Electronic Communications of the EASST
Volume X (2012)

Proceedings of the XII Spanish Conference on Programming and Computer Languages (PROLE 2012)

Semantics of structured normal logic programs

Edelmira Pasarella, Fernando Orejas, Elvira Pino, Marisa Navarro

2 pages

Guest Editors: María-del-Mar Gallardo
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Semantics of structured normal logic programs

Edelmira Pasarella[1], Fernando Orejas[1], Elvira Pino[1], Marisa Navarro[2] *

[1] Departament de LSI, Universitat Politècnica de Catalunya, Jordi Girona, 1-3. 08034 Barcelona, Spain
[2]Departamento de LSI, Universidad del País Vasco, Paseo Manuel de Lardizabal, 1, Apdo 649, 20080 San Sebastián, Spain

Abstract: In this paper we provide semantics for normal logic programs enriched with structuring mechanisms and scoping rules. Specifically, we consider constructive negation and expressions of the form $Q \supset G$ in goals, where Q is a program unit, G is a goal and $\supset$ stands for the so-called embedded implication. Allowing the use of these expressions can be seen as adding block structuring to logic programs. In this context, we consider static and dynamic rules for visibility in blocks. In particular, we provide new semantic definitions for the class of normal logic programs with both visibility rules. For the dynamic case we follow a standard approach. We first propose an operational semantics. Then, we define a model-theoretic semantics in terms of ordered structures which are a kind of intuitionistic Beth structures. Finally, an (effective) fixpoint semantics is provided and we prove the equivalence of these three definitions. In order to deal with the static case, we first define an operational semantics and then we present an alternative semantics in terms of a transformation of the given structured programs into flat ones. We finish by showing that this transformation preserves the computed answers of the given static program.

Keywords: semantics, normal logic programs, embedded implication, visibility rules, structuring mechanism, intuitionistic structures

1 Summary

Regarding dynamic normal logic programs, the intuition of the interpretation of embedded implications is that, given a program P, to prove the query $Q \supset G$ it is necessary to prove G with the program $P \cup Q$. This is formalized in a seminal work by Miller, using the inference rule shown in Figure 1. In Miller's approach, both implications, clausal implication and embedded implication,

$$\frac{P \cup Q \vdash_{dyn} G}{P \vdash_{dyn} Q \supset G}$$

Figure 1: Dynamic rule

* This work has been partially supported by the Spanish CICYT project FORMALISM (ref. TIN2007-66523) and by the AGAUR Research Grant ALBCOM (ref. SGR 20091137)

are interpreted as intuitionistic implications. In particular, a main result of this work is that the proof-theoretic semantics for this kind of programs can be given in terms of intuitionistic logic. Moreover, he proposed a model-theoretic semantics in terms of Kripke models. In our paper we extend this approach to the case of programs that also include negation. First, we introduce an operational semantics for the class of normal logic programs with embedded implication. This semantics is an extension of constructive negation with the above rule to handle implication goals. Extending similarly the model-theoretic semantics, both to define the logical semantics and the least fixpoint semantics of a program, is quite more involved. In particular, a main difficulty comes from the non-monotonic nature of negation in logic programming which does not fit well with modularity. We have the intuition that both connectives could have a natural and reasonably simple semantics in terms of intuitionistic (Beth) models and this semantics would make more explicit the intuitionistic nature of negation in logic programming already pointed out by other authors. Therefore, we define the Beth models that capture our intuition together with their associated forcing relation. Then, we introduce an immediate consequence operator showing that it is monotonic and continuous. Moreover, we show that the least fixpoint of this operator coincides with the least model of the given program. Finally, the operational semantics is proved to be sound and complete with respect to the least fixpoint semantics.

To deal with static normal logic programs, we consider the same kind of programs as in previous case, but interpreting embedding with static scoping. The rule for handling implications in this case is shown in Figure 2. The intuition of the notation $P|Q \vdash_{st} G$ is that we want to solve G in

$$\frac{P|Q \vdash_{st} G}{P \vdash_{st} Q \supset G}$$

Figure 2: Static rule

a scope consisting of two nested blocks of definitions, P and Q, where Q is *local* to P, i.e. $P|Q$ represents this kind of block inclusion. To solve the goal G in the scope of $P|Q$ we can use any definition that is present in either P or Q. This is just as in block-structured procedural languages where, for solving a reference in a given scope, we may use any visible definition from any of the blocks which are global to that reference. To define a semantics to this class of programs, instead of developing a new framework to define a model-theoretic semantics, we show how these programs can be transformed into standard normal programs. Moreover, we prove that this translation is sound and complete with respect to the operational semantics of the extended programs. In addition, it must be pointed out that this transformation is easy to implement, which means that we can easily build this kind of extension on top of a standard logic programming language. This approach has been used in previous work to deal with positive propositional static programs and to translate modal logic programs with embedded implication into Horn programs. To achieve our goal, we first introduce an operational semantics for the class of static normal logic programs with embedded implications. Then, we present the transformation semantics and, finally, we prove the soundness and completeness of the transformation.