Towards a Systematic Approach to Representing Knowledge in Declarative Logic Programming (INVITED TALK)

Michael Gelfond

Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968
mgelfond@cs.utep.edu

"The only effective way to raise the confidence level of a program significantly is to give a proof of its correctness. But one should not first make the program and then prove its correctness, because then the requirement of providing the proof would only increase the poor programmer's burden. On the contrary: the programmer should let correctness proof and program grow hand in hand. ...If one first asks oneself what the structure of a convincing proof would be and, having found this, then construct a program satisfying this proof's requirements, then these correctness concerns turn out to be a very effective heuristic guidance."

E. Dijkstra, The Humble Programmer

The main goal of this talk is to present some recent results related to representing knowledge in declarative logic programming. We are looking for some insights into the ways to specify knowledge, to gradually transfer an initial specification into an executable logic program and to insure the correctness of this transformation. We hope that such insights will help to facilitate the construction of correct and efficient knowledge based query answering systems. Our approach is complementary to the work on program development in Prolog (see for instance [Dev90]) and doesn't depend on any particular algorithm for executing logic programs. Instead it concentrates on the earlier stages of the program development. We will build on some basic ideas and results from [GP96] where the authors argue that the process of representing knowledge in LP can be divided into three (mutually dependent) parts:

– Elaboration of a natural language description of the domain, which leads to a precise mathematical characterization S of the statements which are true in the domain.
– Construction of a declarative logic program T satisfying the (elaborated) specification S. Statements which are true in S can in principle be computed using the entailment relation of T. An inference mechanism associated with T can be used for rapid prototyping.
– Design of an efficient program Π computing (or approximating) the entailment relation of T. Π will be a logic program whose construction from T can be partially automated.

In [GP96] the authors suggest that, at least in some cases, a formal (elaborated) specification of the domain can be obtained from specifications of its parts by applying certain operators on specifications called *specification constructors* and that a better understanding of these operators can substantially facilitate the programming process by providing the programmer with a useful heuristic guidance. In this talk we will further substantiating this claim by introducing a new constructor called *input opening* and discussing some results which help to transform specifications built by applying this constructor to declarative logic programs. The notion of input opening presented in this talk is a generalization of earlier notion of interpolation from [BGK93].

The method of specifying knowledge representation problems via specification constructors and of using these specifications for the development of their logic programming representations will be illustrated by design of a simple, but fairly powerful program representing simple hierarchical domains.

References

[BGK93] C. Baral, M. Gelfond, and O. Kosheleva. Approximating general logic programs, *Proc. ILPS*, pp. 181-198, 1993.

[Dev90] Y. Deville. *Logic Programming: systematic program development*, Clark, K., series editor, Addison-Wesley Publishing Co., 1990.

[GP96] M. Gelfond and H. Przymusinska. Towards a theory of elaboration tolerance: logic programming approach. *International Journal of Software Engineering and Knowledge Engineering*, 6(1):89-112, 1996.