The Strong Semantics for Logic Programs

JIANHUA CHEN jianhua@bit.csc.lsu.edu
SUKHAMAY KUNDU kundu@bit.csc.lsu.edu
Computer Science Department, Louisiana State University Baton Rouge, LA 70803, USA

Abstract. Recently, the well-founded semantics of a logic program \( P \) has been strengthened to the well-founded semantics-by-case (WFc) and this in turn has been strengthened to the extended well-founded semantics (WFE). Both \( \text{WF}_C(P) \) and \( \text{WF}_E(P) \) have the logical consequence property, namely, if an atom \( A_j \) is true in the theory \( \text{Th}(P) \), then \( A_j \) is true in the semantics as well. However, neither \( \text{WF}_C \) nor \( \text{WF}_E \) has the GCWA property, i.e., if an atom \( A_j \) is false in all minimal models of \( P \), \( A_j \) may not be false in \( \text{WF}_C(P) \) (resp. \( \text{WF}_E(P) \)). We extend the ideas in \( \text{WF}_C \) and \( \text{WF}_E \) to define a strong well-founded semantics \( \text{WF}_S \) which has the GCWA property. The strong semantics \( \text{WF}_S(P) \) is defined by combining GCWA with the notion of derived rules. Here we use a new Type-III derived rules in addition to those used in \( \text{WF}_C \) and \( \text{WF}_E \). The relationship between \( \text{WF}_S \) and \( \text{WF}_C \) is also clarified.

Keywords: logic program, declarative semantics, GCWA-property, derived rules

1. Introduction

This paper is an extended version of (Chen and Kundu, 1991). We assume throughout that the logic program \( P \) is propositional, i.e., it does not contain predicates and variables. The general approach in defining a declarative (model based) semantics of a logic program \( P \) consists of the following two steps. We write \( \Delta(P) = \{ \text{clause}(p) : p \in P \} \), where \( \text{clause}(p) \) denotes the clause from of \( p \) (e.g., if \( p: "a \leftarrow b, \neg c" \), then \( \text{clause}(p) = a \lor \neg b \lor c \)). Also, we use the short notation \( \text{Th}(P) \) for \( \text{Th}(\Delta(P)) \); similarly, \( \text{CIRC}[P] \) for the circumscription of \( \Delta(P) \), \( \text{MM}(P) \) for the minimal models of \( \Delta(P) \), etc. Note that \( \text{CIRC}[P] \) is independent of the syntactic structure of the rules in \( P \). The models of \( \text{CIRC}[P] \) are precisely \( \text{MM}(P) \).

(1) Define a preference criterion (a partial order) among the propositions in \( P \) using the syntactic structure of the rules in \( P \) and perhaps some information from \( \text{CIRC}[P] \). This in turn gives rise to a preference criterion among the minimal models of \( \Delta(P) \). (The preference criterion may be defined by an iterative process; e.g., as in the case of stratified logic programs.)

(2) Define the semantics of \( P \) to be the set of most preferred minimal models of \( P \) (or, equivalently, the set of formulas which are satisfied by each preferred model) according to the preference criterion in (1). A proposition (or, more generally, a formula) is then defined to be true in \( P \) if any only if it is true in each preferred minimal model.

Since the preferred minimal models usually form a proper subset of \( \text{MM}(P) \), they define a stronger semantics of \( P \) than that given by \( \text{CIRC}[P] \). The distinctions among various declarative semantics arises in two ways: (i) use of different amount of information from \( \text{CIRC}[P] \) (for simplifying the rules in \( P \) and perhaps eliminating some of those rules), and (ii) use of different preference criterion in (1). The preferred minimal models often equal, or are closely related to, the models of a prioritized circumscription of \( \Delta(P) \), depending on the type of rule simplifications and rule eliminations used.
We say a semantics $\Sigma_1$ for logic programs is stronger (tighter or more refined) than another semantics $\Sigma_2$, denoted by $\Sigma_1 \triangleright= \Sigma_2$, if for all $P$ we have $\Sigma_1(P) \triangleright= \Sigma_2(P)$, with inequality for some $P$, where $\Sigma_j(P)$ denotes the set of formulas which are true in $P$ for the semantics $\Sigma_j$. Ideally, one would like to define the preference criterion in (1) in such a way that there is a unique preferred model because in that case the resulting semantics $\Sigma$ has the property that for each proposition $p$, either $p \in \Sigma$ or $\neg p \in \Sigma$ and hence one can easily determine if a goal is true or false in $\Sigma$. Unfortunately, it is not always possible to define such a preference criterion that is also intuitively appealing. The (weakly) stratified programs, or more generally, the well-founded programs are but few cases which results in a unique preferred minimal model. (By a well-founded program $P$, we mean a definite program $P$ for which the well-founded semantics, WF-semantics in short, defines a complete 2-valued model. In general, the WF-semantics (Van Gelder, Ross and Schlipf, 1991) defines only a partial model.)

Unless stated otherwise, a semantics $\Sigma(P)$ will consists of a set of positive and negative literals which are true in each preferred minimal model. (In particular, it may define only a partial model of $P$.)

For the program $P_1 = \{a \leftarrow b, a \leftarrow \neg b, b \leftarrow \neg a\}$, we have $a = \text{true} = \neg b$ is the unique minimal model of $\Delta(P_1)$ and thus it should also be the preferred model. However, the WF-semantics of $P_1$ is given by $WF(P_1) = \emptyset$, i.e., none of $a$ and $b$ is assigned a true/false value. This shortcoming of WF-semantics is caused by the fact that it makes use of only a limited (literal level) information from $\text{CIRC}[P]$.

The WF-semantics-by-case (WF$_C$-semantics, in short), defined by Schlipf (1990), is known to be stronger than the WF-semantics and partially alleviates the above limitation of WF-semantics. It has the logical consequence property that if an atom $p \in \text{CIRC}[P]$, or equivalently, $p \in \Delta(P)$, then $p = \text{true}$ in WF$_C(P)$. The WF$_C$-semantics accomplishes this property by using the notion of derived rules, which essentially provide a representation of certain clauses in $\text{Th}(P)$ in the form of rules. (Some of those clauses may not be in $\Delta(P)$ and even if they are in $\Delta(P)$ the derived rules may give alternate representations for them.) In particular, all atomic clauses in $\text{Th}(P)$ are represented in this way and this gives rise to the property stated above. For the program $P_1$ above, we have $WF_C(P_1) = \{a, \neg b\} \triangleright= WF(P_1)$. The WF$_C$-semantics does not, however, have a similar property for the negative literals: if $\neg p = \text{true}$ in $\text{CIRC}[P]$, then $\neg p = \text{true}$ in $WF_C(P)$. This property is called the generalized closed world assumption property, in short, the GCWA-property.

The extended well-founded semantics (WF$_E$-semantics, in short), defined by Hu and Yuan (1991), is known to be stronger than the WF$_C$-semantics and partially alleviates the above limitation of WF$_C$-semantics. It uses the same derived rules as in the WF$_C$-semantics, but it uses a more general criterion to assign the false value to an atom and this makes it stronger than the WF$_C$-semantics. For the program $P_2 = \{a \leftarrow c, b \leftarrow \neg d, c \leftarrow \neg b, \neg d, d \leftarrow \neg a\}$, we have $WF_E(P_2) = \emptyset = WF_C(P_2) = WF(P_2)$ although $\neg c = \text{true}$ in $\text{CIRC}[P_2]$. One reason that WF$_C(P_2)$ fails to assign $\neg c = \text{true}$ is that it does not recognize that the only rule $c \leftarrow \neg b, \neg d$ for $c$ (there being no other derived rule for $c$) does not correspond to a minimal positive clause in $\text{Th}(P_2)$. Note that $\neg c = \text{true}$ in $\text{CIRC}[P_2]$ if and only if there is no minimal positive clause in $\text{Th}(P_2)$ involving $c$. Our newly defined strong semantics for $P_2$ first assigns $\neg c = \text{true}$ on the basis that the atom $c$ is not part of any minimal positive clause in $\text{Th}(P)$, and hence it recognizes that $\neg c = \text{true}$ in $\text{CIRC}[P_2]$. This in turn gives $\neg a = \text{true} = d = \neg b$. To find all minimal positive clauses in $\text{Th}(P)$, it is sufficient to consider a new type of derived rules, called Type-III, in addition to the derived