CUBICO
Optimized Compilation Techniques for Pervasive Computation
TIC 2002-0055

Francisco Bueno Carrillo *
Facultad de Informática
Universidad Politécnica de Madrid
28660-Boadilla del Monte, Madrid
Phone: +341-336-7435 ó +341-352-4803

Abstract
This project pursues the advance in the state of the art of program manipulation
techniques and their integration as tools in an advanced programming system with high
language expressiveness, to adapt it for pervasive computing environments. The main line
of work is the extension of the program compilation and development aid tools so that
they can be used in such an environment to reduce program resource consumption and to
instrument programs to perform dynamic control of such consumption during execution.
The final system should be an ideal candidate for programming ambient intelligence in
mobile and pervasive computing environments, because of both the features of the lan-
guage and also the power of its compilation techniques.
Keywords: Program Debugging, Maintenance, Documentation, Assertions, Abstract
Interpretation, Global Analysis, Specialization, Parallelization, Visualization, Logic and
Constraint Programming.

1 Objectives of the project
The main aim of this project is to extend the Ciao language [6] and its compile time program
manipulation and analysis tools [8, 5] in a way that they become suitable for pervasive com-
puting intelligent environments. Ciao is a next generation multi-paradigm programming lan-
guage that offers great expressiveness for programming distributed and intelligent multi-agent
systems. The formal techniques and tools of Ciao (the CiaoPP system) allow sophisticated
program manipulation with the aim of optimization and/or debugging of programs. In this
project we attempt to apply and adapt these formal techniques and tools to the task of mak-
ing programs suitable for unconventional (in the sense of their memory constraints, processor
limitations, etc.) computing devices.

*Email: buenof@ifi.upm.es
From this main target the following concrete objectives follow: development of new techniques of transformation and compilation of programs with a focus on reducing resource consumption; development of new program analysis techniques for relevant properties of the aforementioned transformations; development and implementation of a new mobile computing model and its inclusion in the Ciao system; performing necessary changes to the Ciao language for executing on non-conventional computing platforms (like PDA’s); and implementation of means for distributed execution of Ciao and the research of possible improvements.

At a greater level of detail, we can emphasize the following concrete scientific-technical objectives, which are elements of the workplan: an automated technique for program resource consumption control (task H); improved techniques to apply cost, determinacy, and non-failure analysis to the resource consumption analysis (task G); improvements to the precision of type analysis by means of the inclusion of other kinds of relevant information (task E); a generic scheme for compositional analysis of programs built by means of components (task F); a generic scheme of abstract evaluation for specialization/simplification of programs (task I); an optimizing compiler that incorporates all previous techniques (task J); a mobile computing model for object oriented logic programming (task B); a management mechanism for distributed execution of constraints (task A); a formal semantic model for the aforementioned language features (task D); and an integrated system including the language, the execution models, and the development tools, which should be operative in a non-conventional computing platform (tasks C, K, and I).

![GANTT chart of tasks](image)

The workplan is carried out according to figure 1 that shows a GANTT chart of tasks.
2 Level of achievement of the project

2.1 Distributed Management of constraints (task A)

The design of a distributed constraint system and a first prototype with disjunctive parallelism have been achieved. The design is intended to exploit the labeling of one of the variables keeping the sequence when the variables are selected. The labeling process gives a defined ground value or adds a new constraint which reduces the domain of each variable [20]. A prototype have been made also for the case of conjunctive parallelism by means of improving the backtracking algorithm (ABT) of Yoko that allows to divide a set of constraints among a number of agents and solve it in a distributed way. Finally, both implementations have been integrated into Ciao, preserving the typical mechanisms of the language and using inherited syntaxes and semantics of previous parallelism and concurrency approaches. Also, a configuration environment for distributed execution has been designed. It extends the concept of “active module” in order to allow the development of distributed applications regardless of properties related to the topology or the architecture, which can be configured without modifying the source code [13].

2.2 Mobile Computing: meta-objects (task B)

An object oriented system [33] has been designed and implemented which is compatible with the ISO-Prolog semantics and is based on an advanced module system. An extension of this system for distributed and concurrent execution has also been implemented [11]. We have incorporated several syntactical extensions to the Ciao language for database access either from external files or database managers [16, 14, 15], also an extension to implement fuzzy control agents [21]. Then we have designed a mobile computing model, presented in [22], that is consistent with the distributed programming model and the module and object systems of Ciao [6]. Essentially, we have adopted the Proof-Carrying Code model, where a proof is attached to the code sent which guarantees that the execution of the code is safe. Unlike previous formulations, the suggested method [4] is entirely based on abstract interpretation techniques. Its implementation has required the development of a proof checker tool [1] than can verify the validity of the certificate, and thus compliance with the safety policy, by inspection of the code and the certificate. In [3] a large amount of experiments has been presented which take into account the resource consumption and are relevant in pervasive systems. In this way, a receiver node can use this technology to refuse code not observing a safety policy that limit the resource consumption. A detailed description for the case of computational cost has been presented in [2].

2.3 Application Platform: PDA (task C)

A first step has been made in the design and implementation of compilation techniques that preserve the semantics of the original programs but achieve compiled programs that consume less amount of resources (memory, CPU time, etc.) by means of program optimization and specialization [30, 29]. In addition, tests of the feasibility of the execution of a reduced version of the Ciao system have been made in a standard PDA with no need for special hardware.
2.4 Formal Semantics (task D)

A new language (Hiord) and a new semantics for type-free higher order logic programming with predicate abstraction have been developed [10]. This work is based on an extension of uniform algebras, a higher order semantics with types that allows to create special types for several software components and for constraints [18, 19]. Also, a new compact formalization of modules based on higher order predicate abstraction, suitable for agent and object definition, has been introduced.

2.5 Type Analysis (task E)

Various type analyzers have been implemented, with several precision degrees from more abstract to more concrete values [38]. The implementation of abstract operations for the evaluation of builtins leads to type analyses precise enough to obtain concrete values, which are very useful for abstract specialization [37].

2.6 Compositional Analysis (task F)

A generic framework for context-sensitive compositional analysis of modular programs has been designed, which is parameterized with respect to the analysis domain, and is applicable to each software component separately [36]. This framework is created as an extension of the analysis model of CiaoPP, which allows the use of all features of the analyzer. It has been implemented and integrated into the CiaoPP system.

2.7 Resource Consumption Analysis (task G)

The design and implementation of an abstract domain for determinacy analysis [26] has been developed and incorporated to the CiaoPP system [8, 5]. A new domain for non-failure analysis has been introduced. It is able to detect calls which return at least one solution (i.e., they terminate and not fail) [7]. Combining the determinacy, non-failure, and cost analyses, we have also developed and implemented a cardinality analysis, which infers the number of solutions of procedure calls.

A profiler has been designed and implemented that allows to analyze the performance of a system, obtaining information about number of calls, failures, retries, and execution times [27]. Also, a tool has been designed and implemented which uses the profiler and allows to adjust the constants of cost functions for a given system and a given complexity model [28].

2.8 Resource Consumption Control (task H)

Program transformation techniques have been developed for reducing the execution cost of the program. They are also applicable to reducing resource consumption (that is memory waste, and cpu time). In the case that it is not possible to reduce resource consumption by means of the aforementioned transformations, the techniques allow to transform the program in a way that the program itself can control it dynamically [22].

A first design has been made of a framework for program transformation to rearrange calls so that possibly-failing goals are executed ahead of non-failing goals where possible. This transformation is made using the information of the non-failure analysis developed in task G.
2.9 Abstract Evaluation (task I)

We have developed an overall view of all specialization techniques available in the CiaoPP system\cite{25, 24, 23}, from which we have considered the new and promising possibilities of specialization allowed by the use of the new type abstract domains described in task E. A new framework has been introduced that gives the first full integration among abstract interpretation and partial evaluation. It allows to improve the results of both of them separately \cite{35}. New unfolding techniques for partial evaluation have been developed. These techniques allow to apply efficiently local control strategies \cite{34}. An analysis and specialization system that incorporates all the aforementioned features has been implemented into CiaoPP.

2.10 Optimizing Compiler (task J)

A framework for optimizing compilation has been designed, that produces low level code, finer grained than the abstract machine’s usual one, and uses analysis information to optimize it \cite{31}. Tools for compilation to C have been incorporated to the Ciao system in addition to global improvements that affect the execution and compilation performance. Also, low level assertions have been introduced to the object code of programs, improving robustness of compiled programs.

3 Performance Indicators

3.1 Project’s publications

The project has produced the following publications: 2 articles in books \cite{36, 24}; 1 Ph.D. Thesis \cite{9}; 1 Master’s Thesis \cite{32}; 1 article in refereed journal \cite{21}; 11 articles in refereed conferences \cite{1, 3, 4, 7, 10, 26, 31, 34, 17, 15, 37}; 1 workshop proceedings \cite{12}; 2 Tutorials \cite{22, 25}; 4 publications in workshops \cite{2, 35, 16, 30}; and 9 technical reports and manuals \cite{6, 8, 5, 18, 19, 20, 27, 28, 35}.

3.2 Graduate Students within the project

The project has sponsored the research activities of six (6) PhD students, one of them currently following his studies at U. of New Mexico (USA) and another one who has already defended his thesis.

3.3 Participation in international projects

The group has participated in the following projects:

- ASAP Project (November 1, 2002 - October 21, 2005) EU IST FET, IST-2001-38059 *Advanced Specialization and Analysis for Pervasive Systems*. In collaboration with the Bristol and Southampton Universities (U.K.), and Roskilde University (Denmark).
3.4 Technological transfer

It is remarkable that the Ciao system, partially developed in the present project, has been transferred to the Italian company Conecta Srl, which has participated in the European project IST-2001-34717 AMOS together with our research group. In AMOS the programming environment Ciao has been the main development platform.

In addition, there is a large number of groups in both academy and industry that are users and co-developers of the Ciao programming environment, with which there has been a lot of interaction in the context of this project. In fact, there have been more than 2500 installations of the system all over the world and the user list is quite active: Also, the system is cited in research forums as one of the referents in the constraint logic programming area. Thus, it is to hope that the project results obtain great visibility through their incorporation to the system.

References


