Layered Models Top-Down Querying of Normal Logic Programs

Luís Moniz Pereira and Alexandre Miguel Pinto

Centro de Inteligência Artificial (CENTRIA)
Universidade Nova de Lisboa
2829-516 Caparica, Portugal
{lmp,amp}@di.fct.unl.pt

Abstract. For practical applications, the use of top-down query-driven proof-procedures is essential for an efficient use and computation of answers using Logic Programs as knowledge bases. Additionally, abductive reasoning on demand is intrinsically a top-down search method. A query-solving engine is thus highly desirable.

The current standard 2-valued semantics for Normal Logic Programs (NLPs), the Stable Models (SMs) semantics, does not allow for top-down query-solving because it does not enjoy the relevance property — and moreover, it does not guarantee the existence of a model for every NLP. To overcome these current limitations we introduce here a new 2-valued semantics for NLPs — the Layered Models semantics — which conservatively extends the SMs, enjoys relevance and guarantees model existence among other useful properties. Moreover, for existential query answering there is no need to compute total models, but just the partial models that sustain the answer to the query, or one might simply know a model one exists without producing it; relevance ensures these can be extended to total models.

A first implementation of a query-solving engine based on this new semantics is presented and described here. It uses the XSB-Prolog engine and its XASP interface to Smodels, thereby providing a useful tool built as a hybrid of the two systems and taking advantage of the best of each.

Conclusions and further work end the paper.

Keywords: Smodels, XSB-XASP, Relevance, Semantics.

1 Introduction

The semantics of Stable Models (SM) is a cornerstone for the definition of some of the most important results in logic programming of the past two decades, providing an increase in logic programming declarativity and a new paradigm for program evaluation. When we need to know the 2-valued truth value of all the literals in a logic program for the problem we are modeling and solving, the only solution is to produce complete models. In such a case, tools like SModels [13] or DLV [5] can be adequate because they can indeed compute whole models. However, the lack of some important properties of language semantics, like relevance, cumulativity and guarantee of model existence (enjoyed by, say, Well-Founded Semantics [10] (WFS)), somewhat reduces its applicability.
in practice, namely regarding abduction, creating difficulties in required pre- and post-processing. But WFS in turn does not produce 2-valued models, though these are often desired, nor guarantees 2-valued model existence.

Furthermore, in SM semantics, in an abductive reasoning situation, computing the whole model entails pronouncement about each of the abducibles whether or not they are relevant to the problem at hand, and subsequently filtering the irrelevant ones. When we just want to find an existential answer to a query, we either compute a whole model and check if it entails the query (the way SM semantics does), or, if the underlying semantics we are using enjoys the relevance property — which SM semantics do not— we can simply use a top-down proof-procedure (à la Prolog), and abduce by need. In this second case, the user does not pay the price of computing a whole model, nor the price of abducting all possible abducibles or their negations, and then filtering irrelevant ones, because the only abducibles considered will be those needed for answering the query.

The current standard 2-valued semantics for NLPs, the Stable Models [11] semantics, does not allow for top-down query-solving precisely because it does not enjoy the relevance property — and moreover, does not guarantee the existence of a model. Furthermore, frequently there is no need to compute whole models, like its implementations do, but just the partial models that sustain the answer to a query. Relevance ensures these can be extended to whole models.

To overcome these inherent limitations we developed a new 2-valued semantics for NLPs— the Layered Models (LM) semantics— which conservatively extends the SMs, and enjoys relevance and guarantee of model existence and other useful properties.

The core reason SM semantics fails to guarantee model existence for every NLP is that it does not provide a semantics to Odd Loops Over Negation (OLONs)[1]. In fact, the SM semantics community uses its inability to handle odd loops as a means to write Integrity Constraints (ICs).

Example 1. Odd Loop Over Negation as Integrity Constraint. Indeed, using Stable Models, one would write an IC in order to prevent $X$ being in any model with the single rule for some atom ‘$a$’: $a \leftarrow \neg a, X$. Since the SM semantics cannot provide a semantics to this rule whenever $X$ holds, this type of OLON is used as IC.

The LM semantics provides a semantics to all NLPs. ICs are implemented with rules for reserved atom $falsum$, of the form $falsum \leftarrow X$, where $X$ is the body of the IC we wish to prevent being true. This does not prevent $falsum$ from being in some models. To avoid them the user must either conjoin goals with $\neg falsum$ or, if inconsistency examination is desired, a posteriori discard such models. LM semantics separates OLON semantics from IC compliance.

After a brief note on notation and background definitions, we present the formal definition of LM semantics and overview its useful properties. A section describing the current implementation follows and the directions in the development of the next version of our query-solving engine. Conclusions and future work close the paper.

1 OLON is a loop with an odd number of default negations in its circular call dependency path.