *C* una cláusula de *S*. Si *C* es de rango restringido entonces *C* pertenece a *S'*.

2) Sea *C* una cláusula de *S* que no es de rango restringido. Si *C* es de la forma $A \leftarrow$, donde *A* es una disyunción de átomos que contienen las variables $x_1,\ldots, x_n$, entonces la cláusula $A \leftarrow dom(x_1) \wedge \ldots \wedge \text{dom}(x_n)$ pertenece a *S'*.

3) Sea *C* una cláusula de *S* que no es de rango restringido. Si *C* es de la forma $A \leftarrow B$, donde *A* es una disyunción de átomos y *B* es una conjunción de átomos tal que las variables $x_1,\ldots, x_n$ ocurren en *A* y no en *B*, entonces $A \leftarrow B \wedge dom(x_1) \wedge \ldots \wedge \text{dom}(x_n)$ pertenece a *S'*.

4) Por cada constante *c* que aparece en *S*, la cláusula $\text{dom}(c)\leftarrow$ se incluye en *S'*. Si *S* no contiene ninguna constante la se usa una constante artificial para generar el universo de Herbrand.

5) Por cada símbolo de función n-aria *f* que aparece en *S*, se incluye en *S'* la cláusula $dom(f(x_1), f(x_2),\ldots, f(x_n)) \leftarrow dom(x_1) \wedge dom(x_2) \wedge \ldots \wedge dom(x_n)$

Claramente el predicado *dom* que se añade a cada una de las cláusulas que no son de rango restringido establecen la instanciación de estas cláusulas sobre los términos del universo de Herbrand en tiempo de ejecución. La transformación anterior puede ser comparada con la skolemización, que aunque no preserva la equivalencia lógica y sí que mantiene la equivalencia desde el punto de vista de la consistencia. Así es sencillo comprobar que si se toma un modelo de la teoría transformada *S'* y se le eliminan todos los átomos del predicado *dom* se obtiene un modelo de la teoría inicial *S*. Existe una correspondencia uno a uno entre los modelos de ambas teorías. Por lo que la transformación anterior preserva la consistencia. A continuación se puede observar la aplicación de esta transformación al Ejemplo 3.14.

Ejemplo 3.15 (Transformación a rango restringido)

Sea *S* el conjunto de cláusula del Ejemplo 3.14. Al aplicar la transformación de la Definición 3.9 se obtiene el conjunto de cláusulas *S'* siguiente:

$C_1 = \leftarrow p(a)$ $\quad\quad$ $C_4 = dom(a) \leftarrow$

$C_2 = \leftarrow q(b)$ $\quad\quad$ $C_5 = dom(b) \leftarrow$

$C_3 = p(x) \vee q(x) \leftarrow dom(x)$

Si ahora se aplicará el demostrador SATCHMO a la teoría *S'*, éste fallaría indicando correctamente que ésta es consistente.

Hay que indicar que el demostrador SATCHMO, tanto la versión con razonamiento hacia delante como la versión con razonamiento mixto, que presentan en [Manthey y Bry, 1988] no es completo. El problema reside esencialmente en la forma en la cual estos autores realizan la aplicación del punto 2) de su definición. El metaprograma en Prolog que estos autores proponen como definición del demostrador no realiza la aplicación del punto 2) de forma exhaustiva. Su propuesta elige una cláusula comprueba si no es satisfecha por la interpretación parcial *M* y, en caso de no serlo pasa al punto 3). Así, en algunos casos de recursividad, el demostrador no es capaz de comprobar el paso 2) para cada una de las cláusulas de la teoría de forma exhaustiva. El siguiente ejemplo ilustra esta circunstancia.

Ejemplo 3.16 (Incompletitud del demostrador) [Manthey y Bry, 1988]

Sea *S* el siguiente conjunto de cláusulas:

$C_1 = p(a) \leftarrow$

$C_2 = \leftarrow p(f(x)) \wedge p(g(x))$

$C_3 = \mathrm{p}(f(x)) \leftarrow p(x)$

$C_4 = p(g(x)) \leftarrow p(x)$

Claramente *S* es inconsistente. Sin embargo si la implementación del demostrador SATCHMO de la Definición 3.7 no realiza la aplicación del punto 2) de la definición de forma exhaustiva, por lo que no le es posible llegar a concluir su inconsistencia. A continuación se presenta la secuencia de operaciones que realiza este demostrador:

punto 1), inicialización de M . $M = \{true\}$
punto 2), se determina que $C1$ es falsa en M.
punto 3), se incluye $p(a)$ en M. $M = \{p(a)\}$
punto 4), *false* no está en M. Se va al punto 2)
para tratar otra cláusula.
punto 2), se determina que C_3 es falsa en M.
punto 3), se incluye $p(f(a))$ en M. $M = \{p(f(a)), p(a)\}$
punto 4), *false* no está en M. Se va al punto 2)
para tratar otra cláusula.
punto 2), se determina que C_1 es falsa en M.
punto 3), se incluye $p(f(f(a)))$ en M. $M = \{ p(f(f(a))), p(f(a)), p(a)\}$
punto 4), *false* no está en M. Se va al punto 2)
para tratar otra cláusula.
¼

Como se puede observar, el problema reside en la forma en la cual el demostrador determina las cláusulas de la teoría falsas en M y las trata posteriormente. Se podría decir que el demostrador intenta utilizar la misma cláusula para determinar si es falsa en M y generar nuevos átomos a introducir en M mientras le sea posible. Este comportamiento en presencia de cierto tipo de recursividad da lugar que el demostrador sea incapaz de determinar la inconsistencia. En el ejemplo si no se usa la cláusula C_4 no es posible concluir la inconsistencia de la teoría.

La solución que proponen los autores es la modificación de la forma en la cual se aplica el punto 2) de la definición del demostrador SATCHMO. Esta modificación consiste en realizar una aplicación del punto 2) y 3) una vez a todas las cláusulas de la teoría y determinar así el conjunto de cláusulas a que son falsas en M y el conjunto de átomos a introducir. En palabras de los autores: *"La completitud se puede conseguir si las cláusulas son generadas sistemáticamente nivel por nivel. Primero, todos las cláusulas de la teoría que son falsas en la interpretación dada son determinados sin modificar la interpretación. Entonces todos los átomos necesarios para satisfacer esas cláusulas son introducidos en la interpretación a la vez"*. A continuación se puede observar los pasos de la demostración aplicando la modificación que se acaba de indicar. Hay que destacar que la interpretación parcial corresponde a la unión de las interpretaciones de los diferentes niveles.

punto 1), inicialización de M . $M_0 = \{true\}$ nivel 0
punto 2), se determinan todas las cláusulas que son
falsas en M. Se obtiene C_1.
punto 3), se incluye los átomos en M necesarios para
satisfacer las cláusulas del punto 2) anterior $p(a)$. $M_1 = \{p(a)\}$ nivel 1
punto 4), *false* no está en M. Se va al punto 2).

punto 2), se determinan todas las cláusulas que son
falsas en M. Se obtiene C_3 y C_4.
punto 3), se incluye los átomos en M necesarios para
satisfacer las cláusulas del punto 2) anterior $p(a)$. $M_2 = \{p(f(a), p(g(a))\}$ nivel 2
punto 4), *false* no está en M. Se va al punto 2).
punto 2), se determinan todas las cláusulas que son
falsas en M. Se obtiene C_3 y C_4.
punto 3), se incluye los átomos en M necesarios para
satisfacer las cláusulas del punto 2) anterior $p(a)$. $M_3 = \{p(f(f(a)), p(f(g(a))), p(g(f(a)),$
$p(g(g(a)), false\}$ nivel 3
punto 4), *false* está en M.

En [Manthey y Bry, 1988] se afirma la completitud del SATCHMO para la comprobación de la inconsistencia para teorías clausales transformadas adecuadamente si no son de rango restringido, si se realiza la aplicación de la Definición 3.7 según el método presentado anteriormente. A continuación se incluye el programa Prolog que se presenta en [Manthey y Bry, 1988] como formalización del demostrador SATCHMO este método.

```
nivel_consistente:- nivel_consistente(1).

nivel_consistente(N):-
        se_viola_nivel(N), !,
        vuelta_atrás(limpiar_nivel(N)),
        satisfacer_nivel(N),
        N1 is N + 1,
        nivel_consistente(N1).
nivel_consistente(N).

satisfacer_nivel(N):-
        generada(N,C),
        not C,!,
        satisfacer(C),
        satisfacer_nivel(N).
satisfacer_nivel(N).

se_viola_nivel(N):-
        se_viola(C),
        not generada(N,C),
        assert(generada(L,C)),
        fail.
se_viola_nivel(L):-
        generada(L,X).

limpiar_nivel(N):-
        retract(generada(L,X)),
        fail.
limpiar_nivel(L).
```

Para terminar la presentación del demostrador SATCHMO se va a hacer mención a un trabajo reciente que introduce mejoras substanciales [Loveland, Reed y Wilson, 1995]. En este trabajo estos autores proponen una nueva versión, denominada SATCHMORE (acrónimo de *SATCHMO with RElevancy*), que solventa los problemas que tiene el SATCHMO con razonamiento mixto. Si se supone que el SATCHMO con razonamiento mixto utiliza un Prolog

adecuado para el tratamiento de las cláusulas Horn de la teoría (esto implica unificación correcta y terminación), uno de los principales problemas del demostrador es la forma en la cual realiza el razonamiento hacia delante para la construcción de la interpretación parcial haciendo uso de las cláusulas disyuntivas. Este problema consiste esencialmente en que SATCHMO emplea cada una de las cláusulas disyuntivas que se encuentran en la teoría para ir construyendo la interpretación parcial y comprobando que ésta no hace falsa la subteoría Horn, incluso si alguna de estas cláusula no es relevante. El siguiente ejemplo muestra que la existencia de cláusulas disyuntivas que no son relevantes a la subteoría Horn provoca la explosión de la demostración del SATCHMO.

Ejemplo 3.17 (Tratamiento problemático de las cláusulas irrelevantes a la teoría Horn en el SATCHMO)

Sea S un conjunto formado por las siguientes cláusulas en formato SATCHMO con razonamiento mixto:

$C_1 = false$:- p $\quad\quad$ $C_5 = r \Rightarrow a; b$

$C_2 = false$:- q $\quad\quad$ $C_6 = r \Rightarrow p\ ;\ c\ ;\ d$

$C_3 = c$:- d $\quad\quad$ $C_7 = r \Rightarrow p\ ;\ q$

$C_4 = r$

La construcción de la interpretación parcial se puede representar como un árbol en el cual cada nodo corresponde a una aserción. La interpretación parcial corresponde a los átomos de una rama del árbol desde su raíz. Toda rama del árbol termina en *false*, indicando que alguna cláusula de la teoría subHorn es falsa en la interpretación parcial, indicando que la interpretación parcial formada por los átomo de esa rama no es un modelo.

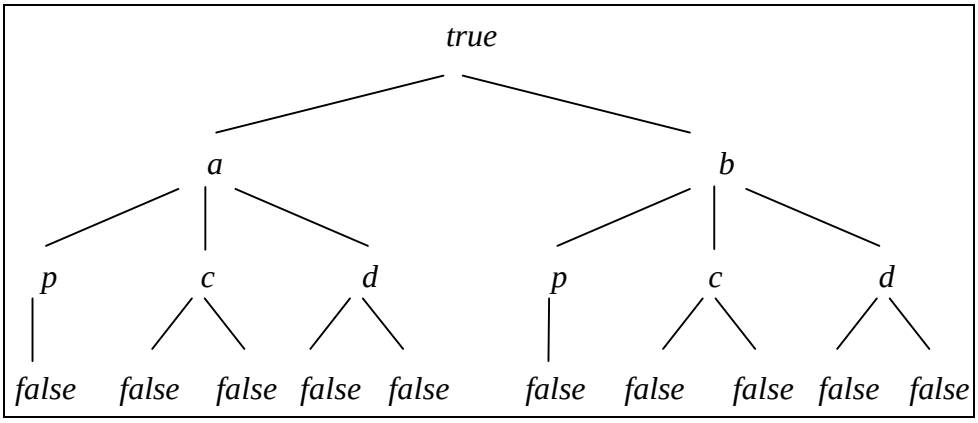


Figura 3.7: Árbol de la construcción de la interpretación parcial del Ejemplo 3.17.

Como se puede observar en la figura, SATCHMO elige la primera cláusula disyuntiva y comprueba si el cuerpo de la cláusula es cierto en la interpretación parcial. De serlo elige uno de los átomos de la cabeza adecuadamente instanciado y lo introduce en la interpretación, y así sucesivamente. Estudiando la teoría del ejemplo se puede comprobar que las cláusulas C_5 y C_6 no contribuyen a la demostración de la inconsistencia de la teoría. De hecho, si se observa el árbol de la figura anterior se puede ver que los subárboles con raíces *a* y *b* son idénticos; al elegir esta cláusula C_5 se duplica el espacio de búsqueda sin conseguir ninguna ventaja. Similarmente, la cláusula C_3 contiene átomos *c* y *d*, que no contribuyen a ninguna demostración de la falsedad de alguna cláusula Horn, y por tanto también dan como resultado un aumento estéril del espacio de búsqueda.

Este problema claramente sólo se da cuando la teoría a tratar no es mínimamente inconsistente, circunstancia que puede ser habitual en ciertos campos de aplicación de la demostración automática. La solución aportada por estos autores, que corresponde a una revisión y mejora del trabajo de Ramsay [Ramsay, 1991], es el analizar las cláusulas disyuntivas y utilizar una de éstas para realizar un paso de razonamiento hacia delante únicamente cuando la cláusula sea relevante a la demostración de la inconsistencia. De forma intuitiva una cláusula disyuntiva es relevante cuando cada uno de los literales de su cabeza han intervenido en alguna derivación correspondiente a la comprobación de la consistencia de la subteoría Horn incluida en la teoría a tratar. Claramente esta definición es dependiente del contexto, es decir según cual sea la interpretación parcial en un cierto instante el conjunto de cláusulas relevantes puede cambiar. Los autores proponen un técnica de etiquetado y desetiquetado de los literales que han intervenido en comprobación de la consistencia de la teoría Horn y que se realiza en tiempo de ejecución. Basándose en estos literales, durante la fase de aplicación de razonamiento hacia delante se establece si la cláusula disyuntiva seleccionado es relevante o no. El siguiente ejemplo ilustra la explicación anterior.

Ejemplo 3.18 (Comparativa del funcionamiento del SATCHMO y SATCHMORE) [Loveland, Reed y Wilson, 1995]

Sea el *S* el siguiente conjunto de cláusulas en formato SATCHMO con razonamiento mixto:

$C_1 = false$:- p, q
$C_2 = false$:- m, q
$C_3 = false$:- s
$C_4 = false$:- t
$C_5 = r$
$C_6 = r \Rightarrow p; q$
$C_7 = r \Rightarrow q ; m$
$C_8 = r \Rightarrow s ; t$

La figura siguiente corresponde al árbol SATCHMO del ejemplo anterior:

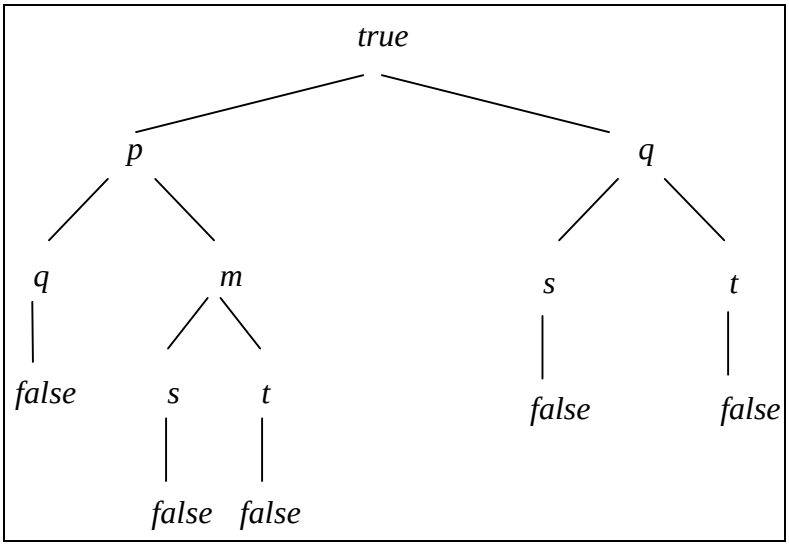


Figura 3.8: Árbol SATCHMO del Ejemplo 3.18

Observando la teoría del ejemplo se puede ver claramente que ésta no es mínimamente inconsistente (el subconjunto de cláusulas mínimamente inconsistente es C_3, C_4, C_5 y C_8) y que las cláusulas C_6 y C_7 no son relevantes en la demostración de la inconsistencia. SATCHMORE en el primer paso de la deducción, prueba que la subteoría Horn es consistente y detecta que los literales *p*, *m*, *s* y *t* son relevantes, ya que son usados en el árbol de fallo finito SLD de :-*false*. Así en la fase de razonamiento hacia delante las cláusula disyuntivas relevantes serán aquellas tales que cada uno de los átomos de su cabeza son relevantes. Así, la única cláusula disyuntiva que es relevante es C_8, siendo la única empleada. La siguiente figura muestra el árbol SATCHMORE para el mismo ejemplo.

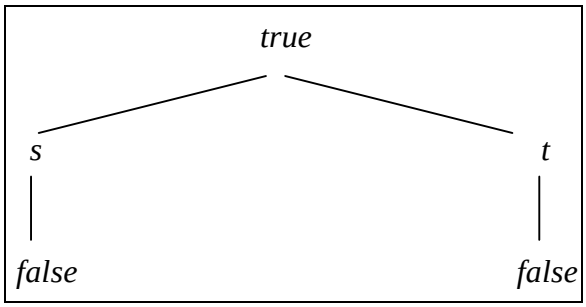


Figura 3.9: Árbol SATCHMORE del Ejemplo 3.18

Aplicando esta técnica del etiquetado y uso en el razonamiento hacia delante de las cláusula relevantes se consigue que el SATCHMO utilice el subconjunto de cláusulas mínimamente inconsistente de la teoría a tratar, en la mayoría de los casos.

Finalmente se van a presentar algunos ejemplos que ponen de manifiesto el principal problema, a juicio del autor de la presente tesis, de la técnica en la que se basa el demostrador SATCHMO (y que por tanto también sufre la versión mejorada SATCHMORE) y que está relacionado con el tratamiento de las teorías que no son de rango restringido. La solución que

los autores dan, la transformación de la Definición 3.9, claramente introduce un comportamiento en el demostrador que en algunos casos se puede tachar de muy ineficiente y en otros casos impide obtener respuestas con respecto a la consistencia del problema planteado.

Ejemplo 3.19 (Ineficiencia del SATCHMO en el tratamiento de teorías que no son de rango restringido)

Sea S una teoría inconsistente que no es de rango restringido formada por las cláusulas:

$$C_1 = \leftarrow p(f^n(a))$$
$$C_2 = p(x) \leftarrow$$

La transformación para convertir a un conjunto de cláusulas de rango restringido en forma SATCHMO genera las siguientes sentencias:

$$S_1 = p(f^n(a)) \Rightarrow false$$
$$S_2 = dom(X) \Rightarrow p(X)$$
$$S_3 = dom(X) \Rightarrow dom(f(X))$$
$$S_4 = true \Rightarrow dom(a)$$

La demostración de la inconsistencia de S por el demostrador SATCHMO constará de n pasos en los cuales el demostrador tiene que generar los n términos desde a hasta $f^n(a)$. Sin embargo cualquier demostrador basado en resolución o resolución lineal demostraría la inconsistencia en un solo paso.

Generalizando el ejemplo anterior, si se considera un conjunto de cláusulas mayor y un universo de Herbrand más extenso, esta ineficiencia se puede volver incluso incapacidad para obtener la demostración buscada.

Ejemplo 3.20 (Incapacidad para demostrar la consistencia)

Sea S una teoría consistente que no es de rango restringido formada por las cláusulas:

$$C_1 = \leftarrow p(g(X)) \wedge p(f(X))$$
$$C_2 = p(g(x)) \leftarrow$$

La transformación para convertir a un conjunto de cláusulas de rango restringido en forma SATCHMO y SATCHMORE genera las siguientes sentencias:

Transformación SATCHMO	Transformación SATCHMORE
$S_1 = p(g(X)), p(f(X)) \Rightarrow false$	$SM_1 = false$:- $p(g(X), p(f(X))$
$S_2 = dom(X) \Rightarrow p(g(X))$	$SM_2 = p(g(X))$:- $dom(X)$
$S_3 = dom(X) \Rightarrow dom(g(X))$	$SM_3 = dom(X) \Rightarrow dom(g(X))$
$S_4 = dom(X) \Rightarrow dom(f(X))$	$SM_4 = dom(X) \Rightarrow dom(g(X))$
$S_5 = true \Rightarrow dom(a)$	$SM_5 = dom(a)$

En este ejemplo las dos versiones del demostrador irán introduciendo en cada iteración un nuevos átomos del predicado *dom* debido a las sentencias S_3 y S_4, y a SM_3 y SM_4. Como el universo de Herbrand es infinito este proceso no acabará nunca, siendo el demostrador incapaz de fallar finitamente y responder que la teoría es consistente. En contraposición a este comportamiento, cualquier demostrador basado en el principio de resolución falla en dos pasos de demostración, concluyendo así la consistencia de la teoría.

Estos dos ejemplo anteriores ponen de manifiesto la principal limitación de SATCHMO: el tratamiento de las teorías que no son de rango restringido. La solución aportada por los autores, y que no ha sido mejorada posteriormente, no es satisfactoria ya que implica la instanciación de las cláusulas que no son de rango restringido usando los elementos del universo de Herbrand, volviendo a encontrarse con los mismos problemas que llevaron a denominar el método de los demostradores de David y Putman de la década de los 60 *método del Museo Británico*. Según el autor de esta tesis habría que pensar en alguna técnica que permitiera tratar las cláusulas de rango restringido de manera más adecuada, sin pasar por su instanciación a través del universo de Herbrand. Así, los átomos que se incorporaran a la interpretación parcial podrían estar desinstanciados, indicando su carácter universal, incorporando un tratamiento adecuado cuando fueran utilizados en la demostración de la falsedad de alguna cláusula. Este tratamiento debería utilizar la instanciación que posiblemente se produjera sobre estos átomos para que en la vuelta a los puntos de elección se pudiera utilizar de la forma adecuada.

3.4.3. CONCLUSIONES DEL DEMOSTRADOR SATCHMO

El demostrador SATCHMO está basado, al igual que el resto de los demostradores presentados en este capítulo, en un paradigma de búsqueda, construcción y eliminación de un posible modelo de la teoría a tratar. Sin embargo, el método de usado en el SATCHMO es marcadamente distinto al usado en el procedimiento de Eliminación de Modelos o sus derivados. La diferencia radica en que en el SATCHMO no se sigue una técnica de construcción lineal del modelo a partir de una cierta cláusula, haciendo uso de una razonamiento hacia atrás que descompone el problema a tratar en subproblemas, sino que utiliza de forma exhaustiva una

técnica de razonamiento hacia delante, o mixta en las versiones mejoradas, que permite revisar el conjunto posible de modelos de la teoría tratada.

Hay que destacar que los resultados experimentales que obtuvo SATCHMO fueron espectaculares para algunos problemas clásicos, como el *Streamroller* propuesto originalmente por Schubert y estudiado en [Walther, 1985] [Stickel, 1986] [Manthey y Bry, 1988], atrayendo todas las miradas de la comunidad de la demostración lógica inicialmente. Más tarde, este interés se diluyo al salir a la luz algunas de las carencias básicas del demostrador: tratamiento incorrecto de teorías que no son mínimamente inconsistentes, de teorías que no son de rango restringido, fracaso en la demostración de la consistencia en casos muy sencillos, etc.

3.5. LOS PROCEDIMIENTOS NEAR-HORN PROLOG

Los sistemas englobados bajo la denominación de *near-Horn Prolog* (en forma abreviada *nH-Prolog*) son una de las aportaciones más recientes de D. Loveland y sus colaboradores al campo de la demostración automática [Loveland, 1987] [Smith y Loveland, 1988] [Loveland, 1991] [Reed, Loveland y Smith, 1991] [Loveland y Reed, 1993]. El objetivo principal de los sistemas nH-Prolog es el de extender mecanismo de deducción de Prolog (o de forma más precisa del procedimiento SLD) para que sea capaz de tratar con cláusula disyuntivas, pero sin perder la forma básica de funcionamiento del Prolog. Así el tipo de problemas que es capaz de abordar los sistemas nH-Prolog con relativa eficiencia son aquellos que contienen un número de cláusulas disyuntivas pequeño. Esta limitación, puesta de manifiesto por los autores, es presentada como una ventaja, ya que según ellos con una ligera extensión a los sistemas Prolog se es capaz de tratar un tipo de problemas con una presencia de la disyunción pequeña y que son habituales en muchos campos cercanos a la programación lógica, como el tratamiento de planes, manipulación de bases de datos deductivas disyuntivas, etc.

Los sistemas nH-Prolog están basados en el uso de la regla denominada *splitting rule* (regla de división), que se deriva de uno de los criterios de simplificación propuestos por Davis y Putman para su sistema de demostración [Chang y Lee, 1973]. Brevemente esta regla se puede enunciar como sigue:

> Dado un conjunto de cláusulas S y una cláusula C de la forma $D_1 \vee D_2$, donde D_1 y D_2 son dos disyunciones de literales que contienen al menos un literal, $S \cup \{C\}$ es inconsistente si y sólo si $S \cup \{D_1\}$ y $S \cup \{D_2\}$ son inconsistentes de forma compatible. La compatibilidad en la demostración de la inconsistencia de las partes de la cláusula C se exige por la posible presencia de variables comunes en D_1 y D_2.

La regla de división permite dividir un problema inicial en subproblemas, incluso hasta llegar a tratar únicamente con cláusula unitarias. Los sistemas nH-Prolog realizan la

combinación de esta regla junto con el mecanismo de razonamiento lineal del procedimiento SLD. Este hecho queda patente en las siguientes palabras de D. Loveland: “presentar una deducción nH-Prolog como una secuencia de deducciones Prolog, una para cada caso a ser tenido en consideración”. Esto es, la aplicación de la regla de división permitirá reducir la teoría inicial a un conjunto de subteorías Horn en las cuales se podrá aplicar Prolog para obtener la repuesta buscada. Por otro lado es interesante destacar el principal objetivo que persiguen los autores en el desarrollo de los sistemas nH-Prolog y que también queda perfectamente recogido en las siguiente palabras de D. Loveland: “disponer de un procedimiento cuyo tiempo de computación crezca sólo en proporción a la desviación del conjunto de entrada con respecto a un conjunto Horn”.

De forma intuitiva una derivación nH-Prolog se puede entender como una serie de derivaciones Prolog, o más formalmente derivaciones SLD, tantas cuanto más se desvíe el conjunto inicial de cláusulas de ser Horn. Así si se supone que el conjunto inicial de cláusula S está formado por una conjunto de cláusulas Horn H y una cláusula disyuntiva C de la forma $L_1 \vee L_2 \leftarrow B$, donde B es una conjunción de átomos, nH-Prolog demuestra la inconsistencia de $H \cup \{C\}$ aplicando la regla de división sobre la cláusula C y probando que $H \cup \{L_1 \leftarrow B\}$ y $H \cup \{L_1 \leftarrow B\}$ son inconsistentes de forma compatible. Ya que estos dos últimos conjuntos son Horn para realizar esta prueba se utiliza un sistema Prolog. Claramente, este tratamiento se puede generalizar cuando la teoría contiene un conjunto indeterminado de cláusulas disyuntivas realizando la aplicación de la regla de división tantas veces como sea necesario.

En el presente capítulo se va a realizar la presentación de los principales sistemas nH-Prolog siguiendo las líneas principales de los trabajos de D. Loveland y sus colaboradores [Loveland, 1987] [Smith y Loveland, 1988] [Loveland, 1991] [Reed, Loveland y Smith, 1991] [Loveland y Reed, 1993]. Estos sistemas son los siguientes: nH-Prolog simple, la versión más sencilla de estos sistemas que no es completa; nH-Prolog progresivo, versión más compleja que es completo; y el nH-Prolog con herencia, sistema también completo que presenta ciertas optimizaciones con respecto al anterior.

Antes de presentar estos sistemas y estudiar sus principales características, es necesario introducir algunas definiciones previas.

3.5.1. DEFINICIONES PREVIAS

El formato usado en los sistemas nH-Prolog para la representación de las cláusulas es similar al utilizado en los sistemas Prolog con las dos modificaciones siguientes:

a) La cabeza de una cláusula puede contener más de un átomo, estando entonces éstos separados por un “;”, operador correspondiente a la disyunción lógica. Estas cláusula

se denominan *multicabeza* y serán aquellas que no son Horn (por ejemplo, la cláusula $p \vee q \vee \neg r$ se representa por p; q :- r.

b) La negación, que se puede representar en las teorías a las que se puede aplicar los sistemas nH-Prolog, es la negación lógica y no la negación como fallo de los sistemas Prolog que usan para ella el operador *not*. La negación lógica que se representa en los sistemas nH-Prolog se hace en las cláusulas disyuntivas en los átomos que están en la cabeza, como muestra la cláusula del párrafo anterior.

Por tanto las cláusulas de entrada en formato nH-Prolog tienen la forma siguiente:

$A_1; A_2; \ldots; A_m$:- $B_1, B_2, \ldots, B_n$

donde A_i $(1 \leq i \leq m)$ y B_j $(1 \leq i \leq n)$ son átomos, $m \geq 0$, $n \geq 0$ y $m+n>0$.

Loveland además propone una transformación para tratar las cláusulas negativas y construir un conjunto soporte de la teoría inicial. La transformación es la siguiente: toda cláusula negativa de la forma :- $B_1, B_2, \ldots, B_n$ se transforma a la cláusula q:- $B_1, B_2, \ldots, B_n$, donde q es un predicado que no aparece en la teoría inicial. La demostración de la inconsistencia de la teoría se hará buscando una refutación desde la cláusula inicial negativa unitaria $\leftarrow q$, hecho que se representa de la forma ?- q.

Previamente a las definiciones del concepto de deducción en los sistemas nH-Prolog se va a realizar una presentación de los formatos de representación de estas deducciones de manera informal.

Una deducción nH-Prolog está compuesta por una secuencia de *bloques* que representan deducciones parciales en la derivación, de forma que un bloque observado de forma aislada se corresponde esencialmente con una derivación SLD. Existen dos tipos de bloques: el primero en la derivación nH-Prolog, denominado *bloque de inicio* (que será el único si la teoría inicial es Horn) y el resto de los bloques que se denominan *bloques de reinicio*.

La deducción contenida en un cierto bloque se representa como una secuencia de *cadenas* $F_1, F_2, \ldots, F_n$, tal que cada F_i $(1 \geq i \geq n)$ es de la forma C # $a[D]$, donde C es una conjunción de átomos separados entre sí por comas, denominada *continuación*, a es un átomo denominado *cabeza activa* y D es una lista de átomos denominada *lista de cabezas diferidas*. Al símbolo # se le denomina *muro*.

A continuación se van a presentar formalmente las reglas de inferencia que permiten a partir de una cadena en una deducción obtener la siguiente cadena.

Definición 3.10 (Regla de reducción) [Loveland, 1991]

Sea F_i una cadena en una deducción de la forma $L_1, L_2, \ldots, L_n \# a_i$ [D_i]. La cadena siguiente F_{i+1} en la deducción se obtiene por reducción si se cumple uno de los siguientes puntos:

a) Existe una variante de una cláusula de la teoría inicial de la forma A :- B tal que A es un átomo y B una conjunción de átomos y existe un umg σ de L_1 (el átomo más a la izquierda en la continuación) y A. Entonces, $(B, L_2, \ldots, L_n)\sigma \# a_i\sigma$ [$D_i\sigma$] es la cadena siguiente en la deducción.

b) Existe una variante de una cláusula de la teoría inicial de la forma $A_1; A_2; \ldots; A_m$:- B tal que A_i ($1 \geq i \geq m$) es un átomo y B una conjunción de átomos y existe un umg σ de L_1 (el átomo más a la izquierda en la continuación) y A_i. Entonces, la cadena siguiente en la deducción es la cadena $(B, L_2, \ldots, L_n)\sigma \# a_i\sigma$ [$(A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m, D_i)\sigma$].

La aplicación de la regla de reducción en el caso a) es similar a la aplicación de un paso de resolución en el procedimiento SLD. En el caso b) se realiza la misma operación junto con la inclusión en la lista de cabezas diferidas del resto de átomos de la cabeza sobre los que no se realiza la unificación. En cualquiera de los dos casos la unificación se aplica a todos los elementos de la cadena resultante.

Definición 3.11 (Regla de cancelación) [Loveland, 1991]

Sea F_i una cadena en una deducción de la forma $L_1, L_2, \ldots, L_n \# a_i$ [D_i]. La cadena siguiente en la deducción F_{i+1} se obtiene por cancelación si existe un umg σ entre L_1 y a_i, siendo entonces dicha cadena de la forma $(L_2, \ldots, L_n)\sigma \# a_i\sigma$ [$D_i\sigma$].

Como se puede observar en las definiciones de las dos reglas de inferencias la regla de selección del átomo es el primero por la izquierda (de forma similar a los sistemas Prolog) aunque Loveland pone de manifiesto en [Loveland, 1991] que esta restricción es innecesaria pudiéndose utilizar cualquier otra regla de selección sin perder por ello las propiedades de los demostradores.

A continuación se presenta la definición de deducción nH-Prolog que posteriormente se refinará, dando lugar así a los diferentes sistemas nH-Prolog, nH-Prolog simple, nH-Prolog progresivo y nH-Prolog con herencia.

Definición 3.12 (Deducción nH-Prolog) [Loveland, 1991]

Dado un conjunto S de cláusulas en formato nH-Prolog, una deducción nH-Prolog en S es una secuencia de cadenas $F_1, F_2, \ldots, F_n$ que cumplen las siguientes condiciones:

1) F_1 es la cadena inicial que tiene la forma C_1 # □, de forma que su continuación C_1 es la meta original, no tiene cabeza activa y la lista de cabezas diferidas está vacía. El bloque que comienza con F_1 se denomina *bloque inicial*.

2) Para toda cadena F_i ($i<n$) en la deducción cuya continuación no es vacía, la cadena siguiente en la deducción F_{i+1} se obtiene por la aplicación de la regla de deducción o reducción.

3) Para toda cadena F_i ($i<n$) en la deducción de la forma C_i # a_i [D_i] de manera que su continuación C_i es vacía y su lista de cabezas diferidas D_i no es vacía, la siguiente cadena en la deducción F_{i+1} corresponde a un bloque de reinicio si se cumple que bloque en curso es el inicial o es uno de reinicio en el cual se ha producido una operación de cancelación. Entonces dicho bloque de reinicio comienza con la cadena F_{i+1} que es de la forma C_{i+1} # a_{i+1} [D_{i+1}] tal que a_{i+1} es el primer átomo de la lista de cabezas diferidas D_i y D_{i+1} se obtiene de D_i a base de eliminar el primer átomo a_{i+1} de la lista (la forma de la continuación C_{i+1} corresponde a los distintos sistemas nH-Prolog que se presentarán más adelante).

4) La última cadena en la deducción F_n de la forma C_n # a_n [D_n] ha de cumplir uno de los siguientes puntos:

   a) La continuación C_n es vacía y el bloque en curso es el inicial. Entonces la deducción se considera que ha tenido éxito.

   b) La continuación C_n es vacía, la lista de cabezas diferidas D_n está vacío, el bloque en curso es de reinicio y se ha producido una operación de cancelación. Entonces la deducción se considera que ha tenido éxito.

   c) La continuación C_n no es vacía y el bloque en curso es el inicial o de reinicio y no se puede aplicar ni la regla de reducción ni la de cancelación. Entonces la deducción se considera fallada.

   d) La continuación C_n es vacía, la lista de cabezas diferidas D_n está vacío, el bloque en curso es de reinicio y no se ha producido una operación de cancelación. Entonces la deducción se considera fallada.

Como se comentó anteriormente una deducción nH-Prolog se divide en un conjunto de bloques, el inicial y los de reinicio. Claramente en el bloque inicial sólo es posible aplicar la regla de reducción ya que en él no existe cabeza activa. En los bloques de reinicio se puede aplicar los dos reglas, reducción y cancelación. Al terminar un bloque, esto es cuando la continuación de la cadena en curso está vacía, si la lista de cabezas diferidas no está vacía es necesario establecer

un bloque de reinicio, cuya cabeza activa será la primera por la izquierda de la lista de cabezas diferidas del bloque que termina. Este bloque de reinicio hereda la lista de cabezas diferidas del bloque anterior exceptuando la primera que se convierte, como se acaba de decir, en cabeza activa. Un bloque de reinicio acaba con éxito siempre que se aplique al menos una cancelación. Los diferentes sistemas nH-Prolog que se presentan a continuación aparecen por las diferentes alternativas que existen para establecer la continuación de la cadena inicial de los bloques de reinicio.

3.5.2. nH-PROLOG SIMPLE

El nH-Prolog simple es el más sencillo de los sistemas nH-Prolog. El nH-Prolog simple es equivalente al procedimiento SLD para teorías Horn, siendo por tanto completo para estas teorías. Sin embargo, el nH-Prolog simple no es completo para teorías clausales generales. El interés de este sistema radica en su sencillez, ya que de una manera comprensible y clara establece las ideas fundamentales y la forma de funcionamiento de los sistemas nH-Prolog.

La definición de una deducción nH-Prolog simple es idéntica a la Definición 3.12, con la única diferencia de que en el punto 3) se establece cuál es la continuación de la cadena inicial de los bloques de reinicio. Para el nH-Prolog simple la continuación de la cadena inicial de los bloques de reinicio es una variante de la meta inicial a probar. A continuación se presentan algunos ejemplos para clarificar la forma de funcionamiento.

Ejemplo 3.21 (Deducción nH-Prolog simple)

Sea S el siguiente conjunto de cláusulas expresado en formato nH-Prolog:

$C_1 = q$:- $p(X)$	transformación de la meta inicial $\leftarrow p(x)$
$C_2 = p(a)$; $p(X)$:- $q(X)$	
$C_3 = q(b)$	

Abajo se muestra una deducción nH-Prolog simple que demuestra la inconsistencia de dicho conjunto:

0.	?- q	
1.	:- $p(X)$	
2.	:- $q(X)$ # [$p(a)$]	
3.	:- # [$p(a)$]	fin del bloque de inicio, sustitución computada X/b
4.	:- q # $p(a)$ □	reinicio
5.	:- $p(X)$ # $p(a)$ □	
6.	:- # $p(a)$ □	cancelación, sustitución computada X/a

Obsérvese que el nH-Prolog simple obtiene dos sustituciones computadas en los dos bloques, el inicial y el de reinicio, X/b y X/a respectivamente. Así se puede establecer que la disyunción de la negación de la meta inicial instanciada por las sustituciones es consecuencia lógica del resto de cláusulas de S, esto es $\{C_2, C_3\} \models p(a) \vee p(b)$.

Ejemplo 3.22 (Deducción del nH-Prolog simple)

Sea S el siguiente conjunto inconsistente de cláusulas en formato nH-Prolog:

$C_1 = q$:- $p(X), q(X)$ transformación de la meta inicial $\leftarrow p(x) \wedge q(x)$
$C_2 = q(X)$:- $p(X)$
$C_3 = p(X)$:- $q(X)$ $C_4 = p(X)$; $q(X)$

Abajo se puede observar una deducción nH-Prolog simple que demuestra la inconsistencia de S.

0.	?- q	
1.	:- $p(X), q(X)$	
2.	:- $q(X)$ # $[q(X)]$	
3.	:- # $[p(X), q(X)]$	fin del bloque de inicio
4.	:- q # $p(X)$ $[q(X)]$	reinicio
5.	:- $p(X')$, $q(X')$ # $p(X)$ $[q(X)]$	
6.	:- $q(X)$ # $p(X)$ $[q(X)]$	cancelación, sustitución X'/X
7.	:- $p(X)$ # $p(X)$ $[q(X)]$	
8.	:- # $p(X)$ $[q(X)]$	cancelación, fin de bloque
9.	:- q # $q(X)$ □	reinicio
10.	:- $p(X')$, $q(X')$ # $q(X)$ □	
11.	:- $q(X')$, $q(X')$ # $q(X)$ □	
12.	:- $q(X)$ # $q(X)$ □	cancelación, sustitución X'/X
13.	:- # $q(X)$ □	cancelación

La demostración formal de la corrección del nH-Prolog simple se puede encontrar en los trabajos de Loveland y sus colaboradores, sin embargo se desea presentar aquí la idea intuitiva de esta corrección que a su vez también proporciona una idea clara del funcionamiento de los sistemas nH-Prolog. La idea básica que subyace a los sistemas nH-Prolog es combinar el Prolog y el análisis de casos para conseguir una forma de razonamiento aplicable a los problemas que no son Horn pero sin apartarse demasiado del razonamiento Prolog. Se va a simplificar la exposición al caso proposicional para clarificarlo lo máximo posible. El problema de probar la inconsistencia de un cierto conjunto de cláusulas se reduce al transformar éste al formato nH-Prolog a demostrar que $S \cup \{\leftarrow q\}$ es inconsistente, esto es que la meta q es consecuencia

lógica del resto del conjunto S de las cláusulas transformadas, o sea que $S \models q$. En el caso de existir una demostración Prolog de q usando las cláusulas Horn de S, bastaría para probarlo. Esto es lo que sucede si el bloque inicial termina y la lista de cabezas diferidas este vacía. Claramente el problema está demostrado, ya que la única regla que se puede aplicar en el bloque inicial es la reducción que es equivalente a la resolución de entrada. Si no existe una demostración que únicamente use cláusulas Horn, entonces la regla de división puede ser utilizada para conseguir una demostración por análisis de casos. La forma de regla de división que se aplica en los sistemas nH-Prolog afirma que dado un conjunto S de cláusulas, una cláusula disyuntiva $A_1 \vee A_2 \vee \ldots \vee A_m \leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_n$ y la meta a probar q, entonces

$S \cup \{A_1 \vee A_2 \vee \ldots \vee A_m \leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_n\} \models q$ si
$S \cup \{A_i \leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_n\} \models q$, para algún $1 \leq i \leq m$, y
$S \cup \{A_j\} \models q$, para cada $j = 1, \ldots, i\text{-}1, i\text{+}1, \ldots, m$

En otras palabras, $S \cup \{A_i \leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_n\} \models q$ puede ser entendido como el caso en el cual todos los átomos de la cabeza excepto A_i son descartados (o asumidos como falsos). Los otros casos de la forma $S \cup \{A_j\} \models q$ representan las alternativas donde cada átomo de la cabeza es supuesto cierto y añadido como un hecho condicional. La regla de división afirma que la combinación de estos casos basta para demostrar que q es consecuencia lógica del conjunto original de cláusulas. Ya que la aplicación de la regla de división produce casos en los que el número de cláusulas disyuntivas disminuye en uno, cualquier conjunto de cláusulas puede ser repetidamente dividido hasta llegar eventualmente a obtener casos en los que se pueden construir derivaciones Prolog. La explicación anterior es aplicable al problemas generales con variables pero teniendo presente que la forma de aplicación de la regla de división ha de tener en cuenta las diferentes sustituciones que se producen en los casos que va tratando.

El nH-Prolog simple es correcto pero no completo, debido a que la continuación en los bloques de reinicio es la meta original y esto puede provocar la aparición de bucles en las demostraciones que impidan probar el problema inicial. El siguiente ejemplo, debido a D. Loveland [Loveland, 1991], muestra este problema.

Ejemplo 3.23 (Incompletitud del nH-Prolog simple)

Sea S el siguiente conjunto inconsistente de cláusulas en formato nH-Prolog:

$C_1 = q$:- a, b transformación de la meta inicial $\leftarrow a \wedge b$

$C_2 = a$:- c	$C_5 = b$:- e
$C_3 = a$:- d	$C_6 = b$:- f
$C_4 = c$; d	$C_7 = e$; f

Abajo se encuentra una deducción nH-Prolog simple infinita en la que es posible observar las causas de la incompletitud del nH-Prolog simple.

0.	?- *q*	
1.	:- *a, b*	
2.	:- *c, b*	
3.	:- *b* # [*d*]	
4.	:- *e* # [*d*]	
5.	:- # [*f, d*]	fin del bloque de inicio
6.	:- *q* # *f* [*d*]	reinicio
7.	:- *a, b* # *f* [*d*]	
8.	:- *c, b* # *f* [*d*]	
9.	:- *b* # *f* [*d, d*]	
10.	:- *f* # *f* [*d, d*]	
11.	# *f* [*d, d*]	cancelación
12.	:- *q* # *d* [*d*]	reinicio
13.	:- *a, b* # *d* [*d*]	
14.	:- *d, b* # *d* [*d*]	
15.	:- *b* # *d* [*d*]	cancelación
16.	:- *e* # *d* [*d*]	
17.	:- # *d* [*f, d*]	
18.	:- *q* # *f* [*d*]	reinicio (idéntico a línea 6)

etc.

El motivo por el cual nH-Prolog simple no es completo es por la forma en la que trata la cabeza activa y la lista de cabezas diferidas. Se puede comprobar que en el nH-Prolog simple las cabezas diferidas son añadidas tan rápido como son eliminadas.

3.5.3. NH-PROLOG PROGRESIVO

La incompletitud del nH-Prolog simple es alcanzada en el nH-Prolog progresivo modificando la cadena inicial utilizada en los bloques de reinicio. Más exactamente la diferencia consiste en la continuación del bloque de reinicio, que en el nH-Prolog simple es una variante de la cadena inicial y en el nH-Prolog progresivo es una cadena anterior en la deducción que cumple ciertos requisitos. Las siguientes definiciones previas son necesarias para la presentación de este sistema.

Definición 3.13 (Meta apelante de una submeta) [Loveland, 1991]

Se denomina *meta apelante* a una submeta *g* de una continuación en una deducción nH-Prolog a la meta que provoca el uso de una cláusula que introduce a la meta *g* en la continuación. La meta inicial de cualquier bloque no tiene meta apelante.

Definición 3.14 (Camino de ancestros de una submeta) [Loveland, 1991]

Dada una meta *g* en una continuación de una deducción nH-Prolog, el *camino de ancestros de g* contiene a *g*, a la meta apelante de *g* y cada una de las metas apelantes de cualquier miembro del camino de ancestros.

Definición 3.15 (Lista de ancestros de una cabeza diferida y lista de ancestros de un bloque) [Loveland, 1991]

Dada una deducción nH-Prolog, los siguientes puntos definen los conceptos de lista de ancestros de una cabeza diferida y lista de ancestros de un bloque:

a) La lista de ancestros del bloque inicial es la lista vacía.
b) La lista de ancestros de un bloque de reinicio es la lista de ancestros de su cabeza activa.
c) La lista de ancestros de una cabeza diferida es la secuencia de ancestros de la meta apelante que introdujo dicha cabeza junto con la lista de ancestros del bloque en el que fue diferida.

Las listas de ancestros, por tanto, se asocian a las cabezas diferidas y sus variables serán instanciadas por cualquier sustitución que afecte a dichas cabezas.

Ejemplo 3.24 (Listas de ancestros de bloques de cabezas diferidas)

Sea *S* el conjunto de cláusulas del Ejemplo 3.22. Abajo se puede observar la deducción nH-Prolog simple para *S*.

0.	?- q	
1.	:- $p(X)$, $q(X)$	
2.	:- $q(X)$ # [$q(X)$]	
3.	:- # [$p(X)$, $q(X)$]	fin del bloque de inicio
4.	:- q # $p(X)$ [$q(X)$]	reinicio
5.	:- $p(X')$, $q(X')$ # $p(X)$ [$q(X)$]	
6.	:- $q(X)$ # $p(X)$ [$q(X)$]	cancelación, sustitución X'/X
7.	:- $p(X)$ # $p(X)$ [$q(X)$]	
8.	:- # $p(X)$ [$q(X)$]	cancelación, fin de bloque
9.	:- q # $q(X)$ □	reinicio
10.	:- $p(X')$, $q(X')$ # $q(X)$ □	
11.	:- $q(X')$, $q(X')$ # $q(X)$ □	
12.	:- $q(X)$ # $q(X)$ □	cancelación, sustitución X'/X
13.	:- # $q(X)$ □	cancelación

Para los tres bloques existentes la lista ancestros de las cabezas diferidas son las siguientes:

nº de cadena	*cabezas diferidas*	*lista de ancestros*
2	$q(X)$	$[q, p(X)]$
3	$p(X)$, $q(X)$	$[q, q(X)]$, $[q, p(X)]$
4	$p(X)$, $q(X)$	$[q, q(X)]$, $[q, p(X)]$
8	$p(X)$, $q(X)$	$[q, q(X)]$, $[q, p(X)]$
9	$q(X)$	$[q, p(X)]$

La lista de ancestros de una cabeza, ya sea ésta activa o diferida, se corresponde con los distintos átomos seleccionados a lo largo de la deducción que dan lugar a la introducción en la deducción de dicha cabeza. Obsérvese que las listas de ancestros se corresponden con los ancestros del sistema MESON basado en secuencias, propuesto por Plaisted [Plaisted, 1988], pero aplicando en este caso un refinamiento negativo, al mantenerse en la lista de ancestros los átomos seleccionados en el cuerpo de las cláusulas en formato implicativo, esto es los literales negativos de las cláusulas en formato disyuntivo.

Definición 3.16 (Criterio de determinación de la continuación de la cadena inicial de los bloques de reinicio para el nH-Prolog progresivo) [Loveland, 1991]

La continuación de la cadena inicial de un bloque de reinicio en el nH-Prolog progresivo es un ancestro de la lista de ancestros de la cabeza activa de dicho bloque.

Con esta forma de determinación de la continuación de la cadena de un bloque de reinicio el nH-Prolog progresivo es correcto y completo, como demuestra Loveland en [Loveland, 1991]. Sin embargo se puede fácilmente observar de la definición anterior que este criterio es indeterminista y para alguna de las posibles formas de selección del ancestro puede dar lugar a que el demostrador sea incapaz de encontrar la refutación. Es, por tanto, necesario permitir el uso de los diferentes ancestros como meta de reinicio hasta que la demostración con éxito sea encontrada, ya que de no ser así la completitud se pierde. Como ejemplo, baste señalar que si el ancestro elegido para establecer la continuación de la cadena inicial del bloque de reinicio es siempre la meta inicial[6] de la deducción, entonces el nH-Prolog progresivo es equivalente al nH-Prolog simple perdiendo, por tanto, su completitud. Claramente el criterio que se establezca para la selección de este ancestro también tendrá implicaciones en el comportamiento temporal del demostrador. Por estos motivos expuestos, se considera interesante examinar brevemente las características más procedurales del sistema nH-Prolog progresivo.

[6] Esto siempre es posible ya que la meta inicial está en la lista de ancestros de cualquier cabeza.

El nH-Prolog progresivo, como el resto de los sistemas nH-Prolog, sigue una estrategia de búsqueda en profundidad, aunque, como el propio autor afirma, podrían ser aplicadas otras estrategias como la búsqueda en profundidad iterada. En la aplicación de una de las reglas de inferencia de los sistema nH-Prolog en una deducción, el criterio de selección del átomo de la continuación de la cadena actual sobre el cual se aplica dicha regla (denominado habitualmente *regla de computación*) consiste en elegir el átomo situado más a la izquierda, de manera similar a la utilizada por los sistemas Prolog. Dada la lista de cabezas diferidas de la cadena final de un bloque, el criterio de selección de la cabeza activa del bloque de reinicio siguiente (denominada *regla de computación de bloque*) consiste en elegir la cabeza situada más a la izquierda. Sin embargo las propiedades de corrección y completitud de los sistemas nH-Prolog son independientes de estos criterios, siendo posible, por tanto, modificarlos para conseguir incrementar la eficiencia.

La característica fundamental del nH-Prolog progresivo reside en el uso de búsquedas alternativas para la demostración de una meta, lo que en terminología Prolog se suele conocer como el "redo". Para las metas de un bloque diferentes de la meta inicial se sigue el criterio, similar al del Prolog, de reintentar la siguiente cláusula unificable con la meta en la secuencia de cláusulas del programa. El cambio con respecto al Prolog consiste en que sólo se reconsidera una meta inicial de un bloque de reinicio. El criterio que se sigue para elegir dicha meta consiste en seleccionarla entre los elementos de la lista de ancestros ordenada de más cercanos a más lejanos, esto es, seleccionado en primer lugar la meta que dio lugar a la introducción de la cabeza activa y si ésta no permite la prueba del problema, se selecciona la siguiente, y así sucesivamente hasta llegar al ancestro más lejano que corresponde a la meta inicial de la deducción. Una vez seleccionada dicha meta de la continuación de la cadena inicial del bloque de reinicio, el nH-Prolog progresivo utiliza un criterio adicional para la elección de la cláusula unificable con la meta en curso que consiste en seleccionar primero a la siguiente cláusula, según el orden establecido en el programa, a la cláusula que fue utilizada cuando la cabeza fue diferida. Loveland justifica estos criterios en [Loveland, 1991] afirmando que éstos son subóptimos atendiendo a la eficiencia temporal. Así, llega a concluir que el nH-Prolog progresivo explora como mucho el doble de espacio que el nH-Prolog simple en el caso de que este último fuera capaz de encontrar la demostración del problema.

Las propiedades de corrección y completitud del nH-Prolog progresivo se establecen en los siguientes dos teoremas, cuyas demostraciones pueden encontrarse en [Loveland, 1991].

Teorema 3.3 (Corrección del nH-Prolog progresivo) [Loveland, 1991]

Sea P un programa nH-Prolog[7], R una regla de computación y R_b una regla de computación de bloque. Si existe una demostración con éxito nH-Prolog progresivo (a partir de P y usando R y R_b) entonces el programa P es inconsistente.

Teorema 3.4 (Completitud del nH-Prolog progresivo) [Loveland, 1991]

Sea P un programa nH-Prolog. Si P es inconsistente entonces existe una demostración con éxito nH-Prolog progresivo a partir de P usando una regla de computación R y una regla de computación de bloque R_b.

En un trabajo posterior Loveland y Reed [Loveland y Reed, 1993], proponen una nueva versión denominada nH-Prolog con herencia, que se puede considerar una optimización del nH-Prolog simple. Este sistema es correcto y completo para la demostración de teoremas en teorías clausales. La peculiaridad fundamental es el uso en un cierto bloque de las cabezas activas canceladas en bloques anteriores en la deducción. Este mecanismo es denominado *herencia* por los autores y de ahí su nombre. Así, en los bloques se pueden utilizar para realizar operaciones de cancelación tanto la cabeza activa natural como las heredadas. Para llevar a cabo este mecanismo es necesario ir almacenado de forma dinámica las cabezas activas que son usadas en operaciones de cancelación.

La idea intuitiva del uso de las cabezas heredadas es que las mismas permanecen como hipótesis para aquella parte de la deducción concerniente a su demostración. Aunque pueda parecer que este uso de las cabezas heredadas es similar la conocida utilización de lemas, no es equivalente ya que estos últimos son hechos demostrados, mientras que las cabezas se han de ver como presunciones válidas dentro del ámbito de su propia demostración. El uso de lemas acarrea bastante problemas cuando son universales, esto es con presencia de variables, ya que hay que establecer claramente cual es el contexto en el cual estos lemas son aplicables. La herencia de esta versión nH-Prolog hay que verla, en cierto modo, equivalente a Eliminación de Modelos en las que las derivaciones subsidiarias son construidas en teorías cada vez más amplias al tomar en consideración el modelo que se está construyendo.

El nH-Prolog con herencia sigue un criterio de determinación de la continuación de la cabeza inicial de los bloques de reinicio similar al nH-Prolog simple, esto es, hace uso de una variante de la meta de la continuación de la cadena inicial de la deducción. A pesar de compartir esta característica con el nH-Prolog simple, que hacía que este último no fuera completo, el nH-Prolog con herencia si lo es debido al mecanismo de uso de las cabezas heredadas.

[7] Un programa nH-Prolog es equivalente a un conjunto de cláusulas en formato nH-Prolog con un cierto orden establecido.

La justificación que da Loveland en [Loveland y Reed, 1993] del nH-Prolog con herencia es que, en general, se obtienen deducciones más cortas que las que proporcionan los otros sistemas nH-Prolog al utilizar información adicional durante las mismas, reduciendo así el espacio de búsqueda. Lo que no comenta Loveland es el coste temporal que conlleva el almacenamiento y tratamiento de la herencia siendo, por lo tanto, imposible la evaluación total del demostrador.

3.5.4. CONCLUSIONES DE LOS SISTEMAS NH-PROLOG

Como se comentó al principio de este apartado el propósito fundamental de los sistemas nH-Prolog es el de tratar de forma eficiente teorías cláusulas generales que sean casi Horn, esto es, que el número de cláusulas disyuntivas en la teoría sea pequeño mediante un procedimiento que sea lo más parecido posible, conceptual y proceduralmente, al Prolog. Este propósito es claramente perseguido por Loveland para conseguir unos sistemas que sean fácilmente admitidos en las comunidades de la Programación Lógica y Bases de Datos Deductivas como estándares, como lo es el Prolog, para el tratamiento de teorías que no son Horn. Los sistemas nH-Prolog pueden entenderse como una combinación del Prolog y el análisis de casos, de manera que las deducciones nH-Prolog pueden verse como un conjunto de derivaciones Prolog que van reduciendo las metas iniciales y en las cuales las cabezas de las cláusulas disyuntivas que interviene dan lugar a los casos que son tratados en otras deducciones. El análisis de casos se realiza mediante la aplicación de la regla de división subyacente al tratamiento de las cabezas diferidas y los bloques de reinicio.

Como critica a estos sistemas hay que destacar, en primer lugar, que la sencillez y claridad que Loveland pretende dejar patente y que se puede observar en los sistemas más simples, nH-Prolog simple, se convierte en obscuridad y complejidad cuando se quiere conseguir sistemas completos o más eficientes, cayendo en las misma circunstancias que se intentaban evitar. En segundo lugar, se echa de menos en los trabajos de Loveland y sus colaboradores los resultados experimentales. Las afirmaciones que realizan en cuanto a eficiencia y costes espaciales o temporales no son corroboradas por resultados experimentales, a pesar de que Loveland afirma que todos sus sistemas se encuentran implementados.

3.6. EL DEMOSTRADOR SETHEO

El objetivo del sistema SETHEO [Letz, Schuman, Bayerl y Bibel, 1992] es el de dotar a la comunidad de la Inteligencia Artificial de un contexto apropiado para la representación y tratamiento de problemas. Este sistema satisface los requerimientos de tener una expresividad suficiente para cubrir la mayoría de los problemas, poseer una semántica bien definida y, por último, proporcionar unos mecanismos de inferencia eficientes. El sistema SETHEO se puede

observar desde dos perspectivas diferentes: por un lado, puede ser usado como intérprete de un lenguaje de programación, denominado LOP, que extiende y mejora Prolog considerablemente, conservando sus prestaciones; por otro lado, el sistema es una demostrador de teoremas para la lógica de primer orden. Este segundo aspecto será el que se presentará en esta sección.

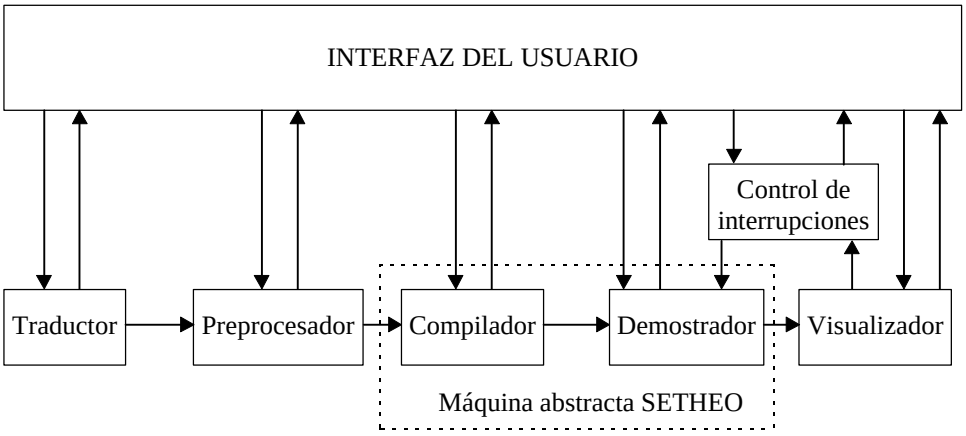
El demostrador SETHEO es correcto y completo para la demostración de teoremas en lógica de primer orden. Sus componentes más relevantes son: un módulo de preprocesamiento para reducir el tamaño del problema a tratar, una unidad de demostración basada en un refinamiento del método de conexión que es equivalente a Eliminación de Modelos. Aparte de estos dos componentes existen otros que proporcionan al sistema un entorno más agradable al usuario, como son, un traductor de fórmulas de primer orden a cláusulas, un visualizador de demostraciones, y un módulo que gestiona la interrupciones en la demostración. La figura de abajo muestra la arquitectura del sistema SETHEO.

Figura 3.10: Arquitectura del sistema SETHEO

Los módulos fundamentales son el preprocesador y la máquina abstracta SETHEO, compuesta por un compilador y el propio demostrador. Las características más relevantes de estos módulos se expondrán a continuación.

3.6.1. PREPROCESAMIENTO

El modulo de preprocesamiento de que dispone el SETHEO es potente y permite reducir los problemas en muchos casos. Este énfasis en el preprocesamiento está motivado por la analogía con la forma de proceder del humano al abordar un problema, que antes de tratar de resolver con un método específico intenta analizar y reducir el problema. Así, el tamaño del espacio de búsqueda del problema se reduce, en algunos casos radicalmente. El modulo de



preprocesamiento del SETHEO incluye los siguientes tratamientos:

- reducción de tautologías,
- resolución unitaria restringida,
- reducción por subsumción,
- reducción pura,
- reducción de literales aislados.

Los tres primeros preservan la equivalencia lógica, mientras que los dos últimos sólo preservan la (in)consistencia del problema a tratar. Desde el punto de vista de la demostración de teoremas esta diferencia no es significativa. Los tres primeros tratamientos se aplican a cada una de las cláusulas del problema de forma aislada, a diferencia de los dos últimos que necesitan tratar el conjunto de cláusulas que formalizan el problema globalmente. Una explicación de cada una de estas reducción se puede encontrar en [Bibel, 1987] [Robinson, 1965] [Letz, Schuman, Bayerl y Bibel, 1992].

Los autores ponen de manifiesto la importancia del preprocesamiento para la reducción de los problemas y dan como dato que un 30% de los problemas en los experimentos fueron reducidos y que incluso un 10% de los mismos se demostraron en el propio preproceso.

3.6.2. COMPILACIÓN

En la fase de compilación se prepara el problema para su ejecución en el procedimiento de demostración principal. Esta fase comprende la generación de un grafo de conexión, la transformación de las cláusulas originales en contrapositivas, el reordenamiento de los literales en los cuerpos de las contrapositivas y, opcionalmente, la inclusión de código para la aplicación de algún método de poda del espacio de búsqueda.

La idea del grafo de conexión está basado en el concepto de unificación débil [Eder, 1985]. Este grafo retiene para cada cláusula del problema las posibles conexiones, correspondientes a pasos de resolución, que pueden darse. De esta forma, de manera inmediata se puede saber cuáles son las cláusula de entrada a utilizar para un literal seleccionado de una cláusula. Claramente este método presenta la ventaja de evitar, en numerosos casos, una elevada cantidad de aplicaciones de la unificación.

Ya que, como se verá más adelante, el procedimiento de demostración principal está basado en las técnicas de compilación a la Prolog es necesario la generación de contrapositivas si se pretende aprovechar la eficiencia del mismo. Así, a partir de las cláusula generales se obtiene un conjunto de contrapositivas. La forma de la mismas y sus construcción ya se expuso anteriormente en la presente tesis.

El reordenamiento de los literales en las contrapositivas puede producir un decremento del coste de localizar la demostración al reducirse la vuelta atrás. Esta característica es compartida con los sistemas Prolog. Sin embargo estos últimos dejan al programador la tarea de buscar el ordenamiento óptimo para cada problema. No es así en el sistema SETHEO, ya que el ordenamiento es fijado por el sistema en base a ciertos criterios. La idea que subyace para elegir el mejor ordenamiento es la siguiente: si una tarea se divide en subtareas (aplicación de una extensión) y no está claro qué subtarea puede ser resuelta, entonces es razonable intentar primero aquella con menos probabilidades de tener éxito. Así se consigue reducir al mínimo el trabajo redundante de probar varias veces la mismas subtareas. La forma de realizar el reordenamiento por SETHEO es estática en tiempo de compilación, aunque los autores apuntan que éste también se puede efectuar dinámicamente en tiempo de ejecución con un coste computacional alto. El reordenamiento que aplica el SETHEO da preferencia a los literales de las contrapositivas que satisfacen uno de estos criterios:

- literales con el menor valor de conexión[8] en la matriz.
- literales más especializado (por ejemplo, literales base).
- literales que comparten más variables con la cabeza.

Para cada uno de estos criterios se define una función numérica y posteriormente se obtiene un valor normalizado correspondiente a las tres criterios.

3.6.3. MÁQUINA DE INFERENCIA

La máquina de inferencia del sistema SETHEO está basado en Eliminación de Modelos. De forma más precisa, está basado en un refinamiento del método de conexión, del cual Eliminación de Modelos es una realización formato lineal. La definiciones formales en el contexto del método de conexión no se incluyen, pero de forma intuitiva la idea consiste en realizar un árbol o tabla de conexión en el cual sea posible comprobar los caminos que se pueden formar usando las cláusulas del problema en busca de la inconsistencia. En este contexto se definen las reglas de inferencia de Eliminación de Modelos: extensión y reducción.

La completitud de la máquina de inferencia se establece si para cada problema inconsistente se detecta que lo es. Claramente, la completitud sólo queda garantizada si se emplea un procedimiento de búsqueda completo que realmente recorra el espacio (posiblemente infinito) asociado al problema tratado. El procedimiento utilizado en SETHEO se basa en la búsqueda iterativa con vuelta atrás como en [Stickel, 1988], ya expuesta en el presente capítulo.

[8] Este valor depende de la cantidad de contrapositivas cuya cabeza puede unificar con el literal.

La diferencia fundamental del sistema SETHEO con respecto al PTTP reside en la utilización de tres diferentes forma de establecer el límite a la profundidad a base de tomar:

- el número de pasos de inferencia realizados,
- el tamaño de la lista de ancestros, o
- el número de copias utilizadas de cada contrapositiva.

Los autores apuntan que estas tres formas dan lugar a un crecimiento diferente del espacio de búsqueda en cada iteración. En particular, el crecimiento del espacio de la primera forma es el suave, mientras que el de la última es el más rápido. También apuntan que la segunda forma es la que experimentalmente ha dado mejores resultados.

Los métodos de poda utilizados en el SETHEO para reducir el espacio de búsqueda son los siguientes: reducción de la vuelta atrás por medio de la detección de respuestas más generales, eliminación de tautologías y aplicación de la subsumción. También sugieren en sus trabajos la utilidad de la aplicación de factorización, lemas y de esquemas de demostración para la reducción del espacio aunque no están implementados en el sistema actualmente desarrollado. Los siguientes párrafos se dedican a exponer con más detalle las podas aplicadas por el sistema SETHEO.

Hay que destacar que con respecto a la eliminación de tautologías, los autores exponen la necesidad de realizar dos comprobaciones: ninguna cláusula en la demostración es una tautología y que ninguna meta en la demostración tiene un ancestro idéntico a si misma. Destacan el elevado coste de una comprobación constante dinámica y proponen la inclusión de código en compilación que realice la primera comprobación y la ejecución esporádica de la segunda comprobación. El código insertado consiste en desigualdades que prohiben la aparición de instancias de cláusulas que sean una tautología.

La subsumción es uno de los métodos de prueba esenciales en demostración de teoremas por resolución, como se destaca en [Wos y Overbeek, 1993]. La subsumción permite eliminar cualquier cláusula implicada por otra durante el proceso de demostración, evitando de esta forma el tratamiento de información redundante que en algunos casos puede llegar a ser enorme. Sin embargo hay que tener en cuenta que la aplicación de la subsumción exhaustivamente en tiempo de ejecución puede llegar a tener un coste tan grande que no compense la poda conseguida. Los autores proponen un método muy sencillo y económico de la aplicación de la subsumción que consisten en la detección, por criterios sintácticos y en tiempo de compilación, de posibles cláusulas subsumibles y la inclusión de código (también incluyendo desigualdades) en dichas cláusula que captura los efectos de la subsumción durante la ejecución de la demostración. Este hecho se muestra en el siguiente ejemplo.

Ejemplo 3.25 (La subsumción en el SETHEO) [Letz, Schuman, Bayerl y Bibel, 1992]

Sea las siguientes cláusulas:

$C_1 = p(x, z) \vee \neg\, p(x, y) \vee \neg\, p(x, y)$
$C_2 = p(v, v)$
$C_3 = p(a, b)$
$C_4 = \neg\, p(b, c)$

Se puede comprobar rápidamente que las siguientes instancias de C_1 son subsumidas por las otras cláusulas: $x = z$, subsumida por la cláusulas C_2; $x = a$ y $z = b$, subsumida por la cláusula C_3; y por último $x = b$ y $y = c$, debido al segundo literal, y $x = b$ y $z = c$ debido al último literal subsumida por C_4 en los dos casos. Esto lleva a la introducción de un conjunto de desigualdades que evitan que aparezcan en la demostración estas instancias subsumidas. Una de las contrapositivas de la cláusula C_1 con esta modificación es la siguiente:

$$p(x, z) \leftarrow neq(x, z) \wedge neq([x, z], [a, b]) \wedge p(x, y) \wedge$$
$$neq(x, z) \wedge neq([x, z], [a, b]) \wedge neq([x, y], [b, c]) \wedge neq([y, z], [b, c]) \wedge p(y, z)$$
$$\wedge$$
$$neq(x, z) \wedge neq([x, z], [a, b]) \wedge neq([x, y], [b, c]) \wedge neq([y, z], [b, c])$$

El predicado $neq(t_1, t_2)$ es predefinido y falla inmediatamente si sus argumentos son sintácticamente idénticos.

Esta modificación corresponde a la siguiente regla obtenida de forma experimental por los autores: incluir las desigualdades de la subsumción[9] delante de cada submeta y detrás de la última a no ser que éstos contengan variables vírgenes. Claramente así se consigue que en el momento en el que se produzca una instanciación de las variables se realice la comprobación correspondiente.

3.6.4. IMPLEMENTACIÓN

Los autores han hecho especial hincapié en la implementación del sistema SETHEO. SETHEO está implementado como una máquina abstracta basada en registros. La máquina y sus instrucciones están basadas en la máquina abstracta de Warren. Para más detalles consultar [Letz, Schuman, Bayerl y Bibel, 1992].

Respecto a la unificación aplicada en el sistema SETHEO, los autores argumenta que la unificación completa es necesaria en raras ocasiones, por lo que han implementado diferentes

[9] También se incluyen las del tratamiento de la tautología, aunque en el ejemplo no se ha hecho.

instrucciones para la unificación teniendo en cuenta el tipo de término a unificar. De esta forma, únicamente se aplica la unificación completa cuando es necesario. Los algoritmos utilizados para la unificación completa son el de Robinson (exponencial) y el Corbin (cuadrático).

3.6.5. EXPERIMENTOS

Los resultados del conjunto de experimentos incluidos en el trabajo [Letz, Schuman, Bayerl y Bibel, 1992] son realmente buenos y su análisis permite una apreciación más profunda del funcionamiento del sistema SETHEO. Los datos incluidos corresponden a costes temporales y espaciales asociados a la resolución de un conjunto amplio de problemas clásicos de la demostración automática. Los costes temporales hacen referencia a la fase de preproceso, compilación y demostración de los problemas. Respecto al preproceso se puede observar que en ningún caso sobrepasa un segundo de duración y en una gran porcentaje de problemas se consigue una reducción del problema, e incluso la demostración del mismo. La información más interesante se extrae de los datos asociados a la demostración. Para cada problema tratado se han realizado pruebas sin podas y con ellas, y utilizando las tres formas de limitación de la profundidad en la búsqueda iterada. Al analizar todos estos datos, quizás, lo que quede más patente sea la necesidad del uso de las podas.

Finalmente, hay que indicar que los autores incluyen una escueta comparación con el demostrador PTTP, el demostrador más cercano al SETHEO, para un conjunto de problemas en la cual se pueden observar las diferencias en términos de resultados experimentales entre los dos demostradores. Los autores achacan estas diferencias, en el caso de los mejores resultados del SETHEO, a las tres forma de limitación de la profundidad usadas por éste (PTTP únicamente utiliza la limitación de la profundidad por número de pasos de inferencia ejecutados) y el tratamiento más adecuado de las tautologías. En el caso de los mejores resultados del PTTP, los autores dan como explicación el mejor ratio de inferencia y el uso de una reducción de la vuelta atrás más sofisticada.

3.6.6. CONCLUSIONES DEL SISTEMA SETHEO

El sistema SETHEO se puede afirmar que es uno de los más potentes y elaborados. Entre las características más destacables están en primer lugar el uso del preprocesamiento y la compilación en el tratamiento de los problemas a abordar para la reducción de los mismo y la obtención de información que en tiempo de ejecución será fundamental para la demostración de los problemas. En segundo lugar la construcción de una máquina abstracta para la implementación del procedimiento de prueba permite abordar problemas y obtener soluciones a los mismos que no están al alcance de otros sistemas (por ejemplo el PTTP). La tercera característica es la implementación de las podas, fundamentalmente el tratamiento de las

tautologías y la reducción por subsumción, que se aplican en el SETHEO. Estas podas son realizadas incluyendo código en tiempo de compilación dentro de las instrucciones que posteriormente en durante la ejecución y dinámicamente servirá para realizar las mencionadas podas. El autor de la presente tesis considera esta aproximación muy interesante al reducir el coste temporal de la aplicación de la poda de forma muy considerable.

CAPÍTULO 4:

4. EL PARADIGMA DE RESOLUCIÓN SL*

4.1. INTRODUCCIÓN

Como ya se comentó en los capítulos anteriores, existen numerosos procedimientos para la demostración automática en teorías clausales [Robinson, 1965] [Loveland, 1968] [Kowalski y Kuehner, 1971] [Plaisted, 1988] [Loveland, 1991] [Stickel, 1992]. Ninguno de estos procedimientos ha adquirido la difusión y utilización de los procedimientos de resolución SLD (para programas definidos) y SLDNF (para programas normales, interpretando la negación como fallo) dentro del contexto de la programación lógica. La principal característica de estos dos procedimientos que ha hecho que sean los más conocidos es su simplicidad. Por otra parte, esta misma simplicidad limita su posibilidad de aplicación a ciertas subclases de las teorías de primer orden. El objetivo que se plantea en el presente trabajo es definir un procedimiento de resolución, que manteniendo la simplicidad de los procedimientos de resolución SLD y SLDNF, sea aplicable a cualquier teoría clausal de forma correcta y completa.

Posiblemente el procedimiento más extendido sea resolución SLD para programas definidos (teorías Horn sin cláusulas negativas) [Kowalski, 1974] [Apt y van Emden, 1982] [Lloyd, 1987]. Existen dos extensiones bien conocidas de resolución SLD para programas que

no son definidos. La primera, que es ya un estándar en programación lógica, consiste en admitir átomos negados en el cuerpo de las cláusulas y procesarlos mediante la negación como fallo; esta extensión da lugar al procedimiento SLDNF [Clark, 1978] [Lloyd, 1987]. Claramente, SLDNF permite razonamiento no-monotónico, lo cual le da un gran potencial, pero orienta este procedimiento a otros campos del razonamiento automático. Por otro lado SLDNF no es completo para la demostración de teoremas en teorías clausales generales. La segunda extensión consiste en añadir al procedimiento SLD *resolución de ancestro* y posiblemente otras reglas de inferencia, dando lugar a sistemas como SLI [Minker y Zanon, 1982] [Minker y Rajasekar, 1990] y otros demostradores de teoremas. Todos ellos, incluso SLD, provienen de resolución SL [Kowalski y Kuehner, 1971] o directamente de Eliminación de Modelos [Loveland, 1968].

Esencialmente, resolución SL* consiste en añadir al procedimiento SLD un mecanismo de derivaciones subsidiarias que, técnicamente, es similar a la extensión del procedimiento SLD con negación como fallo. Las derivaciones SL* son ejecutadas en distintos rangos de acuerdo con una elección indeterminista que controla la utilización de ancestros. Tales elecciones pueden influir en el comportamiento de la resolución SL*, dando lugar a diferentes instancias adecuadas para distintos propósitos. Más adelante se verá que diferentes procedimientos, instancias de resolución SL*, corresponden a procedimientos ya conocidos.

4.2. DEFINICIÓN DE RESOLUCIÓN SL*

Antes de abordar la definición de resolución SL* son necesarios introducir algunos conceptos preliminares. Una *cláusula* es de la forma $H \leftarrow B$, donde la *cabeza* H es una disyunción $A_1 \vee \ldots \vee A_n$ (n≥0) de átomos A_n ($1 \leq i \leq n$) y el *cuerpo* B es una conjunción $D_1 \wedge \ldots \wedge D_m$ de átomos D_j ($1 \leq j \leq m$). Para m=0, el símbolo "←" puede ser obviado. Una cláusula con $n \leq 1$ se denomina *Horn*, con n=1 se denomina *definida*, con n=0 *negativa*, y con $m+n$=1 *unitaria*. La *cláusula vacía* ($m+n$=0) es denotada por □. Cada átomo en la cabeza (resp. el cuerpo) de una cláusula corresponde a un literal positivo (resp. negativo). Una *teoría* es un conjunto finito de cláusulas. Una teoría es *Horn* (resp. *definida*) si cada una de sus cláusulas es Horn (resp. definida). Una teoría es *de rango restringido* si cada una de las variables en la cabeza de cada cláusula de la teoría aparece también en el cuerpo.

En lo referente a las posibles respuestas que se pueden obtener, las siguientes definiciones previas son necesarias: Sea T una teoría y C una cláusula. Una *solución* (*respuesta*) para C en una teoría T es un conjunto de sustituciones $\Theta=\{\theta_1, \ldots, \theta_n\}$ (n>0) tal que $T \models \forall(\neg C\theta_1 \vee \ldots \vee \neg C\theta_n)$, donde $\forall$ denota el cierre universal. Para n=1, Θ es *definida*[10]; para n>1, Θ es

[10] En este caso se empleará indistintamente como respuesta el conjunto Θ o su único elemento, la sustitución θ.

indefinida. Θ es *concisa* si ningún subconjunto propio de Θ es una solución; en caso contrario, Θ es *redundante*. Θ es *base* si cada θ_i ($1 \leq i \leq n$) es una sustitución base.

Por otro lado, se tomarán por comodidad las siguientes convenciones: para una cláusula o conjunto de cláusulas S y una sustitución θ, $S\theta$ se obtiene aplicando θ a cada una de las variables de S. Además, si un literal L asume el papel de cláusula unitaria, entonces esa cláusula es, o bien $A\leftarrow$ si $L=A$, o bien $\leftarrow A$ si $L=\neg A$, donde A es un átomo.

La forma de la definición del paradigma de resolución lineal con selección conducida SL* es similar a la de la definición de la resolución SLD de [Lloyd, 1987] y sus extensiones para el proceso de negación como fallo de [Clark, 1978] [Lloyd, 1987] [Eshghi y Kowalski, 1989]. La definición de resolución SL*, así como su aplicación a varios campos del razonamiento automático, se puede encontrar en [Casamayor, Decker y Marqués, 1993] [Decker y Casamayor, 1993] [Casamayor y Decker, 1995].

La primera definición corresponde a una refutación SL* de rango 0. El rango de una refutación indica la profundidad de computaciones subsidiarias que ha necesitado. De esta forma, un rango 0 corresponde a una refutación que no ha utilizado ninguna computación subsidiaria. Una derivación subsidiaria se inicia cuando en la derivación de rango superior se selecciona un literal que se encuentra en la *elección de ancestros*. Este literal pasará a ser un nuevo ancestro para la derivación subsidiaria cuya cláusula raíz será este literal. Así, la decisión de si se inicia o no una derivación, es controlada por la elección de ancestros, atendiendo al literal seleccionado. La siguiente definición formaliza este nuevo concepto.

Definición 4.1 (Elección de ancestros)

Sea T una teoría. Una elección de ancestros para T, es un subconjunto (posiblemente vacío) del conjunto de todos los literales del lenguaje de primer orden subyacente a T.

A continuación se presenta la definición de refutación SL* de rango 0. Para hacer más compacta y legible la definición se asume que el literal seleccionado L en una cláusula de la forma $A_1 \vee \ldots \vee A_n \leftarrow D_1 \wedge \ldots \wedge D_m$ es o A_i ($1 \leq i \leq n$) o $\neg D_j$ ($1 \leq j \leq m$).

Definición 4.2 (Refutación SL* de rango 0)

Sea T una teoría, C una cláusula (*cláusula raíz*), *Anc* un conjunto de cláusulas (*ancestros*) y *Ch* una elección de ancestros. Una *refutación SL* desde C en T vía Ch usando Anc de rango* 0 se define como sigue: la refutación consiste en una secuencia de cláusulas $C_0,\ldots, C_n$, una secuencia de cláusulas $C'_1,\ldots,C'_n$ (*cláusulas padre-lejano*) y una secuencia de sustituciones $\theta_1,\ldots, \theta_n$ ($n \geq 0$), tal que $C_0=C$, $C_n=\square$ y, en cada cláusula C_i distinta de la cláusula vacía, un literal L_i es seleccionado tal que $L_i \notin Ch$ y C_{i+1} es un resolvente de C_i y C'_{i+1} sobre L_i y un literal L de C'_{i+1} usando el umg θ_{i+1} de los átomos de L_i y L, siendo la cláusula padre-lejano C'_{i+1}

una variante fresca de una cláusula de T o una cláusula de Anc_i, siendo $Anc_0=Anc$ y $Anc_i=Anc\theta_1\ldots\theta_i$.

De esta definición se pueden destacar varios puntos: la única regla de inferencia utilizada es la resolución binaria entre la cláusula actual y una cláusula padre lejano que proviene de la teoría inicial T o del conjunto de ancestros instanciados por la sustitución actual. Además, cada Anc_j es una instancia de Anc_i ($j>i$), esto es, ningún nuevo ancestro es añadido al conjunto inicial. Se destaca que la exigencia de que cada uno de los literales seleccionados no pertenezca a la elección de ancestros es consecuencia del rango de la refutación, rango 0, ya que si el literal perteneciera a la elección de ancestros una derivación subsidiaria se iniciaría sobre él y el rango no podría ser 0. Por otro lado, si el conjunto de ancestros Anc es vacío, una refutación SL* de rango 0 es idéntica a una refutación de resolución de entrada.

Definición 4.3 (Refutación SL* de rango k)

Sea T una teoría, C una cláusula, Anc un conjunto de cláusulas y Ch una elección de ancestros. Una *refutación desde C en T vía Ch usando Anc de rango k* se define como sigue: La refutación consiste en una secuencia de cláusulas $C_0,\ldots, C_n$, una secuencia de cláusulas padre-lejano $C'_1,\ldots, C'_n$ una secuencia de $\theta_1,\ldots, \theta_n$ de substituciones ($n\geq 0$), tal que $C_0=C$, $C_n=\square$ y, en cada cláusula C_i distinta de la cláusula vacía, un literal L_i es seleccionado tal que, para $Anc_0=Anc$ y $Anc_i=Anc\theta_1\ldots\theta_i$, los siguientes puntos se cumplen:

i) Si $L_i \notin Ch$ o $L_i \in Anc_i$, entonces C_{i+1} es un resolvente de C_i y C'_{i+1} sobre L_i y un literal L' de C'_{i+1} usando el umg θ_{i+1} de los átomos de L_i y L', cumpliéndose que o bien la cláusula padre-lejano C'_{i+1} es una variante fresca de una cláusula de T o bien $C'_{i+1} \in Anc_i$.

ii) Si $L_i \in Ch$ y $L_i \notin Anc_i$, entonces existe una refutación SL* desde L_i en T vía Ch usando $Anc_i \cup \{L_i\}$, de rango menor que k, con sustitución computada θ_{i+1}, y C_{i+1} es el resolvente de C_i y C'_{i+1} tal que la cláusula padre-lejano C'_{i+1} es igual a $\widetilde{L}_i\theta_{i+1}$ donde $\widetilde{L}_i$ es el literal complementario de L_i.

Además existe un cierto i, $0\leq i\leq n$, tal que se cumple el punto ii) y la refutación SL* desde L_i en T vía Ch usando $Anc_i \cup \{L_i\}$ es de rango k-1.

La composición de $\theta_1\ldots\theta_n$ se denomina *substitución computada* de la refutación. También se denomina a n la *longitud* de la refutación.

Definición 4.4 (Refutación SL*)

Sea T una teoría, C una cláusula, Anc un conjunto de cláusulas y Ch una elección de ancestros. Una *refutación SL* desde C en T vía Ch usando Anc* es una refutación SL* desde C en T vía Ch usando S de rango k, para algún k.

En la definición de refutación SL* se puede observar que los objetos que se manejan son los siguientes: la teoría *T*, la cláusula raíz *C*, el conjunto de ancestros *Anc* y la elección de ancestros *Ch*. Para clarificar el funcionamiento de resolución SL*, conviene introducir la forma habitual de estos objetos en la refutación de mayor rango:

- La cláusula raíz *C* puede ser cualquier tipo de cláusula y desde ella se inicia la computación mediante resolución SL*. Habitualmente, esta cláusula constituirá la negación del teorema a demostrar o de la pregunta sobre la cual se desean obtener posibles soluciones. La única decisión a tomar sobre *C* es si se incluye en la teoría *T* o en el conjunto *Anc*, de forma excluyente. Si *C* se incluye en *T* (y por tanto no se incluye en *Anc*) durante la computación será posible utilizar variantes frescas de la cláusula raíz. Al contrario, si *C* se incluye en *Anc* (y por tanto no se incluye en *T*), no será posible utilizar variantes frescas de *C*, aunque si que se podrá utilizar *C* instanciada por la sustitución actual en la computación como cláusula padre-lejano. Cada una de estas dos alternativas corresponden a distintas formas de razonamiento automático, como se comentará más adelante.

- La teoría *T* está formada por la cláusulas que constituyen el conocimiento sobre el cual se desea comprobar si la cláusula inicial *C* está relacionada de alguna forma. Siempre se usan variantes frescas de cláusulas de *T* como cláusulas padre-lejano en la computación.

- El conjunto de ancestros *Anc* se definió genéricamente como un conjunto de cláusulas. Esto puede llevar a confusión ya que los ancestros siempre han correspondido a los literales seleccionados hasta el punto actual en la computación. En la resolución SL* este hecho sigue siendo cierto con la única excepción de que la cláusula inicial *C* puede estar en el conjunto de ancestros *Anc* en algunos procedimientos instancias de resolución SL*. Así, en la derivación inicial el conjunto *Anc* puede ser o el conjunto vacío o contener como único elemento la cláusula inicial *C*. Si inicialmente *Anc* es el conjunto vacío, entonces las derivaciones subsidiarias que se utilicen tendrán como conjunto de ancestros únicamente los literales previamente seleccionados. Por otro lado, es interesante destacar que cuando un elemento de *Anc* es utilizado como cláusula padre-lejano, lo es tal y como aparece, esto es, no se usa una variante fresca de él.

- La elección de ancestros *Ch* controla la aplicación del paso ii) de la definición. Este paso es el que posibilita la llamada a computaciones subsidiarias de rango inferior. Aunque *Ch* puede corresponder a cualquier conjunto de literales, posteriormente se verá que solamente algunas elecciones dan lugar a procedimientos instancias de resolución SL* que son completos.

Como ya se ha comentado, la distintas decisiones asociadas a estos objetos dan lugar a instancias de resolución SL* que son procedimientos adecuados para diferentes propósitos dentro del campo del razonamiento automático. Posteriormente se verán los más significativos con detalle.

Si se analiza la definición de resolución SL*, se puede observar que las dos reglas de inferencia que pueden ser usadas en SL* son *resolución binaria* y la *regla de la división*. La aplicación de resolución binaria corresponde al punto i) de la definición y de la regla de división al punto ii).

En el punto i) que corresponde a la aplicación de un paso de resolución binaria es posible observar que esta aplicación se realiza entre la cláusula en curso en la derivación y una cláusula padre-lejano que pertenece a la teoría actual para la derivación, que corresponde a la teoría inicial *T* junto con el conjunto de ancestros para esa derivación. Dependiendo de dónde se extrae la cláusula padre-lejano, pueden ocurrir dos casos: primero se usa una variante fresca de una cláusula de la teoría inicial *T*, entonces este paso se corresponde con un paso de resolución de entrada [Chang y Lee, 1973] (o a una *extensión*, usando la terminología de Eliminación de Modelos [Loveland, 1968]); segundo, la cláusula padre-lejano es un ancestro del conjunto Anc, entonces este paso se corresponde con un paso de resolución de ancestro [Chang y Lee, 1973] (o a una *reducción*, según la terminología de Eliminación de Modelos y se corresponde con una aplicación del principio de reducción al absurdo como se presentó en el formato MESON en el capítulo 2).

La aplicación de la regla de división, en general, puede explicarse como sigue: supóngase que *T* es una teoría consistente y *C* es una cláusula de la forma $L_1 \vee \ldots \vee L_n$ (que por simplicidad se supondrá libre de variables). El problema de determinar la consistencia de $T \cup \{C\}$ puede ser reducido (o divido a) dos subproblemas: determinar la inconsistencia de $T \cup \{L_i\}$ y la inconsistencia de $T \cup \{L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \vee \ldots \vee L_n\}$[11]. En el caso en que *C* contenga variables, la aplicación de la regla anterior debe realizarse teniendo en cuenta las variables comunes de L_i y $L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \vee \ldots \vee L_n$ para garantizar la corrección. Así, esta misma regla es aplicada en resolución SL* en el punto ii) como sigue: la comprobación de la inconsistencia de C_i (la cláusula en curso) con la teoría en curso (la teoría inicial *T* y el conjunto de ancestros en curso Anc_i) se realiza demostrando, en primer lugar, la inconsistencia del literal seleccionado L_i con la teoría en curso y, en segundo lugar, la inconsistencia de C_{i+1} (el resto de C_i) con la teoría en curso. La demostración de la inconsistencia de L_i con la teoría en curso se efectúa mediante una refutación subsidiaria de rango inferior, para la cual L_i es incorporado al conjunto de ancestros.

[11] En general, la división de la cláusula inicial puede hacerse eligiendo cualquiera dos subcláusulas tal que su unión dé lugar a la cláusula inicial.

La incorporación de L_i al conjunto de ancestros inhibe el uso de variantes frescas de dicho literal en la refutación subsidiaria, lo cual garantiza que la negación del literal instanciado por la sustitución computada de la refutación subsidiaria, es decir $\widetilde{L}_i\,\theta_{i+1}$, es consecuencia lógica de la teoría en curso. Este hecho es muy importante, ya que garantiza la demostración de la inconsistencia del resto del problema, es decir C_{i+1} (que se obtiene como resolvente de C_i y $\widetilde{L}_i\,\theta_{i+1}$), por medio de la refutación que llamó a la refutación subsidiaria es correcto.

Por otra parte, hay que destacar que la paralelización es otra de las posibles formas de aplicación de la regla de división que no se usa en resolución SL*. La paralelización de la regla de la división se puede efectuar ejecutando en paralelo tanto la computación subsidiaria como la de rango superior que llamó a ésta. Si no existen variables compartidas entre el literal seleccionado en la cláusula en curso, raíz de la computación subsidiaria, y el resto de la cláusula en curso se puede realizar ejecución completamente en paralelo de las dos computaciones. Si existen variables compartidas esta ejecución en paralelo ha de realizarse de manera coherente, actuando las variables compartidas como puntos de control entre ellas. Esta aproximación de aplicación de la regla de división no ha sido abordada en resolución SL*, aunque se considera que tiene el suficiente interés como para retormarla en estudios posteriores.

Las condiciones que aparecen en los puntos i) y ii) de la definición aseguran que la aplicación de la regla de división (y por tanto, el uso de computaciones subsidiarias) es controlado por la elección de ancestros *Ch*. Únicamente se puede llamar a una computación subsidiaria cuando el literal seleccionado L_i se encuentra en la elección de ancestro *Ch*. Por tanto, ya que los ancestros en resolución SL* se corresponden con las raíces de los derivaciones activas, se deduce que el espacio de búsqueda asociado a los ancestros es controlado por *Ch*. En las condiciones de estos dos puntos además se comprueba la pertenencia del literal seleccionado al conjunto de ancestros, asegurando de esta forma que la regla de división no va a ser aplicada de forma infinita e innecesariamente en los pasos iniciales de derivaciones subsidiarias. Así, únicamente es posible aplicar una resolución binaria en estos pasos.

El último punto necesita un estudio más profundo para comprender cual es la influencia que tiene la elección de ancestros *Ch* en el comportamiento de la resolución SL*. De la definición de resolución SL* se puede observar que la aplicación de un paso de la regla de división está controlada por la pertenencia o no del literal seleccionado a la elección de ancestros *Ch*. Por ejemplo, si la elección de ancestros es el conjunto vacío, entonces nunca se aplicará la regla de división, aplicándose siempre, por tanto, pasos de resolución binaria. Por tanto, la elección de ancestros puede entenderse como un mecanismo de control similar, de algún modo, a la función de selección (mediante la cual se determina el literal seleccionado en la cláusula en curso). Sin embargo, a diferencia de la función de selección, no toda elección de ancestros garantiza la completitud del correspondiente procedimiento instancia de resolución

SL*. Por todo lo comentado se puede deducir que la forma de obtener un procedimiento instancia de resolución SL* es determinando lo siguiente:

a) si la cláusula raíz C pertenece a la teoría inicial T o al conjunto de ancestros Anc (o sea, si $C \in T$ y $Anc=\emptyset$ o $C \notin T$ y $Anc=\{C\}$).

b) qué literales forman la elección de ancestros Ch.

Como ejemplos de procedimientos instancias de resolución SL* cabe destacar los siguientes:

• T es una teoría definida, C es una cláusula negativa que no pertenece a T, $Anc=\emptyset$ y $Ch=\emptyset$. Así, la correspondiente instancia de SL* es el procedimiento SLD.

• T es una teoría, C es una cláusula que pertenece a T, $Anc=\emptyset$ y $Ch=\emptyset$. Así, el procedimiento instancia de SL* corresponde a resolución de entrada.

En los siguientes dos ejemplos se presentan dos refutaciones SL* vía diferentes elecciones de ancestros, clarificando la influencia de la elección de ancestros en la resolución SL*.

Ejemplo 4.1 (Refutación SL* vía ALL)

Sea T la teoría formada por las siguientes cláusulas:

$p(x) \leftarrow q(x)$	1
$p(x) \leftarrow r(x)$	2
$q(a) \vee r(a) \leftarrow$	3
$s(a) \leftarrow$	4
$\leftarrow p(x) \wedge s(x)$	5

y ALL la elección de ancestros formada por todos los literales del lenguaje subyacente. La siguiente figura muestra una refutación SL* desde $\leftarrow p(x) \wedge s(x)$ en T vía ALL usando $\emptyset$

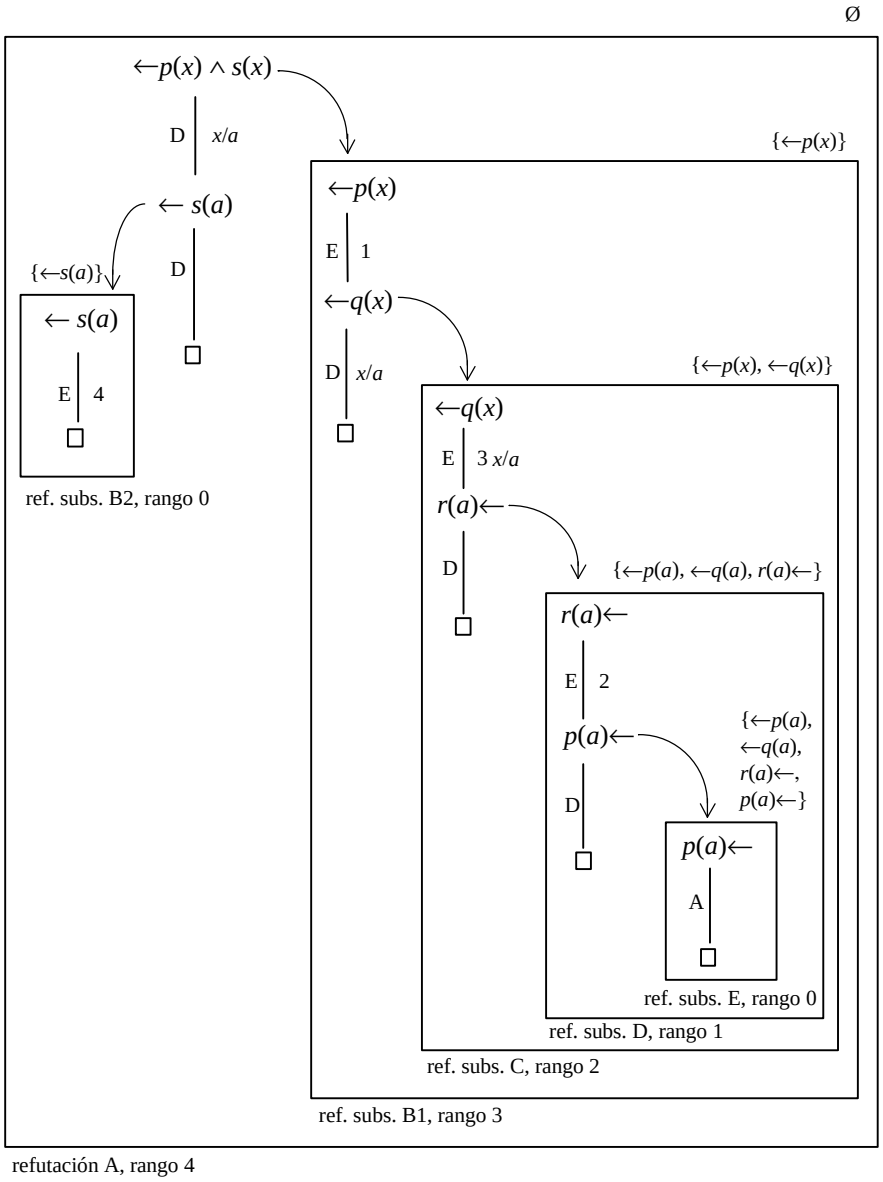


Figura 4.1: Refutación SL* vía ALL para el Ejemplo 4.1

Los diferentes reglas de inferencias que se aplican en resolución SL* se denotan como sigue: resolución de entrada (E), resolución de ancestro (A) y regla de división (D). Cada arco que une una cláusula en curso con su inmediata sucesora está etiquetado por la regla de inferencia aplicada, el número de cláusula utilizada si se aplica resolución de entrada y el umg asociado cuando es significativo. El literal seleccionado en la cláusula en curso se subraya cuando es necesario. Cada rectángulo representa una refutación de un cierto rango. El rectángulo más externo representa la refutación de rango superior y los internos las refutaciones subsidiarias. Además, en la parte superior sobre cada rectángulo se ha incluido el conjunto de ancestros inicial para cada refutación, en la parte inferior izquierda debajo de cada rectángulo se ha asociado un identificador a cada refutación, mostrando también el rango. Las refutaciones que aparecen en la figura son las siguientes:

• refutación A: refutación SL* desde $\leftarrow p(x) \wedge s(x)$ en T vía ALL usando $\varnothing$ de rango 4 con sustitución computada $\{x/a\}$.

• refutación subsidiaria B1: refutación SL* desde $\leftarrow p(x)$ en T vía ALL usando $\{\leftarrow p(x)\}$ de rango 3 con sustitución computada $\{x/a\}$.

• refutación subsidiaria B2: refutación SL* desde $\leftarrow s(a)$ en T vía ALL usando $\{\leftarrow s(a)\}$ de rango 0 con sustitución computada la sustitución identidad.

• refutación subsidiaria C: refutación SL* desde $\leftarrow q(x)$ en T vía ALL usando $\{\leftarrow p(x), \leftarrow q(x)\}$ de rango 2 con sustitución computada $\{x/a\}$.

• refutación subsidiaria D: refutación SL* desde $r(a)\leftarrow$ en T vía ALL usando $\{\leftarrow p(a), \leftarrow q(a), r(a)\leftarrow\}$ de rango 1 con sustitución computada la sustitución identidad.

• refutación subsidiaria E: refutación SL* desde $p(a)\leftarrow$ en T vía ALL usando $\{\leftarrow p(a), \leftarrow q(a), r(a)\leftarrow, p(a)\leftarrow\}$ de rango 0 con sustitución computada la sustitución identidad. Esta refutación, en su único paso, realiza un resolución de ancestro con el ancestro $\leftarrow p(a)$.

Es interesante destacar que la cláusula raíz, incluida en la teoría inicial, no ha sido utilizada en esta refutación como cláusula de entrada. Por otro lado, también hay que darse cuenta de que cuando la regla de la división es aplicada, encontrándose una refutación subsidiaria para el literal seleccionado, en la refutación de rango superior se usa el hecho de que la negación del literal seleccionado instanciado por la sustitución computada es consecuencia lógica de la teoría en curso para resolver y continuar con la derivación. Este hecho no se ha indicado en la figura por resultar evidente.

Si se presta atención a la refutación A en el paso 0, el literal $\neg p(x)$ es seleccionado en la cláusula en curso $\leftarrow p(x) \wedge s(x)$ (la cláusula raíz) y como pertenece a la elección de ancestros (ya que ésta contiene a todos los literales del lenguaje) se divide el problema de comprobar la inconsistencia de esta cláusula con la teoría en curso (la teoría inicial, ya que el conjunto de ancestros es vacío), en comprobar la inconsistencia de $\leftarrow p(x)$ y la del resto de la cláusula, es decir $\leftarrow s(x)$. La comprobación de la inconsistencia de $\leftarrow p(x)$ es efectuada por la refutación subsidiaria B1 que devuelve como resultado la sustitución computada $\{x/a\}$. Esta refutación B1 asegura que la negación de literal instanciado por la sustitución computada, $p(a)$, es consecuencia lógica de la teoría en curso. Por tanto, usando este hecho, el resto del problema que queda por refutar es $\leftarrow s(x)$ instanciado por la sustitución computada en la refutación subsidiaria anterior, esto es $\leftarrow s(a)$. Esta forma de aplicación de la regla de la división cuando la cláusula sobre la cual se aplica contiene variables es correcta, como se probará más adelante. En el paso 1 de la refutación A, la cláusula en curso es $\leftarrow s(a)$, aplicándose de nuevo la regla de división ya que el literal seleccionado $\neg s(a)$ pertenece a la elección de ancestros. Esta aplicación

podría parecer sorprendente a primera vista ya que la cláusula actual sólo contiene un literal. Sin embargo, la lectura que se puede hacer de esta aplicación es que el problema de comprobar la inconsistencia de $\leftarrow s(a)$ se divide en la comprobación de la inconsistencia de dos subcláusulas $\leftarrow s(a)$, correspondiente al literal seleccionado, y la de la cláusula vacía. La inconsistencia de la cláusula vacía se demuestra de forma inmediata. Por tanto, basta con demostrar la inconsistencia de $\leftarrow s(a)$ mediante una refutación subsidiaria (la refutación B1). Esta forma de proceder puede parecer poco eficiente, pero hay que recordar que el llamar a una derivación subsidiaria desde un cierto literal seleccionado tiene una implicación que no se da si se resuelve en la derivación de rango superior: este literal es incorporado a la lista de ancestros. Así, ese literal puede ser utilizado en pasos de resolución de ancestro en la derivación subsidiaria, lo que no es posible en la derivación de rango superior.

El siguiente ejemplo presenta una refutación SL* para la misma teoría y la misma cláusula raíz, pero ahora la elección de ancestros es el conjunto vacío.

Ejemplo 4.2 (Refutación SL* vía EMPTY)

Sea T la teoría formada por las siguientes cláusulas:

$p(x) \leftarrow q(x)$	1	$s(a) \leftarrow$	4
$p(x) \leftarrow r(x)$	2	$\leftarrow p(x) \wedge s(x)$	5
$q(a) \vee r(a) \leftarrow$	3		

y EMPTY, el conjunto vacío, la elección de ancestros. La siguiente figura muestra una refutación SL* desde $\leftarrow p(x) \wedge s(x)$ en T vía EMPTY usando $\varnothing$

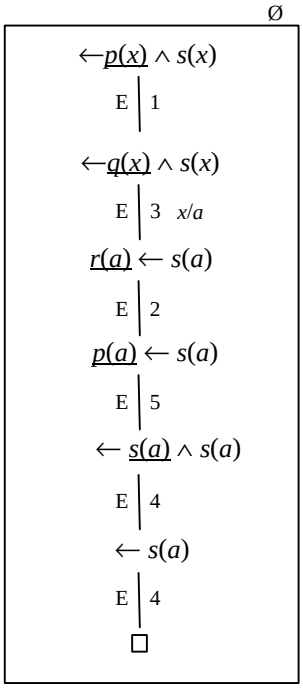


Figura 4.2: Refutación SL* vía EMPTY para el Ejemplo 4.2

La figura muestra un refutación SL* desde $\leftarrow p(x) \wedge s(x)$ en T vía EMPTY usando $\varnothing$ de rango 0 con sustitución computada $\{x/a\}$. Obviamente, el rango de esta refutación es 0 ya que la elección de ancestros es EMPTY. Como ya se comentó, el rango denota el anidamiento máximo de las refutaciones subsidiarias utilizadas por una cierta refutación. Ya que las derivaciones subsidiarias se llaman desde los literales seleccionados que se encuentran en la elección de ancestros, si esta última es EMPTY no se llamará a ninguna derivación subsidiaria. Por tanto, el rango de toda refutación SL* vía EMPTY es 0. Por otra parte, también hay que destacar que la única regla de inferencia utilizada en la refutación es la resolución de entrada. Este hecho es consecuencia de dos motivos: primero, la regla de división no se puede aplicar, por las razones que se acaban de explicar; segundo, la resolución de ancestro tampoco es aplicable ya que el conjunto de ancestros inicial es el conjunto vacío y la única forma de incrementar este conjunto es cuando se efectúa un paso de regla de división (o sea, cuando se llama a una computación subsidiaria). Por tanto, se puede observar que esta refutación SL* desde $\leftarrow p(x) \wedge s(x)$ en T vía EMPTY usando $\varnothing$ es similar a un refutación de resolución de entrada desde $\leftarrow p(x) \wedge s(x)$ en T.

El tamaño de las dos refutaciones presentadas en las dos figuras podría llevar a engaño y pensar que resolución SL* vía EMPTY da lugar a refutaciones más sencillas que resolución SL* vía ALL. En general esto no es cierto. Como contraejemplo, piénsese que en vez de existir la cláusula unitaria $s(a)\leftarrow$ en la teoría, fuera posible inferir que $s(a)$ es consecuencia lógica de la teoría pero mediante un refutación de gran longitud. En este caso, en la refutación SL* vía ALL anterior sólo habría que substituir está refutación por la refutación subsidiaria desde $\leftarrow s(a)$ en T vía ALL usando $\{\leftarrow s(a)\}$ (refutación B1). Sin embargo, en la refutación SL* vía EMPTY habría que substituir esta refutación dos veces por los por pasos de resolución de entrada últimos, dando lugar a una refutación más compleja[12]. Más adelante se volverá a retomar la discusión sobre la eficiencia y complejidad de las diferentes instancias de resolución SL* según la elección de ancestros utilizada.

El siguiente resultado expresa la corrección de la resolución SL*.

Teorema 4.1 (Corrección de SL* resolución)

Sea T una teoría, C una cláusula de la forma $H\leftarrow B$, Anc un conjunto de cláusulas y Ch una elección de ancestros. Si existe una refutación SL* desde C en T vía Ch usando Anc con substitución computada θ, entonces $T \cup Anc\theta \models \forall((\neg H \wedge B)\theta)$[13].

[12] Se podría argumentar acerca de la posibilidad de utilizar la fusión (o en general factorización) en la cláusula en curso $\leftarrow s(a) \wedge s(a)$ en esta refutación para evitar este problema. Evidentemente, esto conllevaría la utilización de otra regla de inferencia, introduciendo una nueva problemática.

[13] Las demostraciones de todos los resultados dados en la presente tesis se pueden encontrar en el Apéndice A.

El resultado de completitud del SL* no se enuncia ya que es dependiente de cuál sea el procedimiento instancia que se defina a base de establecer cuál es la teoría inicial y cuál es la elección de ancestros utilizada. Esto se ilustra en los siguientes dos ejemplos:

Ejemplo 4.3 (Necesidad del uso de la cláusula inicial en la demostración de teoremas)

Sea T la teoría $\{p(x)\leftarrow q(x),\ p(x)\leftarrow r(x),\ q(a) \vee r(b)\leftarrow\}$. Es fácil comprobar que el conjunto formado por la cláusula $\leftarrow p(x)$ y T es inconsistente. Sin embargo, no existe una refutación SL* desde $\leftarrow p(x)$ en T vía cualquier elección de ancestros usando $\{\leftarrow p(x)\}$. Claramente, el problema estriba en la ausencia de la cláusula raíz en la teoría inicial, lo que impide la reutilización de variantes frescas de ella. Esto hecho es bien conocido, ya que se sabe que para determinar la inconsistencia de un conjunto de cláusulas usando resolución lineal, la cláusula inicial puede ser usada varias veces (en cada una de ellas se usa una variante fresca).

Ejemplo 4.4 (Incompletitud de la elección de ancestros EMPTY)

Sea T la teoría $\{p \leftarrow q,\ q \leftarrow p,\ p \vee q\leftarrow\}$. También resulta obvio que $T \cup \{\leftarrow p \wedge q\}$ es inconsistente. Sin embargo, no existe una refutación SL* desde $\leftarrow p \wedge q$ en $T \cup \{\leftarrow p \wedge q\}$ vía EMPTY usando $\varnothing$. Esto no ha de extrañar ya que, como se mencionó antes, resolución SL* vía EMPTY es idéntica a resolución de entrada, y es bien conocido que resolución de entrada no es completa para teorías clausales.

Como se acaba de mostrar en el anterior ejemplo la elección de ancestros es determinante en el comportamiento del procedimiento correspondiente, instancia de resolución SL*. Es más, incluso determina si el procedimiento es completo o no. La siguiente definición determina la condición para que una elección de ancestros garantice la completitud del procedimiento instancia de resolución SL*.

Definición 4.5 (Elección de ancestros homogénea)

Sea T una teoría y Ch una elección de ancestros. Ch es una *elección de ancestros homogénea* para T si y sólo si para todo literal $L \in Ch$ se cumple que cualquier generalización de L pertenece a Ch.

La propiedad de homogeneidad de la elección de ancestros asegura que el uso de las computaciones subsidiarias es independiente de la función de selección utilizada. De esta forma si al utilizar una función de selección se elige un literal en una derivación y se llama sobre él a una computación subsidiaria, al estar éste en la elección de ancestros, entonces al cambiar de función de selección, llegado el punto donde se elige a este literal o una instancia del mismo también se llamará sobre él a una computación subsidiaria. Por otra parte esta propiedad permitirá más delante establecer el resultado que permitirá realizar el levantamiento del caso base al caso general.

Definición 4.6 (Elección de ancestros completa)

Sea *T* una teoría y *Ch* una elección de ancestros. *Ch* es una *elección de ancestros completa para T* si *Ch* es homogénea y para cada átomo *A* del lenguaje subyacente a *T*, o bien dicho átomo o su negación o ambos pertenecen a *Ch*.

La propiedad de completitud dada en la definición anterior, exige la presencia bien del átomo bien de su negación para todo átomo del lenguaje subyacente a la teoría. Como ejemplos de elecciones de ancestros completas se pueden destacar las siguientes:

- ALL: formada por todos los literales del lenguaje subyacente.
- POS: formada por todos los literales positivos del lenguaje subyacente.
- NEG: formada por todos los literales negativos del lenguaje subyacente.

Otra elección que también merece ser destacada, aunque no es una elección de ancestros completa es el conjunto vacío, que se ha denominado anteriormente EMPTY. Esta elección de ancestros es interesante ya que el procedimiento instancia resultante corresponde a resolución de entrada.

Se va a pasar ahora a dar un conjunto de definiciones técnicas para poder enunciar resultados posteriores asociados a resolución SL*.

La siguiente definición se da por comodidad para las definiciones posteriores de derivación SL* y de árbol SL*.

Definición 4.7 (Paso de derivación SL*)

Sea *T* una teoría, *Anc* un conjunto de cláusulas, *C*, *C'* y *C''* tres cláusulas, *L* un literal seleccionado en *C*, θ una substitución y *Ch* una elección de ancestros. Se dice que *C'' es derivada en un paso SL* desde C en T vía Ch usando Anc (sobre L, utilizando la cláusula padre lejano C' y la substitución* θ) si *C''* es un resolvente de *C* y *C'* sobre *L* y algún literal *L'* en *C'* y además:

- Si $L \notin Ch$ o $L \in Anc$, entonces *C'* es o una variante fresca de una cláusula de *T* o $C' \in Anc$, y θ es el umg de los átomos de *L* y *L'*.
- Si $L \in Ch$ y $L \notin Anc$, entonces existe una refutación SL* desde *L* en *T* vía *Ch* usando $Anc \cup \{L\}$ con sustitución computada θ y *C'* es $\tilde{L}\theta$, donde $\tilde{L}$ es el literal complementario de *L*.

Ahora ya se está en disposición de dar la definición de derivación SL* y de árbol SL*.

Definición 4.8 (Derivación SL*)

Sea T una teoría, C una cláusula, Anc un conjunto de cláusulas y Ch una elección de ancestros. Una *derivación SL* desde C en T vía Ch usando Anc* se define como sigue: La derivación consiste de una secuencia de cláusulas C_0, C_1, … y una secuencia de sustituciones θ_1, θ_2, … de, tal que $C_0=C$, $Anc_0=Anc$, $Anc_i=Anc\theta_1\ldots\theta_i$ ($i\geq 1$) satisfaciendo los siguientes puntos:

- para cada i, C_{i+1} ($i\geq 0$) es derivado en un paso de derivación SL* desde C_i en T vía Ch usando Anc_i sobre un literal L_i utilizando la sustitución θ_{i+1}.

- Si la secuencia C_0, C_1, … de cláusulas es finita, entonces o bien:
  (i) la última cláusula es vacía, o
  (ii) el literal L_i seleccionado en la última cláusula, no pertenece a Ch o pertenece a Anc_i, y ninguna cláusula (variante fresca) de T ni ningún elemento de Anc_i puede ser resuelto con esta cláusula sobre L_i, o
  (iii) el literal L_i seleccionado en la última cláusula pertenece a Ch, no pertenece a Anc_i y no existe una refutación SL* desde L_i en T usando $Anc_i \cup \{L_i\}$.

Una derivación SL* cuya última cláusula es la cláusula vacía se denomina *derivación con éxito*. Una derivación SL* con éxito es una refutación SL*. Es necesario destacar que una derivación SL* con una secuencia de cláusulas finitas no implica un computación finita. Esto es debido al apartado (iii) del último punto, ya que la computación subsidiaria desde A_i puede no tener fin.

Definición 4.9 (Árbol SL*)

Sea T una teoría, C una cláusula, Anc un conjunto de cláusulas y Ch una elección de ancestros. Un *árbol SL* desde C en T vía Ch usando Anc* se define como sigue:

- cada nodo del árbol es una (posiblemente vacía) cláusula y su raíz es C.
- cada arco del árbol está etiquetado con un sustitución. Para el nodo raíz C, θ_C denota la substitución identidad. Para cualquier otro nodo C' del árbol, $\theta_{C'}$ denota la composición de las sustituciones que etiquetan la secuencia de arcos en el camino desde C hasta C'.
- Sea C' un nodo intermedio. Supóngase que L es el literal seleccionado en C', entonces cada cláusula derivada en un paso SL* desde C' en T vía Ch usando $Anc\theta_{C'}$ sobre L utilizando la substitución θ, es un nodo hijo de C'. El arco desde el nodo C' hacia el nodo hijo es etiquetado con θ.
- Sea C' un nodo terminal. Supóngase que L es el literal seleccionado en C'. Entonces o bien:
  (i) L no pertenece a Ch o pertenece a $Anc\theta_{C'}$, y ninguna cláusula (variante fresca) de T ni ningún elemento de $Anc\theta_{C'}$ puede ser resuelto con C' sobre L, o

(ii) L pertenece a Ch, no pertenece a $Anc\theta_{C'}$ y no existe una refutación SL* desde L en T vía Ch usando $Anc\theta_{C'} \cup \{L\}$.

- Los nodos que son la cláusula vacía no tienen hijos.

De la misma forma que para derivaciones SL* con secuencia finita de cláusulas, un árbol SL* finito no implica que el espacio de búsqueda representado por el árbol sea finito, ya que las computaciones subsidiarias pueden no tener fin.

A continuación se va a estudiar los diferentes procedimientos, instancias de resolución SL*, que se pueden obtener determinando la teoría a utilizar, el conjunto de ancestros inicial y la elección de ancestros. Estos procedimientos serán adecuados para distintos tipos de problemas y se incluirán los resultados obtenidos para ellos.

4.3. DISTINTAS INSTANCIAS DE RESOLUCIÓN SL*

Como ya se comentó anteriormente es posible obtener diferentes procedimientos instancias de resolución SL* a base de establecer tres parámetros:

a) La teoría inicial de la cual se toman las cláusulas de entrada. Esencialmente, hay que decidir cuál es el tipo de teoría a utilizar (definida, Horn, etc.) y si la cláusula raíz está incluida o no en ella.

b) El conjunto inicial de ancestros, decidiendo si es el conjunto vacío o si contiene como único elemento a la cláusula raíz. En caso de tomar la segunda alternativa, la teoría inicial no debe contener la cláusula raíz, ya que entonces es intranscendente que esta cláusula esté incluida en el conjunto inicial de ancestros.

c) La elección de ancestros que se va a utilizar. Ya se comentó que la elección de ancestros influye de una forma radical en el comportamiento de resolución SL*, determinando incluso la completitud de resolución SL* para determinados problemas.

En el siguiente apartado se van a presentar varios procedimientos, instancias de resolución SL*, que son adecuados para diferentes propósitos: demostración automática de teoremas, comprobación de la inconsistencia, resolución de problemas, obtención de respuestas definidas e indefinidas. Estos procedimientos se obtienen decidiendo los puntos a) y b) anteriores, esto es, cuál es la teoría inicial y cuál es el conjunto de ancestros inicial.

4.3.1. DEMOSTRACIÓN DE TEOREMAS Y RESOLUCIÓN DE PROBLEMAS

La genericidad del conjunto *Anc* en la definición de resolución SL* es conveniente para, por un lado, capturar el aspecto dinámico del conjunto de ancestros que crece con la profundidad de los rangos (pero no con la longitud de las derivaciones), y, por otro lado, dejar abierta la decisión de si la raíz inicial es un ancestro o no inicialmente. Las versiones de resolución SL* para demostración de teoremas (o comprobación de la inconsistencia) y para la resolución de problemas definidos (o la obtención de respuestas definidas), como se define posteriormente, se distinguen por el uso de la raíz inicial bien como un ancestro bien como un cláusula de entrada.

Definición 4.10 (Refutación, derivación y árbol SLP)

Sea *T* una teoría, *C* una cláusula y *Ch* una elección de ancestros.

a) Una *refutación SLP desde C en T* (*vía Ch*) es una refutación SL* desde *C* en *T* vía *Ch* usando {*C*}.
b) Una *derivación SLP desde C en T* (*vía Ch*) es una derivación SL* desde *C* en *T* vía *Ch* usando {*C*}.
c) Un *árbol SLP desde C en T* (*vía Ch*) es un árbol SL* desde *C* en *T* vía *Ch* usando {*C*}.

Definición 4.11 (Refutación, derivación y árbol SLT)

Sea *T* una teoría, *C* una cláusula y *Ch* una elección de ancestros.

a) Una *refutación SLT desde C en T* (*vía Ch*) es una refutación SL* desde *C* en $T \cup \{C\}$ vía Ch usando $\varnothing$.
b) Una *derivación SLT desde C en T* (*vía Ch*) es una derivación SL* desde *C* en $T \cup \{C\}$ vía Ch usando $\varnothing$.
c) Un *árbol SLT desde C en T* (*vía* Ch) es un árbol SL* desde *C* en $T \cup \{C\}$ vía Ch usando $\varnothing$.

(Se omite "vía *Ch*" si no importa vía qué elección de ancestros se obtiene una derivación)

En SLT, la cláusula raíz *C* está incluida en el conjunto de cláusula de entrada *T*, mientras que en SLP no lo está. Simétricamente, *C* es un ancestro en SLP, mientras que en SLT no lo es. De hecho, sería correcto pero redundante el hecho de definir *C* como un ancestro en SLT, ya que toda derivación que se obtuviera usando instancias de *C* se obtiene, de todas formas, usando variantes frescas de *C*, ya que es una cláusula de entrada. Si la cláusula inicial *C* es base la diferencia entre SLT y SLP desaparece, obviamente. Esta diferencia puede ser observada claramente en el siguiente ejemplo.

Ejemplo 4.5 (Diferencia entre el procedimiento SLP y SLT)

Para la teoría $T = \{p(a) \vee p(b) \leftarrow\}$ y la cláusula $\leftarrow p(x)$, el procedimiento SLP falla finitamente (vía cualquier elección de ancestros). La siguiente figura muestra un árbol SLP vía ALL que es finito y todos sus ramas corresponden a derivaciones falladas.

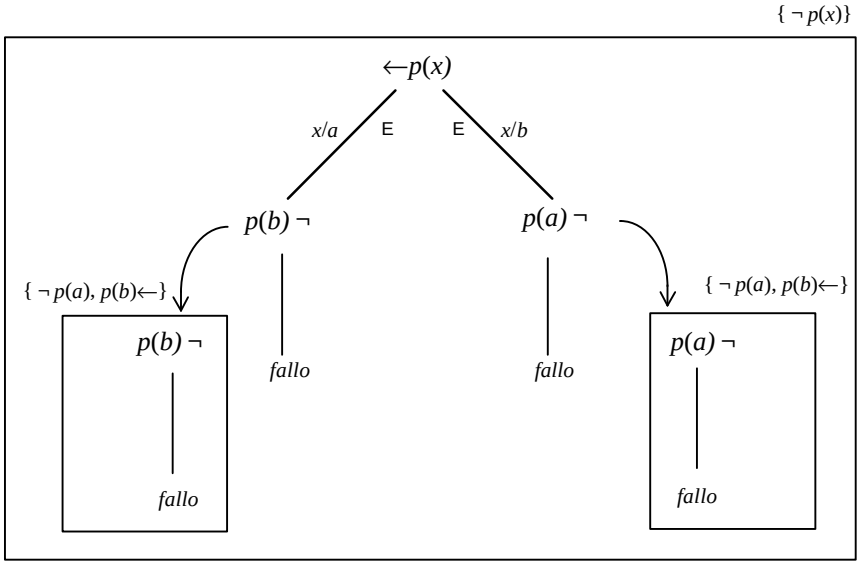


Figura 4.3: Árbol SLP vía ALL para el Ejemplo 4.5

La figura muestra un árbol SLP desde $\leftarrow p(x)$ en T en el cual aparecen dos árboles subsidiarios. Cada árbol se encuentra enmarcado por un rectángulo y en la parte superior izquierda de cada rectángulo se muestra el conjunto de ancestros del árbol correspondiente. El conjunto de ancestros para el árbol subsidiario de la izquierda es $\{\leftarrow p(a), p(b)\leftarrow\}$, esto es el conjunto inicial $\{\leftarrow p(x)\}$ instanciado por la sustitución subsidiaria en curso $\{x/a\}$ junto con la raíz del árbol $p(b)\leftarrow$. El árbol subsidiario falla inmediatamente ya que no existe ninguna cláusula de entrada en T ni ningún ancestro que pueda resolver con $p(b)\leftarrow$. El mismo razonamiento se puede hacer para el árbol subsidiario de la derecha. El árbol SLP desde $\leftarrow p(x)$ en T con todos sus ramas falladas indica que no existe una sustitución θ tal que $T \models \forall(p(x)\theta)$. En oposición a esto, el procedimiento SLT tiene éxito al refutar $\leftarrow p(x)$ en T (sin importar la elección de ancestros que se utilice). En la siguiente figura se puede observar un árbol SLT desde $\leftarrow p(x)$ en T vía ALL que contiene dos refutaciones.

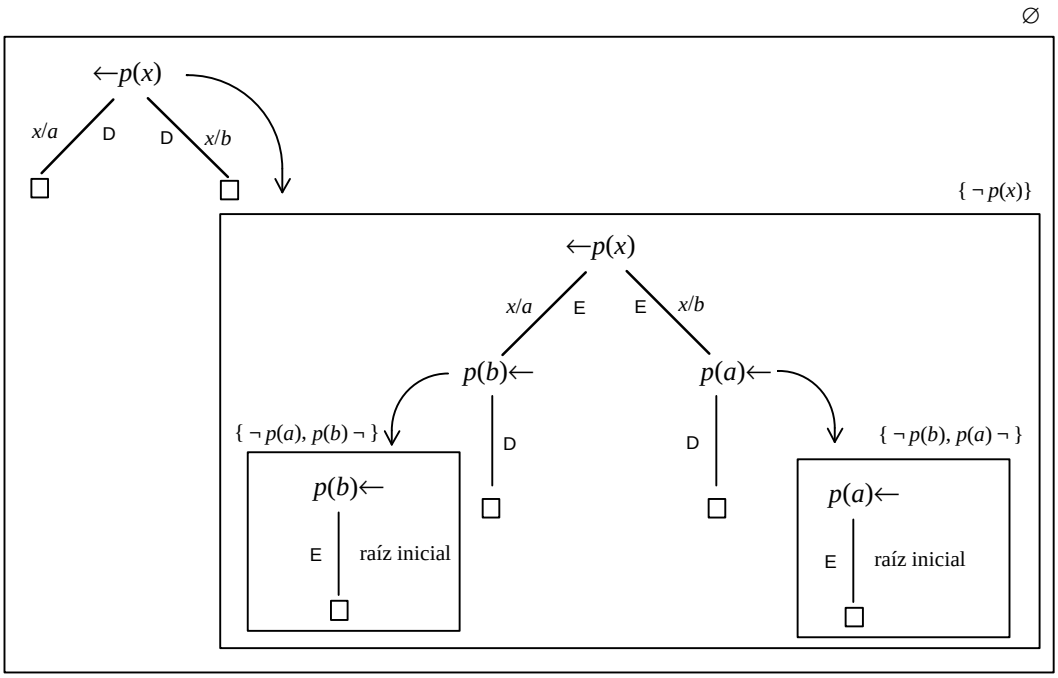


Figura 4.4: Árbol SLT vía ALL para el Ejemplo 4.5

Las dos refutaciones que aparecen en el árbol SLT desde $\leftarrow p(x)$ en T indican que $T \cup \{\leftarrow p(x)\}$ es inconsistente, es decir, $\exists(p(x))$[14] es un teorema de T. Además también es posible obtener la respuesta indefinida $T \models p(a) \vee p(b)$ de cada una de los dos refutaciones presentes en el árbol. En general, el procedimiento SLP siempre computa respuestas definidas, mientras que SLT computa tanto respuestas definidas como indefinidas.

Las siguientes definiciones formalizan las respuestas que obtienen los procedimientos SLP y SLT.

Definición 4.12 (Respuesta computada de una refutación SLP)

Sea R una refutación SLP desde alguna cláusula C. Entonces, la *respuesta computada* de R es la restricción de la sustitución computada de R a las variables de C.

Definición 4.13 (Respuesta computada de una refutación SLT)

Sea R una refutación SLT desde alguna cláusula C y θ la sustitución computada de R. Además, $\rho_1, \ldots, \rho_m$ ($m \geq 0$) son los renombramientos de las variables de C tal que $\{C_{\rho_i} : 1 \leq i \leq m\}$ es el conjunto de las variantes frescas de C usadas como cláusulas de entrada en R, incluyendo las refutaciones subsidiarias. Entonces, $\Theta = \{\theta_0, \rho_1\theta_1, \ldots, \rho_m\theta_m\}$ es la *respuesta computada* de R, donde θ_0 es la restricción de θ a las variables de C y θ_i es la restricción de θ a las variables de C_{ρ_i} ($1 \leq i \leq m$).

[14] $\exists$ denota el cierre existencial.

Intuitivamente, la respuesta computada Θ de *R*, siendo *R* una refutación SLT, se obtiene aplicando la composición de las sustituciones usadas en *R* a la cláusula raíz y a cada una de sus variantes usadas como cláusulas de entrada en algún paso de *R*. Para m=0, Θ es definida y coincide con la repuesta computada por SLP; para m>0, Θ es indefinida.

Los siguientes dos ejemplos ilustran las diferentes características de los procedimientos SLP y SLT.

Ejemplo 4.6 (Características de los procedimientos SLP y SLT)

La refutación SLT desde ←$p(x)$ en *T* (cláusulas 1 a 6) vía POS es también un refutación SLP, ya que, después del primer paso, la raíz inicial no es usada más veces. La notación utilizada es la habitual: las reglas de inferencia se denotan por D (regla de división), E (resolución de entrada) y A (resolución de ancestro). Las cláusulas de entrada son identificadas por números. Cuando sea necesario el literal seleccionado se subraya. Las computaciones de diferentes rangos (ya sean refutaciones, derivaciones o árboles) son enmarcadas en rectángulos anidados. Unicamente las sustituciones que contribuyen a la respuesta computada son incluidas en la figura.

$p(x) \leftarrow q(x) \wedge r(x)$	1
$q(x) \leftarrow s(x)$	2
$q(f(x)) \leftarrow t(x)$	3
$r(a) \vee t(b) \leftarrow$	4
$t(x) \vee s(x) \leftarrow r(x)$	5
$\leftarrow q(x) \wedge t(y)$	6

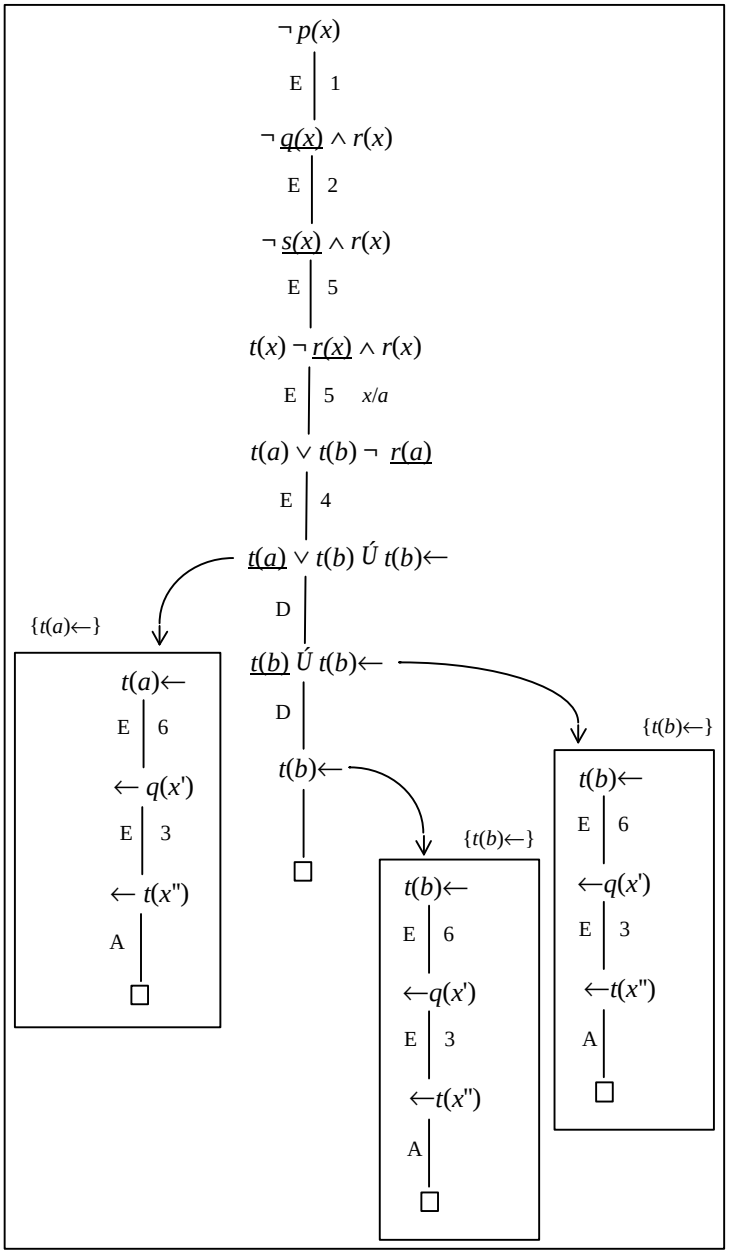


Figura 4.5: Refutación SLT vía POS para el Ejemplo 4.6

La respuesta computada por el procedimiento SLP es {*x*/*a*} (la computada por SLT es la respuesta definida {{*x*/*a*}}) indicando que *p*(*a*) es una consecuencia lógica de la teoría *T*.

Ejemplo 4.7

En el siguiente ejemplo [Chang y Lee, 1973] [Stickel, 1988] se describe una teoría de los número naturales (cláusulas 1 a 8). Siguiendo a [Stickel, 1988], las cláusulas se pueden interpretar como sigue: (1) *x* divide a *x*; (2) si *x* divide a *y* e *y* divide a *z*, entonces *x* divide a *z*; (3) o bien *x* es primo o existe un divisor *g(x)* de *x* que (4) es mayor que 1 y (5) es menor que *x*; (6) si *x* es mayor que 1 y menor que *n*, entonces existe un primo *f(x)* que (7) es un divisor de *x*; (8) *n* es mayor que 1. Por último la cláusula raíz es ¬ *pr(x) Ù div(x, n)*, que se interpreta como que no existe un número primo divisor de *n* (esta cláusula también se puede entender como la

pregunta ¿existe un número primo divisor de n?). La refutación SLT vía $\{\neg pr(x), less(1,g(x)\}$ que se muestra en la figura siguiente responde a esta pregunta con un "sí". Los literales son seleccionados de izquierda a derecha, primero en el cuerpo.

$div(x, x)$	1	$pr(x) \vee less(g(x), x) \leftarrow$	5
$div(x, z) \leftarrow div(x, y) \wedge div(y, z)$	2	$pr(f(x)) \leftarrow less(1, x) \wedge less(x, n)$	6
$pr(x) \vee div(g(x), x) \leftarrow$	3	$div(f(x), x) \leftarrow less(1, x) \wedge less(x, n$	7
$pr(x) \vee less(1, g(x)) \leftarrow$	4	$less(1, n) \leftarrow$	8

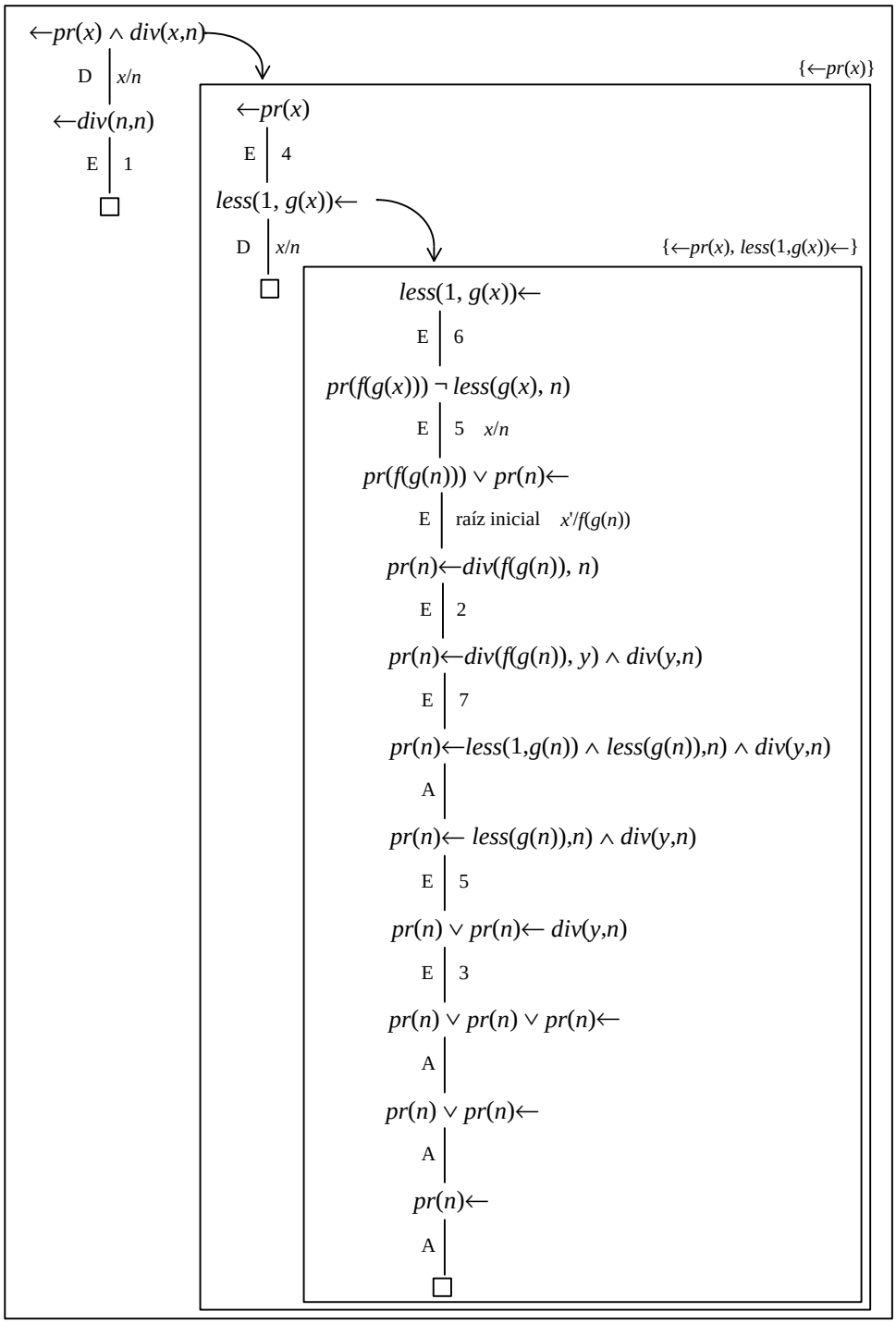


Figura 4.6: Refutación SLT vía $\{\neg pr(x), less(1,g(x)\}$ para el Ejemplo 4.7

La refutación SLT demuestra que existe un primo divisor de *n*. La respuesta computada es la respuesta indefinida concisa {*x*/*n*, *x*/*f*(*g*(*n*))}, que se puede interpretar como que o bien el primo divisor de *n* es el propio *n*, o bien es un primo divisor de un divisor de *n*. El uso de la cláusula raíz como tal y como cláusula de entrada contribuyen de igual forma a la generación de esta respuesta computada. Es interesante destacar que ningún intento de refutar la pregunta con el procedimiento SLP tiene éxito. Claramente, los pasos de los procedimientos SLT y SLP desde la raíz hasta el paso en el cual se usa una variante de la cláusula raíz son idénticos. En ese paso SLP falla, ya que no es posible aplicar ningún paso de inferencia sobre el literal seleccionado *pr*(*f*(*g*(*n*)). Esto es así porque en el procedimiento SLP la cláusula raíz pertenece al conjunto de ancestros y no es una cláusula de entrada. De esta manera, los ancestros en la derivación SLP en el instante del paso en cuestión son {←*pr*(*n*)∧*div*(*n*, *n*), ←*pr*(*n*), less(1,*g*(*n*)←}. Claramente, el ancestro ←*pr*(*n*)∧*div*(*n*, *n*) correspondiente a la cláusula raíz , no es unificable con el literal seleccionado en la cláusula en curso. El fallo SLP se interpreta como que no existe una respuesta definida a la pregunta ←*pr*(*x*)∧*div*(*x*,*n*) en la teoría dada.

La elección de ancestros utilizada en la refutación SLT anterior ha sido {¬*pr*(*x*), *less*(1,*g*(*x*))}. La razón de haber utilizado esta elección de ancestros es el ilustrar la importancia de la elección de ancestros para mejorar la eficiencia del procedimiento. Para este mismo problema también existe una refutación SLT vía EMPTY, POS, NEG y ALL. La diferencia fundamental de estas refutaciones frente a la presentada anteriormente es que todas ellas realizan un número mayor de pasos de inferencia. Desde esta perspectiva, se podría decir que la elección de ancestros anterior es óptima con respecto al problema presentado. Así, la elección de ancestro no sólo es fundamental para la obtención de la completitud en los procedimientos sino que además permite adecuar los procedimientos a los problemas a tratar, permitiendo mejorar la eficiencia de los mismos. En el apartado siguiente se realizará una mayor profundización sobre esta interesante cuestión.

Los siguientes teoremas presentan los resultados fundamentales de los procedimientos SLP y SLT.

Teorema 4.2 (Corrección y completitud de SLP para la obtención de respuestas definidas)

Sea *T* una teoría y *C* una cláusula. El procedimiento SLP es correcto y completo para la obtención de respuestas definidas, como se expresa en los puntos a) y b).

a) Si existe una refutación SLP desde *C* en *T* con respuesta computada θ, entonces θ es una respuesta definida para *C* en *T*.

b) Si *T* es consistente, θ es una respuesta definida para *C* en *T* y *Ch* una elección de ancestros completa entonces, existe una refutación SLP desde C en *T* vía *Ch* con respuesta computada ϕ y una sustitución σ tal que θ=ϕσ.

Obviamente cada sustitución computada por SLP lo es también por SLT. Por tanto, reemplazar SLP por SLT en el teorema 4.2 b) conduce a un resultado válido. Sin embargo, esto mismo en el teorema 4.2 a) sería incorrecto, ya que, por ejemplo existe dos refutaciones SLT desde $\leftarrow p(x)$ en $\{p(a) \vee p(b) \leftarrow\}$ con sustituciones computadas x/a y x/b y ninguna de ellas es una respuesta definida. También es interesante destacar que la ausencia de respuestas definidas no implica que $T \not\models \exists(\neg C)$, ya que pueden existir respuestas indefinidas.

Teorema 4.3 (Corrección y completitud de SLT para la comprobación de la inconsistencia)

Sea *T* una teoría y *C* una cláusula. El procedimiento SLT es correcto y completo para la comprobación de la inconsistencia como se expresa en los puntos a) y b), respectivamente

a) Si existe una refutación SLT desde C en *T*, entonces $T \cup \{C\}$ es inconsistente.

b) Si *T* es consistente, $T \cup \{C\}$ es inconsistente y *Ch* es una elección de ancestros completa, entonces existe una refutación SLT desde *C* en *T* vía *Ch*.

La sustitución de SLT por SLP en el teorema anterior en el punto a) conduce a un resultado válido. Sin embargo, la misma sustitución en b) sería incorrecto. Claramente, al no incluirse la cláusula raíz en el conjunto de cláusulas de entrada en el procedimiento SLP, éste no es completo en la comprobación de la inconsistencia.

Teorema 4.4 (Corrección y completitud de SLT en la obtención de respuestas)

Sea *T* una teoría y *C* una cláusula. El procedimiento SLT es correcto y completo en la obtención de respuestas, como se expresa en los puntos a) y b), respectivamente.

a) Si existe una refutación SLT desde *C* en *T* con respuesta computada Θ, entonces Θ es una respuesta de *C* en *T*.

b) Si T es consistente, Θ es una respuesta para *C* en *T* y *Ch* una elección de ancestros completa entonces, existe una refutación SLT desde *C* en *T* vía *Ch* con respuesta computada Φ tal que, para cada ϕ en Φ, existe una sustitución θ en Θ y una sustitución σ tal que $\theta=\phi\sigma$.

En el resultado del teorema anterior en el punto a), Θ no tiene que ser necesariamente concisa. Por ejemplo, la respuesta computada $\{\{x/a\}, \{x/b\}\}$ para $\leftarrow p(x)$ en $T=\{p(a) \vee p(b) \leftarrow, p(x) \leftarrow q(x), q(a)\}$ es redundante, reflejando la propia redundancia presente en *T*. Intuitivamente, el punto b) del teorema anterior indica que cada respuesta es capturada por una respuesta computada.

4.3.2. LA ELECCIÓN DE ANCESTROS

En este punto se va a profundizar sobre la importancia de la elección de ancestros en el comportamiento de resolución SL*. Existen dos aspectos importantes de resolución SL* en los cuales interviene de forma fundamental la elección de ancestros: por un lado la completitud de los procedimientos correspondientes vía la elección de ancestros elegida y, por otro lado, la eficiencia de dichos procedimientos atendiendo a aspectos como el tamaño del espacio de búsqueda, coste de las reglas de inferencia aplicadas, etc. Los siguientes apartados presentan estos puntos con mayor profundidad.

4.3.2.1. LA ELECCIÓN DE ANCESTROS Y LA COMPLETITUD DE RESOLUCIÓN SL*

Como ya se comentó en los apartados anteriores, la elección de ancestros determina si el procedimiento instancia correspondiente a dicha elección es completo en la resolución de un cierto problema. Este hecho por sí mismo ya determina la importancia de la elección de ancestros. Así, por ejemplo, los procedimientos SLT y SLP vía EMPTY no son completos para los objetivos de comprobación de la inconsistencia y resolución de problemas. En el siguiente ejemplo se puede observar este hecho.

Ejemplo 4.8 (Completitud y elección de ancestros)

Sea T la teoría formada por las cláusulas 1 a 3. Fácilmente se puede comprobar que $T \cup \{\leftarrow p \wedge q\}$ es inconsistente (resp. que $T \models p \wedge q$). Sin embargo, no existe una refutación SLT desde $\leftarrow p \wedge q$ en T vía EMPTY (resp. una refutación SLP desde $\leftarrow p \wedge q$ en T vía EMPTY). Claramente, SLT y SLP son equivalentes ya que la raíz inicial es base. En la siguiente figura se puede observar un árbol SLT desde $\leftarrow p \wedge q$ en T vía EMPTY.

$p \leftarrow q$	1
$q \leftarrow p$	2
$p \vee q \leftarrow$	3

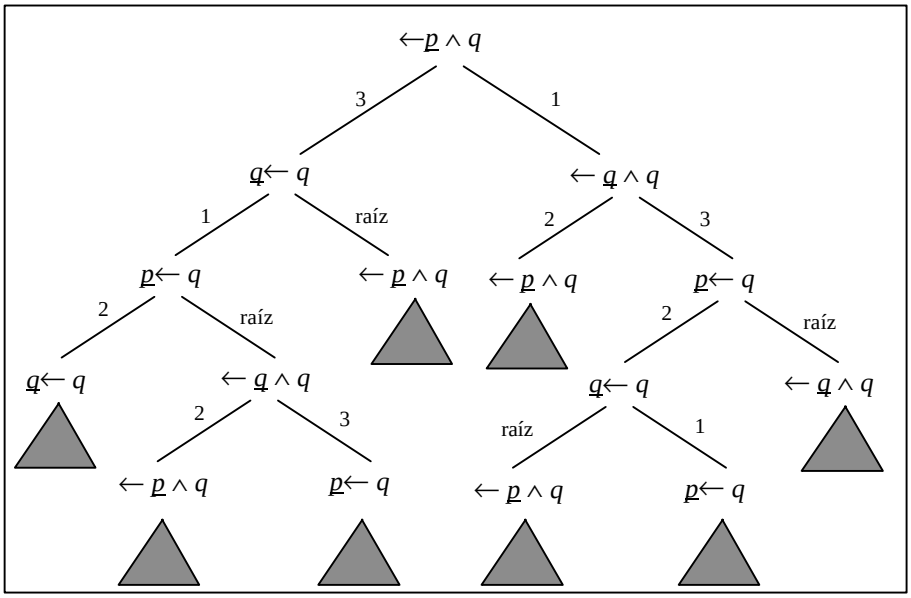


Figura 4.7: Árbol SLT vía EMPTY para el Ejemplo 4.8

En la figura el literal seleccionado está subrayado y cada arco se ha etiquetado con la cláusula de entrada utilizada. Evidentemente, ya que la elección de ancestros utilizada es EMPTY, la única regla de inferencia utilizada es resolución de entrada y además no existen computaciones subsidiarias. El triángulo sombreado se ha utilizado para denotar que la cláusula en curso (la que se encuentra justo sobre el triángulo) es idéntica a una cláusula anterior en el árbol. De esta forma, el subárbol denotado por el triángulo es idéntico al que parte de esa cláusula padre. Claramente, se pueden observar dos hechos: primero, SLT (o SLP) vía EMPTY no es capaz de refutar el problema dado[15]; segundo, ya que el árbol que se obtiene es infinito, SLT (o SLP) vía EMPTY no termina de computar este problema. Este segundo hecho está también asociado a la elección de ancestros, ya que al ser la elección de ancestros EMPTY el procedimiento no realiza computaciones subsidiarias y no almacena los literales que ha ido seleccionando, siendo incapaz de resolver el problema. La siguiente figura presenta un árbol SLT vía ALL para el mismo problema.

[15] Lo cual no ha de extrañar, ya que este problema es un clásico ejemplo de la incompletitud de resolución de entrada, que ya se ha comentado que es equivalente a SLT vía EMPTY.

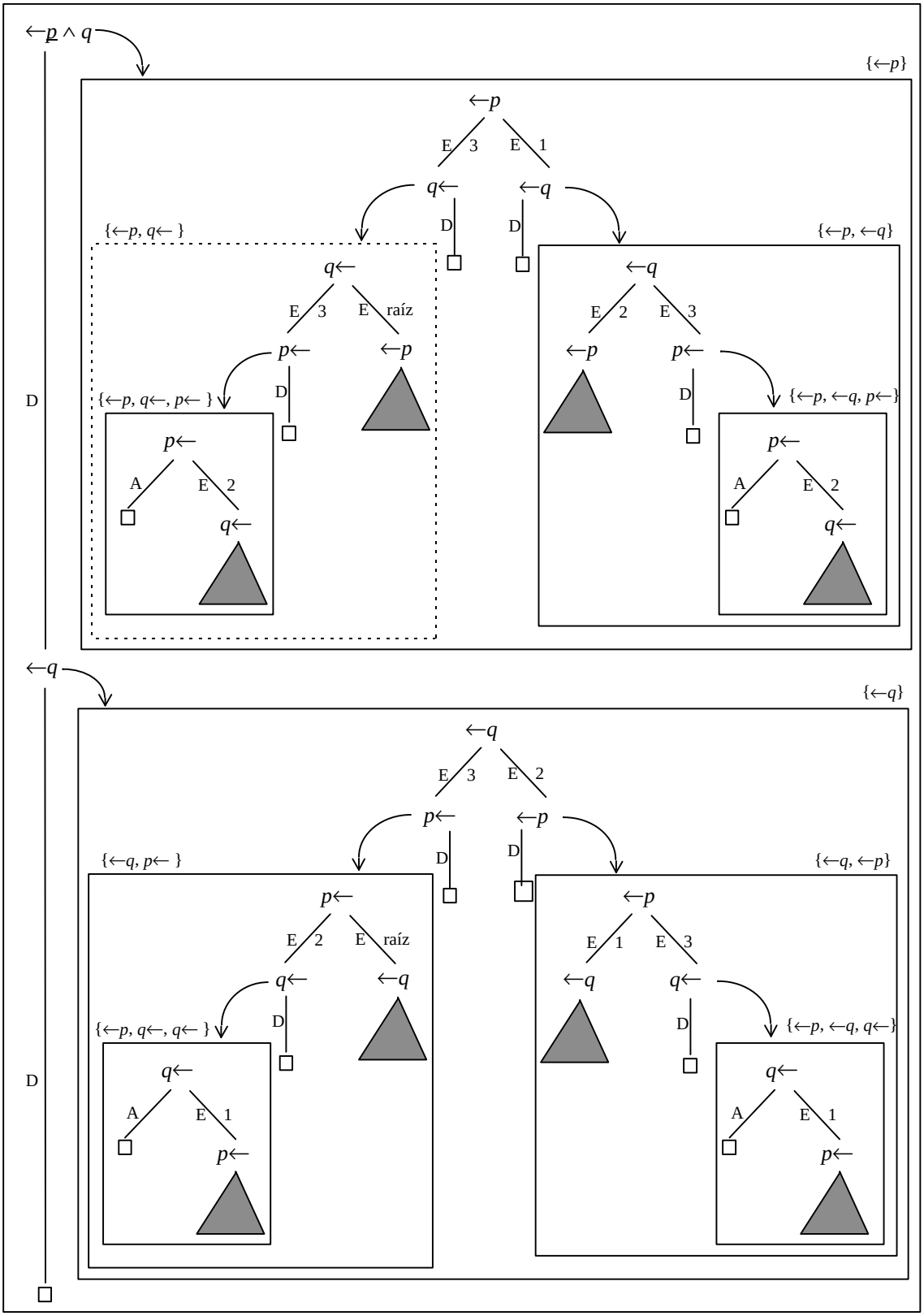


Figura 4.8: Árbol SLT vía ALL para el Ejemplo 4.8

En la figura se puede observar que el procedimiento SLT vía ALL sí es capaz de refutar el problema del Ejemplo 4.8. La diferencia esencial es la presencia de los ancestros en los árboles de rango más inferior que permite llegar a la cláusula vacía. Por otro parte, es interesante

destacar que el árbol SLT vía ALL, al igual que el SLT vía EMPTY, es infinito. También como en la figura anterior, los triángulos sombreados denotan que el subárbol que parte de la cláusula que se encuentra justo arriba es idéntico al subárbol de una cláusula anterior. Evidentemente, si fuera posible evitar el recorrer el espacio asociado a estos subárboles representados por los triángulos sombreados, es decir si se pudiera podar estos árboles, los procedimientos serían más eficientes. Esto conduce al segundo aspecto asociado a la elección de ancestros: el tamaño del espacio de búsqueda.

4.3.2.2. LA ELECCIÓN DE ANCESTROS Y LA EFICIENCIA DE RESOLUCIÓN SL*

Si se presta atención a la Figura 4.8 del árbol SLT vía ALL , se puede observar que cada uno de los literales que se encuentran sobre los subárboles representados por los triángulos sombreados es idéntico a uno de sus ancestros. Como ejemplo, obsérvese el árbol que se encuentra enmarcado por el rectángulo de contorno discontinuo: la raíz de ese árbol es $q\leftarrow$ y el conjunto de ancestros es $\{\leftarrow p,\ q\leftarrow\}$; por otro lado la cláusula hija de la raíz del árbol a la derecha es $\leftarrow p$, que se obtiene como el resolvente de la raíz del árbol $q\leftarrow$ y la cláusula de entrada $\leftarrow p \wedge q$ (la raíz inicial). El literal de la cláusula unitaria $\leftarrow p$ ya se encuentra como tal en el conjunto de ancestros, por tanto es inútil continuar la computación desde él, ya que de existir alguna refutación, ésta aparecerá en el árbol cuya raíz es el ancestro idéntico. De forma más intuitiva, el hecho de que el literal seleccionado en la cláusula en curso sea igual a un ancestro corresponde a la presencia de recursividad no constructiva en la teoría. Lo anteriormente enunciado se puede considerar como una poda y es utilizado en numerosos procedimientos [Loveland, 1968] [Stickel, 1988] [Letz, Schuman, Bayerl y Bibel, 1992]. Esta poda se puede resumir de la siguiente forma: la derivación falla cuando el literal seleccionado en la cláusula en curso es idéntico a un ancestro. La forma más sencilla de incorporar este mecanismo al SL* es el añadir a las definiciones de refutación, derivación y árbol SL* una condición más que formalice esta poda. Como un ejemplo de estas nuevas definiciones, a continuación se presenta la definición correspondiente a la refutación SL* con esta modificación.

Definición 4.14 (Refutación SL* con poda por igualdad con ancestro)

Sea T una teoría, C una cláusula, Anc un conjunto de cláusulas y Ch una elección de ancestros. Una *refutación SL* desde C en T vía Ch usando Anc de rango k* se define como sigue: La refutación consiste en una secuencia de cláusulas $C_0,\ldots, C_n$, una secuencia de cláusulas padre lejano $C'_1,\ldots, C'_n$ una secuencia de substituciones $\theta_1,\ldots, \theta_n$ ($n\geq 0$), tal que $C_0=C$, $C_n=\square$ y, en cada cláusula C_i distinta de la cláusula vacía, un literal L_i es seleccionado tal que, para $Anc_0=Anc$ y $Anc_i=Anc\theta_1\ldots\theta_i$, los siguientes puntos se cumplen:

i) Si $L_i \notin Ch$ o $L_i \in Anc_i$, entonces C_{i+1} es un resolvente de C_i y C'_{i+1} sobre L_i y un literal L' de C'_{i+1} usando el umg θ_{i+1} de los átomos de L_i y L', cumpliéndose que o bien la cláusula padre-lejano C'_{i+1} es una variante fresca de una cláusula de T o bien $C'_{i+1} \in Anc_i$.

ii) Si $L_i \in Ch$ y $L_i \notin Anc_i$, entonces existe una refutación desde L_i en T vía Ch usando $Anc_i \cup \{L_i\}$, de rango menor que k, con sustitución computada θ_{i+1}, y C_{i+1} es el resolvente de C_i y C'_{i+1} tal que la cláusula padre-lejano C'_{i+1} es igual $\tilde{L}_i\theta_{i+1}$, donde $\tilde{L}_i$ es el literal complementario de L_i.

iii) Si $i>0$ entonces $L_i \notin Anc_i$.

Además existe un cierto i, $0 \leq i \leq n$, tal que se cumple el punto ii) y la refutación SL* desde L_i en T vía Ch usando $Anc_i \cup \{L_i\}$ es de rango k-1.

El nuevo punto iii) corresponde a esta poda. La condición $i>0$ es necesaria para que no se aplique la poda en el primer paso de la refutación, ya que en este primer paso el literal seleccionado pertenece necesariamente al conjunto de ancestros[16]. Las definiciones de derivación SL* y árbol SL* pueden extenderse con esta poda de forma sencilla. Por otro lado, hay que destacar que esta poda no es necesaria para garantizar la corrección y completitud de los procedimientos basados en resolución SL*. Asimismo, esta poda se puede considerar como un mecanismo sencillo de eliminación de tautologías, como ya se comentó en anteriores capítulos. El siguiente ejemplo ilustra la anterior afirmación.

Ejemplo 4.9 (Poda por igualdad con ancestro y reducción de tautologías)

Dado el conjunto de cláusulas $\{p \vee q \leftarrow, \leftarrow p \wedge q\}$ se puede observar que es consistente. Es más, al aplicar un paso de resolución sobre cualquier de los dos literales posibles se obtiene una cláusula tautológica. Veámoslo en la siguiente derivación de resolución lineal desde $\leftarrow p \wedge q$ en ese conjunto de cláusulas.

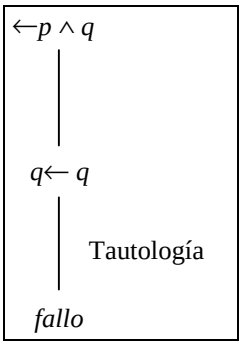


Figura 4.9: Tautología en una resolución lineal

[16] Como ejemplo, piénsese que el literal ráiz de una computación subsidiaria ha sido incorporado al conjunto de ancestros por la computación de rango superior y en el primer paso es necesario su selección.

Ahora se presenta la derivación SLT desde $\leftarrow p \wedge q$ en $\{p \vee q\leftarrow\}$ vía ALL.

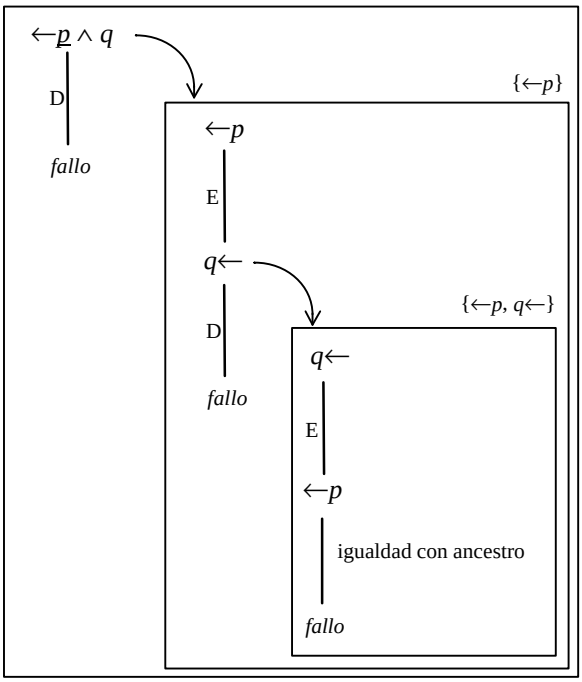


Figura 4.10: Poda por igualdad con ancestro

Hay que destacar que la poda utilizando la igualdad del literal seleccionado con un ancestro detecta las tautologías a posteriori. Esto es, en la derivación SLT anterior se puede observar que el corte o poda de la derivación se produce después de haber efectuado un paso más de resolución que en la derivación de resolución lineal. La explicación es que resolución SL* detecta las tautologías mediante la inspección de los literales seleccionados previamente, mientras que en resolución lineal esta detección se realiza directamente en la cláusula en curso.

Es interesante destacar que la poda por igualdad con ancestro evita el uso de información proveniente de una tautología aunque no se dé la presencia de una tautología en la derivación. Esto puede verse claramente en el siguiente ejemplo.

Ejemplo 4.10 (Poda por igualdad con ancestro y detección del uso de información tautológica)

Sea T el siguiente conjunto de cláusulas $\{ p\leftarrow q, q\leftarrow p \wedge r\}$ y sea C la cláusula $\leftarrow p \wedge t$. Claramente se puede comprobar que los dos resolventes que se pueden obtener de las cláusulas en T son dos tautologías, por lo que su uso es inútil. Además, como T no contiene más cláusulas será consistente junto con cualquier cláusula, en particular C. A continuación se muestra la derivación lineal desde C en T

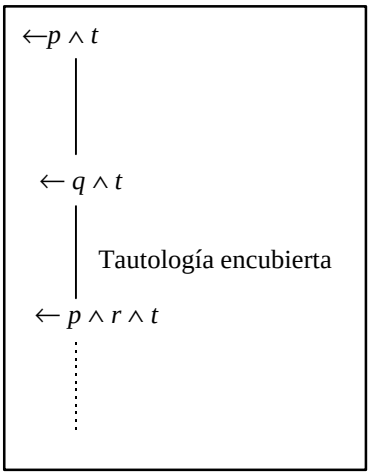


Figura 4.11: Derivación lineal para el Ejemplo 4.10

Se puede ver fácilmente que la continuación de la derivación es inútil ya que la última cláusula mostrada es subsumida por la cláusula raíz. Esta problemática introducida por las tautología no explícitas se resuelve en resolución SL* mediante la poda por igualdad con ancestro, como se puede ver en la figura siguiente.

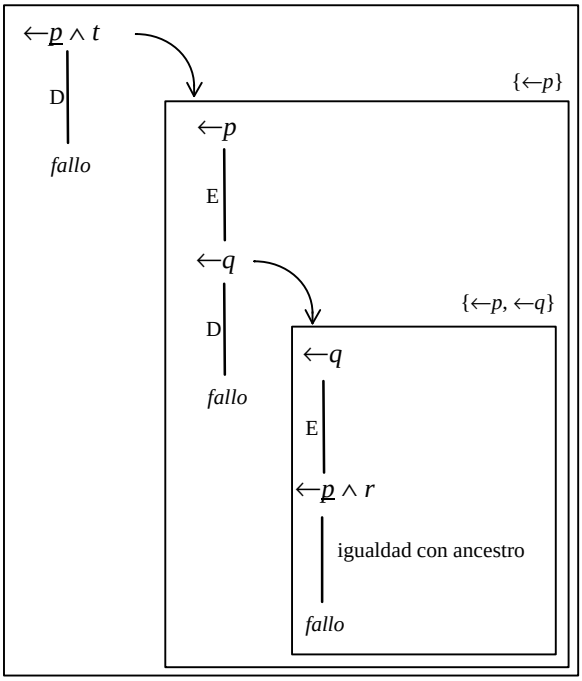


Figura 4.12: Derivación SL* para el Ejemplo 4.10

Teniendo en cuenta que la poda por igualdad con ancestro se aplica cuando el literal seleccionado es sintácticamente igual a un ancestro anterior, se podría pensar que esta poda se podría extender al caso en que el literal seleccionado sea unificable con un ancestro anterior. Evidentemente, esto llevaría a una aplicación más frecuente, a una poda mayor y a un espacio de búsqueda más reducido. Lamentablemente, la aplicación de este criterio es inadecuada ya que da

lugar a la pérdida de la completitud de los procedimientos. Este problema se ilustra con el siguiente ejemplo.

Ejemplo 4.11 (La aplicación de la poda por unificabilidad con ancestro)

Sea T el conjunto de las cláusulas 1 a 3 abajo indicadas. Evidentemente, la teoría formada por $T \cup \{\leftarrow p(x) \wedge s(x)\}$ es inconsistente. Sin embargo no existe una refutación SLT desde $\leftarrow p(x) \wedge s(x)$ en T vía ALL aplicando la poda por unificabilidad con ancestro.

$p(x) \leftarrow p(y)$	1
$p(a) \leftarrow$	2
$s(b) \leftarrow$	3

La figura inferior muestra un árbol SLT desde $\leftarrow p(x) \wedge s(x)$ en T vía ALL, aplicando dicha poda.

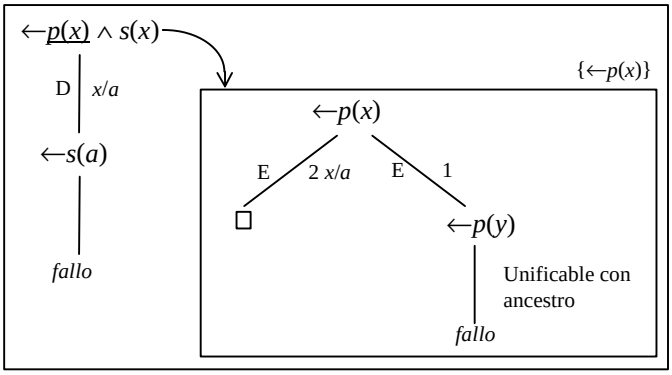


Figura 4.13: Árbol SLT vía ALL con poda por unificabilidad con ancestro para el Ejemplo 4.11

A partir de ahora cuando se haga referencia a resolución SL* o alguno de sus procedimientos instancias, se asumirá que el mecanismo de aplicación de la poda anterior está incorporado.

Por otro parte, es interesante destacar la importancia de la elección de ancestro para la aplicación de la poda por igualdad con ancestro, ya que, ésta se aplica más veces, genéricamente hablando, cuantos más literales contenga la elección de ancestros. El siguiente ejemplo presenta un caso en el cual SLT vía POS es incapaz de dar una solución, a diferencia de SLT vía ALL que falla.

Ejemplo 4.12 (Elección de ancestros y poda por igualdad sintáctica con ancestros)

Los árboles SLT desde $\leftarrow p$ en $\{p \leftarrow q, q \leftarrow p\}$ vía POS y vía ALL se pueden observar en la siguiente Figura 4.14.

Como se puede observar SLT vía POS es incapaz de detener la computación ya que la condición de poda que se ha dado no se puede aplicar al no utilizarse ninguna computación

subsidiaria. Sin embargo, SLT vía ALL es capaz de computar la consistencia de este problema al fallar finitamente. En general cuanto mayor sea la elección de ancestros más posibilidades hay de aplicar esta poda, dando por consiguiente espacios de búsqueda más reducidos. En particular para resolución SL* vía ALL (ya sea el procedimiento SLT o SLP) se cumple el siguiente resultado.

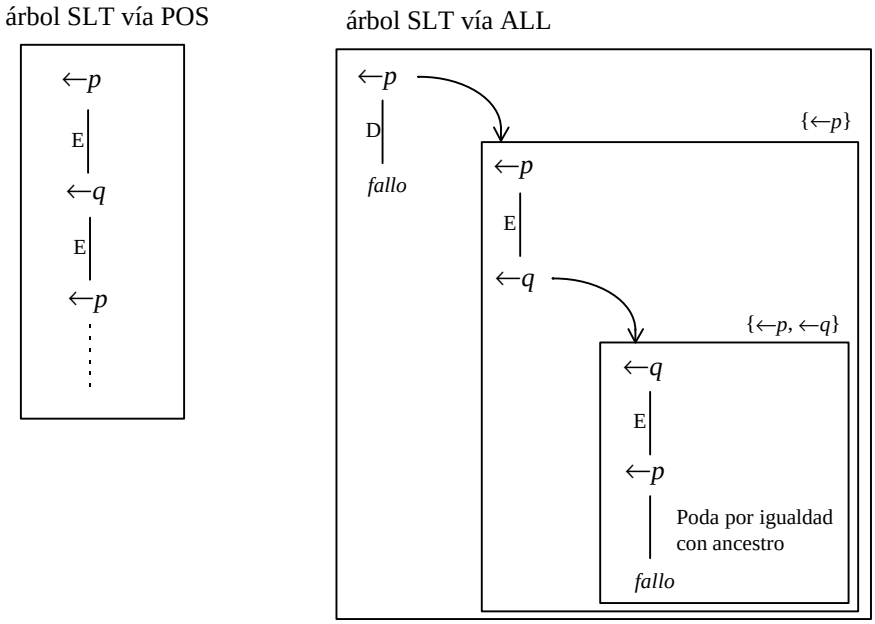


Figura 4.14: Árboles SLT vía POS y vía ALL para el Ejemplo 4.12

Teorema 4.5 (Poda por igualdad con ancestro y elección de ancestros)

Sea T una teoría consistente formada por cláusulas base y C una cláusula base. Entonces los siguientes dos puntos se cumplen:

a) SLT vía ALL es un procedimiento de decisión de la inconsistencia de $T \cup \{C\}$.
b) SLP vía ALL es un procedimiento de decisión de $T \models \neg C$.

Para acabar solamente recordar que existen otras posibles podas para la reducción del tamaño del espacio de búsqueda que no están asociadas a la elección de ancestro y que por ser más técnicas se dejan para el apartado en el cual se tratarán aspectos asociados a la implementación de los procedimientos.

Para terminar este apartado se va a presentar otro aspecto asociado a la eficiencia de resolución SL*: la influencia de la elección de ancestros en la longitud de la refutación más corta asociada a un cierto problema y en el número de aplicaciones de las diferentes reglas de inferencia que se realizan para refutar un problema dado.

De los apartados anteriores se podría deducir que cuanto mayor sea la elección de ancestros mayor va a ser la eficiencia del procedimiento correspondiente. Sin embargo, está

relación (mayor elección de ancestros, mayor eficiencia) no está clara. El Ejemplo 4.7 sirve para ilustrar esta problemática. En la refutación SLT desde $\leftarrow pr(x) \wedge div(x,n)$ vía $\{\neg pr(x), less(1,g(x))\}$ que se muestra en la Figura 4.15 se aplican los siguientes pasos de reglas de inferencia: 2 pasos de la regla de división (D), 9 pasos de resolución de entrada (E) y 4 pasos de resolución de ancestro (A). La elección de ancestros utilizada es óptima para este problema, en el sentido de que no existe ninguna refutación con menos pasos. Con la misma función de selección (de izquierda a derecha, primero en el cuerpo) y usando las misma cláusulas de entrada, se obtiene una refutación SLT vía ALL que contiene 13 pasos D, 9 pasos E y 4 pasos A. Usando otras cláusulas de entrada distintas, ya que existen pasos de resolución de ancestro en las otras refutaciones que no pueden efectuarse con la elección de ancestros POS, se obtiene una refutación SLT vía POS con 4 pasos D, 15 pasos E y 1 paso A. Para este mismo problema la refutación más corta SLT vía EMPTY tiene 18 pasos E. Pero la pregunta clave está todavía sin responder: ¿cuál de todos los procedimientos es el más eficiente? Para responder a esta pregunta se analizará cada una de las reglas de inferencia. Para ello se va a suponer una derivación con una cláusula en curso C_i y un literal seleccionado L_i. Además el conjunto de ancestros para esta derivación es Anc_i.

+ resolución de entrada: sin duda es la regla en general más costosa y cuya aplicación provoca un aumento exponencial del espacio de búsqueda. Para aplicar esta regla se ha de buscar un literal en alguna cláusula que permita aplicar un paso de resolución binaria. El coste de la aplicación de esta regla es proporcional al número de literales existente en las cláusulas de la teoría inicial.

+ resolución de ancestro: la aplicación de esta regla pasa por encontrar un elemento en el conjunto de ancestros con el cual se pueda resolver la cláusula en curso sobre el literal seleccionado. La aplicación de esta regla no provoca un aumento del espacio de búsqueda, excepto cuando el ancestro seleccionado sea la cláusula raíz y ésta no sea unitaria. El coste de la aplicación de esta regla es proporcional al número de elementos del conjunto de ancestros, aunque puede mejorarse realizando una indexación en el conjunto.

+ regla de división: la aplicación de esta regla tiene un coste despreciable, ya que únicamente realiza la descomposición del problema de refutar la cláusula actual en dos subproblemas que se refutan de forma separada. En sí misma esta regla no se tiene un coste comparable a las otras dos. Además esta regla no tiene influencia sobre el espacio de búsqueda.

Atendiendo a este análisis se puede concluir que el espacio de búsqueda asociado para refutar un cierto problema es mayor cuanto mayor sea el número de pasos de resolución de entrada de la refutación de menor longitud (teniendo en cuenta las refutaciones subsidiarias que

pueda utilizar). Además, cada paso de resolución de ancestro permite reducir el problema en curso y cuando un paso de resolución de ancestro "posible"[17] no se aplica (por no haber sido considerado ancestro el literal previamente seleccionado al no estar en la elección de ancestros) provoca un aumento de la longitud de la refutación. El siguiente ejemplo ilustra esta afirmación.

Ejemplo 4.13 (Aumento de la longitud de las refutaciones al no aplicar algunos pasos de resolución de ancestro)

Sea *T* la teoría formada por las siguientes cláusulas:

$p \leftarrow q$	
$q \leftarrow r_1$	$q \leftarrow s_1$
$r_1 \leftarrow r_2$	$s_1 \leftarrow s_2$
$r_2 \leftarrow r_3$	$s_2 \leftarrow s_3$
…	…
$r_{n-1} \leftarrow r_n$	$s_{n-1} \leftarrow s_n$
$r_{n-1} \vee r_n \leftarrow$	$s_n \leftarrow gran\_espacio$

A parte de estas cláusulas, existen otras en *T* que crean un espacio de búsqueda muy grande sin ninguna refutación para s_n. Esto se ha denotado por el predicado *gran_espacio*. La siguiente figura muestra a la izquierda un árbol SLT desde $\leftarrow p$ en T vía $\{\neg r_{n-1}\}$ (que es la elección de ancestros óptima para este problema) y a la derecha un árbol SLT desde $\leftarrow p$ en *T* vía EMPTY.

[17] "Posible" en el sentido de que existe un literal previamente seleccionado que es complementario y unificable con el literal seleccionado en la cláusula en curso.

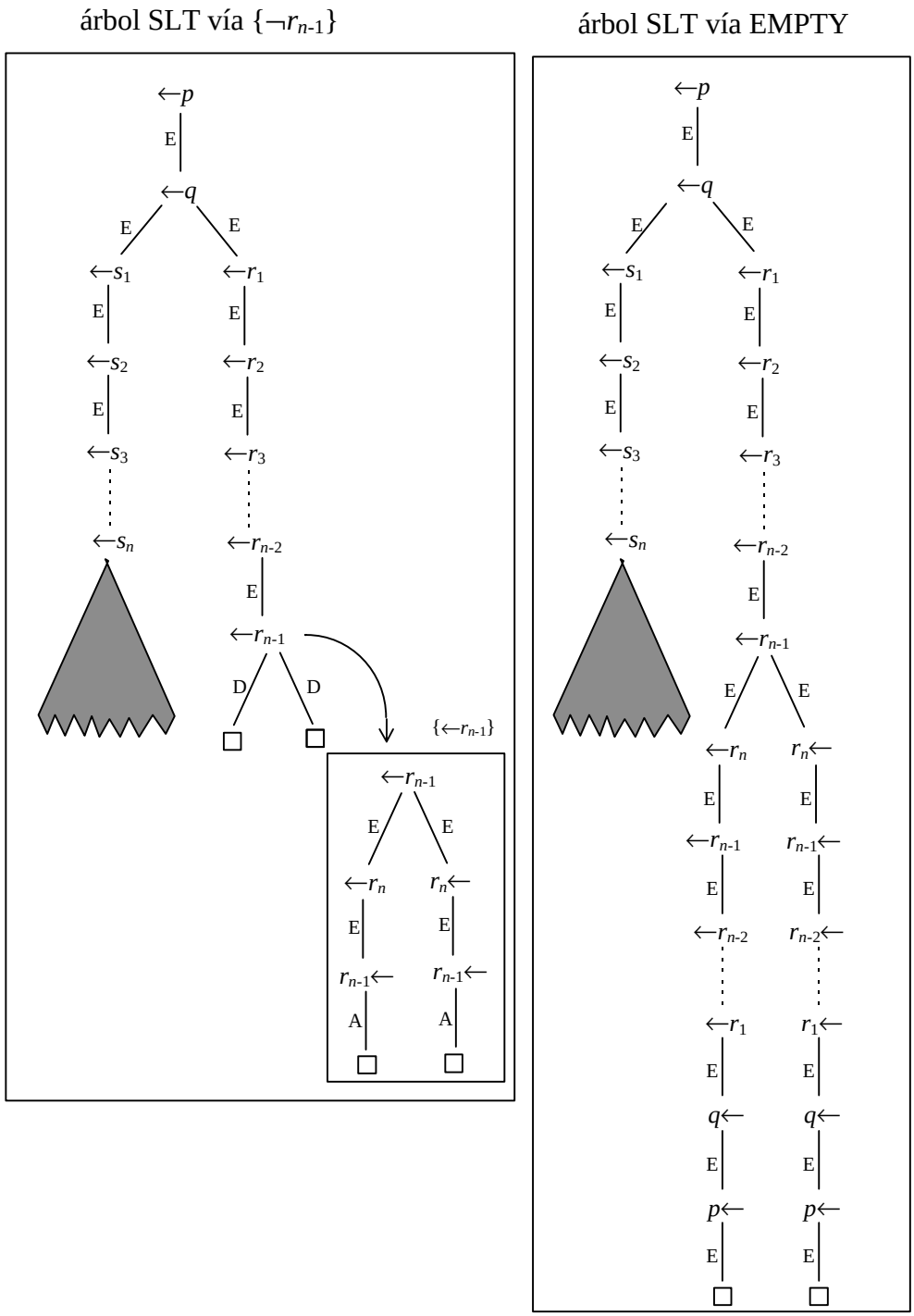


Figura 4.15: Árboles SLT vía {¬r_{n-1}} (izquierda) y vía EMPTY (derecha)

Como se puede observar en la figura las refutaciones del árbol SLT vía {¬r_{n-1}} tienen 1 paso D, n+2 pasos E un paso A. Sin embargo las refutaciones del árbol SLT vía EMPTY tienen 2n+3 pasos E. El motivo de este aumento de los pasos E en las refutaciones de SLT vía EMPTY, como se puede observar en la figura, es que los pasos de resolución de ancestro del árbol SLT vía {¬r_{n-1}} de la izquierda se han de sustituir por derivaciones parciales en el árbol

SLT vía EMPTY de la derecha, aumentando de esta forma la longitud de las refutaciones y teniendo que recorrer un espacio mucho mayor[18] para poder encontrar la cláusula vacía. En este ejemplo el problema asociado al espacio de búsqueda puede parecer trivial, pero, como ya se comentó en los capítulos iniciales, en general la capacidad de poder resolver ciertos problemas de carácter "profundo" recae en la posibilidad de reducir el espacio de búsqueda que tienen asociado. Como conclusión se podría decir que si existe la posibilidad de aplicar resolución de ancestro en una derivación, la elección de ancestro ha de posibilitar esta aplicación. Así, la elección de ancestros ALL, según este aspecto, sería la más conveniente ya que permite la aplicación de toda resolución de ancestro posible.

Por otra parte, la poda propuesta en el apartado anterior se aplica con mayor frecuencia cuanto mayores sean los conjuntos de ancestros en las derivaciones. De aquí se puede concluir que, en general, cuanto mayor sea la elección de ancestros menor será el espacio de búsqueda asociado, ya que esta poda se aplica con mayor frecuencia.

En el siguiente ejemplo la teoría está formada por las cinco cláusulas, que se muestran explícitamente, más otras cláusulas (que no se muestran) que producen una demostración de gran longitud para el predicado *long*. El ejemplo es interesante porque ilustra que, vía ciertas elecciones de ancestros, múltiples demostraciones de *long* son necesarias, y vía elecciones más adecuadas estas múltiples demostraciones no son necesarias.

Ejemplo 4.14 (Derivaciones redundantes)

Sea T la teoría formada por las cláusulas 1 a 5. La siguiente figura muestra una refutación SLT desde $\leftarrow p \wedge s$ en T vía $\{\neg q, \neg s\}$ (que es la elección óptima para este problema). La función de selección utilizada es de izquierda a derecha, primero en el cuerpo.

$p \leftarrow q \wedge long$	1
$q \leftarrow r$	2
$q \vee r \leftarrow$	3
$s \vee t \leftarrow$	4
$s \leftarrow t$	5

[18] Incluso podría ser dramáticamente mayor y provocar que no sea posible encontrar la refutación.

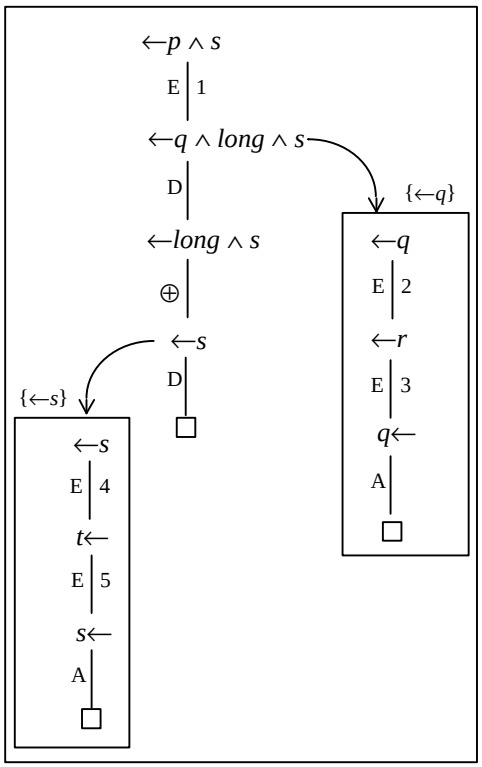


Figura 4.16: Refutación SLT vía $\{\neg q, \neg s\}$ para el Ejemplo 4.14

La refutación SLT vía $\{\neg q, \neg s\}$ contiene 2 pasos D, 5 pasos E, 2 pasos A y además una demostración para *long* (indicada con la etiqueta ⊕). Con la misma función de selección, la refutación SLT más corta vía POS contiene 6 pasos D, 19 pasos E, 2 pasos A y además 4 demostraciones para ←*long*. La refutación SLT vía ALL, sólo contiene también una demostración para *long* y además 8 pasos D, 5 pasos E y 2 pasos A. El motivo del aumento de las demostraciones efectuadas para el predicado *long* en la refutación SLT vía POS recae en la necesidad de suplir pasos de resolución de ancestro posibles por derivaciones parciales, al no haber sido seleccionado los literales correspondientes (en este caso los literales negativos). Al utilizar estas derivaciones parciales la cláusula 1 como cláusula de entrada se necesita demostrar el predicado *long*, tantas veces como usos se realicen de esta cláusula. Obviamente, en el procedimiento SLT vía ALL este fenómeno no se produce ya que todos los pasos posibles de resolución de ancestro son realizados ya que todo literal está en la elección de ancestros ALL.

De todo los dicho anteriormente se podría deducir que cuanto mayor sea la elección de ancestros más eficiente es el procedimiento vía dicha elección. Esta afirmación no es totalmente cierta. Como se ha mencionado anteriormente la elección de ancestros está totalmente relacionada con la resolución de ancestro, que es necesaria para conseguir la completitud de resolución SL*, y con la poda por igualdad con ancestro, fundamental para la reducción del espacio de búsqueda. Por otro lado, tanto la resolución de ancestro como la poda por igualdad con ancestro tienen asociada un coste computacional. Así, para aplicar cualquiera de ellas una vez seleccionado el literal en la cláusula en curso hay que comprobar que existe un elemento en

el conjunto de ancestros con el cual este literal se puede resolver o al cual sea idéntico, respectivamente. De esta forma, cuanto mayor sea el conjunto de ancestros mayor será el coste de estas operaciones. Claramente, el tamaño del conjunto de ancestros está determinado por la elección de ancestros, ya que el conjunto de ancestros aumenta de manera inversa al rango de las derivaciones. Por ejemplo, la refutación SLT desde C en T es de rango k porque existe una refutación subsidiaria de rango 0 cuyo conjunto de ancestros contiene k elementos. En cada paso de esta refutación de rango 0 se selecciona un literal y se comprueba si se puede resolver o es idéntico con un elemento del conjunto de ancestros. Claramente, esta comprobación produce una sobrecarga que en algunos casos puede no dar lugar a ninguna aplicación de un paso de resolución de ancestro o de una poda por igualdad con ancestro. Como ejemplo de un caso extremo se puede citar el siguiente: supóngase que T es una teoría Horn y C una cláusula negativa tal que $T \cup \{C\}$ es inconsistente. Claramente, existe al menos una refutación SLT desde C en T vía ALL, ya que este procedimiento es completo. Sin embargo, en ninguna refutación SLT desde C en T vía ALL se aplica un paso de resolución de ancestro, ya que la cláusula inicial C es negativa y T es Horn. De esta forma, la utilización de la elección de ancestros ALL está provocando una sobrecarga en el procedimiento que no conlleva ninguna ganancia. Para este ejemplo, en el contexto de las teorías Horn, el procedimiento más eficiente es SLT vía EMPTY si sólo se tiene en cuenta la resolución de ancestro. En caso de estar interesado en obtención de respuestas, el procedimiento a utilizar para estas teorías es SLP vía EMPTY.

A continuación se va a realizar un estudio de la conveniencia de las elecciones de ancestros según el tipo de teorías que se esté tratando. Inicialmente este estudio se realizará teniendo en cuenta que resolución de ancestro es el único uso del conjunto de ancestros. Posteriormente se extenderá a la poda por igualdad con ancestro. Este primer estudio se va a realizar atendiendo al tipo de teoría que se este tratando, que como ya se apuntaba en el párrafo anterior es esencial. A continuación se presentan los casos más claros:

ELECCIÓN DE ANCESTROS, RESOLUCIÓN DE ANCESTRO Y TIPOS DE TEORÍAS

Teorías Horn

Como ya se mencionó anteriormente los procedimientos vía EMPTY (ya sea SLT o SLP) son los más adecuados para esta clase de teorías. Por la forma de estas teorías, no es posible aplicar un paso de resolución de ancestro en ninguna derivación, siendo por tanto inútil el utilizar conjuntos de ancestros distintos del conjunto vacío.

Teorías near-Horn

Las teorías near-Horn [Loveland, 1991] están caracterizadas por la escasa presencia de cláusulas disyuntivas (aquellas que tienen más de un literal positivo). El sobrenombre de “near-Horn” proviene de este hecho. Son teorías cuyas cláusulas son en su mayoría cláusulas Horn y contienen un número muy pequeño de cláusulas que no son Horn. Para estas teorías la elección de ancestros POS es más adecuada, ya que con esta elección de ancestros los procedimientos correspondientes van a utilizar muy pocas computaciones subsidiarias y, por tanto, conjuntos de ancestros muy pequeños. En el siguiente ejemplo, en [Loveland, 1991], se presenta una teoría near-Horn y la pregunta que se desea resolver.

Ejemplo 4.15 (Elección de ancestros POS y teorías near-Horn)

Sea *T* la teoría formada por las cláusulas 1 a 8. La pregunta que se desea resolver es qué personas acuden al instituto, formalizada por la cláusulas negativa ←*acude_instituto*(*x*). La siguiente figura muestra un refutación SLP desde ←*acude_universidad*(*x*) en T vía POS.

acude_universidad(*x*) ← *título_secundaria*(*x*) ∧ *dinero*(*x*)	1	
título_secundaria(*x*) ← *estudia*(*x*)		2
título_secundaria(*x*) ← *instituto_nocturno*(*x*) ∧ *trabajo_extra*(*x*)		3
instituto_nocturno(*x*) ← *suspende_instituto*(*x*)	4	
estudia(*x*) ∨ *suspende_instituto*(*x*) ←		5
dinero(*juan*) ←		6
dinero(*enrique*) ←		7
trabajo_extra(*juan*) ←		8

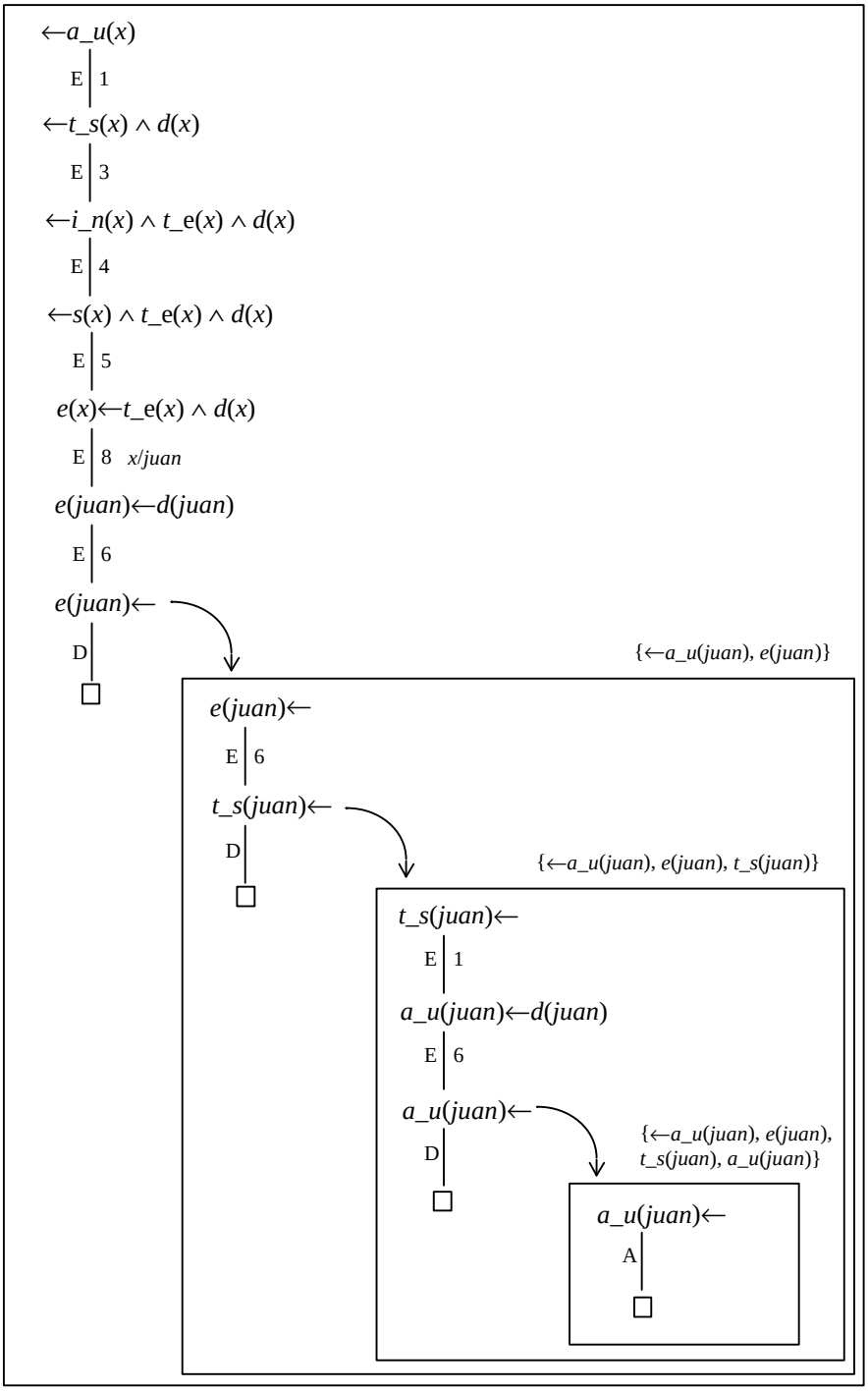


Figura 4.17: Refutación SLP vía POS para el Ejemplo 4.15

En la figura los predicados se han denotado por sus iniciales. La función de selección utilizada es de izquierda a derecha, primero en el cuerpo. La refutación SLP vía POS de la figura es de rango 3 y efectúa 9 pasos E, 1 paso A y sólo 3 pasos D. Como se puede observar, los conjuntos de ancestros utilizados son relativamente pequeños. La refutación SLP vía ALL que usa las mismas cláusulas de entrada de la teoría T es de rango 5 y efectúa 7 pasos E, 1 paso A y 7 pasos D. La diferencia en este ejemplo puede no parecer excesiva, pero hay que tener en cuenta que en los casos más reales casi todas las cláusulas son Horn y si la cláusula inicial es una

cláusula negativa[19], las refutaciones SLT o SLP vía POS que empleen cláusulas disyuntivas son muy pocas. Así, las refutaciones SLT o SLP vía POS generalmente sólo efectuarán pasos de resolución de entrada sin utilizar refutaciones subsidiarias. A diferencia de este comportamiento, si se utiliza la elección de ancestros ALL, se obtendrían refutaciones que sin usar cláusulas de entrada disyuntivas utilizarían refutaciones subsidiarias y conjuntos de ancestros diferentes del conjunto vacío pese a no realizar ningún paso de resolución de ancestro.

Teorías near-Horn de rango restringido

Este tipo de teorías suele ser el usual cuando se trabaja en el campo de las bases de datos disyuntivas. Es usual exigir a las bases de datos (en general) que cumplan la propiedad de rango restringido, de forma que no contengan cláusulas que afirmen propiedades de manera universal. Este tipo de afirmaciones se representan con cláusulas que no son de rango restringido. Así, la cláusula *hombre(x) Ú mujer(x)* ¬ expresa que toda persona ha de ser o un hombre o una mujer. Este tipo de afirmaciones no se utilizan en el campo de las bases de datos para evitar el uso de dominios generales sobre las variables. De esta forma, el ejemplo anterior se podría transformar en la cláusula de rango restringido *hombre(x) Ú mujer(x)* ¬ *persona(x)* de forma que el predicado *persona* determinaría cuales son las instancias de la variables *x* que han de cumplir esta cláusula. La propiedad de rango restringido posibilita la utilización de elecciones de ancestros denominadas *base*, que conllevarán la aplicación de la regla de la división a los literales seleccionados que sean base. Esta restricción si va unida a la construcción de derivaciones justas no implica una pérdida de la completitud. Se van a introducir las elecciones de ancestros que corresponden a esta forma de aplicación de la regla de división.

Definición 4.15 (Elección de ancestros base)

Sea *T* una teoría y *Ch* una elección de ancestros. *Ch* es una *elección de ancestros base* para *T* si para cada átomo *A* base del lenguaje subyacente a *T*, o bien dicho átomo o su negación o ambos pertenecen a *Ch*.

El siguiente teorema presenta los resultados de completitud de los procedimientos SLT y SLP vía una elección de ancestros base[20]. Como ya se mencionó en el párrafo anterior, es necesario que la función de selección sea la adecuada para alcanzar la completitud cuando se utilizan elecciones de ancestros base. Esto lleva a definir derivación SL* *justa*, que tiene bastante semejanza con la propiedad del mismo nombre que se puede encontrar en [Lloyd, 1987], aunque la finalidad sea distinta. La definición de derivación SL* justa se presenta a continuación.

[19] Esta elección siempre se puede realizar sin pérdida de generalidad.

[20] Los resultados de corrección no se incluyen ya que están subsumidos por los del Teorema 4.2, Teorema 4.3 y Teorema 4.4.

Definición 4.16 (Derivación SL* justa)

Una derivación SL* es *justa* si se cumple uno de los siguientes puntos:

i) es fallada.

ii) para cada literal L en la derivación, (alguna versión más instanciada de) L es seleccionado en un número finito de pasos.

iii) C_i es la última cláusula de la derivación, L_i es el literal seleccionado en C_i, se aplica sobre L_i un paso de la regla de división y la derivación subsidiaria desde L_i es justa.

Por extensión, se define árbol SL* justo como aquel que cumple que todas sus ramas son derivaciones SL* justas. De igual forma se define refutación SL* justa como aquella derivación SL* justa cuya última cláusula es la cláusula vacía. Ahora se está en disposición de presentar los resultados antes mencionados.

Teorema 4.6 (Completitud de los procedimientos SLT y SLP vía POS_BASE para teorías de rango restringido)

Sea T una teoría consistente, C una cláusula tal que $T \cup \{C\}$ es de rango restringido y sea POS_BASE la elección de ancestros que contiene todos los literales positivos base del lenguaje subyacente, entonces los siguientes puntos se cumplen:

a) si θ es una respuesta definida para C en T, entonces existe una refutación SLP justa desde C en T vía POS_BASE con respuesta computada θ.

b) $T \cup \{C\}$ es inconsistente, entonces existe una refutación SLT justa desde C en T vía POS_BASE.

La razón por la cual los procedimientos SLP y SLT vía una elección de ancestros POS_BASE son completos cuando se exige que la teoría sea de rango restringido y las derivaciones justas es debido a que estos dos requisitos aseguran que la aplicación de la regla de la división, indispensable para conseguir la completitud, se va a realizar en algún paso, ya que toda variable en cualquier derivación va a instanciarse.

Claramente, vía una elección de ancestros base, ni SLT ni SLP son completos para teorías que no son de rango restringido. Por ejemplo, ni SLT ni SLP pueden refutar $\leftarrow p(x) \wedge q(x)$ en la teoría $\{p(x) \leftarrow q(x), q(x) \leftarrow p(x), p(x) \vee q(x) \leftarrow\}$, aunque $\forall(p(x) \wedge q(x))$ es consecuencia lógica de dicha teoría. Por otra parte, la necesidad de utilizar derivaciones justas queda patente en la incapacidad del procedimiento SLT, vía la elección de ancestros base compuesta por todos los literales positivos base, de refutar $\leftarrow p(x) \wedge q(x) \wedge s(x)$ en la teoría formada por $\{ p(x) \leftarrow q(x), q(x) \leftarrow p(x), p(x) \vee q(x) \leftarrow, s(a) \leftarrow\}$ si la función de selección es de izquierda a derecha y para la construcción de la siguiente cláusula en curso se colocan los literales que provienen de la cláusula padre lejano en el lugar del literal seleccionado.

Las ventajas que ofrece el utilizar las elecciones de ancestros base son tres: primero, la posibilidad de omitir la aplicación de la sustitución computada al conjunto de ancestros en cada paso de la derivación, ya que los ancestros son literales base (excepto en el caso de la cláusula inicial que se podría tratar de forma diferenciada); segundo, la fácil aplicación de la regla de división en la derivación que llamó a la refutación subsidiaria, ya que esta aplicación se reduce a eliminar de la cláusula actual el literal seleccionado que es raíz de la refutación subsidiaria; y tercero, la sencilla paralelización de la regla de la división, ya que al ser el literal seleccionado base, la derivación subsidiaria puede ser computada en paralelo con la de rango superior. Esta última ventaja es la más importante ya que permite un incremento considerable de la eficiencia de resolución SL*. Esto queda patente en el siguiente ejemplo.

Ejemplo 4.16 (Paralelización de las computaciones subsidiarias)

Sea *T* la teoría formada por las cláusulas 1 a 6 del Ejemplo 4.6.

$p(x) \leftarrow q(x) \wedge r(x)$	1
$q(x) \leftarrow s(x)$	2
$q(f(x)) \leftarrow t(x)$	3
$r(a) \vee t(b) \leftarrow$	4
$t(x) \vee s(x) \leftarrow r(x)$	5
$\leftarrow q(x) \wedge t(y)$	6

La siguiente figura muestra una refutación SLT desde $\leftarrow p(x)$ en *T* usando POS_BASE. La función de selección utilizada es de izquierda a derecha y primero en la cabeza. La refutación ilustra una posible forma de computación de la regla de división en paralelo de la siguiente forma: mientras no exista un átomo base en la cabeza de la cláusula actual aplicar el paso de regla de inferencia correspondiente (o resolución de entrada o de ancestros); si existe un átomo base o más en la cabeza de la cláusula actual aplicar en paralelo sobre cada uno de ellos un paso de la regla de división y, si existe algún otro átomo seleccionado por la función de selección, aplicar también en paralelo un paso de la regla de inferencia correspondiente (o resolución entrada o de ancestros).

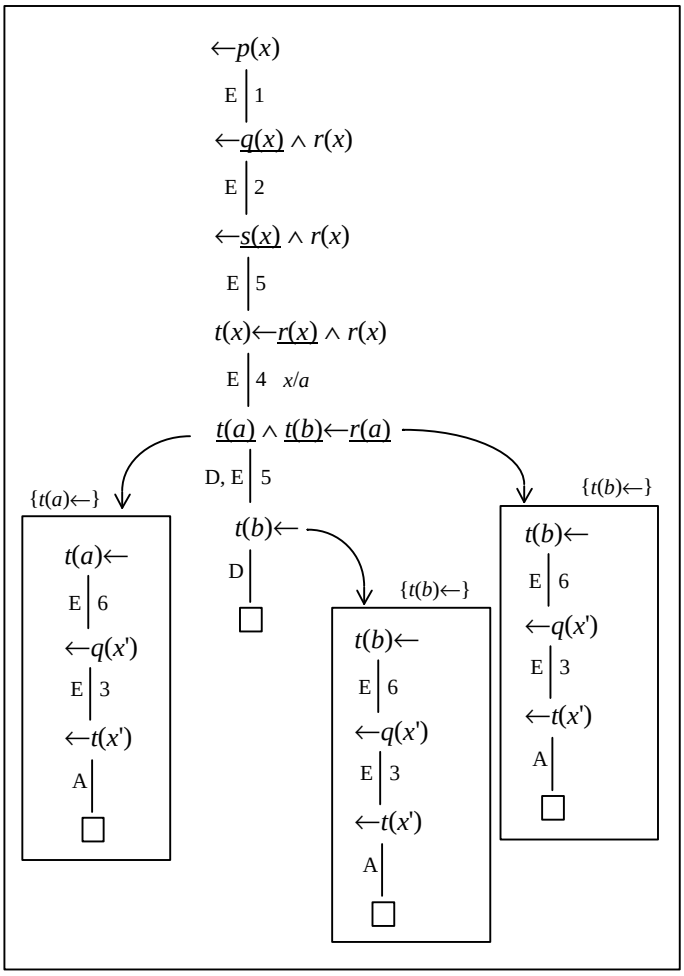


Figura 4.18: Refutación SLT vía POS_BASE para el Ejemplo 4.5

Los cuatro primeros pasos de esta refutación son pasos de resolución de entrada. En la quinta cláusula en curso se puede observar la aplicación en paralelo de dos pasos de la regla de división sobre los literales $t(a)\leftarrow$ y $t(b)\leftarrow$ y de un paso de resolución de entrada sobre el literal en la cláusula $\leftarrow r(a)$. De esta forma, la derivación principal no ha de detenerse esperando que la computación subsidiaria dé lugar a una refutación, sino que inmediatamente después de realizar la llamada a la computación subsidiaria puede continuar aplicando algún otro paso de otra regla de inferencia. Evidentemente, esta computación en paralelo de las derivaciones subsidiarias y la derivación que las lanzó provoca un incremento de la eficiencia considerable.

Finalmente, es interesante destacar que para las teorías que se están tratando, teorías near-Horn de rango restringido, la elección de ancestros más adecuada es la elección POS_BASE, introducida en el Ejemplo 4.16. La razón recae en los siguientes dos argumentos:

- las teorías son near-Horn, por tanto, la elección de ancestros adecuada es POS, como ya se vio en el apartado anterior. Por otro parte,

- las teorías son de rango restringido, por tanto, las elecciones de ancestros adecuadas tiene que ser base, esto es, sus elementos son literales base.

De estos dos puntos anteriores se puede deducir que la elección de ancestro más adecuada para las teorías near-Horn de rango restringido es POS_BASE. Añadir, además, que usualmente las bases de datos disyuntivas son teorías de este tipo, siendo, por tanto, los procedimientos SLT vía POS_BASE y SLP vía POS_BASE los más adecuados para la comprobación de la integridad u obtención de respuestas indefinidas y para la obtención de respuestas definidas, respectivamente, en este tipo de bases de datos.

Teorías que no sean Horn ni near-Horn

En estas teorías la presencia de cláusulas disyuntivas es considerablemente numerosa. Este hecho implica que la posibilidad de aplicar un paso posible de resolución de ancestro a lo largo de una derivación SL* sea bastante alta. Por tanto, la elección de ancestros ALL es, dentro de las elecciones de ancestros vistas hasta ahora, la más adecuada para este tipo de teorías.

El último punto que se va a tratar sobre la elección de ancestros está relacionado con la posibilidad de determinar la elección de ancestros óptima para un problema dado. En el Ejemplo 4.7, y Ejemplo 4.14 se han utilizado elecciones de ancestros "óptimas" con respecto a la resolución de ancestro. Estas elecciones son óptimas porque no existe ninguna elección de ancestros con menos elementos que, para el mismo problema y la misma función de selección, dé lugar a una refutación en la cual se aplican todos los pasos posibles de resolución de ancestro. Como ya se ha comentado en los párrafos anteriores la elección de ancestros ALL posibilita la aplicación de todo paso posible de resolución de ancestro, pero a costa de la sobrecarga de mantener conjuntos de ancestros con un número de elementos elevado y de aplicar un paso de la regla de división cada vez que un literal es seleccionado. Por otra parte, elecciones de ancestros completas más pequeñas, como POS, pueden mejorar esta sobrecarga pero tienen el inconveniente de no realizar pasos posibles de resolución de ancestro, con el consiguiente perjuicio sobre la eficiencia. En lo que resta de este apartado, se va a presentar en primer lugar una caracterización de la elección de ancestros óptima para un problema dado, que a pesar de no ser computable clarifica las elecciones de ancestros buscadas. En segundo lugar se presentará un algoritmo capaz de computar una elección de ancestros que aunque es posible que no sea la elección de ancestros óptima para el problema dado se puede considerar una elección de ancestros subóptima.

Para ilustrar el problema que se está tratando se presenta un sencillo ejemplo.

Ejemplo 4.17 (Elección de ancestros subóptima)

Sea T la teoría $\{p\leftarrow q,\ q\leftarrow r,\ q \vee r\leftarrow\}$. El árbol SLT desde $\leftarrow p$ en T vía ALL contiene dos refutaciones, como muestra la figura de abajo.

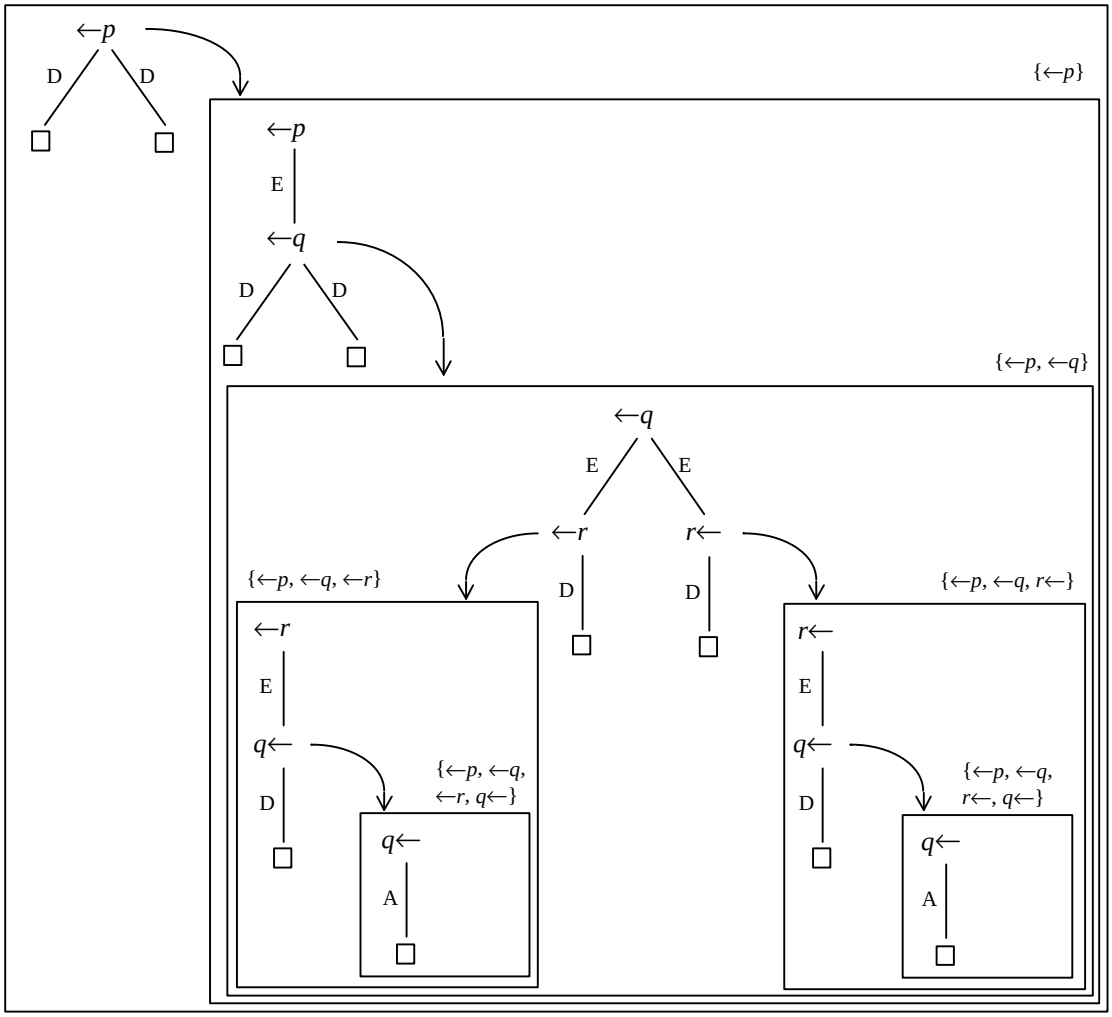


Figura 4.19: Árbol SLT vía ALL para el Ejemplo 4.17

En este árbol SLT se puede observar que el único paso de resolución de ancestro que se aplica utiliza el ancestro $\leftarrow q$. Además después se verá que este es el único paso de resolución de ancestro que se puede aplicar. Por tanto, $\neg q$ debería ser el único el literal utilizado como ancestro, siendo así, el único literal sobre el cual se aplique la regla de la división. La siguiente figura muestra el árbol SLT desde $\leftarrow p$ en T vía $\{\neg q\}$ que es la elección de ancestros óptima para este problema.

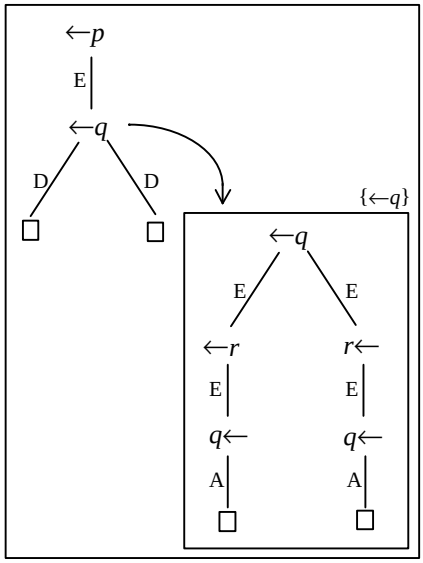


Figura 4.20: Árbol SLT vía $\{\neg q\}$ para el Ejemplo 4.17

Como se puede observar el árbol SLT vía $\{\neg q\}$ también tiene dos refutaciones, pero que únicamente contienen un paso de la regla de división que se aplica sobre el literal $\neg q$. Evidentemente las refutaciones son más simples, con menos pasos de reglas de inferencia y los conjuntos de ancestros utilizados son más pequeños. Claramente la elección de ancestros $\{\neg q\}$ es más conveniente que ALL para este problema.

La pregunta obvia es: ¿cómo se puede saber si un literal ha de estar o no en la elección de ancestros? La respuesta a esta pregunta es que un literal ha de estar en la elección de ancestros si sobre el se puede aplicar un paso de resolución de ancestro en alguna derivación que se inicia desde la cláusula raíz. Esto puede parecer obvio, pero no es así, ya que justamente en las condiciones bajo las cuales se realiza una aplicación de un paso de resolución de ancestro se puede encontrar la forma de determinar si un literal ha de estar o no en la elección de ancestros. Para simplificar la explicación se supondrá que únicamente se realizan pasos de resolución de entrada. Analizando la aplicación de un paso de resolución de ancestro se puede ver que su aplicación implica el siguiente proceso: se selecciona un literal L_i en la cláusula C_i y se realiza un paso de resolución de entrada utilizando una cláusula de entrada C'_{i+1} sobre L_i y un literal L'_{i+1} de C'_{i+1}; más adelante, se selecciona un literal L_j en la cláusula en curso C_j, $j>i$ (posterior en la derivación a C_i), que es complementario a L_i y de forma que los átomos de L_i y L_j son unificables. Si se toma las cláusulas de entradas utilizadas en la derivación desde C_i hasta C_j se puede construir una derivación de entrada desde C'_{i+1} de forma que la última cláusula de la derivación contiene los literales L'_{i+1} y L_j (o una instancia de ellos) tal que son unificables, esto es sobre ellos se puede aplicar una operación de factorización. Veámoslo en el Ejemplo 4.17, construyendo la derivación de entrada correspondiente:

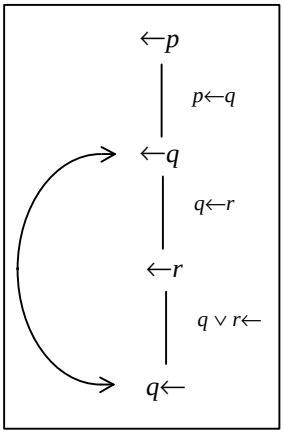


Figura 4.21: Derivación de entrada para el Ejemplo 4.17

Como se puede observar, $\neg q$ es seleccionado en la segunda cláusula en curso y sobre el se realiza un paso de resolución de ancestro porque existen dos cláusulas $q \leftarrow r$ y $q \vee r \leftarrow$ que están conectadas mediante una derivación de entrada. Como ya se comentó anteriormente, es posible aplicar una factorización sobre el literal complementario a $\neg q$ en el resolvente de las dos cláusulas, es decir $q \vee q \leftarrow$. Siguiendo estas ideas se va a dar una caracterización sintáctica de estas elecciones de ancestros que se denominará *elección de ancestros óptima para la resolución de ancestro*.

Definición 4.17 (Elección de ancestros óptima para la resolución de ancestro)

Sea T una teoría y C una cláusula. Una elección de ancestros Ch es una *elección de ancestros óptima para la resolución de ancestro* de C en T si cumple los siguientes dos puntos:

i) Si existen dos cláusulas C_1 y C_2 en $T \cup \{C\}$ tal que contienen dos literales L_1 y L_2 y hay una derivación de entrada desde C_1 en la que se usa como cláusula de entrada C_2 de forma que en la última cláusula de la derivación es posible aplicar una factorización sobre los literales L'_1 y L'_2 donde L'_1 y L'_2 son instancias de L_1 y L_2, y además existe un literal L resoluble con L'_1 y L'_2 tal que L se encuentra en una cláusula de entrada usada en una derivación de entrada desde C en T, entonces L ha de estar en Ch.

ii) Dado un literal $L \in Ch$, se cumple que todo literal instancia de L también pertenece a Ch.

Claramente, los literales que se encuentran en una elección de ancestros óptima para la *resolución de ancestro* son aquellos sobre los cuales se pueden aplicar pasos posibles de *resolución de ancestro* en SL*. Así, dado un cierto problema, para toda refutación SLT (o SLP) vía ALL existe una refutación SLT (o SLP) vía la elección de ancestros óptima que realiza los mismos pasos de *resolución de ancestro* y un número menor o igual de pasos de la regla de división (usando, por tanto, unos conjuntos de ancestros más pequeños). Por otra parte, el

siguiente teorema asegura que los resultados de completitud de SLP y SLT vía una elección de ancestros completa se mantienen vía una elección de ancestros óptima.

Teorema 4.7 (Completitud del SLP y SLT vía elección de ancestros óptima para la resolución de ancestro)

Sea *T* una teoría, *C* una cláusula y *Ch* la elección de ancestros óptima para la resolución de ancestro de *C* en *T* . Los siguientes puntos aseguran la completitud de los procedimientos SLP y SLT vía *Ch*.

a) Si *T* es consistente y θ es una respuesta definida para *C* en *T*, entonces existe una refutación SLP desde *C* en *T* vía *Ch* con respuesta computada ϕ y una sustitución σ tal que $\theta=\phi\sigma$.

b) Si *T* es consistente y $T \cup \{C\}$ es inconsistente, entonces existe una refutación SLT desde *C* en *T* vía *Ch*.

c) Si *T* es consistente y Θ es una respuesta para *C* en *T*, entonces existe una refutación SLT *C* en *T* vía *Ch* con respuesta computada Φ tal que, para cada ϕ en Φ, existe una sustitución θ en Θ y una sustitución σ tal que $\theta=\phi\sigma$.

Evidentemente, la computación de una elección de ancestros óptima para la resolución de ancestro asociada a un problema dado aplicando una técnica que se corresponda con la Definición 4.17 puede que no termine. Para solucionar este problema, la idea es tratar de encontrar una elección de ancestros que, a pesar de no ser la óptima, se aproxime a ella. La forma de construir esta elección de ancestros, que se denominará *elección de ancestros subóptima para la resolución de ancestro*, viene dada por la siguiente definición.

Definición 4.18 (Elección de ancestros subóptima para la resolución de ancestro)

Sea *T* una teoría, *C* una cláusula y $\tilde{C}\tilde{h}$ la elección de ancestros óptima para resolución de ancestro de $\tilde{C}$ en $\tilde{T}$, donde $\tilde{C}$ es la cláusula proposicional correspondiente a C y $\tilde{T}$ es la teoría proposicional correspondiente a *T*. Una elección de ancestros *Ch* es una *elección de ancestros subóptima para la resolución de ancestro* de *C* en *T* si para todo literal $\tilde{L}$ en $\tilde{C}\tilde{h}$ se cumple que todas las variantes de todas las instancias del literal *L* pertenece a *Ch*, de forma que *L* es el literal con el mismo símbolo de predicado y mismo signo que $\tilde{L}$ y totalmente desinstanciado.

Proposición 4.1

Sea *T* una teoría y *C* una cláusula. Si *OP* es la elección de ancestros óptima para la resolución de ancestro de *C* en *T* y *SOP* es la elección de ancestros subóptima para la resolución de ancestro de *C* en *T*, entonces $OP \subseteq SOP$.

Esta proposición asegura que todos los resultados de completitud dados en el Teorema 4.7 se siguen cumpliendo cuando se usa la elección de ancestros subóptima. Por otro lado, si se está trabajando con un problema proposicional, la elección de ancestros óptima y subóptima coinciden. Evidentemente, el cálculo de la elección de ancestros subóptima si que se puede realizar de forma finita, ya que se está trabajando sobre la versión proposicional del problema. Para el Ejemplo 4.6 se puede comprobar que la elección de ancestros óptima contiene a $r(x)$, $t(x)$, $\neg t(x)$ y todos los literales variantes de $r(x)$, $t(x)$ y $\neg t(x)$. La elección de ancestro subóptima contiene a $r(x)$, $t(x)$, $\neg t(x)$, $q(x)$, $\neg q(x)$, todas sus instancias y las variantes de éstas. En este ejemplo la elección de ancestro subóptima es bastante peor ya que incluye los literales provenientes de los literales proposicionales q y $\neg q$ sobre los que nunca se realizan pasos de resolución de ancestro por problemas de unificación.

ELECCIÓN DE ANCESTROS Y PODA POR IGUALDAD CON ANCESTRO

Para acabar esta sección vamos a retomar el tema de la relación entre la elección de ancestros y la poda por igualdad con ancestro. Como se mencionó anteriormente la elección de ancestros ALL es la que permite aplicar más veces la poda por igualdad con ancestro, ya que esta elección da lugar a aplicar sobre cada uno de los literales seleccionados la regla de división, incorporándolos al conjunto de ancestros y haciendo más probable que se pueda seleccionar un literal en la computación que sea idéntico a alguno de ellos. Sin embargo, hay que tener también en cuenta que, como desventaja, los conjuntos de ancestros que hay que manejar en las computaciones vía ALL son mayores, produciéndose así una sobrecarga que, en el caso peor, puede no dar lugar a la aplicación de ninguna poda. Es por tanto interesante estudiar si existe alguna forma de determinar a priori cuales son los literales que pueden dar lugar a la aplicación de la poda por igualdad con ancestro. La respuesta a esta pregunta la da la propia esencia de la poda por igualdad con ancestro. Como se mencionó anteriormente la poda por igualdad con ancestro está relacionada con la reducción de tautologías en las computaciones. Así, siempre que se realice una poda por igualdad con ancestro es posible detectar una tautología en alguna de las cláusula de la teoría o en un resolvente de éstas. Esta relación ya se presentó en el Ejemplo 4.9.

La conexión existente entre la poda por igualdad con ancestro y la reducción de tautologías va permitir determinar los literales que han de incluirse en la elección de ancestros para permitir la aplicación de la poda siempre que sea posible. La idea es similar a la determinación de la elección de ancestros óptima pero en este caso en vez de buscar factorizaciones se buscarán tautologías. La siguiente definición formaliza estas ideas.

Definición 4.19 (Elección de ancestros óptima para la poda por igualdad con ancestro)

Sea T una teoría y C una cláusula. Una elección de ancestros Ch es una *elección de ancestros óptima para la poda por ancestros* de C en T si cumple los siguientes dos puntos:

i) Si existen dos cláusulas C_1 y C_2 en $T \cup \{C\}$ tal que contienen dos literales L_1 y L_2 y hay una derivación de entrada desde C_1 en la que se usa como cláusula de entrada C_2 de forma que en la última cláusula de la derivación contiene dos literales L'_1 y L'_2 idénticos y de signo opuesto, donde L'_1 y L'_2 son instancias de L_1 y L_2, y además existe un literal L resoluble con L'_1 y L'_2 tal que L se encuentra en una cláusula de entrada usada en una derivación de entrada desde C en T, entonces L ha de estar en Ch.

ii) Dado un literal $L \in Ch$, se cumple que todo literal instancia de L también pertenece a Ch.

En la definición anterior se puede observar que el objetivo es conseguir que la elección de ancestro incluya todos los literales (y las variantes de éstos) tales que sea posible generar un resolvente de cláusulas de la teoría inicial que sea una tautología por aparecer en él el literal y su opuesto. El siguiente ejemplo ilustra esta definición.

Ejemplo 4.18 (Elección de ancestros y poda por igualdad con ancestro)

Sea S el siguiente conjunto de cláusulas:

$p \vee q \leftarrow$	1
$\leftarrow p \wedge q$	2
$r \leftarrow t \wedge p \wedge q$	3
$t \leftarrow$	4

La figura siguiente presenta el árbol SLT desde $\leftarrow r$ en S vía la elección de ancestro $\{\neg p, \neg q\}$, que es la que se obtiene aplicando la Definición 4.19, que desde el punto de vista de la poda por igualdad con ancestro es la óptima.

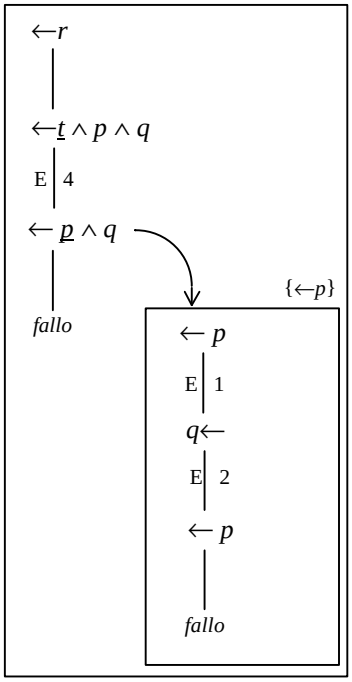


Figura 4.22: Árbol SLT para el Ejemplo 4.19

Es interesante destacar por otro lado que ya que no es posible realizar ninguna resolución de ancestro en el ejemplo anterior la elección de ancestros es la óptima desde las dos perspectivas: la resolución de ancestro y la poda por igualdad con ancestro.

Como ocurría con la elección de ancestros óptima para la resolución de ancestro, no es posible obtener a priori la elección de ancestros óptima para la poda por igualdad con ancestro. De la misma forma, se puede trabajar con una elección de ancestros que se aproxime a la óptima y que sea posible computarla en tiempo de compilación. A esta elección de ancestros se la denomina *elección de ancestros subóptima para la poda por igualdad con ancestro* y a continuación se presenta su definición.

Definición 4.20 (Elección de ancestros subóptima para la poda por igualdad con ancestro)

Sea T una teoría, C una cláusula y $\widetilde{C}\widetilde{Ch}$ la elección de ancestros óptima para la poda por igualdad con ancestro de $\widetilde{C}$ en $\widetilde{T}$, donde $\widetilde{C}$ es la cláusula proposicional correspondiente a C y $\widetilde{T}$ es la teoría proposicional correspondiente a T. Una elección de ancestros Ch es una *elección de ancestros subóptima para la poda por igualdad con ancestro* de C en T si para todo literal $\widetilde{L}$ en $\widetilde{C}\widetilde{Ch}$ se cumple que todas las variantes de todas las instancias del literal L pertenece a Ch, de forma que L es el literal con el mismo símbolo de predicado y mismo signo que $\widetilde{L}$ y totalmente desinstanciado.

Proposición 4.2

Sea *T* una teoría y *C* una cláusula. Si *OP* es la elección de ancestros óptima para la poda por igualdad con ancestro de *C* en *T* y *SOP* es la elección de ancestros subóptima la poda por igualdad con ancestro de *C* en *T*, entonces $OP \subseteq SOP$.

Finalmente, en los experimentos se utilizarán los procedimientos vía diferentes elecciones de ancestros para obtener resultados que confirmen las argumentaciones teóricas que se han dado hasta ahora. Así, se utilizaran procedimientos vía EMPTY, POS, ALL y la elección de ancestros subóptima para la resolución de ancestro y la poda por igualdad con ancestro. De aquí en adelante siempre que se haga referencia a la elección de ancestro óptima o subóptima se entenderá que es la elección de ancestros óptima o subóptima, respectivamente, tanto para resolución de ancestro como para la poda por igualdad con ancestro.

4.4. IMPLEMENTACIÓN DE RESOLUCIÓN SL*

Resolución SL* y los distintos procedimientos instancias han sido implementados en Prolog mediante metaprogramas que siguen, en líneas generales, la definición de resolución SL*. Las diferencias fundamentales entre los distintos metaprogramas utilizados consisten en las distintas elecciones de ancestros utilizadas, los distintos criterios de recorrido del espacio de búsqueda, las distintas podas utilizadas y, por último, las funciones de selección probadas. Las características fundamentales de la implementación realizada son las siguientes:

a) Implementación mediante metaprogramas en Prolog.
b) El algoritmo de unificación utilizado incluye el test de comprobación de ocurrencia.
c) Uso de diferentes elecciones de ancestros utilizadas: ALL, POS, EMPTY y la subóptima.
d) La estrategia de recorrido del espacio de búsqueda es en profundidad iterada.
e) Utilización de diferentes funciones de selección de literales.
f) Aplicación de distintas podas.

Se va a discutir cada una de estas características con más detalle a continuación:

4.4.1. IMPLEMENTACIÓN MEDIANTE METAPROGRAMAS EN PROLOG

La implementación de resolución SL* se ha realizado mediante un metaprograma en Prolog (tipo intérprete Vanilla [Lloyd, 1987]), siguiendo básicamente la definición de refutación SL*. Las distintas variantes de dicho metaprograma utilizan diferentes estrategias de recorrido del espacio de búsqueda, diferentes funciones de selección y distintas podas. El motivo de la utilización de la metaprogramación para la implementación ha sido la mayor flexibilidad, legibilidad y modificabilidad que proporciona.

La implementación de resolución SL* sigue las ideas que M. Stickel ha aplicado en su demostrador PTTP [Stickel, 1988] [Stickel, 1992] que ya se comentaron en el capítulo 3, pero obteniendo un metaprograma. Como en el PTTP existe una fase de precompilación en la cual se transforma el problema original, un conjunto de cláusulas, en un conjunto de reglas Prolog. Estas reglas Prolog serán cargadas por el demostrador, también implementado mediante un programa Prolog, que, ya en ejecución, tratará de demostrar su inconsistencia. Así, la diferencia esencial reside en que Stickel obtiene como resultado de la precompilación un programa Prolog que incorpora el problema a demostrar y el demostrador como un todo. El proceso de demostración en el PTTP es sencillamente la ejecución de dicho programa. El autor de la presente tesis cree que la aproximación que se ha tomado es más adecuada ya que permite cambiar el comportamiento de los demostradores, modificando la elección de ancestros, o la función de selección o la limitación de la profundidad, sin más que modificando el programa Prolog que implementa el demostrador y sin necesidad de precompilar los problemas, permitiendo de este modo una mayor flexibilidad y modificabilidad en el sistema.

A continuación se procede a relatar las operaciones que se realizan en la fase de precompilación. Esencialmente en esta fase se intenta transformar el problema original, un conjunto de cláusulas, a un formato que, teniendo en cuenta que la implementación se ha realizado en Prolog, aproveche al máximo las características de Prolog (indexación de las reglas, función de selección, etc.). En la implementación realizada, esta transformación también va a permitir incluir información que permita realizar operaciones del proceso de demostración (reglas de inferencia, estrategias de búsqueda, podas, etc.) de una forma correcta y adecuada. Este proceso de precompilación tiene en todos los casos un coste despreciable, incluso en los problemas más extensos. En la fase de precompilación se realizan las siguientes operaciones:

- Obtención de las contrapositivas de las cláusulas originales, para aprovechar la indexación del Prolog.
- Generación del código necesario para la realización del test de ocurrencia en la unificación.
- Obtención y almacenamiento del tamaño de las cláusulas, útil para realizar la búsqueda en profundidad iterada con limitación por el número de pasos de inferencia realizados.
- Transformación del problema para obtener la cláusula inicial.
- Obtención de la elección de ancestros subóptima.

Existen muchos otros procesos a realizar en la precompilación que no se han incluido, tal vez el más significativo sea un proceso de simplificación y de reducción del problema. Como se apunta en [Letz, Schuman, Bayerl y Bibel, 1992] este proceso de simplificación y reducción es muy importante ya que en muchos casos consigue reducir notablemente el problema, incluso demostrarlo, con un coste mínimo.

OBTENCIÓN DE LAS CONTRAPOSITIVAS

En las diferentes implementaciones de los distintos procedimientos instancias de resolución SL*, una contrapositiva de una cláusula se implementa mediante un instrucción Prolog que incluye código añadido para realizar diferentes operaciones en ejecución. Así, por ejemplo, la cláusula:

$p(x, g(x)) \vee q(f(y,f(a)) \leftarrow t(a,b)$

y una contrapositiva de ella:

$p(x, g(x)) \leftarrow \neg\, q(f(y,f(a)) \wedge t(a,b)$

se representa como la aserción Prolog:

$p(p(X, g(X)), [n\_q(f(Y,f(a)), t(a,b)]).$

donde los símbolos de variables del lenguaje original se representan mediante variables del lenguaje Prolog. Como se puede observar cada contrapositiva se representa mediante un regla Prolog que no tiene cuerpo y que se construye de acuerdo a los siguientes puntos:

a) el símbolo del predicado de la cabeza de la contrapositiva es igual al símbolo del predicado del átomo de Prolog (en el ejemplo *p*).
b) la cabeza de la contrapositiva coincide con el primer argumento del átomo Prolog (en el ejemplo $p(X, g(X))$).
c) el cuerpo de la contrapositiva se representa en el segundo argumento de la regla Prolog como una lista de átomos (en el ejemplo $n\_q(f(Y,f(a)), t(a,b))$).

Las razones para utilizar la representación arriba indicada son las siguientes:

- Como se desea poder utilizar diferentes funciones de selección de forma sencilla, es necesario el control de la ejecución del Prolog, haciendo inviable la utilización del formato PTTP.

- Ya que el sistema Prolog utilizado indexa las reglas por el símbolo del predicado de la cabeza y su primer argumento, el formato elegido permite la realización de la resolución de entrada de forma eficiente.

- Se representa la negación mediante la modificación del símbolo de predicado, de esta forma $\neg\, q(f(Y,f(a)), t(a,b))$ se representa como $n\_q(f(Y,f(a)), t(a,b))$. La razón de esta elección en vez de utilizar un operador es el reducir al mínimo la descomposición de términos durante la ejecución.

TEST DE OCURRENCIA EN LA UNIFICACIÓN

Para evitar la necesidad de la utilización de unificación correcta (que incorpore el test de ocurrencia) se ha optado por aplicar una solución similar a la de Stickel en el PTTP [Stickel, 1992]. Durante la precompilación detectar aquellos literales que requieren el test de ocurrencia para aplicar sobre ellos una unificación correcta, y transformar adecuadamente la regla para que sea posible la aplicación de una unificación sin test de ocurrencia de forma general. Si se toma la cláusula del apartado anterior:

$$p(x, g(x)) \vee q(f(y,f(a)) \leftarrow t(a,b)$$

es obvio que para realizar unificaciones sobre el literal $p(x, g(x))$ es necesario la utilización de una unificación con test de ocurrencia. La manera de transformar esta cláusula es cambiar la representación de la contrapositiva que tiene como cabeza a uno de estos literales. Así, en este caso la contrapositiva a transformar es:

$$p(x, g(x)) \leftarrow \neg\, q(f(y,f(a)) \wedge t(a,b)$$

que se transforma de la forma siguiente:

$$p(p(X, g(X1)), [n\_q(f(Y,f(a)), t(a,b)], [ocu(X, X1)]).$$

Como se puede observar se ha incluido un nuevo argumento en la regla Prolog que representa a la contrapositiva que incluye las parejas de variables sobre las cuales es necesario realizar una unificación correcta. La manera de detectar estas parejas es la siguiente: cuando una variable x ocurre dos o más veces en la cabeza de la contrapositiva, estas ocurrencias se eliminan introduciendo nuevas variables $x1$, $x2$, etc. sobre las cuales después habrá que realizar una unificación correcta con la variable x. De esta forma se evita el tener que utilizar de forma general una unificación con test de ocurrencia, disminuyendo por tanto el coste de dicha operación en el demostrador.

OBTENCIÓN Y ALMACENAMIENTO DEL TAMAÑO DE LAS CLÁUSULAS

Esta operación consiste en contar el número de literales de las cláusulas iniciales y almacenar esta información en las reglas Prolog correspondientes a las contrapositivas de dicha cláusula. Esta información se introduce como un argumento más en dichas reglas. Si se toma la cláusula que se ha usado como ejemplo en las operaciones anteriores:

$$p(x, g(x)) \vee q(f(y,f(a)) \leftarrow t(a,b)$$

una de las contrapositivas de esta cláusula se representa por la siguiente regla Prolog:

$$p(p(X, g(X1)), [n\_q(f(Y,f(a)), t(a,b)], 3).$$

El número de literales de la cláusula se utilizará durante la ejecución para implementar de forma eficiente la búsqueda iterada con limitación por el número de pasos de inferencia. Además también será útil para implementar algunas podas asociadas a esta forma de búsqueda que posteriormente se comentarán.

TRANSFORMACIÓN DEL PROBLEMA PARA OBTENER LA CLÁUSULA INICIAL.

Como se comentó anteriormente en los sistemas basados en resolución lineal una condición asociada a la completitud es la exigencia de, bien que el problema sea mínimamente inconsistente, bien que se identifique la cláusula inicial de forma que la teoría (sin incluir la cláusula inicial) sea consistente y junto con la cláusula inicial no lo sea. En el primer caso cualquier cláusula de la teoría puede ser utilizada como cláusula inicial y en el segundo, únicamente la que se identifica como tal. Resolución SL* no es una excepción y requiere que se cumpla esta exigencia. En el caso en el cual se sepa cuál es la cláusula inicial (bien porque el usuario la proporciona o porque es fácilmente identificable) simplemente se usa dicha cláusula como inicial. En caso de no saber de forma explícita cuál es la cláusula inicial es necesario transforma el problema para obtener una cláusula inicial y una teoría restante que cumpla estas condiciones. Para ello en la precompilación se transforma adecuadamente para lograr este objetivo. En los problemas tratados se han distinguido dos casos:

a) Problemas sin conocimiento semántico de los mismos: supongamos que la teoría original es *T*. La transformación consiste en la transformación de las cláusulas negativas de *T* de forma que cada una de ellas se cambia por una cláusula Horn cuya cabeza es un átomo proposicional cuyo predicado no aparecía en el lenguaje soporte del problema original. Así la cláusula negativa siguiente:

   $$\leftarrow p(x, a) \wedge q(f(x))$$

   se transforma a la cláusula Horn:

   $$inicial \leftarrow p(x, a) \wedge q(f(x))$$

   donde *inicial* es un predicado nuevo que no está en el lenguaje soporte de la teoría inicial. De esta forma, la teoría obtenida, llámesele *T'*, es consistente y se cumple que $T' \cup \{\leftarrow inicial\}$ es inconsistente si y sólo si *T* lo es. Así la cláusula inicial de la demostración es $\leftarrow inicial$.

b) Problemas con conocimiento semántico de los mismos: usualmente es el caso de problemas de ámbitos formales en los cuales están claramente identificadas las teorías formales que se utilizan, los axiomas, las hipótesis, los teoremas a demostrar, etc. Éste es el caso de los problemas utilizados en las pruebas experimentales de la presente tesis

y que corresponden a lo que se conoce como la librería TPTP (acrónimo de *Thousand Problems on Theorem Proving*) [Suttner y Sutcliffe, 1996]. Para este tipo de problemas se puede utilizar el método del punto a), aunque se ha considerado más adecuado aplicar el siguiente: supóngase que el problema consiste en una teoría *T* cuyas cláusulas se pueden identificar como axiomas *AX*, hipótesis H_1, H_2, …, H_n y la conclusión *C*, donde tanto las hipótesis como la conclusión son literales (en otros palabras se desea comprobar que $AX \models H_1 \wedge H_2 \ldots \wedge H_n \rightarrow C$), entonces, sabiendo que *AX* es consistente, el problema se reduce a saber si $AX \cup \{H_1, H_2, \ldots, H_n, \widetilde{C}\}$ es inconsistente, donde $\widetilde{C}$ es el literal complementario de *C*. Generalmente, se puede asumir que las hipótesis son consistentes con los axiomas, con lo cual la cláusula raíz sería la cláusula unitaria formada por $\widetilde{C}$. En el caso en que la asunción anterior no se pueda realizar se aplicaría una transformación equivalente a la del punto a) tanto sobre la conclusión como sobre las hipótesis.

La cláusula inicial de la demostración es denotada de la forma mediante la utilización de un metapredicado *cinicial*. De esta forma si la cláusula inicial en la demostración de un problema es la siguiente $\leftarrow p(x) \wedge q(x, a)$, el conjunto de reglas Prolog que representaran a dicho problema incluirá la siguiente regla *cinicial*([*p*(*x*), *q*(*x*, *a*)]), que será utilizada para lanzar la demostración inicialmente. Claramente, también aparecerán las reglas correspondientes a las contrapositivas de dicha cláusula en la representación en Prolog del problema, como para el resto de las cláusulas[21].

OBTENCIÓN DE LA ELECCIÓN DE ANCESTROS SUBÓPTIMA

Esta operación da como resultado la elección de ancestros subóptima atendiendo tanto a la resolución de ancestro como a la poda por igualdad con ancestro. La forma de realizar esta operación ha sido mediante la obtención del grafo de dependencias en la versión proposicional del conjunto de cláusulas iniciales y buscando en dicho grafo factorizaciones y tautologías. Los literales que están implicados en la factorización o en las tautologías son marcados en las reglas Prolog que representan a las contrapositivas de las cláusulas donde aparecen, permitiendo saber en ejecución si un literal seleccionado en la derivación está o no en la elección de ancestros y es, por tanto, necesario lanzar una computación subsidiaria cuyo conjunto de ancestros incluya dicho literal.

[21] En el caso de que se esté interesado en la resolución de problemas definidos y en utilizar, por tanto, el procedimiento SLP, las contrapositivas de la cláusula inicial no serán incluidas en la representación Prolog del problema.

La forma en la cual se realiza esta operación es la siguiente: primero se obtiene la versión proposicional del problema a tratar; segundo se realiza un análisis de esta versión proposicional detectando qué literales son aquellos que participan en las factorizaciones o en la aparición de tautologías; y tercero, dichos literales son marcados en la versión general adecuadamente al paso anterior. Este marcado se realiza incluyendo una regla Prolog que define la elección de ancestros. Por ejemplo si la elección de ancestros es $\{p(x,y), \neg q(z)\}$ se incluyen las siguientes reglas:

ancestro(*p*(_,_)).
ancestro(*n_q*(_)).

Así representada la elección de ancestros, la forma de comprobar la pertenencia del literal seleccionado en la cláusula en curso a la elección de ancestros durante la ejecución del demostrador consiste simplemente en realizar una llamada a dicha regla con el literal seleccionado como argumento.

4.4.2. UNIFICACIÓN CORRECTA

En cualquier disciplina en la cual se utilice el principio de resolución de Robinson, demostración de teoremas, programación lógica, sistemas expertos, etc., el algoritmo de unificación utilizado debería ser correcto, cumpliendo el test de ocurrencia. En particular, en la demostración de teoremas, esta característica se hace indispensable debido al tipo de problemas a demostrar. Hay que tener en cuenta que la unificación es necesaria en la aplicación de la resolución de entrada y de la resolución de ancestro. En el primer caso la utilización de la técnica detallada en la sección anterior permite reducir el coste de la utilización de una unificación correcta, detectando y transformar en el preproceso aquellas contrapositivas cuyas cabezas requieren el uso de la unificación correcta con el test de ocurrencia. En las otras contrapositivas (la gran mayoría) la unificación utilizada es la habitual de los sistemas Prolog que no incluye el test de ocurrencia. En el caso de la resolución de ancestro no existe la posibilidad de aplicar esta técnica ya que éstos son manipulados en ejecución, por lo que es necesario utilizar siempre la unificación correcta.

En la implementación realizada no ha sido necesario desarrollar la unificación correcta ya que el sistema Prolog utilizado (SICSTUS Prolog) proporciona tanto la unificación sin test de ocurrencia como la unificación correcta.

4.4.3. UTILIZACIÓN DE DIFERENTES ELECCIONES DE ANCESTROS: ALL, POS, EMPTY Y LA SUBÓPTIMA

Se han implementado demostradores que corresponden al procedimiento SLT vía distintas elecciones de ancestros. En particular se han utilizado las elecciones ALL, POS, EMPTY y la subóptima para el problema tratado. El empleo de distintas elecciones se ha plasmado en distintos tiempos, pasos de inferencia de la refutación encontrada y pasos de inferencia totales en el espacio de búsqueda recorrido que se han obtenido en la resolución de un mismo problema. Los resultados obtenidos mediante la elección ALL han podido ser comparados con resultados de otros autores [Stickel, 1988] [Letz, Schuman, Bayerl y Bibel, 1992] aunque hay que dejar claro que esta comparación no tiene demasiado interés por los diferentes contextos de implementación utilizados. Los resultados obtenidos mediante las otras elecciones de ancestros no han podido ser comparados con resultados de demostradores similares. Por otra parte la comparación de los resultados obtenidos usando las diferentes elecciones de ancestros es en sí mismo interesante, ya que proporciona algunas ideas de la adecuación de éstas a distintos tipos de problemas.

4.4.4. LA ESTRATEGIA DE RECORRIDO DEL ESPACIO DE BÚSQUEDA: EN PROFUNDIDAD ITERADA

La estrategia de recorrido del espacio de búsqueda es esencial para la consecución de un procedimiento de demostración. Así, una estrategia de recorrido inadecuada puede dar lugar a la incompletitud de la implementación de un cierto procedimiento. Éste es el caso de los sistemas Prolog comerciales que, en aras de una mayor legibilidad procedural, utilizan la estrategia de recorrido del espacio de búsqueda en profundidad. Para que un cierto sistema, implementación de un cierto procedimiento, sea completo debe asegurar que si en el espacio de búsqueda existe una respuesta, ésta es encontrada en un tiempo finito. La estrategia de recorrido del espacio de búsqueda que se ha implementado es la búsqueda en profundidad iterada, que es la solución adoptada por Stickel en su PTTP [Stickel, 1992] y en el SETHEO [Letz, Schuman, Bayerl y Bibel, 1992]. Esencialmente, esta estrategia consiste en realizar una búsqueda en profundidad acotada, o por niveles, incrementándose dicha cota o nivel sucesivamente hasta localizar la posible solución. Como ya se mencionó esta estrategia presenta la ventaja de tener un coste asintótico espacial y temporal óptimo, junto con la posibilidad de utilizar información heurística para realizar podas. En el análisis de la estrategia de búsqueda en profundidad iterada que se hizo en la presentación del PTTP en el capítulo 3 de esta tesis se mostró que si *fr* es el factor de ramificación del árbol de prueba, entonces con la estrategia de búsqueda en profundidad iterada se efectúan aproximadamente *fr(fr-1)* veces el número de operaciones que se efectúan con la estrategia de búsqueda en profundidad. Este valor puede ser reducido, si el factor de ramificación no es muy elevado.

Existen diferentes criterios para determinar la profundidad de una derivación. El uso de uno u otro y su aplicación para determinar la cota o nivel da lugar a recorridos distintos del espacio de búsqueda, lo que evidentemente se traduce en resultados temporales distintos cuando se aplican a la resolución de un mismo problema. Sin embargo, como se menciona en [Stickel, 1992] [Letz, Schuman, Bayerl y Bibel, 1992], ninguno de ellos se muestra claramente superior a los otros de forma uniforme. Los principales criterios utilizados habitualmente son los siguientes: tamaño de la lista de ancestros (o rango de la derivación en terminología de resolución SL*), número de pasos de reglas de inferencia (es decir, de pasos de resolución de entrada y de ancestros) y número de usos que se realiza de cada cláusula en la demostración. El demostrador SETHEO [Letz, Schuman, Bayerl y Bibel, 1992] hace un uso combinado de las tres estrategias en paralelo. En los procedimientos instancias de resolución SL* se pueden utilizar cualquiera de las tres estrategias con las siguientes peculiaridades:

a) Tamaño de la lista de ancestros o rango de la derivación: esta medida de la profundidad de una derivación se puede aplicar únicamente cuando se usa la elección de ancestros ALL. Usando esta elección sobre todo literal seleccionado se aplica un paso de la regla de división, de forma que este literal es incorporado al conjunto de ancestros de la derivación subsidiaria que tiene a dicho literal como raíz. Para otras elecciones de ancestros es posible, dependiendo de cual sea la función de selección elegida, que nunca se seleccione un literal que pertenezca a la elección de ancestros, teniendo dicha derivación siempre una profundidad constante que le impediría llegar a la cota máxima que esté en ese momento establecida. Por otra parte, este criterio presenta el inconveniente de provocar un crecimiento alto (a veces explosivo) del espacio de búsqueda entre niveles de profundidad sucesivos, lo que generalmente se traduce en un buen comportamiento temporal de los procedimientos que lo usan para problemas pequeños, que se ve contrarrestado por un mal comportamiento para problemas más profundos, como es el caso de problema *wos1* [Wos y Overbeek, 1993] o el Steamroller [Stickel, 1988], como se pudo comprobar en las diferentes pruebas realizadas previas a la elección de la implementación definitiva. Otro inconveniente significativo del mismo estriba en la dificultad de definir algunas podas que permitan acotar el número de pasos de inferencias de la demostración.

b) Número de pasos de reglas de inferencia (resolución de entrada y de ancestros): este criterio presenta la ventaja de un crecimiento del espacio de búsqueda entre niveles sucesivos más lento. Por este mismo motivo presenta un comportamiento temporal más flojo para problemas pequeños, al realizar habitualmente un número mayor de iteraciones de las que se efectúan utilizando el criterio anterior. Por otro lado, presenta la ventaja de poder utilizar información heurística para definir podas. Una muy sencilla implementada en el presente trabajo consiste en que para una cota máxima *n*, no intentar la demostración de la cláusula en curso cuyo número de literales sea superior a *n*. La implementación de

dicha poda exige determinar el número de literales de la cláusula en curso en tiempo de ejecución, lo cual puede realiza dinámicamente con un algoritmo de coste lineal. La aproximación utilizada, como ya se comentó, consiste en determinar el número de literales de cada cláusula de la teoría y almacenarlo en precompilación, y posteriormente en tiempo de ejecución, gestionarlo de forma adecuada, incrementado de esta forma la eficiencia. Existen posibles mejoras de esta poda aplicada, una de ellas se resume en, dada un cláusula en curso que no se puede demostrar al aplicar la poda anterior, encontrar la cláusula de menor tamaño que puede resolver con el literal seleccionado. El menor de todos los tamaños determinados de esta forma se utilizará para incrementar la cota máxima de la siguiente iteración. Evidentemente el coste de la aplicación de esta mejora es bastante alto, por lo que no se llevó a cabo su implementación.

c) Número de veces que se utiliza una cláusula de entrada: este criterio presenta el inconveniente de provocar un crecimiento explosivo del espacio de búsqueda, por lo que para problemas cuya demostración es profunda este crecimiento puede dar lugar a que no sea posible demostrarlo.

d) Criterio híbrido de los dos anteriores: provoca un comportamiento temporal intermedio para problemas pequeños y presenta en algunos casos resultados aceptables para problemas más profundos (por ejemplo, el problema Steamroller). En este criterio la profundidad aumenta cuando se aplica un paso de la regla de división (o el rango de la derivación) o un paso de resolución binaria que origine una nueva demostración. Presenta la ventaja de un crecimiento intermedio entre niveles sucesivos y la posibilidad de utilizar información heurística para la aplicación de podas.

En el actual trabajo se decidió utilizar únicamente el criterio que determina la profundidad de la búsqueda por el número de pasos de inferencia que se realizan ya que proporcionaba un crecimiento más controlado y la posibilidad de aplicar ciertas podas que no eran utilizables para los otros criterios. Estos motivos permitían prever que con este criterio se iba a conseguir demostrar un mayor número de problemas aunque, tal vez con unos resultados peores.

4.4.5. UTILIZACIÓN DE DISTINTAS FUNCIONES DE SELECCIÓN

Como es bien conocido las propiedades de corrección y completitud del procedimiento SLD son independientes de la función de selección o regla de computación [Lloyd, 1987] que se emplee. Los procedimientos instancias de resolución SL* también poseen esta propiedad. Se realizaron pruebas utilizando las siguientes funciones de selección: seleccionar el literal más a la izquierda y seleccionar el literal más instanciado (de forma total o parcial). La primera función tiene la ventaja de implementarse con coste temporal constante, las otras funciones, que se

discutirán a continuación, tienen un coste temporal lineal con el número de símbolos de la cláusula en el peor de los casos.

Generalmente, la función de selección del literal más instanciado proporciona buenos resultados experimentales, en particular en los problemas más profundos. Ello es debido a que esta función, por término medio, determina un factor de ramificación del árbol de demostración más pequeño que otras funciones de selección (por ejemplo el más a la izquierda), lo cual provoca una evidente mejora temporal. Además esta función de selección permite la aplicación de manera más frecuente de alguna de las podas que se describen en el punto siguiente, reduciéndose aún más si cabe el espacio de búsqueda asociado a esta función de selección. Por otro lado, en algunos casos que se presentarán posteriormente su uso ha supuesto un empeoramiento de los resultados.

La implementación de la función de selección del literal más instanciado puede ser temporalmente costosa. Antes de nada, cabría preguntarse qué se entiende por el literal más instanciado de una cláusula. En términos generales, se puede decir que el literal L está más instanciado que el literal L' si el conjunto de posibles instancias de L es menor que el de L'. Así, por ejemplo *p(x)* esta más instanciado que *q(y,z)*, *p(a)* está más instanciado que *p(x)*, etc. Sin embargo, es interesante destacar que *q(f(x,y),z)* está más instanciado que *q(y,z)*, aunque a primera vista no parezca así. En general, la determinación conlleva el estudio sintáctico del literal para detectar la presencia de variables ya sea como argumentos directos o como en términos argumentos del literal. Todo ello hace bastante costosa la aplicación de esta función por lo que se hace preciso aplicar alguna de las siguientes tres simplificaciones que llevan a asumir como el literal más instanciado a: primera, el literal que contiene menos variables; segunda, el primer literal por la izquierda totalmente instanciado, caso de no existir se toma el primer literal por la izquierda; tercera, el literal que contiene menos variables que son argumentos de dicho literal, esto es las variables que aparezcan en funciones que sean argumentos del literal no son tenidas en cuenta. La primera simplificación presenta un coste temporal que es lineal con el número de símbolos de la cláusula. La segunda y tercera tiene un coste temporal sublineal. En la práctica, la última simplificación es la que presenta un mejor comportamiento de entre las tres reduciendo generalmente, y a veces de forma notable, el tiempo de demostración obtenido mediante la función de selección del literal más a la izquierda.

4.4.6. APLICACIÓN DE PODAS

Además de la poda descrita en el punto de la estrategia del recorrido del espacio de búsqueda, se han implementado y probado las siguientes podas:

a) Podar aquellas derivaciones en las cuales, en un cierto paso, el literal seleccionado en la cláusula en curso es sintácticamente idéntico a un ancestro del conjunto de ancestros. Esta

poda se corresponde con punto iii) de la definición de resolución SL* modificada (Definición 4.14). Esta poda detecta la presencia de recursividad no constructiva o de tautologías en las derivaciones, podando aquellas derivaciones que no puedan dar lugar a un refutación. Esta poda, que no es necesaria para garantizar ni la corrección ni la completitud, se hace indispensable en la práctica ya que reduce de forma drástica el espacio de búsqueda.

b) Podar aquellas derivaciones en las cuales, en un cierto paso, el literal seleccionado en la cláusula en curso es subsumido por un ancestro del conjunto de ancestros. Esta poda se aplica con más frecuencia que la anterior, lo cual supone una reducción mayor del espacio de búsqueda, pero, desafortunadamente, su aplicación conlleva la pérdida de la completitud. Por ejemplo, no es posible encontrar una refutación para el problema *cl*6 [Chang y Lee, 1973] aplicando esta poda. El Ejemplo 4.11 muestra un sencillo problema para el cual no se encuentra la refutación debido al uso de esta poda.

c) Cuando el literal seleccionado en la cláusula en curso en una derivación sea igual al complementario de un ancestro del conjunto de ancestro, entonces se aplica únicamente un paso de resolución de ancestro, o sea se impide la aplicación de otras reglas de inferencia. La razón de esta poda es que el paso de resolución de ancestro obtiene la solución más general para el literal seleccionado, siendo cualquiera otra solución que se encuentre aplicando otra regla de inferencia un caso particular de la anterior. Esta poda es enunciada por Stickel en [Stickel, 1988], y fue estudiada en el capítulo 3 de la presente tesis.

d) Cuando el literal seleccionado en la cláusula en curso en una derivación sea subsumido por el literal complementario de una cláusula unitaria de la teoría, entonces se aplica únicamente una paso de resolución de entrada, esto es se impide la aplicación de otras reglas de inferencia. Esta poda también es enunciada por Stickel en [Stickel, 1988].

Las podas c) y d) se probaron de forma conjunta, dando lugar a reducción de los tiempos de demostración de problemas profundos, pero incrementando los tiempos de los problemas sencillos. Esto es debido a que la sobrecarga que implica su aplicación no se ve recompensada por reducciones sustanciales en los problemas más sencillos.

e) Cuando la respuesta computada de una refutación subsidiaria del literal seleccionado en la cláusula en curso no produce una instanciación de ninguna variable del literal no se ha de seguir recorriendo el árbol subsidiario buscando otras refutaciones, ya que cualquier otra respuesta que se pueda encontrar será igual o menos general que la ya encontrada. Las podas c) y d) son casos particulares de esta poda. Esta poda también es enunciada por Stickel en [Stickel, 1988].

f) Cuando el literal seleccionado en la cláusula en curso sea base (totalmente instanciado) y se encuentra una refutación subsidiaria para él, no hace falta seguir recorriendo el árbol subsidiario para localizar más refutaciones. Esta poda es un caso particular de la poda anterior, pero su aplicación es muy poco costosa, ya que la comprobación del hecho de que un literal es base es muy sencilla. Esta poda es particularmente adecuada si la función de selección elegida es la del literal más instanciado ya que entonces su aplicación es bastante frecuente. Experimentalmente ofrece mejoras considerables en algunos problemas.

g) Podar aquellas derivaciones en las cuales, en un cierto paso, el literal seleccionado en la cláusula en curso es subsumido por una cláusula unitaria, distinta de la cláusula raíz. Si una cláusula unitaria subsume a un literal de un cláusula, también subsume a la cláusula completa. Esta poda corresponde a una aplicación parcial de la regla de eliminación de cláusulas subsumidas definida por Davis y Putman [Chang y Lee, 1973]. Su aplicación en costosa, no ofreciendo buenos resultados experimentales hasta el momento.

Ejemplo 4.19 (Poda por subsumción por cláusulas unitarias)

En el siguiente ejemplo se muestra la aplicación de la poda por subsumción del literal seleccionado en la cláusula en curso por una cláusula unitaria. Sea T las cláusulas 1 a 5 indicadas a continuación, además de otras cláusulas que generan un espacio de búsqueda amplio para el predicado *gran_espacio*. Se supone que $T \cup \{\leftarrow p(x)\}$ es consistente. En la figura siguiente se muestra dos árboles SLT desde $\leftarrow p(x)$ en T vía POS, el de la izquierda corresponde a la aplicación de la poda por subsumción del literal seleccionado, el de la derecha sin esta poda. La función de selección es de izquierda a derecha, primero en el cuerpo.

$p(x) \vee t(x) \leftarrow r(x)$	1	$\leftarrow t(x) \wedge s(x)$	4
$r(a) \leftarrow$	2	$s(x) \leftarrow gran\_espacio(x)$	5
$t(a) \leftarrow$	3		

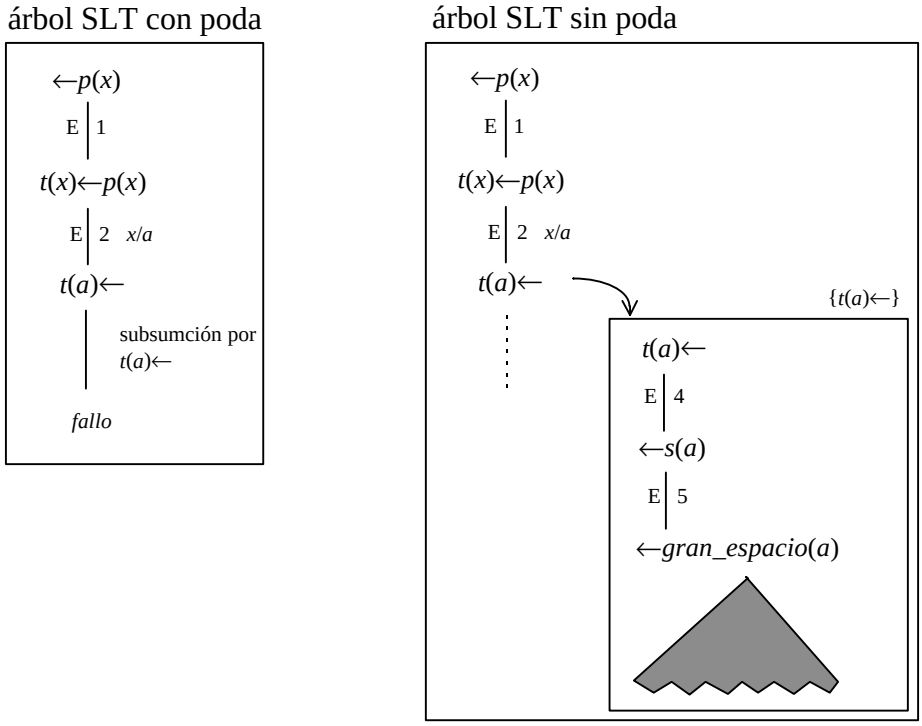


Figura 4.23: Árboles SLT vía POS para el Ejemplo 4.19 con poda por subsumción (izq.) y sin ella (der.)

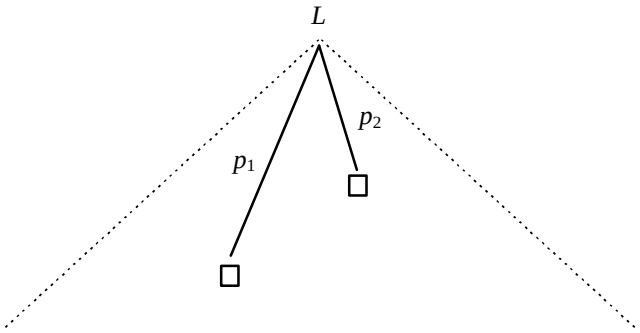
Como se puede observar, el árbol de la izquierda es mucho más pequeño que el árbol de la derecha, que incluso podría ser infinito. Para este problema el demostrador correspondiente a SLT vía POS con la poda que se está tratando respondería rápidamente que el problema es consistente al fallar de forma finita. Sin embargo, el demostrador correspondiente a SLT vía POS sin esta poda tardaría mucha más en responder, o incluso no lo haría si el árbol de la derecha fuera infinito.

Es necesario destacar una problemática asociada a las podas c), d) y a la poda e) que subsume a las dos primeras, cuando se combinan con la búsqueda en profundidad iterada con el criterio de determinación de la profundidad mediante el número de pasos de reglas de inferencia realizados. Esta combinación puede dar lugar a un comportamiento indeseable y que incluso en algunas ocasiones puede llegar a producir la pérdida de la completitud. La detección de este problema vino provocada por la aparición de resultados muy llamativos en la parte experimental del presente trabajo y que se comentan más adelante. Debido a la importancia de este hecho y a que el autor no ha encontrado ninguna mención del mismo se considera necesario explicarlo con detalle.

PROBLEMÁTICA DE LA COMBINACIÓN DE LA PODA POR RESPUESTA MÁS GENERAL Y LA BÚSQUEDA EN PROFUNDIDAD ITERADA CON DETERMINACIÓN DE PROFUNDIDAD POR PASOS DE REGLAS DE INFERENCIA

En principio la poda por respuesta más general, que engloba las podas c), d) y e), puede parecer correcta, entendiendo por *correcta*, que su aplicación no puede provocar bien la eliminación de espacio de búsqueda que contenga demostraciones importantes, bien la pérdida de la completitud del procedimiento. Esta poda consiste, en esencia, en descartar el espacio de búsqueda que falta por recorrer al encontrar una refutación para un cierto literal cuya respuesta computada es la más general para el mismo. Sin embargo esta poda en combinación con la búsqueda en profundidad iterada con determinación de profundidad por pasos de reglas de inferencia puede provocar las circunstancias que se explican a continuación:

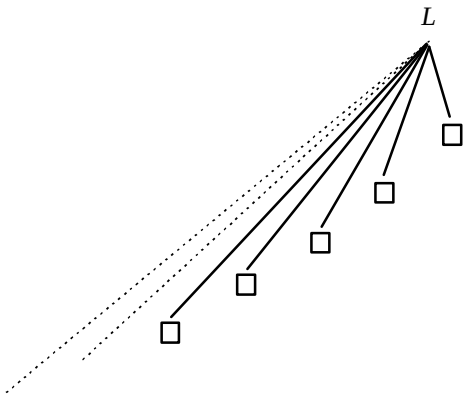
Supóngase que la cláusula en curso *C* contiene dos literales *L* y *L'*, de forma que la función de selección elige primero a *L* y luego a *L'*. Supóngase también que el árbol de estados para el literal *L* esté representado por la siguiente figura.



En la figura se muestran dos refutaciones para *L*, en la de la izquierda se emplean p_1 pasos de reglas de inferencia y en la de la derecha p_2, de forma que p_1 es notablemente mayor que p_2. Además la refutación de la izquierda proporciona una respuesta más general para *L*. Supóngase además que el recorrido del árbol es de izquierda a derechas. Por último, existe una refutación para el literal *L'* que combinada con cualquiera de las dos refutaciones para *L* permite resolver *C*. Además la refutación para *L'* implica el uso de *p'* pasos de reglas de inferencia, de forma que $p_2 + p' > p_1$. Se plantea ahora el siguiente estado de la computación: la cláusula en curso es *C* y la profundidad máxima es ese momento es *p*, tal que $p_2+p'>p$. La manera de proceder de la computación consiste en, una vez seleccionado *L*, demostrarlo en profundidad p_2 y, posteriormente intentar demostrar *L'*. Claramente *L'* no puede ser demostrado con la profundidad límite restante, $p-p_2$, por lo que fallará. La profundidad se incrementa y se vuelve a intentar la demostración. El problema reside en que antes de alcanzar la profundidad máxima necesaria para demostrar *L* y *L'* usando la refutación de profundidad p_2 se llega a la profundidad máxima necesaria para demostrar *L* mediante la refutación de la izquierda, en la cual se usan más pasos de reglas de inferencia, en concreto p_1, y se produce una poda por

respuesta más general para la computación en curso. Esta poda provoca que el procedimiento no considere todo el espacio a la derecha de esta refutación y por consiguiente se alcanza la refutación de la derecha. Por tanto, como la profundidad de esta refutación p_1 es mayor que p_2, es necesario volver a incrementar la profundidad máxima para poder alcanzar la refutación para *L'*. Así el espacio de búsqueda se incrementa al aplicar esta poda cuando lo que se pretendía era todo lo contrario. De esta forma la refutación que localizaría el procedimiento no sería la menor, con lo que claramente se obtendría un resultado peor.

En el caso más extremo el espacio de búsqueda para *L* puede contener infinitas refutaciones (en presencia de recursividad) escalonadas a izquierda con profundidad mayor cuanto más a la izquierda, como muestra la figura inferior. Si se da la coincidencia de que función de selección elija los literales de forma que la profundidad máxima para encontrar el problema se acopla con la profundidad usada en una serie de refutaciones, es posible que el procedimiento no sea capaz de encontrar la refutación existente. La siguiente figura ilustra este hecho.



Claramente esta combinación de circunstancias es muy infrecuente, pero posible. En el conjunto de los problemas abordados se localizaron tres en los que se produjeron estas circunstancias y el procedimiento no fue capaz de demostrarlos. Hay que añadir que la poda por igualdad con ancestro puede evitar que se puede caer en este problema, en la mayoría de las ocasiones, por lo que la elección de ancestros ALL es la que consigue que se dé este problema en un número menor de ocasiones. La solución que se plantea a esta problemática, que no se ha probado experimentalmente, es la siguiente: la construcción de árboles que tengan las refutaciones más cortas más a la izquierda (si el recorrido va a ser de izquierda a derecha) o al menos la construcción de árboles para teorías con recursividad que tengan infinitas refutaciones de forma que se localicen las más largas más a la derecha. Tal vez para concluir es interesante realizar dos indicaciones, la esencia por la cual esta poda presenta estos inconvenientes se puede asociar al hecho de que combinada con la búsqueda en profundidad iterada pierda el sustento en el cual se fundamenta ya que en este tipo de búsqueda el encontrar una refutación para un cierto literal proporciona, aparte de una respuesta al mismo, la profundidad en la cual se demuestra.

Así, aunque la respuesta para un literal sea la más general, la profundidad es una dato que no puede desdeñarse a la hora de eliminar otras posibles refutaciones. Para terminar este apartado avanzar, a falta de una comprobación teórica y experimental, que posiblemente este problema sea general para la búsqueda en profundidad iterada sea cual sea el criterio de determinación de la profundidad.

Para terminar con la presente sección sólo añadir que la búsqueda, estudio y aplicación de diferentes podas que reduzcan el tamaño del espacio de búsqueda de una forma considerable es extraordinariamente importante en el campo del razonamiento automático. En estas podas se podrían encontrar similitudes con ciertos aspectos del razonamiento humano, que es capaz de centrar su poder deductivo en ciertos aspectos de los problemas desechando otros que no conducen a la solución buscada. Evidentemente, se puede demostrar que todas las podas que se han aplicado son correctas, en el sentido de preservar la corrección y completitud del demostrador que las aplica, a diferencia de algunas de las que se aplican en el razonamiento humano, basadas en cierto conocimiento heurístico del problema que se está tratando. La aplicación de podas reduce el tamaño del espacio de búsqueda, provocando en algunos casos una reducción tal que, a nivel práctico, un problema pasa de no ser resoluble a serlo. Sin embargo, el propio cálculo y aplicación de una cierta poda puede tener un coste elevado, provocando, por tanto, una contraprestación debido a la sobrecarga. La relación entre la reducción del tamaño del espacio de búsqueda y la sobrecarga de la aplicación de la poda establece la conveniencia práctica de su aplicación, que siempre debe ser evaluada experimentalmente.

4.5. RESULTADOS EXPERIMENTALES

En esta sección se van describir los procedimientos que se han utilizado finalmente y sus características, así como los resultados obtenidos para un conjunto de problemas y las conclusiones más destacables de los mismos. El procedimiento que se ha utilizado en estas pruebas ha sido únicamente SLT. El motivo de ello es que el conjunto de problemas utilizados han sido extraídos de libros y artículos del campo de la demostración automática de teoremas. En particular se han utilizado los problemas de la librería de problemas de demostración de teoremas TPTP, *Thousand Problems on Theorem Proving* [Suttner y Sutcliffe, 1996]. También se han incluido algunos problemas que son propuestos habitualmente en la literatura, [Chang y Lee, 1973] [Stickel, 1988] [Stickel, 1992] [Manthey y Bry, 1988]. Para algunos de estos últimos ha sido posible comparar con otros ya que se conocían los resultados experimentales de otros autores. El objetivo principal de la parte experimental de la presente tesis es comprobar la eficiencia de resolución SL* (en particular del procedimiento SLT) y ver como varía su comportamiento según cual sea la elección de ancestros utilizada. En concreto se pretende

corroborar que la elección de ancestros subóptima es al menos tan adecuada como la elección ALL y que mejora el comportamiento del procedimiento en algunos casos.

ENTORNO DE TRABAJO

La implementación y todas las pruebas se realizaron en un SUN SPARC 5 utilizando el sistema SICSTUS Prolog. Es interesante destacar que este sistema proporciona dos tipos de unificaciones, una que no incluye el test de ocurrencia y otra correcta que si lo incluye. Este hecho, que no es muy común en otros sistemas, facilitó la implementación considerablemente al no ser necesario el desarrollo de un algoritmo de unificación correcto. Todos los tiempos mostrados en los resultados experimentales, que se muestran en milisegundos, corresponden a tiempo de cómputo de CPU. Hay que indicar que en la obtención de los resultados el tiempo de cómputo dedicado para un problema se limitó a 1000 segundos. Este tiempo de cómputo se midió utilizando las primitivas correspondientes del SICSTUS Prolog. El código de la implementación utilizada tanto para el preprocesador como para el demostrador se puede encontrar en el Apéndice B.

PROBLEMAS RESUELTOS

El conjunto de problemas resueltos se han extraído de los libros y artículos que se incluyen a continuación:

- Del libro [Chang y Lee, 1973]: los problemas numerados como *cl*1 a *cl*9. Son problemas simples como se puede observar en esta pequeña explicación: los seis primeros (*cl*1 a *cl*6) son problemas de sistemas asociativos y de teoría de grupos que siempre incluyen en su definición los axiomas de la asociatividad de la operación producto y los tres últimos (*cl*7 a *cl*9) corresponden a algunos teoremas bien conocidos de la teoría de números. Todos ellos son resolubles utilizando resolución de entrada o unitaria. Stickel en [Stickel, 1988] [Stickel, 1992] da resultados experimentales para los mismos utilizando su demostrador PTTP.

- Del artículo [Wos y Overbeek, 1993]: el problema denominado *wos*1. Este problema es el asociado a la afirmación de que todo grupo en el cual el producto de todo elemento consigo es el elemento neutro es un grupo abeliano. El enunciado dado por los autores incluye todos los axiomas necesarios para definir completamente la teoría de grupos, incluyendo los axiomas de la igualdad.

- Del artículo [Stickel, 1986]: el problema denominado *Steamroller*. Este es un problema muy conocido en el campo del razonamiento automático, siendo enunciado por primera vez por Schubert. Una solución del este problema se menciona en [Walther, 1985] y

Stickel en [Stickel, 1986] define una axiomatización del mismo, junto con algunos resultados experimentales obtenidos por diferentes demostradores. El problema, que modela la existencia de un mundo de animales que se alimentan unos de otros, se corresponde con la pregunta de si existe un tipo particular de animal que cumple determinadas condiciones en su alimentación. Un buen resultado experimental para este problema fue obtenido por el demostrador SATCHMO y puede encontrarse en [Manthey y Bry, 1988].

- De la librería *TPTP Library: Thousand Problems on Theorem Proving* [Suttner y Sutcliffe, 1996], el conjunto completo de problemas.

Las características fundamentales de los problemas de la librería TPTP pueden encontrarse en el Apéndice C. En el Apéndice B se puede encontrar un problema de esta librería en el formato original y en el formato usado en la implementación del procedimiento SLT. También se incluye en este apéndice el código de la implementación del este último.

ELECCIONES DE ANCESTROS

Las elecciones de ancestros que se han utilizado son las siguientes: POS, ALL, EMPTY y la elección subóptima para el problema correspondiente. Los resultados experimentales muestran la adecuación de cada una de ellas para los diferentes problemas tratados. Generalmente, la elección de ancestros ALL es la que da lugar al mejor resultado experimental (este hecho no ha de extrañar, ya que el conjunto de problemas utilizados proviene esencialmente del campo de la demostración automática de teoremas). Sin embargo, para algunos problemas esto no se cumple y las razones se expondrán más adelante en el apartado en el que se presentan las tablas de resultados.

ESTRATEGIA DE RECORRIDO DEL ESPACIO DE BÚSQUEDA

Como ya se comentó en apartados anteriores, la estrategia de recorrido del espacio de búsqueda ha sido siempre la denominada búsqueda en profundidad iterada. La manera de recorrer el espacio de búsqueda es la siguiente: comenzando por una profundidad inicial, intentar encontrar una refutación en esa profundidad; de no encontrarse incrementar la profundidad en una cierta cantidad e intentarlo de nuevo; y así sucesivamente hasta que se localice la refutación. Stickel en sus trabajos utiliza como profundidad inicial cero e incrementa la misma en uno en cada pasada. Se ha tomado el mismo criterio para poder comparar los resultados obtenidos con los suyos, aunque se considera que sería más idóneo utilizar una profundidad inicial mayor que cero y tomar incrementos superiores a uno. El criterio de determinación de la profundidad utilizado fue siempre tomando el número de pasos de

inferencia aplicados de resolución de entrada y de ancestros. El motivo para usar este criterio de determinación de la profundidad es debido a que proporciona un crecimiento más suave del espacio de búsqueda, por lo que era de esperar que permitiera demostrar un mayor número de problemas. Además este criterio permitía la utilización de ciertas podas antes comentadas que no eran aplicables en los otros. Por otra parte, ya que el principal interés se centraba en el estudio del comportamiento de los procedimientos atendiendo a las diferentes elecciones de ancestros utilizadas, no se ha considerado interesante el utilizar otros criterios de determinación de la profundidad de la demostración.

FUNCIÓN DE SELECCIÓN

La función de selección que se ha utilizado es una aproximación de aquella que selecciona el literal más instanciado de la cláusula en curso. Esta aproximación consiste en elegir aquel literal que contiene menos variables que son argumentos de dicho literal. Esta función de selección, aproximación de la que selecciona el literal más instanciado, es la que dio mejores resultados en las prueba preliminares. Es interesante destacar que esta función de selección es muy adecuada atendiendo a las podas que se han aplicado en los procedimientos y que se describirán a continuación.

PODAS

Las podas que se han aplicado en los procedimientos utilizados en la obtención de los resultados experimentales son las siguientes:

- Poda por igualdad con ancestro: si el literal seleccionado en la cláusula en curso en una derivación es idéntico a uno de los ancestros de la derivación, entonces dicha derivación falla. Esta poda ya ha sido comentada numerosas veces y es una de las formas de detección de tautologías en las demostraciones. El coste de aplicación de dicha poda depende del tamaño del conjunto de ancestros y por tanto dependerá de la elección de ancestros que se esté utilizando. Los resultados experimentales demuestran que su uso es imprescindible en numerosos problemas, ya que reduce considerablemente el tamaño del espacio de búsqueda asociado.

- Poda por respuesta más general: si un literal que es raíz inicial de una computación subsidiaria es demostrado sin instanciar ninguna de sus variables, entonces no es necesario aplicar vuelta atrás para obtener más refutaciones asociadas a dicho literal. Esta poda corresponde a la reducción de la aplicación de la vuelta atrás para obtener más respuestas para una cierta meta. Claramente, si una de las respuestas obtenidas es la más general, no tiene ninguna utilidad seguir la búsqueda de más respuestas. La

aplicación de esta poda tiene un coste considerable ya que la comprobación de la no instanciación de variables en ejecución de una cierta meta en los sistemas Prolog es compleja. Existe un caso particular de esta poda que presenta ciertas características interesantes. Este caso es el siguiente: si el literal que es raíz de una computación subsidiaria es base (totalmente instanciado) y se encuentra una refutación para el mismo, no es necesario aplicar vuelta atrás para obtener más refutaciones. Obviamente, cualquier refutación de una meta base proporciona la respuesta más general de la misma. La ventaja de esta poda para el caso base es que la comprobación de que un término es base es muy económico en los sistemas Prolog, por lo que se consigue un menor coste. Por otro lado, la contrapartida está en que la poda para el caso base consigue una reducción menor del espacio de búsqueda ya que se aplica menos veces que la poda general. Esto da como resultado que en numerosos problemas los resultados temporales de los procedimientos con la poda general son peores que los de los procedimientos con la poda para el caso base. Sin embargo, existen algunos problemas que los procedimientos con la versión base de la poda no consiguen demostrar, mientras que los otros sí. Por este motivo se consideró utilizar la poda general en los procedimientos utilizados para la obtención de los resultados experimentales.

ANÁLISIS DE LOS RESULTADOS

La información sobre los problemas abordados y sus características junto con los resultados temporales y espaciales de los procedimientos instancias de resolución SL* utilizados puede encontrase en el Apéndice C. En este apartado se expondrán las conclusiones del análisis de esta información. La tabla siguiente muestra el resumen esencial del trabajo experimental.

Dominio	*Prob.*	*Ins.*	*Cláusulas*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pred*	*Fun*	*PMT*	*Res*	*% Res.*
ALG	4	2	102,0	5,0	22,5	208,5	90,5	4,5	6,5	28,0	4,0	0	0,0
ANA	19	2	12,5	4,0	1,0	28,0	0,0	3,0	1,0	13,0	6,0	0	0,0
BOO	52	41	26,4	0,0	13,1	51,7	26,6	3,7	1,7	6,9	2,7	10	24,8
CAT	59	46	29,1	0,6	7,6	63,4	28,8	3,7	3,2	5,7	2,8	27	58,7
CID	4	2	50,0	0,0	14,5	104,5	60,5	3,0	3,0	15,0	8,5	0	0,0
CIV	4	1	61,0	0,0	48,0	77,0	71,0	3,0	2,0	24,0	4,0	0	0,0
COL	141	82	8,9	0,0	4,3	14,4	14,3	3,0	1,0	5,0	5,7	53	64,6
COM	6	5	23,4	0,6	9,4	45,4	5,6	3,8	6,6	15,0	2,8	5	100,0
GEO	165	79	74,4	8,3	14,8	181,5	82,2	7,9	3,2	11,5	2,3	48	60,8
GRA	1	1	12,0	7,0	0,0	32,0	0,0	3,0	5,0	0,0	0,0	1	100,0
GRP	313	170	21,2	0,6	11,4	38,9	20,5	4,0	2,4	5,9	2,5	104	61,2
HEN	64	62	20,7	0,0	9,0	40,1	18,7	4,2	2,2	5,9	2,9	26	41,9
LAT	10	4	38,5	0,0	20,5	78,0	11,0	5,0	2,5	13,0	1,5	0	0,0
LCL	278	180	8,0	0,0	5,2	12,0	4,7	3,0	1,3	4,4	4,8	105	58,3
LDA	23	10	21,3	1,4	11,6	34,2	23,8	3,0	1,8	10,1	3,2	1	10,0
MSC	12	11	15,0	2,2	2,6	33,2	1,4	3,8	4,9	5,6	2,4	8	72,7
NUM	309	30	94,0	5,7	17,9	203,6	90,2	3,9	6,8	23,5	4,3	19	63,3
PLA	30	30	29,1	0,1	17,9	50,7	0,0	3,9	2,1	13,9	2,9	11	36,7
PRV	9	8	23,0	1,9	9,9	44,9	16,4	4,1	2,5	7,0	2,6	5	62,5
PUZ	45	35	28,9	4,7	8,5	65,2	3,3	4,8	6,4	6,5	1,7	21	60,0
RNG	100	60	33,4	0,1	16,4	64,5	33,1	4,0	2,0	8,1	3,4	19	31,7
ROB	36	15	13,0	0,0	6,5	20,8	19,4	3,0	1,2	5,4	6,3	4	26,7
SET	695	186	170,5	11,3	32,1	358,7	164,6	5,0	9,8	40,2	5,4	93	50,0
SYN	350	313	240,1	1,6	25,6	689,0	0,7	4,4	33,2	4,0	1,2	267	85,3
TOP	24	5	32,4	5,6	3,2	92,0	0,0	4,6	7,6	12,6	2,4	5	100,0
Promedio	2753	1380	47,4	2,4	14,8	139,1	44,1	3,9	4,8	11,5	3,4	832	60,3

Tabla 1: Características de los problemas abordados y resultados globales[22]

Toda la información referente a las características de los problemas de cada dominio corresponde únicamente a los problemas inconsistentes, que son los que el procedimiento debería resolver. Como conclusión global se puede afirmar que la efectividad de resolución SL* (en particular del procedimiento SLT con la elección de ancestros ALL) es aceptable, teniendo en cuenta cuál ha sido su desarrollo y el objetivo de éste. Llegar a demostrar un 60% de los problemas inconsistentes se puede considerar un buen resultado. Observando los dominios relevantes de la tabla (aquellos que contiene 10 o más problemas) se puede comprobar que la incapacidad del procedimiento es más acentuada en los dominios LDA, ROB, BOO, RNG, PLA y HEN que presentan las siguientes características combinadas: uso amplio de la teoría de la igualdad, número de símbolos de predicado bajo, alta presencia de funciones y una profundidad

[22] Aclaración a nombre de las columnas de la Tabla 1: *Dominio*, el nombre de los dominios de los problemas de la librería *TPTP*; *Prob.*, número de problemas del dominio; *Ins.*, número de problemas inconsistentes del dominio, *Cláusula*, promedio del número de cláusulas de los problemas; *CnHn*, promedio del número de cláusula que no son Horn; *CU*, promedio del número de cláusula unitarias; *Lit.*, promedio del número de literales; *Lit. Ig.*, promedio del número de literales que tienen que ver con la teoría de la igualdad; *TMC*, el promedio del tamaño máximo de cláusulas; *Pred*, el promedio del número de predicados de los problemas; *Fun*, el promedio del número de funciones en los problemas; *PMT*, el promedio de la profundidad máxima de los términos en los problemas; *Res*, número de problemas resueltos por el procedimiento SLT con elección de ancestros ALL; y por último *% Res*, el tanto por ciento de problemas resueltos sobre los inconsistentes.

de término elevada. Las razones de esta incapacidad radican en, primero, la falta de adecuación del procedimiento para el tratamiento de la igualdad como ya se comentó; segundo, el bajo número de símbolos de predicado conduce a que usualmente el factor de ramificación del árbol de estados sea muy alto y por tanto también el espacio de búsqueda asociado; y tercero, la presencia de un alto número de funciones junto con una elevada profundidad máxima de los términos provoca que el coste de la unificación se eleve considerablemente, impidiendo que el procedimiento demuestre el problema dentro del tiempo limitado de que dispone. Por el contrario aquellos dominios que no presentan una o más de estas características (SYN, MSC, COL, NUM, GRP y PRV) son tratados con mejores porcentajes de éxito. Los dominios COL y PRV puede llamar especialmente la atención ya que tienen todas las características para que no sean tratados de forma eficiente por el procedimiento SLT y sin embargo se obtiene un tanto por ciento de éxito del 65,8% y 62,5%, respectivamente. Esto es debido a que el número de cláusulas de los problemas es muy bajo (8,9 y 23 en promedio, respectivamente) y en relación a éste el número de cláusula unitarias es muy alto (4,3 y 9,9 en promedio, respectivamente). Claramente esto permite comprender que los problemas de este dominio no tienen una complejidad muy alta. Por otro lado en el dominio NUM se presenta un caso en el cual se puede observar que a pesar de tener algunas de las características que no lo hacen adecuado sin embargo tiene otras características que compensan estas desventajas, como son el número alto de símbolos de predicados y la considerable presencia de cláusulas unitarias en los problemas.

A continuación se incluyen varias tablas con los principales resultados tanto temporales como espaciales de los procedimientos SLT vía las distintas elecciones de ancestros en la cual se basará el estudio comparativo de éstos. En las tablas siguientes se incluye cada uno de los dominios tratados con el número de problemas resueltos.

Dominio	*ALL*	*POS*	*EMPTY*	*SUBOP*
ALG	0	0	0	0
ANA	0	0	0	0
BOO	10	9	9	10
CAT	27	26	26	27
CID	0	0	0	0
CIV	0	0	0	0
COL	53	49	49	53
COM	5	5	5	5
GEO	48	48	48	48
GRA	1	0	0	1
GRP	104	101	97	104
HEN	26	24	24	26
LAT	0	0	0	0
LCL	105	103	103	105
LDA	1	1	1	1
MSC	8	6	5	8
NUM	19	19	19	19
PLA	11	11	11	11
PRV	5	4	4	5
PUZ	21	19	14	21
RNG	19	17	17	19
ROB	4	4	4	4
SET	93	89	85	93
SYN	267	262	240	267
TOP	5	5	3	5
Totales	832	802	764	832

Tabla 2: Problemas resueltos de cada dominios según la elección de ancestros

Como se puede observar en la tabla, los dominios en los que el número de problemas resueltos es similar al utilizar los diferentes procedimientos vía las distintas elecciones de ancestros son COM, GEO, LDA (aunque no sea significativo), NUM, PLA, ROB y TOP. También se puede ver que los dominios donde se dan las mayores diferencias entre la elección de ancestros ALL y la elección POS son GRA (aunque no sea significativo), MSC, PRV, PUZ y RNG. Por otro lado si se comparan los resultados de los procedimientos vía POS y EMPTY se puede comprobar que los dominios en los que el procedimiento vía la elección de ancestros EMPTY resuelve comparativamente menos que vía POS son PUZ, TOP y MSC. Las tablas que se incluyen a continuación permitirán entender los motivos de los resultados de los procedimientos vía las diferentes elecciones de ancestros.

Dominio	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
BOO	215265,0	0,00	11,7	0,0	1989943,5	4,5	24,8	80389,1
CAT	48802,2	0,00	7,8	718,4	777255,4	1024,6	98,5	19779,3
COL	89677,4	0,00	6,9	0,0	557913,1	1,5	2,6	32342,3
COM	132,0	0,20	10,2	9,0	1226,0	9,8	2,0	74,6
GEO	10123,3	0,00	5,2	7,3	298132,5	6,0	2,6	2572,8
GRA	1450,0	12,00	17,0	4647,0	21032,0	7849,0	0,0	5405,0
GRP	56590,5	0,71	11,8	43461,4	398583,4	44984,4	13,4	17130,6
HEN	37652,7	0,00	8,9	0,0	426526,9	1895,1	2685,4	26678,8
LCL	56607,0	0,00	12,9	21,2	234256,5	7,6	39,3	8066,3
LDA	81270,0	0,00	13,0	0,0	791773,0	25,0	0,0	27630,0
MSC	111446,3	1,38	14,9	292517,8	754286,9	13211,8	34119,0	250344,5
NUM	1463,2	0,63	6,7	138,9	28817,4	59,4	14,6	893,5
PLA	30586,4	0,00	14,4	1,1	163413,4	7924,0	3,3	747,6
PRV	179596,0	0,20	7,8	24548,8	3173339,2	13374,0	140,4	151120,0
PUZ	61892,4	1,67	14,6	61635,3	767297,5	19605,2	28911,2	13733,8
RNG	94946,3	0,00	8,5	834,6	1855938,5	64,7	270,7	83764,6
ROB	19177,5	0,00	11,0	0,0	132090,5	3,8	4,0	9881,3
SET	25908,6	0,42	5,4	1223,6	1678623,5	97,1	188,8	1017,5
SYN	4196,0	0,66	7,9	827,9	43562,6	2109,7	339,7	1510,5
TOP	114392,0	2,20	6,6	85915,4	927835,4	11842,8	17756,2	10329,2
Promedios	37303,9	0,4	8,9	10921,4	494298,3	7295,4	1396,4	14163,2

Tabla 3: Resultados temporales y espaciales por dominios para SLT vía ALL[23]

[23] Aclaración al nombre de las columnas de la Tabla 3: *Dominio*, los dominios de los problemas de la librería *TPTP*; *Tiempo*, el tiempo usado en la resolución del problema; *RA ref*, los pasos de resolución de ancestro dados en la refutación encontrada; *RE ref*, los pasos de resolución de entrada dados en la refutación encontrada; *RA árbol*, los pasos totales de resolución de ancestro realizados en el árbol recorrido; *RE árbol*, los pasos de resolución de entrada efectuados en el árbol recorrido; *Poda base*, la cantidad de veces que se ha podido aplicar la poda asociada a la resolución de un literal base; *Poda RG*, el número de aplicaciones de la poda correspondiente al hallazgo de un refutación con respuesta más general para un literal; y por último, *Poda IA*, el número de veces que se podido utilizar la poda por igualdad con ancestro. Todos los datos corresponden a promedios del conjunto de problemas de cada dominio.

Dominio	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
BOO	472798,9	0,0	11,6	0,0	2208616,7	4,6	1,8	61271,7
CAT	46154,6	0,0	7,7	644,4	872149,8	4464,5	317,5	1592,3
COL	89180,6	0,0	6,3	0,0	571935,4	1,4	3,4	0,0
COM	238,0	0,2	10,2	9,0	2218,6	9,8	2,0	12,8
GEO	12023,1	0,0	5,2	3,0	333690,5	5,6	2,9	22,9
GRP	76705,7	0,5	11,4	38465,9	498258,4	41295,7	19,9	9257,2
HEN	61227,9	0,0	8,5	0,0	901757,7	1805,0	1654,5	0,0
LCL	96569,3	0,0	12,8	22,3	454192,1	17,3	239,0	64,3
LDA	210100,0	0,0	13,0	0,0	2107632,0	25,0	0,0	0,0
MSC	111651,7	0,3	15,0	8358,3	930529,7	1013,7	7966,3	216,3
NUM	3781,6	0,1	7,7	212,4	65385,6	296,2	16,0	378,2
PLA	27323,6	0,0	14,4	0,0	163485,9	7924,0	1,1	741,9
PRV	220,0	0,0	7,3	51,3	2678,8	63,5	1,0	9,8
PUZ	6488,1	1,4	15,4	1204,1	37820,4	8260,0	67,3	789,5
RNG	20484,1	0,0	8,1	1143,5	451626,5	8,4	36,8	2567,8
ROB	87440,0	0,0	11,0	0,0	719075,0	3,8	4,0	0,0
SET	16315,1	0,2	5,4	113,4	1086717,0	26,3	8,3	86,9
SYN	5723,0	0,6	8,1	2395,7	46047,8	4322,0	252,0	750,3
TOP	142900,0	1,8	7,0	75925,2	1197396,6	10879,6	11370,2	5491,4
Promedios	44016,9	0,3	8,9	6254,4	427550,1	7198,9	309,6	2296,7

Tabla 4: Resultados temporales y espaciales por dominios para SLT vía POS

Dominio	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
BOO	401754,4	0,0	11,6	0,0	5508844,8	4,6	1,8	0,0
CAT	40225,0	0,0	7,7	0,0	917839,8	4441,4	292,5	0,0
COL	80486,1	0,0	6,3	0,0	571935,4	1,4	3,4	0,0
COM	246,0	0,0	10,6	0,0	3316,8	8,2	2,0	0,0
GEO	10855,0	0,0	5,2	0,0	334883,3	5,6	2,9	0,0
GRP	40562,4	0,0	8,6	0,0	481973,8	9708,7	10,1	0,0
HEN	50911,3	0,0	8,5	0,0	901757,7	1805,0	1654,5	0,0
LCL	85294,3	0,0	12,8	0,0	458688,3	17,3	239,0	0,0
LDA	182580,0	0,0	13,0	0,0	2107632,0	25,0	0,0	0,0
MSC	43740,0	0,0	12,0	0,0	802036,0	522,8	104,2	0,0
NUM	34372,6	0,0	8,2	0,0	665621,1	2759,1	208,7	0,0
PLA	22749,1	0,0	14,4	0,0	170011,8	8033,9	1,1	0,0
PRV	150,0	0,0	7,3	0,0	2585,0	55,5	0,0	0,0
PUZ	12586,4	0,0	14,9	0,0	109727,5	18998,4	2,5	0,0
RNG	18305,3	0,0	8,1	0,0	512932,1	12,9	37,2	0,0
ROB	76707,5	0,0	11,0	0,0	719075,0	3,8	4,0	0,0
SET	19326,8	0,0	5,6	0,0	1406544,8	34,5	1,6	0,0
SYN	6181,8	0,0	8,8	0,0	71156,6	23956,2	213,9	0,0
TOP	0,0	0,0	2,3	0,0	25,0	1,7	0,0	0,0
Promedios	37035,7	0,0	8,8	0,0	528284,2	9546,8	170,9	0,0

Tabla 5: Resultados temporales y espaciales por dominios para SLT vía EMPTY

Como se puede observar en las tablas anteriores los datos correspondientes a la aplicación de la resolución de ancestro y a la poda por igualdad con ancestro marcan las diferencias entre el uso de las elecciones ALL, POS y EMPTY. Los dominios donde se daba la mayor diferencia entre la elección de ancestro ALL y las elecciones POS y EMPTY corresponde a aquellos donde se ha utilizada con mayor frecuencia la resolución con ancestros y/o la poda por igualdad con ancestro. Por ejemplo en el dominio MSC (en el que se dan la las mayores diferencias con

un número de problemas resueltos significativo) se puede comprobar que, en promedio, se realizan 1,38 pasos de resolución de ancestro en la refutación encontrada, 292517,8 pasos de resolución de ancestro en el árbol y 250344,5 aplicaciones de la poda por igualdad con ancestro. Claramente, el procedimiento SLT vía POS tiene más dificultades al tratar con los problemas de este dominio al aumentar el espacio de búsqueda ya que, por una parte, ha de suplir algunos pasos de resolución de ancestro mediante varios pasos de resolución de entrada y de ancestro (con lo cual refutación encontrada es mayor profundidad) y, por otra parte, no puede aplicar de la poda por igualdad con ancestro en algunas ocasiones al considerar como ancestros únicamente los literales positivos. Esto mismo, pero de forma más acusada, se puede argumentar para el procedimiento SLT vía EMPTY. La comparación de los procedimientos vía las diferentes elecciones de ancestros usando estos datos no resulta fácil ya que en los dominios donde se ve más claramente las diferencias entre el procedimiento SLT vía ALL, vía POS y, sobre todo, vía EMPTY los resultados se ven afectados por la ausencia de datos acerca de los problemas que no pudieron resolverse vía POS y EMPTY. La mejor forma de ver estas diferencias es analizar los problemas resueltos por SLT vía ALL que no resuelve SLT vía POS y, de forma análoga, los problemas que resuelve SLT vía POS que no resuelve SLT vía EMPTY. Las siguientes tablas muestran los resultados de estos problemas.

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
BOO006-2	952560	0	13	0	6617386	4	232	170392
CAT006-4	902460	0	14	0	11619263	6633	0	358659
COL057-1	1039450	0	12	0	5330463	3	16	181260
COL060-2	1015170	0	15	0	7248461	3	0	512316
COL060-3	244650	0	14	0	1691524	3	0	131984
COL061-3	821460	0	15	0	5560593	4	0	408920
GRA001-1	1450	12	17	4647	21032	7849	0	5405
GRP012-3	270750	0	13	0	3537618	4	170	247797
GRP135-1.002	186870	8	36	256055	1103125	220875	0	93358
GRP135-2.002	190050	8	36	256055	1125020	220875	0	93358
HEN004-4	619240	0	15	0	6241874	17135	49733	268739
HEN005-2	152190	0	14	0	2058757	23894	9986	213958
LCL040-1	216750	0	20	0	105042	80	86	15840
LCL112-1	1569860	0	20	0	6326814	141	847	281239
MSC006-1	704750	7	19	2326452	3771696	80884	271403	1822930
MSC007-2.002	73380	0	19	11826	846865	22963	0	130606
PRV007-1	897250	1	10	122511	15857206	66616	698	755401
PUZ023-1	118350	2	18	99928	1327022	20443	19735	18644
PUZ025-1	1079090	4	15	1171477	14175383	266956	585603	251294
RNG001-5	948790	0	13	0	17609610	124	4612	719851
RNG038-1	662000	0	12	0	12728817	833	289	768677
SET012-1	174450	2	12	89713	3368499	3916	13004	72378
SET055-6	3020	2	6	9	199852	14	4	25
SET232-6	415990	1	7	597	28060815	46	44	2764
SET233-6	419390	1	7	597	28060825	46	44	2764
SYN082-1	13200	8	10	23186	162225	21653	0	18095
SYN098-1.002	190	10	22	448	1821	990	0	48
SYN345-1	1820	17	15	3642	12178	4620	20	3237
SYN349-1	18740	14	9	26104	71744	1	26643	26884
SYN350-1	870	6	5	893	9048	346	351	856

Tabla 6: Problemas resueltos por SLT vía ALL que no resuelve SLT vía POS (30 problemas)

Como se puede observar en la tabla anterior las causas que provocan que SLT vía POS no sea capaz de resolver estos problemas recaen en dos motivos: primero, que en la refutación encontrada por SLT vía ALL se hayan realizado pasos de resolución de ancestro que el SLT vía POS no puede aplicar (un ejemplo muy llamativo es GRA001-1); y segundo, que el espacio de búsqueda generado por el SLT vía POS sea mayor que el generado por el SLT vía ALL porque se aplica menos veces la poda por igualdad con ancestro al no considerar ancestros los literales negativos. Dentro de los problemas del segundo caso se puede distinguir aquellos en los que el SLT vía ALL obtuvo una refutación sin pasos de resolución de ancestro, los cuales cabría pensar que el SLT vía POS los debía demostrar (por ejemplo, HEN004-4, HEN005-2, etc.). Sin embargo, si se observa la columna de derecha, *Poda IA*, se puede ver que para estos problemas contiene valores elevados (generalmente superiores a 100.000). Al realizar menos podas por igualdad con ancestro el espacio de búsqueda aumenta considerablemente impidiendo que el procedimiento SLT vía POS encuentre la refutación dentro del tiempo límite de computación. Un estudio análogo pero entre los procedimientos vía POS y EMPTY se puede realizar utilizando los datos de la tabla que se incluye a continuación.

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
GRP131-1.002	501130	12	77	458571	3287252	544602	0	104208
GRP131-2.002	528130	12	77	458571	3432626	544602	0	104208
GRP132-1.002	469340	12	77	425757	3109651	532348	0	95888
GRP132-2.002	487810	12	77	425757	3250250	532348	0	95888
MSC004-1	410660	2	30	49984	1605535	3501	47249	0
PUZ009-1	270	3	8	219	5234	317	0	625
PUZ014-1	2640	10	19	3811	20132	8188	0	2027
PUZ015-2.003	4360	2	19	1950	48020	9719	0	7734
PUZ029-1	360	5	27	256	1944	1443	0	88
PUZ032-1	6770	4	8	2504	70185	548	781	1113
SET043-5	0	2	3	7	43	7	0	0
SET044-5	120	2	5	41	1298	23	0	81
SET045-5	10	2	4	5	63	3	0	8
SET046-5	200	5	5	255	1656	146	26	114
SYN007-1.002	20	2	5	14	624	37	0	26
SYN011-1	180	2	15	112	1370	589	0	13
SYN029-1	10	2	8	5	85	28	0	5
SYN030-1	8390	29	49	9630	94933	19096	0	6885
SYN031-1	70	2	6	24	907	25	0	30
SYN032-1	140	10	30	205	818	701	0	0
SYN034-1	170	6	7	163	1118	190	0	53
SYN044-1	50	3	8	29	670	85	0	52
SYN051-1	10	2	5	5	70	13	0	0
SYN052-1	30	2	5	11	379	27	1	8
SYN053-1	100	2	8	32	1181	192	0	4
SYN054-1	30	3	10	22	158	71	0	0
SYN070-1	322860	9	16	543377	1437534	637123	0	92109
SYN081-1	40	2	6	21	596	0	0	21
SYN084-2	15710	17	22	5609	59042	16829	0	1599
SYN315-1	30	2	5	11	231	24	6	8
SYN321-1	50	2	6	17	397	35	0	13
SYN323-1	30	2	5	12	189	28	0	12
SYN327-1	610	3	6	378	8241	448	0	412
SYN343-1	0	2	3	4	33	4	0	0
SYN352-1	46590	8	7	22130	443569	9728	14143	4992
SYN354-1	21870	10	7	7867	262657	5624	459	9014
TOP001-2	57780	3	14	28787	437973	3454	4839	13500
TOP005-2	656720	6	14	350839	5548935	50939	52012	13957

Tabla 7: Problemas resueltos por SLT vía POS que no resuelve SLT vía EMPTY (38 problemas)

Como se puede comprobar en la tabla la razón de la incapacidad del SLT vía EMPTY para resolver estos problemas recae en la incompletitud del mismo en el tratamiento de problemas clausales generales. Como puede verse en todos los problemas ha sido necesario la utilización de dos o más pasos de resolución de ancestro. Existen dos problemas realmente demostrativos de este aspecto, SET043-5 y SYN343-1, que SLT vía ALL y vía POS resuelve con extremada rapidez y sin embargo SLT vía EMPTY es incapaz de demostrarlos. Añadir que los problemas resueltos por el SLT vía ALL y que no puede resolver el SLT vía EMPTY son la unión de los anteriores (los incluidos en la Tabla 6 y en la Tabla 7).

Por último es interesante realizar un análisis comparativo de los resultados temporales a de los distintos procedimientos, de forma que sea posible encontrar las razones que a las mejores o peores resultados de los procedimientos. Las tablas incluidas a continuación muestran de forma promediada por dominios los problemas en los que un demostrador obtiene mejores resultados temporales que otro. Se han considerado únicamente los resultados temporales que se diferencian en más de 10 milisegundos.

			ALL			*EMPTY*		
Dominio	*mejor tiempo %*	*Nº prob*	*RA ref*	*RE ref*	*Poda IA*	*RA ref*	*RE ref*	*Poda IA*
BOO	71,5	9	0,0	11,6	70388,8	0,0	11,6	0,0
CAT	29,1	17	0,0	8,4	10287,7	0,0	8,5	0,0
COL	41,7	27	0,0	8,6	17763,1	0,0	8,6	0,0
COM	64,8	2	0,5	9,0	186,5	0,0	10,0	0,0
GEO	26,0	11	0,0	7,0	10907,4	0,0	7,0	0,0
GRP	40,5	38	0,0	9,7	15337,6	0,0	9,7	0,0
HEN	74,8	16	0,0	11,3	13184,4	0,0	11,3	0,0
LCL	42,4	53	0,0	15,7	10181,7	0,0	15,7	0,0
LDA	55,5	1	0,0	13,0	27630,0	0,0	13,0	0,0
MSC	51,4	1	0,0	15,0	49220,0	0,0	15,0	0,0
NUM	53,1	12	0,9	8,2	1414,2	0,0	10,5	0,0
PLA	29,3	2	0,0	12,0	1382,0	0,0	12,0	0,0
PUZ	73,4	5	0,8	22,0	1735,4	0,0	24,0	0,0
RNG	46,4	10	0,0	9,5	10277,0	0,0	9,5	0,0
ROB	67,4	4	0,0	11,0	9881,3	0,0	11,0	0,0
SET	48,3	6	1,0	5,7	341,3	0,0	7,7	0,0
SYN	48,4	15	1,5	22,3	6771,4	0,0	37,9	0,0
Totales	46,2	229	0,2	11,9	13506,3	0,0	13,1	0,0

Tabla 8: Problemas en los que SLT vía ALL aventaja a SLT vía EMPTY

Se puede observar que las ventajas que obtiene el SLT vía ALL frente al SLT vía EMPTY vienen dadas por, primero, la imposibilidad de este último de utilizar la resolución de ancestro lo que provoca un incremento en la longitud de la demostración y, segundo, la aplicación de la poda por igualdad con ancestro que realiza el SLT vía ALL lo que reduce el espacio de búsqueda que ha de recorrer. Como se puede ver las mejoras en porcentaje son considerables (un 46,2 en el cómputo general), el coste que tiene para el SLT vía EMPTY el suplir las carencias de no aplicar la resolución de ancestro y la poda por igualdad con ancestro es muy elevado. Si a esto se le añade el hecho concerniente al conjunto de problemas que resueltos por el SLT vía ALL no puede demostrar el SLT vía EMPTY (ver la Tabla 6 y la Tabla 7) se puede entender las desventajas de no aplicar estas dos operaciones.

			ALL			*POS*		
Dominio	*mejor tiempo %*	*Nº prob*	*RA ref*	*RE ref*	*Poda IA*	*RA ref*	*RE ref*	*Poda IA*
BOO	75,7	9	0,0	11,6	70388,8	0,0	11,6	61271,7
CAT	38,1	20	0,0	8,2	8760,8	0,0	8,3	2070,0
COL	48,0	28	0,0	8,5	17129,5	0,0	8,5	0,0
COM	35,6	3	0,3	10,3	124,3	0,3	10,3	21,3
GEO	31,1	17	0,0	6,6	7249,5	0,0	6,6	63,6
GRP	38,3	53	1,1	16,6	25411,3	1,0	16,7	17641,1
HEN	79,2	16	0,0	11,3	13184,4	0,0	11,3	0,0
LCL	47,8	56	0,0	15,3	9639,4	0,0	15,3	118,2
LDA	61,3	1	0,0	13,0	27630,0	0,0	13,0	0,0
MSC	55,0	3	1,0	16,3	16406,7	0,7	19,0	432,7
NUM	54,5	12	0,9	8,2	1414,2	0,2	9,8	598,8
PLA	40,8	2	0,0	12,0	1382,0	0,0	12,0	1341,5
PRV	18,5	2	0,0	10,0	89,5	0,0	10,0	19,5
PUZ	52,2	13	2,2	16,8	1419,8	2,0	18,3	1153,7
RNG	41,6	13	0,0	8,8	7920,8	0,0	8,8	3357,9
ROB	72,0	4	0,0	11,0	9881,3	0,0	11,0	0,0
SET	12,2	52	0,4	5,8	244,5	0,3	6,1	125,9
SYN	36,0	50	1,9	16,8	2652,5	2,5	18,6	3915,8
TOP	22,8	2	5,5	13,0	25823,0	4,5	14,0	13728,5
Totales	40,4	356	0,6	12,1	11135	0,7	12,6	5155,2

Tabla 9: Problemas en los que SLT vía ALL aventaja a SLT vía POS

Las razones por las cuales el procedimiento SLT vía ALL obtiene mejores resultados que el SLT vía POS son muy similares a las de la comparación ALL frente a EMPTY. Así la imposibilidad de aplicar la resolución de ancestro sobre los literales positivos del SLT vía POS y el mayor uso de la poda por igualdad con ancestro del SLT vía ALL provoca la obtención de unos mejores resultados. Un dominio a destacar es el SYN, donde el promedio de los pasos de resolución de ancestro de POS supera a las del ALL. Ello es debido a que las refutaciones encontradas por el SLT vía POS al no poder utilizar completamente la resolución de ancestro son de una mayor longitud, como se puede ver, supliendo los pasos de resolución de ancestro por deducciones de cierta longitud. Esto claramente provoca un aumento del espacio de búsqueda y un peor resultado temporal.

			POS			*ALL*		
Dominio	*mejor tiempo %*	*Nº prob*	*RA ref*	*RE ref*	*Poda IA*	*RA ref*	*RE ref*	*Poda IA*
LCL	5,5	9	0,0	22,4	0,0	0,0	22,4	1101,4
PLA	12,6	5	0,0	19,8	1095,6	0,0	19,8	1092,0
SET	51,3	1	1,0	12,0	1081,0	2,0	12,0	3774,0
SYN	67,7	3	0,0	14,7	0,0	0,0	18,0	73798,0
Totales	20,4	18	0,1	19,8	364	0,1	20,4	13363,4

Tabla 10: Problemas en los que SLT vía POS aventaja a SLT vía ALL

Si de estos resultados se excluyeran los datos del problema SET009-1 del dominio SET y de los problemas SYN213-1 y SYN266-1 del dominio SYN en los que se está produciendo un comportamiento muy peculiar del procedimiento SLT vía ALL debido a la poda por respuesta más general y que se comentó anteriormente en la sección 4.4.6, se vería que el SLT vía POS

aventaja al SLT vía ALL por el incremento de eficiencia que supone llevar un conjunto de ancestro más pequeño. Esto se puede ver fácilmente en los resultados de la tabla en los dominios LCL y PLA. Claramente en aquellos problemas donde el SLT vía ALL no saca partido de la aplicación completa de la resolución de ancestro y de la poda por igualdad con ancestro obtiene unos resultados ligeramente peores.

			POS			*EMPTY*		
Dominio	*mejor tiempo %*	*Nº prob*	*RA ref*	*RE ref*	*Poda IA*	*RA ref*	*RE ref*	*Poda IA*
COM	61,5	1	1,0	8,0	64,0	0,0	10,0	0,0
NUM	89,3	3	0,7	12,3	2339,3	0,0	15,3	0,0
PUZ	64,0	2	1,0	40,5	720,5	0,0	43,5	0,0
RNG	71,2	1	0,0	8,0	1489,0	0,0	8,0	0,0
SET	98,0	2	1,0	5,0	10,0	0,0	7,0	0,0
SYN	59,4	4	5,8	19,3	24,8	0,0	77,0	0,0
Totales	74,0	13	2,3	17,0	779,3	0,0	36,4	0,0

Tabla 11: Problemas en los que SLT vía POS aventaja a SLT vía EMPTY

Se puede ver que las importantes ventajas que logra el SLT vía POS frente al SLT vía EMPTY vienen provocadas esencialmente por la imposibilidad del uso de la resolución de ancestro de este último. Así, el SLT vía EMPTY deben suplir los pasos de resolución de ancestro por una deducciones de resolución de entrada de cierta longitud, incrementando el espacio de búsqueda a recorrer. Obviamente el hecho de que un paso de resolución de ancestro se pueda suplir con un conjunto de pasos de resolución de entrada se da únicamente en un conjunto pequeño de problemas, habiendo bastantes más problemas en los que un paso de resolución de ancestro no se puede sustituir por un conjunto de pasos de resolución de entrada provocando la incapacidad del SLT vía EMPTY para resolverlos (ver Tabla 7).

			EMPTY			*ALL*		
Dominio	*mejor tiempo %*	*Nº prob*	*RA ref*	*RE ref*	*Poda IA*	*RA ref*	*RE ref*	*Poda IA*
CAT	20,2	2	0,0	6,0	0,0	0,0	6,0	35,0
COM	19,2	2	0,0	13,0	0,0	0,0	13,0	0,0
GEO	7,4	1	0,0	6,0	0,0	0,0	6,0	743,0
GRP	46,9	8	0,0	22,4	0,0	1,3	21,9	45446,8
LCL	16,0	13	0,0	19,8	0,0	0,0	19,8	766,5
MSC	11,7	2	0,0	12,0	0,0	0,0	12,0	0,0
PLA	26,0	6	0,0	18,7	0,0	0,0	18,7	910,0
PRV	29,5	1	0,0	14,0	0,0	0,0	14,0	9,0
PUZ	19,9	3	0,0	10,0	0,0	0,3	11,3	183,7
RNG	7,9	1	0,0	6,0	0,0	0,0	6,0	131,0
SET	11,3	38	0,0	6,3	0,0	0,2	6,0	329,4
SYN	18,4	25	0,0	17,8	0,0	0,0	17,4	9323,7
Totales	18,0	102	0,0	13,2	0,0	0,2	13,0	6138,4

Tabla 12: Problemas en los que SLT vía EMPTY aventaja a SLT vía ALL

Como se puede observar en la tabla el SLT vía EMPTY aventaja al SLT vía ALL en aquellos problemas en los en este último no puede sacar partido de la posibilidad del uso de la resolución de ancestro y de la poda por igualdad con ancestro, por lo que se ve perjudicado por

la sobrecarga de llevar el conjunto de ancestros. Aun así la mejora obtenida por el SLT vía EMPTY siendo significativa no es demasiado elevada.

			EMPTY			*POS*		
Dominio	*mejor tiempo %*	*Nº prob*	*RA ref*	*RE ref*	*Poda IA*	*RA ref*	*RE ref*	*Poda IA*
BOO	14,5	9	0,0	11,6	0,0	0,0	11,6	61271,7
CAT	16,6	21	0,0	8,1	0,0	0,0	8,1	1971,4
COL	12,8	23	0,0	8,9	0,0	0,0	8,9	0,0
COM	24,8	3	0,0	12,0	0,0	0,0	12,0	0,0
GEO	10,6	11	0,0	7,2	0,0	0,0	7,2	98,3
GRP	22,1	47	0,0	11,9	0,0	0,1	12,0	11378,4
HEN	17,1	16	0,0	11,3	0,0	0,0	11,3	0,0
LCL	13,4	71	0,0	16,0	0,0	0,0	16,0	93,2
LDA	13,1	1	0,0	13,0	0,0	0,0	13,0	0,0
MSC	16,1	3	0,0	13,0	0,0	0,0	13,0	432,7
NUM	16,0	9	0,0	8,9	0,0	0,0	8,9	18,6
PLA	17,1	8	0,0	17,0	0,0	0,0	17,0	1020,1
PRV	31,5	2	0,0	10,0	0,0	0,0	10,0	19,5
PUZ	36,0	5	0,0	11,0	0,0	0,0	13,6	391,0
RNG	16,6	8	0,0	9,9	0,0	0,0	9,9	5270,5
ROB	13,6	4	0,0	11,0	0,0	0,0	11,0	0,0
SET	15,2	54	0,0	6,3	0,0	0,1	6,1	138,5
SYN	16,5	42	0,0	18,6	0,0	0,0	18,6	1915,8
Totales	16,4	337	0,0	12,0	0,0	0,0	12,1	3789,5

Tabla 13: Problemas en los que SLT vía EMPTY aventaja a SLT vía POS

Los dominios donde el promedio de aplicación de la resolución de entrada en el SLT vía POS supera el del SLT vía EMPTY (GRP y PUZ) contienen uno o varios problemas donde se está produciendo el ya mencionado fenómeno peculiar provocado por la poda por respuesta más general que se comento anteriormente en la sección 4.4.6 y que da lugar a un aumento innecesario de la longitud de la refutación encontrada y por tanto un incremento del espacio de búsqueda. Si no se consideran dichos problemas, las razones por las cuales el SLT vía EMPTY obtiene mejores resultados que el SLT vía POS son idénticas a las que se dieron para la comparación SLT vía EMPTY y vía ALL, sólo que acentuadas por las características del SLT vía POS, ya que éste hace un uso limitado de la resolución de ancestro y de la poda por igualdad con ancestro, consiguiendo por tanto menos ventajas.

Por último se incluyen los datos referentes al procedimiento SLT vía la elección de ancestros subóptima (SUBOP). La primera tabla muestra por dominios el número de problemas cuya elección de ancestros subóptima difiere de la elección ALL sobre el total de problemas.

Dominio	*Nº prob. SUBOP=ALL*	*Nº prob. SUBOP<>ALL*
ALG	4	0
ANA	19	0
BOO	52	0
CAT	59	0
CID	4	0
CIV	4	0
COL	139	2
COM	2	4
GEO	165	0
GRA	0	1
GRP	286	27
HEN	64	0
LAT	10	0
LCL	230	48
LDA	23	0
MSC	9	3
NUM	305	4
PLA	28	2
PRV	8	1
PUZ	28	17
RNG	100	0
ROB	36	0
SET	689	6
SYN	105	245
TOP	19	5
Totales	2388	365

Tabla 14: Problemas en los que SUBOP difiere de la ALL

Como se puede observar el número de problemas cuya elección de ancestros subóptima difiere de la ALL es realmente bajo. La razón recae en el tipo de problemas que incluye la librería TPTP, centrada en el campo de la demostración automática. Como se puede ver que los dominios SYN y PUZ son los que, proporcionalmente, contienen un número de problemas cuya elección de ancestros subóptima es diferente de la ALL. Claramente estos dominios son los más lejanos a los campos de las matemáticas y la lógica.

Para terminar se incluye una tabla que muestra los resultados temporales, promediados por dominios, del procedimiento SLT vía SUBOP para aquellos problemas cuya elección de ancestros subóptima difiere de la ALL comparándolos con los resultados obtenidos por el SLT vía ALL. Sólo se han considerados aquellas diferencias superiores a 10 milisegundos.

Dominio	*mejor tiempo %*	*Tamaño SUBOP*	*Tamaño ALL*	*Nº prob.*
GRP	3,9	4,6	9,2	19
LCL	-2,8	3,0	4,0	22
MSC	-0,1	7,7	11,3	3
PUZ	3,7	11,5	14,8	8
SYN	3,2	55,3	82,0	28
TOP	-0,1	7,5	9,0	2
Totales	1,6	22,3	33,3	82

Tabla 15: Resultados temporales de SLT vía SUBOP comparados con los de SLT vía ALL

Los resultados espaciales del procedimiento SLT vía SUBOP no se incluyen porque coinciden completamente con los del procedimiento SLT vía ALL ya que realiza las mismas operaciones, tanto a nivel de reglas de inferencia como de podas. Se puede comprobar que los resultados obtenidos a nivel global por el SLT vía SUBOP no suponen una mejora sustancial con los que obtiene el SLT vía ALL. El motivo de ello es el tipo de problemas que se están manejando en la librería *TPTP* cuya elección de ancestros subóptima no tiene un tamaño muy inferior que la elección ALL, a nivel global. Se puede observar que los dominios donde la elección SUBOP es más pequeña que la elección ALL, en proporción, se obtienen unos mejores resultados. Cabe destacar que la mayor mejora fue de un 33% en el problema SYN333-1 y el mayor empeoramiento fue de un 9,4% en el problema LCL200-1. Como se puede ver en la tabla existen algunos dominios donde la ventaja que obtiene SLT vía SUBOP al utilizar conjuntos de ancestros menores se ve compensada e incluso sobrepasada por la desventaja de tener que comprobar al seleccionar un literal si pertenece o no a la elección de ancestros. Aquellos dominios donde la elección de ancestros SUBOP difiere poco de la ALL es donde se da este fenómeno con mayor grado.

Del conjunto de datos incluidos y del análisis realizado se puede concluir que para el conjunto de problemas que se ha tratado en el presente trabajo la elección de ancestros ALL es la más adecuada para obtener un procedimiento SLT que permita abordar los problemas con garantías de éxito y con una eficiencia considerable. Este resultado no es sorprendente, ya que los problemas abordados caen directamente en el campo de la demostración automática y tiene unas características (presencia de la teoría de la igualdad, número de cláusulas que no son Horn, profundidad máxima de los términos, etc.) que los hacen especialmente complejos para utilizar las otras elecciones de ancestros.

COMPARACIÓN CON OTROS RESULTADOS

Como ya se ha comentado en varias ocasiones, el objetivo de la implementación desarrollada de resolución SL* (y en particular del procedimiento SLT) no era el conseguir un demostrador eficiente y capaz de demostrar la mayoría de los problemas con resultados similares o mejores que los de otros demostradores, sino el permitir contrastar

experimentalmente las aportaciones teóricas del presente trabajo. Aún así, se ha considerado interesante comparar los resultados obtenidos por el procedimiento SLT vía ALL, que es la elección de ancestros con mejor comportamiento para los problemas tratados, con los resultados de los procedimientos y sistemas más cercanos el PTTP y el SETHEO. Antes de incluir los resultados indicar que la comparación estrictamente basada en los datos resulta muy difícil ya que los entornos de implementación no son similares y por lo tanto sería necesario realizar una ponderación por este motivo.

A continuación se incluye una tabla con los resultados para un conjunto de problemas comunes de los siguientes procedimientos y/o sistemas: SLT vía ALL, SETEHO [Letz, Schuman, Bayerl y Bibel, 1992], la implementación del PTTP en Lisp [Stickel, 1988] y la implementación del PTTP en Prolog [Stickel, 1992]. Los entornos de implementación han sido los siguientes, respectivamente: SICSTUS Prolog en SUN SPARC 5, C en SUN SPARC 1, Common Lisp en Symbolics 3600 y Prolog en Symbolics 3600. Los tiempos se dan en segundos.

Problema	SLT vía ALL	SETHEO	PTTP (Lisp)	PTTP (Prolog)
cl1	0,01	0,01	0,00	0,00
cl2	0,32	0,55	0,37	0,64
cl3	0,05	0,01	0,05	0,09
cl4	0,01	0,01	0,00	0,01
cl5	0,00	0,01	0,00	0,00
cl6	0,02	0,01	0,00	0,01
cl7	0,01	0,01	0,00	0,00
cl8	0,51	0,48	0,65	2,52
cl9	0,03	0.1	0,03	0,14
steamroller	15,93	0,6	-	-
wos1	14,82	2,05	57,6	-

Tabla 16: Resultados comparativos de diferentes procedimientos y sistemas[24]

Como se puede observar en la tabla, los resultados del SLT vía ALL son muy aceptables e incluso ligeramente mejores que los de la versión Lisp del PTTP y bastante mejores que la versión implementada en Prolog del PTTP. Comprando los resultados del SLT vía ALL con los del SETHEO se puede ver que son similares e incluso mejores en algunos problemas excepto en *steamroller* y *wos*1. El motivo de esta acusada diferencia es que para estos problemas el resultado lo obtiene SETHEO usando una busqueda en profundidad iterada con determinación de la profundidad mediante el tamaño del conjunto de ancestros. Si se comparan con los resultados de SETHEO para la determinación de la profundidad mediante el número de pasos de las reglas de inferencia (12,7 para *steamroller* y 26,2 para *wos*1) se puede concluir que los resultados son muy aceptables.

[24] Los guiones en las celdas indican que no se ha podido encontrar el resultado para el problema en la literatura consultada.

4.6. RESOLUCIÓN SL* FRENTE A OTROS DEMOSTRADORES DE TEOREMAS

Resolución SL* se puede ver como un refinamiento del procedimiento Eliminación de Modelos o de resolución SL. La definición de resolución SL* se ha dado de forma totalmente rigurosa y se ha intentado mantenerla lo más cercana posible al campo de la programación lógica. Así, a diferencia de Eliminación de Modelos o de resolución SL, la definición de SL* es fácilmente comprensible para las personas familiarizadas con este campo. También, se cree que la definición de resolución SL* es clara y directa, permitiendo en todo momento saber cuál es la teoría sobre la que se trabaja (cláusulas iniciales junto con el conjunto de ancestros), cuál es la meta a refutar (la cláusula en curso en la derivación), cuáles son las operaciones que se pueden aplicar (resolución de entrada, resolución de ancestro y regla de división) y qué significado tiene cada una de ellas y cómo se han de interpretar las respuestas que se obtienen de la computación de un cierto problema. A parte de la aportación que corresponde a la definición de SL*, existen otras dos aportaciones destacables:

- Flexibilidad a la hora de utilizar resolución SL* para demostración de teoremas o resolución de problemas: fácilmente se pueden obtener los dos procedimientos fundamentales de resolución SL*, el procedimiento SLT y SLP, que están enfocados cada uno de ellos a estas dos tareas. La manera de obtener estos procedimientos consiste en definir cuál va a ser la teoría y el conjunto de ancestros inicial. Si la cláusula inicial está incluida en la teoría y el conjunto de ancestros es vacío se obtiene el procedimiento SLT para la demostración de teoremas; si la cláusula inicial no está incluida en la teoría y el conjunto de ancestros está compuesto por esta cláusula se obtiene el procedimiento SLP para la resolución de problemas. Así, se puede entender de una forma sencilla cual es la diferencia entre demostrar problemas (o comprobar la inconsistencia) y resolver problemas (u obtener respuestas definidas). Por otra parte también hay que destacar que si se están buscando respuestas indefinidas, se ha de emplear el procedimiento SLT para obtenerlas de las diferentes instancias de la cláusula raíz que se han empleado en una cierta refutación.

- Adecuación de resolución SL* a los problemas a tratar mediante el uso de diferentes elecciones de ancestros: como se ha mencionado anteriormente la elección de ancestros permite controlar la aplicación de resolución de entrada[25] de forma que es posible utilizar elecciones de ancestros que se adecuan al tipo de problema tratar. Esta adecuación del

[25] También permite controlar la aplicación de la poda por igualdad con ancestro, pero por tratarse de una poda y no tener implicaciones en la completitud, esta operación tiene un rango de importancia menor que la resolución de ancestro.

procedimiento mediante la elección de ancestros al problema a tratar trae consigo una utilización más precisa de los ancestros, recordando en la computación únicamente aquellos sobre los que potencialmente es posible aplicar un paso de resolución de ancestro. Incluso, para algunos tipos de problemas (near-Horn de rango restringido) el uso de la elección de ancestros adecuada (POS_BASE) permite que la aplicación de la regla de división se realice de forma que se computen en paralelo la computación subsidiaria y la de rango superior, incrementándose así la eficiencia del demostrador de forma considerable. Desde el punto de vista del autor de la presente tesis, la definición y uso de las elecciones de ancestros como mecanismo de control de la aplicación de la resolución de ancestro es la mayor aportación del trabajo presentado en la misma. La introducción de la elección de ancestros que permite una variabilidad en la aplicación de resolución de ancestro extiende ampliamente el que usa el refinamiento positivo que presenta D. Plaisted [Plaisted, 1990] ya que éste es una elección de ancestros particular para la cual este autor no da una justificación de su utilidad ni de su aplicación a ciertos tipos de subproblemas. Además hay que destacar que otros investigadores han cuestionado la aplicación de este refinamiento positivo de forma general [Stickel, 1992].

Respecto a la implementación realizada de resolución SL*, y en particular del procedimiento SLT, hay que destacar que, aun no siendo el principal objetivo de la presente tesis el desarrollo completo de un procedimiento, se ha conseguido un sistema con una eficiencia razonable y que ha permitido obtener resultados experimentales para su confrontación con las argumentaciones teóricas. Claramente existen otros sistemas que por su complejidad, potencia y eficiencia obtienen resultados claramente mejores que los presentados en las secciones anteriores. Entre estos sistemas se pueden citar como más cercanos el PTTP de M. Stickel [Stickel, 1992] y el SETHEO de W. Bibel y otros investigadores [Letz, Schuman, Bayerl y Bibel, 1992]. De los dos sistemas el PTTP es el más próximo por el uso de la tecnología Prolog para la implementación, aunque cabe destacar las siguientes diferencias:

- La implementación del procedimiento SLT se ha basado en una aproximación por metaprogramación, en la cual se distingue perfectamente entre el programa Prolog que representa el problema a tratar, que tiene una representación constante, y el programa Prolog que implementa el procedimiento SLT y en el cual se pueden incorporar modificaciones, que pueden tener que ver con la función de selección o la elección de ancestros u otras, de forma sencilla y clara y que no comportan una recompilación de los problemas a tratar.

- El PTTP usa como función de selección “primero por la izquierda”, la usual en los sistemas Prolog. A diferencia la implementación del procedimiento SLT emplea una función de selección que es una aproximación de coste lineal a “seleccionar el más instanciado”. Se ha considerado que esta función de selección es más adecuada porque, a

pesar de conllevar un coste mayor, permite una reducción importante del factor de ramificación del árbol y, por consiguiente, del espacio de búsqueda. Por otra parte esta función de selección también tiene implicaciones muy importantes en la poda de vuelta atrás aplicada en el procedimiento SLT, consiguiendo que ésta se aplique con mayor frecuencia y por tanto reduciendo también en mayor medida el espacio de búsqueda.

- La elección de ancestros que usa el PTTP es siempre ALL con lo cual asegura la aplicación de la regla de inferencia de resolución de ancestro siempre que sea posible y la aplicación de la poda por igualdad con ancestro siempre que se dé. El procedimiento SLT presentado es capaz de refinar esta elección de ancestros usando la denominada subóptima que permite la aplicación de resolución de ancestro y la poda por igualdad con ancestro siempre que sea posible pero teniendo que utilizar y almacenar conjuntos de ancestros menores. Esta ventaja en problemas clásicos del campo de la demostración de teoremas quizás no sea demasiado evidente, como demuestran los resultados obtenidos, pero en otros campos en los cuales sea necesario la demostración de teoremas o la comprobación de la inconsistencia esta ventaja es mucho más clara.

De la comparación con el sistema SETHEO se puede concluir que el preprocesamiento que se realiza en la implementación presentada puede ser ampliamente mejorada incorporando ciertas reducciones del SETHEO que permitirían disminuir la complejidad del problema a tratar. Un análisis detallado de las reducciones que se aplican en el sistema SETHEO conducen a comprender que las reducciones que se puedan realizar en tiempo de compilación y que no conlleven un coste muy elevado han de ser aplicadas ya que permiten reducir la complejidad del problema en numerosas ocasiones. Esta reducción puede ser la clave, en el caso más extremo, de la resolución del problema. Otro aspecto interesante a comparar con el sistema SETHEO es la función de selección. En este sistema, a diferencia de la implementación de resolución SL*, la función de selección es “primero por la izquierda” pero en el preprocesamiento se realiza un ordenamiento de los literales de las contrapositivas buscando aquel que, potencialmente, dé lugar al factor de ramificación del árbol de estados más pequeño. El ordenamiento está basado en tres datos: la desinstanciación del literal, el número de variables que comparte con la cabeza de la contrapositiva y el número de resolventes que puede generar. Esta aproximación permite obtener unos factores de ramificación en el árbol de estados muy contenidos con un coste muy reducido. La implementación elegida de resolución SL* utiliza una aproximación totalmente distinta ya que la función de selección es una aproximación de coste lineal de “el más instanciado”. Claramente el coste en compilación de la función de selección de la implementación de SL* es nulo, mejorando el coste de la del sistema SETHEO. Sin embargo el coste en ejecución de la función de selección del sistema SETHEO es menor que la de la implementación del resolución SL*. La comparación se hace difícil ya que se tendría que disponer de costes temporales y espaciales de ambos sistemas en implementaciones equivalentes

para poder llegar a alguna conclusión precisa. Tal vez, una función de selección que incorpore los aspectos más adecuados de ambas aproximaciones permitiría conseguir reducciones más elevadas del factor de ramificación del árbol de estados, y por tanto del espacio de búsqueda, con unos costes temporales más reducidos.

CONCLUSIONES

Las principales conclusiones del trabajo presentado se pueden dividir según se trate de aspectos teóricos o experimentales. A nivel teórico se ha desarrollado resolución SL*, un paradigma basado en resolución lineal para la demostración automática. Claramente la aproximación tomada en resolución SL* ya ha sido explotada por otros demostradores y sistemas aunque es posible destacar como aportaciones propias las siguientes: por un lado la definición de resolución SL*, que es detallada, rigurosa y formulada dentro del campo de la programación lógica. Esta definición permite clarificar el funcionamiento de resolución SL*, entendiendo en cualquier punto de sus computaciones cuales son los pasos de inferencia utilizados (resolución de entrada, resolución de ancestro y regla de división), sobre qué objetos se aplica (cláusula actual en la derivación, conjunto de ancestros, cláusula de entrada) y cuál es la teoría utilizada (cláusula de entrada y conjunto de ancestros). El autor del presente trabajo piensa que esta formulación de resolución SL* es una aportación importante ya que evita la ardua tarea que tienen que realizar las personas que trabajan en el campo de la Programación Lógica pero que no son expertas en el campo de la demostración automática para estudiar y entender procedimientos o sistemas como Eliminación de Modelos, resolución SL, MESON, SETHEO, etc.

Por otro parte, se ha definido resolución SL* como un paradigma, parametrizándolo mediante el conjunto de ancestros y la elección de ancestros. La utilización de un conjunto de ancestros parametrizable ha permitido englobar en una sola propuesta las aproximaciones de comprobación de la inconsistencia y obtención de respuestas, dejando muy claro en dónde radica la diferencia. Sobre la elección de ancestros es necesario extenderse ampliamente ya que se considera que es una de las principales aportaciones. La elección de ancestros se puede entender como un mecanismo de control, de igual forma que la función de selección, que permite influir en la manera en la que la resolución de ancestro va a ser aplicada en las computaciones. Como ya se comentó en el capítulo 4, uno de los principales objetivos era extender el procedimiento SLD (o más en general, resolución de entrada) para conseguir alcanzar la completitud en el tratamiento de teorías clausales generales, sin perder por ello la simplicidad y el contexto formal del SLD. Para alcanzar este objetivo se incorpora como una nueva regla de inferencia la resolución de ancestro, pero a diferencia de otras propuestas, su aplicación no es universal en todo paso de las computaciones, sino que está controlada por la elección de ancestros. De esta forma es posible, tomando cierta elección de ancestros, adecuar el procedimiento instancia de resolución SL* para un tipo de teorías a tratar. Esta aportación que hasta donde sabe el autor es novedosa, establece un nuevo mecanismo de control, que

permite al usuario de resolución SL* orientarla a los problemas que vaya a tratar. Por otro lado también, se ha presentado la implicación de la elección de ancestros en la eficiencia de los procedimientos, estudiando la relación de ésta con la poda por igualdad con ancestro. Esta poda ampliamente aplicada por muchas aproximaciones de la literatura también se ve influida por la elección de ancestros que se utilice, permitiendo establecer un equilibrio entre la mejora del coste espacial que se obtiene por su aplicación y el empeoramiento del coste temporal por su computación. Por último, destacar también, que se ha presentado un método sintáctico que permite calcular, dado un problema a tratar, cual es la elección de ancestros óptima (o al menos la subóptima) tanto para la aplicación de la resolución de ancestro como de la poda por igualdad con ancestro. Así, es posible obtener el procedimiento instancia de resolución SL* que se adapta certeramente al problema a tratar.

Con respecto a las aportaciones concernientes a los aspectos experimentales, hay que indicar que no ha sido un objetivo fundamental obtener una implementación eficiente de resolución SL* sino más bien el desarrollar una implementación correcta que permita contrastar los aspectos teóricos a nivel experimental de forma ágil y flexible. Por este motivo se ha utilizado un entorno Prolog, que permite un desarrollo rápido y claro evitando la necesidad de implementar muchos aspectos. En el desarrollo se pueden distinguir varias partes, como son: la precompilación, el procedimiento instancia de resolución SL* utilizado y aspectos propiamente de implementación (algoritmo de unificación, recorrido del espacio de búsqueda, función de selección, podas, etc.). Con respecto a la precompilación se han seguido las ideas desarrolladas en el SETHEO [Letz, Schuman, Bayerl y Bibel, 1992]. Así, esta precompilación, además de transformar el problema a tratar al formato que se ha elegido, realiza reducciones orientadas a mejorar la eficiencia del trabajo posterior del procedimiento (esencialmente, el cálculo del tamaño de cada cláusula, la aplicación restringida del algoritmo de unificación correcto y la obtención de la elección de ancestro subóptima). En lo referente a la obtención del procedimiento instancia de resolución SL* hay que destacar que se ha aplicado la metaprogramación, en vez de utilizar aproximaciones como la de M. Stickel en el PTTP que obtiene una representación única que engloba al demostrador y al problema a tratar [Stickel, 1992]. De esta forma se obtiene un programa Prolog que corresponde a un procedimiento instancia de SL* que como entrada toma un programa Prolog que corresponde al problema a tratar. Así, se separa la representación de los problemas de la de los procedimientos, permitiendo aplicar a los mismos problemas diferentes procedimientos sin necesidad de obtener nuevas representaciones. En lo referente a los aspectos más propios de implementación, es importante destacar los siguientes puntos: se ha optimizado el uso del algoritmo de unificación correcto mediante las técnicas de precompilación anteriormente descritas; el recorrido del espacio de búsqueda se ha realizado mediante la búsqueda en profundidad iterada limitada por el numero de pasos de inferencia realizado, garantizando así la completitud y un crecimiento suave del espacio de búsqueda en iteraciones sucesivas; se ha utilizado una función de selección que es

una aproximación lineal a la que obtiene el literal más instanciado de una cláusula, de este modo se mejora la eficiencia sin perder por ello una reducción considerable del factor de ramificación del árbol de estados y por consiguiente del espacio de búsqueda; y por último, en lo referente a las podas se han implementado dos, la poda por igualdad con ancestro y la poda por respuesta más general, que permiten una reducción considerable del espacio de búsqueda y no tienen un coste demasiado elevado. Destacar también que el trabajo experimental realizado permitió detectar una problemática asociada a la interacción entre la búsqueda en profundidad iterada y la poda por respuesta más general, que, hasta donde conoce el autor de la presente tesis, no aparece citada en la literatura. Como se comentó esta problemática que impide que se localice la refutación de menor longitud, puede provocar, en el mejor de los casos, una ineficiencia considerable y, en el peor de ellos, la pérdida de la completitud. Por último, añadir que se han abordado la demostración de los problemas de la librería *TPTP* de [Suttner y Sutcliffe, 1996], junto con un conjunto de problemas clásicos de la demostración automática (los problemas *cl*1 al *cl*9 de [Chang y Lee, 1973], el problema *wos*1 de [Wos y Overbeek, 1993] y el problema *steamroller* de [Stickel, 1986]). Los problemas de la librería *TPTP* se han utilizado para analizar el comportamiento del procedimiento SLT vía las elecciones de ancestros ALL, POS, EMPTY y SUBOP. Los resultados, tanto temporales como espaciales, han permitido observar la capacidad del procedimiento SLT vía cada una de estas elecciones para tratar diferentes tipos de problemas (dependiendo de si incluyen o no teoría de la igualdad, de el número de predicados, de la cantidad de cláusulas unitarias, disyuntivas, Horn, etc.). Del análisis realizado se ha podido concluir que la elección de ancestros ALL es la más adecuada para el tratamiento de problemas del ámbito clásico de la demostración automática, ya que para estos EMPTY no es completo y POS obtiene peores resultados, en general. En cuanto a la elección SUPOP comentar que para este tipo de problemas obtiene unos en resultados temporales en ejecución muy parecidos a la elección ALL, mejorando discretamente para un conjunto pequeño de problemas y empeorando, también discretamente, para muy pocos problemas. Teniendo en cuenta el coste de calcular la elección SUBOP en compilación, se concluye que la elección ALL es la más adecuada. Los problemas tratados que no corresponden a la librería *TPTP* han servido para realizar una pequeña comparación con los resultados que obtienen el PTTP, tanto en su versión implementada Lisp como en Prolog, y el SETHEO. Se ha podido observar que los resultados obtenidos son comparables e incluso en algunos casos superiores a los de estos demostradores, indicando las posibilidades que tiene la aproximación tomada para la implementación de resolución SL*.

TRABAJO FUTURO

El trabajo futuro que se pretende realizar está orientado tanto a aspectos teóricos o como experimentales. A continuación se presentan los principales puntos de este trabajo:

- Estudio de la posibilidad de definir resolución SL* de forma que la aplicación de la regla de división sea mediante computaciones paralelas en vez de subsidiarias. Claramente, existen algunas teorías y elecciones de ancestros que presentan un mejor comportamiento para éstas que son perfectamente adecuadas a esta aproximación, como son las teorías de rango restringido y las elecciones de ancestro base, como ya se indicó en el capítulo 4. El objetivo sería estudiar la posibilidad de generalizar la aplicación en paralelo de la regla de división.

- Desarrollo de una implementación de la versión de resolución SL* con la aplicación de la regla de la división mediante computaciones paralelas y el estudio experimental de su adecuación a diferentes tipos de teorías. Como marco de implementación parece conveniente la utilización de un sistema Prolog con paralelismo.

- Incorporación a resolución SL* de reglas de inferencia especialmente adecuadas para el tratamiento de la igualdad, como son demodulación, paramodulación, etc. [Wos, Robinson, Carson y Shalla, 1967] [Wos y Robinson, 1970]. Como se ha podido comprobar en los resultados experimentales, con bastante frecuencia resolución SL* es incapaz de resolver los problemas que incorporan igualdad debido a las especiales características de ésta. Por ello, se cree que es imprescindible introducir reglas de inferencia más potentes y que estén orientadas al tratamiento de la igualdad.

- Estudio de la posibilidad de utilización de otros mecanismos en resolución SL* como son la factorización, uso de lemas, subsumción, etc. Aunque estos mecanismos pueden ser fácilmente incorporados a resolución SL*, lo importante es investigar las ventajas y desventajas de su utilización, tanto de forma teórica como experimental. Mención especial cabe hacer a la subsumción, el autor del presente trabajo opina que es imprescindible su incorporación a los demostradores si se quiere que éstos sean capaces de abordar problemas de los denominados “profundos” o “complejos”. Se es de la misma opinión que los autores del trabajo [Wos y Overbeek, 1993], que argumentan y defienden la necesidad de la utilización de la subsumción para el tratamiento de este tipo de problemas.

- Mejora del proceso de precompilación, incorporándole reducciones como las del sistema SETHEO [Letz, Schuman, Bayerl y Bibel, 1992]. Como ya se indicó al presentar el sistema SETHEO estas reducciones permiten simplificar, e incluso resolver, gran número de problemas con unos costes reducidos. Además, estas reducciones en muchos casos permiten que posteriormente el demostrador sea capaz de resolver el problema considerado.

- Estudio de una función de selección que permita una mayor reducción del factor de ramificación del árbol de estados. Este aspecto es muy importante porque las pruebas

realizadas con diferentes funciones de selección demostraron la influencia que tiene ésta en el tamaño del espacio de búsqueda. Una función de selección que esté basada en la información que se pueda extraer en la fase de precompilación sobre la ramificación que producen cada uno de los literales de las cláusulas, siguiendo las aproximaciones presentadas en [Kowalski y Kuehner, 1971] y en [Letz, Schuman, Bayerl y Bibel, 1992], parece la más conveniente.

- Análogo al estudio experimental que se ha realizado en la presente tesis centrado sobre un conjunto de problemas del campo de la demostración automática, se propone un estudio tanto teórico como experimental sobre otros tipos de problemas de campos en los que se utilice el razonamiento automático como son bases de datos deductivas (comprobación de la integridad de restricciones generales, bases de datos disyuntivas, etc.), sistemas expertos, planificación, etc. Es este estudio se buscarían las instancias de resolución SL* más adecuadas a cada uno de ellos.

BIBLIOGRAFÍA

[Apt y van Emden, 1982]
K.R. Apt y M.H. van Emden, *Contributions to the Theory of Logic Programming*, en Journal of ACM 29, 3, 841-862, 1982.

[Bibel, 1987]
W. Bibel, *Automated Theorem Proving*, Vieweg Verlag, Braunschweig, segunda edición, 1987.

[Casamayor, Decker y Marqués, 1993]
J.C. Casamayor, H. Decker y F. Marqués, *A mechanism for Verificating Knowledge Scheme Specifications*, en proc. EUROVAV'93 (First European Conf. on Verification and Validation), ISBN: 8460458172, pag. 103-116. 1993.

[Casamayor y Decker, 1995]
J.C. Casamayor y H. Decker, *A Hypothetical Query Answering Procedure for first-order databases*, en A. Aamodt y J. Kamorowski (eds.), proc. Scandinavian Conf. on Artificial Intelligence, ISBN: 9051992211, pags. 368-372, IOS Press. 1995.

[Cavedon, 1991]
L. Cavedon, *Acyclic logic programs and the completeness of SLDNF resolution*, en Theoretical Computer Science, 86, 81-92, 1991.

[Chang y Lee, 1973]
C.L. Chang y R.C.T. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.

[Clark, 1978]
K.L. Clark, *Negation as Failure*, en Logic and Databases (H. Gallaire y J. Minker eds.), 293-322, Plenum Press, New York, 1978.

[Cuena, 1985]
J. Cuena, *Lógica Informática*, Alianza Editorial, Madrid, 1985.

[Decker, 1988]
H. Decker, *Domain-independent and range-restricted formulas and deductive databases*, en S. Bourgault y M. Dincbas (eds.), *Programming en Logique* (Actes du 7ème séminire), Trégastel (Francia), 385-397, 1988.

[Decker, 1991]
H. Decker, *On generalized cover axioms*, en proc. 8th Int. Conference on Logic Programming, 693-707, MIT Press, 1991.

[Decker y Casamayor, 1993]
H. Decker y J.C. Casamayor, *A Prolog-like paradigm for reasoning in first-order theories*, en D. Saccà (ed.), proc. GULP'93 (*Atti dell' Ottavo Convegno sulla Programmazione Logica*), pags. 217-233, Mediterranean Press, 1993.

[Demolombe, 1994]
R. Demolombe, *A syntactical characterization of a subset of domain indepent formulas*, en Journal of ACM ,

[Eder, 1985]
E. Eder, *Properties of substitutions and unifications*, Journal of Symbolic Computation, 1, 245-260, 1985

[Eshghi y Kowalski, 1989]
K. Eshghi y R.A. Kowalski, *Abduction compared with negation as failure*, en proc. 6th Int. Conference on Logic Programming, MIT Press, 1989.

[Gardner, 1985]
G. Gardner, *Máquina y diagramas lógicos*, Alianza Editorial, Madrid 1985

[Garey y Johnson, 1979]
M.R. Garey y D.S. Johnson, *Computers and Intractability. A guide of the Theory of NP-Completeness*, Freeman & Co., New York, 1979.

[Hamilton, 1981]
A.G. Hamilton, *Lógica para matemáticos*, Paraninfo, Madrid, 1981.

[Horowitz, 1978]
E. Horowitz, *Fundamentals of Computer Algorithms*, Pitman Pub. Ltd., 1978.

[Kowalski y Kuehner, 1971]
R.A. Kowalski y D. Kuehner, *Linear resolution with selection function*, Artificial Intelligence 2, 227-260.

[Kowalski, 1974]
R.A. Kowalski, *Predicate Logic as a Programming Language*, Proc. del 6º IFIP Congress, 569-574, North-Holland, 1974.

[Kowalski, 1975]
R.A. Kowalski, *A Proof Procedure using Connection Graphs*, en Journal of ACM 22, 4, 572-595, 1975.

[Knight, 1989]
K. Knight, *Unification: A Multidisciplinary survey*, ACM, Computing surveys 21, 93-124, 1989.

[Letz, Schuman, Bayerl y Bibel, 1992]
R. Letz, J. Schuman, S. Bayerl y W. Bibel, *SETHEO: A High-Performance Theorem Prover*, Journal of Automated Theorem 8, 183-212, 1992.

[Lloyd, 1987]
J.W. Lloyd, *Foundations of Logic Programming*, Springer-Verlag, 1987.

[Loveland, 1968]
D.W. Loveland, *Mechanical theorem proving by model elimination*, Journal of ACM, 15, 236-251, 1968.

[Loveland, 1978]
D.W. Loveland, *Automated Theorem Proving: a Logical Basis*, North Holland, Amsterdam 1978.

[Loveland, 1987]
D.W. Loveland, *Near-Horn Prolog*, en proc. 4th Int. Conf. on Logic Programming (ed. J. Lassez), 456-469, MIT Press, 1987

[Smith y Loveland, 1988]
B.T. Smith y D.W. Loveland, *A Simple near-Horn Prolog Interpreter*, en proc 5th Int. Conf. on Logic Programming, (ed. R.A. Kowalski y K.A. Bowen), MIT Press, 1988.

[Loveland, 1991]
D.W. Loveland, *Near-Horn Prolog and Beyond*, Journal of Automated Reasoning 7, 1-26, 1991.

[Reed, Loveland y Smith, 1991]
D.W. Reed, D.W. Loveland y B.T. Smith, *The Near-Horn Approach to Disjunctive Logic Programming*, proc. 2nd Int. Workshop on Extensions of Logic Programming, Lecture Notes in Artificial Intelligence, Springer-Verlag, 1991.

[Loveland y Reed, 1993]
D.W. Loveland y D.W. Reed, *A Near-Horn Prolog for Compilation*, en Computational Logic, essays in honour of Alan Robinson, MIT Press, 1993.

[Loveland, Reed y Wilson, 1995]
D.W. Loveland, D.W. Reed y D.S. Wilson, *SATCHMORE: SATCHMO with Relevancy*, en Journal of Automated Reasoning 14, 325-351, 1995.

[Luckham, 1970]
D. Luckham, *Refinement theorems in resolution theory*, Sym. on Automatic Demonstration, Lecture Notes in Math. 125, 163-190, Springer-Verlag, Berlin, 1970.

[Manthey y Bry, 1988]
R. Manthey y F. Bry, *SATCHMO: a theorem prover implemented in Prolog*, proc. of 9th CADE, Argonne 1988.

[Marqués, 1992]
F. Marqués, *Implementación de un procedimiento de resolución por filtrado de ancestros en LISP*, trabajo 5º de Inteligencia Artificial, Valencia, 1992.

[McCune, 1990]
W. McCune, *OTTER 2.0 users guide*, informe técnico ANL-90/9, Argonne National Laboratories, Argonne, 1990.

[Minker y Zanon, 1982]
J. Minker y G. Zanon, *An extension to Linear Resolution with selection function*, en Information Processing Letters 14, 4, 191-193, 1982.

[Minker y Rajasekar, 1990]
J. Minker y A. Rajasekar, *A fixpoint semantics for disjunctive logic programs*, en Journal of Logic Programming 9(1), 45-74, 1990.

[Nilson, 1987]
N.J. Nilson, , *Principios de la Inteligencia Artificial*, Díaz de Santos, Madrid 1987.

[Plaisted, 1988]
D.A. Plaisted, *Non-Horn clause logic programming without contrapositives*, en Journal of Automated Reasoning 4, 287-325, 1988.

[Plaisted, 1990]
D.A. Plaisted, *A sequent-style model elimination strategy and a positive refinement*, en Journal of Automated Reasoning 6, 389-402, 1990.

[Ramsay, 1991]
A. Ramsay, *Generating relevant models*, en Journal of Automated Reasoning 7, 359-368, 1991.

[Robinson, 1965]
J.A. Robinson, *A machine-oriented logic based on Resolution Principle*, Journal of ACM 12, 23-41, 1965.

[Robinson, 1965b]
J.A. Robinson, *Automated deduction with hyper-resolution*, Int. J. Comp. Math. 1, 227-234, 1965.

[Robinson, 1979]
J.A. Robinson, *Logic: Form and Function*, University Press, Edinburgh, 1979.

[Robinson, 1992]
J.A. Robinson, *Logic and Logic Programming*, Journal of ACM 35, 42-65, 1992.

[Shepherdson, 1991]
J. C. Shepherdson, *Correct Answers to Allowed Programs and Queries are Ground*, en Journal of Logic Programming 11, 359-362, 1991.

[Simon, 1973]
H.A. Simon, *Las ciencias de lo artificial*, A.T.E., Barcelona, 1973.

[Stickel, 1984]
M.E. Stickel, *A Prolog technology theorem prover*, New Generation Computing 2, 371-383.

[Stickel, 1986]
M.E. Stickel, *Schubert´s steamroller problem: Formulations y solutions*, en Journal of Automated Reasoning 2, 89-101, 1986.

[Stickel, 1988]
M.E. Stickel, *A Prolog Technology Theorem Prover: Implementation by an extended Prolog compiler*, en Journal of Automated Reasoning 4, 353-380, 1988.

[Stickel, 1992]
M.E. Stickel, *A Prolog technology theorem prover: a new exposition and implementation in Prolog*, en Theoretical Computer Science 104, 109-128.

[Suttner y Sutcliffe, 1996]
C.B. Suttner y G. Sutcliffe, *The TPTP Problem Library* (TPTP v1.2.1 - TR Date 18.6.96), en informe técnico AR-96-02, Institut für Informatik, Technische Universität München, Munich, Alemania, 1996.

[Vielle, 1986]
L. Vielle, *Recursive Axioms in Deductive Databases: The Query-Subquery Approach*, en proc. 1th Conference on Expert Database Systems, 1986.

[Walther, 1985]
C. Walther, *A mechanical solution of Schubert´s steamroller by many-sorted resolution*, Artificial Intelligence 26, 217-224, 1985.

[Wos, Robinson, Carson y Shalla, 1967]
L. Wos, G. Robinson, D. Carson y L. Shalla, *The concept of demodulation in Theorem Proving*, Journal of ACM 14, 698-709, 1967.

[Wos y Robinson, 1970]
L. Wos y G. Robinson, *Paramodulation and set support*, en Sym. on Automated Demonstration, Lecture Notes in Math. 125, Springer-Verlag, Berlín, 1970, 276-310.

[Wos y Overbeek, 1993]
L. Wos, y R. Overbeek, *Subsumption, a sometimes undervalued procedure*, en Computational Logic, essays in honour of Alan Robinson, MIT press, 1993.

APÉNDICE A: DEMOSTRACIONES

En este apéndice se van a incluir las demostraciones de los resultados que se han dado en el capítulo 4.

Teorema 4.1 (Corrección de SL* resolución)

Sea T una teoría, C una cláusula de la forma $H \leftarrow B$, Anc un conjunto de cláusulas y Ch una elección de ancestros. Si existe una refutación SL* desde C en T vía Ch usando Anc con substitución computada θ, entonces $T \cup Anc\theta \models \forall((\neg H \wedge B)\theta)$.

Demostración:

Sea C de la forma $A_1 \vee \ldots \vee A_g \leftarrow D_1 \wedge \ldots \wedge D_h$ ($g \geq 0$, $h \geq 0$ y $g+h>0$). La demostración se realiza por doble inducción en el rango k de la refutación y la longitud n de la misma.

Se probará primero para rango $k=0$. Supóngase que la longitud de la refutación es $n=1$. Eso significa que C es de la forma $A_1 \leftarrow$ (resp. $\leftarrow D_1$), y que existe una cláusula perteneciente a la teoría T o al conjunto de ancestros Anc de la forma $A \leftarrow$ (resp. $D \leftarrow$) y $A_1\theta_1 = A\theta_1$ (resp. $D_1\theta_1 = D\theta_1$). Ya que la cláusula $\leftarrow A_1\theta_1$ (resp. $D_1\theta_1 \leftarrow$) es una instancia de una cláusula de T o $Anc\theta_1$, se cumple que $T \cup Anc\theta \models \forall((\neg A_1\theta_1)$ (resp. que $T \cup Anc\theta \models \forall((D_1\theta_1)$). Por tanto, el resultado se cumple ya que $\theta = \theta_1$.

Ahora se supondrá que el resultado se cumple para refutaciones de longitud n-1. Sea $\theta_1, \ldots, \theta_n$ la secuencia de umgs usados en la refutación de longitud n. Como el rango de la refutación es 0 sólo se aplica resolución de entrada para obtener la siguiente cláusula en la refutación. Sea L es el literal seleccionado en C_0 y sea C_1' la primera cláusula de entrada tomada de T o Anc. La siguiente cláusula en la refutación C_1, de la forma $H_1 \leftarrow B_1$, es el resolvente de C_0 y C_1' sobre L y un literal L' de C_1' usando el umg θ_1. Ya que existe una refutación desde C_1 en T usando $Anc\theta_1$ de longitud n-1 con sustitución computada $\theta_2 \ldots \theta_n$, por la hipótesis de inducción se cumple que $T \cup Anc\theta_1\theta_2 \ldots \theta_n \models (\neg H_1 \wedge B_1)\theta_2 \ldots \theta_n$. Teniendo en cuenta que C_1 es el resolvente de C_0 y C_1', que $\theta_1\theta_2 \ldots \theta_n = \theta$ y que $C_1'\theta$ es una instancia de T o $Anc\theta$ se puede concluir que $T \cup Anc\theta \models \forall((\neg H \wedge B)\theta)$.

Se supone ahora que el resultado se cumple para refutaciones de rango k-1. Supóngase que el rango de la refutación es k y la longitud de la refutación es n. La demostración se realiza por inducción sobre el número s de refutaciones subsidiarias que se lanzan desde la refutación. Para $s=1$, se supondrá que la refutación subsidiaria se llama en el primer paso de la refutación.

De no ser así, basta con tomar la subderivación que parte de la cláusula donde se encuentra el literal sobre el que existe la primera refutación subsidiaria para realizar la demostración y, posteriormente, tomar la subderivación de entrada desde la raíz inicial hasta la cláusula donde se encuentra dicho literal y combinar ambos resultados. Por tanto, sea L_0 el literal seleccionado en C_0. Así, existe una refutación SL* subsidiaria desde L_0 en T usando $Anc \cup \{L_0\}$ de rango menor que k con sustitución computada θ_1. Por tanto se cumple que $T \cup Anc\theta_1 \models \forall(\widetilde{L}_0\theta_1)$. Por otra parte, sea C_1 es la siguiente cláusula en la refutación de la forma $H_1 \leftarrow B_1$ para la cual existe una refutación SL* desde C_1 en T usando $Anc\theta_1$ de rango 0 con sustitución computada $\theta_2 \ldots \theta_n$. Por tanto, se cumple que $T \cup Anc\theta_1\theta_2 \ldots \theta_n \models (\neg H_1 \wedge B_1)\theta_2 \ldots \theta_n$. Empleando estos dos resultados y el hecho de que C_1 es el resolvente de C_0 y $\widetilde{L}_0\theta_1$ se puede concluir que el resultado $T \cup Anc\theta \models \forall((\neg H \wedge B)\theta)$.

Ahora se supondrá que el resultado se cumple para refutaciones SL* de rango k que usan s-1 refutaciones subsidiarias. Por los motivos anteriormente comentados, se supondrá que la refutación de rango k que usa s refutaciones subsidiarias, usa una de ellas en el primer paso de la refutación. Sea L_0 el literal seleccionado en C_0. Así, existe una refutación SL* subsidiaria desde L_0 en T usando $Anc \cup \{L_0\}$ de rango menor que k con sustitución computada θ_1. Por tanto se cumple que $T \cup Anc\theta_1 \models \forall(\widetilde{L}_0\theta_1)$. Por otra parte, sea C_1 la siguiente cláusula en la refutación de la forma $H_1 \leftarrow B_1$ para la cual existe una refutación desde C_1 en T usando $Anc\theta_1$ de rango k con sustitución computada $\theta_2 \ldots \theta_n$ que usa s-1 refutaciones subsidiarias. Por la hipótesis de inducción, se cumple que $T \cup Anc\theta_1\theta_2 \ldots \theta_n \models (\neg H_1 \wedge B_1)\theta_2 \ldots \theta_n$. Haciendo uso de estos dos resultados y del hecho de que C_1 es el resolvente de C_0 y $\widetilde{L}_0\theta_1$ se puede concluir que el resultado $T \cup Anc\theta \models \forall((\neg H \wedge B)\theta)$. ■

A continuación se introducirán dos resultados intermedios mediante dos lemas necesarios para demostrar la completitud de resolución SL* tanto para la demostración de teoremas, mediante el procedimiento SLT, como para resolución de problemas, mediante el procedimiento SLP. Para ello es necesario previamente dar la definición de refutación no restringida SL*. Esencialmente esta definición es idéntica a la Definición 4.4 con la salvedad de utilizar unificadores en vez de unificadores más generales. A continuación se dan estos dos lemas.

Lema 4.1 (Lema del umg)

Sea T una teoría, C una cláusula, Anc un conjunto de cláusulas y Ch una elección de ancestros homogénea. Si existe una refutación no restringida SL* desde C en T usando Anc vía Ch con secuencia de unificadores $\phi_1 \ldots \phi_n$, entonces existe una refutación SL* desde C en T usando Anc vía Ch del mismo rango y longitud, con la misma secuencia de cláusulas padre lejano y con la secuencia de umg $\theta_1 \ldots \theta_n$ de forma que $\theta_1 \ldots \theta_n = \phi_1 \ldots \phi_n\sigma$ para una cierta sustitución σ.

Demostración:

Sea C de la forma $A_1 \vee \ldots \vee A_g \leftarrow D_1 \wedge \ldots \wedge D_h$ ($g \geq 0$, $h \geq 0$ y $g+h>0$). La demostración se realiza por doble inducción en el rango k y la longitud n de la misma.

Se probará primero para rango k=0. Para la longitud n de la refutación igual a 1 el resultado es evidente, ya que entonces la refutación no restringida tiene como secuencia de cláusulas $C_0=C$ y $C_1=\square$, siendo por tanto C una cláusula unitaria. Por tanto existe una cláusula padre lejano C_1' de T o Anc tal que los átomos de C y C_1' unifican mediante ϕ_1. Sea θ_1 el umg entre estos dos átomos. Por tanto ha de existir una sustitución σ tal que $\phi_1=\theta_1\sigma$. Además también existe una refutación SL* desde C en T usando Anc vía Ch con secuencia de cláusulas $C_0=C$ y $C_1=\square$, secuencia de cláusulas padre lejano C_1' y secuencia de umg θ_1.

Ahora se supondrá que el resultado se cumple para refutaciones de longitud n-1. Sea $C_0=C$, C_1, …, $C_n=\square$ la secuencia de cláusulas de la refutación no restringida SL* desde C en T usando Anc vía Ch con secuencia de cláusulas padre lejano C_1', …, C_n' y secuencia de unificadores ϕ_1, …, ϕ_n. Sea L_0 el literal seleccionado en C_0 y sea L_1' el literal de C_1' sobre el cual se resuelve. Entonces existe un umg θ_1 entre los átomos de L_0 y L_1' de manera que $\phi_1=\theta_1\sigma$ para alguna sustitución σ. Ahora se puede construir una refutación no restringida SL* desde C usando Anc vía Ch con secuencia de cláusulas $C_0=C$, $\hat{C}_1$, C_2, …, $C_n=\square$ donde $C_1=\hat{C}_1\sigma$, secuencia de cláusula padre lejano C_1', …, C_n' y secuencia de unificadores θ_1, $\sigma\phi_2$, …, ϕ_n. Por la hipótesis de inducción, existe una refutación SL* desde $\hat{C}_1$ en T usando $Anc\theta_1$ con secuencia de cláusulas $\hat{C}_1, \hat{C}_2$, …, $\hat{C}_n=\square$, secuencia de cláusulas padre lejano C_2', …, C_n' y secuencia de umgs θ_2, …, θ_n, de forma que $\sigma\phi_2$, …, $\phi_n=\theta_2\ldots\theta_n\rho$, para una cierta sustitución ρ. Ahora es sencillo construir una refutación SL* desde C en T usando Anc vía Ch con secuencia de cláusulas $C_0=C$, $\hat{C}_1, \hat{C}_2$, …, $\hat{C}_n=\square$, secuencias padre lejano C_1', …, C_n' y secuencia de umgs θ_1, …, θ_n tal que $\phi_1\ldots\phi_n=\theta_1$, …, $\theta_n\rho$ ya que $\phi_1\ldots\phi_n=\theta_1\sigma\phi_2\ldots\phi_n$ y $\theta_1\sigma\phi_2\ldots\phi_n=\theta_1$, …, $\theta_n\rho$.

Ahora se supone que el resultado se cumple para refutaciones no restringidas de rango k-1. Supóngase que existe una refutación no restringida SL* desde C en T usando Anc vía Ch con secuencia de cláusulas $C_0=C$, …, $C_n=\square$, secuencia de cláusulas padre lejano C_1', …, C_n' y secuencia de unificadores $\phi_1\ldots\phi_n$ de rango k. La demostración para este caso se realizará por inducción sobre el número s de refutaciones subsidiarias. Para s=1, se supone que la refutación es llamada en el primer paso de la refutación. De no ser así, basta con tomar la subderivación que parte de la cláusula donde se encuentra el literal sobre el que existe la primera refutación subsidiaria para realizar la demostración y, posteriormente, tomar la subderivación de entrada desde la raíz inicial hasta la cláusula donde se encuentra dicho literal y combinar ambos resultados. Por tanto, sea L_0 el literal seleccionado en C_0 de forma que $C_1'=\tilde{L}_0\phi_1$. Existe una refutación no restringida desde L_0 en T usando $Anc \cup \{L_0\}$ de rango k-1 con sustitución computada ϕ_1. Por la hipótesis de inducción sobre el rango, existe una refutación SL* desde L_0 en T usando $Anc \cup \{L_0\}$ vía Ch del mismo rango y longitud con la misma secuencia de cláusulas

y de cláusulas padre lejano sustitución computada θ_1 tal que $\phi_1=\theta_1\sigma$ para una cierta sustitución σ. Así, existe una refutación no restringida SL* desde C en T usando Anc vía Ch con secuencia de cláusulas $C_0=C$, $\hat{C}_1$, C_2, …, $C_n=\square$ donde $C_1=\hat{C}_1\sigma$, secuencia de cláusula padre lejano C_1', …, C_n' y secuencia de unificadores θ_1, $\sigma\phi_2$, …, ϕ_n. Por tanto, aplicando el resultado para el caso de rango de la refutación igual a 0, existe una refutación SL* desde $\hat{C}_1$ en T usando $Anc\theta_1$ vía Ch con secuencia de cláusulas $\hat{C}_1, \hat{C}_2$, …, $\hat{C}_n=\square$, secuencia de cláusulas padre lejano C_2', …, C_n' y secuencia de umgs θ_2,…, θ_n, de forma que $\sigma\phi_2$, …, $\phi_n= \theta_2\ldots\theta_n\rho$, para una cierta sustitución ρ. Ahora es sencillo ver que existe una refutación SL* desde C en T usando Anc vía Ch con secuencia de cláusulas $C_0=C$, $\hat{C}_1, \hat{C}_2$, …, $\hat{C}_n=\square$, secuencias padre lejano C_1', …, C_n' y secuencia de umgs θ_1, …, θ_n tal que $\phi_1\ldots\phi_n= \theta_1$, …, $\theta_n\rho$ ya que $\phi_1\ldots\phi_n= \theta_1\sigma\phi_2\ldots\phi_n$ y $\theta_1\sigma\phi_2\ldots\phi_n=\theta_1$, …, $\theta_n\rho$.

Se supone que el resultado se cumple para refutaciones no restringidas SL* de rango k y que usan s-1 refutaciones subsidiarias. Supóngase que existe una refutación no restringida SL* desde C en T usando Anc vía Ch con secuencia de cláusulas $C_0=C$, …, $C_n=\square$, secuencia de cláusulas padre lejano C_1', …, C_n' y secuencia de unificadores $\phi_1\ldots\phi_n$ de rango k y que usa s refutaciones subsidiarias. Por los motivos anteriormente expuestos, se supone que esta refutación llama a una subsidiaria en el primer paso de la refutación. Sea L_0 el literal seleccionado en C_0 de forma que $C_1'= \tilde{L}_0\phi_1$. Por tanto, existe una refutación no restringida desde L_0 en T usando $Anc \cup \{L_0\}$ de rango menor que k con sustitución computada ϕ_1. Por la hipótesis de inducción sobre el rango, existe una refutación SL* desde L_0 en T usando $Anc \cup \{L_0\}$ vía Ch del mismo rango y longitud con la misma secuencia de cláusulas y de cláusulas padre lejano sustitución computada θ_1 tal que $\phi_1=\theta_1\sigma$ para una cierta sustitución σ. Así, existe una refutación no restringida SL* desde C en T usando Anc vía Ch con secuencia de cláusulas $C_0=C$, $\hat{C}_1$, C_2, …, $C_n=\square$ donde $C_1=\hat{C}_1\sigma$, secuencia de cláusula padre lejano C_1', …, C_n' y secuencia de unificadores θ_1, $\sigma\phi_2$, …, ϕ_n. Por tanto, aplicando la hipótesis de inducción, existe una refutación SL* desde $\hat{C}_1$ en T usando $Anc\theta_1$ vía Ch con secuencia de cláusulas $\hat{C}_1, \hat{C}_2$, …, $\hat{C}_n=\square$, secuencia de cláusulas padre lejano C_2', …, C_n' y secuencia de umgs θ_2, …, θ_n, de forma que $\sigma\phi_2$, …, $\phi_n= \theta_2\ldots\theta_n\rho$, para una cierta sustitución ρ. Hay que indicar que esta afirmación es cierta porque la elección de ancestros es homogénea, por tanto para cualquier literal seleccionado L en una cláusula de la refutación no restringida SL* que pertenece a Ch, se cumplirá que en la refutación SL* en el mismo punto y seleccionado el correspondiente literal $\hat{L}$, éste también pertenece a Ch por la definición de elección de ancestros homogénea. Ahora es sencillo ver que existe una refutación SL* desde C en T usando Anc vía Ch con secuencia de cláusulas $C_0=C$, $\hat{C}_1, \hat{C}_2$, …, $\hat{C}_n=\square$, secuencias padre lejano C_1', …, C_n' y secuencia de umgs θ_1, …, θ_n tal que $\phi_1\ldots\phi_n= \theta_1$, …, $\theta_n\rho$ ya que $\phi_1\ldots\phi_n= \theta_1\sigma\phi_2\ldots\phi_n$ y $\theta_1\sigma\phi_2\ldots\phi_n=\theta_1$, …, $\theta_n\rho$. ■

Lema 4.2 (Lema del alzamiento)

Sea *T* una teoría, *C* una cláusula, *Anc* un conjunto de cláusulas, *Ch* una elección de ancestros homogénea y θ una sustitución. Si existe una refutación SL* desde *C*θ en *T* usando *Anc*θ vía *Ch* con sustitución computada $\theta_1 \ldots \theta_n$, entonces existe una refutación SL* desde *C* en *T* usando *Anc* vía *Ch* del mismo rango y longitud con respuesta computada $\phi_1 \ldots \phi_n$ tal que $\theta\theta_1 \ldots \theta_n = \phi_1 \ldots \phi_n\sigma$ para alguna sustitución σ.

Demostración:

La demostración se realiza por inducción sobre el rango *k* de la refutación SL*. Para rango *k*=0, sea *L*θ el literal seleccionado en *C* en el primer paso de la refutación SL* desde *C*θ en *T* usando *Anc*θ vía *Ch*, *C*$_1$' la primera cláusula padre lejano, *L*' el literal seleccionado en *C*$_1$' y θ_1 un umg entre los átomos de *L*θ y *L*'. Así, $\theta\theta_1$ es un unificador de los átomos de *L* y *L*'. Por otra parte, si *C*$_1$' pertenece a *T* se supone que θ no actúa sobre *C*$_1$', por lo que el resultado de resolver *C* y *C*$_1$' usando el unificador $\theta\theta_1$ es la misma cláusula siguiente *C*$_1$; si *C*$_1$' pertenece a *Anc*θ, es de la forma $\hat{C}\theta$ y *L*' es de la forma $\hat{L}\theta$, por tanto $\theta\theta_1$ es unificador de los átomos de *L* y $\hat{L}$. De esta forma el resultado de resolver *C* y $\hat{C}$ usando $\theta\theta_1$ es también *C*$_1$. Por tanto, existe una refutación no restringida SL* desde *C* en *T* usando *Anc* vía *Ch*. Aplicando el Lema 4.1 el resultado se cumple.

Ahora se supone que el resultado se cumple para rango *k*-1. Supóngase una refutación SL* desde *C*θ en *T* usando *Anc*θ vía *Ch* de rango *k* cuya secuencia de cláusulas es *C*$_0$=*C*θ, …, *C*$_n$=□, secuencia de cláusulas padre lejano *C*$_1$', …, *C*$_n$' y secuencia de umgs $\theta_1 \ldots \theta_n$. Supóngase que una refutación subsidiaria es usada en el primer paso de la refutación. De no ser así, basta con tomar la subderivación que parte de la cláusula donde se encuentra el literal sobre el que existe la primera refutación subsidiaria para realizar la demostración y, posteriormente, tomar la subderivación de entrada desde la raíz inicial hasta la cláusula donde se encuentra dicho literal y combinar ambos resultados. Por tanto, sea *L*θ el literal seleccionado en *C*θ. Por tanto *C*$_1$'= $\tilde{L}\,\theta\theta_1$. Además existe una refutación SL* desde *L*θ en *T* usando *Anc*θ ∪ {*L*θ} vía *Ch* de rango menor que *k* con sustitución computada θ_1. Por la hipótesis de inducción, existe una refutación SL* desde *L* en *T* usando *Anc* ∪ {*L*} vía *Ch* del mismo rango y misma secuencia de cláusulas padre lejano con sustitución computada ϕ_1 tal que $\theta\theta_1 = \phi_1\sigma$ para alguna sustitución σ. De esta forma existe una refutación no restringida SL* desde *C* en *T* usando *Anc* vía *Ch* con secuencia de cláusulas *C*$_0$=*C*, …, *C*$_n$=□, secuencia de cláusulas padre lejano *C*$_1$'= $\tilde{L}\,\phi_1\sigma$, …, *C*$_n$' y secuencia de unificadores $\phi_1\sigma$, θ_2, …, θ_n. Aplicando el Lema 4.1 se cumple que existe una refutación SL* desde *C* en *T* usando *Anc* vía *Ch* del mismo rango con sustitución computada τ_1, …, τ_n de forma que $\phi_1\sigma\theta_2, \ldots, \theta_n = \tau_1, \ldots, \tau_n\rho$ para una cierta sustitución ρ. Por tanto, $\theta\theta_1\theta_2, \ldots, \theta_n = \tau_1, \ldots, \tau_n\rho$ cumpliéndose el resultado. ■

Lema 4.3 (Completitud del procedimiento SL* para el caso base)

Sea T una teoría consistente formada por cláusulas base y sea C una cláusula base. Si $T \cup \{C\}$ es inconsistente entonces,

a) existe una refutación SL* desde C en $T \cup \{C\}$ usando $\varnothing$ vía *Ch* para una cierta elección de ancestros *Ch* completa.
b) existe una refutación SL* desde C en T usando $\{C\}$ vía *Ch* para una cierta elección de ancestros *Ch* completa.

Demostración:

La demostración de los dos resultados es idéntica ya que la cláusula C es base y por tanto no hay diferencia al tratarla como una cláusula más en la teoría o como un ancestro. La demostración procede por inducción sobre el número de literales en $T \cup \{C\}$. Si sólo existe un literal, entonces C es una cláusula unitaria que contiene el literal L. Ya que T es consistente, contiene una única cláusula unitaria $\tilde{L}$. Fácilmente, se puede comprobar que existe una refutación SL* desde C en T usando $\varnothing$ vía *Ch*. La forma de la elección de ancestros determinará si la refutación es de rango 1 ó 0, según L esté o no en *Ch*.

Se supone que el resultado se cumple cuando existen n-1 literales en $T \cup \{C\}$.

Caso 1:

C es una cláusula unitaria que contiene el literal L. Sea S la teoría obtenida a partir de T eliminando aquellas cláusulas que contienen a L y quitando el literal $\tilde{L}$ [26] del resto de cláusulas en donde aparezca. S es inconsistente, ya que si no $T \cup \{C\}$ no lo sería. Sea T' un subconjunto de S tal que T' es mínimamente inconsistente. T' debe contener una cláusula C' que se obtuvo a base de quitar el literal $\tilde{L}$, si no T' sería un subconjunto de T violando el hecho de que T es consistente. Ya que T' es mínimamente inconsistente, $T' - \{C'\}$ es consistente. Ya que el número de literales en T' es menor que n, por la hipótesis de inducción, existen una refutación SL* desde C' en $T' \cup \{C'\}$ usando $\varnothing$ vía *Ch* con secuencia de cláusulas $C_0' = C', \ldots, C_n' = \square$. A partir de esta refutación es fácil ver, que existe una refutación desde C en $T \cup \{C\}$ usando $\varnothing$ vía *Ch* con secuencia de cláusulas $C_0 = C, C_1 = C', \ldots, C_n, \ldots, C_{n+m} = \square$ utilizando como cláusulas padre lejano las mismas que en la anterior refutación con la salvedad de que estas cláusulas se toman de $T \cup \{C\}$, por lo que algunas de ellas (se supone que $m>0$) pueden contener el literal $\tilde{L}$. Cada una de las m veces que se seleccione en la refutación $\tilde{L}$ puede suceder dos casos: si $\tilde{L}$ pertenece a la elección de ancestros *Ch* se usará una refutación subsidiaria de rango 0 y un

[26] Cuando una cláusula contiene a un literal L, éste será de la forma A o $\neg A$ según el átomo A esté en la cabeza o en el cuerpo de la cláusula. De esta forma, al hablar de las cláusulas que contienen al literal complementario de L se está haciendo referencia a aquellas en las cuales aparece el átomo A en el cuerpo o en la cabeza, respectivamente.

solo paso que corresponde a resolver con C; si $\tilde{L}$ no pertenece a la elección de ancestros Ch se aplicará un paso de resolución de entrada en la propia refutación utilizando C como cláusula padre lejano. De esta forma se obtiene la refutación buscada.

Caso 2:

Sea C de la forma $A_1 \vee \ldots \vee A_g \leftarrow D_1 \wedge \ldots \wedge D_h$ ($g \geq 0$, $h \geq 0$ y $g+h>0$) y sea L el literal seleccionado. Sea C' la cláusulas que se obtiene quitando L de C. Ahora se define la teoría T' como aquella que se obtiene a partir de T eliminando aquellas cláusulas que contengan $\tilde{L}$ y quitando L del resto de cláusulas donde aparezca. Así definidas, $T' \cup \{C'\}$ es inconsistente, ya que si no $T \cup \{C\}$ también sería consistente. A continuación se demuestra que T' es consistente. Sea M un modelo de T. Ya que C es falso en M, L es también falso M. Utilizando la definición de T' es sencillo probar que M es también un modelo de T', por lo que T' es consistente. Ya que $T' \cup \{C'\}$ contiene n-1 literales, aplicando la hipótesis de inducción, existen una refutación SL* desde C' en $T \cup \{C'\}$ usando $\varnothing$ vía Ch que se denotará por d_1'. Ahora si se toma la refutación d_1' y se vuelve a poner L en las cláusulas padre lejano en las que se quitó, se obtiene una derivación parcial formada por una derivación para cada refutación (la refutación de rango superior y todas las subsidiarias) de la misma longitud en las cuales aparece el literal L en las cláusulas de la secuencia. Por otra parte ya que $T \cup \{L\}$ es inconsistente, aplicando el resultado del caso 1, ha de existir una refutación SL* desde L en $T \cup \{L\}$ usando $\varnothing$ vía Ch, que se denotará por d_2. En este punto se pueden dar dos alternativas: que L pertenezca a la elección de ancestros Ch, ya que la refutación d_2 es equivalente a una refutación SL* desde L en T usando $\{L\}$ vía Ch, es sencillo demostrar a partir de la derivación parcial obtenida de d_1' y la refutación d_2 que existe una refutación desde C en $T \cup \{C\}$ usando $\varnothing$ vías Ch en la cual se usan como refutaciones subsidiarias la refutación d_2 tantas veces como el literal L es seleccionado. Por el contrario, L no pertenezca a la elección de ancestros Ch de manera que no sea posible usar la refutación d_2 subsidiariamente en la refutación que se está buscando. En estas circunstancias cabe distinguir dos casos:

> *Caso* 2.1: L no es utilizada como cláusula padre lejano en la refutación d_2 desde L en $T \cup \{L\}$ usando $\varnothing$ vía Ch. Así, es sencillo construir una refutación SL* desde C en $T \cup \{C\}$ usando $\varnothing$ vía Ch utilizando la derivación obtenida a partir de d_1' y la refutación d_2, pero en este caso, la refutación d_2 utilizada en el mismo rango que la refutación que se construye y no como subsidiaria.
>
> *Caso* 2.2: L es utilizada como cláusula padre lejano en la refutación d_2 desde L en $T \cup \{L\}$ usando $\varnothing$ vía Ch. Ahora el problema reside en que la refutación d_2 no se puede combinar con la derivación parcial obtenida a partir de d_1' ya que los pasos en los cuales L era utilizada en d_2 como cláusula padre lejano no pueden reproducirse al no estar L ni en el conjunto de cláusula ni en el conjunto de ancestros. Ahora bien, el hecho de que en la refutación d_2 se utiliza L como cláusula padre lejano implica que existe un paso en dicha

refutación en la cual la cláusula en curso contiene el literal $\tilde{L}$. Como *Ch* es una elección de ancestros completa $\tilde{L}$ ha de pertenecer a *Ch*. Por tanto, cuando se selecciona $\tilde{L}$ una refutación de rango inferior es llamada, que en particular sólo tendrá un paso correspondiente a la resolución con la cláusula padre lejano *L*. Combinando la derivación parcial obtenida a partir de d_1' y d_2, al llegar al paso correspondiente al anteriormente mencionado, se llamará a una derivación SL* subsidiaria desde $\tilde{L}$ en *T* usando *Anc* ∪ { $\tilde{L}$ } vía *Ch*, siendo *Anc* el conjunto de ancestros para la refutación de rango superior. En el primer paso de esta refutación subsidiaria es posible resolver con *C* obteniendo como siguiente cláusula en la derivación *C'*. Utilizando la refutación d_1', refutación SL* desde *C'* en *T'* usando ∅ vía *Ch*, así como el hecho de que $\tilde{L}$ es un ancestro por lo que puede ser usado como cláusula padre lejano, es sencillo demostrar que existe una refutación SL* desde $\tilde{L}$ en *T* usando *Anc* ∪ { $\tilde{L}$ } vía *Ch*, y por tanto que existe una refutación desde *C* en *T* usando ∅ vía *Ch*. ■

Teorema 4.2 (Corrección y completitud de SLP para la obtención de respuestas definidas)

Sea *T* una teoría y *C* una cláusula. El procedimiento SLP es correcto y completo para la obtención de respuestas definidas, como se expresa en los puntos a) y b).

a) Si existe una refutación SLP desde *C* en *T* con respuesta computada θ, entonces θ es una respuesta definida para *C* en *T*.

b) Si *T* es consistente, θ es una respuesta definida para *C* en *T* y *Ch* una elección de ancestros completa entonces, existe una refutación SLP desde C en *T* vía *Ch* con respuesta computada ϕ y una sustitución σ tal que θ=ϕσ.

Demostración:

El resultado a) de corrección se demuestra directamente como corolario del Teorema 4.1, ya que una refutación SLP desde *C* en *T* con respuesta computada θ es una refutación SL* desde *C* en *T* usando {*C*} vía una cierta elección de ancestros *Ch* con sustitución computada θ' de forma que θ es la restricción de θ' a las variables de *C*. Por tanto aplicando este teorema y suponiendo que *C* es de la forma $H \leftarrow B$, se cumple que $T \cup \{C\theta'\} \models \forall((\neg H \wedge B)\theta')$. Ya que θ es la restricción de θ' a las variables de *C*, se cumple $T \cup \{C\theta\} \models \forall((\neg H \wedge B)\theta)$. Ahora, directamente se puede demostrar que $T \models \forall((\neg H \wedge B)\theta)$.

El resultado b) de completitud se demuestra como sigue: ya que *T* es consistente y θ es una respuesta definida para *C* en *T*, entonces $T \models \forall((\neg H \wedge B)\theta)$. Supóngase que $x_1, \ldots, x_i$ son las variables de $(\neg H \wedge B)\theta$. Sean $a_1, \ldots, a_i$ constantes diferentes que no aparecen ni en *T* ni $(\neg H \wedge B)\theta$. Sea φ la sustitución $\{ x_1/a_1, \ldots, x_i/a_i\}$. Entonces se cumple que $T \models (\neg H \wedge B)\theta\varphi$. Ya que $T \cup \{H \leftarrow B)\theta\varphi$ es inconsistente, por el teorema de Herbrand, existe un conjunto finito *T'* de instancias base de cláusulas de *T* tal que $T' \cup \{H \leftarrow B)\theta\varphi$ es inconsistente. Aplicando el Lema

4.3, existe una refutación SL* desde $(H\leftarrow B)\theta\varphi$ en T' usando $\{H\leftarrow B)\theta\varphi$ vía *Ch*. Tomando esta refutación es sencillo construir una refutación no restringida SL* desde $(H\leftarrow B)\theta\varphi$ en T usando $\{H\leftarrow B)\theta\varphi$ vía *Ch* con secuencia de unificadores $\phi_1\ldots\phi_n$. Aplicando el Lema 4.1, existe una refutación SL* desde $(H\leftarrow B)\theta\varphi$ en T usando $\{H\leftarrow B)\theta\varphi$ vía *Ch*. Ya que a_j no aparece en ni T ni en $(\neg H\wedge B)\theta$ por reemplazamiento de a_j por x_j, $1\leq j\leq i$, en esta refutación y teniendo en cuenta que $(H\leftarrow B)\theta\varphi$ es base se obtiene una refutación SL* desde SL* desde $(H\leftarrow B)\theta$ en T usando $\{(H\leftarrow B)\theta\}$ vía *Ch* con sustitución computada $\phi_1\ldots\phi_n$ que no afecta a las variables de $(H\leftarrow B)\theta$. Así $(H\leftarrow B)\theta\phi_1\ldots\phi_n= (H\leftarrow B)\theta$. Aplicando ahora el Lema 4.2, existe una refutación SL* desde $H\leftarrow B$ en T usando $\{H\leftarrow B\}$ vía *Ch* del mismo rango y longitud con respuesta computada $\phi_1\ldots\phi_n$ tal que $\theta\theta_1\ldots\theta_n= \phi_1\ldots\phi_n\sigma'$ para alguna sustitución σ'. Sea ϕ la sustitución $\phi_1\ldots\phi_n$ restringida a las variables de $H\leftarrow B$. Entonces $\theta=\phi\sigma$, donde σ es una restricción apropiada de σ'. Teniendo en cuenta que la refutación SL* desde $H\leftarrow B$ en T usando $\{H\leftarrow B\}$ vía *Ch* es una refutación SLP desde $H\leftarrow B$ en T vía *Ch* y que ϕ es la respuesta computada de esta refutación, el resultado se cumple. ■

Teorema 4.3 (Corrección y completitud de SLT para la comprobación de la inconsistencia)

Sea T una teoría y C una cláusula. El procedimiento SLT es correcto y completo para la comprobación de la inconsistencia como se expresa en los puntos a) y b), respectivamente

a) Si existe una refutación SLT desde C en T, entonces $T \cup \{C\}$ es inconsistente.

b) Si T es consistente, $T \cup \{C\}$ es inconsistente y *Ch* es una elección de ancestros completa, entonces existe una refutación SLT desde C en T vía *Ch*.

Demostración:

El resultado a) de corrección se demuestra directamente como corolario del Teorema 4.1, ya que una refutación SLT desde C en T es una refutación SL* desde C en $T \cup \{C\}$ usando $\varnothing$ vía una cierta elección de ancestros *Ch*.

El resultado b) de completitud se demuestra como sigue: ya que $T \cup \{C\}$ es inconsistente por el teorema de Herbrand existe un conjunto finito de T' instancias base de cláusulas de T y un conjunto de instancias base $C_1, \ldots, C_n$ de C de forma que $T' \cup \{C_1,\ldots, C_n\}$ es inconsistente, y además como T' es consistente se puede demostrar que $T' \cup \{C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n\}$ es consistente para $1\leq i\leq n$. Supóngase que existiera un C_i para el cual no se cumple, entonces el conjunto de instancias que se tomarían sería $\{C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n\}$. Claramente como T' es consistente y $T \cup \{C\}$ es inconsistente al menos ha de existir una instancia base de C que sea inconsistente con T'. De esta forma, aplicando el Lema 4.3, existe una refutación SL* desde C_i en $T' \cup \{ C_1, \ldots, C_n\}$ usando $\varnothing$ vía *Ch*. A partir de esta refutación es posible construir una refutación no restringida SL* desde C_i en $T \cup \{C\}$ usando $\varnothing$ vía *Ch*. Por tanto aplicando el Lema 4.1, existe una refutación SL* desde C_i en $T \cup \{C\}$ usando $\varnothing$ vía *Ch*. Claramente como

C_i es igual $C\theta$ y aplicando el Lema 4.2, existe una refutación SL* desde C en $T \cup \{C\}$ usando $\varnothing$ vía Ch. Teniendo en cuenta que la refutación SL* desde C en $T \cup \{C\}$ usando $\varnothing$ vía Ch es una refutación SLT desde C en T vía Ch, el resultado se cumple. ■

Antes de demostrar la validez de los resultados del procedimiento SLT para la obtención de respuestas indefinidas es necesario presentar un resultado intermedio.

Lema 4.4

Sea T una teoría, C una cláusula, Anc un conjunto de cláusulas y Ch una elección de ancestros de forma que existe una refutación SL* desde C en T usando Anc vía Ch con sustitución computada θ. Sea PL el conjunto de cláusulas padre lejano usadas en la refutación incluyendo las refutaciones subsidiarias tal que pertenezcan a $T \cup Anc$. Si S_1 y S_2 son dos conjuntos de cláusulas de forma S_1 y S_2 no son vacíos, $S_1 \cap S_2=\varnothing$ y $S_1 \cup S_2= PL\cup\{C\}$, entonces $S_1\theta \models \bigvee_i \neg H_i\theta \wedge B_i\theta$ de forma que $H_i \leftarrow B_i$ pertenece a S_2.

Demostración:

La demostración se realiza por doble inducción sobre la longitud n y el rango k de la misma.

Se probará primero para rango k=0. Supóngase que la longitud de la refutación es n=1. Esto significa que C es de la forma $A_1\leftarrow$ (resp. $\leftarrow D_1$), y existe una cláusula perteneciente a la teoría T o al conjunto de ancestros Anc de la forma $A\leftarrow$ (resp. $D\leftarrow$) y $A_1\theta=A\theta$ (resp. $D_1\theta=D\theta$). Claramente el resultado se cumple para cualquier par de conjuntos S_1, S_2.

Ahora se supondrá que el resultado se cumple para refutaciones de longitud n-1. Sea $C_0=C$, C_1,…, $C_n=\square$ la secuencia de cláusulas, C_1',…, C_n' la secuencia de cláusulas padre lejano pertenecientes a $T \cup Anc$ y θ_1,…, θ_n la secuencia de umg de la refutación SL* de longitud n. Ya que existe una refutación SL* desde C_1 en T usando Anc vía Ch con sustitución computada $\theta_2\ldots\theta_n$ y un conjunto de cláusulas padre lejano $\{C_2',\ldots, C_n'\}$ entonces por la hipótesis de inducción para cualquier par de conjuntos S_1' y S_2' formados por la cláusula inicial C_1 y las cláusulas padre lejano de manera que no sean vacíos, sean disjuntos y que tanto C_1 como toda cláusula padre lejano C_i', $2\leq i£n$, pertenece a S_1' o S_2' es cierto que $S_1'\theta_2\ldots\theta_n \models \bigvee_i \neg H_i\theta_2\ldots\theta_n \wedge B_i\theta_2\ldots\theta_n$ de forma que $H_i\leftarrow B_i$ pertenece a S_2'. Ya que C_1 se obtenido por un paso de resolución de entrada a partir de C con cláusula padre lejano C_1' que pertenece a $T\cup Anc$ usando el umg θ_1 se cumple que $\{C\theta_1, C_1'\theta_1\} \models \neg H_1\theta_1 \wedge B_1\theta_1$ donde $C_1=H_1\leftarrow B_1$. A partir de este resultado se sencillo comprobar si se toma S_1 y S_2 a partir de S_1' y S_2' a los que se les añade $C\theta_1\theta_2\ldots\theta_n$, $C_1'\theta_1\theta_2\ldots\theta_n$ adecuadamente se cumple el resultado.

Se supone ahora que el resultado se cumple para refutaciones de rango k-1. Supóngase que el rango de la refutación es k y la longitud de la refutación es n. La demostración se realiza

por inducción sobre el número *s* de refutaciones subsidiarias que se lanzan desde la refutación. Para *s*=1, se supondrá que la refutación subsidiaria se llama en el primer paso de la refutación. De no ser así, basta con tomar la subderivación que parte de la cláusula donde se encuentra el literal sobre el que existe la primera refutación subsidiaria para realizar la demostración y, posteriormente, tomar la subderivación de entrada desde la raíz inicial hasta la cláusula donde se encuentra dicho literal y combinar ambos resultados. Por tanto, sea L_0 el literal seleccionado en C_0. Así, existe una refutación SL* subsidiaria desde L_0 en T usando $Anc \cup \{L_0\}$ de rango menor que k con sustitución computada θ_1 y conjunto de cláusulas padre lejano $E_1,\ldots, E_m$ pertenecientes a $T \cup Anc \cup \{L_0\}$. Por la hipótesis de inducción se cumple que para cualquier par de conjuntos S_1' y S_2' formados por la cláusula inicial L_0 y las cláusulas padre lejano de manera que no sean vacíos, sean disjuntos y que tanto L_0 como toda cláusula padre lejano E_i, $1 \leq i \leq m$, pertenece a S_1' o S_2', es cierto que $S_1'\theta_1 \models \bigvee_i \neg H_i\theta_1 \wedge B_i\theta_1$ de forma que $H_i \leftarrow B_i$ pertenece a S_2'. Por otra parte, existe una refutación SL* desde C_1, siguiente cláusula en la refutación, en T usando Anc vía Ch de rango 0 con sustitución computada $\theta_2 \ldots \theta_n$ y secuencia de cláusula padre lejano $C_2', \ldots, C_n'$. Por los resultados anteriores se cumple que para cualquier par de conjuntos S_3' y S_4' formados por la cláusula inicial C_1 y las cláusulas padre lejano de manera que no sean vacíos, sean disjuntos y que tanto C_1 como toda cláusula padre lejano C_i', $2 \leq i \leq n$, pertenece a S_3' o S_4' es cierto que $S_3'\theta_2 \ldots \theta_n \models \bigvee_i \neg H_i\theta_2 \ldots \theta_n \wedge B_i\theta_2 \ldots \theta_n$ tal que cada $H_i \leftarrow B_i$ pertenece a S_4'. Ya que C_1 se obtenido por un paso de resolución de entrada a partir de C con cláusula padre lejano C_1' usando el umg θ_1 se cumple que $\{C\theta_1, C_1'\theta_1\} \models \neg H_1\theta_1 \wedge B_1\theta_1$ donde $C_1 = H_1 \leftarrow B_1$. A partir de este resultado se sencillo comprobar que si se toma S_1 y S_2 a partir de los conjuntos S_1', S_2', S_3' y S_4' añadiéndoles $C\theta_1\theta_2 \ldots \theta_n$, $C_1'\theta_1\theta_2 \ldots \theta_n$ adecuadamente se cumple el resultado.

Ahora se supondrá que el resultado se cumple para refutaciones SL* de rango *k* que usan *s*-1 refutaciones subsidiarias. Por los motivos anteriormente descritos, también se supondrá que la refutación de rango *k* que usa *s* refutaciones subsidiarias, usa una de ellas en el primer paso de la refutación. Sea L_0 el literal seleccionado en C_0. Así, existe una refutación SL* subsidiaria desde L_0 en T usando $Anc \cup \{L_0\}$ de rango menor que k con sustitución computada θ_1 y conjunto de cláusulas padre lejano $\{E_1,\ldots, E_m\}$ pertenecientes a $T \cup Anc \cup \{L_0\}$. Por tanto por los resultados anteriores se cumple que para cualquier par de conjuntos S_1' y S_2' formados por la cláusula inicial L_0 y las cláusulas padre lejano de manera de manera que no sean vacíos, sean disjuntos y que tanto L_0 como toda cláusula padre lejano E_i, $1 \leq i \leq m$, pertenece a S_1' o S_2', es cierto que $S_1'\theta_1 \models \bigvee_i \neg H_i\theta_1 \wedge B_i\theta_1$ tal que cada $H_i \leftarrow B_i$ pertenece a S_2'. Por otra parte, existe una refutación SL* desde C_1, siguiente cláusula en la refutación, en T usando Anc vía Ch con sustitución computada $\theta_2 \ldots \theta_n$ y secuencia de cláusulas padre lejano $C_2', \ldots, C_n'$ pertenecientes a $T \cup Anc$ de rango igual o inferior a k que usa s-1 refutaciones subsidiarias. Por la hipótesis de inducción, se cumple que para cualquier par de conjuntos S_3' y S_4' formados por la cláusula inicial C_1 y las cláusulas padre lejano de manera que no sean vacíos, sean disjuntos y que tanto

C_1 como toda cláusula padre lejano C_i', 2≤*i*£*n*, pertenece a S_3' o S_4' es cierto que $S_3'\theta_2\ldots\theta_n \models \bigvee_i \neg H_i\theta_2\ldots\theta_n \wedge B_i\theta_2\ldots\theta_n$ de forma que $H_i \leftarrow B_i$ pertenece a S_4'. Ya que C_1 se obtenido por un paso de resolución de entrada a partir de C con cláusula padre lejano C_1' usando el umg θ_1 se cumple que $\{C\theta_1, C_1'\theta_1\} \models \neg H_1\theta_1 \wedge B_1\theta_1$ donde $C_1 = H_1 \leftarrow B_1$. A partir de este resultado se sencillo comprobar si se toma S_1 y S_2 a partir de los conjuntos S_1', S_2', S_3' y S_4' añadiéndoles $C\theta_1\theta_2\ldots\theta_n$, $C_1'\theta_1\theta_2\ldots\theta_n$ adecuadamente se cumple el resultado. ■

Teorema 4.4 (Corrección y completitud de SLT en la obtención de respuestas)

Sea T una teoría y C una cláusula. El procedimiento SLT es correcto y completo en la obtención de respuestas, como se expresa en los puntos a) y b), respectivamente.

a) Si existe una refutación SLT desde C en T con respuesta computada Θ, entonces Θ es una respuesta de C en T.

b) Si T es consistente, Θ es una respuesta para C en T y *Ch* una elección de ancestros completa entonces, existe una refutación SLT desde C en T vía *Ch* con respuesta computada Φ tal que, para cada ϕ en Φ, existe una sustitución θ en Θ y una sustitución σ tal que θ=ϕσ.

Demostración:

La demostración del punto a) es como sigue: sea C_1', …, C_n' la secuencia de cláusulas padre lejano usadas en la refutación SLT desde C en T incluyendo las refutaciones subsidiarias y que pertenezcan a $T \cup \{C\}$. Ya que una refutación SLT desde C en T es una refutación SL* desde C en $T \cup \{C\}$ usando ∅, por el Lema 4.4 se puede formar dos conjuntos S_1 y S_2 de manera que toda cláusula incluida en S_2 es o C o una variante de C y el resto de cláusula padre lejano se encuentran en S_1, tal que $S_1\theta \models \bigvee_i \neg H_i\theta \wedge B_i\theta$ de forma que $H_i \leftarrow B_i$ pertenece a S_2. Claramente como las cláusula de S_1 son instancias de cláusulas de T se cumple que $T \models \bigvee_i \neg H_i\theta \wedge B_i\theta$ de forma que $H_i \leftarrow B_i$ pertenece a S_2. Por otra parte la respuesta computada Θ es igual a $\{\theta_0, \rho_1\theta_1, \ldots, \rho_m\theta_m\}$ donde $\rho_1, \ldots, \rho_m$ ($m \geq 0$) son los renombramientos de las variables de C en las variantes de C que se han usado. Además, θ_0 es la restricción de θ a las de C y θ_i es la restricción de θ a las variables de la variante de C usada. Aplicando cada una los elementos de Θ a bien la cláusula inicial C bien a la variante de C correspondiente se obtiene que $T \models \forall(\neg C\theta_0 \vee \neg C\theta_1 \ldots \vee \neg C\theta_m)$.

La demostración del punto b) se realiza por inducción sobre el número de elementos que componen la respuesta Θ. Sea Θ de la forma $\{\theta_1, \ldots, \theta_j\}$ ($j>0$). El caso que $j=1$ se demuestra directamente ya que éste corresponde a obtener una respuesta definida para el problema. Ya que SLP es completo para la obtención de estas respuestas y toda refutación SLP es una refutación SLT el resultado se cumple.

Supóngase que el resultado se cumple para respuestas de j-1 elementos. Ahora si la respuesta Θ no es concisa quiere decir que existe un Θ', subconjunto propio de Θ, que es una respuesta de C en T. Ya que Θ' contiene como mucho j-1 elementos, se aplica la hipótesis de inducción y el resultado se cumple. Si por el contrario la respuesta Θ es concisa se puede probar que $T \cup \{C\theta_1,\ldots, C\theta_{i-1}, C\theta_{i+1},\ldots, C\theta_j\} \models \forall(\neg C\theta_i)$ $(1 \leq i \leq j)$ de forma que $T \cup \{C\theta_1,\ldots, C\theta_{i-1}, C\theta_{i+1},\ldots, C\theta_j\}$ es consistente. Por tanto, y ya que SLP es completo para la obtención de respuesta, se demuestra que existe una refutación SLP desde C en $T \cup \{C\theta_1,\ldots, C\theta_{i-1}, C\theta_{i+1},\ldots, C\theta_j\}$ con respuesta θ y una sustitución σ tal que $\theta_i=\theta\sigma$. Esta refutación se puede ver que es equivalente a una refutación no restringida SL* desde C en $T \cup \{C\}$ con la utilización de los unificadores correspondientes. Aplicando ahora el Lema 4.1 y Lema 4.2 se puede obtener el resultado buscado. ■

Teorema 4.5 (Poda por igualdad con ancestro y elección de ancestros)

Sea T una teoría consistente formada por cláusulas base y C una cláusula base. Entonces los siguientes dos puntos se cumplen:

a) SLT vía ALL es un procedimiento de decisión de la inconsistencia de $T \cup \{C\}$.
b) SLP vía ALL es un procedimiento de decisión de $T \models \neg C$.

Demostración:

Ya que C es una cláusula base los dos procedimientos coinciden, por tanto bastará demostrarlo para uno de los dos (se demostrará el resultado a)). Por otro lado, la corrección y completitud están demostrados por tratarse de un caso particular del Teorema 4.3. Así, sólo será necesario demostrar que en el caso de que $T \cup \{C\}$ sea consistente no existe una derivación infinita. Supongamos que existe dicha derivación. Como el conjunto de literales en $T \cup \{C\}$ es finito y la elección de ancestros es ALL, en algún cierto punto debe de cumplirse que el literal seleccionado en la cláusula en curso ya pertenezca al conjunto de ancestros. Por la propia definición esta circunstancia está prohibida, por tanto el resultado se cumple. ■

Teorema 4.6 (Completitud de los procedimientos SLT y SLP vía elección de POS_BASE para teorías de rango restringido)

Sea T una teoría consistente, C una cláusula tal que $T \cup \{C\}$ es de rango restringido y sea POS_BASE la elección de ancestros que contiene todos los literales positivos base del lenguaje subyacente, entonces los siguientes puntos se cumplen:

a) si θ es una respuesta definida para C en T, entonces existe una refutación SLP justa desde C en T vía POS_BASE con respuesta computada θ.
b) $T \cup \{C\}$ es inconsistente, entonces existe una refutación SLT justa desde C en T vía POS_BASE.

Demostración:

Supóngase que la función de selección es de la siguiente forma, se selecciona primero en el cuerpo hasta que la cláusula en curso sea positiva y luego en la cabeza de forma justa, de manera que en algún momento todo literal es seleccionado.

Inicialmente es necesario probar que los lemas 4.1 y 4.2 se cumplen para las condiciones del teorema. Fácilmente, se puede probar esto procediendo de igual forma y teniendo en cuenta cuál es la función de selección empleada y la elección de ancestros utilizada.

El resultado a) se demuestra de igual forma que el punto b) del Teorema 4.2 procediendo de igual forma y aplicando los resultados intermedios antes mencionados.

De igual forma se demuestra el resultado b), siguiendo la demostración del punto b) del Teorema 4.3 y aplicando los resultados intermedios de citados en el párrafo anterior. ■

Teorema 4.7 (Completitud del SLP y SLT vía elección de ancestros óptima para la resolución de ancestro)

Sea *T* una teoría, *C* una cláusula y *Ch* la elección de ancestros óptima para la resolución de ancestro de *C* en *T* . Los siguientes puntos aseguran la completitud de los procedimientos SLP y SLT vía *Ch*.

a) Si *T* es consistente y θ es una respuesta definida para *C* en *T*, entonces existe una refutación SLP desde *C* en *T* vía *Ch* con respuesta computada ϕ y una sustitución σ tal que $\theta=\phi\sigma$.

b) Si *T* es consistente y $T \cup \{C\}$ es inconsistente, entonces existe una refutación SLT desde *C* en *T* vía *Ch*.

c) Si *T* es consistente y Θ es una respuesta para *C* en *T*, entonces existe una refutación SLT desde *C* en *T* vía *Ch* con respuesta computada Φ tal que, para cada ϕ en Φ, existe una sustitución θ en Θ y una sustitución σ tal que $\theta=\phi\sigma$.

Demostración:

Las demostraciones de los tres puntos a), b) y c) proceden de igual forma que las de los puntos b) del Teorema 4.2, la del punto b) de Teorema 4.3 y la del punto b) del Teorema 4.4, respectivamente. La diferencia reside en el hecho de que la elección de ancestros no es completa, aunque sí es homogénea. Por tanto los Lemas 4.1 y 4.2 se siguen cumpliendo y Lema 4.3 hay que revisarlo. La utilización del hecho de ser la elección de ancestros completa en dicho lema se realiza en el caso 2 de la demostración. En dicho caso se plantean dos alternativas una que el literal seleccionado esté en la elección de ancestros y otra que no esté. Si se da la primera alternativa la demostración se realiza directamente y si se da la segunda se plantean dos

posibilidades: que el literal se utilice como cláusula padre lejano en la refutación subsidiaria que parte de él o que no. Caso de que se cumpla la segunda posibilidad la demostración tampoco supone problema y se realiza normalmente. Es en la primera posibilidad donde se hace uso del hecho de que la elección de ancestros sea completa. Esto es, cuando el literal seleccionado no está en la elección de ancestros y éste es usado como cláusula padre lejano en la refutación subsidiaria que parte de él. Simplemente basta con demostrar que esta posibilidad no se puede dar si la elección de ancestros es óptima, ya que bajo estas condiciones existe una factorización sobre este literal usando las cláusula padre lejano de la refutación subsidiaria y teniendo en cuenta la definición de elección de ancestros óptima el literal debe pertenecer a la elección de ancestros. Por lo que los resultados se cumplen. ■

Proposición 4.1

Sea *T* una teoría y *C* una cláusula. Si *OP* es la elección de ancestros óptima para la resolución de ancestro de *C* en *T* y *SOP* es la elección de ancestros subóptima para la resolución de ancestro de *C* en *T*, entonces $OP \subseteq SOP$.

Demostración: Inmediata.

Proposición 4.2

Sea *T* una teoría y *C* una cláusula. Si *OP* es la elección de ancestros óptima para la poda por igualdad con ancestro de *C* en *T* y *SOP* es la elección de ancestros subóptima la poda por igualdad con ancestro de *C* en *T*, entonces $OP \subseteq SOP$.

Demostración: Inmediata.

APÉNDICE B: CÓDIGO DE LA IMPLEMENTACIÓN

En este apéndice se incluye en primer lugar un ejemplo de problema de la librería *TPTP* y el programa Prolog obtenido en el proceso de precompilación, y, en segundo lugar, el código del procedimiento SLT vía la elección de ancestros ALL.

A continuación se puede ver el problema *COL001-1* en su formato original.

```
%--------------------------------------------------------------------------
% File     : COL001=WkFxdPtSK-1 : TPTP v1.2.0. Released v1.0.0.
% Domain   : Combinatory Logic
% Problem  : Weak fixed point for S and K
% Version  : [Wos & McCune, 1988] (equality) axioms.
% English  : The weak fixed point property holds for the set P consisting
%            of the combinators S and K alone, where ((Sx)y)z = (xz)(yz)
%            and (Kx)y = x.

% Refs     : Smullyan, R. M. (1985), To Mock a Mocking Bird and Other
%            Logic Puzzles, Knopf, New York.
%          : Wos L., McCune W.W (1988), Challenge Problems Focusing
%            on Equality and Combinatory Logic: Evaluating Automated
%            Theorem-Proving Programs, Proceedings of the 9th
%            International Conference on Automated Deduction (Argonne, IL,
%            1988), (Lecture Notes in Computer Science, 310), 714-729.
% Source   : [Wos & McCune, 1988]
% Names    : C1 [Wos & McCune, 1988]

% Status   : unsatisfiable
% Syntax   : Number of clauses           :    8 (   0 non-Horn)(   4 unit)
%            Number of literals          :   13 (  13 equality)
%            Maximal clause size         :    3
%            Number of predicate symbols :    1 (   0 propositional)
%            Number of function symbols  :    4 (   3 constant)
%            Number of variables         :   18 (   1 singleton)
%            Maximal term depth          :    4

% Comments :
%          : tptp2X: -ftptp COL001-1.p
%--------------------------------------------------------------------------
input_clause(reflexivity,axiom,
    [++equal(A,A)]).

input_clause(symmetry,axiom,
    [--equal(A,B),
     ++equal(B,A)]).

input_clause(transitivity,axiom,
    [--equal(A,B),
     --equal(B,C),
     ++equal(A,C)]).
```

```
input_clause(s_definition,axiom,
    [++equal(apply(apply(apply(s,A),B),C),apply(apply(A,C),apply(B,C)))]).

input_clause(k_definition,axiom,
    [++equal(apply(apply(k,A),B),A)]).

input_clause(apply_substitution1,axiom,
    [--equal(A,B),
     ++equal(apply(A,C),apply(B,C))]).

input_clause(apply_substitution2,axiom,
    [--equal(A,B),
     ++equal(apply(C,A),apply(C,B))]).

input_clause(prove_fixed_point,theorem,
    [--equal(A,apply(combinator,A))]).
%----------------------------------------------------------------------
```

El programa Prolog obtenido al aplicar la precompilación para adecuar el problema al formato elegido así como para aplicar las reducciones comentadas en el capítulo 4 se puede ver a continuación.

```
/*  PROBLEMA: COL001-1.slt  */

/*  CONTRAPOSITIVAS DEL PROBLEMA  */

/*  reflexivity - axiom */
equal(equal(B,A), [], 1, [p(A,B)]).


/*  symmetry - axiom */
n_equal(n_equal(A,B), [n_equal(B,A)], 2, []).
equal(equal(B,A), [equal(A,B)], 2, []).


/*  transitivity - axiom */
n_equal(n_equal(A,B), [equal(B,C),n_equal(A,C)], 3, []).
n_equal(n_equal(B,C), [equal(A,B),n_equal(A,C)], 3, []).
equal(equal(A,C), [equal(A,B),equal(B,C)], 3, []).


/*  s_definition - axiom */
equal(equal(apply(apply(apply(s,D),E),F),apply(apply(C,G),apply(B,A))), [], 1,
[p(A,G),p(A,F),p(B,E),p(C,D)]).


/*  k_definition - axiom */
equal(equal(apply(apply(k,B),_),A), [], 1, [p(A,B)]).


/*  apply_substitution1 - axiom */
n_equal(n_equal(A,B), [n_equal(apply(A,C),apply(B,C))], 2, []).
equal(equal(apply(A,D),apply(B,C)), [equal(A,B)], 2, [p(C,D)]).
```

```
/*  apply_substitution2 - axiom */
n_equal(n_equal(A,B), [n_equal(apply(C,A),apply(C,B))], 2, []).
equal(equal(apply(D,A),apply(C,B)), [equal(A,B)], 2, [p(C,D)]).


/*  prove_fixed_point - theorem */
n_equal(n_equal(B,apply(combinator,A)), [], 1, [p(A,B)]).
fal(fal, [equal(A,apply(combinator,A))], 2, []).


/*  TABLA  DE  NEGACION  */
negar(equal(A,B), n_equal(A,B)).
negar(n_equal(A,B), equal(A,B)).
negar(fal, n_fal).
negar(n_fal, fal).


/*  CLAUSULA INICIAL  */
n_fal(n_fal, [], 1, []).
cinicial([fal], 1).

/*  FIN DEL PROBLEMA: COL001-1.slt  */

/*  Tiempo precompilacion: 30 msegs.  */
```

Por último se incluye el código del programa Prolog correspondiente al procedimiento SLT vía la elección de ancestros ALL.

```
/* ------------------------------------------------------ */
/*         PROCEDIMIENTO SLT. vs 29 b 19-06-96            */
/* ------------------------------------------------------ */

/* Indexacion del Prolog                                 */
/* Profundidad: Pasos de inferencia                       */
/* Seleccion: mas instanciado                             */
/* Corte Ground                                           */
/* Corte por no instanciacion                             */
/* Sicstus 3.0                                            */

/* Clausula: pr(cabeza, lista_cuerpo,num_lit,lis_unf)    */
/* Donde:
            cabeza = literal seleccionado
            cuerpo = lista literales restantes
            num_lit= numero total literales
            lis_unf= lista de parejas de variables a
                   ser unificadas correctamente    */

/* De cada clausula existen todas sus contrapositivas en */
/* formato implicativo.                                   */

/*     Resolvente de un literal y un programa:           */

resolvente(Lit,Res,NumL) :- comp(X,Lit,Res,N,LU),
                     call(X),
                     checkvar(LU),
                            NumL is N-1.
```

```
/*     Demostracion de un literal, dado el conjunto      */
/*     de ancestros Anc.                                 */

/* Caso ground. */

dem(L,Anc,PI,PO) :- ground(L),!,demg(L,Anc,PI,PO),!.

demg(L,Anc,PI,PO) :- negar(L,NoL),
                pertenece(NoL,Anc),
                PO is PI - 1,!.

demg(L,Anc,PI,PO):- resolvente(L,Res,NumL), PI1 is PI - 1,
                     demo(Res,NumL,Anc,PI1,PO),!.

/* Caso no ground. */

dem(L,Anc,PI,PO) :- negar(L,NoL),
                   (copy_term(NoL,T),
                    pertenece(NoL,Anc),
                   ((variant(NoL,T),!);true)),
               PO is PI - 1.

dem(L,Anc,PI,PO) :- copy_term(L,T),
                   resolvente(L,Res,NumL),PI1 is PI-1,
                   demo(Res,NumL,Anc,PI1,PO),
                   ((variant(L,T),!);true).


/*     Demostracion de una clausula S, en  P U S,        */
/*     con NumL literales, considerando                  */
/*     el conjunto de ancestros Anc,                     */
/*     Spliting-Rule.                                    */

/*         P = Profundidad de la demostracion.           */

demo([],_,_,P,P):-!.

demo(Claus,NumL,Anc,PI,PO) :- PI > 0, corte2(PI,NumL),
                     mas_inst2(Lit,Claus,Claus2),
                     \+(corte0(Lit,Anc)),
                     anadir(Lit,Anc,Anc2),
                P1 is PI-NumL+1,
                     dem(Lit,Anc2,P1,PO1),
                P3 is PO1+NumL-1, NumL2 is NumL-1,
                     demo(Claus2,NumL2,Anc,P3,PO).

/*     8.- Profundidad iterada.                          */
/*         P = Profundidad demostracion inicial.         */

pi(Prob,Claus,N,P,Ti) :-  (demo(Claus,N,[],P,_),
                   statistics(runtime,[Tn,_]),
                   T is Tn - Ti,
                   acaba(Prob,si,T,P));
                        (statistics(runtime,[Tn,_]),
                     T is Tn - Ti,
                     ((T > 1000000, acaba(Prob,no,T,P),!);
                          (P1 is P + 1, pi(Prob,Claus,N,P1,Ti)))).
```

```
acaba(Prob,F,T,P) :-format("~a~18|~t~d~22|~t~d~33|~t~a~37|~n",
                    [Prob,P,T,F]).

/*         Demostracion desde la clausula inicial,       */
/*         empezando con una profundidad  0.             */
/*         realizada en un tiempo (segs) T.              */

d(Problema) :- compile(Problema),cinicial(Claus,N),
                       statistics(runtime,[Ti,_]),
                       pi(Problema,Claus,N,N,Ti).

/* Condiciones de corte. Por subsumcion no es completa. */

corte0(Lit,Anc) :- per_igual(Lit,Anc).
corte1(Lit,Anc) :- pertenece(A,Anc), subsume(A,Lit).
corte2(P,C) :- P >= C.


/* Descomposicisn y composicion de una clausula */

desc(Cls,Cab,Cue,N,LU) :- Cls =..[_,Cab,Cue,N,LU].

comp(Cls,Cab,Cue,N,LU) :-functor(Cab,F,_),Cls=..[F,Cab,Cue,N,LU].

comp2(Cls,Cab,Cue,N,LU):-functor(Cab,F,Na),functor(Cab2,F,Na),
                  Cls =..[F,Cab2,Cue,N,LU].

/* Parejas de variables que deben unificar correctamente */
/* Depende del Prolog, Cuidado!! */

checkvar([]) :-!.
checkvar([p(X1,X2)|R]) :- occur(X1,X2), checkvar(R).

/*                                                       */
/*        Funciones de seleccion.                        */

/* Modificadas para demostradores slt21 ...  (4/9/95)     */
/* Se utilizan los predicados num_var, prim_rest y quitar.*/
/* Funciones de seleccion. */
/* Seleccion del literal mas a la izquierda.              */

mas_izq(Lit,Claus,Claus2) :- prim_rest(Lit,Claus,Claus2).

/*mas_inst2: Menos instanciada y menos costosa que mas_inst*/

mas_inst2(MasIn,Cl,Resto):-
          el_mas_ins(Cl,MasIn,Resto).

el_mas_ins([MasIn],MasIn,[]):-!.
el_mas_ins([Gr|Resto],Gr,Resto):- ground(Gr),!.
el_mas_ins([L1|C],MasIn,Resto):-
       el_mas_ins(C,L2,R),
       elegir(MasIn,Resto,L1,L2,C,R).

elegir(MasIn,Resto,MasIn,L2,Resto,_):-
       mas_ins(MasIn,L2),!.
```

```
elegir(MasIn,[L1|R],L1,MasIn,_,R).

mas_ins(A1,A2):-
       cuenta_var(A1,V1),
       cuenta_var(A2,V2),!,
       V1=<V2.

cuenta_var(A,V):-
       A=..[_|Larg],
       cuenta_no_base(Larg,V).

cuenta_no_base([],0).
cuenta_no_base([T|C],V):- ground(T),!, cuenta_no_base(C,V).
cuenta_no_base([_|C],V):- cuenta_no_base(C,V1), V is V1+1.

?-use_module(library(terms)).

/* TAD conjunto. */

/* Unificacion con Occur's Check, mediante el predicado occur */

/* Las operaciones que debe proporcionar son: */

/* Determinar si un conjunto es el conjunto vacio.   */
/* Test de pertenencia de un elemento a un conjunto. */
/* Extraer, borrando, un elemento de un conjunto.    */
/* A$adir un elemento a un conjunto.                 */


/* ------------------------------------------------ */
/* Para esta implementacion se utilizan listas.     */
/* ------------------------------------------------ */

/* Definicion de conjunto vacio:                    */

vacio([]).

/* Definicion de pertenencia: (unificando)          */
/* Es necesario el Occur's Check, puesto que X puede */
/* estar instanciado.                               */

pertenece(X,[X1|_]) :-occur(X,X1).
pertenece(X,[_|Z]) :- pertenece(X,Z).

/* Definicion de pertenencia: (igualdad)            */
/*                                                  */

per_igual(X,[X1|_]) :-X==X1,!.
per_igual(X,[_|Z]) :- per_igual(X,Z).

pertany(X,[H|_],B) :- ((X==H,B=true,!);(occur(X,H),B=false)).
pertany(X,[_|Z],B) :- pertany(X,Z,B).

/* Definicion de cardinal (card) de un conjunto.    */

card([],0) :-!.
card([_|B],C1) :- card(B,C), C1 is C+1.
```

```
/* Definicion de extraer:                              */
/* Elem puede estar instanciado,  hace falta la        */
/* unificacion con Occur's Check.                      */

extraer(Elem,[Elem1|Y],Y) :- occur(Elem,Elem1).
extraer(Elem,[Y|Z],[Y|W]) :- extraer(Elem,Z,W).

/* Definicion de quitar:                               */
/* Elem, est  siempre instanciado. Determinista.       */

quitar(Elem,[Elem1|Y],Y) :- Elem==Elem1.
quitar(Elem,[Y|Z],[Y|W]) :- quitar(Elem,Z,W).

/* Definicion de prim_rest:                            */
/* Es una versi"n determinista y de coste constante    */
/* de extraer, no hace falta el Occur's Check.         */

prim_rest(Elem,[Elem|Y],Y).

/* num_var, numero de variables en un termino.         */

num_var(T,1) :- var(T),!.
num_var(T,0) :- atomic(T),!.
num_var(T,N) :- T =..[_|B],num_vars(B,N).

num_vars([],0) :- !.
num_vars([H|B],N):-num_var(H,N1),num_vars(B,N2), N is N1+N2.

/* Definicion de a$adir un elemento a un conjunto.     */
/* Con duplicados. */
/* Coste constante, sin Occur's Check.                 */

anadir(Elem,Conj,[Elem|Conj]).

anadir1(Elem,Conj,[Elem|Conj]):-
      ancestro(Elem),!.
anadir1(Elem,Conj,Conj).

/* Definicion de  append:                              */

append([],L,L) :-!.
append([I|C],L,[I|L1]) :- append(C,L,L1).

/*                                                          */
/* Predicados para unificacion con occur's check y subsumcion */
/*                                                          */
/* En los programas aparecen como 'occur' y subsume         */
/* Version para sicstus-3, se utiliza la libreria 'term'.   */

?-use_module(library(terms)).

/* Unificacion */

occur(X,X1):-unify_with_occurs_check(X,X1).

/* Subsumcion, unificando el termino mas general */

subsume(X,X1):-subsumes(X,X1).
```

APÉNDICE C: PROBLEMAS ABORDADOS Y RESULTADOS EXPERIMENTALES

En este apéndice se incluye un conjunto de tablas que muestran la información sobre las características de los problemas resueltos, así como los resultados experimentales obtenidos con los procedimientos SLT vía las diferentes elecciones de ancestros utilizadas. La información sobre las características del resto de problemas abordados se puede encontrar en [Suttner y Sutcliffe, 1996]. A continuación se incluye una breve descripción de los dominios de los problemas:

- ALG (Álgebra): un álgebra es un conjunto con un sistema de operaciones definidos en ella. Número de problemas: 4.
- ANA (Análisis): el Análisis es una rama de las Matemáticas concerniente con funciones y límites. La mayor parte del análisis es cálculo diferencial e integral, y teoría de funciones. Número de problemas: 19.
- BOO (Álgebra booleana): un álgebra booleana es un conjunto de elementos con dos operaciones binarias que son idempotentes, conmutativas y asociativas. Las operaciones son mutuamente distributivas. Número de problemas: 52.
- CAT (Teoría de categorías): una categoría es una estructura matemática junto con un morfismo que preserva dicha estructura. Número de problemas: 59.
- CID (Diseño de circuitos): se utiliza para construir un circuito conociendo los patrones de entrada y los requerimientos de los patrones de salida. Número de problemas: 4.
- CIV (Verificación de circuitos): se utiliza para asegurar que un circuito previamente diseñado va a comportarse de la forma adecuada. Número de problemas: 4.
- COL (Lógica combinatoria): es una sistema forma que satisface dos combinadores, la propiedad reflexiva, simétrica y transitiva y dos axiomas de substitución de igualdad para las funciones. Número de problemas: 141.
- COM (Teoría de la Computación): la Teoría de la Computación es un subcampo de la Informática que trata sobre aspectos teóricos tales como decidibilidad, completitud y complejidad computacional. Número de problemas: 6.

- GEO (Geometría): es una rama de las Matemáticas que trata con las medidas, propiedades y relaciones de los puntos, rectas, ángulos, planos y sólidos. En la librería TPTP se incluyen problemas sobre la geometría del plano basada en el sistema axiomático de Tarski para la geometría euclideana. Número de problemas: 165.

- GRA (Teoría de grafos): un grafo es un conjunto no vacío de nodos junto con los arcos que los unen. Número de problemas: 1.

- GRP (Teoría de grupos): un grupo es un sistema de elementos que es cerrado bajo una operaciones binaria que es asociativa, y que en relación a la cual existe un elemento que satisface la identidad y para todo elemento existe el denominado inverso. Número de problemas: 313.

- HEN (Modelos de Henkin): los modelos de Henkin proporcionan una semántica generalizada para lógicas de orden superior. Número de problemas: 64.

- LDA (Álgebra distributivas por la izquierda): este tipo de álgebras están relacionadas con grandes cardinales. Número de problemas: 23.

- LAT (Teoría de retículos): un retículo es un conjunto de elementos con dos operaciones binarias que son idempotentes, conmutativas y asociativas que además satisfacen la ley de absorción. Número de problemas: 10.

- MSC (Miscelánea): una colección de problemas que provienen de un dominio concreto. Número de problemas: 12.

- NUM (Teoría de los números): trata el estudio de los enteros y sus propiedades. Número de problemas: 309.

- PLA (Planificación): es el proceso de determinar la secuencia de acciones que han de ser realizadas por un agente para alcanzar el estado deseado. Número de problemas: 30.

- PRV (Verificación de programas): este campo de la Informática trata de la demostración formal de si los programas realizan las tareas para las cuales están diseñados. Número de problemas: 9.

- PUZ (Rompecabezas): Número de problemas: 45.

- RNG (Teoría de anillos): un anillo es un sistema de elementos que es un grupo abeliano con la operación de suma, y relativa a la cual existe un elemento que satisface la identidad y además todo elemento tiene su inverso. Número de problemas: 100.

- ROB (Álgebra de Robbins): problemas que tratar sobre las condiciones necesarias para que un álgebra que es casi booleana lo sea. Número de problemas: 36.

- SET (Teoría de conjuntos): axiomática y propiedades de los conjuntos. Número de problemas: 695.

- SYN (Sintácticos): un conjunto de problemas sintácticos que no tienen una interpretación obvia. Número de problemas: 350.

- TOP (Topología): trata el estudio de las configuraciones geométricas que no se ven alteradas por deformaciones elásticas. Número de problemas: 24.

En la página siguiente se incluye una tabla con la descripción de las características básicas de los problemas abordados. Los nombres dados a las columnas corresponden a las siguientes características: *Cl*, número total de cláusulas; *CnHn*, número de cláusulas que no son Horn; *CU*, *número de cláusulas* unitarias; *Lit*, número total de literales del problema; *Lit Ig*, número de literales relacionados con la teoría de la igualdad; *TMC*, tamaño de la cláusula con más literales; *Pre*, número total de símbolos de predicados; *Pro*, número de predicados proposicionales; *Fun*, número de símbolos de funciones; *Con*, número de constantes; *Var*, número total de variables; *VU*, numero de variables que sólo ocurren una vez; y por último, *PMT*, profundidad máxima de los términos del problema.

Nombre	*CI*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
BOO003-1	37	0	12	95	24	5	3	0	6	3	126	0	2
BOO003-2	23	0	16	31	31	3	1	0	6	3	44	0	3
BOO003-4	17	0	10	25	25	3	1	0	6	3	34	0	3
BOO004-1	37	0	12	95	24	5	3	0	6	3	126	0	2
BOO004-2	23	0	16	31	31	3	1	0	6	3	44	0	3
BOO004-4	17	0	10	25	25	3	1	0	6	3	34	0	3
BOO005-1	37	0	12	95	24	5	3	0	6	3	126	0	2
BOO006-1	37	0	12	95	24	5	3	0	6	3	126	0	2
BOO006-2	23	0	16	31	31	3	1	0	6	3	44	0	3
BOO012-1	37	0	12	95	25	5	3	0	6	3	126	0	3
BOO013-1	41	0	16	99	25	5	3	0	8	5	126	0	2
CAT001-3	34	2	8	71	41	4	3	0	8	4	64	4	3
CAT001-4	26	0	8	51	29	3	3	0	7	4	46	2	3
CAT002-3	35	2	8	74	44	4	3	0	8	4	67	4	3
CAT002-4	27	0	8	54	32	3	3	0	7	4	49	2	3
CAT003-3	34	2	8	71	41	4	3	0	8	4	64	4	3
CAT003-4	26	0	8	51	29	3	3	0	7	4	46	2	3
CAT004-3	35	2	8	74	44	4	3	0	8	4	67	4	3
CAT004-4	27	0	8	54	32	3	3	0	7	4	49	2	3
CAT005-1	34	0	10	80	22	4	4	0	5	2	88	5	2
CAT006-1	34	0	10	80	22	4	4	0	5	2	88	5	2
CAT006-4	25	0	6	50	27	3	3	0	5	2	45	2	3
CAT007-1	33	0	9	79	22	4	4	0	5	2	88	5	2
CAT007-3	12	2	5	23	15	3	2	0	6	2	15	0	2
CAT011-1	32	0	8	78	22	4	4	0	4	1	88	5	3
CAT011-2	15	0	6	26	26	3	1	0	4	1	27	0	3
CAT012-1	32	0	8	78	22	4	4	0	4	1	88	5	3
CAT012-3	31	2	6	66	37	4	3	0	5	1	61	4	3
CAT012-4	23	0	6	46	25	3	3	0	4	1	43	2	3
CAT013-1	32	0	8	78	22	4	4	0	4	1	88	5	3
CAT013-3	31	2	6	66	37	4	3	0	5	1	61	4	3
CAT013-4	23	0	6	46	25	3	3	0	4	1	43	2	3
CAT014-1	32	0	8	78	22	4	4	0	4	1	88	5	3
CAT014-2	15	0	6	26	26	3	1	0	4	1	27	0	3
CAT016-3	31	2	6	66	36	4	3	0	5	1	61	4	3
CAT016-4	23	0	6	46	24	3	3	0	4	1	43	2	3
CAT017-3	31	2	6	66	36	4	3	0	5	1	61	4	3
CAT017-4	23	0	6	46	24	3	3	0	4	1	43	2	3
CID002-1	58	0	24	96	57	3	2	0	11	3	163	1	4
COL001-1	8	0	4	13	13	3	1	0	4	3	18	1	4
COL001-2	11	0	7	16	16	3	1	0	7	6	23	1	6
COL002-1	10	0	6	15	15	3	1	0	6	5	23	0	4
COL002-2	11	0	6	17	15	3	2	0	6	5	24	0	5
COL002-3	11	0	6	17	15	3	2	0	6	5	24	0	6
COL007-1	7	0	3	12	12	3	1	0	3	2	15	0	3
COL008-1	8	0	4	13	13	3	1	0	4	3	17	0	4
COL009-1	8	0	4	13	13	3	1	0	4	3	18	0	4
COL010-1	8	0	4	13	13	3	1	0	4	3	19	0	4
COL011-1	8	0	4	13	13	3	1	0	4	3	18	0	4
COL012-1	7	0	3	12	12	3	1	0	3	2	15	0	4
COL013-1	8	0	4	13	13	3	1	0	4	3	18	0	4
COL014-1	8	0	4	13	13	3	1	0	4	3	17	0	3
COL015-1	8	0	4	13	13	3	1	0	4	3	17	0	4
COL016-1	9	0	5	14	14	3	1	0	5	4	19	0	4
COL017-1	9	0	5	14	14	3	1	0	5	4	19	0	4

Nombre	*CI*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
COL018-1	9	0	5	14	14	3	1	0	5	4	20	0	4
COL019-1	9	0	5	14	14	3	1	0	5	4	21	0	4
COL020-1	9	0	5	14	14	3	1	0	5	4	22	0	4
COL021-1	9	0	5	14	14	3	1	0	5	4	20	0	4
COL022-1	9	0	5	14	14	3	1	0	5	4	19	0	4
COL023-1	8	0	4	13	13	3	1	0	4	3	19	0	4
COL024-1	9	0	5	14	14	3	1	0	5	4	20	0	4
COL025-1	8	0	4	13	13	3	1	0	4	3	18	0	4
COL026-1	8	0	4	13	13	3	1	0	4	3	18	0	4
COL027-1	8	0	4	13	13	3	1	0	4	3	19	0	4
COL028-1	8	0	4	13	13	3	1	0	4	3	19	0	4
COL029-1	8	0	3	14	14	3	1	0	3	1	17	0	4
COL030-1	9	0	4	15	15	3	1	0	4	2	20	0	4
COL031-1	9	0	4	15	15	3	1	0	4	2	19	0	4
COL032-1	9	0	4	15	15	3	1	0	4	2	19	0	4
COL033-1	10	0	5	16	16	3	1	0	5	3	21	0	4
COL035-1	10	0	5	16	16	3	1	0	5	3	22	0	4
COL039-1	10	0	5	16	16	3	1	0	5	3	21	0	4
COL040-1	9	0	4	15	15	3	1	0	4	2	21	0	4
COL044-1	9	0	4	15	15	3	1	0	4	2	21	0	4
COL045-1	9	0	5	14	14	3	1	0	5	4	20	0	4
COL048-1	9	0	5	14	14	3	1	0	5	4	19	0	4
COL050-1	10	0	4	17	17	3	1	0	4	2	23	0	3
COL051-1	10	0	4	17	17	3	1	0	3	1	23	0	3
COL052-1	12	0	5	20	20	3	1	0	7	4	25	0	3
COL052-2	16	0	5	29	23	3	2	0	7	3	31	0	3
COL053-1	10	0	3	18	18	3	1	0	6	3	24	0	5
COL054-1	10	0	3	18	18	3	1	0	5	3	24	0	3
COL055-1	7	0	2	13	13	3	1	0	3	2	14	0	2
COL056-1	11	0	5	18	18	3	1	0	5	3	23	1	3
COL057-1	11	0	6	17	17	3	1	0	6	4	25	0	4
COL058-1	7	0	3	12	12	3	1	0	2	1	15	0	3
COL060-2	8	0	4	13	13	3	1	0	6	5	17	0	7
COL060-3	8	0	4	13	13	3	1	0	6	5	17	0	9
COL061-3	8	0	4	13	13	3	1	0	6	5	17	0	9
COL066-3	9	0	5	14	14	3	1	0	7	6	20	0	10
COL070-1	8	0	4	13	13	3	1	0	4	3	19	0	4
COL075-2	10	0	4	17	17	3	1	0	5	2	22	1	4
COM001-1	11	0	7	17	0	3	4	0	11	8	11	1	3
COM002-1	19	0	15	25	0	3	4	0	22	16	11	1	3
COM002-2	19	1	15	25	0	3	4	0	22	16	11	1	3
COM003-2	43	2	1	109	0	5	17	0	10	6	103	24	2
COM004-1	25	0	9	51	28	5	4	0	10	5	52	1	3
GEO002-3	84	9	18	194	92	8	3	0	11	5	358	8	3
GEO003-1	56	6	8	142	71	8	3	0	10	5	269	3	2
GEO003-2	54	5	8	134	70	8	3	0	10	5	261	3	2
GEO003-3	81	9	17	188	91	8	3	0	11	5	349	6	3
GEO011-2	61	6	8	153	73	8	4	0	8	3	285	3	2
GEO011-3	140	21	27	358	125	8	4	0	10	3	569	14	3
GEO011-4	61	6	8	151	75	8	4	0	9	3	277	3	3
GEO011-5	39	2	8	90	39	8	4	0	6	3	143	3	3
GEO014-2	54	5	8	134	70	8	3	0	10	5	261	3	2
GEO015-2	55	5	9	135	70	8	3	0	12	7	261	3	2
GEO015-3	56	5	10	136	70	8	3	0	12	7	263	3	2
GEO016-2	55	5	9	135	70	8	3	0	12	7	261	3	2
GEO016-3	57	5	10	138	70	8	3	0	12	7	267	3	2
GEO017-2	55	5	9	135	70	8	3	0	12	7	261	3	2

Nombre	*CI*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
GEO017-3	58	5	10	140	70	8	3	0	12	7	271	3	2
GEO018-2	55	5	9	135	70	8	3	0	12	7	261	3	2
GEO018-3	58	5	10	140	70	8	3	0	12	7	271	3	2
GEO019-2	55	5	9	135	70	8	3	0	12	7	261	3	2
GEO019-3	58	5	10	140	70	8	3	0	12	7	271	3	2
GEO021-2	55	5	9	135	70	8	3	0	12	7	261	3	2
GEO021-3	58	5	10	140	70	8	3	0	12	7	271	3	2
GEO022-2	56	5	10	136	70	8	3	0	14	9	261	3	2
GEO022-3	64	5	11	151	70	8	3	0	14	9	291	3	2
GEO024-2	54	5	8	134	70	8	3	0	10	5	261	3	2
GEO024-3	73	5	14	167	82	8	3	0	11	5	322	6	2
GEO027-3	78	6	17	180	86	8	3	0	12	6	334	6	2
GEO029-3	77	6	16	179	86	8	3	0	11	5	334	6	2
GEO035-2	54	5	8	134	71	8	3	0	11	6	261	3	2
GEO035-3	63	5	9	152	71	8	3	0	11	6	297	3	2
GEO038-2	55	5	9	135	71	8	3	0	13	8	261	3	2
GEO038-3	65	5	11	154	72	8	3	0	13	8	300	4	2
GEO039-2	56	5	10	136	71	8	3	0	12	7	261	3	2
GEO039-3	84	9	20	191	92	8	3	0	13	7	351	7	3
GEO041-3	88	9	21	200	94	8	3	0	12	6	363	9	3
GEO047-3	96	13	21	228	100	8	3	0	13	7	392	10	3
GEO053-3	126	20	29	322	122	8	3	0	13	6	518	14	3
GEO054-2	57	5	9	139	75	8	3	0	11	5	269	3	2
GEO054-3	68	5	11	160	77	8	3	0	11	5	313	6	2
GEO055-2	57	5	9	139	75	8	3	0	11	5	269	3	2
GEO055-3	68	5	11	160	77	8	3	0	11	5	313	6	2
GEO056-2	58	5	10	140	77	8	3	0	11	5	269	3	2
GEO056-3	71	5	14	163	79	8	3	0	11	5	317	6	2
GEO057-2	57	5	9	139	76	8	3	0	10	4	269	3	2
GEO057-3	70	5	13	162	78	8	3	0	10	4	317	6	2
GEO058-2	58	5	10	140	77	8	3	0	11	5	269	3	2
GEO058-3	73	5	15	166	82	8	3	0	11	5	320	6	2
GEO059-3	79	9	15	186	90	8	3	0	11	5	345	6	3
GEO064-3	133	21	28	343	125	8	4	0	13	6	545	14	3
GEO065-3	133	21	28	343	125	8	4	0	13	6	545	14	3
GEO066-3	133	21	28	343	125	8	4	0	13	6	545	14	3
GRA001-1	12	7	0	32	0	3	5	5	0	0	0	0	0
GRP001-1	20	0	9	40	16	4	2	0	6	4	49	0	2
GRP001-5	7	0	5	13	0	4	1	0	4	4	15	0	1
GRP003-1	5	0	3	11	0	4	1	0	3	2	14	0	2
GRP003-2	8	0	4	18	2	4	2	0	4	2	24	0	2
GRP004-1	5	0	3	11	0	4	1	0	3	2	15	1	2
GRP004-2	8	0	4	18	2	4	2	0	4	2	24	0	2
GRP005-1	9	0	6	18	0	4	2	0	3	2	19	0	2
GRP006-1	9	0	6	18	0	4	2	0	3	2	19	0	2
GRP007-1	20	0	9	40	17	4	2	0	4	2	50	0	2
GRP009-1	20	0	9	40	17	4	2	0	6	4	48	0	2
GRP010-1	19	0	8	39	16	4	2	0	5	3	48	0	2
GRP012-1	20	0	9	40	16	4	2	0	7	5	48	0	2
GRP012-2	20	0	9	40	17	4	2	0	7	5	48	0	2
GRP012-3	18	0	7	38	17	4	2	0	5	3	48	0	3
GRP013-1	22	0	10	43	16	4	2	0	7	5	52	0	2
GRP017-1	22	0	11	42	17	4	2	0	6	4	48	0	2
GRP018-1	18	0	7	38	17	4	2	0	4	2	48	0	2
GRP019-1	18	0	7	38	17	4	2	0	4	2	48	0	2
GRP020-1	18	0	7	38	17	4	2	0	4	2	48	0	3
GRP021-1	18	0	7	38	17	4	2	0	4	2	48	0	3

Nombre	*Cl*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
GRP022-1	18	0	7	38	17	4	2	0	4	2	48	0	3
GRP022-2	12	0	7	18	18	3	1	0	4	2	21	0	3
GRP023-1	18	0	7	38	17	4	2	0	3	1	48	0	2
GRP023-2	12	0	7	18	18	3	1	0	3	1	21	0	3
GRP027-1	41	1	18	82	37	6	3	0	10	6	101	1	5
GRP027-2	39	1	18	78	33	6	3	0	10	6	95	1	5
GRP028-1	4	0	3	7	0	4	1	0	3	0	11	0	2
GRP028-2	20	0	5	44	24	4	2	0	4	0	61	0	2
GRP028-3	6	0	4	12	0	4	1	0	4	0	19	0	2
GRP029-1	17	0	5	38	18	4	2	0	4	1	49	0	2
GRP029-2	16	0	5	36	16	4	2	0	4	1	47	0	2
GRP030-1	16	0	5	36	16	4	2	0	4	2	46	0	2
GRP031-1	16	0	5	36	16	4	2	0	4	2	47	1	2
GRP031-2	6	0	4	12	0	4	1	0	4	2	17	1	2
GRP032-3	21	0	8	46	17	4	3	0	4	2	53	0	2
GRP033-3	22	0	7	50	17	4	3	0	5	2	55	0	2
GRP033-4	23	0	7	52	19	4	3	0	5	2	57	0	2
GRP034-3	21	0	8	46	17	4	3	0	4	2	53	0	2
GRP034-4	9	0	6	18	0	4	2	0	4	2	20	0	2
GRP036-3	27	0	8	58	20	4	3	0	5	2	60	0	2
GRP037-3	30	0	9	65	22	4	3	0	6	3	68	0	2
GRP038-3	25	0	11	51	17	4	3	0	6	4	54	0	2
GRP041-2	8	0	4	18	3	4	2	0	4	2	24	0	2
GRP042-2	9	0	5	19	4	4	2	0	5	3	24	0	2
GRP043-2	10	0	6	20	5	4	2	0	6	4	24	0	2
GRP044-2	10	0	6	20	3	4	2	0	7	5	24	0	2
GRP045-2	10	0	6	20	3	4	2	0	7	5	24	0	2
GRP046-2	9	0	5	19	4	4	2	0	6	4	24	0	2
GRP047-2	9	0	5	19	4	4	2	0	6	4	24	0	2
GRP123-6.003	20	2	11	42	12	5	4	0	3	3	34	0	1
GRP123-7.003	26	2	16	50	12	5	7	0	3	3	37	0	1
GRP123-8.003	36	3	21	79	15	6	8	0	4	4	54	0	1
GRP123-9.003	22	4	11	52	12	5	5	0	6	6	38	0	1
GRP124-6.003	20	2	11	42	12	5	4	0	3	3	34	0	1
GRP124-7.003	26	2	16	50	12	5	7	0	3	3	37	0	1
GRP124-8.003	36	3	21	79	15	6	8	0	4	4	54	0	1
GRP126-1.002	10	1	5	21	5	4	3	0	2	2	19	0	1
GRP126-2.002	13	1	7	26	5	4	5	0	2	2	22	0	1
GRP126-3.002	22	2	11	53	8	6	6	0	3	3	39	0	1
GRP126-4.002	14	3	5	37	5	5	3	0	5	5	31	0	1
GRP131-1.002	10	1	4	27	7	5	3	0	2	2	26	0	1
GRP131-2.002	13	1	6	32	7	5	5	0	2	2	29	0	1
GRP132-1.002	10	1	4	27	7	5	3	0	2	2	26	0	1
GRP132-2.002	13	1	6	32	7	5	5	0	2	2	29	0	1
GRP135-1.002	9	1	4	20	5	4	3	0	2	2	18	0	1
GRP135-2.002	12	1	6	25	5	4	5	0	2	2	21	0	1
GRP136-1	30	0	19	42	42	3	1	0	7	3	65	2	3
GRP137-1	30	0	19	42	42	3	1	0	7	3	65	2	3
GRP139-1	30	0	19	42	42	3	1	0	8	4	65	2	3
GRP140-1	30	0	19	42	42	3	1	0	8	4	65	2	3
GRP142-1	28	0	17	40	40	3	1	0	7	3	65	2	3
GRP143-1	28	0	17	40	40	3	1	0	7	3	65	2	3
GRP144-1	28	0	17	40	40	3	1	0	7	3	65	2	3
GRP145-1	28	0	17	40	40	3	1	0	7	3	65	2	3
GRP146-1	30	0	19	42	42	3	1	0	8	4	65	2	3
GRP148-1	30	0	19	42	42	3	1	0	8	4	65	2	3
GRP150-1	28	0	17	40	40	3	1	0	7	3	65	2	3

Nombre	*CI*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
GRP151-1	28	0	17	40	40	3	1	0	7	3	65	2	3
GRP152-1	28	0	17	40	40	3	1	0	7	3	65	2	3
GRP153-1	28	0	17	40	40	3	1	0	7	3	65	2	3
GRP154-1	29	0	18	41	41	3	1	0	8	4	65	2	3
GRP155-1	29	0	18	41	41	3	1	0	8	4	66	3	3
GRP156-1	29	0	18	41	41	3	1	0	8	4	65	2	3
GRP157-1	29	0	18	41	41	3	1	0	8	4	65	2	3
GRP158-1	29	0	18	41	41	3	1	0	8	4	65	2	3
GRP160-1	28	0	17	40	40	3	1	0	6	2	65	2	3
GRP161-1	28	0	17	40	40	3	1	0	6	2	65	2	3
GRP162-1	30	0	19	42	42	3	1	0	8	4	65	2	3
GRP163-1	30	0	19	42	42	3	1	0	8	4	65	2	3
GRP165-1	29	0	18	41	41	3	1	0	6	2	65	2	3
GRP166-3	30	0	19	42	42	3	1	0	7	3	65	2	3
GRP168-1	29	0	18	41	41	3	1	0	8	4	65	2	4
GRP168-2	29	0	18	41	41	3	1	0	8	4	65	2	4
GRP176-1	28	0	17	40	40	3	1	0	9	5	65	2	4
GRP176-2	29	0	18	41	41	3	1	0	9	5	67	2	4
GRP182-1	28	0	17	40	40	3	1	0	6	2	65	2	3
GRP182-2	31	0	20	43	43	3	1	0	6	2	68	2	3
GRP182-3	28	0	17	40	40	3	1	0	6	2	65	2	3
GRP182-4	31	0	20	43	43	3	1	0	6	2	68	2	3
GRP186-4	31	0	20	43	43	3	1	0	7	3	68	2	4
GRP188-1	28	0	17	40	40	3	1	0	7	3	65	2	3
GRP188-2	31	0	20	43	43	3	1	0	7	3	68	2	3
GRP189-1	28	0	17	40	40	3	1	0	7	3	65	2	3
GRP189-2	31	0	20	43	43	3	1	0	7	3	68	2	3
HEN001-1	20	0	5	47	17	6	3	0	4	3	55	3	2
HEN001-3	15	0	6	28	16	3	2	0	4	3	31	3	3
HEN001-5	11	0	6	18	18	3	1	0	4	3	21	3	4
HEN002-1	20	0	5	47	17	6	3	0	4	3	55	3	2
HEN002-2	21	0	6	48	17	6	3	0	4	3	56	4	2
HEN002-3	15	0	6	28	16	3	2	0	4	3	31	3	3
HEN002-4	16	0	7	29	17	3	2	0	4	3	32	4	3
HEN002-5	11	0	6	18	18	3	1	0	4	3	21	3	4
HEN003-3	15	0	6	28	16	3	2	0	4	3	31	3	3
HEN003-4	17	0	8	30	18	3	2	0	4	3	33	5	3
HEN003-5	11	0	6	18	18	3	1	0	4	3	21	3	4
HEN004-4	18	0	9	31	19	3	2	0	4	3	34	5	3
HEN005-2	26	0	11	53	17	6	3	0	6	5	59	5	2
HEN006-4	21	0	11	36	19	3	2	0	6	5	37	5	3
HEN007-2	29	0	12	61	17	6	3	0	8	7	67	5	2
HEN007-4	22	0	11	38	19	3	2	0	6	5	40	5	3
HEN007-6	28	0	12	57	17	6	3	0	8	7	62	5	2
HEN008-1	23	0	8	50	17	6	3	0	8	7	55	3	2
HEN008-2	30	0	12	65	17	6	3	0	8	7	72	5	2
HEN008-3	16	0	7	29	15	3	2	0	6	5	31	3	3
HEN008-4	23	0	11	40	19	3	2	0	6	5	43	5	3
HEN008-5	12	0	7	19	19	3	1	0	6	5	21	3	4
HEN008-6	19	0	10	32	18	3	2	0	6	5	34	5	3
HEN010-4	24	0	11	42	21	3	2	0	4	3	47	5	4
HEN010-6	23	0	11	39	21	3	2	0	4	3	44	5	4
HEN012-3	15	0	6	28	15	3	2	0	4	3	31	3	3
LCL006-1	4	0	3	6	0	3	1	0	4	3	7	0	4
LCL007-1	4	0	3	6	0	3	1	0	4	3	7	0	4
LCL008-1	3	0	2	5	0	3	1	0	3	2	5	0	4
LCL009-1	3	0	2	5	0	3	1	0	4	3	5	0	4

Nombre	*Cl*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
LCL010-1	3	0	2	5	0	3	1	0	4	3	5	0	4
LCL011-1	3	0	2	5	0	3	1	0	4	3	5	0	4
LCL013-1	3	0	2	5	0	3	1	0	4	3	5	0	5
LCL022-1	3	0	2	5	0	3	1	0	4	3	5	0	4
LCL023-1	3	0	2	5	0	3	1	0	4	3	5	0	4
LCL025-1	5	0	4	7	0	3	1	0	5	4	8	1	4
LCL027-1	5	0	4	7	0	3	1	0	3	2	8	1	4
LCL029-1	6	0	5	8	0	3	1	0	3	2	10	3	4
LCL033-1	3	0	2	5	0	3	1	0	4	3	7	2	6
LCL035-1	3	0	2	5	0	3	1	0	3	2	7	2	6
LCL040-1	7	0	6	9	0	3	1	0	5	3	11	1	4
LCL041-1	7	0	6	9	0	3	1	0	4	2	14	2	5
LCL043-1	7	0	6	9	0	3	1	0	3	1	14	2	5
LCL044-1	7	0	6	9	0	3	1	0	3	1	14	2	5
LCL045-1	7	0	6	9	0	3	1	0	4	2	14	2	5
LCL046-1	5	0	4	7	0	3	1	0	3	1	8	1	4
LCL064-1	5	0	4	7	0	3	1	0	5	3	9	1	4
LCL064-2	4	0	3	6	0	3	1	0	4	3	7	1	4
LCL065-1	5	0	4	7	0	3	1	0	3	1	9	1	4
LCL066-1	5	0	4	7	0	3	1	0	4	2	9	1	4
LCL069-1	5	0	4	7	0	3	1	0	2	0	13	3	5
LCL072-1	5	0	4	7	0	3	1	0	4	2	12	2	6
LCL076-1	5	0	4	7	0	3	1	0	3	1	9	1	4
LCL076-2	6	0	5	8	0	3	1	0	3	1	10	1	4
LCL076-3	6	0	4	10	0	3	1	0	3	1	12	1	4
LCL077-1	5	0	4	7	0	3	1	0	3	1	9	1	4
LCL077-2	6	0	4	10	0	3	1	0	3	1	12	1	4
LCL079-1	7	0	6	9	0	3	1	0	5	3	9	1	4
LCL081-1	3	0	2	5	0	3	1	0	2	1	6	2	4
LCL082-1	3	0	2	5	0	3	1	0	3	2	6	2	4
LCL083-1	3	0	2	5	0	3	1	0	3	2	6	2	4
LCL083-2	4	0	3	6	0	3	1	0	3	2	7	2	4
LCL087-1	3	0	2	5	0	3	1	0	3	2	7	2	5
LCL091-1	3	0	2	5	0	3	1	0	3	2	7	2	5
LCL096-1	5	0	4	7	0	3	1	0	4	3	16	0	7
LCL097-1	4	0	3	6	0	3	1	0	6	5	11	0	7
LCL098-1	3	0	2	5	0	3	1	0	6	5	7	0	7
LCL101-1	4	0	3	6	0	3	1	0	5	4	10	0	6
LCL102-1	5	0	4	7	0	3	1	0	5	4	12	0	6
LCL104-1	4	0	3	6	0	3	1	0	5	4	9	0	6
LCL106-1	4	0	3	6	0	3	1	0	4	3	8	0	5
LCL107-1	3	0	2	5	0	3	1	0	5	4	9	0	7
LCL108-1	3	0	2	5	0	3	1	0	4	3	9	0	7
LCL110-1	6	0	5	8	0	3	1	0	3	1	11	1	4
LCL111-1	6	0	5	8	0	3	1	0	5	3	11	1	4
LCL112-1	6	0	5	8	0	3	1	0	3	1	11	1	4
LCL117-1	3	0	2	5	0	3	1	0	4	3	5	0	4
LCL118-1	3	0	2	5	0	3	1	0	4	3	5	0	4
LCL120-1	3	0	2	5	0	3	1	0	4	3	5	0	5
LCL126-1	4	0	3	6	0	3	1	0	4	3	8	0	5
LCL130-1	3	0	2	5	0	3	1	0	5	4	6	0	6
LCL132-1	11	0	6	17	17	3	1	0	4	2	22	0	4
LCL143-1	22	0	9	38	30	3	2	0	8	4	48	0	4
LCL169-1	9	0	6	14	0	3	2	0	3	1	17	1	5
LCL170-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL171-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL172-1	9	0	6	14	0	3	2	0	5	3	17	1	6

Nombre	*Cl*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
LCL173-1	9	0	6	14	0	3	2	0	5	3	17	1	6
LCL174-1	9	0	6	14	0	3	2	0	5	3	17	1	6
LCL175-1	9	0	6	14	0	3	2	0	3	1	17	1	5
LCL176-1	9	0	6	14	0	3	2	0	3	1	17	1	5
LCL177-1	9	0	6	14	0	3	2	0	3	1	17	1	5
LCL178-1	9	0	6	14	0	3	2	0	3	1	17	1	5
LCL181-2	4	1	2	6	0	2	2	2	0	0	0	0	0
LCL182-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL185-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL186-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL187-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL188-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL189-1	9	0	6	14	0	3	2	0	4	2	17	1	6
LCL190-1	9	0	6	14	0	3	2	0	5	3	17	1	5
LCL192-1	9	0	6	14	0	3	2	0	5	3	17	1	5
LCL193-1	9	0	6	14	0	3	2	0	5	3	17	1	5
LCL194-1	9	0	6	14	0	3	2	0	5	3	17	1	5
LCL195-1	9	0	6	14	0	3	2	0	5	3	17	1	5
LCL196-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL197-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL198-1	9	0	6	14	0	3	2	0	4	2	17	1	6
LCL199-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL200-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL201-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL202-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL203-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL204-1	9	0	6	14	0	3	2	0	4	2	17	1	6
LCL205-1	9	0	6	14	0	3	2	0	4	2	17	1	6
LCL206-1	9	0	6	14	0	3	2	0	4	2	17	1	6
LCL207-1	9	0	6	14	0	3	2	0	4	2	17	1	6
LCL208-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL210-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL211-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL212-1	9	0	6	14	0	3	2	0	4	2	17	1	7
LCL213-1	9	0	6	14	0	3	2	0	4	2	17	1	7
LCL214-1	9	0	6	14	0	3	2	0	4	2	17	1	6
LCL215-1	9	0	6	14	0	3	2	0	4	2	17	1	5
LCL216-1	9	0	6	14	0	3	2	0	4	2	17	1	6
LCL217-1	9	0	6	14	0	3	2	0	4	2	17	1	6
LCL218-1	9	0	6	14	0	3	2	0	4	2	17	1	6
LCL226-1	9	0	6	14	0	3	2	0	5	3	17	1	6
LCL230-1	9	0	6	14	0	3	2	0	5	3	17	1	6
LCL230-2	4	1	3	6	0	3	3	3	0	0	0	0	0
LCL231-1	9	0	6	14	0	3	2	0	5	3	17	1	7
LDA003-1	14	0	7	25	16	3	2	0	5	4	26	1	3
SET027-7	167	8	34	344	160	5	10	0	41	11	396	36	6
SET043-5	2	1	0	4	0	2	1	0	1	1	2	0	1
SET044-5	4	1	0	8	0	2	1	0	2	1	6	0	2
SET045-5	4	1	1	8	0	3	1	0	2	1	7	2	2
SET046-5	3	2	0	7	0	3	1	0	2	1	4	0	2
SET050-6	161	8	32	334	160	5	10	0	42	12	379	26	6
SET051-6	161	8	32	334	160	5	10	0	42	12	379	26	6
SET052-6	161	8	32	334	160	5	10	0	42	12	379	26	6
SET053-6	161	8	32	334	160	5	10	0	42	12	379	26	6
SET054-6	160	8	31	333	160	5	10	0	39	9	379	26	6
SET054-7	164	8	31	341	160	5	10	0	39	9	395	36	6
SET055-6	159	8	30	332	160	5	10	0	39	9	378	26	6

Nombre	*CI*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
SET055-7	165	8	31	344	160	5	10	0	39	9	398	36	6
SET056-6	162	8	33	335	161	5	10	0	40	10	379	26	6
SET056-7	168	8	34	347	161	5	10	0	40	10	399	36	6
SET057-6	162	8	33	335	161	5	10	0	40	10	379	26	6
SET057-7	168	8	34	347	161	5	10	0	40	10	399	36	6
SET058-6	162	8	33	335	161	5	10	0	40	10	379	26	6
SET058-7	168	8	34	347	161	5	10	0	40	10	399	36	6
SET059-6	162	8	33	335	161	5	10	0	40	10	379	26	6
SET059-7	168	8	34	347	161	5	10	0	40	10	399	36	6
SET060-6	160	8	31	333	160	5	10	0	40	10	379	26	6
SET060-7	170	11	32	357	164	5	10	0	40	10	407	36	6
SET061-7	171	11	33	358	164	5	10	0	39	9	409	37	6
SET062-7	172	11	34	359	164	5	10	0	39	9	410	38	6
SET063-7	174	11	36	361	165	5	10	0	39	9	411	39	6
SET064-7	175	11	36	363	166	5	10	0	39	9	412	39	6
SET065-6	160	8	31	333	160	5	10	0	38	8	379	26	6
SET065-7	175	12	35	364	166	5	10	0	38	8	413	39	6
SET073-7	185	15	40	382	175	5	10	0	40	10	431	41	6
SET074-7	185	15	40	382	175	5	10	0	40	10	431	41	6
SET075-7	185	15	40	382	175	5	10	0	42	12	431	41	6
SET077-6	160	8	31	333	160	5	10	0	39	9	379	26	6
SET077-7	188	15	39	390	177	5	10	0	39	9	442	45	6
SET078-6	160	8	31	333	160	5	10	0	40	10	379	26	6
SET078-7	189	15	40	391	177	5	10	0	40	10	443	46	6
SET079-7	192	15	42	395	178	5	10	0	39	9	446	47	6
SET080-7	192	15	41	396	178	5	10	0	38	8	447	47	6
SET081-6	161	8	32	334	161	5	10	0	40	10	379	26	6
SET081-7	194	15	43	398	179	5	10	0	40	10	447	47	6
SET090-7	204	19	43	422	194	5	10	0	40	9	463	47	6
SET093-6	162	8	32	336	163	5	10	0	40	9	381	26	6
SET093-7	208	21	43	430	200	5	10	0	41	9	468	47	6
SET095-7	210	21	43	435	202	5	10	0	42	10	471	47	6
SET098-7	214	23	44	444	208	5	10	0	41	9	476	47	6
SET101-6	160	8	31	333	160	5	10	0	40	10	379	26	6
SET101-7	216	25	44	450	212	5	10	0	42	10	482	49	6
SET102-6	160	8	31	333	160	5	10	0	40	10	379	26	6
SET102-7	216	25	44	450	212	5	10	0	42	10	482	49	6
SET108-6	161	8	32	334	160	5	10	0	40	10	379	26	6
SET108-7	224	28	47	467	219	5	10	0	42	10	500	54	6
SET117-6	161	8	32	334	161	5	10	0	39	9	379	26	6
SET117-7	237	36	47	493	238	5	10	0	43	9	518	54	6
SET118-6	161	8	32	334	160	5	10	0	39	9	379	26	6
SET118-7	237	36	47	493	237	5	10	0	43	9	518	54	6
SET152-6	188	8	39	387	182	5	11	0	46	12	437	33	6
SET153-6	188	8	39	387	182	5	11	0	47	13	437	33	6
SET158-6	190	8	41	389	182	5	11	0	49	15	437	33	6
SET196-6	188	8	39	387	181	5	11	0	48	14	437	33	6
SET197-6	188	8	39	387	181	5	11	0	48	14	437	33	6
SET231-6	192	8	43	391	185	5	11	0	49	15	440	33	6
SET232-6	189	8	40	388	181	5	11	0	49	15	437	33	6
SET233-6	189	8	40	388	181	5	11	0	49	15	437	33	6
SET234-6	189	8	40	388	181	5	11	0	49	15	437	33	6
SET239-6	189	8	40	388	181	5	11	0	50	16	437	33	6
SET240-6	189	8	40	388	181	5	11	0	51	17	437	33	6
SET241-6	189	8	40	388	181	5	11	0	51	17	437	33	6
SET242-6	190	8	41	389	181	5	11	0	50	16	437	33	6
SET252-6	188	8	39	387	181	5	11	0	49	15	437	33	6

Nombre	*CI*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
SET253-6	188	8	39	387	181	5	11	0	49	15	437	33	6
SET296-6	188	8	39	387	182	5	11	0	47	13	437	33	6
SET451-6	188	8	39	387	181	5	11	0	46	12	437	33	6
SET479-6	189	8	40	388	181	5	11	0	47	13	437	33	6
SET553-6	188	8	39	387	181	5	11	0	47	13	437	33	6
SET563-6	190	8	41	389	181	5	11	0	52	18	437	33	6
SYN003-1.006	24	0	4	54	0	3	18	18	0	0	0	0	0
SYN004-1.007	15	0	2	40	0	3	14	14	0	0	0	0	0
SYN005-1.010	11	0	10	20	0	10	10	0	1	1	10	0	1
SYN006-1	7	2	4	12	0	3	6	0	3	1	13	7	4
SYN007-1.002	8	5	0	24	0	4	2	2	0	0	0	0	0
SYN008-1	6	3	3	13	0	4	7	7	0	0	0	0	0
SYN009-1	7	1	6	12	0	6	4	0	3	3	9	6	1
SYN011-1	8	2	1	17	0	3	7	7	0	0	0	0	0
SYN014-2	26	13	7	70	43	6	2	0	6	4	29	0	2
SYN028-1	6	1	1	12	0	3	5	5	0	0	0	0	0
SYN029-1	5	1	1	10	0	3	3	3	0	0	0	0	0
SYN030-1	9	1	0	22	0	3	5	5	0	0	0	0	0
SYN031-1	5	2	0	10	0	2	1	0	2	1	8	3	2
SYN032-1	7	3	0	17	0	3	6	6	0	0	0	0	0
SYN033-1	4	0	3	7	0	4	1	0	3	0	11	0	2
SYN034-1	3	2	0	7	0	3	1	0	2	1	4	0	2
SYN035-1	3	0	1	8	0	4	2	0	1	0	6	2	2
SYN040-1	4	0	2	6	0	2	2	2	0	0	0	0	0
SYN041-1	4	0	4	4	0	1	2	2	0	0	0	0	0
SYN044-1	6	2	0	13	0	3	3	3	0	0	0	0	0
SYN045-1	4	2	1	7	0	2	3	3	0	0	0	0	0
SYN046-1	3	0	2	4	0	2	2	2	0	0	0	0	0
SYN047-1	5	1	2	10	0	3	4	4	0	0	0	0	0
SYN048-1	2	0	2	2	0	1	1	0	1	0	2	2	2
SYN049-1	3	0	2	4	0	2	2	0	2	0	3	2	2
SYN050-1	5	0	3	9	0	3	4	0	3	2	6	4	2
SYN051-1	4	1	0	8	0	2	2	1	2	2	2	2	1
SYN052-1	5	1	0	10	0	2	2	1	1	1	4	4	1
SYN053-1	5	3	1	12	0	3	2	1	2	2	4	4	1
SYN054-1	6	2	0	13	0	3	4	0	2	2	5	0	1
SYN055-1	7	2	1	16	0	3	5	0	2	2	6	0	1
SYN057-1	7	0	4	13	0	4	5	0	2	2	4	0	1
SYN058-1	9	0	3	16	0	3	6	0	3	3	4	3	1
SYN060-1	7	4	1	13	0	2	4	0	1	1	6	0	1
SYN061-1	6	1	2	10	0	2	5	0	1	1	4	0	1
SYN062-1	7	0	3	14	0	3	6	0	1	1	4	0	1
SYN063-1	7	3	2	22	0	5	1	0	5	5	4	4	1
SYN063-2	3	0	2	4	0	2	1	0	2	2	0	0	1
SYN064-1	2	0	2	2	0	1	1	0	2	0	4	2	2
SYN065-1	7	0	3	15	0	3	3	0	3	1	15	1	2
SYN066-1	6	1	2	10	0	2	3	0	5	1	12	4	2
SYN068-1	7	0	1	13	0	2	4	0	3	1	6	0	2
SYN069-1	9	2	1	27	0	5	6	0	3	1	11	0	2
SYN070-1	9	4	2	26	0	5	4	0	3	2	9	0	2
SYN073-1	8	1	2	17	10	3	2	0	2	1	17	1	2
SYN079-1	4	0	3	6	0	3	1	0	4	3	3	0	2
SYN080-1	7	0	3	12	12	3	1	0	4	2	12	2	3
SYN081-1	3	1	0	6	0	2	1	0	1	0	3	0	2
SYN082-1	8	4	0	25	0	4	1	0	4	2	9	0	2
SYN083-1	7	0	3	12	12	3	1	0	5	4	15	0	4
SYN084-2	7	2	1	19	0	4	1	0	4	3	2	0	3

Nombre	*CI*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
SYN085-1.010	12	0	11	22	0	11	11	11	0	0	0	0	0
SYN088-1.010	22	0	21	32	0	11	11	0	2	2	10	0	1
SYN089-1.002	25	0	5	61	0	3	16	16	0	0	0	0	0
SYN095-1.002	25	0	5	61	0	3	16	0	1	1	21	1	1
SYN098-1.002	68	25	2	176	0	4	54	54	0	0	0	0	0
SYN103-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN104-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN105-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN106-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN107-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN108-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN109-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN110-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN111-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN112-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN113-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN114-1	369	0	39	106	0	5	48	0	5	5	627	160	1
SYN115-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN116-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN117-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN118-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN119-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN120-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN121-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN122-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN123-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN124-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN125-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN126-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN127-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN128-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN129-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN130-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN131-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN132-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN133-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN134-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN135-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN136-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN137-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN138-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN139-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN140-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN141-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN144-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN145-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN146-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN147-1	369	0	39	106	0	5	48	0	5	5	627	160	1
SYN148-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN149-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN150-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN151-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN152-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN153-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN154-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN157-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN158-1	369	0	39	106	0	5	48	0	5	5	626	160	1

Nombre	*CI*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
SYN159-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN160-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN161-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN162-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN164-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN165-1	369	0	39	106	0	5	48	0	5	5	627	160	1
SYN166-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN167-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN168-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN169-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN170-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN171-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN172-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN173-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN174-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN175-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN176-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN177-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN178-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN181-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN182-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN183-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN184-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN185-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN186-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN187-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN188-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN189-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN190-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN191-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN192-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN193-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN194-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN195-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN196-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN197-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN198-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN199-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN200-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN201-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN202-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN203-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN204-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN205-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN206-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN207-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN208-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN209-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN210-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN211-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN212-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN213-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN215-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN216-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN217-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN218-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN219-1	369	0	39	106	0	5	48	0	5	5	626	160	1

Nombre	*Cl*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
SYN220-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN221-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN222-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN223-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN224-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN225-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN226-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN227-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN228-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN229-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN230-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN231-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN232-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN233-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN234-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN235-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN236-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN237-1	369	0	39	106	0	5	48	0	5	5	627	160	1
SYN238-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN239-1	369	0	39	106	0	5	48	0	5	5	627	160	1
SYN240-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN241-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN242-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN243-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN244-1	369	0	39	106	0	5	48	0	5	5	627	160	1
SYN245-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN246-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN247-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN248-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN249-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN250-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN251-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN253-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN254-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN255-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN256-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN257-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN258-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN259-1	369	0	39	106	0	5	48	0	5	5	627	160	1
SYN260-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN261-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN262-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN263-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN264-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN265-1	369	0	39	106	0	5	48	0	5	5	627	160	1
SYN266-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN267-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN268-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN270-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN271-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN272-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN273-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN274-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN275-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN276-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN277-1	369	0	39	106	0	5	48	0	5	5	627	160	1
SYN278-1	369	0	39	106	0	5	48	0	5	5	628	162	1

Nombre	*Cl*	*CnHn*	*CU*	*Lit*	*Lit Ig*	*TMC*	*Pre*	*Pro*	*Fun*	*Con*	*Var*	*VU*	*PMT*
SYN279-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN280-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN281-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN282-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN283-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN284-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN285-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN286-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN287-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN288-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN289-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN290-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN291-1	369	0	39	106	0	5	48	0	5	5	628	162	1
SYN292-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN293-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN294-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN295-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN296-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN297-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN298-1	369	0	39	106	0	5	48	0	5	5	627	161	1
SYN299-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN300-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN301-1	369	0	39	106	0	5	48	0	5	5	626	160	1
SYN315-1	4	1	0	8	0	2	2	0	2	1	4	4	2
SYN318-1	4	0	3	6	0	3	3	0	2	2	2	1	1
SYN319-1	7	1	1	13	0	2	3	0	3	1	11	6	2
SYN321-1	4	1	0	8	0	2	2	0	3	3	4	3	1
SYN323-1	4	1	0	8	0	2	2	0	1	1	4	0	1
SYN325-1	5	2	1	9	0	2	2	0	1	0	5	1	2
SYN326-1	7	3	2	12	0	2	3	0	2	0	7	0	2
SYN327-1	6	2	0	14	0	3	1	0	2	1	6	0	2
SYN331-1	7	1	4	10	0	2	1	0	1	0	14	4	2
SYN333-1	3	0	1	8	0	4	2	0	1	0	6	2	2
SYN336-1	5	0	5	5	0	1	1	0	2	1	9	6	2
SYN338-1	3	0	3	3	0	1	1	0	1	0	4	2	2
SYN339-1	2	0	2	2	0	1	1	0	1	0	4	2	2
SYN340-1	2	0	2	2	0	1	1	0	1	0	6	2	2
SYN341-1	2	0	2	2	0	1	1	0	2	0	4	1	2
SYN343-1	3	1	0	6	0	2	1	0	2	1	5	5	2
SYN345-1	8	4	0	24	0	3	1	0	3	2	13	5	2
SYN346-1	4	0	2	6	0	2	1	0	3	1	8	3	2
SYN349-1	10	7	0	36	0	4	1	0	2	0	20	0	2
SYN350-1	6	2	0	16	0	3	1	0	2	1	12	0	2
SYN352-1	7	4	1	18	0	3	1	0	3	2	12	0	2
SYN354-1	7	1	2	17	0	4	2	0	3	2	10	1	2
TOP001-2	13	1	3	27	0	3	5	0	7	2	25	0	3
TOP002-2	3	1	2	4	0	2	2	0	4	2	3	1	2
TOP004-1	113	23	4	340	0	8	22	0	37	3	361	60	3
TOP004-2	21	1	5	59	0	6	5	0	8	2	59	6	2
TOP005-2	12	2	2	30	0	4	4	0	7	2	27	0	2

Tabla 17: Características de los problemas resueltos

A continuación se incluyen las tablas con los resultados temporales y espaciales de todos los problemas resueltos por el procedimiento SLT vía las diferentes elecciones de ancestros. En primer lugar se incluye la tabla de resultados para el SLT vía ALL.

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
BOO003-1	3470	0	10	0	67035	4	0	3753
BOO003-2	403170	0	13	0	3332059	4	16	131492
BOO003-4	68600	0	13	0	607017	4	0	31408
BOO004-1	1420	0	10	0	26913	4	0	2030
BOO004-2	337470	0	13	0	2837456	4	0	113174
BOO004-4	72510	0	13	0	651355	4	0	33419
BOO005-1	3590	0	10	0	66617	4	0	4850
BOO006-1	5970	0	10	0	113055	4	0	6594
BOO006-2	952560	0	13	0	6617386	4	232	170392
BOO012-1	303890	0	12	0	5580542	9	0	306779
CAT001-3	30480	0	9	1597	792673	330	101	6381
CAT001-4	3210	0	9	0	44415	83	0	1890
CAT002-3	5820	0	8	237	152488	123	22	1520
CAT002-4	1580	0	8	0	26275	55	0	820
CAT003-3	700	0	7	21	17920	21	0	223
CAT003-4	210	0	7	0	2979	8	0	132
CAT004-3	269230	0	10	17506	6714343	3459	1013	50599
CAT004-4	23150	0	10	0	368939	1151	0	12452
CAT005-1	39870	0	12	0	611445	8119	764	50702
CAT006-1	38510	0	12	0	589617	7567	760	49305
CAT006-4	902460	0	14	0	11619263	6633	0	358659
CAT007-1	140	0	7	0	2274	8	0	110
CAT007-3	0	0	4	0	21	4	0	0
CAT011-1	20	0	6	0	353	4	0	31
CAT011-2	40	0	7	0	404	4	0	53
CAT012-1	20	0	6	0	290	6	0	27
CAT012-3	780	0	8	26	18433	30	0	282
CAT012-4	220	0	8	0	3140	16	0	181
CAT013-1	20	0	6	0	267	4	0	27
CAT013-3	380	0	7	11	8225	14	0	162
CAT013-4	180	0	7	0	2464	7	0	128
CAT014-1	20	0	6	0	376	6	0	31
CAT014-2	130	0	8	0	1279	4	0	143
CAT016-3	130	0	6	0	2255	2	0	38
CAT016-4	80	0	6	0	1173	2	0	32
CAT017-3	130	0	6	0	2454	2	0	41
CAT017-4	150	0	7	0	2130	3	0	71
COL001-1	294440	0	12	0	1978220	1	6	86557
COL001-2	270	0	7	0	2009	1	0	105
COL002-1	40	0	6	0	362	1	0	23
COL002-2	90	0	8	0	643	1	0	73
COL002-3	30	0	7	0	175	1	0	19
COL007-1	0	0	2	0	5	1	0	0
COL008-1	0	0	4	0	35	1	0	2
COL009-1	130	0	7	0	1105	1	0	88
COL010-1	10	0	4	0	35	1	0	2
COL011-1	319830	0	12	0	2028129	1	23	90762
COL012-1	0	0	2	0	5	1	0	0
COL013-1	10	0	2	0	5	1	0	0
COL014-1	0	0	2	0	5	1	0	0
COL015-1	10	0	4	0	31	1	0	2
COL016-1	0	0	2	0	5	1	0	0
COL017-1	10	0	4	0	35	1	0	2
COL018-1	0	0	2	0	5	1	0	0
COL019-1	30	0	6	0	318	1	0	23
COL020-1	40	0	6	0	307	1	0	23

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
COL021-1	0	0	4	0	35	1	0	2
COL022-1	0	0	4	0	35	1	0	2
COL023-1	130	0	7	0	1049	1	0	88
COL024-1	0	0	4	0	35	1	0	2
COL025-1	0	0	4	0	35	1	0	2
COL026-1	120	0	7	0	1049	1	0	88
COL027-1	140	0	7	0	1050	1	0	88
COL028-1	140	0	7	0	1049	1	0	88
COL029-1	0	0	2	0	5	1	0	0
COL030-1	300	0	8	0	2227	1	0	217
COL031-1	10	0	5	0	67	1	0	7
COL032-1	6850	0	10	0	45694	1	0	3592
COL033-1	369670	0	12	0	2026707	11	20	95259
COL035-1	3090	0	9	0	19234	1	0	1114
COL039-1	150	0	7	0	1133	1	0	98
COL040-1	277830	0	12	0	1519199	1	36	82690
COL044-1	277430	0	12	0	1519199	1	36	82690
COL045-1	10	0	4	0	38	1	0	2
COL048-1	0	0	4	0	35	1	0	2
COL050-1	0	0	5	0	50	1	0	4
COL051-1	60	0	7	0	524	1	0	49
COL052-1	15440	0	10	0	116778	1	0	6877
COL052-2	20450	0	11	0	168372	9	0	10309
COL053-1	10	0	4	0	29	1	0	2
COL054-1	170	0	7	0	1699	1	0	76
COL055-1	0	0	3	0	15	2	0	0
COL056-1	210	0	7	0	2103	1	0	123
COL057-1	1039450	0	12	0	5330463	3	16	181260
COL058-1	150	0	8	0	1283	1	0	145
COL060-2	1015170	0	15	0	7248461	3	0	512316
COL060-3	244650	0	14	0	1691524	3	0	131984
COL061-3	821460	0	15	0	5560593	4	0	408920
COL070-1	110	0	7	0	981	1	0	87
COL075-2	44760	0	11	0	297208	1	0	18276
COM001-1	0	0	7	0	62	4	0	0
COM002-1	160	0	13	0	933	4	0	0
COM002-2	150	0	13	0	933	4	0	0
COM003-2	120	1	8	45	1586	35	10	74
COM004-1	230	0	10	0	2616	2	0	299
GEO002-3	0	0	3	0	45	3	0	0
GEO003-1	40	0	6	2	641	4	0	32
GEO003-2	20	0	6	0	307	2	0	28
GEO003-3	10	0	3	0	27	2	0	0
GEO011-2	0	0	5	0	69	5	0	3
GEO011-3	20	0	5	0	516	5	0	8
GEO011-4	10	0	5	0	78	5	0	3
GEO011-5	10	0	5	0	66	5	0	3
GEO014-2	10	0	4	0	19	3	0	0
GEO015-2	20	0	6	0	353	3	0	23
GEO015-3	10	0	4	0	46	3	0	1
GEO016-2	0	0	4	0	25	3	0	0
GEO016-3	0	0	4	0	35	3	0	0
GEO017-2	280	0	8	0	5999	4	0	247
GEO017-3	10	0	5	0	216	3	0	10
GEO018-2	30	0	6	0	316	3	0	22
GEO018-3	40	0	6	0	676	3	0	51
GEO019-2	0	0	4	0	39	3	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
GEO019-3	10	0	4	0	66	3	0	1
GEO021-2	10	0	6	0	261	4	0	17
GEO021-3	30	0	5	0	506	4	0	36
GEO022-2	250	0	8	0	6019	4	0	247
GEO022-3	30	0	5	0	1104	3	0	15
GEO024-2	30	0	6	0	535	2	0	39
GEO024-3	30	0	5	0	747	2	0	66
GEO027-3	1420	0	7	0	38440	9	0	925
GEO035-2	0	0	3	0	26	2	0	0
GEO035-3	0	0	3	0	26	2	0	0
GEO038-2	10	0	5	0	139	3	0	11
GEO038-3	20	0	5	0	178	3	0	13
GEO039-3	2390	0	7	2	105022	4	4	584
GEO041-3	28930	0	6	19	1376199	10	2	743
GEO047-3	38060	0	7	59	1714527	54	74	2396
GEO053-3	118180	0	7	158	5496668	77	46	6942
GEO054-2	10	0	5	0	141	3	0	11
GEO054-3	0	0	3	0	27	2	0	0
GEO055-2	10	0	5	0	157	3	0	9
GEO055-3	40	0	5	0	1512	3	0	28
GEO056-2	9440	0	8	16	191847	2	0	4009
GEO056-3	30	0	5	0	686	3	0	14
GEO057-2	50	0	6	0	936	2	0	33
GEO057-3	0	0	3	0	39	2	0	0
GEO058-2	285790	0	9	94	5344209	2	0	106510
GEO058-3	290	0	6	0	8021	4	0	151
GEO059-3	340	0	6	0	12031	4	0	261
GEO064-3	10	0	4	0	181	4	0	0
GEO065-3	0	0	4	0	273	4	0	0
GEO066-3	0	0	4	0	365	4	0	0
GRA001-1	1450	12	17	4647	21032	7849	0	5405
GRP001-1	16050	0	11	0	238408	3	3	13091
GRP001-5	410	0	11	0	3407	3	0	531
GRP003-1	40	0	11	0	230	3	0	3
GRP003-2	780	0	11	0	7948	3	0	141
GRP004-1	10	0	8	0	91	1	0	8
GRP004-2	40	0	8	0	443	3	0	9
GRP005-1	0	0	5	0	12	3	0	0
GRP006-1	10	0	8	0	42	3	0	0
GRP007-1	0	0	4	0	54	3	0	2
GRP009-1	2720	0	10	0	40570	5	0	2892
GRP010-1	80	0	8	0	1142	3	0	116
GRP012-1	190	0	8	0	2928	3	0	197
GRP012-2	254500	0	13	0	3510010	4	170	230301
GRP012-3	270750	0	13	0	3537618	4	170	247797
GRP013-1	2310	0	9	0	38444	3	0	1702
GRP017-1	50	0	7	0	595	4	0	54
GRP018-1	10	0	4	0	39	3	0	2
GRP019-1	0	0	4	0	39	3	0	2
GRP020-1	0	0	4	0	39	3	0	2
GRP021-1	10	0	4	0	39	3	0	2
GRP022-1	40	0	7	0	638	4	0	72
GRP022-2	42420	0	13	0	370203	3	0	19440
GRP023-1	0	0	4	0	26	3	0	2
GRP023-2	0	0	5	0	35	3	0	6
GRP027-1	20	0	5	0	309	1	0	16
GRP027-2	20	0	5	0	291	1	0	16

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
GRP028-1	0	0	5	0	17	1	0	0
GRP028-2	10	0	5	0	114	1	0	3
GRP028-3	0	0	5	0	48	1	0	0
GRP029-1	39930	0	11	0	557883	1	1	60271
GRP029-2	14090	0	11	0	189579	1	1	19599
GRP030-1	3550	0	11	0	51458	3	1	4845
GRP031-1	50	0	8	0	845	1	0	83
GRP031-2	20	0	8	0	101	1	0	4
GRP032-3	0	0	5	0	23	3	0	1
GRP033-3	10	0	7	0	96	3	0	3
GRP033-4	10	0	7	0	98	3	0	3
GRP034-3	20	0	8	0	302	3	0	35
GRP034-4	10	0	8	0	61	3	0	0
GRP036-3	320	0	8	0	4879	11	0	336
GRP037-3	5870	0	9	0	97809	237	0	6181
GRP038-3	10	0	5	0	64	3	0	1
GRP041-2	0	0	4	0	9	3	0	0
GRP042-2	0	0	6	0	37	3	0	0
GRP043-2	20	0	8	0	218	5	0	6
GRP044-2	10	0	7	0	139	3	0	2
GRP045-2	190	0	10	0	1902	4	0	46
GRP046-2	50	0	9	0	482	4	0	13
GRP047-2	1280	0	12	0	12809	5	0	353
GRP123-6.003	163920	1	23	194606	1078090	171670	0	29318
GRP123-7.003	162960	1	23	194606	1078090	171670	0	29318
GRP123-8.003	163690	1	23	194606	1078090	171670	0	29318
GRP123-9.003	164300	1	23	194606	1078090	171670	0	29318
GRP124-6.003	439110	2	24	489046	3021700	486613	0	81185
GRP124-7.003	438430	2	24	489046	3021700	486613	0	81185
GRP124-8.003	436820	2	24	489046	3021700	486613	0	81185
GRP126-1.002	390	0	11	321	3520	170	0	194
GRP126-2.002	420	0	11	321	3635	170	0	194
GRP126-3.002	420	0	11	325	3894	191	0	194
GRP126-4.002	11630	0	11	7747	222498	2953	978	2747
GRP131-1.002	436960	12	77	461288	2402446	530039	0	104208
GRP131-2.002	443260	12	77	461288	2460739	530039	0	104208
GRP132-1.002	397650	12	77	415514	2267981	513017	0	95888
GRP132-2.002	404680	12	77	415514	2324229	513017	0	95888
GRP135-1.002	186870	8	36	256055	1103125	220875	0	93358
GRP135-2.002	190050	8	36	256055	1125020	220875	0	93358
GRP136-1	0	0	5	0	49	3	0	6
GRP137-1	0	0	5	0	49	3	0	6
GRP139-1	400	0	8	0	3041	4	0	177
GRP140-1	180150	0	11	0	1141322	3	16	35434
GRP142-1	0	0	4	0	21	3	0	2
GRP143-1	140	0	7	0	1107	4	0	87
GRP144-1	110	0	7	0	822	3	0	71
GRP145-1	20	0	6	0	194	4	0	20
GRP146-1	420	0	8	0	3041	4	0	177
GRP148-1	180620	0	11	0	1141322	3	16	35434
GRP150-1	10	0	5	0	98	4	0	6
GRP151-1	0	0	2	0	5	2	0	0
GRP152-1	320	0	8	0	2400	4	0	192
GRP153-1	10	0	5	0	73	3	0	8
GRP154-1	20	0	6	0	179	4	0	18
GRP155-1	30	0	6	0	180	4	0	18
GRP156-1	6460	0	9	0	44836	3	0	2009

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
GRP157-1	20	0	6	0	179	4	0	18
GRP158-1	20	0	6	0	180	4	0	18
GRP160-1	10	0	2	0	5	2	0	0
GRP161-1	0	0	2	0	5	2	0	0
GRP162-1	78370	0	11	0	517242	4	8	15625
GRP163-1	61390	0	11	0	408770	4	4	11406
GRP165-1	60900	0	11	0	413117	4	0	13828
GRP166-3	60960	0	11	0	413893	4	0	13828
GRP168-1	14190	0	10	0	92401	4	0	4056
GRP168-2	14500	0	10	0	92897	4	0	4071
GRP176-1	10	0	5	0	76	3	0	8
GRP176-2	10	0	5	0	76	3	0	8
GRP182-1	10	0	5	0	73	3	0	8
GRP182-2	10	0	5	0	76	3	0	8
GRP182-3	10	0	5	0	73	3	0	8
GRP182-4	10	0	5	0	76	3	0	8
GRP186-4	528120	0	12	0	3202449	6	24	85359
GRP188-1	300	0	8	0	2400	4	0	192
GRP188-2	370	0	8	0	2679	4	0	200
GRP189-1	10	0	5	0	73	3	0	8
GRP189-2	10	0	5	0	76	3	0	8
HEN001-1	0	0	3	0	12	3	0	0
HEN001-3	0	0	3	0	15	3	0	0
HEN001-5	0	0	2	0	6	2	0	0
HEN002-1	0	0	3	0	12	3	0	0
HEN002-2	0	0	3	0	12	3	0	0
HEN002-3	0	0	3	0	15	3	0	0
HEN002-4	0	0	3	0	15	3	0	0
HEN002-5	0	0	2	0	5	2	0	0
HEN003-3	13330	0	13	0	153293	450	792	11904
HEN003-4	14580	0	12	0	165694	599	898	11415
HEN003-5	10670	0	12	0	109533	275	425	6595
HEN004-4	619240	0	15	0	6241874	17135	49733	268739
HEN005-2	152190	0	14	0	2058757	23894	9986	213958
HEN006-4	2480	0	10	0	33649	128	52	2440
HEN007-2	15960	0	11	0	247471	868	1326	19211
HEN007-4	140	0	7	0	2142	8	0	213
HEN007-6	12160	0	11	0	184781	734	294	16884
HEN008-1	48960	0	14	0	688690	1249	1431	61718
HEN008-2	33830	0	11	0	518262	1354	2816	31543
HEN008-3	4850	0	12	0	57989	149	148	6398
HEN008-4	3520	0	10	0	51794	114	41	2931
HEN008-5	4890	0	11	0	50465	152	165	3965
HEN008-6	2650	0	11	0	31181	187	60	3029
HEN010-4	11600	0	11	0	165821	641	270	8872
HEN010-6	8750	0	11	0	112367	627	244	7348
HEN012-3	19170	0	14	0	215844	687	1140	16485
LCL006-1	7460	0	16	0	29475	4	6	1781
LCL007-1	0	0	4	0	11	3	0	0
LCL008-1	80	0	12	0	354	3	0	15
LCL009-1	13870	0	22	0	52477	4	2	1867
LCL010-1	540	0	18	0	2252	2	1	11
LCL011-1	34930	0	24	0	126910	2	3	1079
LCL013-1	10	0	6	0	20	2	0	0
LCL022-1	37770	0	24	0	140926	3	11	4788
LCL023-1	15450	0	22	0	58785	3	1	2080
LCL025-1	6140	0	16	0	24482	31	5	2377

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
LCL027-1	10	0	8	0	94	3	0	5
LCL029-1	174630	0	18	0	778993	16	133	71880
LCL033-1	430	0	16	0	1517	2	1	5
LCL035-1	110	0	14	0	393	2	2	1
LCL040-1	216750	0	20	0	105042	80	86	15840
LCL041-1	60	0	8	0	317	4	0	14
LCL043-1	20	0	6	0	49	3	0	1
LCL044-1	60	0	8	0	272	3	0	11
LCL045-1	5010	0	12	0	23010	17	2	975
LCL046-1	10	0	6	0	24	2	0	0
LCL064-1	5380	0	16	0	22673	18	3	2186
LCL064-2	1720	0	16	0	7197	18	2	1141
LCL065-1	87000	0	20	0	350842	86	195	33938
LCL066-1	4810	0	16	0	21015	4	3	2093
LCL069-1	8690	0	18	0	36915	2	0	2986
LCL072-1	8320	0	18	0	35400	2	0	2244
LCL076-1	106260	0	20	0	432451	31	158	44251
LCL076-2	0	0	4	0	13	3	0	0
LCL076-3	3930	0	12	0	27353	5	1	619
LCL077-1	18370	0	18	0	75830	7	24	7610
LCL077-2	310	0	10	0	2324	2	1	41
LCL079-1	10	0	8	0	62	3	0	6
LCL081-1	4440	0	22	0	17436	2	3	158
LCL082-1	330	0	16	0	1374	2	1	5
LCL083-1	1009240	0	32	0	5032544	3	400	61295
LCL083-2	7900	0	20	0	35615	3	7	6885
LCL087-1	80170	0	26	0	331146	2	2	573
LCL091-1	1110780	0	34	0	4201479	2	65	8071
LCL096-1	300	0	10	0	843	2	0	27
LCL097-1	2500	0	14	0	5073	2	0	208
LCL098-1	380	0	14	0	890	2	0	0
LCL101-1	52460	0	24	0	110030	2	1144	17506
LCL102-1	25380	0	16	0	70760	2	556	2320
LCL104-1	253560	0	24	0	605548	2	2	58928
LCL106-1	50	0	10	0	202	3	0	6
LCL107-1	6640	0	20	0	12552	2	0	0
LCL108-1	288340	0	28	0	446925	2	0	16
LCL110-1	142530	0	18	0	603717	18	39	26910
LCL111-1	700	0	12	0	3151	3	0	134
LCL112-1	1569860	0	20	0	6326814	141	847	281239
LCL117-1	20	0	10	0	112	2	0	5
LCL118-1	670	0	18	0	2801	4	0	332
LCL120-1	680	0	16	0	2275	2	0	106
LCL126-1	40	0	10	0	174	3	1	0
LCL130-1	1230	0	18	0	3092	2	25	33
LCL132-1	190230	0	14	0	1333086	9	394	63512
LCL143-1	53580	0	9	2226	1359660	27	3	7071
LCL169-1	0	0	3	0	8	3	0	0
LCL170-1	0	0	3	0	8	3	0	0
LCL171-1	0	0	3	0	8	3	0	0
LCL172-1	0	0	3	0	8	3	0	0
LCL173-1	0	0	3	0	8	3	0	0
LCL174-1	10	0	5	0	55	4	0	0
LCL175-1	0	0	3	0	8	3	0	0
LCL176-1	0	0	5	0	42	2	0	0
LCL177-1	20	0	7	0	104	3	0	6
LCL178-1	30	0	7	0	146	3	0	7

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
LCL181-2	0	0	3	0	15	3	0	0
LCL182-1	5280	0	13	0	26382	3	0	1570
LCL185-1	10	0	5	0	48	2	0	0
LCL186-1	10	0	5	0	52	2	0	0
LCL187-1	30	0	7	0	210	3	0	9
LCL188-1	30	0	7	0	178	3	0	8
LCL189-1	40	0	7	0	242	3	0	9
LCL190-1	10	0	5	0	59	4	0	0
LCL192-1	220	0	9	0	1248	6	0	75
LCL193-1	70	0	7	0	397	4	0	12
LCL194-1	270	0	9	0	1552	6	0	73
LCL195-1	6840	0	13	0	34459	7	0	1946
LCL196-1	42590	0	17	0	192202	5	0	14674
LCL197-1	50	0	7	0	332	5	0	13
LCL198-1	107120	0	17	0	471858	5	0	30446
LCL199-1	1380	0	13	0	6837	3	0	530
LCL200-1	320	0	11	0	1768	3	0	141
LCL201-1	2450	0	13	0	12473	3	0	878
LCL202-1	7120	0	15	0	33789	5	0	2520
LCL203-1	2650	0	13	0	13351	3	0	904
LCL204-1	2550	0	13	0	12589	3	0	878
LCL205-1	11310	0	15	0	52864	5	0	3464
LCL206-1	2740	0	13	0	13469	3	0	904
LCL207-1	570	0	11	0	3064	3	0	220
LCL208-1	21090	0	15	0	95165	4	0	6247
LCL210-1	63430	0	17	0	286478	5	0	18560
LCL211-1	860	0	11	0	4621	2	0	297
LCL212-1	10	0	5	0	40	4	0	0
LCL213-1	1920	0	11	0	9503	6	0	476
LCL214-1	1460	0	11	0	7561	6	0	414
LCL215-1	5990	0	13	0	30050	7	0	1772
LCL216-1	940	0	11	0	5128	6	0	314
LCL217-1	1520	0	11	0	8094	6	0	397
LCL218-1	5080	0	13	0	24933	3	0	1538
LCL226-1	10	0	5	0	51	4	0	0
LCL230-1	35150	0	15	0	152695	5	0	9716
LCL230-2	0	0	4	0	22	4	0	0
LCL231-1	52410	0	15	0	227983	5	0	12762
LDA003-1	81270	0	13	0	791773	25	0	27630
MSC001-1	105880	0	15	925	1376857	1261	726	49220
MSC002-1	370	0	12	0	3200	3	0	0
MSC002-2	320	0	12	0	2237	3	0	0
MSC003-1	30	1	10	5	129	34	0	0
MSC004-1	6810	3	22	934	33044	512	823	0
MSC005-1	30	0	10	0	267	34	0	0
MSC006-1	704750	7	19	2326452	3771696	80884	271403	182293
MSC007-2.002	73380	0	19	11826	846865	22963	0	130606
NUM001-1	540	0	8	0	5310	2	0	249
NUM002-1	230	0	8	0	2216	3	0	123
NUM003-1	120	0	8	0	1203	3	0	78
NUM004-1	190	0	8	0	1781	3	0	97
NUM009-1	820	1	5	4	33343	14	1	73
NUM014-1	10	1	6	1	48	6	0	3
NUM015-1	530	4	10	243	3837	336	30	437
NUM016-1	50	2	8	14	358	22	2	23
NUM016-2	40	2	9	10	220	26	0	0
NUM019-1	0	0	4	0	28	3	0	2

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
NUM022-1	60	1	9	6	622	12	0	55
NUM023-1	0	0	3	0	10	2	0	0
NUM024-1	7210	0	9	0	79242	2	8	6586
NUM025-1	0	0	4	0	23	3	0	0
NUM025-2	10	0	4	0	23	3	0	0
NUM027-1	13280	0	10	2244	201245	665	236	8873
NUM139-1	20	0	4	0	450	3	0	0
NUM180-1	4670	1	6	117	217067	18	1	376
NUM228-1	20	0	4	0	505	3	0	1
PLA001-1	2190	0	13	0	17638	1	0	0
PLA002-1	1380	0	12	12	12585	25	36	0
PLA003-1	100	0	8	0	731	1	0	81
PLA006-1	20	0	7	0	126	4	0	0
PLA007-1	121650	0	24	0	659625	42318	0	2171
PLA016-1	70350	0	23	0	377223	15520	0	1208
PLA017-1	90	0	10	0	634	26	0	0
PLA019-1	116080	0	24	0	612377	25636	0	2079
PLA020-1	10	0	5	0	31	2	0	0
PLA022-1	23760	0	16	0	111522	3321	0	2683
PLA022-2	820	0	16	0	5055	310	0	2
PRV003-1	10	0	4	0	247	3	0	2
PRV005-1	40	0	5	0	837	5	0	18
PRV006-1	240	0	6	49	5454	24	0	170
PRV007-1	897250	1	10	122511	15857206	66616	698	755401
PRV009-1	440	0	14	184	2952	222	4	9
PUZ001-1	10	1	11	5	111	39	0	0
PUZ002-1	20	0	12	0	78	12	0	0
PUZ003-1	20	0	12	0	119	15	0	12
PUZ004-1	0	0	11	0	69	11	0	2
PUZ008-1	0	0	3	0	16	2	0	0
PUZ008-2	50	0	13	0	309	13	0	91
PUZ009-1	40	3	6	53	994	81	0	218
PUZ011-1	10	0	6	0	42	6	0	0
PUZ012-1	510	0	18	664	3612	552	80	335
PUZ013-1	40	1	11	17	407	83	0	38
PUZ014-1	420	9	19	668	4037	1607	0	675
PUZ015-2.003	2110	2	19	1786	28021	6516	0	6848
PUZ016-2.003	600	2	11	228	8569	1084	0	1600
PUZ020-1	10	1	5	1	121	13	0	11
PUZ022-1	510	0	9	55	4062	417	0	189
PUZ023-1	118350	2	18	99928	1327022	20443	19735	18644
PUZ024-1	160	1	7	49	2311	7	31	27
PUZ025-1	1079090	4	15	1171477	14175383	266956	585603	251294
PUZ029-1	200	5	21	189	1326	874	0	62
PUZ031-1	91130	0	70	13646	498578	111969	0	6937
PUZ032-1	6460	4	9	5576	58060	1009	1686	1426
RNG001-3	34790	0	20	0	337012	5	31	15757
RNG001-4	23400	0	10	0	491511	2	201	23516
RNG001-5	948790	0	13	0	17609610	124	4612	719851
RNG002-1	5990	0	10	0	123899	5	0	6928
RNG003-1	6650	0	10	0	139620	5	0	7729
RNG005-2	30	0	6	0	653	3	0	13
RNG006-1	60	0	7	0	1224	3	0	42
RNG006-2	70	0	7	0	1387	3	0	8
RNG011-5	0	0	2	0	5	2	0	0
RNG023-6	60	0	7	0	541	3	0	57
RNG023-7	60	0	7	0	541	3	0	57

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
RNG024-6	60	0	7	0	541	3	0	57
RNG024-7	70	0	7	0	541	3	0	57
RNG037-2	60	0	7	0	1258	4	0	28
RNG038-1	662000	0	12	0	12728817	833	289	768677
RNG038-2	40	0	7	0	617	3	0	7
RNG040-1	380	0	6	65	13698	3	0	131
RNG040-2	116300	0	9	15182	3639675	147	4	46604
RNG041-1	5170	0	8	610	171681	75	7	2008
ROB010-1	48240	0	13	0	302839	3	16	20298
ROB013-1	3230	0	11	0	21226	3	0	1849
ROB016-1	1430	0	9	0	10624	5	0	943
ROB021-1	23810	0	11	0	193673	4	0	16435
SET001-1	10	0	5	0	24	4	0	0
SET003-1	10	1	5	1	123	4	0	1
SET004-1	0	1	5	1	123	4	0	1
SET006-1	10	1	5	2	126	5	0	1
SET008-1	390	1	7	318	7326	32	45	148
SET009-1	15230	2	12	16920	183088	3810	4259	3774
SET012-1	174450	2	12	89713	3368499	3916	13004	72378
SET024-3	4870	0	7	42	275187	83	5	149
SET024-4	4670	0	7	34	261721	73	4	147
SET024-6	70	0	5	12	1924	3	0	9
SET024-7	100	0	5	14	3165	3	0	12
SET025-3	30	0	5	0	616	3	0	15
SET025-4	30	0	5	0	585	3	0	13
SET025-6	20	0	4	0	537	3	1	2
SET025-7	30	0	4	0	825	3	1	2
SET027-3	332440	1	7	473	25430214	59	16	517
SET027-4	46230	1	7	203	3442697	57	12	414
SET027-6	19910	0	8	190	1297074	24	1	593
SET027-7	20380	0	8	190	1313517	54	1	731
SET043-5	10	2	3	6	31	6	0	0
SET044-5	50	2	5	34	541	13	0	68
SET045-5	10	2	4	5	63	3	0	8
SET046-5	170	5	5	228	1525	138	19	140
SET050-6	3050	0	6	34	199384	6	1	80
SET051-6	3160	0	6	46	202381	6	1	110
SET052-6	90	0	5	4	4846	4	1	7
SET053-6	100	0	5	10	5215	4	1	9
SET054-6	0	0	3	0	33	3	0	0
SET054-7	0	0	3	0	33	3	0	0
SET055-6	3020	2	6	9	199852	14	4	25
SET055-7	0	0	4	0	29	4	0	1
SET056-6	340	0	6	0	11015	10	0	61
SET056-7	360	0	6	0	11289	11	0	76
SET057-6	300	0	6	18	8297	10	0	74
SET057-7	330	0	6	18	9441	10	0	90
SET058-6	290	0	6	18	8369	12	0	74
SET058-7	330	0	6	18	9513	12	0	90
SET059-6	280	0	6	26	8757	10	0	88
SET059-7	330	0	6	26	10923	10	0	102
SET060-6	80	1	4	19	2788	4	0	16
SET060-7	100	1	4	19	3688	4	0	19
SET061-7	160	0	6	18	5256	4	1	43
SET062-7	0	0	3	0	21	3	0	0
SET063-7	10	0	4	0	233	4	0	0
SET064-7	50	0	4	0	3983	4	0	2

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SET065-6	20	0	5	0	603	2	2	5
SET065-7	30	0	5	0	840	2	2	5
SET073-7	2670	0	5	12	192317	2	0	13
SET074-7	2690	0	5	20	192983	2	0	14
SET075-7	5480	0	5	38	389170	4	0	55
SET077-6	20	0	4	0	461	3	1	2
SET077-7	30	0	4	0	643	3	1	2
SET078-6	40	0	5	0	1012	4	1	6
SET078-7	0	0	3	0	32	3	0	0
SET079-7	3150	0	5	69	211733	10	0	78
SET080-7	10	0	3	0	94	3	0	0
SET081-6	1950	0	7	206	93040	15	0	230
SET081-7	360	0	6	24	12684	5	0	68
SET090-7	1250	1	4	90	65318	15	4	100
SET093-6	296450	0	6	22	20664988	4	2	375
SET093-7	2850	0	4	0	203801	3	0	3
SET095-7	5080	1	5	243	268897	9	0	266
SET098-7	300	0	5	0	12923	4	0	26
SET101-6	11000	0	7	18	269322	28	20	1244
SET101-7	180	0	5	0	4864	10	2	27
SET102-6	140	0	5	0	3237	8	0	23
SET102-7	180	0	5	0	4468	10	2	28
SET108-6	7190	1	5	257	462897	19	0	222
SET108-7	28510	1	5	639	1806174	56	18	571
SET117-6	276560	0	6	22	19239807	4	2	372
SET117-7	3020	0	4	0	217483	3	0	3
SET118-6	0	0	4	0	199	3	0	0
SET118-7	10	0	4	0	601	3	0	0
SET152-6	3510	1	5	5	249434	5	0	31
SET153-6	3610	1	5	21	251871	5	0	46
SET158-6	3280	0	5	12	246551	4	0	35
SET196-6	20	1	4	3	395	5	0	2
SET197-6	20	1	4	3	415	5	0	2
SET231-6	10	0	4	0	158	4	0	0
SET232-6	415990	1	7	597	28060815	46	44	2764
SET233-6	419390	1	7	597	28060825	46	44	2764
SET234-6	54690	0	7	674	3252402	36	30	1553
SET239-6	280	0	6	18	8205	6	0	65
SET240-6	3360	0	7	222	137716	17	1	386
SET241-6	3410	0	7	222	137726	17	1	386
SET242-6	5060	0	6	141	328254	6	0	178
SET252-6	3140	1	6	117	119460	18	1	415
SET253-6	3240	1	6	117	120556	18	1	449
SET296-6	0	0	2	0	8	2	0	0
SET451-6	5860	1	6	117	170730	18	1	660
SET479-6	9870	0	6	473	640674	60	2	376
SET553-6	3600	1	6	117	129008	18	2	447
SET563-6	190490	0	6	13	13581314	4	0	237
SYN003-1.006	840	0	17	158	8759	3637	0	0
SYN004-1.007	15450	15	42	34631	101046	96943	0	0
SYN005-1.010	20	0	11	0	211	2	0	0
SYN006-1	20	0	7	0	115	1	3	0
SYN007-1.002	20	2	5	13	431	36	0	16
SYN008-1	0	0	4	0	35	4	0	0
SYN009-1	10	0	7	0	121	6	0	0
SYN011-1	100	4	11	91	938	392	0	0
SYN014-2	50	0	5	0	1347	8	0	22

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN028-1	10	2	6	4	56	11	0	0
SYN029-1	10	2	6	4	40	17	0	1
SYN030-1	20	5	14	27	243	100	0	36
SYN031-1	40	2	5	24	460	17	0	14
SYN032-1	50	8	20	92	373	293	0	0
SYN033-1	10	0	5	0	14	1	0	0
SYN034-1	130	6	7	143	923	184	0	76
SYN035-1	10	0	9	0	27	1	14	0
SYN040-1	10	0	3	0	15	3	0	0
SYN041-1	0	0	2	0	6	2	0	0
SYN044-1	10	3	7	18	176	51	0	31
SYN045-1	0	0	5	0	47	11	0	0
SYN046-1	0	0	3	0	11	3	0	0
SYN047-1	10	1	6	1	108	25	0	0
SYN048-1	0	0	2	0	4	1	0	0
SYN049-1	0	0	3	0	11	1	2	0
SYN050-1	10	0	4	0	29	2	0	0
SYN051-1	10	2	5	6	57	17	0	0
SYN052-1	10	2	5	12	179	25	7	12
SYN053-1	20	2	5	23	282	47	0	0
SYN054-1	30	3	10	22	158	71	0	0
SYN055-1	40	2	11	33	223	77	0	7
SYN057-1	10	0	9	6	95	23	0	0
SYN058-1	10	0	6	0	58	8	0	0
SYN060-1	0	1	4	1	28	5	0	0
SYN061-1	0	0	5	0	14	4	0	0
SYN062-1	10	1	6	2	102	18	0	3
SYN063-1	0	0	5	0	102	5	0	0
SYN063-2	0	0	3	0	11	3	0	0
SYN064-1	0	0	2	0	4	1	0	0
SYN065-1	0	0	4	0	15	1	0	0
SYN066-1	0	0	5	0	21	1	0	0
SYN068-1	10	1	5	1	36	5	0	0
SYN069-1	620	5	17	423	3455	1315	0	142
SYN070-1	53110	10	11	117132	265876	120224	0	15004
SYN073-1	10	0	3	0	15	1	0	0
SYN079-1	0	0	4	0	9	3	0	0
SYN080-1	0	0	3	0	15	3	0	0
SYN081-1	20	2	5	8	118	1	19	9
SYN082-1	13200	8	10	23186	162225	21653	0	18095
SYN083-1	10	0	4	0	23	3	0	2
SYN084-2	7220	17	18	5274	32087	13035	0	1305
SYN085-1.010	0	0	12	0	33	12	0	0
SYN088-1.010	0	0	12	0	33	12	0	0
SYN089-1.002	10	0	4	0	17	4	0	0
SYN095-1.002	0	0	4	0	17	2	0	0
SYN098-1.002	190	10	22	448	1821	990	0	48
SYN103-1	0	0	2	0	3	2	0	0
SYN104-1	0	0	2	0	3	2	0	0
SYN105-1	0	0	3	0	6	2	0	0
SYN106-1	0	0	3	0	6	2	0	0
SYN107-1	10	0	4	0	43	1	0	0
SYN108-1	0	0	4	0	48	1	0	0
SYN109-1	30	0	10	0	323	19	0	27
SYN110-1	40	0	12	0	257	20	4	3
SYN111-1	30	0	10	0	233	20	0	2
SYN112-1	10	0	4	0	48	1	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN113-1	20	0	9	0	170	11	0	10
SYN114-1	10	0	5	0	426	1	0	0
SYN115-1	210	0	14	0	1868	79	3	12
SYN116-1	2600	0	12	0	48970	303	93	16
SYN117-1	6570	0	13	0	127038	619	206	74
SYN118-1	0	0	2	0	3	2	0	0
SYN119-1	0	0	2	0	3	2	0	0
SYN120-1	10	0	3	0	8	2	0	0
SYN121-1	30	0	13	0	136	23	0	2
SYN122-1	0	0	10	0	61	18	0	0
SYN123-1	10	0	6	0	83	2	3	0
SYN124-1	70	0	7	0	1651	13	0	1
SYN125-1	10	0	7	0	47	12	0	0
SYN126-1	10	0	6	0	163	3	0	0
SYN127-1	10	0	6	0	185	3	0	0
SYN128-1	91660	0	16	0	1678489	17717	2597	3938
SYN129-1	91940	0	16	0	1678489	17717	2597	3938
SYN130-1	0	0	2	0	3	1	0	0
SYN131-1	0	0	2	0	3	1	0	0
SYN132-1	10	0	2	0	3	1	0	0
SYN133-1	0	0	2	0	3	1	0	0
SYN134-1	0	0	5	0	16	4	0	0
SYN135-1	10	0	5	0	275	2	0	0
SYN136-1	10	0	5	0	271	2	0	0
SYN137-1	15300	0	23	0	87556	4809	941	248
SYN138-1	130	0	19	0	599	69	8	14
SYN139-1	16860	0	35	0	103771	8395	918	261
SYN140-1	16860	0	35	0	103838	8394	911	265
SYN141-1	310	0	15	0	3161	93	5	14
SYN144-1	40	0	13	0	309	30	0	0
SYN145-1	0	0	2	0	3	1	0	0
SYN146-1	0	0	3	0	25	1	0	0
SYN147-1	0	0	3	0	24	2	0	0
SYN148-1	10	0	9	0	62	11	0	0
SYN149-1	10	0	3	0	24	1	0	0
SYN150-1	0	0	4	0	36	2	0	0
SYN151-1	0	0	4	0	33	2	0	0
SYN152-1	0	0	4	0	36	2	0	0
SYN153-1	10	0	5	0	185	3	0	0
SYN154-1	10	0	5	0	224	3	0	0
SYN157-1	10360	0	14	0	143905	2785	681	4782
SYN158-1	5380	0	14	0	82968	1555	66	615
SYN159-1	5130	0	14	0	82968	1555	66	615
SYN160-1	17910	0	14	0	292100	5173	567	8469
SYN161-1	8240	0	14	0	123836	2247	147	3408
SYN162-1	8200	0	14	0	123836	2248	147	3408
SYN164-1	0	0	2	0	3	2	0	0
SYN165-1	0	0	3	0	29	1	0	0
SYN166-1	10	0	3	0	29	1	0	0
SYN167-1	0	0	3	0	29	1	0	0
SYN168-1	10	0	5	0	153	2	0	0
SYN169-1	10	0	5	0	140	2	1	0
SYN170-1	40	0	6	0	857	1	0	0
SYN171-1	44680	0	33	0	422524	26985	6853	40226
SYN172-1	0	0	2	0	3	2	0	0
SYN173-1	0	0	4	0	11	2	0	0
SYN174-1	10	0	4	0	11	2	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN175-1	10	0	4	0	88	2	1	0
SYN176-1	420	0	10	0	3515	125	2	0
SYN177-1	40	0	7	0	525	12	0	13
SYN178-1	4670	0	15	0	32191	690	366	16
SYN181-1	2240	0	15	0	17730	417	181	87
SYN182-1	10	0	7	0	202	3	0	1
SYN183-1	10	0	7	0	270	4	0	1
SYN184-1	10	0	2	0	3	2	0	0
SYN185-1	0	0	2	0	3	2	0	0
SYN186-1	10	0	6	0	47	5	0	0
SYN187-1	0	0	6	0	77	5	0	0
SYN188-1	0	0	6	0	71	5	0	0
SYN189-1	0	0	8	0	54	7	0	0
SYN190-1	186940	0	24	0	1313898	42444	9030	817
SYN191-1	40	0	10	0	258	23	0	0
SYN192-1	40	0	13	0	526	20	0	0
SYN193-1	50	0	13	0	686	20	0	0
SYN194-1	310	0	15	0	3163	99	5	14
SYN195-1	120	0	19	0	1513	44	12	13
SYN196-1	0	0	5	0	22	2	0	0
SYN197-1	0	0	3	0	7	3	0	0
SYN198-1	10	0	5	0	19	2	0	0
SYN199-1	0	0	5	0	28	2	0	0
SYN200-1	0	0	5	0	22	2	0	0
SYN201-1	10	0	9	0	150	20	0	0
SYN202-1	25030	0	18	0	148047	6919	1389	328
SYN203-1	90	0	12	0	1318	54	2	0
SYN204-1	16750	0	26	0	103340	8266	910	231
SYN205-1	16680	0	26	0	103340	8266	910	231
SYN206-1	20	0	14	0	293	39	0	0
SYN207-1	15300	0	20	0	86475	4593	913	235
SYN208-1	50	0	12	0	337	17	1	1
SYN209-1	30	0	8	0	459	3	0	0
SYN210-1	10	0	8	0	175	3	0	0
SYN211-1	20	0	8	0	183	3	3	0
SYN212-1	20	0	8	0	263	3	0	0
SYN213-1	301180	0	34	0	3218078	76765	20681	220862
SYN215-1	550	0	20	0	4468	197	52	136
SYN216-1	0	0	6	0	35	2	0	0
SYN217-1	20	0	7	0	70	10	0	2
SYN218-1	10	0	6	0	37	1	0	3
SYN219-1	30	0	13	0	136	23	0	2
SYN220-1	30	0	6	0	643	4	0	0
SYN221-1	80	0	7	0	1375	12	0	0
SYN222-1	80	0	7	0	1620	8	3	0
SYN223-1	20	0	6	0	527	6	0	0
SYN224-1	70	0	7	0	1651	12	0	1
SYN225-1	0	0	7	0	41	8	0	0
SYN226-1	30	0	6	0	632	8	0	0
SYN227-1	10	0	7	0	39	10	0	0
SYN228-1	10	0	6	0	34	8	0	0
SYN229-1	80	0	7	0	1487	9	0	0
SYN230-1	80	0	7	0	1487	10	0	0
SYN231-1	80	0	7	0	1577	6	3	0
SYN232-1	70	0	7	0	1566	7	3	0
SYN233-1	30	0	6	0	566	7	0	0
SYN234-1	10	0	7	0	46	9	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN235-1	10	0	7	0	43	11	0	0
SYN236-1	30	0	6	0	558	9	0	0
SYN237-1	0	0	3	0	30	1	0	0
SYN238-1	10	0	3	0	29	1	0	0
SYN239-1	0	0	3	0	29	1	0	0
SYN240-1	0	0	3	0	29	1	0	0
SYN241-1	0	0	3	0	29	2	0	0
SYN242-1	0	0	3	0	27	3	0	0
SYN243-1	10	0	4	0	13	3	0	0
SYN244-1	0	0	3	0	30	1	0	0
SYN245-1	0	0	3	0	30	1	0	0
SYN246-1	0	0	4	0	22	4	0	0
SYN247-1	0	0	3	0	28	1	0	0
SYN248-1	10	0	7	0	64	6	0	0
SYN249-1	10	0	7	0	60	8	0	0
SYN250-1	160	0	14	0	891	152	9	0
SYN251-1	10	0	4	0	13	3	0	0
SYN253-1	17120	0	35	0	103838	8394	911	265
SYN254-1	17140	0	36	0	103819	8418	910	231
SYN255-1	10	0	4	0	38	2	0	0
SYN256-1	0	0	4	0	36	2	0	0
SYN257-1	0	0	2	0	3	2	0	0
SYN258-1	0	0	3	0	29	1	0	0
SYN259-1	0	0	3	0	28	1	0	0
SYN260-1	0	0	3	0	28	1	0	0
SYN261-1	0	0	3	0	29	1	0	0
SYN262-1	10	0	6	0	47	5	0	0
SYN263-1	20	0	7	0	124	5	3	0
SYN264-1	10	0	7	0	49	7	0	0
SYN265-1	10	0	5	0	144	2	0	0
SYN266-1	1600	0	13	0	17714	379	32	531
SYN267-1	40	0	6	0	900	1	0	0
SYN268-1	40	0	6	0	901	1	0	0
SYN270-1	40	0	12	0	280	33	0	0
SYN271-1	44300	0	33	0	422473	26984	6853	40226
SYN272-1	110	0	11	0	787	45	6	11
SYN273-1	190	0	12	0	1464	56	0	6
SYN274-1	0	0	2	0	3	1	0	0
SYN275-1	0	0	2	0	3	1	0	0
SYN276-1	0	0	2	0	3	2	0	0
SYN277-1	0	0	3	0	32	1	0	0
SYN278-1	0	0	3	0	33	1	0	0
SYN279-1	0	0	4	0	17	1	0	0
SYN280-1	0	0	3	0	35	1	0	0
SYN281-1	0	0	3	0	32	1	0	0
SYN282-1	0	0	3	0	34	1	0	0
SYN283-1	10	0	4	0	11	2	0	0
SYN284-1	0	0	4	0	11	2	0	0
SYN285-1	10	0	7	0	50	9	0	0
SYN286-1	0	0	4	0	15	1	0	0
SYN287-1	0	0	3	0	35	1	0	0
SYN288-1	10	0	3	0	34	2	0	0
SYN289-1	0	0	4	0	11	2	0	0
SYN290-1	0	0	3	0	32	1	0	0
SYN291-1	0	0	3	0	34	1	0	0
SYN292-1	0	0	3	0	8	3	0	0
SYN293-1	10	0	6	0	37	8	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN294-1	0	0	6	0	43	10	0	0
SYN295-1	10	0	3	0	32	2	0	0
SYN296-1	10	0	4	0	11	2	0	0
SYN297-1	0	0	3	0	32	2	0	0
SYN298-1	20	0	14	0	293	38	1	0
SYN299-1	20	0	14	0	293	39	0	0
SYN300-1	20	0	14	0	293	39	0	0
SYN301-1	10	0	8	0	161	3	0	0
SYN315-1	20	2	5	10	121	20	1	9
SYN318-1	0	0	3	0	20	2	1	0
SYN319-1	10	0	5	0	150	1	0	0
SYN321-1	20	2	5	16	171	23	0	12
SYN323-1	20	2	5	18	167	38	0	10
SYN325-1	10	0	4	0	42	1	0	0
SYN326-1	10	0	5	0	66	1	5	0
SYN327-1	290	3	5	320	3804	358	0	270
SYN331-1	40	1	4	1	323	1	3	11
SYN333-1	60	0	9	0	193	1	84	0
SYN336-1	0	0	2	0	10	1	0	0
SYN338-1	0	0	2	0	6	1	0	0
SYN339-1	0	0	2	0	4	1	0	0
SYN340-1	0	0	2	0	4	1	0	0
SYN341-1	0	0	2	0	4	1	0	0
SYN343-1	0	2	3	5	30	6	0	0
SYN345-1	1820	17	15	3642	12178	4620	20	3237
SYN346-1	0	1	2	1	14	1	0	0
SYN349-1	18740	14	9	26104	71744	1	26643	26884
SYN350-1	870	6	5	893	9048	346	351	856
SYN352-1	11230	6	7	7210	122041	2885	3365	1719
SYN354-1	550	4	10	1052	4079	684	176	853
TOP001-2	42610	4	13	28806	338741	3998	6576	14914
TOP002-2	0	0	3	0	6	3	0	0
TOP004-1	0	0	2	0	47	1	0	0
TOP004-2	10	0	2	0	22	1	0	0
TOP005-2	529340	7	13	400771	4300361	55211	82205	36732

Tabla 18: Resultados temporales y espaciales del SLT vía ALL

A continuación se incluye la tabla con los resultados obtenidos por el procedimiento SLT vía POS.

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
BOO003-1	15890	0	10	0	358232	4	0	0
BOO003-2	1263080	0	13	0	3332059	4	16	131492
BOO003-4	356420	0	13	0	3336619	4	0	0
BOO004-1	10990	0	10	0	254802	4	0	0
BOO004-2	1190540	0	13	0	2837456	4	0	113174
BOO004-4	372050	0	13	0	3409756	4	0	0
BOO005-1	14550	0	10	0	328484	4	0	0
BOO006-1	19470	0	10	0	439600	4	0	0
BOO012-1	1012200	0	12	0	5580542	9	0	306779
CAT001-3	35780	0	9	1275	898355	305	48	4193
CAT001-4	5640	0	9	0	85411	96	0	0
CAT002-3	6900	0	8	158	180912	119	12	629
CAT002-4	2270	0	8	0	40822	63	0	0
CAT003-3	840	0	7	14	21268	19	0	81

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
CAT003-4	290	0	7	0	4665	8	0	0
CAT004-3	303780	0	10	15290	7680584	4129	591	36384
CAT004-4	44390	0	10	0	751176	1839	0	0
CAT005-1	393350	0	13	0	6435311	53415	3800	0
CAT006-1	403350	0	13	0	6511714	55947	3804	0
CAT007-1	220	0	7	0	4274	10	0	0
CAT007-3	0	0	4	0	21	4	0	0
CAT011-1	40	0	6	0	614	4	0	0
CAT011-2	80	0	7	0	866	4	0	0
CAT012-1	30	0	6	0	474	6	0	0
CAT012-3	1050	0	8	14	24690	39	0	91
CAT012-4	370	0	8	0	5741	25	0	0
CAT013-1	30	0	6	0	451	4	0	0
CAT013-3	500	0	7	4	11194	14	0	22
CAT013-4	260	0	7	0	4062	7	0	0
CAT014-1	40	0	6	0	637	6	0	0
CAT014-2	290	0	8	0	3140	4	0	0
CAT016-3	130	0	6	0	2587	2	0	0
CAT016-4	80	0	6	0	1390	2	0	0
CAT017-3	140	0	6	0	2770	2	0	0
CAT017-4	170	0	7	0	2765	3	0	0
COL001-1	847310	0	12	0	6192105	1	6	0
COL001-2	440	0	7	0	3358	1	0	0
COL002-1	70	0	6	0	573	1	0	0
COL002-2	190	0	8	0	1132	1	0	0
COL002-3	50	0	7	0	265	1	0	0
COL007-1	0	0	2	0	5	1	0	0
COL008-1	0	0	4	0	41	1	0	0
COL009-1	240	0	7	0	2075	1	0	0
COL010-1	10	0	4	0	41	1	0	0
COL011-1	930870	0	12	0	6397939	1	33	0
COL012-1	10	0	2	0	5	1	0	0
COL013-1	0	0	2	0	5	1	0	0
COL014-1	0	0	2	0	5	1	0	0
COL015-1	10	0	4	0	37	1	0	0
COL016-1	0	0	2	0	5	1	0	0
COL017-1	0	0	4	0	41	1	0	0
COL018-1	0	0	2	0	5	1	0	0
COL019-1	50	0	6	0	517	1	0	0
COL020-1	60	0	6	0	502	1	0	0
COL021-1	0	0	4	0	41	1	0	0
COL022-1	0	0	4	0	41	1	0	0
COL023-1	230	0	7	0	1987	1	0	0
COL024-1	10	0	4	0	41	1	0	0
COL025-1	0	0	4	0	41	1	0	0
COL026-1	230	0	7	0	1987	1	0	0
COL027-1	220	0	7	0	1988	1	0	0
COL028-1	230	0	7	0	1987	1	0	0
COL029-1	0	0	2	0	5	1	0	0
COL030-1	680	0	8	0	4839	1	0	0
COL031-1	10	0	5	0	85	1	0	0
COL032-1	17210	0	10	0	120995	1	0	0
COL033-1	930370	0	12	0	5447422	11	20	0
COL035-1	5970	0	9	0	38205	1	0	0
COL039-1	260	0	7	0	1895	1	0	0
COL040-1	727710	0	12	0	4262022	1	53	0
COL044-1	734830	0	12	0	4262022	1	53	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
COL045-1	0	0	4	0	44	1	0	0
COL048-1	0	0	4	0	41	1	0	0
COL050-1	0	0	5	0	68	1	0	0
COL051-1	90	0	7	0	846	1	0	0
COL052-1	28700	0	10	0	231354	1	0	0
COL052-2	37630	0	11	0	332580	9	0	0
COL053-1	10	0	4	0	35	1	0	0
COL054-1	250	0	7	0	2507	1	0	0
COL055-1	0	0	3	0	15	2	0	0
COL056-1	370	0	7	0	3284	1	0	0
COL058-1	310	0	8	0	2599	1	0	0
COL070-1	230	0	7	0	1917	1	0	0
COL075-2	104990	0	11	0	705287	1	0	0
COM001-1	10	0	7	0	62	4	0	0
COM002-1	170	0	13	0	933	4	0	0
COM002-2	190	0	13	0	933	4	0	0
COM003-2	150	1	8	45	1661	35	10	64
COM004-1	670	0	10	0	7504	2	0	0
GEO002-3	0	0	3	0	45	3	0	0
GEO003-1	60	0	6	0	1304	4	0	0
GEO003-2	40	0	6	0	669	2	0	0
GEO003-3	0	0	3	0	27	2	0	0
GEO011-2	0	0	5	0	69	5	0	3
GEO011-3	20	0	5	0	516	5	0	8
GEO011-4	0	0	5	0	78	5	0	3
GEO011-5	0	0	5	0	66	5	0	3
GEO014-2	0	0	4	0	19	3	0	0
GEO015-2	30	0	6	0	491	3	0	0
GEO015-3	0	0	4	0	52	3	0	0
GEO016-2	0	0	4	0	25	3	0	0
GEO016-3	0	0	4	0	35	3	0	0
GEO017-2	510	0	8	0	10568	4	0	0
GEO017-3	20	0	5	0	312	3	0	0
GEO018-2	30	0	6	0	448	3	0	0
GEO018-3	50	0	6	0	1212	3	0	0
GEO019-2	10	0	4	0	39	3	0	0
GEO019-3	10	0	4	0	74	3	0	0
GEO021-2	20	0	6	0	363	4	0	0
GEO021-3	40	0	5	0	890	4	0	0
GEO022-2	520	0	8	0	10588	4	0	0
GEO022-3	40	0	5	0	1390	3	0	0
GEO024-2	60	0	6	0	1107	2	0	0
GEO024-3	50	0	5	0	2417	2	0	0
GEO027-3	2380	0	7	0	65468	9	0	0
GEO035-2	0	0	3	0	26	2	0	0
GEO035-3	0	0	3	0	26	2	0	0
GEO038-2	10	0	5	0	205	3	0	0
GEO038-3	10	0	5	0	256	3	0	0
GEO039-3	3120	0	7	2	122678	5	4	10
GEO041-3	30710	0	6	14	1396991	10	2	186
GEO047-3	41770	0	7	18	1821518	38	78	215
GEO053-3	135340	0	7	30	5966527	72	53	443
GEO054-2	10	0	5	0	207	3	0	0
GEO054-3	0	0	3	0	27	2	0	0
GEO055-2	20	0	5	0	211	3	0	0
GEO055-3	60	0	5	0	2198	3	0	0
GEO056-2	12190	0	8	16	235796	2	0	18

GEO056-3	40	0	5	0	902	3	0	0
GEO057-2	70	0	6	0	1438	2	0	0
GEO057-3	0	0	3	0	39	2	0	0
GEO058-2	348910	0	9	62	6338549	2	0	209
GEO058-3	410	0	6	0	11187	4	0	0
GEO059-3	530	0	6	0	19273	4	0	0
GEO064-3	10	0	4	0	181	4	0	0
GEO065-3	0	0	4	0	273	4	0	0
GEO066-3	10	0	4	0	365	4	0	0
GRP001-1	45700	0	11	0	711719	3	3	0
GRP001-5	1140	0	11	0	10104	3	0	0
GRP003-1	30	0	11	0	245	3	0	0
GRP003-2	930	0	11	0	9519	3	0	0
GRP004-1	20	0	8	0	120	1	0	0
GRP004-2	40	0	8	0	474	3	0	0
GRP005-1	0	0	5	0	12	3	0	0
GRP006-1	0	0	8	0	42	3	0	0
GRP007-1	10	0	4	0	60	3	0	0
GRP009-1	7520	0	10	0	119989	5	0	0
GRP010-1	180	0	8	0	2878	3	0	0
GRP012-1	330	0	8	0	5525	3	0	0
GRP012-2	1178420	0	13	0	3510010	4	170	230301
GRP013-1	4310	0	9	0	79234	3	0	0
GRP017-1	80	0	7	0	1200	4	0	0
GRP018-1	10	0	4	0	45	3	0	0
GRP019-1	0	0	4	0	45	3	0	0
GRP020-1	10	0	4	0	45	3	0	0
GRP021-1	10	0	4	0	45	3	0	0
GRP022-1	100	0	7	0	1402	4	0	0
GRP022-2	168290	0	13	0	1519000	3	0	0
GRP023-1	0	0	4	0	32	3	0	0
Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
GRP023-2	0	0	5	0	51	3	0	0
GRP027-1	20	0	5	0	405	1	0	0
GRP027-2	20	0	5	0	387	1	0	0
GRP028-1	10	0	5	0	17	1	0	0
GRP028-2	10	0	5	0	129	1	0	0
GRP028-3	10	0	5	0	48	1	0	0
GRP029-1	115790	0	11	0	1688353	1	2	0
GRP029-2	48630	0	11	0	692339	1	2	0
GRP030-1	12650	0	11	0	199211	3	2	0
GRP031-1	120	0	8	0	1729	1	0	0
GRP031-2	20	0	8	0	109	1	0	0
GRP032-3	0	0	5	0	25	3	0	0
GRP033-3	10	0	7	0	111	3	0	0
GRP033-4	10	0	7	0	113	3	0	0
GRP034-3	40	0	8	0	502	3	0	0
GRP034-4	10	0	8	0	61	3	0	0
GRP036-3	630	0	8	0	10116	25	0	0
GRP037-3	25080	0	10	0	429745	1087	0	0
GRP038-3	10	0	5	0	66	3	0	0
GRP041-2	0	0	4	0	9	3	0	0
GRP042-2	10	0	6	0	37	3	0	0
GRP043-2	30	0	8	0	240	5	0	0
GRP044-2	10	0	7	0	145	3	0	0
GRP045-2	210	0	10	0	2219	4	0	0
GRP046-2	50	0	9	0	554	4	0	0
GRP047-2	1730	0	12	0	17676	5	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
GRP123-6.003	175150	1	23	183261	1149750	159753	0	24155
GRP123-7.003	175420	1	23	183261	1149750	159753	0	24155
GRP123-8.003	173820	1	23	183261	1149750	159753	0	24155
GRP123-9.003	175670	1	23	183261	1149750	159753	0	24155
GRP124-6.003	476040	0	26	458014	3287672	457485	0	68513
GRP124-7.003	472110	0	26	458014	3287672	457485	0	68513
GRP124-8.003	472790	0	26	458014	3287672	457485	0	68513
GRP126-1.002	520	0	11	399	4241	208	51	188
GRP126-2.002	570	0	11	399	4389	208	51	188
GRP126-3.002	550	0	11	403	4685	230	51	188
GRP126-4.002	13070	0	11	8111	240585	3469	1605	1760
GRP131-1.002	501130	12	77	458571	3287252	544602	0	104208
GRP131-2.002	528130	12	77	458571	3432626	544602	0	104208
GRP132-1.002	469340	12	77	425757	3109651	532348	0	95888
GRP132-2.002	487810	12	77	425757	3250250	532348	0	95888
GRP136-1	10	0	5	0	65	3	0	0
GRP137-1	0	0	5	0	65	3	0	0
GRP139-1	670	0	8	0	5160	4	0	0
GRP140-1	296820	0	11	0	1996385	5	16	0
GRP142-1	0	0	4	0	25	3	0	0
GRP143-1	210	0	7	0	1728	4	0	0
GRP144-1	160	0	7	0	1322	3	0	0
GRP145-1	40	0	6	0	324	4	0	0
GRP146-1	640	0	8	0	5160	4	0	0
GRP148-1	298260	0	11	0	1996385	5	16	0
GRP150-1	20	0	5	0	126	4	0	0
GRP151-1	0	0	2	0	5	2	0	0
GRP152-1	610	0	8	0	4998	4	0	0
GRP153-1	10	0	5	0	93	3	0	0
GRP154-1	30	0	6	0	259	4	0	0
GRP155-1	30	0	6	0	260	4	0	0
GRP156-1	10010	0	9	0	75795	3	0	0
GRP157-1	30	0	6	0	259	4	0	0
GRP158-1	30	0	6	0	260	4	0	0
GRP160-1	0	0	2	0	5	2	0	0
GRP161-1	0	0	2	0	5	2	0	0
GRP162-1	131200	0	11	0	893022	6	8	0
GRP163-1	105050	0	11	0	727126	6	4	0
GRP165-1	124650	0	11	0	881275	4	0	0
GRP166-3	123020	0	11	0	882141	4	0	0
GRP168-1	23980	0	10	0	165794	4	0	0
GRP168-2	23410	0	10	0	166615	4	0	0
GRP176-1	10	0	5	0	96	3	0	0
GRP176-2	10	0	5	0	96	3	0	0
GRP182-1	10	0	5	0	93	3	0	0
GRP182-2	10	0	5	0	96	3	0	0
GRP182-3	20	0	5	0	93	3	0	0
GRP182-4	10	0	5	0	96	3	0	0
GRP186-4	872570	0	12	0	5696458	10	24	0
GRP188-1	660	0	8	0	4998	4	0	0
GRP188-2	700	0	8	0	5394	4	0	0
GRP189-1	10	0	5	0	93	3	0	0
GRP189-2	20	0	5	0	96	3	0	0
HEN001-1	10	0	3	0	12	3	0	0
HEN001-3	0	0	3	0	15	3	0	0
HEN001-5	0	0	2	0	6	2	0	0
HEN002-1	0	0	3	0	12	3	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
HEN002-2	0	0	3	0	12	3	0	0
HEN002-3	0	0	3	0	15	3	0	0
HEN002-4	0	0	3	0	15	3	0	0
HEN002-5	0	0	2	0	5	2	0	0
HEN003-3	154150	0	13	0	2082597	5709	3324	0
HEN003-4	82760	0	12	0	1095361	3563	2352	0
HEN003-5	45370	0	12	0	532962	1928	947	0
HEN006-4	10560	0	10	0	158890	291	73	0
HEN007-2	80520	0	11	0	1326444	2403	6249	0
HEN007-4	340	0	7	0	5754	8	0	0
HEN007-6	65590	0	11	0	1075535	2265	805	0
HEN008-1	514460	0	14	0	7891234	9477	10058	0
HEN008-2	131920	0	11	0	2171503	3045	9907	0
HEN008-3	52100	0	12	0	731218	1117	334	0
HEN008-4	12350	0	10	0	200574	221	50	0
HEN008-5	22070	0	11	0	265241	817	297	0
HEN008-6	15430	0	11	0	214954	581	111	0
HEN010-4	37470	0	11	0	562476	2086	417	0
HEN010-6	31160	0	11	0	432803	2006	389	0
HEN012-3	213210	0	14	0	2894547	7782	4396	0
LCL006-1	12090	0	16	0	52908	4	274	0
LCL007-1	0	0	4	0	11	3	0	0
LCL008-1	90	0	12	0	402	3	0	0
LCL009-1	19520	0	22	0	80836	4	2	0
LCL010-1	510	0	18	0	2264	2	1	0
LCL011-1	34350	0	24	0	134176	2	34	0
LCL013-1	10	0	6	0	20	2	0	0
LCL022-1	55510	0	24	0	228751	3	11	0
LCL023-1	21480	0	22	0	90467	3	1	0
LCL025-1	24390	0	16	0	124084	40	6	0
LCL027-1	20	0	8	0	120	3	0	0
LCL029-1	965720	0	18	0	4955470	569	454	0
LCL033-1	410	0	16	0	1523	2	1	0
LCL035-1	110	0	14	0	394	2	2	0
LCL041-1	60	0	8	0	380	4	0	0
LCL043-1	10	0	6	0	50	3	0	0
LCL044-1	50	0	8	0	300	3	0	0
LCL045-1	7760	0	12	0	37714	34	4	0
LCL046-1	10	0	6	0	24	2	0	0
LCL064-1	23080	0	16	0	116497	27	4	0
LCL064-2	7010	0	16	0	35906	25	3	0
LCL065-1	802460	0	20	0	4072646	342	520	0
LCL066-1	20650	0	16	0	110662	5	4	0
LCL069-1	21850	0	18	0	103563	5	78	0
LCL072-1	19160	0	18	0	90226	2	33	0
LCL076-1	921170	0	20	0	4691249	134	334	0
LCL076-2	10	0	4	0	13	3	0	0
LCL076-3	5100	0	12	0	36595	5	1	0
LCL077-1	113280	0	18	0	587117	10	31	0
LCL077-2	370	0	10	0	2868	2	1	0
LCL079-1	10	0	8	0	83	3	0	0
LCL081-1	4310	0	22	0	17928	2	3	0
LCL082-1	320	0	16	0	1380	2	1	0
LCL083-1	1399910	0	32	0	5472721	3	656	0
LCL083-2	166080	0	20	0	779192	103	17	0
LCL087-1	76440	0	26	0	332138	2	2	0
LCL091-1	1019660	0	34	0	4306733	2	103	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
LCL096-1	330	0	10	0	966	2	0	0
LCL097-1	3130	0	14	0	6847	2	0	0
LCL098-1	360	0	14	0	890	2	0	0
LCL101-1	643800	0	24	0	1678885	2	19186	0
LCL102-1	33000	0	16	0	99184	2	1574	0
LCL104-1	932600	0	24	0	2609879	2	51	0
LCL106-1	40	0	10	0	217	3	0	0
LCL107-1	6180	0	20	0	12552	2	0	0
LCL108-1	261270	0	28	0	446943	2	0	0
LCL110-1	521530	0	18	0	2514280	148	271	0
LCL111-1	1290	0	12	0	6320	3	0	0
LCL117-1	30	0	10	0	118	2	0	0
LCL118-1	1840	0	18	0	8444	4	44	0
LCL120-1	770	0	16	0	2805	2	22	0
LCL126-1	40	0	10	0	174	3	1	0
LCL130-1	1220	0	18	0	3239	2	25	0
LCL132-1	700240	0	14	0	5664237	9	863	0
LCL143-1	59610	0	9	2294	1378787	32	3	6618
LCL169-1	0	0	3	0	8	3	0	0
LCL170-1	0	0	3	0	8	3	0	0
LCL171-1	0	0	3	0	8	3	0	0
LCL172-1	0	0	3	0	8	3	0	0
LCL173-1	10	0	3	0	8	3	0	0
LCL174-1	10	0	5	0	55	4	0	0
LCL175-1	0	0	3	0	8	3	0	0
LCL176-1	10	0	5	0	42	2	0	0
LCL177-1	10	0	7	0	116	3	0	0
LCL178-1	30	0	7	0	165	3	0	0
LCL181-2	0	0	3	0	15	3	0	0
LCL182-1	9010	0	13	0	53328	3	0	0
LCL185-1	10	0	5	0	48	2	0	0
LCL186-1	10	0	5	0	52	2	0	0
LCL187-1	30	0	7	0	235	3	0	0
LCL188-1	30	0	7	0	194	3	0	0
LCL189-1	40	0	7	0	267	3	0	0
LCL190-1	0	0	5	0	59	4	0	0
LCL192-1	280	0	9	0	1669	6	0	0
LCL193-1	70	0	7	0	431	4	0	0
LCL194-1	300	0	9	0	1986	6	0	0
LCL195-1	11100	0	13	0	66214	7	0	0
LCL196-1	171820	0	17	0	1041307	5	0	0
LCL197-1	60	0	7	0	369	5	0	0
LCL198-1	357190	0	17	0	2037875	5	0	0
LCL199-1	2490	0	13	0	14674	3	0	0
LCL200-1	490	0	11	0	3062	3	0	0
LCL201-1	4480	0	13	0	26563	3	0	0
LCL202-1	16610	0	15	0	94850	5	0	0
LCL203-1	4690	0	13	0	27758	3	0	0
LCL204-1	4680	0	13	0	26679	3	0	0
LCL205-1	23590	0	15	0	134159	5	0	0
LCL206-1	4910	0	13	0	27876	3	0	0
LCL207-1	850	0	11	0	5249	3	0	0
LCL208-1	46110	0	15	0	254443	4	0	0
LCL210-1	162030	0	17	0	919324	5	0	0
LCL211-1	1230	0	11	0	7539	2	0	0
LCL212-1	10	0	5	0	40	4	0	0
LCL213-1	2550	0	11	0	14401	6	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
LCL214-1	1960	0	11	0	11723	6	0	0
LCL215-1	9980	0	13	0	59747	7	0	0
LCL216-1	1330	0	11	0	8390	6	0	0
LCL217-1	2010	0	11	0	12262	6	0	0
LCL218-1	8810	0	13	0	51439	3	0	0
LCL226-1	0	0	5	0	51	4	0	0
LCL230-1	79310	0	15	0	414626	5	0	0
LCL230-2	10	0	4	0	22	4	0	0
LCL231-1	108240	0	15	0	560253	5	0	0
LDA003-1	210100	0	13	0	2107632	25	0	0
MSC001-1	258460	0	15	166	3971792	2507	549	1298
MSC002-1	400	0	12	0	3200	3	0	0
MSC002-2	330	0	12	0	2237	3	0	0
MSC003-1	30	0	11	0	147	34	0	0
MSC004-1	410660	2	30	49984	1605535	3501	47249	0
MSC005-1	30	0	10	0	267	34	0	0
NUM001-1	840	0	8	0	8456	2	0	0
NUM002-1	450	0	8	0	4461	3	0	0
NUM003-1	280	0	8	0	2825	3	0	0
NUM004-1	360	0	8	0	3583	3	0	0
NUM009-1	1000	0	6	0	38301	12	1	2
NUM014-1	10	0	7	0	56	6	0	0
NUM015-1	9380	1	16	2643	65641	4075	0	3154
NUM016-1	190	1	11	33	1234	59	0	40
NUM016-2	110	0	15	0	590	112	0	0
NUM019-1	10	0	4	0	32	3	0	0
NUM022-1	140	0	10	0	1429	8	0	1
NUM023-1	0	0	3	0	10	2	0	0
NUM024-1	20750	0	9	0	273250	2	8	0
NUM025-1	0	0	4	0	23	3	0	0
NUM025-2	10	0	4	0	23	3	0	0
NUM027-1	31010	0	10	1147	511922	1303	294	3824
NUM139-1	20	0	4	0	450	3	0	0
NUM180-1	7270	0	8	212	329528	22	1	164
NUM228-1	20	0	4	0	513	3	0	0
PLA001-1	2180	0	13	0	17638	1	0	0
PLA002-1	1290	0	12	0	12547	25	12	0
PLA003-1	190	0	8	0	1567	1	0	0
PLA006-1	20	0	7	0	126	4	0	0
PLA007-1	103650	0	24	0	659625	42318	0	2171
PLA016-1	59920	0	23	0	377223	15520	0	1208
PLA017-1	90	0	10	0	634	26	0	0
PLA019-1	96300	0	24	0	612377	25636	0	2079
PLA020-1	0	0	5	0	31	2	0	0
PLA022-1	36180	0	16	0	111522	3321	0	2683
PLA022-2	740	0	16	0	5055	310	0	20
PRV003-1	10	0	4	0	259	3	0	0
PRV005-1	50	0	5	0	945	5	0	0
PRV006-1	320	0	6	46	6564	24	0	33
PRV009-1	500	0	14	159	2947	222	4	6
PUZ001-1	10	1	11	5	111	39	0	0
PUZ002-1	20	0	12	0	78	12	0	0
PUZ003-1	20	0	12	0	201	19	0	0
PUZ004-1	10	0	11	0	69	11	0	2
PUZ008-1	0	0	3	0	16	2	0	0
PUZ008-2	800	0	13	0	5756	13	0	0
PUZ009-1	270	3	8	219	5234	317	0	625

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
PUZ011-1	10	0	6	0	42	6	0	0
PUZ012-1	3570	0	23	3766	21368	3686	470	1783
PUZ013-1	110	0	15	26	1140	213	0	54
PUZ014-1	2640	10	19	3811	20132	8188	0	2027
PUZ015-2.003	4360	2	19	1950	48020	9719	0	7734
PUZ016-2.003	1060	2	11	231	12488	1242	0	1441
PUZ020-1	80	0	8	0	1381	0	0	15
PUZ022-1	580	0	9	55	4151	417	0	100
PUZ024-1	180	0	8	20	2943	3	28	18
PUZ029-1	360	5	27	256	1944	1443	0	88
PUZ031-1	102430	0	70	10033	523328	131062	0	0
PUZ032-1	6770	4	8	2504	70185	548	781	1113
RNG001-3	89450	0	20	0	937670	17	236	0
RNG001-4	64310	0	10	0	1400557	2	379	0
RNG002-1	15480	0	10	0	336727	5	0	0
RNG003-1	17460	0	10	0	385436	5	0	0
RNG005-2	40	0	6	0	803	3	0	0
RNG006-1	80	0	7	0	1740	3	0	0
RNG006-2	70	0	7	0	1443	3	0	0
RNG011-5	0	0	2	0	5	2	0	0
RNG023-6	110	0	7	0	985	3	0	0
RNG023-7	130	0	7	0	985	3	0	0
RNG024-6	110	0	7	0	985	3	0	0
RNG024-7	120	0	7	0	985	3	0	0
RNG037-2	80	0	7	0	1706	4	0	0
RNG038-2	40	0	7	0	724	3	0	0
RNG040-1	430	0	6	65	13858	3	0	115
RNG040-2	153860	0	9	18747	4393803	3	4	42049
RNG041-1	6460	0	8	627	199238	78	7	1489
ROB010-1	221730	0	13	0	1596704	3	16	0
ROB013-1	11940	0	11	0	89312	3	0	0
ROB016-1	3400	0	9	0	27947	5	0	0
ROB021-1	112690	0	11	0	1162337	4	0	0
SET001-1	0	0	5	0	24	4	0	0
SET003-1	10	0	6	0	131	4	0	0
SET004-1	20	0	6	0	131	4	0	0
SET006-1	0	0	6	0	138	5	0	0
SET008-1	430	0	8	116	8117	30	41	33
SET009-1	7410	1	12	4079	89748	1181	543	1081
SET024-3	5430	0	7	10	278546	87	1	16
SET024-4	5070	0	7	2	264812	77	0	16
SET024-6	80	0	5	12	1942	3	0	7
SET024-7	100	0	5	14	3192	3	0	9
SET025-3	30	0	5	0	828	3	0	0
SET025-4	30	0	5	0	745	3	0	0
SET025-6	20	0	4	0	555	3	1	0
SET025-7	30	0	4	0	859	3	1	0
SET027-3	336660	0	8	431	25441619	49	16	278
SET027-4	48780	0	8	157	3449307	47	12	186
SET027-6	20910	0	8	178	1308054	17	1	209
SET027-7	22080	0	8	180	1331342	59	1	264
SET043-5	0	2	3	7	43	7	0	0
SET044-5	120	2	5	41	1298	23	0	81
SET045-5	10	2	4	5	63	3	0	8
SET046-5	200	5	5	255	1656	146	26	114
SET050-6	3170	0	6	32	199812	4	1	50
SET051-6	3300	0	6	44	202829	4	1	78

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SET052-6	100	0	5	4	4873	4	1	4
SET053-6	110	0	5	10	5242	4	1	6
SET054-6	10	0	3	0	33	3	0	0
SET054-7	10	0	3	0	33	3	0	0
SET055-7	0	0	4	0	34	4	0	0
SET056-6	390	0	6	0	11894	10	0	4
SET056-7	410	0	6	0	12409	11	0	8
SET057-6	330	0	6	16	8730	8	0	41
SET057-7	360	0	6	16	10032	8	0	52
SET058-6	330	0	6	16	8802	10	0	41
SET058-7	370	0	6	16	10104	10	0	52
SET059-6	300	0	6	24	8872	8	0	75
SET059-7	360	0	6	24	11091	8	0	88
SET060-6	80	1	4	19	2788	4	0	16
SET060-7	110	1	4	19	3688	4	0	19
SET061-7	170	0	6	16	5300	2	1	38
SET062-7	0	0	3	0	21	3	0	0
SET063-7	0	0	4	0	233	4	0	0
SET064-7	60	0	4	0	4009	4	0	0
SET065-6	30	0	5	0	653	2	2	0
SET065-7	30	0	5	0	900	2	2	0
SET073-7	2790	0	5	12	192317	2	0	13
SET074-7	2830	0	5	20	192983	2	0	14
SET075-7	5830	0	5	38	389200	4	0	52
SET077-6	10	0	4	0	479	3	1	0
SET077-7	30	0	4	0	673	3	1	0
SET078-6	40	0	5	0	1070	4	1	0
SET078-7	0	0	3	0	32	3	0	0
SET079-7	3280	0	5	69	211797	10	0	73
SET080-7	10	0	3	0	94	3	0	0
SET081-6	2110	0	7	208	94252	11	0	169
SET081-7	400	0	6	22	12952	3	0	55
SET090-7	1280	1	4	90	65578	15	4	86
SET093-6	307790	0	6	22	20665552	4	2	351
SET093-7	3000	0	4	0	203801	3	0	3
SET095-7	5470	1	5	249	271599	7	0	201
SET098-7	330	0	5	0	13300	4	0	0
SET101-6	12750	0	7	4	307482	22	12	2
SET101-7	210	0	5	0	5280	10	2	0
SET102-6	150	0	5	0	3560	8	0	0
SET102-7	200	0	5	0	4893	10	2	0
SET108-6	7590	1	5	259	463718	17	0	185
SET108-7	30480	1	5	643	1809671	54	18	500
SET117-6	289220	0	6	22	19240371	4	2	348
SET117-7	3210	0	4	0	217483	3	0	3
SET118-6	0	0	4	0	199	3	0	0
SET118-7	10	0	4	0	601	3	0	0
SET152-6	3750	1	5	5	258958	5	0	5
SET153-6	3890	1	5	19	261535	5	0	15
SET158-6	3500	0	5	12	246571	4	0	33
SET196-6	20	0	5	2	413	5	0	0
SET197-6	20	0	5	2	433	5	0	0
SET231-6	10	0	4	0	158	4	0	0
SET234-6	58660	0	7	738	3274313	30	30	912
SET239-6	310	0	6	16	8531	4	0	41
SET240-6	3740	0	7	222	142148	11	1	206
SET241-6	3720	0	7	222	142158	11	1	206

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SET242-6	5350	0	6	144	329124	4	0	141
SET252-6	4950	0	8	206	175403	22	1	192
SET253-6	5240	0	8	206	179543	22	1	194
SET296-6	10	0	2	0	8	2	0	0
SET451-6	7790	0	8	202	226139	22	3	159
SET479-6	10260	0	6	477	642035	58	2	313
SET553-6	5540	0	8	202	184370	22	2	169
SET563-6	198810	0	6	13	13581503	4	0	220
SYN003-1.006	1040	0	17	158	8759	3637	0	0
SYN004-1.007	18970	15	42	34631	101046	96943	0	0
SYN005-1.010	30	0	11	0	211	2	0	0
SYN006-1	30	0	7	0	115	1	3	0
SYN007-1.002	20	2	5	14	624	37	0	26
SYN008-1	0	0	4	0	35	4	0	0
SYN009-1	20	0	7	0	121	6	0	0
SYN011-1	180	2	15	112	1370	589	0	13
SYN014-2	60	0	5	0	1522	8	0	6
SYN028-1	0	1	7	3	57	11	0	0
SYN029-1	10	2	8	5	85	28	0	5
SYN030-1	8390	29	49	9630	94933	19096	0	6885
SYN031-1	70	2	6	24	907	25	0	30
SYN032-1	140	10	30	205	818	701	0	0
SYN033-1	0	0	5	0	14	1	0	0
SYN034-1	170	6	7	163	1118	190	0	53
SYN035-1	10	0	9	0	27	1	14	0
SYN040-1	0	0	3	0	15	3	0	0
SYN041-1	0	0	2	0	6	2	0	0
SYN044-1	50	3	8	29	670	85	0	52
SYN045-1	10	0	5	0	47	11	0	0
SYN046-1	0	0	3	0	11	3	0	0
SYN047-1	10	1	6	1	108	25	0	0
SYN048-1	0	0	2	0	4	1	0	0
SYN049-1	0	0	3	0	11	1	2	0
SYN050-1	0	0	4	0	29	2	0	0
SYN051-1	10	2	5	5	70	13	0	0
SYN052-1	30	2	5	11	379	27	1	8
SYN053-1	100	2	8	32	1181	192	0	4
SYN054-1	30	3	10	22	158	71	0	0
SYN055-1	30	3	9	29	217	65	0	8
SYN057-1	20	0	9	6	95	23	0	0
SYN058-1	10	0	6	0	58	8	0	0
SYN060-1	10	0	5	0	29	5	0	0
SYN061-1	0	0	5	0	14	4	0	0
SYN062-1	10	1	6	2	102	18	0	3
SYN063-1	10	0	5	0	102	5	0	0
SYN063-2	10	0	3	0	11	3	0	0
SYN064-1	0	0	2	0	4	1	0	0
SYN065-1	0	0	4	0	15	1	0	0
SYN066-1	0	0	5	0	21	1	0	0
SYN068-1	10	1	5	1	36	5	0	0
SYN069-1	1110	5	19	783	5152	2152	0	91
SYN070-1	322860	9	16	543377	1437534	637123	0	92109
SYN073-1	0	0	3	0	15	1	0	0
SYN079-1	0	0	4	0	9	3	0	0
SYN080-1	0	0	3	0	15	3	0	0
SYN081-1	40	2	6	21	596	0	0	21
SYN083-1	10	0	4	0	27	3	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN084-2	15710	17	22	5609	59042	16829	0	1599
SYN085-1.010	0	0	12	0	33	12	0	0
SYN088-1.010	10	0	12	0	33	12	0	0
SYN089-1.002	10	0	4	0	17	4	0	0
SYN095-1.002	0	0	4	0	17	2	0	0
SYN103-1	0	0	2	0	3	2	0	0
SYN104-1	0	0	2	0	3	2	0	0
SYN105-1	0	0	3	0	6	2	0	0
SYN106-1	0	0	3	0	6	2	0	0
SYN107-1	0	0	4	0	43	1	0	0
SYN108-1	10	0	4	0	48	1	0	0
SYN109-1	40	0	10	0	350	19	0	0
SYN110-1	40	0	12	0	260	20	4	0
SYN111-1	30	0	10	0	235	20	0	0
SYN112-1	10	0	4	0	48	1	0	0
SYN113-1	30	0	9	0	180	11	0	0
SYN114-1	20	0	5	0	426	1	0	0
SYN115-1	230	0	14	0	2008	94	3	0
SYN116-1	2730	0	12	0	49083	307	93	0
SYN117-1	7010	0	13	0	128063	623	206	0
SYN118-1	0	0	2	0	3	2	0	0
SYN119-1	0	0	2	0	3	2	0	0
SYN120-1	10	0	3	0	8	2	0	0
SYN121-1	20	0	13	0	142	23	0	0
SYN122-1	10	0	10	0	61	18	0	0
SYN123-1	10	0	6	0	83	2	3	0
SYN124-1	50	0	7	0	1672	13	0	0
SYN125-1	10	0	7	0	47	12	0	0
SYN126-1	10	0	6	0	163	3	0	0
SYN127-1	20	0	6	0	185	3	0	0
SYN128-1	100190	0	16	0	1777858	18035	2563	0
SYN129-1	100220	0	16	0	1777858	18035	2563	0
SYN130-1	0	0	2	0	3	1	0	0
SYN131-1	0	0	2	0	3	1	0	0
SYN132-1	0	0	2	0	3	1	0	0
SYN133-1	0	0	2	0	3	1	0	0
SYN134-1	10	0	5	0	16	4	0	0
SYN135-1	10	0	5	0	275	2	0	0
SYN136-1	10	0	5	0	271	2	0	0
SYN137-1	16480	0	38	0	93183	6024	959	0
SYN138-1	190	0	19	0	960	113	8	0
SYN139-1	89470	0	35	0	555181	44573	4755	0
SYN140-1	90150	0	35	0	555267	44572	4743	0
SYN141-1	340	0	15	0	3333	108	5	0
SYN144-1	50	0	13	0	309	30	0	0
SYN145-1	0	0	2	0	3	1	0	0
SYN146-1	0	0	3	0	25	1	0	0
SYN147-1	0	0	3	0	24	2	0	0
SYN148-1	10	0	9	0	62	11	0	0
SYN149-1	0	0	3	0	24	1	0	0
SYN150-1	0	0	4	0	36	2	0	0
SYN151-1	0	0	4	0	33	2	0	0
SYN152-1	10	0	4	0	36	2	0	0
SYN153-1	10	0	5	0	185	3	0	0
SYN154-1	10	0	5	0	224	3	0	0
SYN157-1	14930	0	14	0	223773	3025	788	0
SYN158-1	5710	0	14	0	90966	1566	65	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN159-1	5570	0	14	0	90966	1566	65	0
SYN160-1	24970	0	14	0	428325	5449	537	0
SYN161-1	11500	0	14	0	189449	2501	147	0
SYN162-1	11460	0	14	0	189449	2502	147	0
SYN164-1	0	0	2	0	3	2	0	0
SYN165-1	0	0	3	0	29	1	0	0
SYN166-1	10	0	3	0	29	1	0	0
SYN167-1	0	0	3	0	29	1	0	0
SYN168-1	10	0	5	0	153	2	0	0
SYN169-1	10	0	5	0	140	2	1	0
SYN170-1	40	0	6	0	857	1	0	0
SYN171-1	90470	0	33	0	422524	26985	6853	40226
SYN172-1	0	0	2	0	3	2	0	0
SYN173-1	0	0	4	0	11	2	0	0
SYN174-1	10	0	4	0	11	2	0	0
SYN175-1	0	0	4	0	88	2	1	0
SYN176-1	450	0	10	0	3515	125	2	0
SYN177-1	40	0	7	0	590	12	0	0
SYN178-1	4900	0	15	0	32223	690	366	0
SYN181-1	5090	0	16	0	35690	708	368	0
SYN182-1	10	0	7	0	204	3	0	0
SYN183-1	20	0	7	0	272	4	0	0
SYN184-1	0	0	2	0	3	2	0	0
SYN185-1	10	0	2	0	3	2	0	0
SYN186-1	10	0	6	0	47	5	0	0
SYN187-1	10	0	6	0	77	5	0	0
SYN188-1	0	0	6	0	71	5	0	0
SYN189-1	0	0	8	0	54	7	0	0
SYN190-1	191480	0	24	0	1324532	43992	9041	0
SYN191-1	40	0	10	0	258	23	0	0
SYN192-1	50	0	13	0	526	20	0	0
SYN193-1	50	0	13	0	686	20	0	0
SYN194-1	330	0	15	0	3335	114	5	0
SYN195-1	140	0	19	0	1513	44	12	13
SYN196-1	0	0	5	0	22	2	0	0
SYN197-1	0	0	3	0	7	3	0	0
SYN198-1	10	0	5	0	19	2	0	0
SYN199-1	10	0	5	0	28	2	0	0
SYN200-1	0	0	5	0	22	2	0	0
SYN201-1	10	0	9	0	150	20	0	0
SYN202-1	26390	0	18	0	152525	7462	1391	0
SYN203-1	100	0	12	0	1318	54	2	0
SYN204-1	17490	0	26	0	106814	8663	911	0
SYN205-1	17330	0	26	0	106814	8663	911	0
SYN206-1	20	0	14	0	293	39	0	0
SYN207-1	16040	0	26	0	90335	5120	914	0
SYN208-1	50	0	12	0	358	17	1	0
SYN209-1	30	0	8	0	459	3	0	0
SYN210-1	20	0	8	0	175	3	0	0
SYN211-1	20	0	8	0	183	3	3	0
SYN212-1	20	0	8	0	263	3	0	0
SYN213-1	1650	0	24	0	18609	507	43	0
SYN215-1	580	0	20	0	5123	219	42	0
SYN216-1	0	0	6	0	35	2	0	0
SYN217-1	10	0	7	0	76	10	0	0
SYN218-1	0	0	6	0	53	1	0	0
SYN219-1	30	0	13	0	142	23	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN220-1	20	0	6	0	643	4	0	0
SYN221-1	70	0	7	0	1375	12	0	0
SYN222-1	90	0	7	0	1620	8	3	0
SYN223-1	30	0	6	0	527	6	0	0
SYN224-1	80	0	7	0	1672	12	0	0
SYN225-1	10	0	7	0	41	8	0	0
SYN226-1	30	0	6	0	632	8	0	0
SYN227-1	10	0	7	0	39	10	0	0
SYN228-1	10	0	6	0	34	8	0	0
SYN229-1	80	0	7	0	1487	9	0	0
SYN230-1	80	0	7	0	1487	10	0	0
SYN231-1	80	0	7	0	1577	6	3	0
SYN232-1	80	0	7	0	1566	7	3	0
SYN233-1	30	0	6	0	566	7	0	0
SYN234-1	10	0	7	0	46	9	0	0
SYN235-1	10	0	7	0	43	11	0	0
SYN236-1	20	0	6	0	558	9	0	0
SYN237-1	0	0	3	0	30	1	0	0
SYN238-1	10	0	3	0	29	1	0	0
SYN239-1	0	0	3	0	29	1	0	0
SYN240-1	0	0	3	0	29	1	0	0
SYN241-1	10	0	3	0	29	2	0	0
SYN242-1	0	0	3	0	27	3	0	0
SYN243-1	0	0	4	0	13	3	0	0
SYN244-1	0	0	3	0	30	1	0	0
SYN245-1	0	0	3	0	30	1	0	0
SYN246-1	10	0	4	0	22	4	0	0
SYN247-1	0	0	3	0	28	1	0	0
SYN248-1	10	0	7	0	64	6	0	0
SYN249-1	20	0	7	0	60	8	0	0
SYN250-1	170	0	14	0	891	152	9	0
SYN251-1	0	0	4	0	13	3	0	0
SYN253-1	90320	0	35	0	555267	44572	4743	0
SYN254-1	17570	0	36	0	107293	8815	911	0
SYN255-1	0	0	4	0	38	2	0	0
SYN256-1	0	0	4	0	36	2	0	0
SYN257-1	10	0	2	0	3	2	0	0
SYN258-1	10	0	3	0	29	1	0	0
SYN259-1	10	0	3	0	28	1	0	0
SYN260-1	10	0	3	0	28	1	0	0
SYN261-1	0	0	3	0	29	1	0	0
SYN262-1	10	0	6	0	47	5	0	0
SYN263-1	20	0	7	0	124	5	3	0
SYN264-1	10	0	7	0	49	7	0	0
SYN265-1	10	0	5	0	144	2	0	0
SYN266-1	400	0	13	0	5414	157	0	0
SYN267-1	50	0	6	0	900	1	0	0
SYN268-1	50	0	6	0	901	1	0	0
SYN270-1	50	0	12	0	280	33	0	0
SYN271-1	90460	0	33	0	422473	26984	6853	40226
SYN272-1	120	0	11	0	800	45	6	0
SYN273-1	190	0	12	0	1542	56	0	0
SYN274-1	10	0	2	0	3	1	0	0
SYN275-1	0	0	2	0	3	1	0	0
SYN276-1	10	0	2	0	3	2	0	0
SYN277-1	0	0	3	0	32	1	0	0
SYN278-1	0	0	3	0	33	1	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN279-1	10	0	4	0	17	1	0	0
SYN280-1	0	0	3	0	35	1	0	0
SYN281-1	0	0	3	0	32	1	0	0
SYN282-1	0	0	3	0	34	1	0	0
SYN283-1	0	0	4	0	11	2	0	0
SYN284-1	10	0	4	0	11	2	0	0
SYN285-1	10	0	7	0	50	9	0	0
SYN286-1	0	0	4	0	15	1	0	0
SYN287-1	10	0	3	0	35	1	0	0
SYN288-1	0	0	3	0	34	2	0	0
SYN289-1	0	0	4	0	11	2	0	0
SYN290-1	10	0	3	0	32	1	0	0
SYN291-1	0	0	3	0	34	1	0	0
SYN292-1	0	0	3	0	8	3	0	0
SYN293-1	10	0	6	0	37	8	0	0
SYN294-1	10	0	6	0	43	10	0	0
SYN295-1	10	0	3	0	32	2	0	0
SYN296-1	10	0	4	0	11	2	0	0
SYN297-1	0	0	3	0	32	2	0	0
SYN298-1	30	0	14	0	293	38	1	0
SYN299-1	30	0	14	0	293	39	0	0
SYN300-1	30	0	14	0	293	39	0	0
SYN301-1	20	0	8	0	161	3	0	0
SYN315-1	30	2	5	11	231	24	6	8
SYN318-1	10	0	3	0	20	2	1	0
SYN319-1	10	0	5	0	150	1	0	0
SYN321-1	50	2	6	17	397	35	0	13
SYN323-1	30	2	5	12	189	28	0	12
SYN325-1	10	0	4	0	42	1	0	0
SYN326-1	10	0	5	0	66	1	5	0
SYN327-1	610	3	6	378	8241	448	0	412
SYN331-1	120	0	6	0	776	1	3	30
SYN333-1	60	0	9	0	193	1	84	0
SYN336-1	10	0	2	0	10	1	0	0
SYN338-1	0	0	2	0	6	1	0	0
SYN339-1	0	0	2	0	4	1	0	0
SYN340-1	0	0	2	0	4	1	0	0
SYN341-1	0	0	2	0	4	1	0	0
SYN343-1	0	2	3	4	33	4	0	0
SYN346-1	10	1	2	1	14	1	0	0
SYN352-1	46590	8	7	22130	443569	9728	14143	4992
SYN354-1	21870	10	7	7867	262657	5624	459	9014
TOP001-2	57780	3	14	28787	437973	3454	4839	13500
TOP002-2	0	0	3	0	6	3	0	0
TOP004-1	0	0	2	0	47	1	0	0
TOP004-2	0	0	2	0	22	1	0	0
TOP005-2	656720	6	14	350839	5548935	50939	52012	13957

Tabla 19: Resultados temporales y espaciales del SLT vía POS

A continuación se incluye la tabla de los resultados temporales y espaciales obtenidos por el SLT vía EMPTY.

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
BOO003-1	13800	0	10	0	358232	4	0	0
BOO003-2	1076310	0	13	0	10565377	4	16	0
BOO003-4	303270	0	13	0	3336619	4	0	0
BOO004-1	9450	0	10	0	254802	4	0	0
BOO004-2	991880	0	13	0	10048266	4	0	0
BOO004-4	310460	0	13	0	3409756	4	0	0
BOO005-1	12330	0	10	0	328484	4	0	0
BOO006-1	17120	0	10	0	439600	4	0	0
BOO012-1	881170	0	12	0	20838467	9	0	0
CAT001-3	30860	0	9	0	1001818	188	0	0
CAT001-4	4670	0	9	0	85411	96	0	0
CAT002-3	5860	0	8	0	190998	91	0	0
CAT002-4	1890	0	8	0	40822	63	0	0
CAT003-3	700	0	7	0	22111	13	0	0
CAT003-4	250	0	7	0	4665	8	0	0
CAT004-3	272430	0	10	0	8753084	3693	0	0
CAT004-4	36160	0	10	0	751176	1839	0	0
CAT005-1	338640	0	13	0	6435311	53415	3800	0
CAT006-1	351550	0	13	0	6511714	55947	3804	0
CAT007-1	190	0	7	0	4274	10	0	0
CAT007-3	0	0	4	0	21	4	0	0
CAT011-1	30	0	6	0	614	4	0	0
CAT011-2	60	0	7	0	866	4	0	0
CAT012-1	30	0	6	0	474	6	0	0
CAT012-3	860	0	8	0	25608	31	0	0
CAT012-4	310	0	8	0	5741	25	0	0
CAT013-1	30	0	6	0	451	4	0	0
CAT013-3	410	0	7	0	11324	10	0	0
CAT013-4	210	0	7	0	4062	7	0	0
CAT014-1	40	0	6	0	637	6	0	0
CAT014-2	240	0	8	0	3140	4	0	0
CAT016-3	110	0	6	0	2587	2	0	0
CAT016-4	60	0	6	0	1390	2	0	0
CAT017-3	120	0	6	0	2770	2	0	0
CAT017-4	140	0	7	0	2765	3	0	0
COL001-1	767750	0	12	0	6192105	1	6	0
COL001-2	380	0	7	0	3358	1	0	0
COL002-1	60	0	6	0	573	1	0	0
COL002-2	160	0	8	0	1132	1	0	0
COL002-3	50	0	7	0	265	1	0	0
COL007-1	0	0	2	0	5	1	0	0
COL008-1	10	0	4	0	41	1	0	0
COL009-1	200	0	7	0	2075	1	0	0
COL010-1	10	0	4	0	41	1	0	0
COL011-1	831710	0	12	0	6397939	1	33	0
COL012-1	0	0	2	0	5	1	0	0
COL013-1	0	0	2	0	5	1	0	0
COL014-1	10	0	2	0	5	1	0	0
COL015-1	0	0	4	0	37	1	0	0
COL016-1	0	0	2	0	5	1	0	0
COL017-1	10	0	4	0	41	1	0	0
COL018-1	0	0	2	0	5	1	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
COL019-1	50	0	6	0	517	1	0	0
COL020-1	50	0	6	0	502	1	0	0
COL021-1	0	0	4	0	41	1	0	0
COL022-1	0	0	4	0	41	1	0	0
COL023-1	190	0	7	0	1987	1	0	0
COL024-1	0	0	4	0	41	1	0	0
COL025-1	10	0	4	0	41	1	0	0
COL026-1	200	0	7	0	1987	1	0	0
COL027-1	200	0	7	0	1988	1	0	0
COL028-1	200	0	7	0	1987	1	0	0
COL029-1	0	0	2	0	5	1	0	0
COL030-1	610	0	8	0	4839	1	0	0
COL031-1	10	0	5	0	85	1	0	0
COL032-1	15610	0	10	0	120995	1	0	0
COL033-1	834710	0	12	0	5447422	11	20	0
COL035-1	5330	0	9	0	38205	1	0	0
COL039-1	230	0	7	0	1895	1	0	0
COL040-1	673850	0	12	0	4262022	1	53	0
COL044-1	660530	0	12	0	4262022	1	53	0
COL045-1	10	0	4	0	44	1	0	0
COL048-1	10	0	4	0	41	1	0	0
COL050-1	10	0	5	0	68	1	0	0
COL051-1	90	0	7	0	846	1	0	0
COL052-1	25000	0	10	0	231354	1	0	0
COL052-2	32690	0	11	0	332580	9	0	0
COL053-1	0	0	4	0	35	1	0	0
COL054-1	220	0	7	0	2507	1	0	0
COL055-1	0	0	3	0	15	2	0	0
COL056-1	290	0	7	0	3284	1	0	0
COL058-1	270	0	8	0	2599	1	0	0
COL070-1	180	0	7	0	1917	1	0	0
COL075-2	92920	0	11	0	705287	1	0	0
COM001-1	10	0	7	0	62	4	0	0
COM002-1	120	0	13	0	933	4	0	0
COM002-2	130	0	13	0	933	4	0	0
COM003-2	390	0	10	0	7152	27	10	0
COM004-1	580	0	10	0	7504	2	0	0
GEO002-3	0	0	3	0	45	3	0	0
GEO003-1	50	0	6	0	1304	4	0	0
GEO003-2	30	0	6	0	669	2	0	0
GEO003-3	0	0	3	0	27	2	0	0
GEO011-2	0	0	5	0	81	5	0	0
GEO011-3	20	0	5	0	588	5	0	0
GEO011-4	10	0	5	0	90	5	0	0
GEO011-5	0	0	5	0	78	5	0	0
GEO014-2	0	0	4	0	19	3	0	0
GEO015-2	20	0	6	0	491	3	0	0
GEO015-3	10	0	4	0	52	3	0	0
GEO016-2	0	0	4	0	25	3	0	0
GEO016-3	0	0	4	0	35	3	0	0
GEO017-2	460	0	8	0	10568	4	0	0
GEO017-3	10	0	5	0	312	3	0	0
GEO018-2	20	0	6	0	448	3	0	0
GEO018-3	50	0	6	0	1212	3	0	0
GEO019-2	0	0	4	0	39	3	0	0
GEO019-3	0	0	4	0	74	3	0	0
GEO021-2	20	0	6	0	363	4	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
GEO021-3	40	0	5	0	890	4	0	0
GEO022-2	470	0	8	0	10588	4	0	0
GEO022-3	40	0	5	0	1390	3	0	0
GEO024-2	50	0	6	0	1107	2	0	0
GEO024-3	50	0	5	0	2417	2	0	0
GEO027-3	2130	0	7	0	65468	9	0	0
GEO035-2	10	0	3	0	26	2	0	0
GEO035-3	0	0	3	0	26	2	0	0
GEO038-2	10	0	5	0	205	3	0	0
GEO038-3	10	0	5	0	256	3	0	0
GEO039-3	2770	0	7	0	122990	5	4	0
GEO041-3	26780	0	6	0	1408427	10	2	0
GEO047-3	38060	0	7	0	1834006	38	78	0
GEO053-3	122900	0	7	0	5996771	72	53	0
GEO054-2	10	0	5	0	207	3	0	0
GEO054-3	10	0	3	0	27	2	0	0
GEO055-2	10	0	5	0	211	3	0	0
GEO055-3	50	0	5	0	2198	3	0	0
GEO056-2	10940	0	8	0	235830	2	0	0
GEO056-3	40	0	5	0	902	3	0	0
GEO057-2	60	0	6	0	1438	2	0	0
GEO057-3	10	0	3	0	39	2	0	0
GEO058-2	315050	0	9	0	6341178	2	0	0
GEO058-3	370	0	6	0	11187	4	0	0
GEO059-3	450	0	6	0	19273	4	0	0
GEO064-3	0	0	4	0	181	4	0	0
GEO065-3	10	0	4	0	273	4	0	0
GEO066-3	10	0	4	0	365	4	0	0
GRP001-1	39580	0	11	0	711719	3	3	0
GRP001-5	970	0	11	0	10104	3	0	0
GRP003-1	30	0	11	0	245	3	0	0
GRP003-2	780	0	11	0	9519	3	0	0
GRP004-1	20	0	8	0	120	1	0	0
GRP004-2	40	0	8	0	474	3	0	0
GRP005-1	0	0	5	0	12	3	0	0
GRP006-1	10	0	8	0	42	3	0	0
GRP007-1	0	0	4	0	60	3	0	0
GRP009-1	6280	0	10	0	119989	5	0	0
GRP010-1	150	0	8	0	2878	3	0	0
GRP012-1	290	0	8	0	5525	3	0	0
GRP012-2	1006590	0	13	0	17657619	4	350	0
GRP013-1	3740	0	9	0	79234	3	0	0
GRP017-1	70	0	7	0	1200	4	0	0
GRP018-1	10	0	4	0	45	3	0	0
GRP019-1	10	0	4	0	45	3	0	0
GRP020-1	0	0	4	0	45	3	0	0
GRP021-1	10	0	4	0	45	3	0	0
GRP022-1	80	0	7	0	1402	4	0	0
GRP022-2	140000	0	13	0	1519000	3	0	0
GRP023-1	0	0	4	0	32	3	0	0
GRP023-2	0	0	5	0	51	3	0	0
GRP027-1	20	0	5	0	405	1	0	0
GRP027-2	20	0	5	0	387	1	0	0
GRP028-1	10	0	5	0	17	1	0	0
GRP028-2	10	0	5	0	129	1	0	0
GRP028-3	0	0	5	0	48	1	0	0
GRP029-1	100260	0	11	0	1688353	1	2	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
GRP029-2	41900	0	11	0	692339	1	2	0
GRP030-1	10800	0	11	0	199211	3	2	0
GRP031-1	90	0	8	0	1729	1	0	0
GRP031-2	20	0	8	0	109	1	0	0
GRP032-3	0	0	5	0	25	3	0	0
GRP033-3	10	0	7	0	111	3	0	0
GRP033-4	0	0	7	0	113	3	0	0
GRP034-3	40	0	8	0	502	3	0	0
GRP034-4	10	0	8	0	61	3	0	0
GRP036-3	520	0	8	0	10116	25	0	0
GRP037-3	20730	0	10	0	429745	1087	0	0
GRP038-3	0	0	5	0	66	3	0	0
GRP041-2	0	0	4	0	9	3	0	0
GRP042-2	10	0	6	0	37	3	0	0
GRP043-2	20	0	8	0	240	5	0	0
GRP044-2	10	0	7	0	145	3	0	0
GRP045-2	180	0	10	0	2219	4	0	0
GRP046-2	40	0	9	0	554	4	0	0
GRP047-2	1410	0	12	0	17676	5	0	0
GRP123-6.003	115090	0	24	0	1416476	134926	0	0
GRP123-7.003	112500	0	24	0	1416476	134926	0	0
GRP123-8.003	114410	0	24	0	1416476	134926	0	0
GRP123-9.003	113770	0	24	0	1416476	134926	0	0
GRP124-6.003	112270	0	24	0	1394308	132653	0	0
GRP124-7.003	111880	0	24	0	1394308	132653	0	0
GRP124-8.003	113170	0	24	0	1394308	132653	0	0
GRP126-1.002	480	0	11	0	6577	192	39	0
GRP126-2.002	520	0	11	0	6867	192	39	0
GRP126-3.002	520	0	11	0	7303	216	39	0
GRP126-4.002	8080	0	11	0	205865	2093	431	0
GRP136-1	0	0	5	0	65	3	0	0
GRP137-1	10	0	5	0	65	3	0	0
GRP139-1	540	0	8	0	5160	4	0	0
GRP140-1	258790	0	11	0	1996385	5	16	0
GRP142-1	0	0	4	0	25	3	0	0
GRP143-1	180	0	7	0	1728	4	0	0
GRP144-1	140	0	7	0	1322	3	0	0
GRP145-1	30	0	6	0	324	4	0	0
GRP146-1	570	0	8	0	5160	4	0	0
GRP148-1	262140	0	11	0	1996385	5	16	0
GRP150-1	20	0	5	0	126	4	0	0
GRP151-1	0	0	2	0	5	2	0	0
GRP152-1	580	0	8	0	4998	4	0	0
GRP153-1	10	0	5	0	93	3	0	0
GRP154-1	30	0	6	0	259	4	0	0
GRP155-1	20	0	6	0	260	4	0	0
GRP156-1	8960	0	9	0	75795	3	0	0
GRP157-1	30	0	6	0	259	4	0	0
GRP158-1	20	0	6	0	260	4	0	0
GRP160-1	0	0	2	0	5	2	0	0
GRP161-1	0	0	2	0	5	2	0	0
GRP162-1	115830	0	11	0	893022	6	8	0
GRP163-1	93220	0	11	0	727126	6	4	0
GRP165-1	109110	0	11	0	881275	4	0	0
GRP166-3	108330	0	11	0	882141	4	0	0
GRP168-1	20910	0	10	0	165794	4	0	0
GRP168-2	21100	0	10	0	166615	4	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
GRP176-1	10	0	5	0	96	3	0	0
GRP176-2	10	0	5	0	96	3	0	0
GRP182-1	10	0	5	0	93	3	0	0
GRP182-2	10	0	5	0	96	3	0	0
GRP182-3	0	0	5	0	93	3	0	0
GRP182-4	10	0	5	0	96	3	0	0
GRP186-4	755350	0	12	0	5696458	10	24	0
GRP188-1	520	0	8	0	4998	4	0	0
GRP188-2	580	0	8	0	5394	4	0	0
GRP189-1	10	0	5	0	93	3	0	0
GRP189-2	10	0	5	0	96	3	0	0
HEN001-1	0	0	3	0	12	3	0	0
HEN001-3	0	0	3	0	15	3	0	0
HEN001-5	0	0	2	0	6	2	0	0
HEN002-1	0	0	3	0	12	3	0	0
HEN002-2	0	0	3	0	12	3	0	0
HEN002-3	0	0	3	0	15	3	0	0
HEN002-4	0	0	3	0	15	3	0	0
HEN002-5	10	0	2	0	5	2	0	0
HEN003-3	123780	0	13	0	2082597	5709	3324	0
HEN003-4	67780	0	12	0	1095361	3563	2352	0
HEN003-5	38250	0	12	0	532962	1928	947	0
HEN006-4	8660	0	10	0	158890	291	73	0
HEN007-2	67600	0	11	0	1326444	2403	6249	0
HEN007-4	260	0	7	0	5754	8	0	0
HEN007-6	56310	0	11	0	1075535	2265	805	0
HEN008-1	426790	0	14	0	7891234	9477	10058	0
HEN008-2	113560	0	11	0	2171503	3045	9907	0
HEN008-3	42450	0	12	0	731218	1117	334	0
HEN008-4	10290	0	10	0	200574	221	50	0
HEN008-5	18450	0	11	0	265241	817	297	0
HEN008-6	12670	0	11	0	214954	581	111	0
HEN010-4	31640	0	11	0	562476	2086	417	0
HEN010-6	26290	0	11	0	432803	2006	389	0
HEN012-3	177080	0	14	0	2894547	7782	4396	0
LCL006-1	10580	0	16	0	52908	4	274	0
LCL007-1	0	0	4	0	11	3	0	0
LCL008-1	70	0	12	0	402	3	0	0
LCL009-1	17670	0	22	0	80836	4	2	0
LCL010-1	420	0	18	0	2264	2	1	0
LCL011-1	30870	0	24	0	134176	2	34	0
LCL013-1	10	0	6	0	20	2	0	0
LCL022-1	50270	0	24	0	228751	3	11	0
LCL023-1	19160	0	22	0	90467	3	1	0
LCL025-1	21510	0	16	0	124084	40	6	0
LCL027-1	10	0	8	0	120	3	0	0
LCL029-1	835700	0	18	0	4955470	569	454	0
LCL033-1	360	0	16	0	1523	2	1	0
LCL035-1	90	0	14	0	394	2	2	0
LCL041-1	60	0	8	0	380	4	0	0
LCL043-1	10	0	6	0	50	3	0	0
LCL044-1	50	0	8	0	300	3	0	0
LCL045-1	6760	0	12	0	37714	34	4	0
LCL046-1	0	0	6	0	24	2	0	0
LCL064-1	19710	0	16	0	116497	27	4	0
LCL064-2	5880	0	16	0	35906	25	3	0
LCL065-1	678350	0	20	0	4072646	342	520	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
LCL066-1	17570	0	16	0	110662	5	4	0
LCL069-1	18930	0	18	0	103563	5	78	0
LCL072-1	16300	0	18	0	90226	2	33	0
LCL076-1	790830	0	20	0	4691249	134	334	0
LCL076-2	0	0	4	0	13	3	0	0
LCL076-3	4420	0	12	0	36595	5	1	0
LCL077-1	95030	0	18	0	587117	10	31	0
LCL077-2	340	0	10	0	2868	2	1	0
LCL079-1	10	0	8	0	83	3	0	0
LCL081-1	3710	0	22	0	17928	2	3	0
LCL082-1	270	0	16	0	1380	2	1	0
LCL083-1	1242320	0	32	0	5472721	3	656	0
LCL083-2	141550	0	20	0	779192	103	17	0
LCL087-1	66540	0	26	0	332138	2	2	0
LCL091-1	955040	0	34	0	4306733	2	103	0
LCL096-1	290	0	10	0	966	2	0	0
LCL097-1	2910	0	14	0	6847	2	0	0
LCL098-1	330	0	14	0	890	2	0	0
LCL101-1	591820	0	24	0	1678885	2	19186	0
LCL102-1	29750	0	16	0	99184	2	1574	0
LCL104-1	855430	0	24	0	2609879	2	51	0
LCL106-1	40	0	10	0	217	3	0	0
LCL107-1	5770	0	20	0	12552	2	0	0
LCL108-1	245850	0	28	0	446943	2	0	0
LCL110-1	456030	0	18	0	2514280	148	271	0
LCL111-1	1100	0	12	0	6320	3	0	0
LCL117-1	20	0	10	0	118	2	0	0
LCL118-1	1680	0	18	0	8444	4	44	0
LCL120-1	730	0	16	0	2805	2	22	0
LCL126-1	40	0	10	0	174	3	1	0
LCL130-1	1180	0	18	0	3239	2	25	0
LCL132-1	592490	0	14	0	5664237	9	863	0
LCL143-1	56970	0	9	0	1841899	41	0	0
LCL169-1	0	0	3	0	8	3	0	0
LCL170-1	0	0	3	0	8	3	0	0
LCL171-1	0	0	3	0	8	3	0	0
LCL172-1	0	0	3	0	8	3	0	0
LCL173-1	0	0	3	0	8	3	0	0
LCL174-1	10	0	5	0	55	4	0	0
LCL175-1	0	0	3	0	8	3	0	0
LCL176-1	10	0	5	0	42	2	0	0
LCL177-1	20	0	7	0	116	3	0	0
LCL178-1	20	0	7	0	165	3	0	0
LCL181-2	0	0	3	0	15	3	0	0
LCL182-1	7630	0	13	0	53328	3	0	0
LCL185-1	0	0	5	0	48	2	0	0
LCL186-1	0	0	5	0	52	2	0	0
LCL187-1	30	0	7	0	235	3	0	0
LCL188-1	20	0	7	0	194	3	0	0
LCL189-1	40	0	7	0	267	3	0	0
LCL190-1	10	0	5	0	59	4	0	0
LCL192-1	230	0	9	0	1669	6	0	0
LCL193-1	50	0	7	0	431	4	0	0
LCL194-1	260	0	9	0	1986	6	0	0
LCL195-1	9400	0	13	0	66214	7	0	0
LCL196-1	146600	0	17	0	1041307	5	0	0
LCL197-1	40	0	7	0	369	5	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
LCL198-1	310430	0	17	0	2037875	5	0	0
LCL199-1	2120	0	13	0	14674	3	0	0
LCL200-1	420	0	11	0	3062	3	0	0
LCL201-1	3800	0	13	0	26563	3	0	0
LCL202-1	14150	0	15	0	94850	5	0	0
LCL203-1	3980	0	13	0	27758	3	0	0
LCL204-1	3930	0	13	0	26679	3	0	0
LCL205-1	20200	0	15	0	134159	5	0	0
LCL206-1	4160	0	13	0	27876	3	0	0
LCL207-1	730	0	11	0	5249	3	0	0
LCL208-1	39020	0	15	0	254443	4	0	0
LCL210-1	138370	0	17	0	919324	5	0	0
LCL211-1	1030	0	11	0	7539	2	0	0
LCL212-1	0	0	5	0	40	4	0	0
LCL213-1	2180	0	11	0	14401	6	0	0
LCL214-1	1670	0	11	0	11723	6	0	0
LCL215-1	8460	0	13	0	59747	7	0	0
LCL216-1	1120	0	11	0	8390	6	0	0
LCL217-1	1750	0	11	0	12262	6	0	0
LCL218-1	7650	0	13	0	51439	3	0	0
LCL226-1	0	0	5	0	51	4	0	0
LCL230-1	68120	0	15	0	414626	5	0	0
LCL230-2	0	0	4	0	22	4	0	0
LCL231-1	94820	0	15	0	560253	5	0	0
LDA003-1	182580	0	13	0	2107632	25	0	0
MSC001-1	218050	0	15	0	4004329	2540	521	0
MSC002-1	330	0	12	0	3200	3	0	0
MSC002-2	280	0	12	0	2237	3	0	0
MSC003-1	20	0	11	0	147	34	0	0
MSC005-1	20	0	10	0	267	34	0	0
NUM001-1	720	0	8	0	8456	2	0	0
NUM002-1	380	0	8	0	4461	3	0	0
NUM003-1	240	0	8	0	2825	3	0	0
NUM004-1	320	0	8	0	3583	3	0	0
NUM009-1	930	0	6	0	38369	12	1	0
NUM014-1	10	0	7	0	56	6	0	0
NUM015-1	58080	0	19	0	642455	20239	0	0
NUM016-1	1820	0	15	0	18370	517	0	0
NUM016-2	80	0	15	0	590	112	0	0
NUM019-1	0	0	4	0	32	3	0	0
NUM022-1	100	0	10	0	1430	8	0	0
NUM023-1	0	0	3	0	10	2	0	0
NUM024-1	18210	0	9	0	273250	2	8	0
NUM025-1	10	0	4	0	23	3	0	0
NUM025-2	0	0	4	0	23	3	0	0
NUM027-1	565910	0	12	0	11340051	31482	3956	0
NUM139-1	10	0	4	0	450	3	0	0
NUM180-1	6250	0	8	0	311854	16	1	0
NUM228-1	10	0	4	0	513	3	0	0
PLA001-1	1830	0	13	0	17638	1	0	0
PLA002-1	1020	0	12	0	12547	25	12	0
PLA003-1	150	0	8	0	1567	1	0	0
PLA006-1	10	0	7	0	126	4	0	0
PLA007-1	83000	0	24	0	664618	42318	0	0
PLA016-1	50030	0	23	0	379959	15520	0	0
PLA017-1	80	0	10	0	634	26	0	0
PLA019-1	81690	0	24	0	617192	25636	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
PLA020-1	0	0	5	0	31	2	0	0
PLA022-1	31800	0	16	0	170753	4530	0	0
PLA022-2	630	0	16	0	5065	310	0	0
PRV003-1	10	0	4	0	259	3	0	0
PRV005-1	40	0	5	0	945	5	0	0
PRV006-1	240	0	6	0	6553	24	0	0
PRV009-1	310	0	14	0	2583	190	0	0
PUZ001-1	20	0	15	0	209	77	0	0
PUZ002-1	10	0	12	0	78	12	0	0
PUZ003-1	20	0	12	0	201	19	0	0
PUZ004-1	0	0	11	0	73	11	0	0
PUZ008-1	0	0	3	0	16	2	0	0
PUZ008-2	650	0	13	0	5756	13	0	0
PUZ011-1	10	0	6	0	42	6	0	0
PUZ012-1	320	0	13	0	6088	0	0	0
PUZ013-1	80	0	12	0	995	0	0	0
PUZ016-2.003	11950	0	17	0	268356	9458	0	0
PUZ020-1	70	0	8	0	1458	0	0	0
PUZ022-1	460	0	9	0	5304	417	0	0
PUZ024-1	140	0	8	0	3185	3	35	0
PUZ031-1	162480	0	70	0	1244424	255959	0	0
RNG001-3	76950	0	20	0	937670	17	236	0
RNG001-4	55580	0	10	0	1400557	2	379	0
RNG002-1	13200	0	10	0	336727	5	0	0
RNG003-1	14880	0	10	0	385436	5	0	0
RNG005-2	30	0	6	0	803	3	0	0
RNG006-1	70	0	7	0	1740	3	0	0
RNG006-2	60	0	7	0	1443	3	0	0
RNG011-5	0	0	2	0	5	2	0	0
RNG023-6	100	0	7	0	985	3	0	0
RNG023-7	100	0	7	0	985	3	0	0
RNG024-6	110	0	7	0	985	3	0	0
RNG024-7	100	0	7	0	985	3	0	0
RNG037-2	70	0	7	0	1706	4	0	0
RNG038-2	30	0	7	0	724	3	0	0
RNG040-1	350	0	6	0	14326	3	0	0
RNG040-2	127110	0	9	0	4764503	3	0	0
RNG041-1	22450	0	8	0	870265	154	17	0
ROB010-1	198330	0	13	0	1596704	3	16	0
ROB013-1	10620	0	11	0	89312	3	0	0
ROB016-1	2820	0	9	0	27947	5	0	0
ROB021-1	95060	0	11	0	1162337	4	0	0
SET001-1	10	0	5	0	24	4	0	0
SET003-1	10	0	6	0	131	4	0	0
SET004-1	10	0	6	0	131	4	0	0
SET006-1	10	0	6	0	138	5	0	0
SET008-1	330	0	8	0	7794	22	31	0
SET009-1	4660	0	14	0	82659	2064	0	0
SET024-3	4810	0	7	0	277948	87	1	0
SET024-4	4610	0	7	0	265186	77	0	0
SET024-6	50	0	5	0	1974	3	0	0
SET024-7	100	0	5	0	3376	3	0	0
SET025-3	30	0	5	0	828	3	0	0
SET025-4	30	0	5	0	745	3	0	0
SET025-6	10	0	4	0	555	3	1	0
SET025-7	30	0	4	0	859	3	1	0
SET027-3	249740	0	8	0	19871168	39	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SET027-4	44860	0	8	0	3432464	39	0	0
SET027-6	18610	0	8	0	1295505	17	1	0
SET027-7	19300	0	8	0	1318699	59	1	0
SET050-6	2850	0	6	0	200299	4	1	0
SET051-6	2930	0	6	0	203008	4	1	0
SET052-6	80	0	5	0	4871	4	1	0
SET053-6	100	0	5	0	5186	4	1	0
SET054-6	0	0	3	0	33	3	0	0
SET054-7	0	0	3	0	33	3	0	0
SET055-7	10	0	4	0	34	4	0	0
SET056-6	340	0	6	0	11914	10	0	0
SET056-7	360	0	6	0	12465	11	0	0
SET057-6	280	0	6	0	8751	8	0	0
SET057-7	320	0	6	0	10443	8	0	0
SET058-6	270	0	6	0	8823	10	0	0
SET058-7	310	0	6	0	10515	10	0	0
SET059-6	250	0	6	0	8911	8	0	0
SET059-7	320	0	6	0	11471	8	0	0
SET060-6	70	0	5	0	2561	4	0	0
SET060-7	90	0	5	0	3366	4	0	0
SET061-7	140	0	6	0	5142	2	1	0
SET062-7	0	0	3	0	21	3	0	0
SET063-7	10	0	4	0	233	4	0	0
SET064-7	60	0	4	0	4009	4	0	0
SET065-6	20	0	5	0	653	2	2	0
SET065-7	30	0	5	0	900	2	2	0
SET073-7	2540	0	5	0	192699	2	0	0
SET074-7	2530	0	5	0	193267	2	0	0
SET075-7	5250	0	5	0	389802	4	0	0
SET077-6	10	0	4	0	479	3	1	0
SET077-7	10	0	4	0	673	3	1	0
SET078-6	40	0	5	0	1070	4	1	0
SET078-7	0	0	3	0	32	3	0	0
SET079-7	2960	0	5	0	212691	10	0	0
SET080-7	0	0	3	0	94	3	0	0
SET081-6	1670	0	7	0	85635	5	0	0
SET081-7	360	0	6	0	13608	3	0	0
SET090-7	1160	0	5	0	67693	15	4	0
SET093-6	288410	0	6	0	21290057	4	2	0
SET093-7	2690	0	4	0	204017	3	0	0
SET095-7	4740	0	6	0	268500	7	0	0
SET098-7	300	0	5	0	13300	4	0	0
SET101-6	11010	0	7	0	307480	22	12	0
SET101-7	170	0	5	0	5280	10	2	0
SET102-6	120	0	5	0	3560	8	0	0
SET102-7	180	0	5	0	4893	10	2	0
SET108-6	6650	0	6	0	458720	5	0	0
SET108-7	26920	0	6	0	1816332	34	18	0
SET117-6	271540	0	6	0	19833832	4	2	0
SET117-7	2950	0	4	0	217705	3	0	0
SET118-6	10	0	4	0	199	3	0	0
SET118-7	10	0	4	0	601	3	0	0
SET152-6	190280	0	7	0	14132767	9	4	0
SET153-6	191090	0	7	0	14187747	10	3	0
SET158-6	3170	0	5	0	246680	4	0	0
SET196-6	20	0	5	0	389	5	0	0
SET197-6	10	0	5	0	409	5	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SET231-6	0	0	4	0	158	4	0	0
SET234-6	45200	0	7	0	2773475	24	30	0
SET239-6	270	0	6	0	8507	4	0	0
SET240-6	3080	0	7	0	132146	5	1	0
SET241-6	3010	0	7	0	132156	5	1	0
SET242-6	4590	0	6	0	318654	4	0	0
SET252-6	4420	0	8	0	165681	16	1	0
SET253-6	4500	0	8	0	169839	16	1	0
SET296-6	0	0	2	0	8	2	0	0
SET451-6	6730	0	8	0	216096	16	3	0
SET479-6	9150	0	6	0	632701	46	2	0
SET553-6	4710	0	8	0	174439	16	2	0
SET563-6	184270	0	6	0	13608411	4	0	0
SYN003-1.006	660	0	17	0	8917	3637	0	0
SYN004-1.007	579120	0	255	0	5743649	5361045	0	0
SYN005-1.010	20	0	11	0	211	2	0	0
SYN006-1	20	0	7	0	115	1	3	0
SYN008-1	0	0	4	0	35	4	0	0
SYN009-1	10	0	7	0	121	6	0	0
SYN014-2	50	0	5	0	1558	8	0	0
SYN028-1	10	0	10	0	101	25	0	0
SYN033-1	10	0	5	0	14	1	0	0
SYN035-1	10	0	9	0	27	1	14	0
SYN040-1	0	0	3	0	15	3	0	0
SYN041-1	0	0	2	0	6	2	0	0
SYN045-1	0	0	5	0	47	11	0	0
SYN046-1	0	0	3	0	11	3	0	0
SYN047-1	10	0	7	0	109	25	0	0
SYN048-1	0	0	2	0	4	1	0	0
SYN049-1	0	0	3	0	11	1	2	0
SYN050-1	0	0	4	0	29	2	0	0
SYN055-1	60	0	15	0	531	133	0	0
SYN057-1	10	0	9	0	97	23	0	0
SYN058-1	0	0	6	0	58	8	0	0
SYN060-1	0	0	5	0	29	5	0	0
SYN061-1	0	0	5	0	14	4	0	0
SYN062-1	20	0	8	0	163	30	0	0
SYN063-1	0	0	5	0	102	5	0	0
SYN063-2	0	0	3	0	11	3	0	0
SYN064-1	0	0	2	0	4	1	0	0
SYN065-1	0	0	4	0	15	1	0	0
SYN066-1	10	0	5	0	21	1	0	0
SYN068-1	0	0	7	0	51	6	0	0
SYN069-1	2930	0	31	0	21257	7344	0	0
SYN073-1	0	0	3	0	15	1	0	0
SYN079-1	0	0	4	0	9	3	0	0
SYN080-1	10	0	3	0	15	3	0	0
SYN083-1	0	0	4	0	27	3	0	0
SYN085-1.010	0	0	12	0	33	12	0	0
SYN088-1.010	10	0	12	0	33	12	0	0
SYN089-1.002	0	0	4	0	17	4	0	0
SYN095-1.002	0	0	4	0	17	2	0	0
SYN103-1	0	0	2	0	3	2	0	0
SYN104-1	0	0	2	0	3	2	0	0
SYN105-1	0	0	3	0	6	2	0	0
SYN106-1	0	0	3	0	6	2	0	0
SYN107-1	0	0	4	0	43	1	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN108-1	0	0	4	0	48	1	0	0
SYN109-1	30	0	10	0	350	19	0	0
SYN110-1	30	0	12	0	260	20	4	0
SYN111-1	20	0	10	0	235	20	0	0
SYN112-1	0	0	4	0	48	1	0	0
SYN113-1	20	0	9	0	180	11	0	0
SYN114-1	10	0	5	0	426	1	0	0
SYN115-1	210	0	14	0	2008	94	3	0
SYN116-1	2320	0	12	0	49083	307	93	0
SYN117-1	5950	0	13	0	128063	623	206	0
SYN118-1	0	0	2	0	3	2	0	0
SYN119-1	0	0	2	0	3	2	0	0
SYN120-1	0	0	3	0	8	2	0	0
SYN121-1	20	0	13	0	142	23	0	0
SYN122-1	10	0	10	0	61	18	0	0
SYN123-1	10	0	6	0	83	2	3	0
SYN124-1	70	0	7	0	1672	13	0	0
SYN125-1	10	0	7	0	47	12	0	0
SYN126-1	10	0	6	0	163	3	0	0
SYN127-1	10	0	6	0	185	3	0	0
SYN128-1	85540	0	16	0	1777858	18035	2563	0
SYN129-1	85480	0	16	0	1777858	18035	2563	0
SYN130-1	0	0	2	0	3	1	0	0
SYN131-1	0	0	2	0	3	1	0	0
SYN132-1	0	0	2	0	3	1	0	0
SYN133-1	0	0	2	0	3	1	0	0
SYN134-1	10	0	5	0	16	4	0	0
SYN135-1	10	0	5	0	275	2	0	0
SYN136-1	20	0	5	0	271	2	0	0
SYN137-1	14260	0	38	0	93183	6024	959	0
SYN138-1	160	0	19	0	960	113	8	0
SYN139-1	75920	0	35	0	555181	44573	4755	0
SYN140-1	76570	0	35	0	555267	44572	4743	0
SYN141-1	300	0	15	0	3333	108	5	0
SYN144-1	40	0	13	0	309	30	0	0
SYN145-1	0	0	2	0	3	1	0	0
SYN146-1	0	0	3	0	25	1	0	0
SYN147-1	0	0	3	0	24	2	0	0
SYN148-1	10	0	9	0	62	11	0	0
SYN149-1	10	0	3	0	24	1	0	0
SYN150-1	0	0	4	0	36	2	0	0
SYN151-1	0	0	4	0	33	2	0	0
SYN152-1	0	0	4	0	36	2	0	0
SYN153-1	0	0	5	0	185	3	0	0
SYN154-1	10	0	5	0	224	3	0	0
SYN157-1	12550	0	14	0	223773	3025	788	0
SYN158-1	4770	0	14	0	90966	1566	65	0
SYN159-1	4740	0	14	0	90966	1566	65	0
SYN160-1	21420	0	14	0	428325	5449	537	0
SYN161-1	9750	0	14	0	189449	2501	147	0
SYN162-1	9710	0	14	0	189449	2502	147	0
SYN164-1	0	0	2	0	3	2	0	0
SYN165-1	0	0	3	0	29	1	0	0
SYN166-1	0	0	3	0	29	1	0	0
SYN167-1	0	0	3	0	29	1	0	0
SYN168-1	10	0	5	0	153	2	0	0
SYN169-1	10	0	5	0	140	2	1	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN170-1	40	0	6	0	857	1	0	0
SYN171-1	76220	0	33	0	1279441	48466	6940	0
SYN172-1	0	0	2	0	3	2	0	0
SYN173-1	0	0	4	0	11	2	0	0
SYN174-1	0	0	4	0	11	2	0	0
SYN175-1	10	0	4	0	88	2	1	0
SYN176-1	380	0	10	0	3515	125	2	0
SYN177-1	30	0	7	0	590	12	0	0
SYN178-1	4250	0	15	0	32223	690	366	0
SYN181-1	4410	0	16	0	35690	708	368	0
SYN182-1	10	0	7	0	204	3	0	0
SYN183-1	10	0	7	0	272	4	0	0
SYN184-1	0	0	2	0	3	2	0	0
SYN185-1	0	0	2	0	3	2	0	0
SYN186-1	0	0	6	0	47	5	0	0
SYN187-1	0	0	6	0	77	5	0	0
SYN188-1	10	0	6	0	71	5	0	0
SYN189-1	10	0	8	0	54	7	0	0
SYN190-1	165040	0	24	0	1324532	43992	9041	0
SYN191-1	40	0	10	0	258	23	0	0
SYN192-1	40	0	13	0	526	20	0	0
SYN193-1	40	0	13	0	686	20	0	0
SYN194-1	280	0	15	0	3335	114	5	0
SYN195-1	120	0	19	0	1548	44	12	0
SYN196-1	0	0	5	0	22	2	0	0
SYN197-1	0	0	3	0	7	3	0	0
SYN198-1	0	0	5	0	19	2	0	0
SYN199-1	0	0	5	0	28	2	0	0
SYN200-1	10	0	5	0	22	2	0	0
SYN201-1	10	0	9	0	150	20	0	0
SYN202-1	22870	0	18	0	152525	7462	1391	0
SYN203-1	80	0	12	0	1318	54	2	0
SYN204-1	14830	0	26	0	106814	8663	911	0
SYN205-1	14690	0	26	0	106814	8663	911	0
SYN206-1	20	0	14	0	293	39	0	0
SYN207-1	14070	0	26	0	90335	5120	914	0
SYN208-1	50	0	12	0	358	17	1	0
SYN209-1	30	0	8	0	459	3	0	0
SYN210-1	10	0	8	0	175	3	0	0
SYN211-1	10	0	8	0	183	3	3	0
SYN212-1	20	0	8	0	263	3	0	0
SYN213-1	1380	0	24	0	18609	507	43	0
SYN215-1	500	0	20	0	5123	219	42	0
SYN216-1	0	0	6	0	35	2	0	0
SYN217-1	10	0	7	0	76	10	0	0
SYN218-1	0	0	6	0	53	1	0	0
SYN219-1	20	0	13	0	142	23	0	0
SYN220-1	20	0	6	0	643	4	0	0
SYN221-1	70	0	7	0	1375	12	0	0
SYN222-1	70	0	7	0	1620	8	3	0
SYN223-1	20	0	6	0	527	6	0	0
SYN224-1	70	0	7	0	1672	12	0	0
SYN225-1	10	0	7	0	41	8	0	0
SYN226-1	20	0	6	0	632	8	0	0
SYN227-1	0	0	7	0	39	10	0	0
SYN228-1	0	0	6	0	34	8	0	0
SYN229-1	70	0	7	0	1487	9	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN230-1	60	0	7	0	1487	10	0	0
SYN231-1	70	0	7	0	1577	6	3	0
SYN232-1	60	0	7	0	1566	7	3	0
SYN233-1	20	0	6	0	566	7	0	0
SYN234-1	10	0	7	0	46	9	0	0
SYN235-1	10	0	7	0	43	11	0	0
SYN236-1	20	0	6	0	558	9	0	0
SYN237-1	0	0	3	0	30	1	0	0
SYN238-1	0	0	3	0	29	1	0	0
SYN239-1	0	0	3	0	29	1	0	0
SYN240-1	0	0	3	0	29	1	0	0
SYN241-1	0	0	3	0	29	2	0	0
SYN242-1	0	0	3	0	27	3	0	0
SYN243-1	10	0	4	0	13	3	0	0
SYN244-1	0	0	3	0	30	1	0	0
SYN245-1	0	0	3	0	30	1	0	0
SYN246-1	0	0	4	0	22	4	0	0
SYN247-1	0	0	3	0	28	1	0	0
SYN248-1	10	0	7	0	64	6	0	0
SYN249-1	10	0	7	0	60	8	0	0
SYN250-1	140	0	14	0	891	152	9	0
SYN251-1	0	0	4	0	13	3	0	0
SYN253-1	77300	0	35	0	555267	44572	4743	0
SYN254-1	15060	0	36	0	107293	8815	911	0
SYN255-1	0	0	4	0	38	2	0	0
SYN256-1	0	0	4	0	36	2	0	0
SYN257-1	0	0	2	0	3	2	0	0
SYN258-1	10	0	3	0	29	1	0	0
SYN259-1	10	0	3	0	28	1	0	0
SYN260-1	0	0	3	0	28	1	0	0
SYN261-1	0	0	3	0	29	1	0	0
SYN262-1	0	0	6	0	47	5	0	0
SYN263-1	20	0	7	0	124	5	3	0
SYN264-1	10	0	7	0	49	7	0	0
SYN265-1	10	0	5	0	144	2	0	0
SYN266-1	340	0	13	0	5414	157	0	0
SYN267-1	40	0	6	0	900	1	0	0
SYN268-1	40	0	6	0	901	1	0	0
SYN270-1	40	0	12	0	280	33	0	0
SYN271-1	76800	0	33	0	1279379	48465	6940	0
SYN272-1	90	0	11	0	800	45	6	0
SYN273-1	170	0	12	0	1542	56	0	0
SYN274-1	0	0	2	0	3	1	0	0
SYN275-1	0	0	2	0	3	1	0	0
SYN276-1	0	0	2	0	3	2	0	0
SYN277-1	0	0	3	0	32	1	0	0
SYN278-1	0	0	3	0	33	1	0	0
SYN279-1	0	0	4	0	17	1	0	0
SYN280-1	10	0	3	0	35	1	0	0
SYN281-1	0	0	3	0	32	1	0	0
SYN282-1	0	0	3	0	34	1	0	0
SYN283-1	10	0	4	0	11	2	0	0
SYN284-1	0	0	4	0	11	2	0	0
SYN285-1	10	0	7	0	50	9	0	0
SYN286-1	10	0	4	0	15	1	0	0
SYN287-1	10	0	3	0	35	1	0	0
SYN288-1	0	0	3	0	34	2	0	0

Nombre	*Tiempo*	*RA ref*	*RE ref*	*RA árbol*	*RE árbol*	*Poda base*	*Poda RG*	*Poda IA*
SYN289-1	10	0	4	0	11	2	0	0
SYN290-1	10	0	3	0	32	1	0	0
SYN291-1	0	0	3	0	34	1	0	0
SYN292-1	0	0	3	0	8	3	0	0
SYN293-1	10	0	6	0	37	8	0	0
SYN294-1	10	0	6	0	43	10	0	0
SYN295-1	0	0	3	0	32	2	0	0
SYN296-1	10	0	4	0	11	2	0	0
SYN297-1	0	0	3	0	32	2	0	0
SYN298-1	20	0	14	0	293	38	1	0
SYN299-1	20	0	14	0	293	39	0	0
SYN300-1	20	0	14	0	293	39	0	0
SYN301-1	10	0	8	0	161	3	0	0
SYN318-1	10	0	4	0	22	2	1	0
SYN319-1	20	0	5	0	150	1	0	0
SYN325-1	0	0	4	0	42	1	0	0
SYN326-1	10	0	5	0	66	1	5	0
SYN331-1	120	0	6	0	880	1	3	0
SYN333-1	40	0	9	0	193	1	84	0
SYN336-1	0	0	2	0	10	1	0	0
SYN338-1	0	0	2	0	6	1	0	0
SYN339-1	0	0	2	0	4	1	0	0
SYN340-1	0	0	2	0	4	1	0	0
SYN341-1	0	0	2	0	4	1	0	0
SYN346-1	0	0	3	0	15	1	0	0
TOP002-2	0	0	3	0	6	3	0	0
TOP004-1	0	0	2	0	47	1	0	0
TOP004-2	0	0	2	0	22	1	0	0

Tabla 20: Resultados temporales y espaciales del SLT vía EMPTY

Por último la tabla que sigue incluye los resultados temporales obtenidos por el procedimiento SLT vía SUBOP para el conjunto de problemas cuya elección de ancestros subóptima difiere de la elección ALL.

Problema	*Tiempo*
COL002-2	90
COL002-3	40
COM001-1	10
COM002-1	150
COM002-2	150
COM003-2	120
GRA001-1	1460
GRP003-2	760
GRP004-2	40
GRP031-2	20
GRP041-2	0
GRP042-2	10
GRP043-2	20
GRP044-2	10
GRP045-2	170
GRP046-2	50
GRP047-2	1250
GRP123-6.003	157950
GRP123-7.003	157900
GRP123-8.003	157620
GRP123-9.003	158950
GRP124-6.003	428250
GRP124-7.003	421050
GRP124-8.003	423320
GRP126-1.002	390
GRP126-2.002	390
GRP126-3.002	400
GRP126-4.002	10930
GRP131-1.002	421200
GRP131-2.002	429150
GRP132-1.002	385980
GRP132-2.002	389500
GRP135-1.002	182870
GRP135-2.002	184920
LCL169-1	0
LCL170-1	0
LCL171-1	0
LCL172-1	0
LCL173-1	0
LCL174-1	10
LCL175-1	0
LCL176-1	0
LCL177-1	20
LCL178-1	30
LCL181-2	0
LCL182-1	5550
LCL185-1	10
LCL186-1	10
LCL187-1	40
LCL188-1	40

Problema	*Tiempo*
LCL189-1	40
LCL190-1	10
LCL192-1	240
LCL193-1	60
LCL194-1	290
LCL195-1	7090
LCL196-1	43660
LCL197-1	50
LCL198-1	112260
LCL199-1	1400
LCL200-1	350
LCL201-1	2490
LCL202-1	7420
LCL203-1	2740
LCL204-1	2650
LCL205-1	11430
LCL206-1	2770
LCL207-1	580
LCL208-1	21750
LCL210-1	64180
LCL211-1	870
LCL212-1	0
LCL213-1	1900
LCL214-1	1440
LCL215-1	6000
LCL216-1	920
LCL217-1	1540
LCL218-1	5070
LCL226-1	10
LCL230-1	35280
LCL230-2	0
LCL231-1	53510
MSC001-1	107220
MSC002-1	390
MSC002-2	300
NUM014-1	0
NUM015-1	510
NUM016-2	40
NUM022-1	60
PLA001-1	2180
PLA002-1	1370
PRV009-1	450
PUZ001-1	10
PUZ002-1	10
PUZ003-1	10
PUZ004-1	10
PUZ009-1	50
PUZ011-1	10
PUZ012-1	500
PUZ013-1	40

Problema	*Tiempo*
PUZ014-1	440
PUZ015-2.003	2140
PUZ016-2.003	560
PUZ022-1	470
PUZ023-1	114500
PUZ024-1	140
PUZ025-1	106753
PUZ029-1	190
PUZ031-1	87100
SET001-1	0
SET003-1	10
SET004-1	10
SET006-1	10
SET008-1	390
SET009-1	14990
SYN003-1.006	740
SYN004-1.007	14260
SYN005-1.010	20
SYN006-1	20
SYN008-1	0
SYN009-1	10
SYN011-1	100
SYN028-1	0
SYN029-1	10
SYN030-1	20
SYN032-1	50
SYN035-1	10
SYN040-1	0
SYN041-1	0
SYN044-1	10
SYN045-1	0
SYN046-1	0
SYN047-1	10
SYN048-1	0
SYN049-1	10
SYN050-1	0
SYN051-1	10
SYN052-1	10
SYN053-1	20
SYN054-1	30
SYN055-1	30
SYN057-1	10
SYN058-1	10
SYN060-1	10
SYN061-1	10
SYN062-1	10
SYN064-1	0
SYN065-1	10
SYN066-1	10
SYN068-1	10

Problema	*Tiempo*
SYN069-1	580
SYN070-1	53400
SYN085-1.010	0
SYN088-1.010	10
SYN089-1.002	10
SYN095-1.002	0
SYN098-1.002	200
SYN103-1	0
SYN104-1	0
SYN105-1	0
SYN106-1	0
SYN107-1	0
SYN108-1	10
SYN109-1	40
SYN110-1	40
SYN111-1	30
SYN112-1	0
SYN113-1	20
SYN114-1	20
SYN115-1	210
SYN116-1	2430
SYN117-1	6370
SYN118-1	0
SYN119-1	0
SYN120-1	0
SYN121-1	20
SYN122-1	10
SYN123-1	10
SYN124-1	60
SYN125-1	10
SYN126-1	10
SYN127-1	0
SYN128-1	93460
SYN129-1	91860
SYN130-1	10
SYN131-1	0
SYN132-1	0
SYN133-1	0
SYN134-1	0
SYN135-1	10
SYN136-1	10
SYN137-1	15310
SYN138-1	120
SYN139-1	16850
SYN140-1	16930
SYN141-1	300
SYN144-1	40
SYN145-1	0
SYN146-1	0
SYN147-1	0
SYN148-1	10
SYN149-1	0
SYN150-1	0
SYN151-1	0
SYN152-1	0
SYN153-1	10
SYN154-1	10

Problema	*Tiempo*
SYN157-1	10300
SYN158-1	5230
SYN159-1	4810
SYN160-1	17770
SYN161-1	8150
SYN162-1	8110
SYN164-1	0
SYN165-1	10
SYN166-1	0
SYN167-1	10
SYN168-1	0
SYN169-1	10
SYN170-1	40
SYN171-1	51420
SYN172-1	0
SYN173-1	0
SYN174-1	0
SYN175-1	0
SYN176-1	420
SYN177-1	40
SYN178-1	4580
SYN181-1	2230
SYN182-1	20
SYN183-1	20
SYN184-1	0
SYN185-1	0
SYN186-1	0
SYN187-1	10
SYN188-1	0
SYN189-1	10
SYN190-1	188070
SYN191-1	40
SYN192-1	30
SYN193-1	50
SYN194-1	300
SYN195-1	870
SYN196-1	0
SYN197-1	0
SYN198-1	0
SYN199-1	0
SYN200-1	10
SYN201-1	10
SYN202-1	24630
SYN203-1	90
SYN204-1	16530
SYN205-1	16590
SYN206-1	30
SYN207-1	15310
SYN208-1	40
SYN209-1	30
SYN210-1	20
SYN211-1	20
SYN212-1	20
SYN213-1	298450
SYN215-1	530
SYN216-1	10
SYN217-1	10

Problema	*Tiempo*
SYN218-1	0
SYN219-1	20
SYN220-1	40
SYN221-1	70
SYN222-1	80
SYN223-1	30
SYN224-1	80
SYN225-1	10
SYN226-1	20
SYN227-1	0
SYN228-1	0
SYN229-1	70
SYN230-1	80
SYN231-1	70
SYN232-1	70
SYN233-1	30
SYN234-1	10
SYN235-1	10
SYN236-1	30
SYN237-1	0
SYN238-1	0
SYN239-1	0
SYN240-1	0
SYN241-1	0
SYN242-1	0
SYN243-1	0
SYN244-1	0
SYN245-1	0
SYN246-1	10
SYN247-1	0
SYN248-1	20
SYN249-1	20
SYN250-1	150
SYN251-1	0
SYN253-1	17030
SYN254-1	17000
SYN255-1	0
SYN256-1	0
SYN257-1	0
SYN258-1	0
SYN259-1	0
SYN260-1	0
SYN261-1	0
SYN262-1	0
SYN263-1	20
SYN264-1	10
SYN265-1	10
SYN266-1	1600
SYN267-1	40
SYN268-1	40
SYN270-1	40
SYN271-1	50720
SYN272-1	110
SYN273-1	180
SYN274-1	0
SYN275-1	0
SYN276-1	0

Problema	*Tiempo*
SYN277-1	0
SYN278-1	10
SYN279-1	0
SYN280-1	10
SYN281-1	0
SYN282-1	0
SYN283-1	0
SYN284-1	0
SYN285-1	10
SYN286-1	0
SYN287-1	0
SYN288-1	0
SYN289-1	0
SYN290-1	0
SYN291-1	0

Problema	*Tiempo*
SYN292-1	0
SYN293-1	0
SYN294-1	10
SYN295-1	0
SYN296-1	0
SYN297-1	0
SYN298-1	20
SYN299-1	30
SYN300-1	30
SYN301-1	10
SYN315-1	20
SYN318-1	0
SYN319-1	20
SYN321-1	20
SYN323-1	20

Problema	*Tiempo*
SYN325-1	10
SYN326-1	10
SYN333-1	40
SYN336-1	10
SYN338-1	0
SYN339-1	0
SYN340-1	0
SYN341-1	0
SYN346-1	0
TOP001-2	42880
TOP002-2	0
TOP004-1	0
TOP004-2	0
TOP005-2	527200

Tabla 21: Resultados temporales del SLT vía SUBOP