

Mixing a Default Rule with Stable Negation

Jack Minker and Carolina Ruiz

Dept. of Computer Science and Institute for Advance Computer Studies.
Univ. of Maryland. College Park, MD 20742 USA
{minker, cruizc}@cs.umd.edu

Abstract

Logic programs containing only at most one form of default negation have been studied in the literature. We describe a class of logic programs containing multiple forms of default negation. We define a meaning for these programs based on the well-founded semantics and the stable semantics. We investigate properties of the new combined semantics and calculate the computational complexity of determining the existence of models, and of skeptical and credulous reasoning. An effective procedure to construct the collection of models characterizing the semantics of a program is given. Applications to knowledge representation and knowledge base merging are presented.

Introduction

Logic programming and non-monotonic reasoning have developed several theories for interpreting negated information and for deducing it from positive data. Two widely used semantics for default negation are the Well-Founded Semantics (WFS) (Van Gelder, Ross, & Schlipf 1988) and the stable semantics (Gelfond & Lifschitz 1988) which was extended to the 3-valued case in (Przymusinski 1991). These notions of negation have been used as separate ways to interpret and to deduce negated information, i.e. each application has chosen one of these notions of negation and has applied it to all data in the application. However, it is not natural to uniformly use a single rule for negation. Thus, expressive power is gained by allowing the interaction of different theories for negation in the same application. In this paper we study logic programs that contain multiple types of default negation in addition to explicit negation. We describe this class of logic programs and define its semantics.

Background

Let \mathcal{L} denote a first order language. An *extended clause* is a clause of the form: $l_0 \leftarrow l_1, \dots, l_m, \text{not } l_{m+1}, \dots, \text{not } l_n$ where $0 \leq m \leq n$ and the l 's are

literals in the language \mathcal{L} and *not* is a default negation operator. An *extended logic program (elp)* is a (possibly infinite) set of extended clauses.

Since an *elp* is equivalent to the set of all its ground instances, we consider here only propositional *elps*. We require that \mathcal{L} contain special propositions \mathbf{t} , \mathbf{f} and \mathbf{u} , that are intended to denote *true*, *false* and *unknown*, respectively. The following *Truth Ordering* ($<_t$) among truth values is used: $\text{false} <_t \text{unknown} <_t \text{true}$.

Let P be an *elp* written in a propositional language \mathcal{L} . A 3-valued interpretation I of P is a pair $\langle I^+, I^- \rangle$ where I^+ and I^- are disjoint subsets of \mathcal{L} and such that $\mathbf{t} \in I^+$, $\mathbf{f} \in I^-$ and $\mathbf{u} \notin I^+ \cup I^-$. An interpretation I is called *total* if $I^+ \cup I^- \cup \{\mathbf{u}\} = \mathcal{L}$.

The truth value of a sentence φ w.r.t. an interpretation I is denoted by $\mathcal{V}_I(\varphi)$. Given a proposition $a \in \mathcal{L}$, $\mathcal{V}_I(a) = \text{true}$ if $a \in I^+$; $\mathcal{V}_I(a) = \text{false}$ if $a \in I^-$; and $\mathcal{V}_I(a) = \text{unknown}$ otherwise. Truth values of more complex sentences w.r.t. I are computed using the Kleene truth tables in the usual way.

The *Truth Ordering* (\preceq_t) on 3-valued interpretations is defined as follows: Given $I = \langle I^+, I^- \rangle$ and $J = \langle J^+, J^- \rangle$, $I \preceq_t J$ iff $\mathcal{V}_I(a) \preceq_t \mathcal{V}_J(a)$ for all $a \in \mathcal{L}$.

A 3-valued interpretation M is a 3-valued model of P if for every clause $H \leftarrow B$ in P , $\mathcal{V}_M(H) \geq_t \mathcal{V}_M(B)$. M is said to be a \prec_t -*minimal* 3-valued model of P if there is no 3-valued model N of P such that $N \neq M$ and $N \prec_t M$. $\mathcal{MM}_{\prec_t}(P)$ denotes the collection of \prec_t -minimal 3-valued models of P .

A semantics of an *elp* is characterized by a subcollection of its set of models. The WFS of an *elp* P is captured by a unique model. This model can be constructed in at most quadratic time in the length of P (Schlipf 1992). The 3-valued stable semantics of an *elp* is defined below. A 3-valued model M of P is called a *partial (or 3-valued) stable model* of P if M is the \prec_t -minimal model of the *Gelfond-Lifschitz (GL) transformation* P^M of P w.r.t. M defined as follows: P^M is the *elp* free of default negation obtained by replacing in every clause of P all default negated premises

$l = \text{not } c$ which are *true* (resp. *unknown*; resp. *false*) in M by the proposition **t** (resp. **u**; resp. **f**).

Proposition 1. (Przymusinski 1991) *Let P be an elp. The following relationships exist among collections of partial stable models $3\text{-STB}(P)$, stable models $2\text{-STB}(P)$ and the well-founded model $WFS(P)$ of P :*

- i. $3\text{-STB}(P) \subseteq \mathcal{MM}_{\leftarrow t}(P)$.
- ii. $2\text{-STB}(P) \subseteq 3\text{-STB}(P)$.
- iii. $W = WFS(P) \in 3\text{-STB}(P)$. Furthermore, for all $N \in 3\text{-STB}(P)$, $W^+ \subseteq N^+$ and $W^- \subseteq N^-$.

The problem of extending the partial stable (and hence the well-founded) semantics to deal with explicit negation is addressed in (Przymusinski 1991).

Logic Programs with Stable and Well-Founded Negations

In general one may have a logic program that contains multiple default negations as defined below.

Definition 1 (mnlps). A *multiple negation logic program* (*mnlp*) is a (possibly infinite) set of clauses

$$a \leftarrow \bigwedge_{i=1}^l b_i, \bigwedge_{i=1}^{n_1} \text{not}_1 c_i^1, \dots, \bigwedge_{i=1}^{n_m} \text{not}_m c_i^m \quad (1)$$

where a , the b 's and the c 's are literals, $m \geq 0$, and $\text{not}_1, \dots, \text{not}_m$ are distinct default negation operators.

Example 1. $\{a \leftarrow \text{not}_1 b, \text{not}_2 c; b \leftarrow \text{not}_2 c; c \leftarrow \text{not}_1 b, \text{not}_2 d; d \leftarrow a\}$ is an *mnlp* for $m = 2$.

An *mnlp* P can be mapped into an *elp*, denoted by P^{not} , by replacing the m operators $\text{not}_1, \dots, \text{not}_m$ with a unique generic operator *not*. We say I is an interpretation (resp. a model) of an *mnlp* P if and only if I is an interpretation (resp. a model) of P^{not} .

In the remainder of the paper we study the semantics of *mnlps* of the form (1) when $m = 2$. not_1 and not_2 are respectively interpreted using the stable and the well-founded semantics. Hence not_1 will be denoted by not_{STB} and not_2 by not_{WFS} . We call this new combined semantics the **WF3STB** semantics. Since the **WFS** is 3-valued, the 3-valued stable semantics will be used to interpret not_{STB} .

The Well-Founded Partial Stable Semantics

We characterize the semantics of an *mnlp* P using a set of canonical models, called **WF3STB**(P), as follows: We define an extension of the **GL** transformation

that, given a 3-valued model M of P , constructs a new program, $P^{M/\text{not}_{STB}}$, by replacing each literal of the form $\text{not}_{STB} c$ in P by its truth value w.r.t. M . Since $P^{M/\text{not}_{STB}}$ is free of not_{STB} , its well-founded model can be computed. If this model coincides with M then we say M is a **WF3STB** model of P .

Definition 2 (WF3STB Semantics). Let P be an *mnlp* and M be any 3-valued model of P .

1. The **GL** transformation $P^{M/\text{not}_{STB}}$ of P w.r.t. M is the *elp* free of not_{STB} obtained by replacing in every clause of P all default negated premises $\text{not}_{STB} c$ for which $\mathcal{V}_M(\text{not } c) = \text{true}$ (resp. *unknown*; resp. *false*) by the proposition **t** (resp. **u**; resp. **f**).
2. M is a **WF3STB** model of P if

$$M = \text{WFS}(P^{M/\text{not}_{STB}}). \quad (2)$$

Example 2. Let the *mnlp* P in Example 1 have $\text{not}_1 = \text{not}_{STB}$ and $\text{not}_2 = \text{not}_{WFS}$. The **WF3STB** semantics of P is given by: $\text{WF3STB}(P) = \{M_1 = \langle \{b\}; \{a, c, d\} \rangle, M_2 = \langle \emptyset; \emptyset \rangle\}$. M_1 and M_2 satisfy the fixpoint equation (2). Example 7 shows that no other models do. Note that $3\text{-STB}(P^{\text{not}}) = \{\langle \{b\}; \{a, c, d\} \rangle, \langle \emptyset; \emptyset \rangle, \langle \{c\}; \{a, b, d\} \rangle\}$ and $\text{WFS}(P^{\text{not}}) = \{\langle \emptyset; \emptyset \rangle\}$.

The following proposition states that the **WF3STB** semantics interpolates between the **WFS** and the partial stable semantics, as was illustrated above.

Proposition 2. *Let P be an mnlp. Then $\text{WFS}(P^{\text{not}}) \in \text{WF3STB}(P) \subseteq 3\text{-STB}(P^{\text{not}})$.*

Corollary 1. *Let P be an mnlp. Then, $\text{WF3STB}(P) \subseteq \mathcal{MM}_{\leftarrow t}(P)$.*

Proposition 2 guarantees the existence of **WF3STB** models for any *mnlp* P , namely the well-founded model of P^{not} . By Part iii of Proposition 1, skeptical reasoning in the **WF3STB** semantics (determining if a literal is *true* in every **WF3STB** model of P) is equivalent to determining if the literal is *true* in the **WFS**(P^{not}). Credulous reasoning in the **WF3STB** semantics (determining if a literal is *true* in some **WF3STB** model of P) is as complex as credulous reasoning in the non-disjunctive **3-STB** semantics.

Lemma 1. *Credulous reasoning for propositional non-disjunctive elps under the partial stable semantics is NP-complete.*

Proposition 3.1. *The existence of WF3STB models for any $mnlp$ P is guaranteed.*

2. *The complexity of skeptical reasoning for $mnlps$ under the WF3STB semantics is quadratic in the length of the program.*
3. *Credulous reasoning for $mnlps$ under the WF3STB semantics is NP-complete.*

Applications and Examples

Combining Knowledge Bases. An application of the WF3STB semantics is in combining knowledge bases. Let $elps$ P and Q have meanings given by the 3-STB semantics and the WFS respectively. The meaning of the combination of P and Q is obtained by applying the WF3STB semantics to $P \cup Q$.

Example 3. Consider the $elps$ $P = \{a \leftarrow not_{STB} b; b \leftarrow not_{STB} a\}$ and $Q = \{p \leftarrow not_{WFS} q; q \leftarrow not_{WFS} p; r \leftarrow not_{WFS} s\}$. Here, $3\text{-STB}(P) = \{\langle\{a\}; \{b\}\rangle, \langle\{b\}; \{a\}\rangle, \langle\emptyset; \emptyset\rangle\}$, $WFS(Q) = \{\langle\{r\}; \{s\}\rangle\}$, and $WF3STB(P \cup Q) = \{\langle\{a, r\}; \{b, s\}\rangle, \langle\{b, r\}; \{a, s\}\rangle, \langle\{r\}; \{s\}\rangle\}$.

When P and Q are written in disjoint languages, the WF3STB models of $P \cup Q$ can be obtained from the 3-STB models of P and the well-founded model of Q by: $WF3STB(P \cup Q) = 3\text{-STB}(P) \times WFS(Q)$, where this Cartesian product denotes the set $\{\langle M^+ \cup N^+, M^- \cup N^- \rangle \mid M \in 3\text{-STB}(P) \text{ and } N \in WFS(Q)\}$. When P and Q are written in a common language, it seems that there is no automatic way to combine $3\text{-STB}(P)$ and $WFS(Q)$ to obtain $WF3STB(P \cup Q)$.

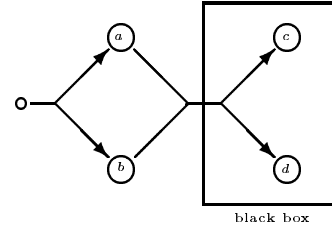
Example 4. Let $P = \{p \leftarrow not_{STB} a\}$ and $Q = \{a\}$. $3\text{-STB}(P) = \{\langle\{p\}; \{a\}\rangle\}$, $WFS(Q) = \{\langle\{a\}; \emptyset\rangle\}$, and $WF3STB(P \cup Q) = \{\langle\{a\}; \{p\}\rangle\}$.

Using Different Criteria to Jump to Conclusions. The presence of multiple kinds of default negation permits us to represent negated information in an application using alternative criteria, instead of being restricted to the same criterion uniformly as is the case when only one form of negation is available. In the case of the WF3STB semantics, the presence of not_{STB} and not_{WFS} enables us to choose whether the default negation of a literal l should be interpreted as partial stability (in the case of $not_{STB} l$) or as unfoundedness (in the case of $not_{WFS} l$).

Example 5. Let $P = P_1^{not} (= P_2^{not})$, where $P_1 = \{a \leftarrow not_{STB} b; b \leftarrow not_{STB} a; p \leftarrow not_{WFS} p; p \leftarrow$

$not_{WFS} b\}$, and $P_2 = \{a \leftarrow not_{WFS} b; b \leftarrow not_{WFS} a; p \leftarrow not_{STB} p; p \leftarrow not_{STB} b\}$. $WF3STB(P_1) = \{\langle\{a, p\}; \{b\}\rangle, \langle\{b\}; \{a\}\rangle, \langle\emptyset; \emptyset\rangle\}$, and $WF3STB(P_2) = \{\langle\emptyset; \emptyset\rangle\}$. Note that in P , a and b depend negatively on each other. In P_1 , the dependency is interpreted using the 3-STB semantics so three cases are considered: (1) a is *true* and b is *false*; (2) a is *false* and b is *true*; and (3) a and b are both *unknown*. In P_2 , the dependency is interpreted using the WFS semantics so only the third case is taken into account. Hence, although $P_1^{not} = P_2^{not}$, $WF3STB(P_1) \neq WF3STB(P_2)$.

Example 6 (A queueing problem). Consider the situation depicted below.



A queue served by two agents a and b splits the work as follows: agent a serves a request if and only if agent b does not serve the request. After either has processed the request, the result is sent to a black box. This black box contains two new agents c and d who split the remaining work as a and b do, namely agent c serves the request if and only if agent d does not.

Notice that, in the presence of a request, there are three possible scenarios to be considered: (1) a serves the initial request; (2) b serves the initial request; or (3) none of them serves the initial request. Since c and d are inside a black box, there is no way to tell from the outside which of them (if any) serves the second stage of the request, hence we conclude in each possible scenario that the identity of the second stage server is *unknown*. This situation can be modeled by the program: $P = \{a \leftarrow not_{STB} b; b \leftarrow not_{STB} a; c \leftarrow not_{WFS} d; d \leftarrow not_{WFS} c\}$. As desired, $WF3STB(P) = \{\langle\{a\}; \{b\}\rangle, \langle\{b\}; \{a\}\rangle, \langle\emptyset; \emptyset\rangle\}$.

Computing the WF3STB Semantics

In this section we describe an effective procedure to construct the collection of WF3STB models of an $mnlp$. Due to Proposition 2, the WF3STB semantics of an $mnlp$ P can be characterized as follows:

$$WF3STB(P) = \{M \in 3\text{-STB}(P^{not}) \mid M = WFS(P^{M/not_{STB}})\}.$$

Given that the WFS can be effectively computed, the existence of a procedure that computes the partial stable semantics implies that of a procedure to compute the WF3STB semantics. An effective procedure

to construct the partial stable semantics has been introduced in (Ruiz & Minker 1995). It is based on a transformation called the *3S-transformation* which applied to a disjunctive *elp* P produces a constrained logic program (free of default negation) P^{3S} , whose minimal consistent 2-valued models correspond to the 3-valued stable models of the original program P .

Example 7. Let P be the *mnlp* of Examples 1 and 2. Using the 3S-transformation, we construct $3\text{-STB}(P^{not}) = \{M_1 = \langle \{b\}; \{a, c, d\} \rangle, M_2 = \langle \emptyset; \emptyset \rangle, M_3 = \langle \{c\}; \{a, b, d\} \rangle\}$. Only M_1 and M_2 satisfy the fixpoint equation (2) so $\text{WF3STB}(P) = \{M_1, M_2\}$.

It is worth noticing that even for the propositional case, the problem of constructing the collection of partial stable models of an *elp* is not tractable.¹ This follows from the observation that one can solve the problem of credulous reasoning for the 3-STB semantics by constructing all the partial stable models of a program. Hence, by Lemma 1, this construction cannot be performed in polynomial time. This implies that the above algorithm to compute the WF3STB semantics is non-polynomial in the length of the program.

Combining Stable Negation with an Arbitrary Default Rule

As in the way in which the WF3STB semantics was defined, one can define a semantics for logic programs combining two forms of default negation, one being the stable semantics (with default operator not_{STB}) and the other being any semantics SEM (e.g. supported, inflationary) defined for the class of *elps* or disjunctive *elps* (with default operator denoted by not_{SEM}). Given such a logic program P , the GL transformation can be used to eliminate occurrences of the stable negation not_{STB} w.r.t. a model² of P . The resulting program contains only not_{SEM} so its meaning under SEM is well-defined. Hence, the combined semantics of P can be taken to be the collection of models of P^{not} satisfying the fixpoint equation $M \in \text{SEM}(P^{M/not_{STB}})$. Computational issues as to whether or not there is an effective procedure to construct this collection of models, and the complexities of different reasoning tasks must be studied on a case-by-case basis.

Conclusions

We have described a class of definite logic programs that contain two types of default negation and explicit

¹We assume here that $P \neq NP$.

²Depending on whether SEM is 2-valued or 3-valued, total or partial models should be considered here, that is, the (total) stable semantics or the partial stable semantics should be used to interpret not_{STB} .

negation, and proposed a semantics, called WF3STB, for these programs. This semantics interpolates between the WFS and the partial stable semantics. We have also provided an effective procedure to construct the collection of WF3STB models of a program.

We proved that, for the propositional case, the existence of models under this semantics is guaranteed and that skeptical and credulous reasoning are respectively quadratic and NP-complete in the length of a program. Hence, these reasoning tasks under the WF3STB semantics have the same computational complexity as under the 3-STB semantics for non-disjunctive *elps*.

We showed that the definition of the WF3STB semantics can be easily extended to larger classes of programs such as the class of extended disjunctive logic programs. Also, semantics for logic programs which combine the stable negation and a negation other than the well-founded can be obtained by mimicking the definition of the WF3STB semantics.

Applications of the WF3STB semantics to knowledge representation and knowledge base merging were presented. More applications in which this semantics is relevant are needed. Also, criteria to determine under which circumstances each type of negation is appropriate should be developed. We plan to investigate these issues in the future, as well as study the expressive power of our new semantics.

Acknowledgements

Support for this paper was provided by the Air Force Office of Scientific Research under grant number 91-0350, and the National Science Foundation under grant numbers IRI-8916059 and IRI 9300691.

References

- Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, and Bowen, eds., *Proc. of the 5th Intl. Conf. and Symp. on Logic Programming*, 1070–1080. The MIT Press.
- Przymusiński, T. C. 1991. Stable semantics for disjunctive programs. *New Gen. Comp.* 9:401–424.
- Ruiz, C., and Minker, J. 1995. Computing stable and partial stable models of extended disjunctive logic programs. In Dix, Pereira, and Przymusiński, eds., *Nonmonotonic Extensions of Logic Programming*. LNCS 927. Springer-Verlag. 205–229.
- Schlipf, J. 1992. A survey of complexity and undecidability results in logic programming. In Blair, Marek, Nerode, and Remmel, eds., *Informal Proc. of the Workshop on Structural Complexity and Recursion-theoretic Methods in Logic Programming.*, 143–164.
- Van Gelder, A.; Ross, K.; and Schlipf, J. 1988. Unfounded sets and well-founded semantics for general logic programs. In *Proc. of the 7th ACM Symp. on Principles of Database Systems.*, 221–230.