Operational Semantics for DyLPs*

F. Banti[1], J.J. Alferes[1], and A. Brogi[2]

[1] CENTRIA, Universidade Nova de Lisboa, Portugal
[2] Dipartimento di Informatica, Università di Pisa, Italy

Abstract. Theoretical research has spent some years facing the problem of how to represent and provide semantics to updates of logic programs. This problem is relevant for addressing highly dynamic domains with logic programming techniques. Two of the most recent results are the definition of the refined stable and the well founded semantics for dynamic logic programs that extend stable model and well founded semantic to the dynamic case. We present here alternative, although equivalent, operational characterizations of these semantics by program transformations into normal logic programs. The transformations provide new insights on the computational complexity of these semantics, a way for better understanding the meaning of the update programs, and also a methodology for the implementation of these semantics. In this sense, the equivalence theorems in this paper constitute soundness an completeness results for the implementations of these semantics.

1 Introduction

In recent years considerable effort was devoted to explore the problem of how to update knowledge bases represented by logic programs (LPs) with new rules. This allows, for instance, to better use LPs for representing and reasoning with knowledge that evolves in time, as required in several fields of application. The LP updates framework has been used, for instance, as the base of the MINERVA agent architecture [14] and of the action description language EAPs [4].

Different semantics have been proposed [1, 2, 5, 6, 8, 15, 18, 19, 23] that assign meaning to arbitrary finite sequences $P_1, \ldots, P_m$ of logic programs. Such sequences are called *dynamic logic programs* (DyLPs), each program in them representing a supervenient state of the world. The different states can be seen as representing different time points, in which case P_1 is an initial knowledge base, and the other $P_i s$ are subsequent updates of the knowledge base. The different states can also be seen as knowledge coming from different sources that are (totally) ordered according to some precedence, or as different hierarchical instances where the subsequent programs represent more specific information. The role of the semantics of DyLPs is to employ the mutual relationships among different states to precisely determine the meaning of the combined program comprised of all individual programs at each state. Intuitively, one can add at the end of

* This work was supported by project POSI/40958/SRI/01, FLUX, and by the European Commission within the 6th Framework P. project Rewerse, no. 506779.

C. Bento, A. Cardoso, and G. Dias (Eds.): EPIA 2005, LNAI 3808, pp. 43–54, 2005.

the sequence, newer rules or rules with precedence (arising from newly acquired, more specific or preferred knowledge) leaving to the semantics the task of ensuring that these added rules are in force, and that previous or less specific rules are still valid (by inertia) only as far as possible, i.e. that they are kept as long as they are not rejected. A rule is rejected whenever it is in conflict with a newly added one (*causal rejection of rules*). Most of the semantics defined for DyLPs [1, 2, 5, 6, 8, 15] are based on such a concept of causal rejection.

With the exception of the semantics proposed in [5], these semantics are extensions of the stable model semantics [10] to DyLPs and are proved to coincide on large classes of programs [8, 11, 13]. In [1] the authors provide theoretical results which strongly suggest that the refined semantics [1] should be regarded as the proper stable model-like semantics for DyLPs based on causal rejection. In particular, it solves some unintuitive behaviour of the other semantics in what regards updates with cyclic rules.

As discussed in [5], though a stable model-like semantics is the most suitable option for several application domains[1] other domains exist, whose specificities require a different approach. In particular, domains with huge amount of distributed and heterogenous data require an approach to automated reasoning capable of quickly processing knowledge, and of dealing with inconsistent information even at the cost of losing some inference power. Such areas demand a different choice of basic semantics, such as the well founded semantics [9]. In [5] a well founded paraconsistent semantics for DyLPs (WFDy) is defined. The WFDy semantics is shown to be a skeptical approximation of the refined semantic defined in [1]. Moreover, it is always defined, even when the considered program is inconsistent, and its computational complexity is polynomial w.r.t. the number of rules of the program. For these reasons we believe that the refined and the well founded semantics for DyLPs are useful paradigms in the knowledge representation field and hence implementations for computing both semantics are in order.

The existing definitions of both semantics are purely declarative, a feature that provides several advantages, like the simplicity of such definitions and the related theorems. However, when facing computational problem like establishing computational complexity and programming implementations, a more operational approach would have several advantages. For providing an operational definition for extensions of normal LPs, a widely used technique is that of having a transformation of the original program into a normal logic program and then to prove equivalence results between the two semantics. In logic programs updates this methodology has been successfully used several times (see, for instance, [2, 8]). Once such program transformations have been established (and implemented), it is then an easy job to implement the corresponding semantics by applying existing software for computing the semantics of normal LPs, like DLV [7] or smodels [20] for the stable model semantics, or XSB-Prolog [22] for the well founded semantics. Following this direction, we provide two transfor-

[1] In particular, the stable model semantics has been shown to be a useful paradigm for modelling NP-complete problems.