Definition of a Parallel Execution Model with Abstract State Machines[10]

Zsolt Németh

LOGFLOW is a fine-grained all-solution parallel (reduced) Prolog system for distributed memory architectures. Its abstract execution model is logicflow [3] that can be considered as a kind of macro dataflow scheme, whereas its abstract machine model is the Distributed Data Driven Prolog Abstract Machine [4] (3DPAM). 3DPAM tries to make a connection between a dataflow based execution model and a kind of von Neumann physical architecture.

However a new, hybrid multithreaded architectural platform offers the possibility to create a more efficient Prolog abstract machine. Its ability to hide latencies due to remote memory access or synchronisation (multithreading) opens a new way for representing Prolog data (heap) and managing the variables. On the other hand, its hybrid feature, i.e. support for both the fast sequential and dataflow execution, is close to the macro dataflow model of LOGFLOW and makes possible an efficient realisation of dataflow nodes and token streams. To exploit the latter property at the abstract machine level, a new abstract execution model is necessary, too. The new execution model can be derived from the logicflow in three steps by changing the way how solution streams are separated, the way how solutions are propagated and by grouping together elementary nodes. Whereas the gain in efficiency is obvious, it is not the case for correctness and semantical equivalence of the models.

Abstract State Machines (Gurevich's ASMs, formerly known as evolving algebras) offer a way for the design and analysis of complex hardware and software systems [1][2]. They are similar to Turing machines in a sense that they simulate algorithms yet, they are able to describe semantics at arbitrary levels of abstraction. An ASM consists of a finite set of transition rules by which the system is driven form state to state, each represented by sets with relations and functions (algebras). By refinement steps a 'more abstract' model can be turned into a 'more concrete' one and by relating their states and transition rules (by proof mapping), their relative correctness and completeness can be proven. In several refinement steps the equivalence of different models can be shown.

In this paper the redesign steps of the logicflow model are presented. The original logicflow model is described by an ASM then the successive steps of the modifications are introduced by new, refined ASMs. The sequence of refinements results the new abstract execution model. If all subsequent models can be mapped to their sibling models with respect to correctness, the original logicflow and the resulted one can be stated as functionally equivalent. Furthermore, the execution model can be related to sequential Prolog models showing different properties.

ASMs are also able to help the design process from the abstract execution model to the abstract machine model. While the previous transition between two execution models remained at the same level of abstraction, the principles of abstract execution model are turned into more specific forms of abstract machine at lower and lower level of abstraction by ASMs. The paper tries to give an insight into the design of compilation schemes and abstract instructions by ASMs.

The aim of the paper is defining a new execution model together with its abstract machine and showing their correctness with respect to their ancestor model and the usual Prolog execution. The ASM methodology seems to be a proper framework to fulfil the two tasks at the same time.

References

[1] E. Börger: High Level System Design and Analysis using Abstract State Machines. ASM Workshop, Magdeburg, 21-22 September, 1998.

[2] Yu. Gurevich: Evolving Algebras: An Attempt to Discover Semantics. In: Current Trends in Theoretical Computer Science. Eds. G. Rozenberg, A. Salomaa, World Scientific, 1993. pp. 266-292.

[3] P. Kacsuk: Execution Models for a Massively Parallel Prolog Implementation. Journal of Computers and Artificial Intelligence, Vol. 17, No. 4. Slovak Academy of Sciences, 1998 (part 1) and Vol. 18, No. 2., 1999 (part 2)

[4] P. Kacsuk: Distributed Data Driven Prolog Abstract Machine. In: P. Kacsuk, M.J.Wise: Implementations of Distributed Prolog. Wiley, 1992.

[10]The work reported in this paper was supported by the National Research Grant (OTKA) registered under No. T-022106.