Parallel Logic Programming for Problem Solving

Ramiro Varela Arias, Camino Rodríguez Vela, Jorge Puente Peinador, and Cesar Alonso González

Received June 1998; revised March 2000

We present a new model for parallel evaluation of logic programs. This model can exploit the main sources of parallelism that the language of logic expresses: independent and parallelism and or parallelism, together with a secondary source emerging as a consequence of the independent and Parallelism: the producer/consumer parallelism. The efficiency is derived from the use of ordered structures for managing the information generated throughout the search process. The model is suitable for evaluating programs with a high degree of non-determinism because it never generates two processes for solving the same subgoal and hence it can exploit the same real parallelism generating a lower number of processes than other models. As an application example, we consider the Job Shop Scheduling problem. We report experimental results showing that logic programs can be designed that exhibit parallelism, and that the use of heuristic information translates into speedup in obtaining answers.

KEY WORDS: Parallel logic programming; ordered structures; heuristics; problem solving.

1. INTRODUCTION

Rule based deduction is a classic technique in Artificial Intelligence with a wide range of applications such as Expert Systems and Problem Solving.

---

1 This work has been supported by the FICYT of the Principado de Asturias under Project PB-TIC-9703.
2 Centro de Inteligencia Artificial, Universidad de Oviedo en Gijón Campus de Viesques, E-33271 Gijón, Spain. E-mail: [ramiro,camino,puente,calonso]@aic.uniovi.es or http://www.aic.uniovi.es/PlpyP/pro/presentacion.htm.
In these domains, the complexity of the computations can be very high and many problems tend to be combinatorially explosive. Hence, it is to be expected that their computation time will be drastically reduced with a parallel execution scheme. On the other hand, logic programming languages have become a powerful tool to express symbolic computations which often appears in this context. The most extended implementation of these languages is Prolog. The procedural semantics of this language involve a sequential control strategy with a classic depth-first and backtracking scheme, and so the execution time often becomes unacceptable when solving some instances of these problems.

These facts, together with the ability of the language of logic to express parallel computations, has meant that parallel schemas for logic programs evaluation appears to be an interesting field of research in order to improve the efficiency of deductive systems. Accordingly, several models for parallel interpretation of logic programs have been proposed over the last years. Most of them exploit one or the two main sources of parallelism of logic programs: AND parallelism and OR parallelism. AND parallelism consists of the simultaneous evaluation of several literals of a query, whereas OR parallelism consists of exploiting several rules with the same conclusion at the same time. These kinds of parallelism can be clearly represented by AND/OR trees. Hence, computations in parallel models are usually represented by means of these trees; every model having its own variant of the AND/OR tree.

In this work, we present an abstract model for parallel interpretation of logic programs. The model was designed to exploit the two main sources of parallelism of logic programming, as well as some secondary sources. The model is abstract in the sense of being independent of every target machine in which it can be subsequently implemented.

On developing a parallel model for logic programming, several problems arise with respect to sequential approaches. For example, when exploiting OR parallelism we have to deal with multiple bindings to the variables of the literals. Moreover, if we exploit AND parallelism at the same time, we have to join the multiple bindings from various literals of a given query. What frequently occurs is that most of the computed solutions to a literal are not compatible with the solutions to the remaining literals and then there are lots of processes whose work turns out to be useless. On the other hand, it is common for the same subgoal to be solved various times when a query is evaluated with respect to a program; this occurs not only with parallel models but also with sequential SLD resolution. Therefore, when designing a parallel model for evaluating logic programs, we have to consider whether or not the management cost that parallelism introduces overcomes the speedup obtained by means of parallelism.