Next Generation Logic Programming Systems

Gopal Gupta

In the last 30 years logic programming (LP), with Prolog as the most representative logic programming language, has emerged as a powerful paradigm for intelligent reasoning and deductive problem solving. In the last 15 years several powerful and highly successful extensions of logic programming have been proposed and researched.

These include *constraint logic programming (CLP)*, *tabled logic programming*, *inductive logic programming*, *concurrent/parallel logic programming*, *Andorra Prolog* and adaptations for non-monotonic reasoning such as *answer set programming*. These extensions have led to very powerful applications in reasoning, for example, CLP has been used in industry for intelligently and efficiently solving large scale scheduling and resource allocation problems, tabled logic programming has been used for intelligently solving large verification problems as well as non-monotonic reasoning problems, inductive logic programming for new drug discoveries, and answer set programming for practical planning problems, etc.

However, all these powerful extensions of logic programming have been developed by extending standard logic programming (Prolog) systems, and each one has been developed in isolation from others. While each extension has resulted in very powerful applications in reasoning, considerably more powerful applications will become possible if all these extensions were combined into a *single system*. This power will come about due to search space being dramatically pruned, reducing the time taken to find a solution given a reasoning problem coded as a logic program. Given the current state-of-the-art, even two extensions are not available together in a single system.

Realizing multiple or all extensions in a single framework has been rendered difficult by the enormous complexity of implementing reasoning systems in general and logic programming systems in particular. Implementing standard Prolog systems is itself quite complex due to the built-in unification and backtracking that have to be supported. Extending a Prolog system to incorporate constraints, or tabling, or parallelism further presents formidable challenges. As a result, even though having multiple extensions in a single system will result in considerably more efficient deductive procedures and more powerful applications, researchers have not ventured further.

In this project, we are attempting to build an efficient reasoning system that combines constraint logic programming, tabled logic programming, and parallel logic programming *all in a single system*. This will result in a logic programming system that will find solutions faster even if the search space is very big. Our main insight, on which our efforts are based, is to design *extremely simple* yet *efficient* implementation techniques for incorporating each of the extensions in a Prolog system.

Reference: G. Gupta, E. Pontelli, K. Ali, M. Carlsson, M. Hermenegildo. Parallel Execution of Prolog Programs: A Survey. In ACM Transactions on Programming Languages and Systems}, Vol 23, No. 4, pp. 472-602.