Contradiction Removal within Well Founded Semantics

Luís Moniz Pereira
José Júlio Alferes
Joaquim Nunes Aparício
CRIA Uninova and DCS, U.Nova de Lisboa
2825 Monte da Caparica, Portugal
({lmp,jja,jna}@fct.unl.pt)

Abstract

Our purpose is to define a semantics that extends Well Founded Semantics for programs with classical negation, and which avoids the absence of models caused by contradictions brought about by closed world assumptions. This extension relies on allowing to take back such closed world assumptions, through making their truth value become undefined, and thus permiting noncontradictory models to appear. We take back such assumptions in a minimal way for all alternative ways of removing contradictions, by means of simple transformations of the original program. The transformed programs have contradiction free Well Founded Models. Moreover, we identify a unique model that defines the semantics of the original program, which is included in all the alternative contradiction free models. This unique model coincides with the Well Founded Model when the latter is noncontradictory. The notions of minimality and contradiction removal employed are useful for dealing with Belief Revision. These techniques for removing contradictions in the sense of classical logic, are also adequate for removing contradiction in the sense of integrity constraints violation. Another important result is that our removal semantics (the contradiction removal semantics) is defined as the Well Founded Model of a derived program obtained by a simple transformation from the original one. Thus no new model determining algorithms are needed. For noncontradictory programs the two programs coincide.

Introduction

Well Founded Semantics (WFS) (introduced in [15]) is a 3-valued semantics extending previous logic program semantics to the class of all normal programs. This semantics has been proven equivalent to a natural extension to 3 values of the Stable Model Semantics of [1], i.e. the 3-valued (or extended) Stable Model Semantics (XSMS) of [10, 12]. The WF Model (WFM) of a program P is the smallest XS Model of P. Recently, in [12], an extension of WFS encompassing programs with classical negation has been proposed. It is similar to the extension of [2] to deal with classical negation within

Stable Model Semantics. This approach transforms a program with classical negation into another without it, by replacing every occurrence of each classically negated literal by a positive literal with a new predicate symbol. Afterwards the program models are obtained as usual. Finally, every contradictory model [1] is rejected. This seems quite reasonable for noncontradictory programs, i.e. those whose WFM is noncontradictory. But for contradictory ones it seems too strong to throw out all models. Consider, for example, the statements: *Birds, not known to be abnormal, fly. Tweety is a bird and does not fly. Socrates is a man.* which can be naturally represented by the program[2]:

$$\begin{array}{rclr} fly(X) & \leftarrow & bird(X), \sim abnormal(X). & bird(tweety) \\ \neg fly(tweety). & & & man(socrates). \end{array}$$

The WFS in the [12] approach to classical negation would lead to the inexistence of models for this program, which in our opinion is inadequate. We should at least be able to say that $Socrates$ is a man. In our view, it is also reasonable to conclude that $tweety$ is a $bird$ and doesn't fly, because the rule saying that it doesn't fly, since it is a fact, makes a stronger statement than the one concluding it $flies$. The latter relies on the assumption of non-abnormality, enforced by the Closed World Assumption (CWA) treatment of the negation as failure involving the abnormality predicate. In fact, whenever an assumption leads to a contradiction it seems logical to be able to take it back in order to remove the contradiction.

Our purpose is to define a semantics that extends WFS for programs with classical negation, and which avoids the absence of models caused by contradictions brought about by closed world assumptions. This extension relies on allowing to take back such CWA assumptions about literals, through making their truth value become undefined, and thus permiting noncontradictory models to appear. We take back such assumptions in a minimal way, in all alternative ways of removing contradictions. Moreover, we identify a single unique model that defines the semantics, this model being included in all the alternative contradiction removing ones. This is akin to the approach of the XSMS of [10, 12]. Another important similarity between our extension to WFS and WFS itself is that the latter removes $\sim$ related contradictions by giving some literals the undefined truth value, and we remove $\neg$ related contradictions in a similarly. These notions and techniques of contradiction removal are also adequate to deal with more general kinds of contradictions, namely the ones arising from integrity constraint violation.

The notions of minimality and contradiction removal employed are useful for dealing with Belief Revision through WFS. Consider for example the noncontradictory program: $P = \{p \leftarrow \sim q; \neg p \leftarrow r, \sim t\}$; and the new information: r. Our proposed semantics for $P \cup \{r\}$ provides the minimal model $\{r\}$, and two extended additional ones, namely $\{r, p, \sim q\}$ and $\{r, \neg p, \sim t\}$. These two models can be seen as alternative minimal changes to the WFM of P in order to incorporate the new information: one making t undefined rather than false by CWA, and the other making q undefined instead. Model $\{r\}$ is obtained by making both t and q undefined. It is the one with sufficient and necessary changes compatible with the new information, whenever no preference is enforced about which relevant CWA literals to unassume, in fact by unassuming them all. Furthermore, the semantics also allows to determine programs from the original one, whose WF Models are each of the noncontradictory models of our extension to WFS, such as the two (maximal) noncontradictory models above.

Another result is that, for any contradictory program, our Contradiction Removal Semantics (CRS) is defined as the WFS of a derived program obtained by a simple transformation from the original one. For noncontradictory programs the two semantics coincide. In fact, the transformation gives the original program. By relying on the WFS of the transformed programs, no new model determining procedures are needed; they can be found in [11, 16, 4, 6].

The structure of this paper is as follows: in section 1 we define the language used and a program transformation similar to the ones in [2] and [12] for dealing with classical negation, and integrity constraints. In section 2 we define some useful sets for establishing the causes of and the removal of contradictions within WFS. Afterwards we define our Contradiction Removal Semantics, based on alternative transformed programs. In section 4 we prove that the changes made to the original WF Model are in a sense minimal, and provide an alternative equivalent definition of the semantics.

1 Language

In what follows a program is a set of rules of the form:

$$H \leftarrow B_1, \ldots, B_n, \sim C_1, \ldots, \sim C_m. \quad m \geq 0, n \geq 0.$$

The symbol $\sim$stands for negation as failure, i.e. negation in the sense of WFS. $H, B_1, \ldots, B_n, C_1, \ldots, C_m$ are classical literals. A classical literal is either an atom A or its classical negation $\neg A$. As in [2] and in [12], we first transform such programs into ones obtained by replacing every occurrence of every classically negated literal, say $\neg l(X)$, by another with the same arguments and a new predicate name, say '$\neg l$'(X). Then we introduce, for every atom $a(X)$ of the program, the integrity constraint $\perp \leftarrow a(X),$'$\neg a$'(X), where $\perp$ stands for not true[3]. In Gelfond's approach [2], these constraints are also present, but as model theoretical pruning devices. Here this is not the case, these rules being treated as any other, and incorporated into the language.

Example 1 The contradictory program:

$$p \quad \leftarrow \quad \sim q. \quad p \quad \leftarrow \quad \sim r. \quad \neg p \quad \leftarrow \quad \sim t. \quad a \quad \leftarrow \quad \sim b.$$

is first transformed into:

$$p \leftarrow {\sim}q. \quad p \leftarrow {\sim}r. \quad \neg p \leftarrow {\sim}t. \quad a \leftarrow {\sim}b. \quad \bot \leftarrow p, \neg p.$$

□

After the above transformation contradiction removal is tantamount to preventing the appearance of literal $\bot$ in the WFS. This notion of contradiction can be extended to encompass the prevention of integrity constraint (IC) violation, if each IC of the form $\leftarrow A_1, \ldots, A_n$ is transformed into the rule $\bot \leftarrow A_1, \ldots, A_n$.

2 Contradiction Support Sets and Contradiction Removal Sets

In this section we define the (assumptive) Contradiction Support Sets and (assumptive) Contradiction Removal Sets. These sets are the main notions required by Contradiction Removal Semantics.

Intuitively, Contradiction Support Sets are sets of $\sim$negative literals present in the WF Model which are sufficient to support $\bot$ in the WFM (and thus support a contradiction)[4]. I.e. given their truth the truth of $\bot$ is inevitable. Contradiction Removal Sets are built from the Contradiction Support Sets. Intuitively, they are minimal sets of literals chosen from the Support Sets such that any support of $\bot$ requires at least one literal in the set. Consequently, if all literals in some Contradiction Removal Set were to become undefined in value then no support of $\bot$ would exist. We shall see how such literals can be made undefined through revising a contradictory program by means of a transformation.

Example 2 Consider the program of example 1. Its contradiction support sets are $\{{\sim}q, {\sim}t\}$ and $\{{\sim}r, {\sim}t\}$, and its contradiction removal sets are $\{{\sim}q, {\sim}r\}$ and $\{{\sim}t\}$.

Suppose we had q and r both undefined. In that case $\bot$ would also be undefined, the program becoming noncontradictory. The same would happen if t alone became undefined. No other set, not containing one of these two alternatives, has this property. □

Definition 2.1 *A Support Set*[5] *of a literal L belonging to the WF Model M_P of a program P, represented as $SS_P(L)$, or $SS(L)$ for short, is obtained as follows:*

- *If L is an atom:*
  - *Choose some rule of P for L where all the literals in its body belong to M_P. One $SS(L)$ is obtained by taking all those body literals plus the literals in some SS of each body literal.*

- *If $L = \sim A$:*
  - *If there are no rules for A in P then the only $SS(L)$ is $\{\}$.*
  - *Otherwise, choose from each rule defined for A, a literal such that its complement*[6] *belongs to M_P. A $SS(L)$ has all those complement literals, and the literals of a SS of each of them.*

By considering all possible rules of P for a literal all its SSs are obtained.

Example 3 Consider the program:

$$p \leftarrow \sim q, r. \quad p \leftarrow \sim b. \quad r \leftarrow a. \quad b \leftarrow q, c. \quad a.$$

whose WF Model is $M_P = \{p, r, a, \sim q, \sim b, \sim c\}$. Let us compute the SSs of p.

By the first rule for p, because $\sim q, r \in M_P$ they belong to a $SS(p)$. As there are no rules for q the only SS of $\sim q$ is $\{\}$. From the single rule for r we get that the single SS of r is $\{a\}$, since the only SS of a is $\{\}$. So one $SS(p) = \{\sim q, r, a\}$. By the second rule for p we get the only other $SS(p) = \{\sim b, \sim q, \sim c\}$. □

Proposition 2.1 *Existence of Support Set*

Every literal L belonging to the WF Model of a program P has at least one support set $SS_P(L)$.

Proof: The proof follows from the definition of WFM as in [10], and is omitted for brevity. ◇

We have a special interest in those negative literals true by CWA, i.e. those for each there are no rules defined for their complement. With the purpose of identifying such literals we define the Assumption Support Set of a literal in the WF Model.

Definition 2.2 *An Assumption Support Set (ASS) of a literal L in the WFM is the (possibly empty) subset of some $SS_P(L)$, which contains exactly all its $\sim$negative elements having no rules for their complement. We represent an ASS of a literal L in a program P as $ASS_P(L)$ or $ASS(L)$, for short. For simplicity we represent these sets using the literal's complements (i.e. with atoms).*

Example 4 The two $ASS(p)$ in the previous example are $\{q\}$ and $\{q, c\}$. □

Definition 2.3 *A $\perp$-Assumption Support Set ($\perp$ASS) is an $ASS(\perp)$.*

These are sets of atoms false by CWA in the WF Model of the program, involved in supporting contradiction (i.e. $\perp$) in the program[7].

Having defined the sets of CWA literals that together support some literal, it is easy to produce sets of CWA literals such that, if all become undefined, the truth of that literal would necessarily become ungrounded.

Definition 2.4 *A Removal Set (RS) of a literal L belonging to the WFM of a program P is a set of atoms formed for the union of some nonempty subset from each* $ASS_P(L)$. *If the empty set is an* $ASS_P(L)$, *then the only* $RS(L)$ *is the empty set. Note that a literal not belonging to the WFM of P, has no RSs defined for it.*

In view of consider minimal changes to the WF Model, we next define those RSs which are minimal in the sense that there is no other RS containing them.

Definition 2.5 *A RS of a literal L* $RS_m(L)$, *in a program P, is minimal iff there exists no* $RS_i(L)$ *in P such that* $RS_m(L) \supset RS_i(L)$. *We represent a minimal RS of L in P as* $MRS_P(L)$.

Definition 2.6 *A Contradiction Removal Set (CRS) of program P is a minimal Removal Set of the (special) literal* $\bot$, *i.e. a CRS of P is a* $MRS_P(\bot)$.

3 Contradiction Removal Semantics

Now we define a contradiction removal semantics based on the CRSs. For this purpose we begin by identifying rules of a special form, which have the effect of prohibiting the falsity of an atom in any model. Such rules can prevent an atom being false by CWA, hence their name:

Definition 3.1 *The CWA inhibition rule for an atom A is* $A \leftarrow \sim A$.

Proposition 3.1 *For any program P containing a CWA inhibition rule for atom A, the truth value of A in any model of P is not false. Moreover, if this is the only rule for A, the truth value of A is undefined in all models.*

Proof: The proof follows directly from the properties of WF Semantics. $\diamondsuit$

These rules allows us, by adding them to a program, to force literals without clauses (false in the WFM), to become undefined. Guided by the CRSs, we now define the Contradiction Free Programs, obtained by adding specific CWA inhibition rules to the original program.

Definition 3.2 *A Neighbour Contradiction Free Program (NCFP) of a program P with CRS* C_i, *is the program obtained by adding to P one CWA inhibition rule for each element of* C_i. *A program with* $n > 0$ *CRSs can be transformed into n different NCFPs. If a program P has no CRSs then we define the only NCFP of P as being P itself*[8].

Note that we only add CWA inhibition rules for the atoms of a CRS. Because such atoms, by definition, have no rules for them, their truth value in the WFM of the $NCFP$ is, by proposition 3.1 above, always undefined.

Lemma 3.1 *If the empty set is not a CRS of a contradictory program P, then every NCFPs of P is noncontradictory.*

Proof: Consider any NCFP of P, say $NCFP_i$, with its corresponding (by hypothesis nonempty) CRS, say CRS_i . By proposition 3.1 every atom in CRS_i is undefined in $WFM(NCFP_i)$. By definition 2.6 of CRS, in every $SS(\bot)$ of P, at least one element was removed (i.e. became undefined) in $NCFP_i$. Thus none of the $SS(\bot)$ remain in $NCFP_i$. As undefined literals cannot make any other literal true or false, no new $SS(\bot)$ are renegated. So, there are no $SS(\bot)$ in $NCFP_i$, and by proposition 2.1, $\bot \notin WF(NCFP_i)$, i.e. $NCFP_i$ is noncontradictory. $\diamondsuit$

Example 5 Consider the contradictory program $P = \{p \leftarrow \sim q; \neg p \leftarrow \sim r\}$. The CRSs of P are $\{q\}$ and $\{r\}$. So it has two NCFPs:

$$\begin{array}{rcl} NCFP_1 & = & \{\bot \leftarrow p, \neg p; p \leftarrow \sim q; \neg p \leftarrow \sim r; q \leftarrow \sim q\} \quad \textit{and} \\ NCFP_2 & = & \{\bot \leftarrow p, \neg p; p \leftarrow \sim q; \neg p \leftarrow \sim r; r \leftarrow \sim r\}. \end{array}$$

$\square$

Each of these two programs can be seen as a transformation of the original program that minimally removes contradiction by taking back CWAs via their inhibition rules[9]. In this example, one can remove the contradiction in p either by going back on the closed world assumption of q or on that of r. The program that has the first effect is $NCFP_1$, the one with the second effect being $NCFP_2$. (Minimality is further discussed in section 4.) Having no preference for choosing just q or just r undefined, it seems natural that the resulting semantics should accomplish the effect of making them both undefined.

Definition 3.3 *Consider a program P resulting from the transformation in section 1. The Contradiction Removal Semantics of P is defined as the WFM[10] of the program:*

$$\bigcup_{i=1}^{n} NCFP_i$$

where $NCFP_i$, for $1 \leq i \leq n$, are all the $NCFPs$ of P. We dub this set the Contradiction Removal Well Founded Model ($CRWFM$).

Note that the Contradiction Removal Semantics is based on the WFM of a single uinque transformed program, as claimed before.

Theorem 3.2 *Relation with the WFM*

Whenever the WFS provides a WFM for a program P, the CRWFM coincides with its WFM.

Proof: If the WFS provides a model then the program is noncontradictory, since contradictory WF models are thrown out by WFS. In this case, by definition (2.1) of support set, there are no (assumptive) contradiction support sets and consequently no contradiction removal sets. So the only NCFP is P itself, and the CRWF Model becomes the WF Model of P. $\diamondsuit$

This theorem expresses why the new semantics is an extension to WFS. For every program WFS provides a semantics for, the CR Semantics also provides the same semantics. Moreover, in cases where the WFS of [12] does not provide any semantics (i.e. the WFM is contradictory), the CR Semantics does provide one, which is a unique and noncontradictory CRWFM.

Lemma 3.3 *The CRWFM of a program P is a subset of the WFM of each $NCFP$ of P.*

Proof: As the set of inhibition rules in the program providing the CRWFM is, by definition, a superset of the inhibition rules in each NCFP, and as (by proposition 3.1) such rules only have the effect of restricting the WFM of a program to which they are added, the CRWFM is always a subset of the WFM of each NCFP. $\diamondsuit$

Theorem 3.4 *Existence of a unique noncontradictory CRWF Model If the empty set is not a CRS of a program P then the CRWF Model exists and is noncontradictory.*

Proof: By hypothesis, program P has no empty CRS. Two cases occur: either there no CRSs and the conclusion follows by theorem 3.2; or there is at least one nonempty CRS. In this case by lemma 3.1 every NCFP of P is noncontradictory, i.e. their WFM does not contain $\perp$. By lemma 3.3, we can conclude that the CRWFM does not contain $\perp$, and thus is noncontradictory. $\diamondsuit$

Example 6 Recall the *"birds fly"* problem from the introduction. The only CRS is $\{abnormal(tweety)\}$. Thus the only NCFP is the original program augmented with $abnormal(tweety) \leftarrow \sim abnormal(tweety)$ and the CRWFM is $\{bird(tweety), \neg fly(tweety), man(socrates)\}$, as expected. $\square$

Example 7 Consider the following statements: *Let's go hiking if it doesn't rain. Let's go swimming if it doesn't rain. If we go swimming we will not go hiking. Let's go swimming if the water isn't cold.* and that they can be rendered by the set of rules P:

$$\begin{array}{rclrcl} hiking & \leftarrow & \sim rain. & swimming & \leftarrow & \sim rain. \\ \neg hiking & \leftarrow & swimming. & swimming & \leftarrow & \sim cold\_water. \end{array}$$

which has no noncontradictory WF Model. Its $ASS(\bot)$s are $\{rain\}$ and $\{rain, cold\_water\}$. Thus its CRSs are:

$$
\begin{array}{rcll}
\{rain\} \cup \{rain\} & = & \{rain\} & \text{and} \\
\{rain\} \cup \{rain, cold\_water\} & = & \{rain, cold\_water\}. &
\end{array}
$$

The only minimal CRS is $\{rain\}$, so the only NCFP of P is:

$$
\begin{array}{rcl rcl}
\bot & \leftarrow & hiking, \neg hiking. & rain & \leftarrow & \sim rain. \\
hiking & \leftarrow & \sim rain. & swimming & \leftarrow & \sim rain. \\
\neg hiking & \leftarrow & swimming. & swimming & \leftarrow & \sim cold\_water.
\end{array}
$$

and the CRWF Model of P is: $\{\sim cold\_water, swimming, \neg hiking\}$ which seems to be the intuitively acceptable result. □

Example 8 Consider the so-called Nixon diamond:

$$
\begin{array}{rcl}
\bot & \leftarrow & pacifist(X), hawk(X). \\
pacifist(X) & \leftarrow & quaker(X), \sim ab\_quaker(X). \\
hawk(X) & \leftarrow & republican(X), \sim ab\_republican(X). \\
quaker(nixon). & & republican(nixon).
\end{array}
$$

This contradictory program P has two NCFPs:

- one by adding to P $ab\_quaker \leftarrow \sim ab\_quaker$,
- another by adding to P $ab\_republican \leftarrow \sim ab\_republican$.

These two programs have the WF Models:

$$
\begin{array}{c}
\{hawk(nixon), quaker(nixon), republican(nixon), \sim ab\_republican(nixon)\}, \\
\{pacifist(nixon), quaker(nixon), republican(nixon), \sim ab\_republican(nixon)\}
\end{array}
$$

The CRWF Model of the program is the WFM of the program resulting from adding both inhibition rules $\{quaker(nixon), republican(nixon)\}$.

The importance of having a single model that determines the semantics (the CRWF Model) can be observed here, since there is no reason for preference between Nixon being a pacifist or a hawk. Nevertheless, the other models also give relevant information[11]. □

Another important use of CR Semantics, provided by the techniques for calculating the CRWF Model, is for Belief Revision within the WFS framework. There the NCFPs are seen as minimal revised theories, mutually incompatible[12], and their union as the minimal revised program when no preferences or choices are available. The application to counterfactual reasoning is explored in [5]

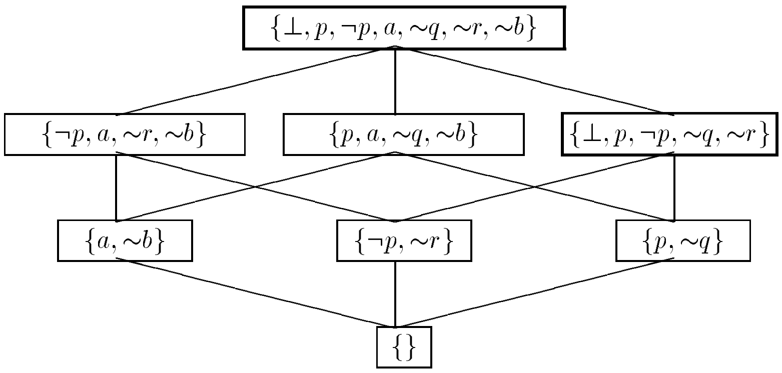
4 Minimality of changes

In this section we explain in what sense the changes in the CRWFM are minimal with respect to the WFM of the original program.

Definition 4.1 *Consider a contradictory program P whose (contradictory) WF Model has $L_1, L_2, \ldots, L_n$ as negative literals true by CWA. Let $Lits_{CWA} = \{L_1, L_2, \ldots, L_n\}$. A submodel of P is the WF Model of the program obtained by adding CWA inhibition rules for each element of some subset of $Lits_{CWA}$, thereby making them undefined in the WFM of derived program.*

Definition 4.2 *Given a contradictory program P we can define a set-inclusion lattice with the sets obtained by removing from its WFM each combination of CWA-true negative literals present in the WFM. The elements of that lattice are readily seen to be the submodels of P.*

Example 9 Consider the program $\{p \leftarrow \sim q; \neg p \leftarrow \sim r; a \leftarrow \sim b\}$. After the usual preliminary transformation, the WF Model obtained is $\{\bot, p, \neg p, a, \sim q, \sim r, \sim b\}$ and $Lits_{CWA} = \{\sim q, \sim r, \sim b\}$. So the submodels are:



where highlighted submodels are contradictory. $\square$

Lemma 4.1 *The WF Model of a NCFP of a program P is a Maximal Noncontradictory Submodel (MNS) of P.*

Proof: NCFPs are defined by the same transformation that defines submodels, but applied only to some of the sets that originate the latter (i.e. $\{S|S$ is a $CRS\} \subset \{S|S$ is a subset of $Lits_{CWA}\}$). So the WF Model of any NCFP is a submodel of P, and by definition it is noncontradictory. Finally, we prove, by contradiction, that there exists no noncontradictory submodel of P greater than the WF Model of the NCFP. Let M be a noncontradictory submodel of P greater than the WF Model of the NCFP. By definition, for M to be noncontradictory, none of the atoms of the CRS considered for the NCFP can be false. So each atom of that CRS must have a CWA inhibition rule. Thus that transformed program is in fact the NCFP and M is its WF Model. $\diamondsuit$

Lemma 4.2 *Each MNS of a program P is the WF Model of some NCFP of P.*

Proof: For a model to be maximal it must rely on adding a minimal number of CWA inhibition rules. By definition of CRS, there is no other combination of atoms false by CWA which, if undefined, would remove contradiction. As the definition of NCFP relies on those sets, and they rely on the same transformation as the submodels, each maximal noncontradictory submodel is the WF Model of some NCFP. $\diamondsuit$

By 4.1 and 4.2, the set of WF Models of the NCFPs of a program is a subset of all its submodels. To be more precise we state the following theorem, that results directly from these lemmas.

Theorem 4.3 *Correspondence between MNSs and NCFPs*
A submodel of program P is a MNS iff it is the WFM of a NCFP. Furthermore, there is a one-to-one correspondence between the MNSs and the NCFPs.

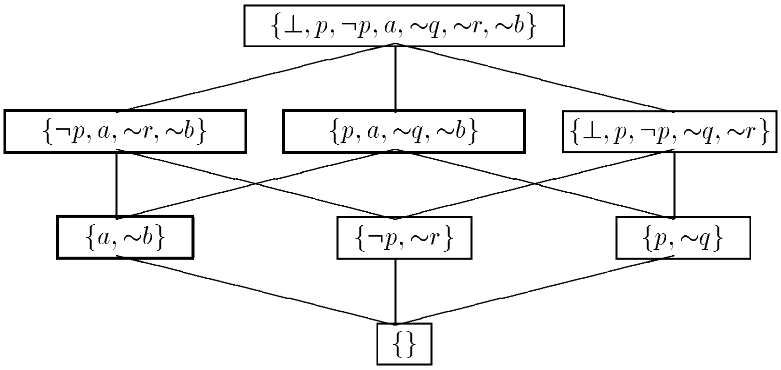
Note that, according to the definition of the lattice, maximal submodels correspond to minimal changes to the program, and minimal changes to the original (contradictory) WF Model. Moreover changing the truth value of CWA literals from false to undefined is less committing than changing it to true. Theorem 4.3 expresses that, in this sense, the NCFPs are the noncontradictory programs obtainable from the original one with minimal changes: CWA inhibition rules achieve, by design, just what is needed, neither more nor less. The next theorem characterizes the CRWF Model of a program, with respect to the lattice of submodels.

Theorem 4.4 *Minimality of Changes*
The CRWF Model of a program P is the meet of all the MNSs of P.

Proof: Let IS_i be the difference between P and the program that originates the MNS, M_i. This program is by theorem 4.3 a $NCFP_i$. It follows from the lattice construction, that the meet of all MNSs is the WFM of the program obtained by adding to the original program P a CWA inhibition rule undefining each element of the union of all IS_i. By definition 3.2, the ISs correspond one-to-one with the CRSs; so by definition 3.3 of the CRWFM it coincides with the meet of all MNSs. $\diamondsuit$

This theorem states that the CRWFM reflects the minimal changes to the WFM when preference among the MNSs is undecided.

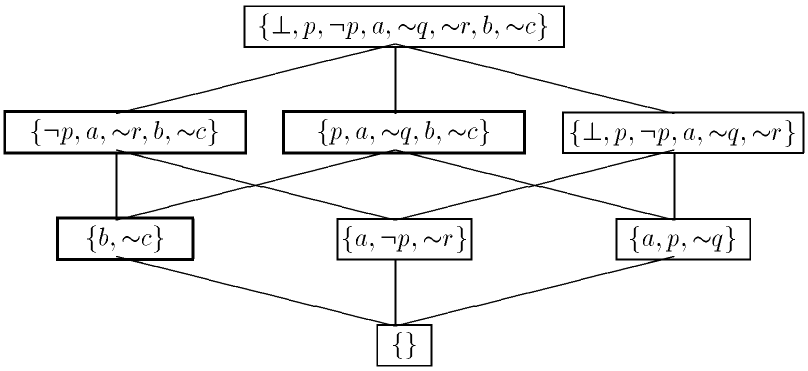
Example 10 Consider the above example. The figure below shows the correspondence between the models of the NCFPs and the submodels of P.



where highlighted submodels correspond to WF Models of some CRP. Their meet is the CRWF Model. □

This theorem shows in what sense the changes made to the WFM are minimal. The WF Model of each NCFP, since they are maximal submodels, are the ones *"closest"* to the WF Model of the original program not containing $\perp$ (i.e. noncontradictory), obtained by removing CWA literals (i.e. making them undefined). Note that the CRWFM is not in general the intersection of the MNSs. A similar situation occurs in WFS, where the intersection of the (set inclusion) maximal XSMs doesn't always correspond to the WFM.

Example 11 Consider the program $\{p \leftarrow \sim q; \neg p \leftarrow \sim r; a \leftarrow p; a \leftarrow \neg p; b \leftarrow \sim c\}$. Its lattice of submodels is:



where highlighted submodels correspond to WF Models of some CRP. Their meet is the CRWF Model.

In this case the CRWF Model is $\{b, \sim c\}$ and the intersection of all MNSs is $\{a, b, \sim c\}$ which is not a member of the lattice. In fact, to require the inclusion of literal a in the CRWFM would require a form of case-analysis involving the rules $a \leftarrow p$ and $a \leftarrow \neg p$, which is not a part of WFS. We explore this possibility in a forthcoming report. □

Proposition 4.1 *The CRWFM of a contradictory program P is contained in the intersection of its MNSs.*

Proof: By definition, the CRWFM of P is the WF Model of P augmented with CWA inhibition rules for each element of the union of all CRSs. As such rules only inhibit the truth or the falsity of a literal, adding them all leads at most to the same set of literals being true or false, compared to adding them separately. $\diamond$

Literals in the intersection of the MNSs of a program P but not in the CRWFM of P can be of special interest in the use of CRS techniques for belief revision. Note that such literals are exactly those that are lost for sure by being undecided, i.e. by making no preference; and that any preference whatsoever would keep them (as they belong to the intersection). They are also useful in this context for finding crucial literals [13, 14]. With this motivation, we show next how to identify them.

The literals in the intersection of all the MNSs not belonging to the CR-WFM are all those which aren't removed by any of the CRS_i corresponding to each MNS_i , i.e. $\forall_j \neg \exists_i RS_j(l) \subseteq CRS_i$, but are removed by their union, i.e. $\exists_j RSj(l) \subseteq \bigcup_i CRS_i$.

Definition 4.3 *An Indecision Set of a program P is the subset of all the literals l in the WF Model of P such that:*

$$\text{(1) } \exists_j RSj(l) \subseteq \bigcup_i CRS_i \text{ and (2) } \forall_j \neg \exists_i RS_j(l) \subseteq CRS_i$$

Example 12 For the program of example 11, its CRSs are $\{q\}$ and $\{r\}$. Let us identify which literals of the WF Model belong to the indecision set IS:

- The only removal set of b is $\{c\}$. Condition (1) is violated, thus $b \notin IS$.
- The only removal set of p is $\{q\}$. Condition (1) is verified, but (2) is not, because $\{q\} \subseteq \{q\}$. Thus $p \notin IS$.
- The only removal set of $\neg p$ is $\{r\}$. Condition (1) is verified, but (2) is not, because $\{r\} \subseteq \{r\}$. Thus $\neg p \subseteq IS$.
- The only removal set of a is $\{q, r\}$. In this case both condition are verified. So $a \in IS$.

So, as might be expected, $IS = \{a\}$. In fact, a is the only literal that depends separately both on p and $\neg p$, only one of which need be removed; so that a is in all WFMs of the NCFPs, but not in their CRWFM, where both p and $\neg p$ are removed. $\square$

5 Ongoing work

An important theme of ongoing research is to find an alternative definition of CR Semantics based on a single transformation of a contradictory program P, that not only gives us the CRWFM but also the several MNSs, the WF

Model of the transformed program being the CRWF Model of the original one. The other noncontradictory XSMs of the transformed program represent the alternative ways of removing contradiction, including the MNSs.

The study of the XSMs in CRS is another important subject of investigation, because the preservation of the XSM structure of the original program is useful for expressing defaults and abduction within the WFS, as shown in [8, 7]. Dealing with these models in the CRS can also be useful for dealing with belief revision and counterfactual reasoning [5] together with default reasoning and abduction.

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.

Notes

[1]By contradictory model we mean one that has some literal p and its classical corresponding negative literal $\neg p$ both true in it. A contradictory program is one having a contradictory WF Model.

[2]Here, and throughout the paper, $\sim$ stands for the negation as failure and $\neg$ for classical negation.

[3]Since, according to WFS, a literal can only be true if it has at least one rule defined for it, we need only add these $\bot$-rules for every pair of atoms $a(X)$ and $\neg a(X)$ that figure (both) as conclusions of rules in the program.

[4]This notion can be seen as a special case of the notion of Suspect Sets, both of wrong and missing solutions, in declarative debugging [9, 3]

[5]An alternative definition of support sets [4] relies on a notion of derivation for a literal in the WFS, and doesn't require the previous availability of the WF Model.

[6]The complement literal of an atom A is $\sim A$; that of a literal of the form $\sim A$ is A.

[7]Note that there is a close relationship between the SSs of $\bot$ and the sets of nogoods of Truth Maintenance Systems.

[8]As we shall see, by theorem 3.2, this is the case iff P is noncontradictory.

[9]Non-minimally contradiction removal programs can be defined similarly to NCFPs, by adding one CWA inhibition rule for each element of a non-minimal $RS(\bot)$, instead of for each element of a CRS (i.e. a minimal $RS(\bot)$). We won't need them in the sequel however, though they are useful for other purposes; c.f. [5].

[10]After replacing every atom of the form '$\neg a$'(X) by $\neg a(X)$.

[11]They correspond to the two usual default extensions.

[12]In the sense that neither contains any other.

References

[1] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *ICLP'88*, pages 1070–1080. MIT Press, 1988.

[2] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *ICLP'90*, pages 579–597, 1990.

[3] M. Calejo, L. M. Pereira and J. N. Aparício. Refining knowledge base updates. In *Simpósio Brasileiro de Inteligência Artificial*, 1990.

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

[5] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Counterfactual reasoning based on revising assumptions. Technical report, CRIA/Uninova, 1991.

[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*, IOS Press, 1991.

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

[9] L. M. Pereira and M. Calejo. A framework for Prolog debugging. In K. B. R. Kowalski, editor, *ICLP'88*. MIT Press, 1988.

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

[11] 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.

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

[13] A. Sattar and R. Goebel. Using crucial literals to select better thoeries. Technical report, U. Alberta, 1990.

[14] H. Seki and A. Takeuchi. An algorithm for finding a query which discriminates competing hypotheses. Technical report, ICOT, 1985.

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

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