A Practical Introduction to Well Founded Semantics

Luís Moniz Pereira José Júlio Alferes
Joaquim Nunes Aparício *

CrIA, Uninova and DCS, Universidade Nova de Lisboa
2825 Monte da Caparica, Portugal

March 25, 1991

Abstract

The Well Founded Semantics appears to be a very suitable semantics for logic programming and nonmonotonic reasoning. Various formal definitions of these semantics have been proposed lately.

The purpose of this paper is to familiarize the newcomer with the Well Founded and the 3-valued (or extended) stable model semantics, with the help of some examples, providing some intuitive and practical ideas for easier and faster calculation of the well founded model and the other extended stable models of a program.

1 Introduction

The Well Founded Semantics (WFS), introduced by [14], is a 3-valued semantics which extends previous logic program semantics to the class of all normal programs.

It extends Perfect Model Semantics [13], from the class of stratified logic programs to the class of all normal programs, avoiding various drawbacks of other proposed approaches, namely the positive and negative recursion problems.

WFS have been proved equivalent to a natural extension to 3 values of the Stable Model Semantics of [3], i.e. the 3-valued (or extended) Stable Model Semantics (XSMS) [11]. The WF Model of a program P is the smallest XSM of P. Recently, WFS have also encompassed programs with classical negation and disjunctive conclusions [11][12][4].

The fact that WFS is well defined for any normal program is very important. A logic program may contain predicates whose truth or falsity is not fully determined by it, thus being undefined, in addition to predicates whose truth and falsity is completely determined by the program.

To illustrate this point, consider the following program [11]:

$$\begin{array}{rcl@{\qquad}rcl} work & \leftarrow & \sim tired. & tired & \leftarrow & \sim sleep. \\ sleep & \leftarrow & \sim work. & paid & \leftarrow & . \\ angry & \leftarrow & work, \sim paid. & & & \end{array}$$

It appears that the first three rules describe only mutual relationships between the propositions tired, work and sleep, without providing sufficient information to determine their truth or falsity. Depending on the point of view, we could describe our knowledge about these propositions as incomplete or perhaps even confusing. On the other hand, regardless of the others' status, the proposition paid must be true and thus angry, via negation as failure, must be false.

*Fax: +351 1 295 5641 Ph: +351 1 295 3156 E-mail: {lmp,jja,jna}@fct.unl.pt Tlx:14542 FCTUNL P

This leads to the unique $XSM = < \{paid\}; \{angry\} >$ of the program, in which *paid* is true, *angry* is false, and *tired*, *sleep* and *work* are undefined (and by convention not mentioned in the XSM). If we later learn, say, that *work* is in fact true, then we will conclude that *sleep* is false and *tired* is true; but our beliefs about *paid* and *angry* will remain unchanged.

Extended Stable Model Semantics also seems an adequate semantics for non-monotonic reasoning, as claimed in [8] and [7]. In fact, as our knowledge about the world is almost always incomplete, we need the ability to describe possible worlds (models of our knowledge) in which some facts are neither true nor false, and worlds where such facts may (or should) be believed or unbelieved. Moreover the WFS always provides a minimal model (the WF Model) which can be seen as our "hard" view of the world, where we only know the truth or falsity of what we take for sure.

WFS has a realistic computational based proof theory, and can be naturally implemented in sound and complete inference machines. SLS-resolution [10] provides one such mechanism for WFS. The Extended Warren Abstract Machine (XWAM) [15] also provides a procedural semantics for WFS for a normal program. Recently, in [5][6], a Prolog top-down procedure for WFS was presented.

The purpose of this paper is to familiarize the newcomer with the WFS, providing some intuitive and practical ideas for easier and faster calculation of the WF Model and the other XSMs of a program. It is organized as follows: In Section 2 a brief formal definition of WFS is presented. Section 3 presents the calculation of the WF Model for some examples, and provides some of its properties that can help to make it easier and faster to calculate. In section 4 the XSMs are formally characterized. Finally, in Section 5 we present intuitive ideas for building such models.

2 Formal Definition of the WFS

As the semantics are 3-valued, we will begin by defining 3-valued interpretations.

Definition 2.1 *3-valued interpretation By a 3-valued Herbrand interpretation I of a language $\mathcal{L}$ we mean any pair $< T; F >$, where T and F are disjoint subsets of the Herbrand base $\mathcal{H}$. The set T contains all ground atoms true in I, the set F contains all ground atoms false in I and the truth value of the remaining atoms, in $U = \mathcal{H} - (T \cup F)$, is unknown (or undefined).*

Proposition 2.1 *Any interpretation $< T; F >$ can be equivalently viewed as a function $I : \mathcal{H} \to \mathcal{V}$ where $\mathcal{V} = \{0, 1/2, 1\}$, defined by:*

- $I(A) = 0$ *if* $A \in F$
- $I(A) = 1/2$ *if* $A \in U$
- $I(A) = 1$ *if* $A \in T$

Definition 2.2 *Truth valuation*

The function $I : \mathcal{H} \to \mathcal{V}$ can be recursively extended to the truth valuation function $\hat{I} : \mathcal{LIT} \to \mathcal{V}$ defined on the set $\mathcal{LIT}$ of all literals of the language.

Given an interpretation $I : \mathcal{H} \to \mathcal{V}$ then the truth valuation $\hat{I}$ corresponding to I is a function $\hat{I} : \mathcal{LIT} \to \mathcal{V}$, as follows, where A is a ground atom:

- $\hat{I}(A) = I(A)$
- $\hat{I}(\sim A) = 1 - I(A)$

Using these definitions let us now define a useful operator for constructing the WF Model of a normal logic program.

Definition 2.3 *The Θ operator([10])*

Suppose that P is a logic program and E is an interpretation. The operator $\Theta_E : \mathcal{I} \rightarrow I$ on the set $\mathcal{I}$ of all 3-valued interpretations of P is defined as follows.

If $I \in \mathcal{I}$ is an interpretation of P and A is a ground atom then $\Theta_E(I)$ is the interpretation defined by:

- $\Theta_E(I)(A) = 1$ *iff there is a rule $A \leftarrow L_1, \ldots, L_n$ in P such that for all $i \leq n$ either $\hat{E}(L_i) = 1$, or L_i is positive and $I(L_i) = 1$;*
- $\Theta_E(I)(A) = 0$ *iff for every rule $A \leftarrow L_1, \ldots, L_n$ in P there is an $i \leq n$, such that either $\hat{E}(L_i) = 0$, or L_i is positive and $I(L_i) = 0$;*
- $\Theta_E(I)(A) = 1/2$ *otherwise.*

Intuitively, the interpretation E represents facts currently known to be true or false. The true atoms in $\Theta_E(I)$ consist of those atoms which can be derived in one step from the program P assumin all the truth values of atoms in E and all the true atoms in I. The false atoms in $\Theta_E(I)$ consist of those atoms whose falsity can be deduced in one step (using the Closed World Assumption) from the program P, assuming the truth value of all atoms in E and the falsity of all false atoms in I. Note that when there are no rules for A, A is false in $\Theta_E(I)$. All other atoms in I are undefined.

Definition 2.4 *Least Model*

If I and J are two interpretations then we say that $I \leq J$ if $I(A) \leq J(A)$ for any ground atom A. If $\mathcal{I}$ is a collection of interpretations, then an interpretation $I \in \mathcal{I}$ is called minimal in $\mathcal{I}$ if there is no interpretation $J \in \mathcal{I}$ such that $J \leq I$ and $I \neq J$. An interpretation I is called least in $\mathcal{I}$ if $I \leq J$, for any other interpretation $J \in \mathcal{I}$. A model of a theory R is called minimal (resp. least) if it is minimal (resp. least) among all models of R.

Theorem 2.1 *([10]) For every interpretation E, the operator Θ_E is monotone and it has a unique least fixed point given by $\Theta_E^{\uparrow\omega}$.*

By definition $\Theta_E^{\uparrow n} = \Theta_E(\Theta_E^{\uparrow n-1})$ and $\Theta_E^{\uparrow 0} = < \{\}; \mathcal{H} >$, where $\mathcal{H}$ is the Herbrand base of the program. ω is the smallest ordinal, as large as necessary such that $\Theta_E^{\uparrow\omega} = \Theta_E(\Theta_E^{\uparrow\omega})$. The definition of $\Theta_E^{\uparrow 0}$ establishes a preference for the Closed World Assumption.

The least fixed point of Θ_E is also denoted $\Omega(E)$, i.e. $\Omega(E) = \Theta_E^{\uparrow\omega}$.

Clearly, Ω constitutes an operator on the set of all interpretations of P. This operator in turn has a F-least fixed point [1] as proved in [10], where the F-least ordering is defined as follows:

Definition 2.5 *F-least model([1])*

If $I = < T; F >$ and $I' = < T'; F' >$ are two interpretations, then we say that $I \leq_F I'$ iff $T \subseteq T'$ and $F \subseteq F'$. An interpretation I is called F-least in a collection of interpretations $\mathcal{I}$ if $I \leq_F J$ for any interpretation $J \in \mathcal{I}$

Finally, let us define the Well Founded Model of a normal program.

Definition 2.6 *Well Founded Model([10])*

We call the unique F-least fixed point M_P of Ω the well founded model of P.

This definition of the WF Model was proved equivalent to the original one [14] in [9].

In order to obtain a constructive definition of the well founded model M_P of a given program P [9] defined the following sequence $\{I_\alpha\}$ of interpretations of P:

- $I_0 = < \{\}; \{\} >$.
- $I_{\alpha+1} = \Omega(I_\alpha) = \Theta_{I_\alpha}^{\uparrow\omega}$.

The WF Model of P is the F-least fixed point of this sequence, i.e. I_λ.

3 Calculating the Well Founded Model

In this section we will give some ideas and present some properties of the WF Model, in order to help the calculation of this model. These properties are only presented informally, and not proved in this paper. Throughout the paper such properties will be remarked.

With this purpose in mind, we start by calculating the WFM according to the formal definition above for some examples.

Example 1 Let P be the program:

$$(1)\ \ p \leftarrow a. \qquad (2)\ \ a \leftarrow . \qquad (3)\ \ q \leftarrow a, {\sim}b.$$

The construction of the WFM is as follows: We start with the interpretation $I_0 = <\{\};\{\}>$, and begin by calculating $I_1 = \Omega(I_0) = \Theta^{\uparrow\omega}_{<\{\};\{\}>}$.

In order to build $\Theta^{\uparrow\omega}_{<\{\};\{\}>}$, by definition we need $<\{\};\mathcal{H}>$, which in this case is $<\{\};\{p,q,a,b\}>$. So $\Theta^{\uparrow 1}_{<\{\};\{\}>} = \Theta_{<\{\};\{\}>}(<\{\};\{p,q,a,b\}>)$.

Let us now calculate carefully this interpretation.

- $\Theta_{<\{\};\{\}>}(<\{\};\{p,q,a,b\}>)(p) = 0$ because the only rule for p is (1) and for this rule a is assumed false by default, i.e. $(<\{\};\{p,q,a,b\}>)(a) = 0$.
- $\Theta_{<\{\};\{\}>}(<\{\};\{p,q,a,b\}>)(q) = 0$ because the only rule for q is (3) and for this rule a assumed false by default, i.e. $(<\{\};\{p,q,a,b\}>)(a) = 0$.
- $\Theta_{<\{\};\{\}>}(<\{\};\{p,q,a,b\}>)(a) = 1$ because there is fact (2) for a.
- $\Theta_{<\{\};\{\}>}(<\{\};\{p,q,a,b\}>)(b) = 0$ because there are no rules for b.

So $\Theta_{<\{\};\{\}>}(<\{\};\{p,q,a,b\}>) = <\{a\};\{p,q,b\}>$.

Note that for every program P, $\Theta^{\uparrow 1}_{<\{\};\{\}>}(AF) = 1$ for every atom AF that has a fact in P and $\Theta^{\uparrow 1}_{<\{\};\{\}>}(An) = 0$ for every atom AN that has no rules in P.

Advice 3.1 *In fact we can begin the calculation of the WF Model by $I_0 = <Facts; NoRules>$ where $Facts$ is the set of all atoms that are a fact in P and $NoRules$ the set of all atoms that have no rules for them in P.*

Note that p and q are false only because a and b are initially assumed to be false; if we had started with $<\{a\};\{b\}>$ rather than $<\{\};\{\}>$ we would have obtained immediately, after one iteration $I_2 = <\{p,q,a\};\{b\}>$ below.

Let us continue now with the example. So: $\Theta^{\uparrow 2}_{<\{\};\{\}>} = \Theta_{<\{\};\{\}>}(<\{a\};\{p,q,b\}>)$. Now:

- $\Theta_{<\{\};\{\}>}(<\{a\};\{p,q,b\}>)(p) = 1$ because the only rule for p is (1) and for this rule a is known to be true from the first iteration.
- $\Theta_{<\{\};\{\}>}(<\{a\};\{p,q,b\}>)(q) = 1/2$ because the only rule for q is (3), and by it we can't prove its value is 1 because, despite the fact that b is false and a is true, ${\sim}b$ is a negative literal; and neither can we prove its value is 0, because a is true in $<\{a\};\{p,q,b\}>$ and b is not false in $<\{\};\{\}>$.
- $\Theta_{<\{\};\{\}>}(<\{a\};\{p,q,b\}>)(a) = 1$ and $\Theta_{<\{\};\{\}>}(<\{a\};\{p,q,b\}>)(b) = 0$ for the same reasons above.

So $\Theta_{<\{\};\{\}>}(<\{a\};\{p,q,b\}>) = <\{p,a\};\{b\}>$. It can be easily calculated that $\Theta^{\uparrow 3}_{<\{\};\{\}>} = \Theta^{\uparrow 2}_{<\{\};\{\}>}$, and this is left as an exercise to the reader.

Thus $I_1 = <\{p,a\};\{b\}>$.

Despite the fact that both p and q should intuitively be true, in the first iteration of ω operator we only get p true. This is due to the fact that in the definition of $\Theta_E(I)$, I is only tested for positive literals, thus delaying the evaluation of negative ones for next iteration where the final I becomes the new E, and now negative literals are present.

So let us continue the calculation by building $I_2 = \Omega(I_1) = \Theta^{\uparrow\omega}_{I_1}$. Here some steps will be omitted for brevity.

- For atoms p, a and b their truth value will remain unchanged (with respect to I_1) in $\Theta_{<\{p,a\};\{b\}>}(<\{\};\{p,q,a,b\}>)$.
- Since $\hat{I}_1(a) = 1$ and $\hat{I}_1(\sim b) = 1 - I_1(b) = 1$, consequently by rule (3) $\Theta_{<\{p,a\};\{b\}>}(<\{\};\{p,q,a,b\}>)(q) = 1$.

So $\Theta^{\uparrow 1}_{<\{p,a\};\{b\}>} = <\{p,q,a\};\{b\}>$ and the reader can easily see that $\Theta^{\uparrow 2}_{<\{p,a\};\{b\}>} = \Theta^{\uparrow 1}_{<\{p,a\};\{b\}>}$, and thus: $I_2 = <\{p,q,a\};\{b\}>$. It is also left to the reader to show that $I_2 = I_3$, and accordingly conclude that the WF Model of P is $<\{p,q,a\};\{b\}>$.

This result is the result a Prolog interpreter would give for this program[1].

Advice 3.2 *This can be generalized by saying that for every program without positive nor negative recursion the WFM coincides with the results a Prolog interpreter would give for such programs.*

□

Example 2 In order to get some more properties useful for building the WF Model, let us consider another program:

(1)	p	$\leftarrow$	$\sim q.$	(4)	a	$\leftarrow$	$\sim b.$	(7)	e	$\leftarrow$	.
(2)	p	$\leftarrow$	$a.$	(5)	b	$\leftarrow$	$\sim a.$	(8)	d	$\leftarrow$	$d.$
(3)	r	$\leftarrow$	$p, r.$	(6)	c	$\leftarrow$	$e.$				

Using one of the properties presented in the last example let us begin the calculation of the WFM with $I_0 = <\{e\};\{q\}>$.

In the first iteration of Ω operator we would get p true by rule (1) because $\sim q$ is true in I_0; c is true by rule (6) because e is true. For the other atoms (r,a,b and d), let us calculate more carefully their values in I_1.

• For atom r its value will be 0 because we start the calculus of Ω with $<\{\};\mathcal{H}>$ and as r **depends positively** (by a positive chain, i.e. with no interposing $\sim$) on itself for all rules defined for it (in this case rule (3)), we will always get a zero valued literal in the body of its rules, so that its interpretation value is always 0.

Advice 3.3 *Generalizing this result we can say that for any atom A that depends positively on itself for all rules defining it, its value in the WFM will be false.*

The dependency relation is easily found from the dependency graph.

Intuitively, the initial interpretation $<\{\};\mathcal{H}>$ establishes a CWA preference valuation on atoms that depend positively on themselves through all their rules (when there are no rules this is valid vacuously). That is, no positive amount of truth can be directly self-supportive.

[1]The answer to **notA** provides the result regarding $\sim$**A**. Correct results are only ensured when all **notA** calls are ground; since programs are assumed ground in this exposition the condition is satisfied

• Since we start by having d as undefined (in I_0) and as d **depends negatively** (by a negative chain, i.e. with one or more interposing $\sim$) on itself for every rule defined for it, this atom will remain undefined in every iteration. Let us look closely at the first iteration:

$\Theta_{I_0}(<\{\};\mathcal{H}>)(d) = 1/2$ because it is neither the case that $\hat{I}_0(\sim d) = 1$ nor that $\hat{I}_0(\sim d) = 0$.

This calculation is obviously valid in further iterations. The same result applies to atoms a and b. Intuitively, if there is a negative dependence of an atom on itself through an odd number of interposing $\sim$ a zero valuation of the atom would lead to a contradiction of value. If all dependencies are negative and odd, the 1-valuation is also contradictory. If some dependency is negative and even, and there is no negative odd dependency, all 3 valuations are possible; if there is also a negative odd dependency only 2 valuations are possible. Given that there is more then one possibility, the WFM prefers undefinition, i.e. the 1/2-valuation. This is why the F-order is used to choose the F-least fixed point of Ω. From all the interpretations that satisfy the CWA in the sense above, the one providing least information (i.e. preference) is adopted. As we will see the choices different from 1/2 are considered in the other XS Models.

So the WF Model of this program is $<p, c, e; q, r>$.

Note that without rule (1)p would be undefined in the WFM. But as in the definition of Θ operator, for obtaining value 1, it is said "$if there is a rule \ldots$" , and there is rule (1), the value 1 overrides the undefined value as it would override the false one if such were the case.

Advice 3.4 *Generalizing this result, we can say that the truth value of an atom A with rules $CA_1, \ldots, CA_n$ in the WFM is the greatest of the truth values for it considering each of the rules alone (i.e. without any of the others present, but keeping the rest of the program).*

Advice 3.5 *Another useful property that can be seen intuitively from this example is that disjoint*[2] *parts of the program can be calculated separately and then the WFM of the program is the union of the WFM of all parts.*

This can be easily seen by dividing the above program into 3 disjoint parts: one with rules from (1) to (5), giving the $WFM_1 = <\{p\};\{q, r\}>$; another with rules (6) and (7) with $WFM_2 = <\{c, e\};\{\}>$; and finally another one with rule (8) giving $WFM_3 = <\{\};\{\}>$. The final WFM is $<p \cup c, e \cup \{\}; q, r \cup \{\} \cup \{\}> = <p, c, e; q, r>$ as expected. □

Advice 3.6 *Summarizing and combining the properties presented above, here are some hints for building the WF Model in a faster and easier way:*

1. *Divide the program into its disjoint parts. Calculate the WFM of each of its parts as if they were separate programs and then make the union of the results to get the WFM of the original program.*

2. *If a program has no positive or negative recursion its WFM is built from the results a Prolog interpreter would give for each of its atoms.*

3. *Otherwise proceed by calculating the value of each atom A following the rules:*

- **3.1** *If A is a fact in the program then the truth value of A is 1.*
- **3.2** *If A appears in the program but has no rules defined for it, then its truth value is 0.*
- **3.3** *If A depends positively on itself via all rules defined for it, then its truth value is 0.*
- **3.4** *If A depends negatively on itself via all rules defined for it, then its truth value is 1/2.*
- **3.5** *If the only rule for A has the values of all the literals of its body already defined, choose for the value of A the smallest of them.*

[2] By disjoint parts of a program we mean parts of it that has disjoint sets of atoms

3.6 *If there is more than one rule for A, then consider all programs obtained by removing all rules for A except one. For each such programs calculate the value of A in each of them. The value of A in the original program is the greatest of the values obtained for it.*

3.7 *Otherwise calculate the value of A with the formal definition.*

These steps need not be followed rigidly in sequence.

Using these hints let us calculate now the WFM for some programs.

Example 3

$$\begin{array}{llllllllllllll} (1) & p & \leftarrow & \sim q. & (3) & r & \leftarrow & \sim t. & (5) & a & \leftarrow & \sim b. & (7) & e \leftarrow d. \\ (2) & p & \leftarrow & q, p. & (4) & t & \leftarrow & \sim r. & (6) & b & \leftarrow & e. & & \end{array}$$

This program can be divided into three disjoint parts, the first with rules (1) and (2); the second with rules (3) and (4); and the third with the remaining rules.

In the first part the value for q is false because there are no rules defined for it. The value for p considering just rule (1) is true because q is false; considering just rule (2), its value should be false by 3.3. So by 3.6 the value of p is true. Thus the WF Model of the first part is $WFM_1 = < \{p\}; \{q\} >$. In the second part, by 3.4, the truth value of both r and t is undefined ($WFM_2 = < \{\}; \{\} >$). Finally the third part has no negative nor positive recursion and so, by 2., $WFM3 = < \{b, e\}; \{a, d\} >$. Thus the WF Model of the original program is $< \{p, b, e\}; \{q, a, d\} >$. □

Example 4 To calculate the WFM of[3]:

$$\begin{array}{llllll} (1) & flies(X) & \leftarrow & bird(X), \sim\neg flies(X). & (4) & bird(a) \leftarrow . \\ (2) & \neg flies(X) & \leftarrow & penguin(X) \sim flies(X). & (5) & penguin(b) \leftarrow . \\ (3) & bird(X) & \leftarrow & penguin(X) & & \end{array}$$

we must consider first its ground version, which is:

$$\begin{array}{llllllllll} (1a) & flies(a) & \leftarrow & bird(a), \sim\neg flies(a). & (1b) & flies(b) & \leftarrow & bird(b), \sim\neg flies(b). \\ (2a) & \neg flies(a) & \leftarrow & penguin(a), \sim flies(a). & (2b) & \neg flies(b) & \leftarrow & penguin(b), \sim flies(b). \\ (3a) & bird(a) & \leftarrow & penguin(a). & (3b) & bird(b) & \leftarrow & penguin(b). \\ (4) & bird(a) & \leftarrow & . & (5) & penguin(b) & \leftarrow & . \end{array}$$

Clearly each of the columns constitutes a disjoint part of the program.

In the first part, as there are no rules defined for $penguin(a)$, this atom is false and so $\neg flies(a)$ is also false. Because of (4), $bird(a)$ is true and, as $\neg flies(a)$ is false, flies(a) is true.

In the second part, $penguin(b)$ is true (because of (5)), and so $bird(b)$ is also true. Given that, with these valuations, $flies(b)$ always depends negatively on itself, its value is undefined. The same applies to $\neg flies(b)$. So the WFM is:

$< \{flies(a), bird(a), bird(b), penguin(b)\}; \{penguin(a), \neg flies(a)\} >$. □

4 The Extended Stable Model Semantics

In this section we will characterize the Extended Stable Model Semantics, as in [9], which are an extension to the Stable Model Semantics (SMS) of [3].

[3]We consider the classical negation of an atom A as being a new atom. So, in this example to have $\neg flies(X)$ is the same as renaming it to, say, $no\_flies(X)$

Definition 4.1 *Non-negative program*

By a non-negative program we mean a logic program whose premises are either positive atoms or the special proposition $\mathbf{u}$. *We assume that every interpretation* I *satisfies* $I(\mathbf{u}) = 1/2$ *and thus* $\hat{I}(\sim\mathbf{u}) = 1/2$. *This special proposition denotes* **unknown** *or* **undefined.**

Theorem 4.1 *Generalization of [2] Every non-negative logic program has a unique least 3-valued model.*

Next we define the program transformation P/M ($PmoduloM$), which is an extension to the GL-transformation defined in [3], with an extra rule to account for the third truth value.

Definition 4.2 *Modulo transformation([9])*

Let P *be a logic program and let* I *be a 3-valued interpretation. By the extended GL-transformation of* P *modulo* I *we mean a new (non-negative) program* P/I *obtained from* P *by performing the following three operations:*

- *Removing from* P *all rules which contain a negative premise* $L = \sim A$ *such that* $\hat{I}(L) = 0$.
- *Removing from* P *all rules those negative* $L = \sim A$ *which satisfy* $\hat{I}(L) = 1$.
- *Replacing in all rules those negative premises* $L = \sim A$ *which satisfy* $\hat{I}(L) = 1/2$ *by* $\mathbf{u}$.

Since the resulting program P/I is non-negative, by theorem 4.1 it has a unique least 3-valued model. We define $\Gamma^*(I)$ (a generalization of the Γ operator [3]) to be the 3-valued least model of P/I.

Definition 4.3 *Extended Stable Model A 3-valued interpretation* I *of a logic program* P *is called an Extended Stable Model of* P *iff* $\Gamma^*(I) = I$.

It is easy to see that this definition extends the definition of SMS. In fact, every Stable Model of a program P is also an XS Model of P.

In order to easily calculate the XS Model of a program let us give a constructive definition of the Γ^* operator. For this purpose let us first define Ψ^* which is a generalization of the van Emden-Kowalski operator Ψ.

Definition 4.4 Ψ^* *operator*

Suppose that P *is a non-negative program,* I *is an interpretation of* P *and* A *is a ground atom. Then* $\Psi^*(I)$ *is an interpretation defined as follows:*

- $\Psi^*(I)(A) = 1$ *iff there is a rule* $A \leftarrow A_1, \ldots, A_n$ *in* P *such that* $I(A_i) = 1$ *for all* $i \leq n$.
- $\Psi^*(I)(A) = 0$ *iff for every rule* $A \leftarrow A_1, \ldots, A_n$ *there is an* $i \leq n$ *such that* $I(A_i) = 0$.
- $\Psi^*(I)(A) = 1/2$, *otherwise.*

In order to iterate this operator we define $\Psi^{*\uparrow n} = \Psi^*(\Psi^{*\uparrow n-1})$ and $\Psi^{*\uparrow 0} = < \{\}; \mathcal{H} >$.

Definition 4.5 Γ^* *operator defined constructively*

Let I *be an interpretation and* P *a logic program. We define* $\Gamma^*(I)$ *in program* P *to be* $\Gamma^*(I) = \Psi^{*\uparrow\omega}(P/I)$.

These semantics were proved equivalent to the WF Semantics, in [9] with the following theorem:

Theorem 4.2 *([9]) The Well Founded Model of a program* P *is the F-least Extended Stable Model of* P. *Consequently the WFS coincides with the XSMS.*

This result can be obtained by proving that the Γ^* operator is equivalent to the Ω operator defined above in section 2. This leads to an alternative constructive definition of the WF Model:

- $I_0 = < \{\}; \{\} >$.
- $I_{\alpha+1} = \Gamma^*(I_\alpha)$.

The WF Model of P is the (F-least) fixed point of this sequence, i.e. I_λ.

Because an XSM I obeys $\Gamma^*(I) = I$ (c.f.definition 4.3), and Γ^* is equivalente to Ω, we have $\Omega(I) = I$, and, by theorem 2.1, $\Theta_I^{\uparrow\omega} = I$ for all XSMs I.

5 Calculating the XSMs

The definition 4.3 of XS Model is not a constructive one. It allows simply a verification, for a given interpretation, if it is or not a stable model. For finding the XSMs of a program P, one has, for each I of the $3^{|\mathcal{L}|}$ interpretations of P ($\mathcal{L}$ being the set of atoms of the program), to calculate $\Gamma^*(I)$ and, if it is equal to I, I is indeed a model.

Example 5 Let P be the program: $P = \{a \leftarrow\sim b; b \leftarrow\sim a\}$. For this program we have first to consider all the 9 possible interpretations of P, which are:

$$\begin{array}{llllllllll} I_1 & = & < \{\}; \{\} > & I_2 & = & < \{\}; \{b\} > & I_3 & = & < \{b\}; \{\} > \\ I_4 & = & < \{\}; \{a\} > & I_5 & = & < \{\}; \{a, b\} > & I_6 & = & < \{b\}; \{a\} > \\ I_7 & = & < \{a\}; \{\} > & I_8 & = & < \{a\}; \{b\} > & I_9 & = & < \{a, b\}; \{\} > \end{array}$$

Then we have to calculate for each interpretation I_i the corresponding $gama I_i$:

$$\begin{array}{llllllllll} \Gamma^*(I_1) & = & < \{\}; \{\} > & \Gamma^*(I_2) & = & < \{a\}; \{\} > & \Gamma^*(I_3) & = & < \{\}; \{a\} > \\ \Gamma^*(I_4) & = & < \{b\}; \{\} > & \Gamma^*(I_5) & = & < \{a, b\}; \{\} > & \Gamma^*(I_6) & = & < \{b\}; \{a\} > \\ \Gamma^*(I_7) & = & < \{\}; \{b\} > & \Gamma^*(I_8) & = & < \{a\}; \{b\} > & \Gamma^*(I_9) & = & < \{\}; \{a, b\} > \end{array}$$

After this we can conclude that the XSMs of P are $< \{\}; \{\} >$ (the WF Model), $< \{b\}; \{a\} >$ and $< \{a\}; \{b\} >$. □

In this section we will not consider the calculation of $\Gamma^*(I)$ for a given I. Instead we will consider how to find a sub-set of the set of all interpretations which are worth considering to test for stability.

Advice 5.1 *According to theorem 4.2, if an atom has a truth value true or false in the WF Model, it will have the same value in all XS Models. So one has only to check for candidate XSMs interpretations those that have this property.*

Thus XSMs may differ only from the WFM on those literals undefined in the latter.

Definition 5.1 *Saturated model*
We say that a model M is saturated if there is no atom A such that $M(A) = 1/2$.

With this we are able to state a result for a particular case.

Proposition 5.1 *If the WF Model of a program P is saturated then it is the only XSM of P.*

Another interesting proposition that might help to find the XSMs of a program is the following:

Proposition 5.2 *Let $XSM_1 = < T_1; F_1 >$ and $XSM_2 = < T_2; F_2 >$ be two XSMs of a program P. Let $XSM1 \cup XSM_2 = < T_1 \cup T_2; F_1 \cup F_2 >$. If $XSM_1 \cup XSM_2$ is an interpretation, i.e. $T_1 \cup T_2$ and $F_1 \cup F_2$ are disjoint, then $XSM_1 \cup XSM_2$ is an XSM of P.*

This proposition suggests to begin by verifying the stability of the smaller interpretations, and then to conclude rapidly for the others. With the results above, let us attempt now to find the XSMs of a more complex example.

Example 6 Let P be the program:

$$a \leftarrow \sim b. \quad c \leftarrow \sim d. \quad d \leftarrow a, \sim c. \quad e \leftarrow \sim f, \sim b. \quad f \leftarrow \sim e.$$

For this program we have 729 (3^6) possible interpretations. Which of them should be considered ?

The WF Model of P is $< \{a\}; \{b\} >$. So it is not worth considering interpretations that don't have a true and b false; this brings the number of interpretations down to 81. Let us first consider those interpretations that have one literal more than the WFM:

$$\begin{array}{llllll} \Gamma^*(< a,c;b >) & = & < \{a\};\{b,d\} > & \Gamma^*(< a;b,c >) & = & < \{a,d\};\{b\} > \\ \Gamma^*(< a,d;b >) & = & < \{a\};\{b,c\} > & \Gamma^*(< a;b,d >) & = & < \{a,c\};\{b\} > \\ \Gamma^*(< a,e;b >) & = & < \{a\};\{b,f\} > & \Gamma^*(< a;b,e >) & = & < \{a,f\};\{b\} > \\ \Gamma^*(< a,f;b >) & = & < \{a\};\{b,e\} > & \Gamma^*(< a;b,f >) & = & < \{a,e\};\{b\} > \end{array}$$

None of them is an XSM. But let's try the resulting interpretations after applying Γ^* to the union of the previous results and the interpretations:

$$\begin{array}{llllll} \Gamma^*(< a,c;b,d >) & = & < \{a,c\};\{b,d\} > > & \Gamma^*(< a,d;b,c >) & = & < \{a,d\};\{b,c\} > \\ \Gamma^*(< a,e;b,f >) & = & < \{a,e\};\{b,f\} > > & \Gamma^*(< a,f;b,e >) & = & < \{a,f\};\{b,e\} > \end{array}$$

These are stable. Thus, by proposition 5.2, the following interpretations are also XSMs of P:

$$< \{a,c,e\};\{b,d,f\} > \quad < \{a,c,f\};\{b,d,e\} > \quad < \{a,d,e\};\{,\} > b,c,f \quad < \{a,d,f\};\{b,c,e\} >$$

From the program it is easy to see that no interpretation that has the same truth value for c and d or for e and f are stable; this brings the number of interpretations down to 9, already found. Thus the XSMs of P are:

$$\begin{array}{rcll} < \{a\};\{b\} > & \text{The WF Model} & < \{a,c\};\{b,d\} > & < \{a,d\};\{b,c\} > \\ < \{a,e\};\{b,f\} > & & < \{a,f\};\{b,e\} > & < \{a,c,e\};\{b,d,f\} > \\ < \{a,c,f\};\{b,d,e\} > & & < \{a,d,e\};\{b,c,f\} > & < \{a,d,f\};\{b,c,e\} > \end{array}$$

□

With the suggestions from this example we can formalize a little bit more the search for XSMs.

Definition 5.2 *Generating set*

The generating set $\mathcal{GS}$ of a program P with WF Model M_P, is the set of all interpretations I built with just all those atoms A of P, such that $M_P(A) = 1/2$, and $I(A) = 0$.

Example 7 In example 6 the generating set is:

$\mathcal{GS}$= $\{< \{\}; \{c\} >, < \{\}; \{d\} >, < \{\}; \{e\} >, < \{\}; \{f\} >, < \{\}; \{c,d\} >, \ldots, < \{\}; \{c,d,e,f\} >\}$ □

Let us now define a partial operator that for some interpretations of a program P gives us XSMs of P.

Definition 5.3 *XS operator*

Let I be an interpretation of a program P. We say that XS is defined for I if there is a finite sequence of interpretations $\{I_\alpha\}$ such that:

- $I_0 = I$

- $I_n = \Gamma^*(I_{n-1}) \cup I_{n-1}$ *where each* I_i *is an interpretation,*

and it has a least fixed point I_ω *such that* $\Gamma^*(I_\omega) = I_\omega$. *If* XS *is defined for* I *we say that* $XS(I) = I_\omega$.

Note that, by definition of the operator and definition 4.3, $XS(I)$ *is always an XSM of* P.

Definition 5.4 $\leq_{False}$

If $I = <T;F>$ *and* $I' = <T';F'>$ *are two interpretations, then we say that* $I \leq_{False} I'$ *iff* $F \subseteq F'$.

We can now provide a constructive definition for XSMs.

Advice 5.2 *Let* P *be a logic program with WF Model* M_P *and generating set* $\mathcal{GS}$.

1. *Let* $XM_0 = \{M_P\}$ *and* $Cand_0 = \mathcal{GS}$.

2. *If* $Cand_i = \{\}$ *then the set of all XSMs of* P *is* XM_i.

3. *Otherwise build* XM_{i+1} *and* $Cand_{i+1}$ *with the following rules*

3.1. *Choose any* $\leq_{False}$ *minimal element* I *of* $Cand_i$. *Let* $I_i = I \cup M_P$.

3.2. *If* XS *does not apply to* I_i *then let* $XM_{i+1} = XM_i$ *and let* $Cand_{i+1} = Cand_i - \{Ii\}$.

3.3. *Otherwise let* $XSM_{new} = XS(I_i)$. *Let* $XM_{i+1} = \{I | \exists M \in XM_i, I = M \cup XSM_{new}$ *and* I *is an interpretation*$\} \cup XM_i$; *by construction, each such* I *is an XSM. Let* $Cand_{i+1} = Cand_i - L_s$, *where* L_s *is the set of all* $C \in Cand_i$ *such that* C *is* $\leq_{False}$ *minimal in* XM_{i+1}.

This result by itself is not very practical because $\mathcal{GS}$ is usually quite a big set. But, as we always begin with the minimal candidates, and we throw out all candidates that are $\leq_{False}$ than any XSM already built, we can construct and restrict the $Cand_i$ set on the fly. Let us see how this works with the help of some examples.

Example 8 Consider program $P = \{p \leftarrow\sim q; r \leftarrow\sim a; a \leftarrow\sim b; b \leftarrow\sim a\}$. The WF Model of P is $M_P = <\{p\};\{q\}>$ and thus $XM_0 = \{<\{p\};\{q\}>\}$ and $Cand_0 = \mathcal{GS}$.

Any interpretation built by adding one atom to M_P is a minimal element of $Cand_0$, so it can be chosen at step 3.1. Let us first consider, e.g., $I_0 = <\{p\};\{q,a\}>$.

$\Gamma^*(I_0) \cup I_0 = <\{p,b,r\};\{q,a\}>$

$\Gamma^*(<\{p,b,r\};\{q,a\}>) \cup <\{p,b,r\};\{q,a\}> = <\{p,b,r\};\{q,a\}>$

and so, by definition, $XS(I0) = <\{p,b,r\};\{q,a\}>$.

Using step 3.3 we have $XM_1 = \{<\{p\};\{q\}>, <\{p,b,r\};\{q,a\}>\}$. $Cand_1$ does not have I_0.

In order to build XM_2 we will now consider another interpretation built by adding another atom to M_P, e.g., $I_1 = <\{p\};\{q,b\}>$. As $XS(I_1) = \{<\{p,a\};\{q,r,b\}>\}$ then $XM_2 = XM_1 \cup \{<\{p,a\};\{q,r,b\}>\}$. Now $Cand_2$ only contains elements that have both a and b, or a and r with truth value false[4] Because of the second and third rules of P, XS does not apply to these interpretations.

For example let us consider $I_2 = <\{p\};\{q,a,b\}>$:

$\Gamma^*(<\{p\};\{q,a,b\}>) \cup <\{p\};\{q,a,b\}> = <\{p,a,b,r\};\{q,a,b\}>$ which is not a valid interpretation. So XS does not apply to I_2.

Thus, applying successively 3.2 we obtain the set of all XSMs of P is:

$$\{<\{p\};\{q\}>, <\{p,a\};\{q,r,b\}>, <\{p,r,b\};\{q,a\}>\}.$$

□

[4]Note that $<\{p\};\{q,b\}> \leq_{False} <\{p\};\{q,b,r\}>$ and $<\{p\};\{q,b,r\}> \leq_{False} <\{p\};\{q,b,r\}>$.

Example 9 Let $P = \{p \leftarrow\sim q; b \leftarrow\sim b, p; a \leftarrow\sim b; b \leftarrow\sim a\}$.
$XM_0 = \{< \{p\}; \{q\} >\}$ and $Cand_0 = \mathcal{GS}$. Let $I_0 = < \{p\}; \{q, a\} >$.
$\Gamma^*(I_0) \cup I_0 = < \{p, b\}; \{q, a\} >$
$\Gamma^*(< \{p, b\}; \{q, a\} >) \cup < \{p, b\}; \{q, a\} > = < \{p, b\}; \{q, a\} >$.
Thus $XM_1 = \{< \{p\}; \{q\} >, < \{p, b\}; \{q, a\} >\}$ and $Cand_1 = \{< \{p\}; \{q, b\} >, < \{p\}; \{q, a, b\} >\}$.
Choosing $I_1 = < \{p\}; \{q, b\} >$ we get:
$\Gamma^*(I_1) \cup I_1 = < \{p, a, b\}; \{q, b\} >$, which is not a valid interpretation (cf. definition 2.1).
So $XM_2 = XM_1$ and $Cand_2 = \{< \{p\}; \{q, a, b\} >\}$. As:
$\Gamma^*(< \{p\}; \{q, a, b\} >) \cup < \{p\}; \{q, a, b\} > = < \{p, a, b\}; \{q, a, b\} >$, the set of all XSMs of P is $\{< \{p\}; \{q\} >, < \{p, b\}; \{q, a\} >\}$. □

Example 10 Let $P = a \leftarrow\sim a; a \leftarrow\sim b; b \leftarrow\sim a$. The WFM of P is $< \{\}; \{\} >$.
$XM_0 = \{< \{\}; \{\} >\}$ and $Cand_0 = \{< \{\}; \{a\} >, < \{\}; \{b\} >, < \{\}; \{a, b\} >\}$.
Let $I_0 = < \{\}; \{a\} >$. $\Gamma^*(I_0) \cup I_0 = < \{a\}; \{a, b\} >$ which is not a valid interpretation. In fact because of the first rule, for every element ACS of $\mathcal{GS}$ such that $ACS(a) = 0$, XS does not apply.
This result can be generalized in order to get an useful property to prune the elements of $\mathcal{GS}$:

Advice 5.3 *An atom that depends on itself through an odd number of $\sim$ for some rule defined for it, is never false in any of the XSMs.*

So the only candidate is now $< \{\}; \{b\} >$ and as $XS(< \{\}; \{b\} >) = < \{a\}; \{b\} >$, the set of all XSMs of P is $\{< \{\}; \{\} >, < \{a\}; \{b\} >\}$. □

Acknowledgements

We thank ESPRIT BRA COMPULOG (no. 3012), Instituto Nacional de Investigação Científica, Junta Nacional de Investigação Científica e Tecnologia and Gabinete de Filosofia do Conhecimento for their support.

References

[1] M. Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.

[2] Van Gelder and R. Kowalski. The semantics of predicate logic as a programming language. *JACM*, 4(23):733–742, 1976.

[3] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. A. Bowen, editors, *5th International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[4] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Contradiction within Well Founded Semantics. In MIT Press, editor, *LPNMR'91 (to appear)*, 1991.

[5] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Top-Down procedures for well founded semantics. Technical report, CRIA/UNINOVA, 1990.

[6] L. M. Pereira, J. N. Aparício, and J. J. Alferes. A derivation procedure for extended stable models. In *IJCAI 91 (to appear)*. Morgan Kaufmann Publishers, 1991.

[7] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Hypothetical reasoning with well founded semantics. In *SCAI'91*, 1991.

[8] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Nonmonotonic reasoning with well founded semantics. In *ICLP'91 (to appear)*, 1991.

[9] H. Przymusinska and T. Przymusinski. *Semantic Issues in Deductive Databases and Logic Programs*. Formal Techniques in Artificial Intelligence. North Holland, 1990.

[10] T. Przymusinski. Every logic program has a natural stratification and an iterated fixed point model. In *8th Symposium on Principles of Database Systems*. ACM SIGACT-SIGMOD, 1989.

[11] T. Przymusinski. Extended stable semantics for normal and disjunctive programs. In *ICLP'90*, pages 459–477, 1990.

[12] T. Przymusinski. Stationary semantics for disjunctive logic programs and deductive databases. In *NACLP'90*, pages 40–57, 1990.

[13] T. C. Przymusinski. Perfect model semantics. In R. A. Kowalski and K. A. Bowen, editors, *5th International Conference on Logic Programming*, pages 1081–1098. MIT Press, 1988.

[14] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, pages 221–230, 1990.

[15] D.S. Warren. The XWAM: A machine that integrats prolog and deductive databases. Technical report, SUNY at Stony Brook, 1989.