On the strong completion of logic programs

Phan Minh Dung

Division of Computer Science, Asian Institute of Technology
GPO Box 2754, Bangkok 10501, Thailand

ABSTRACT

A new completion theory for logic programming called strong completion, is introduced. Similar to the Clark's completion, the strong completion can be interpreted either in two-valued or three-valued logic. We show that

* Two-valued strong completion specifies the stable semantics.
* Three-valued strong completion specifies the well-founded semantics.

Since the strong completion of a logic program P is also a circumscription of P, the open problem as whether or not there exists a circumscription of a logic program P which specifies the stable semantics as well as the well-founded semantics of P, is solved.

We show that the call-consistency condition is sufficient for a logic program to have a stable model. Further we prove that the stable semantics is equivalent to the well-founded semantics if the program is strict and call-consistent.

Keywords Logic programming, negation, predicate completion, stable models, well-founded models, circumscription, two-valued logic, three-valued logic.
1. INTRODUCTION

The semantics of negation is one difficult problem in logic programming. To a first approximation, the semantics may be defined by the Clark's completion [CK]. Given a logic program \( P \), the completion of \( P \), \( \text{comp}(P) \), consists of some equality axioms plus a completed definition of each predicate symbol. Roughly, this completed definition is obtained by replacing the "if" by "iff". The completion of a program can be interpreted either in the two-valued logic [CK] or in a three-valued logic [F]. While the three-valued completion is always consistent, this is not the case for the two-valued completion. But if the program is call-consistent [K,S], then two-valued completion is consistent, too. The three-valued semantics is weaker than the two-valued, in the sense that every query supported in the three-valued semantics is also supported in the two-valued semantics but not conversely. But if the program is strict and call-consistent then these two semantics are equivalent [K].

However, the Clark's completion does not always capture the intended meaning of the program. For example, let \( P \) consist of the single clause \( p \leftarrow p \). Intuitively, we expect that any meaningful semantics of \( P \) would imply that \( p \) is false. But since the completion of \( P \), \( \text{comp}(P) \), is \( p \leftarrow \lnot p \) we can not conclude from \( \text{comp}(P) \) that \( p \) is false. Thus it is necessary to find new ways for specifying the intuitive meaning of logic programs. There are two ways to solve this sort of problems: The two-valued stable model approach and the three-valued well-founded model approach.

The stable semantics of a program \( P \) is specified by the set of stable models of \( P \). A stable model is defined as one that is able to reproduce itself by the Gelfond-Lifschitz transformation [GL]. One problem of the stable semantics is that not very logic program has a stable model. This problem is similar to the consistency problem of the Clark's completion in two-valued logic.

The well-founded semantics is defined by the well-founded model. The well-founded model is the least fixed point of a monotonic operator. Similarly to the three-valued semantics of the Clark's completion, a distinguished feature of the well-founded semantics is that every logic program has an unique well-founded model [GRS]. A proof theory based on the SLS-resolution is also given for well-founded semantics [R,PT2].

One common problem for stable semantics and well-founded semantics is