An argumentation theoretic semantics based on non–refutable falsity

José Júlio Alferes   Luís Moniz Pereira
CRIA, Uninova and DCS, U. Nova de Lisboa*
2825 Monte da Caparica, Portugal
Phone: +351 1 295 31 56   Fax: +351 1 295 56 41
{jja|lmp}@fct.unl.pt

Abstract

We contend that the well–founded semantics (WFS), for normal program, and similarly the well–founded semantics with explicit negation (*WFSX*), for extended ones are, by design, overly careful in deciding about the falsity of some atoms, by leaving them undefined.

We've dealt with this issue in normal programs and have previously defined the O–semantics, one that extends WFS by addjoining to it more negative assumptions, at the expense of undefined literals. The goal of this paper is to generalize that work to extended programs, and define a semantics for such programs that enlarges *WFSX* with more negative assumptions.

To achieve this we view default literals as arguments a rational agent can sustain along with the program. As our goal is to enlarge *WFSX*, we consider the latter as the common reasoning ground and argumentation tool of agents.

With this basis, and in order to define the semantics, we first formalize the concepts of consistent and non–refutable sets of arguments (or of hypotheses). In general several such sets may exist. So, and in order to define a unique semantics, given by a single set of additional assumptions, we introduce an additional non–refutability of arguments criterium – tenability – for always and finally prefering just one set of arguments over another.

1 Introduction

The Well Founded Semantics (WFS) [GRS91] has been proposed as a suitable semantics for normal logic programs. Unlike other semantics, WFS gives mean-

*We thank Esprit BR project Compulog 2 (no. 6810) for its support.

ing to every normal program, is cumulative [Dix91], and can be obtained by a bottom–up fixpoint construction. Because it is a relevant semantics [Dix92] it is susceptible of top–down existencial query procedures. However, as argued in [KM91, PAA92a, PAA93], the WFS is by design overly careful in deciding about the falsity of some atoms, leaving them undefined.

In [PAA92a, PAA93] we introduce a suitable form of closed world assumption (CWA), and use it to safely and undisputably assume false some of the atoms absent from the well founded model of a program. There we contend that a set S of negative literals (assumptions) *added* to a program model $MOD(P)$ by CWA must obey four precepts:

- $MOD(P) \cup S \not\models L$ for any *not* $L \in S$. I.e. the program model added with the set of assumptions identified by our CWA rule must be **consistent**.
- There is no other set of assumptions A such that $MOD(P) \cup A \models L$ for some *not* $L \in S$.
- S must be **unique**.
- S must, additionally, be **maximal**.

Based on these we've then defined the so–called O–semantics for normal logic programs, one more credulous than WFS in the sense that it accepts more negative assumptions.

A number of authors have underscored the advantages of extending logic programming with a second kind of negation $\neg$, for use in deductive databases, knowledge representation, and nonmonotonic reasoning [GL90, KS90, PAA91, Wag91]. Different semantics for extended logic programs with $\neg$-negation (ELP) have appeared [DR91, GL90, KS90, PA92, Prz90, Prz91, Wag91]. [AP92] contrasts some of these, where distinct meanings of $\neg$–negation are identified: classical, strong and explicit. It is argued there that explicit negation is preferable. The well–founded semantics with explicit negation (*WFSX*) [PA92] incorporates this prefered $\neg$–negation, and also inherits the above mentioned desirable properties of WFS (e.g. cumulativity, relevance, bottom–up fixpoint construction, cf. [Alf93]).

However, as for the WFS of normal programs, the *WFSX* of extended programs too is by design overly careful in deciding about the falsity of some atoms, leaving them also undefined.

Example 1 The extended program:

$$\begin{array}{rcl}
win(X) & \leftarrow & move(X,Y), not\ win(Y) \\
worthBetting(X) & \leftarrow & win(X) \\
bet(X) & \leftarrow & worthBetting(X) \\
\neg bet(X) & \leftarrow & not\ worthBetting(X) \\
move(a,a) & & \\
move(b,c) & &
\end{array}$$

expresses that:

- "X is a winning position if there is a move from X to Y and Y is not a winning position";
- "winning position bets are worth betting";
- "bet if a position is worth betting";
- "do not bet if it can be assumed that it is not worth betting" or, in other words, "if there is the possibility of it not being worth betting";
- "moves from position a to position a, and from b to c are legal".

c is not a winning position since it is impossible to move from c. b is a winning position because it is possible to move from b to c and c is not a winning position. a is a position of draw.

Neither $win(a)$ nor *not* $win(a)$ should hold. This is correctly handled by *WFSX*, which assigns the truth–value *undefined* to $win(a)$.

The semantics of this program should also capture the intended meaning that one should not bet in a position of draw. This is not captured by *WFSX*, which leaves $\neg bet(a)$ undefined.

Moreover note that, despite the fact that $win(a)$ is undefined, and the only rule for $worthBetting(a)$ has that atom in the body, *not* $worthBetting(a)$ could be safely assumed, thus giving $\neg bet(a)$.

The main result of this paper is an argumentation semantics for ELPs that:

- enlarges the *WFSX* of noncontradictory programs by adding more negative assumptions, thus reducing undefinedness.
- generalizes the O–semantics of normal programs to extended ones.

To achieve this we view, as usual, default literals as assumptions that might be added to (or imposed on) a program. In other words, default literals correspond to arguments a rational agent can sustain along with the program. As one of our goals is to enlarge *WFSX*, we consider this semantics the common reasoning and argumentation tool of agents. Whereas our approach is based on a pre–defined semantics, and uses argumentation to enlarge that semantics, others (e.g. [BTK93, Dun93]) use argumentation to defined the semantics from scratch. On this basis, and in order to define the semantics, in this paper we formalize the following informally expressed concepts:

First, as we'll see, a set of assumptions A is consistent with a program P if neither of them is contradicted by the *WFSX* of P when A is imposed on it. Inconsistent sets of assumptions correspond to self–refuting sets of arguments.

Then we introduce the notion of sustainable sets of assumptions (or arguments). Intuitively, a set of assumptions A is sustainable if it is consistent and there is no other set of consistent assumptions that can be adopted together with P whose joint *WFSX* contradicts A. An unsustainable set of assumptions A can be seen as refutable sets of arguments, in the sense that there exists another consistent set of assumptions B which if imposed on P produces a model that contradicts A, even if A plus P contradicts B. In the latter case both are unsustainable.

Example 2 In program P :

$$\begin{array}{lcl} c & \leftarrow & not\ b \\ b & \leftarrow & not\ a \end{array} \qquad \begin{array}{lcl} a & \leftarrow & not\ a \end{array}$$

the assumption *not a* is inconsistent because when imposed on P the resulting *WFSX* would entail a. In other words, no agent can sustain the argument *not a* because using *WFSX* makes this argument together with the program self–refuting.

The assumption *not c* is consistent but not sustainable: by imposing *not b*, c follows, and so the argument *not c* is refuted by *not b*. In other words, on the basis of P and *WFSX* rationality, an agent upholding argument *not c* can be refuted (or counterargued) by any other agent sustaining the argument *not b*.

The assumption *not b* is consistent and sustainable. An agent sustaining argument *not b* cannot be refuted by any consistent set of arguments.

To maximally reduce undefinedness we need to identify maximal sustainable sets of assumptions. In the example above $\{not\ b\}$ is the single maximal one. Naturally, in this case the semantics is defined as the result of imposing *not b* on the program.

In general several maximal sustainable sets of assumptions may exist (cf. examples 7 and 8). So, and in order to define a unique semantics given by a single set of additional assumptions, a first approach would be to add only those assumptions common to all maximal sets, i.e. those arguments that are nonrefutable and consensual. However this is too extreme a measure. In fact, and in order to further enlarge the *WFSX* with nonrefutable assumptions, one can introduce an additional criterium – tenability – for always and finally prefering one set of assumptions over another. Or, if you will, we introduce a new form of argument refutation: the tenability of a maximal set of sustainable assumptions requires that it be not refuted when conjoined with another such competing set.

The structure of this paper is as follows: we begin with a brief review of *WFSX*. Then, in section 3, we show how to consistently add nonrefutable negative assumptions to the *WFSX* of an extended program P. In section 4 we introduce tenability, define the O–semantics for extended programs, and show that it accomplishes the goals we've proposed. Finally we draw some comparisons with related work.

Proofs of all theorems are omitted for brevity, and can be found in [Alf93].

2 Preliminary definitions

In this section we define the language of extended logic programs and briefly review the *WFSX* semantics [PA92].

An extended program is a set of rules of the form:

$$L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n \quad (0 \leq m \leq n)$$

where each L_i is an objective literal. An objective literal is either an atom A or its explicit negation $\neg A$. The set of all objective literals of a program P is called the extended Herbrand base of P and denoted by $\mathcal{H}(P)$. The symbol *not* stands for negation by default[1]. *not* L is called a default literal. Literals are either objective or default literals. By *not* $\{a_1, \ldots, a_n, \ldots\}$ we mean $\{not\ a_1, \ldots, not\ a_n, \ldots\}$.

An interpretation of an extended program P is denoted by $T \cup not\ F$, where T and F are disjoint subsets of $\mathcal{H}(P)$. Objective literals in T are said to be *true* in I, objective literals in F *false by default* in I, and in $\mathcal{H}(P) - I$ *undefined* in I.

WFSX follows from WFS for normal programs plus the coherence requirement relating the two forms of negation: *"for any objective literal L, if $\neg L$ is entailed by the semantics then not L must also be entailed"*. This requirement states that whenever some literal is explicitly false then it must be assumed false by default.

Here we present *WFSX* differently from its original definition, based on the application of the Gelfond–Lifschitz Γ operator and on a notion of semi–normal programs. The equivalence between both definitions is proven in [Alf93].

Definition 2.1 (Gelfond–Lifschitz operator [GL90]) *Let P be an extended program and S a set of objective literals. The GL–transformation of P modulo S is the program $\frac{P}{S}$ obtained from P by first substituting every explicitly negated literal $\neg A$ by a new atom $\neg\_A$ and then performing the following operations:*

- *remove from P all rules containing a default literal not A such that $A \in S$;*
- *remove from the remaining rules all default literals.*

Since $\frac{P}{S}$ is a definite program, it has a unique least model J. Let I be the set of objective literals obtained by substituting every atom $\neg\_A$ in J by $\neg A$. We define $\Gamma_P(S) = I$.

In order to impose coherence[2] we introduce:

[1] This designation has been used in the literature instead of the more operational *"negation as failure (to prove)"*.

[2] See [PAA92b] for how the use of semi–normal programs present below imposes coherence in partial stable models.

Definition 2.2 (Seminormal version of a program) *The seminormal version of a program P is the program P_s obtained from P by adding to the (possibly empty) Body of each rule $L \leftarrow Body$ the default literal not $\neg L$, where $\neg L$ is the complement of L wrt explicit negation.*

Below we use $\Gamma(S)$ to denote $\Gamma_P(S)$, and $\Gamma_s(S)$ to denote $\Gamma_{P_s}(S)$.

Definition 2.3 (Partial stable model) *An interpretation $T \cup not\ F$ is called a partial stable model of P iff $T = \Gamma\Gamma_s T$ and $F = \mathcal{H}(P) - \Gamma_s T$.*

Not every extended program has partial stable models.

Example 3 The program $P = \{a; \quad \neg a\}$ has no partial stable models.

Definition 2.4 (Contradictory program) *A program P is* WFSX*–contradictory iff it has no partial stable model.*

Theorem 2.1 (*WFSX* semantics) *Every noncontradictory program P has a least (wrt $\subseteq$) partial stable model, the well–founded model of P.*

3 Adding nonrefutable negative assumptions

In this section we show how to consistently add nonrefutable negative assumptions to the *WFSX* of an extended program P. Informally, it is consistent to add a negative assumption to *WFSX*(P) if no contradiction follows from that addition, and if the assumption atom is not among *WFSX*(P) after adding the assumption to P. A set of assumptions A is refuted (or defeated) by another set, B, if by adding B the complement of some element in A follows.

More formally, to add a negative assumption to the *WFSX* of a program means to first "compile" that assumption into the program. This is achieved simply by substituting, in all body rule occurences, *true* for any of the assumptions to be added, and *false* for their atoms.

Definition 3.1 ($P + A$) *The program $P+A$ obtained by adding to an extended program P a set of negative assumptions $A \subseteq not\ \mathcal{H}$ is the result of:*

- *Deleting from P all rules $H \leftarrow B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m$, such that some $not\ B_i \in A$;*
- *Deleting from the remaining rules all literals $not\ L \in A$.*

Then, since we are interested in adding negative assumptions to the *WFSX* of a program, after adding the assumptions to P we compute the *WFSX* of the resulting program. Moreover, there is no reason not to consider the addition of negative assumptions belonging to the *WFSX* of the original program. Note here the difference, mentioned in the introduction, between this approach and other approaches of using argumentation systems to define the semantics of programs. Whereas our approach is based on a pre–defined semantics, and uses argumentation to enlarge that semantics, others (e.g. [BTK93, Dun93]) use argumentation to defined the semantics from scratch.

Unlike the case of WFS for normal programs, by using *WFSX* of extended programs not every program has a semantics. Thus care must be taken in considering only the addition of assumptions guaranteeing the existence of *WFSX*:

Example 4 The assumption *not b* cannot be added to program P :

$$\begin{array}{rcl} \neg a & \leftarrow & not\ b \\ a & \leftarrow & not\ b \end{array} \qquad \begin{array}{rcl} b & \leftarrow & not\ b \end{array}$$

because $P + \{not\ b\} = \{\neg a \leftarrow;\ a \leftarrow;\ b \leftarrow\}$ is contradictory.

Definition 3.2 (Assumption model) *An assumption model, or A–model for short, of an extended program P with* $\mathrm{WFSX}(P) = T \cup not\ F$ *is a pair* $\langle A; M \rangle$, *where* $not\ F \subseteq A \subseteq not\ \mathcal{H}$, *where* $P + A$ *is a noncontradictory program, and*

$$M = \mathrm{WFSX}(P + A)$$

The meaning of an A–model $\langle A; \mathrm{WFSX}(P + A)\rangle$ *is* $A \cup \mathrm{WFSX}(P + A)$, *i.e. the result of adding assumptions A to the* WFSX *of P.*

As argued above, sets of negative assumptions that lead to an inconsistency are not to be considered. An inconsistency can be obtained either by having both some L and $\neg L$ in the meaning of the A–model, or some A and *not* A. By not considering sets of assumptions that when added to a program lead to a contradiction the first condition for consistency of A–models is guaranteed by definition.

Definition 3.3 (Consistent A–model) *An A–model* $\langle A; M \rangle$ *is consistent iff its meaning is consistent i.e., since by definition M is always consistent, iff there exists no assumption* $not\ L \in A$ *such that* $L \in M$.

Note that for consistency we do not impose that $not\ L \in M$. In other words our notion of consistency is not one that guarantees that arguments are stable. Instead, we just impose consistent A–models to be the result of adding to the *WFSX* of a program a set of assumptions that *cannot be contradicted by the program itself*. However one such A-model can be contradicted by another consistent A–model:

Example 5 Let $P = \{c \leftarrow not\ b;\ \ b \leftarrow not\ a;\ \ a \leftarrow not\ a\}$, whose *WFSX* is $\{not\ \neg a, not\ \neg b, not\ \neg c\}$. P has three consistent A–models:

$$\begin{array}{rcl} M_1 & = & \langle\ \ \{not\ \neg a, not\ \neg b, not\ \neg c\}\ \ ;\ \{not\ \neg a, not\ \neg b, not\ \neg c\}\ \rangle \\ M_2 & = & \langle\ \{not\ \neg a, not\ \neg b, not\ \neg c, not\ b\}\ ;\ \{c, not\ \neg a, not\ \neg b, not\ \neg c\}\ \rangle \\ M_3 & = & \langle\ \{not\ \neg a, not\ \neg b, not\ \neg c, not\ c\}\ ;\ \{not\ \neg a, not\ \neg b, not\ \neg c\}\ \rangle \end{array}$$

An agent which argues in favour of any of these sets of assumptions cannot be contradicted by an agent armed with the same program,. However $\{not\ c\}$ can be counterargued (or defeated) by $\{not\ b\}$. Accordingly we say that $\{not\ c\}$ is refutable, since there is an alternative consistent additions of assumptions that defeats it, i.e. proves its complement c.

As argued in the introduction, we are interested only in sets of assumptions that cannot be refuted. Note that, like for normal programs, also in extended programs the only way of defeating assumption $not\ L$ is by proving L. To capture the notion of non–refutability (or sustainability) we now formally define how an A–model can defeat another, and define sustainable A–model as a non–defeated consistent one.

Definition 3.4 (Defeating) *A consistent A–model $\langle A; M\rangle$ is defeated by consistent A–model $\langle A'; M'\rangle$ iff $\exists not\ a \in A | a \in M'$.*

Definition 3.5 (Sustainable A–model) *An A–model $\langle A; M\rangle$ is sustainable iff it is consistent and not defeated by any consistent A–model.*

Note that every noncontradictory program has at least one sustainable A–model. Indeed, it follows from lemma 3.2 below that the A–model whose negative assumptions are all the negative literals of $WFSX(P)$ is sustainable.

Because we wish to maximally enlarge the $WFSX$ of a program, we are especially interested in sustainable A–models that have a maximal number of added assumption, i.e. are maximal according to the partial order $\leq_a$ where:

$$\langle A_1; M_1\rangle \leq_a \langle A_2; M_2\rangle \quad \text{iff} \quad A_1 \subseteq A_2$$

Example 6 The $\leq_a$–maximal sustainable A–model of the program of example 5 is M_2. Thus, the meaning of P is $\{c, not\ b\} \cup WFSX(P)$.

However several noncompatible maximal sustainable A–models may exist:

Example 7 Let P be:

$$\begin{array}{rcl} c & \leftarrow & not\ c, not\ \neg b \\ \neg b & \leftarrow & a \\ a & \leftarrow & not\ a \end{array}$$

whose $WFSX$ is $\{not\ \neg a, not\ b, not\ \neg c\}$.

Its sustainable A–models are (where $not\ \neg a, not\ b, not\ \neg c$ are omitted for simplicity):

$$\begin{array}{l} \langle\ \{\}\ ; \{\}\ \rangle \\ \langle\ \{not\ \neg b\}\ ; \{\}\ \rangle \\ \langle\ \{not\ c\}\ ; \{\}\ \rangle \end{array}$$

the last two being maximal. Note that we cannot both add $not\ \neg b$ and $not\ c$ to P to obtain a sustainable A–model since

$$\langle \{not\ \neg b, not\ c\}; \{c\} \rangle$$

is inconsistent.

Example 8 Consider program P :

$$\begin{array}{rclcrclcrcl} \neg a & \leftarrow & not\ b & & b & \leftarrow & not\ d & & d & \leftarrow & not\ d \\ a & \leftarrow & not\ c & & c & \leftarrow & not\ d & & & & \end{array}$$

Its sustainable A–models are:

$$\begin{array}{rcl} M_1 & = & \langle \quad \{\} \quad ; \quad \{not\ \neg b, not\ \neg c, not\ \neg d\} \quad \rangle \\ M_2 & = & \langle\ \{not\ b\}\ ; \{\neg a, not\ a, not\ \neg b, not\ \neg c, not\ \neg d\}\ \rangle \\ M_3 & = & \langle\ \{not\ c\}\ ; \{a, not\ \neg a, not\ \neg b, not\ \neg c, not\ \neg d\}\ \rangle \end{array}$$

The last two are maximal. We cannot add both $not\ b$ and $not\ c$ to the program because $P + \{not\ b, not\ c\}$ is contradictory.

So, in general, sustainable A–models are not organized into a complete lattice. However:

Theorem 3.1 *The set of all sustainable A–models of a noncontradictory program is nonempty. On the basis of set union and set intersection among their A sets, the A–models ordered by* $\leq_a$ *form a lower semilattice.*

In order to define a semantics based on a unique set of additional assumptions that are sustainable and consensual, a first approach would be to consider the meet of all maximal sustainable A–models. This would already strictly enlarge the *WFSX* of some programs (cf. example 5). Moreover, it would be a generalization of *WFSX* in the sense that every literal in the *WFSX* of a program is in the meaning of that meet. Also, for every noncontradictory program that meet exist, i.e. a semantics based on that meet is defined for the same programs as *WFSX*.

Lemma 3.2 *Let* $\langle A; \mathrm{WFSX}(P+A)\rangle$ *be a consistent A–model of the extended logic program P. Then for every literal L :*

$$L \in \mathrm{WFSX}(P) \quad \Rightarrow \quad L \in \mathrm{WFSX}(P+A)$$

Theorem 3.3 *An extended logic program P has at least one consistent A–model iff P is noncontradictory. The meaning of any consistent A–model of a noncontradictory program P is a superset of* $\mathrm{WFSX}(P)$.

4 O–semantics for extended programs

To further enlarge the *WFSX* of a program, additional criteria for prefering among competing sets of sustainable assumptions must be introduced. This is achieved in the O–semantics of normal programs by means of tenability. Whereas sustainability of a consistent set of negative assumptions insists that there must be no other consistent set that defeats it (i.e. no hypothetical evidence whose consequences contradict the sustained assumptions), tenability requires additionally that a maximal sustainable set of assumptions be not contradicted by the consequences of adding to it another competing (nondefeating and nondefeated) maximal sustainable set.

In this section we introduce tenability for extended programs, and use this criterium to define a recursive peeling process that takes a lower semilattice of sustainable A–models and obtains a subsemilattice of it, by deleting maximal untenable A–models[3]. This peeling process is repeated and ends up with a complete lattice of sustainable A–models which, for every program P, is by definition its O–semantics. The meaning of P is then specified by the greatest A–model of the semantics, its O–Model.

To illustrate the problem of preference among maximal A–models recall examples 7 and 8 above.

Example 9 Because we wish to maximize the number of negative assumptions, we consider the maximal A–models, which in the case of example 7 are:

$$\begin{array}{c} \langle\ \{not\ \neg b\}\ ;\ \{\}\ \rangle \\ \langle\ \ \{not\ c\}\ \ ;\ \{\}\ \rangle \end{array}$$

The join of these maximal A–models:

$$\langle \{not\ \neg b, not\ c\}; \{c\} \rangle$$

is inconsistent wrt c. This signifies that when assuming *not* c there is an additional set of assumptions entailing c, making this A–model *untenable*. But the same does not apply to *not* $\neg b$. So we can say that *not* $\neg b$ is more primal than *not* c, in the sense that when added jointly with *not* c it causes an inconsistency in c but not one in $\neg b$. In the program that is indeed intuitively the case: *not* c depends on *not* b but not vice–versa, and so *not* b is more primal and unaffeted by *not* c. Thus the preferred A–model is

$$\langle \{not\ \neg b\}, \{\} \rangle$$

and the A–model $\langle \{not\ c\}; \{\} \rangle$ is said untenable. The rationale for the preference is grounded in that the inconsistency of the joint arises wrt c but not wrt $\neg b$.

[3]Note that by deleting maximal A–models, a greater meet is obtained.

Example 10 The maximal sustainable A–models of example 8 are both untenable because by adding the assumptions of them both to P the resulting program is contradictory. In fact:

$$P + \{not\ b, not\ c, not\ \neg b, not\ \neg c, not\ \neg d\} = \quad \begin{array}{rcl} \neg a & & \\ a & & \\ b & \leftarrow & not\ d \\ c & \leftarrow & not\ d \\ d & \leftarrow & not\ d \end{array}$$

The rationale for saying that both maximal sustainable A–models are untenable is grounded in that there is no reason to prefer one over the other, and they cannot be both assumed jointly. Thus in the end none of them is tenable.

Definition 4.1 (Candidate structure) *A Candidate Structure CS of a program P is any subsemilattice of the lower semilattice of all sustainable A–models of P.*

Definition 4.2 (Untenable A–models) *Let $\{\langle A_k; M_k\rangle \mid k \in K\}$ for $k \in K$, be the set of all maximal A–models in Candidate Structure CS, and let:*

$$P_J = P + \bigcup_{k \in K} A_j$$

An A–model $\langle A_i; M_i\rangle$ is untenable wrt CS iff it is maximal in CS and either:

- *P_J is contradictory, or*
- *there exists $not\ a \in A_i$ such that $a \in \mathrm{WFSX}(P_J)$.*

The candidate structure left after removing all untenable A–models of a CS may itself have several untenable elements, some of which might not be untenable A–models in the initial CS. If the removal of untenable A–models is performed repeatedly on the retained Candidate Structure, a structure with no untenable models is eventually obtained, albeit the bottom element of the candidate structure. If this structure has a single maximal element then it denotes the O–semantics. Otherwise the lemma below guarantees that the join of maximal tenable A–models can be safely added to the structure.

Lemma 4.1 *Let $\{\langle A_k; M_k\rangle \mid k \in K\}$ for $k \in K$, be the set of all maximal A–models in Candidate Structure CS. If none of those A–models is untenable then*

$$\langle \bigcup_{k \in K} A_k; \mathrm{WFSX}(P + \bigcup_{k \in K} A_k)\rangle$$

is a sustainable A–model.

Definition 4.3 (Retained CS) *Let CS be a candidate structure, and let A be the union of all assumptions in all A–models of CS. The Retained Candidate Structure $R(CS)$ of CS is defined recursively in the following way:*

- $\langle A; \mathrm{WFSX}(P+A)\rangle \cup CS$ *if there are no untenable A–models in CS;*
- *Otherwise, let Unt be the set of all untenable A–models wrt CS. Then*

$$R(CS) = R(CS - Unt).$$

Definition 4.4 (The O–semantics of extended programs) *The O–semantics of an extended logic program P is the retained candidate structure of the semilattice of all sustainable A–models of P.*

Let $AM = \langle A; M\rangle$ be its maximal element. The O–model of P is the meaning of AM, i.e. $A \cup M$.

The theorems below go to show that in fact the O–semantics defined herein for extended programs is a generalization of that for normal programs, and that it enlarges the *WFSX*, by leaving less literals undefined.

Theorem 4.2 (Generalization of the O–semantics) *Let P be a normal logic program. The "O–semantics" of P (as defined in [PAA92a, PAA93]) coincides with the "O–semantics for extended programs" of P.*

Theorem 4.3 (Existence of the O–semantics) *An extended logic program P is* WFSX*–noncontradictory iff it has O–semantics.*

Theorem 4.4 (Extension of *WFSX*) *Let P be a noncontradictory extended logic program, whose* WFSX*well–founded model is M, and whose O–model is O. Then $M \subseteq O$.*

5 Comparisons

Although several semantics that extend WFS of normal programs by producing less undefinedness exist, to the best of our knowledge, apart from the "improved semantics for extended programs" of [BTK93], no other attempt has been made to combine these extension of WFS with explicit negation. So, in this section we not only compare our approach with the approach of [BTK93], but also compare the application of O–semantics for normal programs with other approaches that extend WFS for those programs.

One semantics extending WFS of normal programs is the extended well-founded semantics (EWFS) [Dix91]. Roughly, EWFM moves closer than the WFM (in the sense of being less undefined) to being the intersection of all minimal Herbrand models of P. With a different notation from that of [Dix91]:

$$EWFM(P) =_{def} WFM(P) \cup T(WFM(P)) \cup not\ F(WFM(P))$$

where:

$$T(\mathcal{I}) =_{def} True(MIN\_MOD(\mathcal{I}, P)),$$
$$F(\mathcal{I}) =_{def} False(MIN\_MOD(\mathcal{I}, P))$$

$\mathcal{I}$ is a three–valued interpretation, and $MIN\_MOD(\mathcal{I}, P)$ is the collection of all minimal two–valued Herbrand models of P consistent with $\mathcal{I}$. For a set $\mathcal{S}$ of interpretations, $True(\mathcal{S})$ (resp. $False(\mathcal{S})$) denotes the set of all atoms which are true (resp. false) in all interpretations of $\mathcal{S}$.

For the program $P = \{a \leftarrow not\ a\}$ we have:

$$WFM(P) = \{\}, MIN\_MOD(\{\}, P) = \{\{a\}\} \quad EWFM(P) = \{a\}$$

The O–Model of P is empty.

The main differences between ours and Dix's approach are:

- Like WFS and unlike EWFM, we insist on the supportedness of positive literals, i.e.:

  $$\text{An atom } A \in M_P \text{ iff } \exists\ A \leftarrow Body \ \mid\ Body \subseteq M_P$$

- Unlike WFS and unlike EWFM, we relax, by allowing undefined bodies with false heads under certain conditions, the requirement of supportedness of negative literals, i.e. we relax:

  $$not\ A \in M_P \text{ iff } \forall\ A \leftarrow Body \ \mid\ Body \text{ is false in } M_P$$

In other words, unlike Dix's approach, we impose that positive literals be *derived* from program rules plus the assumption of some negative literals. Negative literals are assumed if that can be safely done (according to the argumentation principles we've stated).

Example 11 Let P be:

$$\begin{array}{lcl} c & \leftarrow & not\ b \\ b & \leftarrow & not\ a \\ a & \leftarrow & not\ a \end{array}$$

The O–Model of P is $\{c, not\ b\}$. Note that c has a rule whose body is true in the O–Model and it is not the case that all rules for b are false in it. However $not\ b$ can be safely assumed.

The *EWFM* is $\{a, c, not\ b\}$. The atom a is true in the *EWFM* and has no rule with a body true in it. All rules for b have a false body in the *EWFM*.

Another semantics also extending the WFS of normal programs is the "stable theories semantics" [KM91]. Like O–semantics, stable theories also enlarge the WFS by adding more negative assumptions. In order to avoid some unintuitive results of stable theories, Kakas and Mancarella (personal communication) modified their definition, and presented the 'acceptability semantics" . In this recent work, instead of our notion of "sustainable" they present:

A is **KM–coherent**[4] if all assumptions B that defeat A are defeated by A.

i.e., if one insists on the set of assumptions A, no consistent nondefeatable evidence to the contrary can be found. No preferred unique model is identified in their approach, the semantics being defined as the set of consequences common to all KM–coherent sets of assumptions.

However, their approach is still conservative (as noted in [BTK93]). For example consider the program:

$$\begin{array}{rcl} a & \leftarrow & not\ b \\ b & \leftarrow & not\ a, not\ b \end{array}$$

Its consistent A–models are:

$$\begin{array}{lcccr} \langle & \{\} & ; & \{\} & \rangle \\ \langle & \{not\ a\} & ; & \{\} & \rangle \\ \langle & \{not\ b\} & ; & \{a, not\ b\} & \rangle \end{array}$$

Given that the second A–model is defeated by the third one, the O–model is $\{a, not\ b\}$, i.e. *not* b is added.

Using the acceptability semantics approach, both $\{not\ a\}$ and $\{not\ b\}$ are KM–coherent, and so the semantics is $\{\}$.

More recently, in [BTK93], a flexible framework for defining logic programming semantics is presented, based on an argumentation framework similar to the one in [Dun91] but where a different notion of evidence to the contrary, and of how a scenario defeats contrary evidence, are used.

Within this framework the authors can define the well–founded, stable models, preferred extensions, and acceptability semantics. Then, motivated by the very same example presented here that shows the acceptability semantics is too conservative, they define what they call an "improved semantics".

This "improved semantics" views program rules as inference rules, and default literals as new atoms that are abducible. The semantics is defined as the result of adding to the program some sets of abducible atoms, called preferred extensions. A preferred extension must be consistent with the program (i.e. for no A can A and *not* A hold), and must *counterattack* every *attack* made on it. A set of abducibles A *attacks* another B iff $P \cup A$ derives some L such that *not* $L \in B$, and B *counterattacks* A iff it attacks A or A itself is inconsistent.

As shown by the example below, the "improved semantics" exhibits some "strange" behaviour:

[4] The authors just call it coherent. Here we use KM–coherent to avoid confusion with our coherence principle, which is completely unrelated.

Example 12 Consider a program with the single rule:

$$a \leftarrow not\ a, not\ b$$

whose WFS and O–model is $\{not\ b\}$. Note that this is indeed the expected result: since b has no rules, *not* b must belong to the semantics independently of the truth value of a. Thus, from the standpoint of *not* a this program should be equivalent to the one containing the single rule $a \leftarrow not\ a$, and so *not* a cannot be safely added.

However, according to the "improved semantics" $\{not\ a\}$ is a preferred extension of this program since $\{not\ a, not\ b\}$ is the only attack on $\{not\ a\}$, but $P \cup \{not\ a, not\ b\}$, is inconsistent.

Accordingly, there are two preferred extensions $P \cup \{not\ a\}$ and $P \cup \{not\ b\}$, so that neither *not* a nor *not* b follow.

This example shows that the "improved semantics" does not always enlarge WFS. In this case the former sustains more undefinedness than the latter.

In [BTK93] the authors also generalize their "improved semantics" for extended programs, by renaming every explicitly negated literal $\neg A$ to a new atom, say $\neg\_A$, and by introducing inference rules stating that A and $\neg\_A$ derive an inconsistency. In our opinion this approach not only suffers from the problems inherited from the "improved sermantics" of normal programs, but also some new ones due to the way explicit negation is introduced. In particular, their semantics does not in general comply with the coherence requirement of section 2:

Example 13 The program $P = \{\neg a;\ \ a \leftarrow not\ a\}$ has one preferred extension $P \cup \{\}$. Thus according to this semantics the consequences of P are $\{\neg a\}$, so that a is explicitly false and cannot be assumed false by default.

In our opinion this program should be contradictory: a is explicitly declared as false; thus a must be assumed false by default, i.e. *not* a should hold; *not* a derives a so contradiction ensues. This is what O–semantics proclaims.

References

[Alf93] José Júlio Alferes. *Semantics of Logic Programs with Explicit Negation.* PhD thesis, Universidade Nova de Lisboa, October 1993.

[AP92] J. J. Alferes and L. M. Pereira. On logic program semantics with two kinds of negation. In K. Apt, editor, *Int. Joint Conf. and Symp. on LP*, pages 574–588. MIT Press, 1992.

[BTK93] A. Bondarenko, F. Toni, and R. Kowalski. An assumption–based framework for nonmonotonic reasoning. In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. on LP & NMR*, pages 171–189. MIT Press, 1993.

[Dix91] J. Dix. Classifying semantics of logic programs. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LP & NMR*, pages 166–180. MIT Press, 1991.

[Dix92] J. Dix. A framework for representing and characterizing semantics of logic programs. In B. Nebel, C. Rich, and W. Swartout, editors, *3rd Int. Conf. on Principles of Knowledge Representation and Reasoning.* Morgan Kaufmann, 1992.

[DR91] P. M. Dung and P. Ruamviboonsuk. Well founded reasoning with classical negation. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LP & NMR*, pages 120–132. MIT Press, 1991.

[Dun91] P. M. Dung. Negation as hypotheses: An abductive framework for logic programming. In K. Furukawa, editor, *8th Int. Conf. on LP*, pages 3–17. MIT Press, 1991.

[Dun93] P. M. Dung. An argumentation semantics for logic programming with explicit negation. In D. S. Warren, editor, *10th Int. Conf. on LP*, pages 616–630. MIT Press, 1993.

[GL90] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th Int. Conf. on LP*, pages 579–597. MIT Press, 1990.

[GRS91] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[KM91] A. C. Kakas and P. Mancarella. Stable theories for logic programs. In Ueda and Saraswat, editors, *Int. LP Symp.*, pages 85–100. MIT Press, 1991.

[KS90] R. Kowalski and F. Sadri. Logic programs with exceptions. In Warren and Szeredi, editors, *7th Int. Conf. on LP.* MIT Press, 1990.

[PA92] L. M. Pereira and J. J. Alferes. Well founded semantics for logic programs with explicit negation. In B. Neumann, editor, *European Conf. on AI*, pages 102–106. John Wiley & Sons, 1992.

[PAA91] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Nonmonotonic reasoning with well founded semantics. In Koichi Furukawa, editor, *8th Int. Conf. on LP*, pages 475–489. MIT Press, 1991.

[PAA92a] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Adding closed world assumptions to well founded semantics. In *Fifth Generation Computer Systems*, pages 562–569. ICOT, 1992.

[PAA92b] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Default theory for well founded semantics with explicit negation. In D. Pearce and G. Wagner, editors, *Logics in AI. Proceedings of the European Ws. JELIA'92*, pages 339–356. LNAI 633, Springer–Verlag, 1992.

[PAA93] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Adding closed world assumptions to well founded semantics (extended improved version). *Theoretical Computer Science. Special issue on selected papers from FGCS'92*, 1993.

[Prz90] T. Przymusinski. Extended stable semantics for normal and disjunctive programs. In Warren and Szeredi, editors, *7th Int. Conf. on LP*, pages 459–477. MIT Press, 1990.

[Prz91] T. Przymusinski. A semantics for disjunctive logic programs. In Loveland, Lobo, and Rajasekar, editors, *ILPS'91 Ws. in Disjunctive Logic Programs*, 1991.

[Wag91] G. Wagner. A database needs two kinds of negation. In B. Thalheim, J. Demetrovics, and H-D. Gerhardt, editors, *Mathematical Foundations of Database Systems*, pages 357–371. LNCS 495, Springer–Verlag, 1991.