Paralogism Progress Report
(PCA 4136)

José C. Cunha, Pedro D. Medeiros, Manuel B. Carvalhosa, Luís Moniz Pereira
Departamento de Informática
Universidade Nova de Lisboa
2825 Monte da Caparica
Portugal

October 1990

1. Project overview

Our current research in the area of parallel and distributed logic programming systems aims to fulfill the following generic objectives:

(1) obtain improved performance for the execution of Prolog programs on parallel architectures, when compared to its corresponding sequential execution;
(2) move towards the goal of using the logic programming paradigms for efficient parallel problem solving, while hiding the intricacies of programming the underlying parallel computer architectures;
(3) the application of logic programming paradigms to the specification and programming of operating systems and distributed processing systems.

In the past 10 years, progress has been made on identifying the potential sources of parallelism in a (Horn clause) logic program, and devising alternative parallel execution models. However, the ideal of keeping unchanged the declarative semantics of a logic program when going from sequential to parallel execution has proven difficult to achieve. Issues such as automatic exploitation of the distinct types of parallelism, control of concurrent execution, and efficiency of the implementation of each parallel execution model on concrete hardware architectures are still open areas of research .

The research in the scope of this project continues previous work, two major outcomes of which have been the Delta Prolog programming language (Cun 89) and the accumulated experience on the basic mechanisms for the support of distributed logic programs (Cun87; Cun 88).

---

e-mail: jcc@fctunl.rccn.pt
pm@fctunl.rccn.pt
mcv@fctunl.rccn.pt
lmp@fctunl.rccn.pt

1.1 Abstraction levels for distributed logic programming systems

We view a distributed logic programming system as an hierarchy of abstractions, consisting of three distinct levels:

- programming language model
- abstract execution model
- run-time support for parallel architectures

1.1.1 Programming language models

Our global approach is to identify, in a first stage, generic strategies for solving distinct classes of problems where the exploitation of parallel and distributed processing models may be relevant, i.e. where a speed-up may be obtained or where a specific need for distribution (of data or / and control) may arise. In a second stage, we consider programming models suitable for expressing those problem solving strategies, and thus identify a collection of models. The experimentation with the proposed language constructs is guided by the programming of selected problem classes.

We started by considering a distributed programming model extending Horn clause logic with constructs for the specification of distributed systems (Mon 86). At this point in our research we use, as one of the basic models, the Delta Prolog language which supports explicit dependant And-parallelism and inter-process communication.

A Delta Prolog program is a sequence of clauses of the form: H :- G_1,...,G_n. (n_0). The *comma* is the *sequential* composition operator. Whereas H is a Prolog goal, each G_i may be either a Prolog or a Delta Prolog goal. A Delta Prolog goal is either a *split* goal (for the creation of parallel processes), an *event* goal (for inter-process communication and synchronization) or a *choice* goal (for external non-determinism). A Delta Prolog program without Delta Prolog goals is and executes like a Prolog program, so Delta Prolog is an extension to Prolog.

1.1.2 Abstract execution models

This level includes the study of concurrent execution models for the language programs and the analysis of the practicability of their efficient implementations on multiprocessor computer architectures.

This will conduct to the proposal of a virtual architecture integrating the functionality of an efficient Prolog implementation with mechanisms for concurrency and interprocess communication.

1.1.3 Run-time support for parallel architectures

At this level we address the mappings between the computations as defined by the abstract models and the execution paradigms more suitable to the existing paralell and distributed processing models (e.g. shared vs. non-shared memory systems, granularity of parallel computations, inter-process communication schemes vs. physical processor interconnection schemes).

2. Progress report

The beginning of the project was affected by the delay in the delivery of the multiprocessor.
We developed research work in the following tasks:
(i) basic mechanisms for the support of distributed problem solving in logic;
(ii) efficient implementations and mappings into parallel computer architectures

This is related to the fullfilment of the 1st year milestones, i.e. to provide a complete first definition of the virtual architecture (a basic kernel), a first prototype on a single workstation (directed to the test of language and application needs), and a preliminary prototype on the transputer multicomputer.
In the following we summarize the work being done since January 90.

2.1 Develop basic mechanisms for the support of distributed problem solving in logic

The objectives are:

- the design and implementation of a parallel execution system for the support of logic programming, through the provision of a suitable set of primitive mechanisms;
- proposal of a virtual architecture integrating the functionality of an efficient Prolog implementation with mechanisms for concurrency and interprocess communication.

Two approaches were followed for the development of a prototype running on a workstation.

2.1.1 Interpreter based approach

We designed a set of primitives for the support of multiple threads in Prolog computations. They are suitable for the implementation of Delta Prolog and we implemented them within a C-Prolog interpreter (Per 83). The execution model is based on the coroutining of multiple threads, the switching being made at the communication points that correspond to Delta Prolog goals. Currently we have a running prototype (under UNIX) for a subset of the language, supporting parallel goal activation and event goals (for communication and synchronization) but with a simplified backtracking strategy.

2.1.2 Compiler and abstract machine for Delta Prolog

We are designing an abstract machine for the Delta Prolog language. At present we are in the process of defining the instruction set and the operational execution model of the machine. The implementation of the abstract machine will be based on the SB-Prolog system - a public domain version of the WAM (War 83) and the prototype will be first tested on a UNIX environment for a uniprocessor, and then ported to a transputer-based environment.

2.2 Efficient implementations and mappings into parallel architectures

A parallel implementation of Delta Prolog requires system support of multiple executors. They may be mapped to separate processes in a multiprocessed operating system (e.g. Unix) for a single CPU computer, where each process executes an instance of a modified Prolog (e.g. a C-Prolog interpreter plus extensions for process creation and inter-process communication, or an instance of an abstract machine). Alternatively, they may be supported by system processes running on distinct CPUs on multiprocessor machines (shared or non-shared memory) (Cun 90).

In order to develop a Delta Prolog system for distributed-memory machines, we have been using a small transputer-based prototype. This is running on a hardware configuration with 4 processors (cards with 1 T800 and 4 Mbytes of memory developed at this University), the host being an IBM-PC/AT compatible. We plan to adapt the work on this prototype to the Meiko Computing Surface environment.

At a first stage a system layer that provides the basic inter-process communication, file system and interactive input/output requirements was implemented on the above transputer prototype. Basic inter-process communication primitives for Prolog processes have been implemented (and are currently running) on top of this system layer.

The current stage corresponds to the exploitation of a transputer-based architecture for the support of a language model like Delta Prolog, with focus on an efficient mapping between the language constructs requirements and the transputer processing model.

References

(Cun 87) Cunha, J.; Medeiros P.; Carvalhosa M. : Interfacing Prolog to an operating system environment: mechanisms for concurrency and parallelism control. Internal Report, UNL-/87, Computer Science Department, April 1987.

(Cun 88) Cunha, J.: Concurrent execution of a logic programming language, PhD. Thesis, September 1988.

(Cun 89) Cunha, J.; Conceição, M; Pereira, L.; Programming in Delta Prolog, in Proceedings 6th International Logic Programming Conference, MIT Press, July 1989.

(Cun 90) Cunha, J.; Medeiros, P.; Pereira, L.; A distributed logic programming language and its implementation on transputer networks, in Proceedings of the 12th Occam User Group Technical Meeting, IOS Amsterdam, April 1990.

(Mon 86) Monteiro, L. : Distributed logic: a theory of distributed programming in logic. UNL, Computer Science Department, 1986.

(Per 83) Pereira, F. (ed.), C-Prolog User Manual, DAI, Universit of Edinburgh, 1983.

(War 83) Warren, D.H.D., An abstract Prolog instruction set, Technical Note 309, SRI International, Menlo Park, 1983.