Application of simplification theories

Mauricio Osorio[1], Juan Carlos Nieves[1], Gabriel Cervantes[2]

[1] Universidad de las Americas
CENTIA
Sta. Catarina Martir,
Cholula, Puebla
72820 Mexico
josorio@mail.udlap.mx
[2] Benemerita Universidad Autonoma de Puebla
Escuela de Ciencias de la Electronica
75579 Puebla, Mexico
gcervantes@kim.ece.buap.mx

1 Abstract

In this abstract we present different applications of "simplification of theories". By simplification of theories we understand a set of relations defined over a class of theories with a fixed language. The only two general properties that this relations respect are: First, that they are polynomial time computable. Second, if a theory P_1 is related to P under a transformation (that is P_1 is obtained from P using a transformation) and m is a model of P_1, then m is also a model for P.

We discuss applications in three different fields in applied logic: First order theory proving, Well behaved semantics and Answer set programming.

In first order theory proving, given a consistent first order theory T and an atom a, we may be interested in the derivability of a by T, that is, $T \models a$? Using OTTER the problem is traduced as showing that $T \cup \{\neg a\}$ is inconsistent. Unfortunately, this may cause a loop in the process (using OTTER, a well known theorem proving system) when $T \cup \{\neg a\}$ is consistent. In general terms we propose the use of"simplification of theories" that help us to construct a model for $Cl\{(T \cup \{\neg a\}\}$, where Cl denotes the clausal form of the given theory. If we succeed in finding such a model, then $T \cup \{\neg a\}$ has a model and therefore $T \not\models a$. We however need some (strong) conditions on T to be able to apply this method.

We turn now to discuss our applications over well behaved semantics. Of the major semantics proposed for logic programs with negation as failure, the well founded semantics has proved to have appealing and enduring features. It has its advantages and drawbacks. WFS is defined for a larger class of programs and admits an efficient computation, but it has been argued that WFS is by design overly careful in deciding about the falsity of some atoms, leaving them undefined. Extensions of WFS has then been proposed. Simplification of theories (program transformation in this case) has been used to characterize several

semantics as well as to define new semantics that extend WFS. This line of research was started by Dix. [1] Given a logic program P and a certain set of transformations T we apply these transformations to the program P and find a reduced program P_1, that we call the normalform of program P. Then the set of derivables literals is more simple. Of course this set of transformations must be confluent and terminating in order to guarantee a unique final program. We already have proposed some transformations rules that respect this last condition and thus we combine methods from rewriting with logic programming technology and we get a powerful framework for investigating the semantics of logic programs. Most of the well-known semantics are induced by confluent systems.

In a recent book [2] the authors (Brewka, Dix and Konolige) introduced the notion of well-behaved semantics, and aim at a classification of well-behaved semantics according to other, clearly formulated declarative properties. In particular the mentioned book presents three conjectures (conjectures 7.20, 7.21 and 7.22) stating that there are not well-behaved semantics satisfying certain properties other than the currently known semantics. Recently Osorio and Dix show that the first conjecture is false. Also Nieves and Cervantes show that the second conjecture is also false. In both cases the proofs were based on our discussed approach. We have supported reasons for expecting that the third conjecture is also false, and our current research is in this direction.

Our final application is in answer set programming. Under the stable-model semantics, a program P specifies a family of subsets of the Herbrand universe, determined by the collection of its stable models. Each of these subsets represents a possible answer to the problem encoded by P.

Dloop is a program transformation rule (for disjunctive programs) introduced recently. We have proved that the stable semantics is invariant under Dloop. Francois Fages has shown that for tight normal programs, the supported models semantics is equivalent to stable models semantics. We have generalized the result by Fages from normal programs to disjunctive programs. We define a reduction system, sys that includes Dloop and some other well known transformation rules that are correct with respect to the stable semantics. It tuns out that sometimes sys can transform a non tight program into a tight program. We illustrate how can we apply our results to compute stable models efficently. We have several open lines of research about this last issue.

References

1. Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Well-founded Semantics: Confluent Calculi and Iterated GCWA. *Journal of Automated Reasoning*, 20(1):143–165, 1998. (Extended abstract appeared in: Characterizing D-WFS: Confluence and Iterated GCWA. *Logics in Artificial Intelligence, JELIA '96*, pages 268–283, 1996. Springer, LNCS 1126.).
2. Gerd Brewka, Jürgen Dix, and Kurt Konolige. *Nonmonotonic Reasoning: An Overview*. CSLI Lecture Notes 73. CSLI Publications, Stanford, CA, 1997.