Jornada de Seguimiento de Proyectos en Tecnologías InformÃ<ticas, 2003
Programa Nacional de Tecnologías de la Información y las Comunicaciones

STREAM: formal SofTwaRE tools: A Multi–paradigm approach TIC2001-2705-C03

María Alpuente * — T. U. Valencia
Ernesto Pimentel † — U. Málaga
Ginés Moreno ‡ — U. Castilla–La Mancha

Abstract

The demand for high–quality, reliable software has grown in recent years far faster than the technology for producing it. The increasing complexity of such systems and the lack of adequate science and technology to support robust development typically lead to software that is fragile, unreliable, and extremely difficult and labor–intensive to develop, test, and evolve. The STREAM project focuses on developing formal methods, tools and techniques to support the development and management of high–quality software. We use multi–paradigm declarative languages as a tangible means to develop these methods. The formal basis underlying the declarative technology guarantees the correctness and effectiveness of the developed tools and techniques, giving support to automated program analysis, verification, debugging, learning, optimization and transformation.

Start Date	Status	Duration
January 1, 2002	2nd year, running	36 months

Keywords *Software reliability, formal methods, multi–paradigm (declarative) programming, components technology, advanced programming environments, program analysis, specification, verification, debugging, learning, and transformation.*

1 Project Overview

1.1 Baseline

The explosive growth of information technology has fuelled an unprecedent demand for new software that far outweighs the existing technology to produce it. The painful inadequacy of current technology results in large complex software systems whose behavior is not well understood and which often fail in unpredicted ways. The STREAM project is committed to developing methods, techniques and tools that are needed to build high-quality, reliable systems which are evolvable, maintainable, and cost-effective. The thesis of project is that declarative technologies can play a significant role in these tasks, as they are widely recognized as a useful means for providing automated support for robust development of software which

*Email: alpuente@dsic.upv.es
†Email: ernesto@lcc.uma.es
‡Email: Gines.Moreno@uclm.es

is more maintainable and enhanceable over time. In accordance with component-based software development blueprint, the technologies that support analysis, specification, validation, modelling, composition, and optimization of software components are specifically pursued.

1.2 Objectives

In order to present an overall view of the project, we briefly recall the well-known software trilogy, which we describe as follows. In addition to programs themselves, software development involves artifacts such as properties (e.g., specifications, partial correctness properties, and types) and data (e.g., test cases, scenarios, or examples). When we put programs, properties and data at the vertices of a triangle, the edges represent the various processes used to produce one element from another (Figure 1). The STREAM project explores the program-property-data triangle with the aim of automating the corresponding phases of the software process. Specifically, the project investigates:

- theories, languages, methods and tools to support automated analysis, specification, verification, debugging, learning, optimization and transformation of software (components.)
- component–based software modelling and analysis techniques; techniques for assembling provably reliable components into predictably reliable systems.

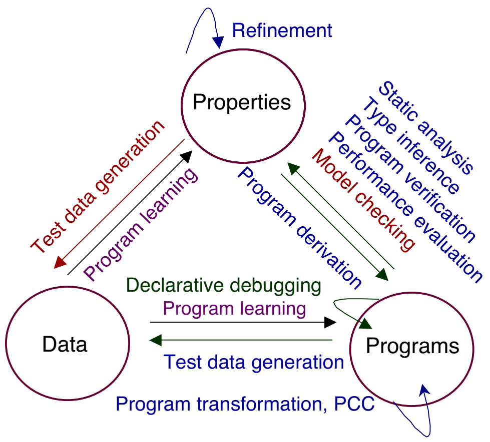

Figure 1: STREAM: Overall view of the project

1.3 Work Themes

To achieve the project objectives, we investigate the four tasks[1] which we describe below:

1. Modelling and analysis of software components (UPV,UMA)
2. Verification, model checking and abstract interpretation (UPV,UMA)

[1] We enclose the partners involved in each task in parenthesis.

3. Performance debugging and optimization (UPV,UMA,UCLM)

4. Declarative debugging and program learning (UPV,UMA)

A number of activities are carried out in cooperation by two of the partners, and thus the complementary expertise of the three teams plays an important role in the project STREAM. The group of Valencia has traditionally worked on semantics–based frameworks for declarative debugging, program learning, strategic programming, formal efficiency measurement, and efficiency improvements by program transformation. The group in Málaga has the necessary expertise in specification and interaction languages, linear logic, type systems, software architectures, and abstract model checking. Finally, the UCLM has wide experience in optimizing compilation and program optimization by partial evaluation and fold/unfold transformation.

Task 1: Modelling and analysis of software components

The high cost and high failure rate of present software projects call for better software development and maintenance technologies. In recent years, incremental and component-based development have been proposed as (separate or combined) remedies to reduce development time and costs, and to increase software quality, especially usability and reliability. However, there are few validated technologies in these areas in industrial use today, reflecting the immaturity of these technologies.

The last decade has seen the emergence of a class of CBSD models and languages variously termed "coordination languages", "configuration languages", and "architectural description languages", which provide a clean separation between individual software components and their interaction within the overall software organization. This separation makes large applications more tractable, supports global analysis, and enhances reuse of software. A number of other formalisms (behavioral types, process algebras, ...) also present good features to model the interaction exhibited by software components.

Strategic programming (i.e., programming with a programmable strategy-guided evaluation mechanism) provides a clean and flexible interface for the (often) necessary participation of the programmer for controlling and improving program execution. A number of programming languages (e.g., CafeOBJ, ELAN, Maude, and Stratego permit (to some extent) the *explicit* specification of evaluation strategies. Other programming languages such as Haskell have a predefined computational strategy whose behavior can be modified (to some extent) by means of program annotations. Programmable strategies can be thought of as as a kind of 'non-declarative' facility, and so extra formal support for the analysis of the program behavior, as well as for guaranteeing key semantic properties such as termination and confluence, is dramatically required.

The primary goal of the project in this area is to advance the state-of-the-art for component-based software development and strategic programming. This includes research in:

- Specification, refinement, and analysis of software components: composing specifications.
- Coordination, architectural, and interface definition languages: implementation, inter–operability, heterogeneity.
- Semantic models and foundations for coordination: component composition, concurrency, dynamic aspects, formal models for interacting agents.

- Type systems, linear logic and mobility.
- Techniques for analyzing and ensuring program properties under program strategies.

Task 2: Verification, model checking and abstract interpretation

Program verification aims at proving that programs meet their specifications, i.e., that the actual program behavior coincides with the desired one. Model checking is a specific approach to the verification of temporal properties of reactive and concurrent systems, which has proven to be particularly successful in the area of finite-state programs. Abstract interpretation is a method for designing and comparing semantics of programs expressing various types of programs properties; it has been very successfully used to infer run-time program properties that can be valuable to optimize programs. Clearly, among these three methods, there are similarities concerning their goals and their application domains. With the growing need for formal methods to reason about complex, infinite-state, and embedded systems, hybrid methods that combine the three areas are bound to be of great importance. The main goal of this task is the cross-fertilization and advancement of hybrid methods focusing on:

- program verification
- static analysis techniques
- model checking
- type systems
- security analyses and program certification; website verification

Task 3: Performance debugging and optimization

In this task, our goal is the design of a performance debugging framework to locate and improve inefficient uses of time, space, or non–determinism, e.g., by the use of profiling and transformation techniques. We are particularly interested in trace–based debugging, where we put the main emphasis on a better representation of program traces w.r.t. the (non–strict) operational semantics of input programs. We intend to develop a prototypical implementation of these frameworks and integrate them into one of the existing implementations of Curry, a modern multi-paradigm language that supports functional, logic, concurrent, and distributed programming. Curry provides specific constructs to support programming-in-the-large, such as modules, polymorphic types, and higher-order functions. The advantage of these and the other features of Curry for the development of realistic applications (including distribution, graphical user interfaces, and web-based interfaces) are well known in the field.

With regard to performance debugging, a specific problem is posed by the use of automatic program optimization techniques. In general, it is well-known that automatic optimization tools (e.g., partial evaluators) often introduce redundancies in the generated code that programmers usually (or ideally) do not write, and the identification of useless code (such as redundant arguments or dead rules) deserves great research effort in the project. We are also interested in fold/unfold transformation systems (guided by automatic transformation strategies) for optimizing programs, as well as in different optimizations of the language implementation. As for the rules+strategies approach, we especially focus on the tupling strategy, which is able to reduce (in some cases) the algorithmic complexity of a given program from exponential to linear complexity. In the context of linear logic based languages, similar techniques based on *frames* and *resources tagging* have presented very good performances. Concerning the languages implementation, we are interested not only in improving the basic operational mechanisms of the languages but also in developing novel transformation techniques which are able to increase the efficiency of the compiled code. This points out the lack of formal frameworks to measure, in a realistic way, the cost of program computations and, thus, the improvement achieved by the program optimization techniques. Concerning this problem, we plan to investigate different cost criteria for measuring the speed-ups achieved by program transformations, which are independent of the specific language implementation.

In summary, the different problems that we tackle in this task follow three main guidelines:

- Performance debugging tools and techniques;
- Program transformation, removal of useless code, and optimizations of the compiler;
- Tools and techniques for analyzing the effectiveness of program optimization.

Task 4: Declarative debugging and program learning

Finding bugs in programs is an old problem in software construction. It is well known that simple debuggers based on the analysis of traces of target programs provide only very limited support. On the other hand, the operational semantics of multi–paradigm languages is more complex than in other languages due to the combination of advanced features like concurrency, "don't know" and "don't care" non-determinism, etc. Hence, there is an urgent need for appropriate debugging tools to support the efficient development of large multi-paradigm programs.

Some recent approaches for the debugging of logic programs have advocated the use of inductive techniques in order to correct bugs. In particular, once a bug has been detected in a program, the inductive techniques allow one to correct it by deriving a new program that is consistent with its specification. The task of revision involves changing the answer set of the given theory by adding previously missing answers or by removing incorrect ones. The synergy between inductive and deductive synthesis techniques is extremely fruitful in this context.

To summarize, while trace–based methods and performance debugging are considered in Task 3 above, in this task, we want to investigate novel techniques for the declarative debugging of multi–paradigm declarative programs. The set of techniques considered here includes:

- declarative diagnosis; abstract diagnosis
- error correction by program learning
- inductive and deductive program synthesis

1.4 Project Organization

Here is a short outline of the project structure with the expected contributions from each site.

Subproject 1: UPV

- **Module M1.1:** Program optimization and profiling techniques
  Profiling techniques (Task T1.1.1), Effectiveness of optimization tools (T1.1.2), Profiling tools (T1.1.3), Tools for analyzing the effectiveness of program optimization (T1.1.4), Integration of results (T1.1.5).
- **Module M1.2:** Modelling and semantic optimization of software components.
  Specification frameworks and its application to declarative languages (T1.2.1), Model checking techniques for declarative languages (T1.2.2), Model declarative debugging techniques for declarative languages (T1.2.3), Analysis of programmable strategies (T1.2.4), Analysis and removal of redundant code (T1.2.5).
- **Module M1.3:** Multi-agent systems: reconfigurable nets and cooperative automata. Formalization of reconfigurable nets (T1.3.1), Formalization of cooperative automata (T1.3.2), Modelling and validation tools for concurrent, multi–agent processes (T1.3.3), Experimental evaluation (T1.3.3).
- **Module M1.4:** Induction in Software Engineering and Databases.
  Functional logic programming with types (T1.4.1), Theory evaluation (T1.4.2), Incrementality and classification problems (T1.4.3), Induction and rational debugging (T1.4.4), Automated decision systems (T1.4.5), Higher order learning (T1.4.6), Integration in the FLIP system (T1.4.7).

Subproject 2: UMA

- **Module M2.1:** Composing specifications in a multi–paradigm environment.
  Composing view points (T2.1.1), Composing specifications (T2.1.2), Composition and consistency of UML view points (T2.1.3), Composition and consistency of RM-ODP view points (T2.1.4), Extending interface description languages by means of coordination languages (T2.1.5), Abstract model checking (new task after the incorporation of M. Mar Gallardo —T2.1.6).
- **Módulo M2.2**: Computational Interpretation of Pure Type Systems (PTSs).
  Relationships of Extended PTSs and Linear Logic (T2.2.1), Extended PTSs, Linear Logic and Mobility (T2.2.2), PTSs as a kernel for functional languages (T2.2.3).

Subproject 3: UCLM

- **Module M3.1:** Optimization of multi–paradigm declarative languages.
  Transformation algorithms for uniform programs (T3.1.1), Implementation of a compiler kernel for functional logic programs (T3.1.2).
- **Module M3.2:** Automatic transformation of multi-paradigm declarative programs.
  Automatic tupling for multi-paradigm declarative programs (T3.2.1), Integration of automatic transformation tools (T3.2.2).

2 Project achievements

During 2002–2003, significant progress has been made towards achieving the project objectives. The activities of the different modules have progressed according to workplan. STREAM results are published at mainstream technical meetings on programming languages, formal methods, and declarative programming (e.g. ESOP, AMAST, RTA, PPDP, CADE, ICALP, LOPSTR, LPAR, FLOPS, ECML, SAS, JELIA, RULE, WRLA, and CSL), international journals (e.g. *Information and Computation, Theoretical Computer Science, Journal of Logic and Computation, IEEE Transactions of Software Engineering, Software Tools for Technology Transfer, New Generation Computing, and Theory and Practice of Logic Programming*), newsletters (*AI Communications*), and workshops. This section summarizes these results. We include a selection of publications representative of STREAM for each individual site. The complete bibliography as well as the software developed in the STREAM project is available via WWW.

In the area of CBSD, significative progress has been made in different contexts. On the one hand, Maude has been proposed as a meta-language to describe different viewpoints in RM-ODP. In particular, the enterprize and information views have been successfully described in Maude. Analysis and verification tools have also been developed for the language, such as a model checker, a termination tool, and an extension which deals with strategy annotations. On the other hand, different formalisms have been studied in order to complement the information provided by interface description languages. In this sense, the expressive power of Linda and process algebras have been extensively explored. A formal methodology for automated component adaptation has also been proposed. In the area of programming language design, some relevant proposals have been made: the Expansion Postponement and Cut Elimination problems have been analyzed for a family of pure type systems, and the tag-frame system has been developed to provide a strategy for resource management in linear logic. This strategy will permit the construction of an abstract machine for some linear logic–based programming languages.

Concerning the abstract interpretation, verification and optimization areas, work has been done at many levels. We are implementing different prototypes and tools such as a parallel strategy for linear computations, a program slicer for Curry and a model checker for `tccp` programs. Several efforts have also been completed in some optimizations based on program analysis and program transformation. We have finished the implementation of a fold/unfold transformation system which is able to perform composition and tupling (semi–)automatically. We also implemented a tool for removing redundant arguments from functional programs. Finally, we defined several optimizations of demand–driven narrowing and needed narrowing. As for the areas of debugging and learning, work has proceeded with the investigation of several of the planned techniques and tools, including an abstract debugger and a program corrector for OBJ as well as integrated programs, and an experimental for inductive learner.

We find it useful to list the project's achievements by following the formal organization of STREAM into its three different sub-projects, one per partner.

2.1 Contributions of each Site

2.1.1 Technical University of Valencia (UPV)

According to work plan, UPV is currently working in the four tasks of “Modeling and analysis of software components”, “Verification, model checking and abstract interpretation”, “Performance debugging and optimization”, and “Declarative debugging and program learning (UPV)”. Work has progressed well in all these areas and many of the objectives have already been achieved.

- Task 1: Modeling and analysis of software components
  - Formalization of a framework for the modeling and analysis of programs that use strategy annotations.
  - Development of tools and techniques for achieving direct proofs of termination of context-sensitive rewriting that also apply to Maude and OBJ programs.
  - Development of techniques for the introduction of demanded computations in eager languages such as Maude and OBJ by using transformations and strategy annotations. A tool is being developed (in cooperation with UMA team) to check the termination of Maude specifications.
  - Development of a software tool for the definition of multi-agent concurrent systems and the validation of the most relevant properties using the model of extended cooperating automata.
  - Development of an automatic tool for the definition and validation of concurrent systems subject to structural dynamic changes using the model of reconfigurable nets.
- Task 2: Verification, model checking and Abstract Interpretation
  - Formalization of a model checking algorithm for tccp programs. The advantages of the cc paradigm are exploited to mitigate the state explosion problem which is common to model checking algorithms.
  - Development of a symbolic model checking algorithm for the tccp language which improves the traditional algorithm by using a symbolic data structure based on Difference Decision Diagrams.
  - Definition of a fully abstract denotational semantics for the tccp paradigm which allows one to develop different analyses for tccp programs. The original semantics of tccp is not fully abstract, hence it can only be used to partially approximate analysis results. In cooperatio with some UMA team members, we are developing an abstract tccp model checker.
  - Construction of a model for hybrid cc programs which can be used for the formulation of an automatic verification method for hybrid systems.
  - Formal description of a term-rewriting, web specification language, which is helpful to express properties about the contents and structure of a given web site. Formalization of a simulation-based verification framework which can be used to verify a given web site w.r.t. a web specification.
- Task 3: Performance debugging and optimization
  - Formalization of a natural semantics for functional logic languages —and its equivalent small-step semantics— as a basis to develop a theoretical framework to estimate the efficiency achieved by multi-paradigm program transformations.
  - Development of the first partial evaluation system for functional logic programs enhanced with cost information for assessing the improvement achieved by each specialized recursive function.

  - Development of a program slicing framework, based on partial evaluation, which is useful for performing debugging tasks, code reuse, etc.
  - Formalization of a notion of redundant argument in functional programs. Development of a tool for analyzing and removing redundant arguments.

- Task 4: Declarative debugging and program learning
  - Formalization of a generic scheme for the declarative debugging of functional logic programs which is based on a (continuous) immediate consequence operator $T_{\mathcal{R}}$ which models computed answers, and is valid for eager as well as lazy programs. We also develop an effective debugging methodology which is based on abstract interpretation. The debugger does not require the user to either provide error symptoms in advance or answer any question concerning program correctness.
  - We endow the abstract diagnoser with a new methodology for synthesizing correct functional logic programs. We propose a hybrid, top-down (unfolding–based) as well as bottom-up (induction–based), approach for the automatic correction of the wrong code, which is driven by a set of evidence examples that are automatically produced as an outcome by the diagnoser. We also provide a prototypical implementation which we use for an experimental evaluation of the system.
  - Development of an (abstract) declarative diagnosis tool and an inductive learning methodology for repairing bugs in OBJ programs. The algorithm is not tied to any particular rewriting strategy, which makes it suitable for correcting errors in OBJ programs even if they are given lazy strategy annotations, as in OnDemandOBJ.
  - Definition of a declarative debugging framework based on a cost function that assigns different cost values to each kind of error and different benefit values to each kind of correct outcome. In this approach, the diagnosis problem is redefined as assigning a real-value probability and cost to each rule, by considering each rule more or less guilty of the overall error and cost of the program.
  - Definition of a method which allows us to generate a set of decision trees from a single evidence in functional logic languages. The collection of trees is based on a shared ensemble such that the trees share their common parts. We generate a multi-tree, which is an AND/OR tree structure from which it is possible to extract a set of hypotheses. Implementation of the inductive system for the learning of decision multi-trees.
  - Definition of an ensemble method which combines different solutions in a multi-tree. Development of a technique to extract one single solution from a hypotheses ensemble without using extra data, based on the use of a random dataset.
  - Definition of a framework for the cost-sensitive induction of hypotheses and their aplication to bioinformatic problems. In order to reduce the cost of misclassification, the multi-tree is constructed by using splitting criteria based on ROC analysis.

References

[1] E. Albert. Partial Evaluation of Multi-Paradigm Declarative Languages. *AI Communications*, 14(4):235–237, 2002.

[2] E. Albert, M. Hanus, F. Huch, J. Oliver, and G. Vidal. A Deterministic Operational Semantics for Functional Logic Languages. In *2002 Joint Conf. on Declarative Programming (AGP'02)*, pages 207–222. U.P. Madrid, 2002.

[3] E. Albert, M. Hanus, F. Huch, J. Oliver, and G. Vidal. An Operational Semantics for Declarative Multi-Paradigm Languages. In Proc. *Int'l Workshop on Reduction Strategies in Rewriting and Programming (WRS 2002)*, volume 70.6 of *ENTCS*.

[4] E. Albert, M. Hanus, F. Huch, J. Oliver, and G. Vidal. An Operational Semantics for Declarative Multi-Paradigm Languages. In *Proc. of the Int'l Workshop on Functional and (Constraint) Logic Programming (WFLP'2002)*, pages 7–20. U. Udine, 2002.

[5] E. Albert, M. Hanus, F. Huch, J. Oliver, and G. Vidal. Operational Semantics for Functional Logic Languages. In *Proc. of the Int'l Workshop on Functional and (Constraint) Logic Programming (WFLP'2002)*, volume 76 of *ENTCS*. Elsevier Science Publishers, 2002.

[6] E. Albert, M. Hanus, F. Huch, J. Oliver, and G. Vidal. Operational Semantics for Lazy Functional Logic Programs. In *2nd International Workshop on Reduction Strategies in Rewriting and Programming (WRS'02)*, pages 97–112. TU Wien, 2002.

[7] E. Albert, M. Hanus, F. Huch, J. Oliver, and G. Vidal. Operational Semantics for Declarative Multi-Paradigm Languages. *Journal of Symbolic Computation*, 2003. Submitted.

[8] E. Albert, M. Hanus, and G. Vidal. A Practical Partial Evaluation Scheme for Multi-Paradigm Declarative Languages. *Journal of Functional and Logic Programming*, 2002(1), 2002.

[9] E. Albert, M. Hanus, and G. Vidal. A Residualizing Semantics for the Partial Evaluation of Functional Logic Programs. *Information Processing Letters*, 85(1):19–25, 2003.

[10] E. Albert, J. Silva, and G. Vidal. Time Equations for Lazy Functional (Logic) Languages. In *2003 Joint Conf. on Declarative Programming (AGP'03)*, 2003.

[11] E. Albert and G. Vidal. The Narrowing-Driven Approach to Functional Logic Program Specialization. *New Generation Computing*, 20(1):3–26, 2002.

[12] E. Albert and G. Vidal. Symbolic Profiling for Multi-paradigm Declarative Languages. In Proc. *Logic-Based Program Synthesis and Transformation (LOPSTR'01)*. Springer LNCS 2372:148–167, 2002.

[13] M. Alpuente, D. Ballis, F.J. Correa, and M. Falaschi. A Multi Paradigm Automatic Correction Scheme. In M. Falaschi, editor, *11th International Workshop on Functional and (Constraint) Logic Programming, WFLP'02*, pages 157–170. Research Report UDMI/18/2992/RR, Dipartimento di Matematica e Informatica, Università di Udine, 2002.

[14] M. Alpuente, D. Ballis, F.J. Correa, and M. Falaschi. Automated Correction of Functional Logic Programs. In P. Degano, editor, *Proc. of 12th European Symp. on Programming (ESOP 2003)*, Springer LNCS 2618:54–68, 2003.

[15] M. Alpuente, D. Ballis, S. Escobar, M. Falaschi, and S. Lucas. Abstract Correction of Functional Programs. In Proc. *Int'l Workshop on Functional and (Constraint) Logic Programming (WFLP'2003)*, volume 86.3 of *ENTCS*. Elsevier B.V., Amsterdam, NL, 2003.

[16] M. Alpuente, D. Ballis, S. Escobar, M. Falaschi, and S. Lucas. Abstract Correction of OBJ-like Programs. In F. Buccafurri, editor, *Proc. of the Int'l Joint Conf. on Declarative Programming, AGP'2003*, pages 422–433, 2003.

[17] M. Alpuente, M. Comini, M. Falaschi, S. Escobar, and S. Lucas. Abstract Diagnosis of Functional Programs. In M. Leuschel, editor, Proc. *Int'l Workshop on Logic Based Program Development and Transformation (LOPSTR'02)*. Springer LNCS 2664: 1–16, 2003.

[18] M. Alpuente and F. Correa. Un Depurador Abstracto, Inductivo y Paramétrico para Programas Multiparadigma. *Revista Colombiana de Computación*, 2003. To apper.

[19] M. Alpuente, F. Correa, and M. Falaschi. A Debugging Scheme for Functional Logic Programs. In M. Hanus, editor, *Selected papers of the 10th Int'l Workshop on Functional and (Constraint) Logic Programming (WFLP 2001)*, volume 64 of *ENTCS*. Elsevier B.V., Amsterdam, NL, 2002.

[20] M. Alpuente, R. Echahed, S. Escobar, and S. Lucas. Redundancy of Arguments Reduced to Induction. In Proc. *Int'l Workshop on Functional and (Constraint) Logic Programming (WFLP'2002)*, volume 76 of *ENTCS*. Elsevier B.V., Amsterdam, NL, 2003.

[21] M. Alpuente, S. Escobar, B. Gramlich, and S. Lucas. Improving on-demand strategy annotations. In Proc. *9th Int'l Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'02)*. Springer LNAI 2514: 1–18, 2002.

[22] M. Alpuente, S. Escobar, B. Gramlich, and S. Lucas. On-demand Strategy Annotations Revisited. *Information and Computation*, 2003. Submitted.

[23] M. Alpuente, S. Escobar, B. Gramlich, and S. Lucas. On-demand Strategy Annotations Revisited. Technical Report DSIC-II/18/03, DSIC, Universidad Politécnica de Valencia, July 2003.

[24] M. Alpuente, S. Escobar, and S. Lucas. A Program Transformation for Implementing on-Demand Strategy Annotations. In R. Pe na, editor, *14th Int'l Workshop onImplementation of Functional Languages, IFL'02,*, pages 409–424. U.C.M. Tech 127-02, 2002.

[25] M. Alpuente, S. Escobar, and S. Lucas. A Transformation for Implementing on-Demand Strategy Annotations. In R. Nieuwenhuis, editor, *3rd Int'l Workshop onImplementation of Logics, WIL'02,*, 2002.

[26] M. Alpuente, S. Escobar, and S. Lucas. Removing Redundant Arguments of Functions. In H. Kirchner, editor, Proc. *9th Int'l Conf. on Algebraic Methodology And Software Technology (AMAST 2002)*. Springer LNCS 2422: 117–131, 2002.

[27] M. Alpuente, S. Escobar, and S. Lucas. Analyses of Redundancy in Term Rewriting Systems. *Theory and Practice of Logic Programming*, 2003. Submitted.

[28] M. Alpuente, S. Escobar, and S. Lucas. On-demand evaluation by program transformation. In J.-L. Giavitto and P.-E. Moreau, editors, Proc. RULE'03, volume 86.2 of *ENTCS*. Elsevier B.V., Amsterdam, NL, 2003.

[29] M. Alpuente, S. Escobar, and S. Lucas. OnDemandOBJ: A Laboratory for Strategy Annotations. In J.-L. Giavitto and P.-E. Moreau, editors, Proc. RULE'03, volume 86.2 of *ENTCS*. Elsevier B.V., Amsterdam, NL, 2003.

[30] M. Alpuente, S. Escobar, and S. Lucas. OnDemandOBJ: An Optimized OBJ Interpreter. In *Terceras Jornadas sobre Lenguajes de Programación (PROLE'03)*, 2003.

[31] M. Alpuente, S. Escobar, and S. Lucas. Correct and complete (positive) strategy annotations for OBJ. In F. Gadducci and U. Montanari, editors, Proc. WRLA 2002, volume 71 of *ENTCS*. Elsevier B.V., Amsterdam, NL, 2003.

[32] M. Alpuente, M. Falaschi, G. Moreno, and G. Vidal. Rules + Strategies for Transforming Lazy Functional Logic Programs. *Theoretical Computer Science*, 2004. To appear.

[33] M. Alpuente, M. Falaschi, and A. Villanueva. Symbolic Model Checking for Timed Concurrent Constraint Programs. In *Terceras Jornadas sobre Lenguajes de Programación (PROLE'03)*, 2003.

[34] M. Alpuente, M. Falaschi, and A. Villanueva. Symbolic Model Checking for `tccp` programs. Technical Report DSIC-II/19/03, DSIC, Technical University of Valencia, 2003. Available at http://www.dsic.upv.es/users/elp/villanue/papers/techrep03.ps.

[35] M. Alpuente, M. Hanus, S. Lucas, and G. Vidal. Specialization of Functional Logic Programs Based on Needed Narrowing. *Theory and Practice of Logic Programming*, 2003. Submitted.

[36] M. Alpuente, P. Julián, M. Falaschi, and G. Vidal. lazy narrowing and needed narrowing: A comparison. In M. Falaschi, editor, *11th International Workshop on Functional and (Constraint) Logic Programming, WFLP'02*, pages 21–34. Research Report UDMI/18/2992/RR, Dipartimento di Matematica e Informatica, Università di Udine, 2002.

[37] M. Alpuente, P. Julián, M. Falaschi, and G. Vidal. Uniform Lazy Narrowing. *Journal of Logic and Computation*, 16(2):287–312, 2003.

[38] S. Antoy and S. Lucas. Demandness in rewriting and narrowing. In M. Comini and M. Falaschi, eds., Proc. WFLP 20002, volume 76 of *ENTCS*. Elsevier B.V., Amsterdam, NL, 2002.

[39] E. Badouel, C. Herrero, and J. Oliver. Cooperating Automata: A model to describe Distributed and Dynamic Systems. In *6eme Seminaire Atelier en Algèbre, Logique et ses Applications (ERAL'02)*, 2002.

[40] E. Badouel, M. Llorens, and J. Oliver. Modelling Concurrent Systems using Reconfigurable Nets. In *6eme Seminaire Atelier en Algèbre, Logique et ses Applications (ERAL'02)*, 2002.

[41] E. Badouel, M. Llorens, and J. Oliver. Modelling Concurrent Systems: Reconfigurable Nets. In *Int'l Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'03)*, pages 1568–1574. CSREA Press, 2003.

[42] D. Ballis, M. Falaschi, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Cost-Sensitive Debugging of Declarative Programs. In *Proc. of the Int'l Workshop on Functional and (Constraint) Logic Programming (WFLP'2003)*, volume 86.3 of *ENTCS*. Elsevier B.V., Amsterdam, NL, 2003.

[43] C. Borralleras, S. Lucas, and A. Rubio. Recursive Path Orderings can be Context-Sensitive. In A. Voronkov, editor, Proc. *18th Inte'l Conf. on Automated Deduction (CADE'02)*, Springer LNAI 2393: 314–331, 2002.

[44] F. Correa. *Depuración declarativa de programas lógico funcionales.* PhD thesis, Universidad Politécnica de Valencia, June 2002.

[45] S. Escobar. Improving (Weakly) Outermost-Needed Narrowing: Natural Narrowing. In G. Vidal, editor, *Proc. of the Int'l Workshop on Functional and (Constraint) Logic Programming (WFLP'2003)*, pages 47–60. Technical Report DSIC II/13/03, Universidad Politécnica de Valencia, 2003.

[46] S. Escobar. Refining Weakly Outermost-Needed Rewriting and Narrowing. In Proc. *5th Int'l Conf. on Principles and Practice of Declarative Programming (PPDP'03)*, pages 113–123. ACM Press, New York, 2003.

[47] S. Escobar. *Strategies and Analysis Techniques for Functional Program Optimization.* PhD thesis, Universidad Politécnica de Valencia, October 2003.

[48] V. Estruch, C. Ferri, J. Hernández, and M. J. Ramírez. Re-designing Cost-sensitive Decision Tree Learning. In *Workshop of Data Mining and Learning*, pages 33–42, 2002.

[49] V. Estruch, C. Ferri, J. Hernández, and M. J. Ramírez. Shared ensemble learning using multi-trees. In Proc. *IBERAMIA 2002*. Springer LNAI 2527:204–213, 2002.

[50] V. Estruch, C. Ferri, J. Hernández, and M. J. Ramírez. SMILES: A multi-purpose learning system. In Proc. *Logics in Artificial Intelligence (JELIA 2002)*, Springer LNCS 2424:529–532, 2002.

[51] V. Estruch, C. Ferri, J. Hernández, and M. J. Ramírez. Un Sistema para la Extracción de Conocimiento en Bioinformática. In *Workshop of Bioinformatics on Artificial Intelligence*, pages 77–87, 2002.

[52] V. Estruch, C. Ferri, J. Hernández, and M. J. Ramírez. Beam search extraction and forgetting strategies on shared ensembles. In Proc. *4th Int'l Workshop on Multiple Classifier Systems (MCS 2003)*. Springer LNAI 2709:206–216, 2003.

[53] V. Estruch, C. Ferri, J. Hernández, and M. J. Ramírez. Simple mimetic classifiers. In Proc. *Int'l Conf. on Machine Learning and Data Mining (MLDM 2003)*. Springer LNCS 2734:156–171, 2003.

[54] M. Falaschi and A. Villanueva. An approach to the model checking problem for `tccp`. In *Universita degli Studi di Udine, http://www.dimi.uniud.it/alicia/papers/techrep02.ps*, 2002.

[55] M. Falaschi and A. Villanueva. Automatic verification of timed concurrent constraint programs. *Theoretical Computer Science*, 2003. Submitted.

[56] C. Ferri. *Multi-Paradigm Learning of Declarative Models*. PhD thesis, Depto. de Sistemas Informáticos y Computación (U. Politécnica de Valencia), 2003.

[57] C. Ferri, P. Flach, and J. Hernández. Learning decision trees using the area under the roc curve. In Proc. *Int'l Conference on Machine Learning*, pages 139–146. IOS Press, Morgan Kaufmann Publishers, 2002.

[58] C. Ferri, P. Flach, and J. Hernández. Improving the AUC of probabilistic estimation trees. In Proc. *14th European Conf. on Machine Learning (ECML 2003)*. Springer LNCS, 2003. To appear.

[59] C. Ferri, J. Hernández, and M. J. Ramírez. From Ensemble Methods to Comprehensible Models. In Proc. *Int'l Conf. on Discovery Science*, Springer LNCS 2534: 165-177, 2002.

[60] C. Ferri, J. Hernández, and M. J. Ramírez. Induction of decision multi-trees using Levin search. In Proc. *Int'l Conf. on Computational Science (ICCS 2002)*. Springer LNCS 2329:166–175, 2002.

[61] C. Ferri, J. Hernández, and M. A. Salido. Volume under the roc surface for multi-class problems. In Proc. *14th European Conf. on Machine Learning (ECML 2003)*. Springer LNCS, 2003. To appear.

[62] B. Gramlich and S. Lucas (editors). *Reduction Strategies in Rewriting and Programming. Int'l Workshop, WRS'02 - Final Proceedings*, volume 70.6 of *ENTCS*. Elsevier B.V., Amsterdam, NL, November 2002.

[63] B. Gramlich and S. Lucas (editors). *Reduction Strategies in Rewriting and Programming. Int'l Workshop Workshop, WRS'03 - Final Proceedings*, volume 86.4 of *ENTCS*. Elsevier B.V., Amsterdam, NL, 2003. to appear.

[64] B. Gramlich and S. Lucas. Modular termination of context-sensitive rewriting. In C. Kirchner, editor, *Proc. of 4th ACM Sigplan Conference on Principles and Practice of Declarative Programming (PPDP'02)*, pages 50–61, ACM Press, New York, 2002.

[65] B. Gramlich and S. Lucas. Simple termination of context-sensitive rewriting. In B. Fischer and E. Visser, editors, *Proc. of 3rd ACM Sigplan Workshop on Rule-Based Programming (RULE'02)*, pages 29–41. ACM Press, New York, 2002.

[66] C. Herrero and J. Oliver. Autómatas Cooperativos Extendidos: sistemas Multi-Agente con dependencia geográfica. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 2003. Submitted.

[67] C. Herrero and J. Oliver. Extended Cooperating Automata. In *2003 IEEE International Conference on Systems, Man and Cybernetics (SMC'03)*. IEEE Press, 2003.

[68] C. Herrero and J. Oliver. Una extensión de los Autómatas Cooperativos. In *Proc. of XI Jornadas de Concurrencia (Colección Treballs d'Informática i Tecnologia, n. 16)*, pages 265–279. U. Jaume I, 2003.

[69] M. Llorens. *Redes Reconfigurables. Modelización y Verificación.* PhD thesis, Depto. de Sistemas Informáticos y Computación (U. Politécnica de Valencia), 2003.

[70] M. Llorens and J. Oliver. Cambios Estructurales en Sistemas Concurrentes: Redes Reconfigurables. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 2003. Submitted.

[71] M. Llorens and J. Oliver. Reconfigurable Nets: A subclass of Net Rewriting systems. In *Proc. of Eighth Asian Computing Science Conference.* Springer LNCS, 2003. Submitted.

[72] M. Llorens and J. Oliver. Sistemas de Reescritura de Redes. In *Proc. of XI Jornadas de Concurrencia (Colección Treballs d'Informática i Tecnologia, n. 16)*, pages 237–250. U. Jaume I, 2003.

[73] M. Llorens and J. Oliver. Structural Dynamic Changes in Concurrent Systems: Reconfigurable Nets. *Transactions on Computers*, 2003. Submitted.

[74] S. Lucas. Context-Sensitive Rewriting Strategies. *Information and Computation*, 178(1):294–343, 2002.

[75] S. Lucas. Context-sensitive rewriting techniques for programs with strategy annotations. In *4th International Workshop on Rewriting Logic and its Applications, WRLA'02*, September 2002. tutorial.

[76] S. Lucas. Lazy Rewriting and Context-Sensitive Rewriting. In M. Hanus, editor, volume 64 of *ENTCS*. Elsevier B.V., Amsterdam, NL, 2002.

[77] S. Lucas. MU-TERM version 1.0. user's manual. Technical Report DSIC-II/1/02, DSIC, Universidad Politécnica de Valencia, January 2002.

[78] S. Lucas. Termination of (Canonical) Context-Sensitive Rewriting. In S. Tison, editor, *Proc. of 13th International Conference on Rewriting Techniques and Applications (RTA'02)*, Springer LNCS 2378:296–310, 2002.

[79] S. Lucas. Polynomials for proving termination of context-sensitive rewriting. In *Terceras Jornadas sobre Lenguajes de Programación (PROLE'03)*, 2003. Submitted.

[80] S. Lucas. Polynomials for proving termination of context-sensitive rewriting. Technical Report DSIC-II/21/03, DSIC, Universidad Politécnica de Valencia, September 2003.

[81] S. Lucas. Semantics of programs with strategy annotations. Technical Report DSIC-II/8/03, DSIC, Universidad Politécnica de Valencia, April 2003.

[82] S. Lucas. Strong and NV-Sequentiality of Constructor Systems. *Information Processing Letters*, 2003. Submitted.

[83] S. Lucas. Termination of Computational Restrictions of Rewriting and Termination of Programs. In A. Rubio, editor, *Extended Abstracts of 6th International Workshop on Termination, WST'03*, pages 44–48. Technical Report DSIC II/15/03, Universidad Politécnica de Valencia, 2003.

[84] S. Lucas. Termination of programs with strategy annotations. *ACM Transactions on Programming Languages and Systems*, 2003. Submitted.

[85] S. Lucas. Termination of programs with strategy annotations. Technical Report DSIC-II/20/03, DSIC, Universidad Politécnica de Valencia, September 2003.

[86] S. Lucas. Semantics of Maude and OBJ* programs with strategy annotations. In *ACM Symposium on Principles of Programming Languages, POPL'04*, 2004. Submitted.

[87] J. G. Ramos, J. Silva, and G. Vidal. An Embedded Language Approach to Router Specification in Curry In Proc. *Int'l Conf. on Current Trends in Theory and Practice of Informatics (SOFSEM 2004)*. Springer LNCS, 2004. To appear.

[88] G. Vidal. A Partial Evaluation Tool for Multi-Paradigm Declarative Programs (invited talk). In *2002 IEEE International Conference on Systems, Man and Cybernetics (SMC'02)*. IEEE Press, 2002.

[89] G. Vidal. Cost-Augmented Narrowing-Driven Specialization. In *Proc. ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation (PEPM'02)*, volume 37(3) of *SIGPLAN NOTICES*, pages 52–62, New York, 2002. ACM Press. Extended version to appear in *Higher-Order and Symbolic Computation* (formerly Lisp and Symbolic Computation)

[90] G. Vidal. Forward Slicing of Multi-Paradigm Declarative Programs Based on Partial Evaluation. In Proc. *Logic-based Program Synthesis and Transformation* (LOPSTR 2002). Springer LNCS 2664:219–237, 2003.

[91] A. Villanueva. *Model Checking for the Concurrent Constraint Paradigm*. PhD thesis, University of Udine in cotutela with Technical University of Valencia, May 2003.

2.1.2 University of Málaga (UMA)

The contributions of the UMA team have been made in the first three tasks previously mentioned, although they have also made some contributions indirectly related to the fourth task.

- Task 1: Modeling and analysis of software components
  - A formalism based on process algebras and the notion of *role* have been used to describe the interaction of CORBA components. The proposal is based on adding protocol information to components whose signature is described by an IDL.

  - Different coordination models (Linda, Reo) have been proposed to define the behavioural protocols of software components. A process calculus has been defined for each coordination model, and a compatibility relation has been defined to analyse the *safe* combination of a component-based system.
  - The Maude language has been proposed as a common formalism to describe enterprise and information viewpoints in RM-ODP.
  - Construction of a prototype deriving Maude modules from UML graphical specifications (class diagrams). This tool permits the analysis of UML diagrams through Maude specifications.
  - A tool is being developed (in cooperation with UPV team) to check the termination of Maude specifications.
  - We have proposed a methodology for supporting the automated adaptation of software components. The adaptation is automatically derived from the behaviour specification of components to be adapted, and the pattern to be fulfilled by the derived adaptor.
  - A *Pure Sequent Calculus* has been defined, providing an equivalent presentation of *Pure Type Systems*. This alternative presentation allows us to study two important open problems for pure type systems: *cut-elimination* and *expansion postponement*.

- Task 2: Verification, model checking and abstract interpretation
  - We have developed an abstract model checking theory, which is dual to the classical one, where both the model and the property to be checked are over-approximated (instead of under-approximating them).
  - A methodology has been proposed to apply abstract model checking techniques to UML state-charts and MSCs (Message Sequence Charts).
  - We have constructed αSPIN, a tool implementing our proposal in the context of Promela and LTL.
  - An operational semantics for full Promelahas been defined. As far as we know this is the first formal semantics covering all features of Promela.
  - In cooperation with some UPV team members, we are applying abstract model checking techniques to *Timed Concurrent Constraint Programming*.

- Task 3: Performance debugging and optimization
  - We have defined an implementation strategy for linear logic languages. The new *Tag-Frame* system proposes a strategy for resource management that reduces the complexity of all the linear connectives.
  - A prototype for the linear logic language Lolli has been constructed to illustrate the possibilities of *tag-frames*.

- Task 4: Declarative debugging and program learning
  - *Tracing Tag-Frame*, a variant of *Tag-Frame*, allowing the debugging of Lolli programs.

References

[1] G. Rodriguez, P. Merino, M. M. Gallardo. An Extension of the ns Simulator for Active Networks Research. Computer Communications, 25(3):189–197, 2002.

[2] J. M. Alvarez, M. Diaz, L.M. Llopis, E. Pimentel, and J.M. Troya. Integrating schedulability analysis and design techniques in SDL. *Journal of Real-Time Systems*, 2003 (37 pages).

[3] J. M. Álvarez, M. Díaza, L.M. Llopis, E. Pimentel, and J.M. Troya. An object oriented methodology for embedded real-time systems. *The Computer Journal*, 46(2), 2003 (24 pages).

[4] A. Brogi, C. Canal, and E. Pimentel. On the specification of software adaptation. En *2nd International Workshop on Foundations of Coordination Languages and Software Architectures*. Pendiente de publicaciòn en *ENTCS*. Preselected for publication in *Science of Computer Programming*, 2003.

[5] Carlos Canal, Lidia Fuentes, Ernesto Pimentel, Antonio Vallecillo, and J. M. Troya. Adding roles to Corba objects. *IEEE Transactions on Software Engineering*, 2003.

[6] F. Durán and J. Meseguer. Structured theories and institutions. *Theoretical Computer Science*, 2003. In press.

[7] F. Durán and A. Vallecillo. Formalizing ODP enterprise specifications in Maude. *Computer Standards & Interfaces*, 25(2):83–102, 2003.

[8] M.M. Gallardo, J. Martínez, P. Merino, and E. Pimentel. A tool for abstraction in model checking. *Software Tools for Technology Transfer*, 2002. Accepted for publication.

[9] M. M. Gallardo, P. Merino, E. Pimentel. Debugging UML Designs with Model Checking. *Journal of Object Technology, 1(2):101-117.* 2002.

[10] M.M. Gallardo, P. Merino, and E. Pimentel. Refinement of ltl formulas for abstract model checking. In Proc. *9th Int'l Static Analysis Symposium.* Springer LNCS 2477, 2002.

[11] F. Gutiérrez and B. C. Ruiz. Expansion Postponement via Cut Elimination in Sequent Calculi for Pure Type Systems. In J.C.M. Baeten, J.K Lenstra, J. Parrow, and G.J. Woeginger, editors, *13th Int'l Colloquium on Automata, Languages and Programming (ICALP2003).* Springer LNCS 2719:956–968, 2003.

[12] J.S. Hodas, P. López, J. Polakova, L. Stoilova, and E. Pimentel. A tag-frame system of resource management for proof search in linear-logic programming. In J. Bradfield, editor, *Annual Conference or the European Association for Computer Science Logic*, Springer LNCS 2471:167–182, 2002.

[13] J. M. Molina-Bravo and E. Pimentel. Composing programs in a rewriting logic for declarative programming. *Theory and Practice of Logic Programming*, 3(2):189–221, March 2003.

[14] A. Roldán, E. Pimentel and A. Brogi. Safe Composition of Linda-based Components, TACoS'03 (ETAPS'03). volume 86.2 of ENTCS. Elsevier B.V., Amsterdam, NL, 2003.

[15] M.M. Gallardo, P. Merino, and E. Pimentel. A generalized semantics of Promelafor abstract model checking. *Formal Aspects of Computing*, 2002. Enviado para su revisión.

2.1.3 University of Castilla-La Mancha

According to plan, Castilla- La Mancha is working mainly on the third task, pursuing the optimization of both multi–paradigm declarative programs and languages.

- Task 3: Performance debugging and optimization
  - We have investigated the formal relation between needed narrowing and another (not so lazy) demand–driven narrowing strategy which is the basis for popular implementations of lazy functional logic languages. We demonstrated that needed narrowing and demand-driven narrowing are computationally equivalent over the class of *uniform* programs.
  - We also formalized a complete refinement of demand-driven narrowing, called *uniform lazy narrowing*, which is still equivalent to needed narrowing over the aforementioned class.
  - We introduced an optimized representation for the data structure of definitional trees which are used to guide the needed narrowing mechanism in some high level implementations of (needed) narrowing into Prolog. We formulated a generic algorithm that builds refined definitional trees.
  - We developed the first fold/unfold transformation system for multi-paradigm declarative languages which are based on needed narrowing. We provided strong correctness results for the transformation system and also an experimental prototype (the SYNTH system), which provides witness for the practicality of the method.
  - We endow the SYNTH system with two powerful heuristics: 1) an automatic *composition* strategy empowered with a eureka generator based on partial evaluation (PE), and 2) an incremental, semi-automatic, *tupling* strategy which is based on the three rules: definition introduction, unfolding and abstraction with folding. Our tupling algorithm can derive (at a low cost) a good set of eureka definitions and performs sophisticated termination tests during the search for regularities.
  - We formalized a *factoring* program transformation that, in some cases, eliminates the non–determinism of the program by introducing a suitable "alternative" operator. We demonstrated the correctness of the factoring transformation under some appropriate conditions and also defined some cost criteria which help to measure the effectiveness of the transformation. We performed some experiments which demonstrated that the factoring transformation may save both execution time and memory consumption.

References

[1] M. Alpuente, M. Falaschi, P. Julián, and G. Vidal. Uniform Lazy Narrowing. *Journal of Logic and Computation*, 13(2):27, March/April 2003.

[2] M. Alpuente, M. Falaschi, G. Moreno, and G. Vidal. Rules + Strategies for Transforming Lazy Functional Logic Programs. *Theoretical Computer Science*, 2004. To appear. (56 pages)

[3] Maria Alpuente, Moreno Falaschi, Pascual Julián, and Germán Vidal. Lazy narrowing and needed narrowing: A comparison. In Proc. *11th Int'l Workshop on Functional and (Constraint) Logic Programming (WFLP 2002), Grado, Italy, June 20-22, 2002*, pages 21–34. UDMI/18/2002/RR, 2002.

[4] S. Antoy, P. Julián-Iranzo, and B. Massey. Improving the efficiency of non-deterministic computations. volume 64 of *ENTCS*. Elsevier B.V., Amsterdam, NL, 2003.

[5] P. Julián-Iranzo. On the correctness of the factoring transformation. In Z. Hu and M. Rodríguez-Artalejo, editors, *Proc. of 6th Int'l Symp. on Functional and Logic Programming (FLOPS'02)*. Springer LNCS 2441:119–133, 2002.

[6] P. Julián-Iranzo. Partial Evaluation of Lazy Functional Logic Programs. *AI Communications, IO Press (Amsterdam)*, 16(1):3, 2003.

[7] P. Julián-Iranzo. Refined definitional trees and prolog implementations of narrowing. In Proc. *12th Int'l Workshop on Functional and (Constraint) Logic Programming (WFLP 2003)*, pages 117–129. UPV, 2003.

[8] G. Moreno. A Safe Transformation System for Optimizing Functional Programs. Technical Report DIAB-02-07-27, UCLM, 2002. Available in URL: `http://www.info-ab.uclm.es/~personal/gmoreno/gmoreno.htm`.

[9] G. Moreno. Automatic Optimization of Multi-Paradigm Declarative Programs. In F. J. Garijo, J. C. Riquelme, and M. Toro, editors, *Proc. of IBERAMIA 2002*. Springer LNAI 2527:131–140, 2002.

[10] G. Moreno. Automatic Tupling for Functional–Logic Programs. Technical Report DIAB-02-07-24, UCLM, 2002. Available at URL: `http://www.info-ab.uclm.es/~personal/gmoreno/gmoreno.htm`.

[11] G. Moreno. Incremental Tupling in a Functional-Logic Setting. In *Proc. of Segundas Jornadas sobre Lenguajes de Programación (PROLE'02), El Escorial (Spain)*, pages 32–48. UCM, 2002.

[12] G. Moreno. Transformation Rules and Strategies for Functional-Logic Programs. *AI Communications, IO Press (Amsterdam)*, 15(2):3, 2002.

[13] G. Moreno. A Narrowing-Based Instantiation Rule for Rewriting-Based Fold/Unfold Transformations. In Proc. *12th Int'l Workshop on Functional and (Constraint) Logic Programming (WFLP 2003), volume 86.4 of* ENTCS. *Elsevier B.V., Amsterdam, NL, 2003.*

3 Results indicators

3.1 Publications

Table 3.1 below summarizes the publications produced by the three STREAM partners during the reported period:

	UPV	UMA	UCLM	TOTAL
Int'l Journal	20 (7 SCI, 13 ENTCS)	19 (8 SCI, 11 ENTCS)	6 (4 SCI, 2 ENTCS)	45
Nat'l Journal	3	0	0	3
Int'l Conf.	24	16	4	44
Int'l Workshop	13	13	3	29
PhD Thesis	5	2	0	7
Tech. Reports	7	2	3	12
TOTAL	72	52	16	140

3.2 Other indicators

Staff Training: A total of 7 PhD theses have already been completed during the development of the project (F. Correa, A. Villanueva, C. Ferri, S. Escobar, M. L. Lloréns, A. J. Fernández and F. Gutierrez). Four PhD fellowships were funded by the Spanish Ministery of Science and Technology (A. Villanueva), regional government (J. Silva), and the partner universities (M. Abril, V. Estruch). Other five research fellows have been recently hired by the project (E. Modroiu, C. Ochoa, D. Ballis, D. Garrido, and C. Villamizar).

International projects: UPV has participated in three bilateral cooperation projects:

- Acción Integrada Hispano-Alemana HA2001-005, with CAU Kiel, focusing on the debugging of multi–paradigm declarative languages, including trace-based debuggers, program profiling and declarative debugging, as well as their application to Curry.
- Acción Integrada Hispano-Austríaca HU2001-0019, with TU Wien, pursuing the analysis and optimization of functional programs under programmable strategies, and their application to CafeOBJ, ELAN, Maude, OBJ3, and Stratego.
- Acción Integrada Hispano-Italiana HI2001-0161, with U. Udine, oriented towards the development of a framework for the development of reactive programs, including model checking and abstract debugging techniques.

UPV and UMA participate in a NoE (IST-2001-33123 CologNET) and are members of SpaRCIM (a part of the *European Research Consortium for Informatics and Mathematics* funded by Spanish MCyT since July 2003). UMA participates in a cooperation project with U. Pisa. With U. Udine and U. Genoa (Italy), and U. Hyderabad (India), UPV has recently submitted a proposal for an *EU-India Economic Cross Cultural Project*, which aims at developing, experimenting and maintaining a number of intelligent web-based knowledge dissemination and validation tools. Finally, UPV and UCLM are involved in an ALFA project proposal led by the University of Minho.

Collaboration with other groups: The consortium has a strong relationship with other groups in the form of joint projects, joint PhD Programmes, co–organization of workshops, stays, etc. The UPV collaborates with the Christian Albrechts Universität CAU Kiel (Ge), IMAG Grenoble (Fr), and Portland State U. (USA) on the development of program specialization, program analysis and debugging techniques for integrated languages. They also collaborate with the U. Udine (It) in the formalization of declarative diagnosis and model checking tools. New collaborations have been set up in the reported period with the Technische Universität Wien (Aut) on Strategic Programming, with the U. Bristol (UK) on Machine Larning, and with the U.P. Catalonia (Sp) regarding termination of rewriting. A collaboration with IRISA Rennes (Fr) has led to the definition of configurable nets, which model changes in concurrent systems. Joint publications derived from the cooperations have been obtained in all cases. In the past two years, 24 stays have been made by UPV researchers to Melbourne University (P. Stuckey), CAU Kiel (M. Hanus), T.U. Wien (B. Gramlich), U. of New Mexico (G. Gupta), U. Udine (M. Falaschi), IMAG Grenoble (R. Echahed), Portland State U. (S. Antoy), and ENSP Yaoundé (E. Baudouel). Also, other researchers have visited the group at UPV. These include Marco Comini (U. Udine) and Frank Huch (CAU Kiel).

At UMA, collaboration with the group led by José Meseguer has continued. The goal is to investigate different aspects of the CRWL and Maude language. UMA also cooperates with the group of Antonio Brogi (U. Pisa) on coordination languages and software architectures, and with Miguel Katrib (U. Havana) on integrating concurrency and assertions. Important contacts continue with Prof. Dale Miller at LIX (Fr) and Prof. Joshua Hodas at Harvey Mudd College (Claremont, USA) to cooperate on the efficient implementation of linear logic–based languages. As a result of these collaborations, a 3-month research visit to Claremont has been made. More recently, new collaborations have been set up with the U. York (A. Evans) and U. Namur (J.-M. Jacquet). Finally, UCLM has a very active ongoing cooperative effort with U. Udine and PSU.

As for other results, we would like to mention the following:

1. In the two-year period, a number of STREAM researchers have served on the program committees of some of the most relevant conferences in the area, such as FLOPS, LPAR, LOPSTR, ACM SIGPLAN PEPM, PPDP, RTA, WFLP, WIL, WRLA, WRS, AGP, etc.

2. Some STREAM members have participated as invited speakers in a number of Int'l conferences and workshops, including the *Int'l Workshop on Rewriting Logic and Applications (WRLA 2002, Pisa, September 2002)*, the annual meeting of the IFIP Working Group 1.6 on Term Rewriting (Copenhagen, July 2002) and the IEEE Int'l Conf. on Systems, Man and Cybernetics (SMC'02, Hammamet, October 2002).

3. UPV organized the *Federated Conference on Rewriting, Deduction, and Programming (RDP 2003)*, consisting of RTA'03 and TLCA'03, the satellite workshops FTP'03, RULE'03, UNIF'03, WFLP'03, WRS'03, and WST'03, and the annual meeting of IFIP WG 1.6. STREAM participants chaired WFLP 2003 and WRS 2001–03 (with T.U. Wien) proceedings of WFLP 2003 and WRS 2003 appeared in the series *Electronic Notes in Theoretical Computer Science (ENTCS)* published by *Elsevier B.V.* UMA organized ICALP 2002.