Nonmonotonic Reasoning with Well Founded Semantics

Luís Moniz Pereira
AI Centre, Uninova and DCS, U. Nova de Lisboa
2825 Monte da Caparica, Portugal
lmp@fct.unl.pt

Joaquim Nunes Aparício
AI Centre, Uninova and DCS, U. Nova de Lisboa
2825 Monte da Caparica, Portugal
jna@fct.unl.pt

José Júlio Alferes
AI Centre, Uninova and DCS, U. Nova de Lisboa
2825 Monte da Caparica, Portugal
jjia@fct.unl.pt

Abstract

Well Founded Semantics is adequate to capture nonmonotonic reasoning if we interpret the Well Founded model of a program P as a (possibly incomplete) view of the world. Thus the Well Founded model may be accepted to be a definite view of the world and the extended stable models as alternative enlarged consistent belief models an agent may have about the world.

Our purpose is to exhibit a modular systematic method of representing nonmonotonic problems with the Well Founded semantics of logic programs. In this paper we use this method to represent and solve some classical nonmonotonic problems. This leads us to consider our method quite generic.

1 Introduction

Well Founded Semantics (WFS) [15] is adequate to capture nonmonotonic reasoning if we interpret the Well Founded model (WFM) of a program P as a (possibly incomplete) view of the world. Thus the WFM may be accepted to be a definite view of the world and the eXtended Stable Models (XSMs) as alternative enlarged consistent belief models an agent may have of the world, all of each containing the WFM. The agent’s world view may be completed (or refined) by hypothesizing (e.g., abducing, using default rules, the closed world assumption, etc.) about unknown information. Przymusinski [11, 12, 13] shows that Well Founded semantics is equivalent to suitable forms of four major nonmonotonic reasoning formalisms (Circumscription, Closed World Assumption, Autoepistemic Logic and Default Logic).

In the case the WFM is a 2-valued model then no degree of freedom is
left to the agent to conjecture or hypothesize about the world. If the WFM is 3-valued, hypothetical reasoning in the agent’s mind is delimited by the unknown facts of the WFM.

Our purpose is to exhibit a modular systematic method of representing nonmonotonic problems with the Well Founded semantics of logic programs. In this paper we present this method and use it to represent and solve some classical nonmonotonic problems. The programs and some others selected from [5] were tested with a WF Semantic interpreter produced in [7], and the desired conclusions were obtained. This encourages us to consider our method quite generic.

We begin by proposing a representation of information in hierarchical taxonomy problems. In the section "Possible Worlds" we represent hypothetical reasoning problems, interpret the results and propose a representation for unknown knowledge. Then, grounded on the former examples, we present a systematization of our problem representation approach. Afterwards, we use our methodology to represent additional classical nonmonotonic problems (planning, event calculus), arguing that it is sufficiently generic. Finally we stir some discussion and compare the method to related work.

2 Hierarchy Representation

In this section we illustrate how to represent a hierarchy in extended logic programs with the Well Founded semantics. In this representation we wish to express general rules, and their exceptions, as well as exceptions to exceptions. For instance, we want to represent general rules such as "birds normally fly" and "penguins normally don’t fly" where each can act as an exception to the other (in the absence of additional information). Furthermore, as in this instance, we wish to be able to express preference for one rule over another in case they conflict and are both applicable. Thus we can prefer the most specific information (e.g. for a penguin, which is a bird we want to conclude that it doesn’t fly, unless there is even more specific information regarding possible exceptions to the "penguins normally don’t fly" rule).

Our presentation is made step by step using the well known "Birds Fly" hierarchical taxonomy.

2.1 One level hierarchy

Let us begin with the simplest version of this problem: Normally birds fly.

We need a rule \( flies(X) \leftarrow bird(X) \) that applies whenever possible, but which can be defeated by exceptions. These exceptions can be had by explicit exception to the conclusion predicate (flies) or by a weaker exception to this rule only. In other words, we want to be able to say that a given bird doesn’t fly and also that for a given bird this rule doesn’t apply.
In order to allow exceptions to the conclusions, we state that a bird flies only if there is no evidence that it doesn’t. Specifically:

\[ \text{flies}(X) \leftarrow \text{bird}(X), \neg \text{flies}(X) \]

To allow exceptions to the rule, we "name" that rule:

\[ \text{flies}(X) \leftarrow \text{bird}(X), \neg \text{flies}(X), \neg \text{bird}\_\text{flies}(X). \tag{1} \]

To capture the notion that the rule applies whenever possible, i.e. that is a default rule, we introduce:

\[ \neg \text{bird}\_\text{flies}(X) \leftarrow \neg \text{bird}\_\text{flies}(X). \tag{2} \]

which with (1) captures the notion that if there is no way to prove that an object is an exception to the rule, then assume that the rule applies.

\[ \neg \text{bird}\_\text{flies}(X) \] accounts for exceptions to \( \text{bird}\_\text{flies}(X) \). By renaming \( \neg \text{bird}\_\text{flies}(X) \) to McCarthy’s \( \text{abnormal}\_\text{bird}(X) \) [6], then from (1) and (2) there follow the rules:

\[ \text{flies}(X) \leftarrow \text{bird}(X), \neg \text{abnormal}\_\text{bird}(X) \]

\[ \text{abnormal}\_\text{bird}(X) \leftarrow \neg \text{flies}(X). \]

We prefer having rule (2), it being a modular statement of an assumption about a predicate. Note also that (2) can be seen as a closed world assumption about \( \neg \text{bird}\_\text{flies}(X) \). Abridging, we represent the initial "Birds Fly" taxonomy with the (extended) logic program:

\[ \Pi_1 : \]
\[ f(X) \leftarrow b(X), \neg f(X), bf(X) \]
\[ bf(X) \leftarrow \neg bf(X) \]

where \( f \) stands for \( \text{flies} \), \( b \) for \( \text{bird} \) and \( bf \) for \( \text{bird}\_\text{flies} \).

Let us see what are the Extended Stable Models (XSM) of \( \Pi_1 \) if we add to it the facts, \( \text{Facts}_1 = \{b(a), b(b), \neg f(b)\} \). \( \Pi_1 \cup \text{Facts}_1 \) has a unique XSM Model coinciding with the Well Founded Model (WFM), which is

\[ \{f(a), \neg f(a), \neg f(b), f(b), b(a), b(b), bf(a), \neg bf(a), bf(b), \neg bf(b)\} \]

The fact that this model is 2-valued reflects that no choices are available to obtain other XSMs. This seems quite an acceptable result. Regarding \( b \) we know for sure that it is a bird and it doesn’t fly. We can say that the rule might apply, because there are no exceptions defined for it. About \( a \) we know that it is a bird, and as there is no exception for \( \text{flies} \), when \( X \) is bound to \( a \), we can say for sure that it flies. Thus the birds fly rule is applied maximally, as default rules are supposed to. This problem domain can next be extended by saying that all penguins are birds, penguins don’t fly, and \( b \) is a penguin.
2.2 Exceptions to exceptions

In a hierarchy, instead of saying that "penguins don’t fly", we would like

 to say that normally penguins don’t fly. This will allow us to express an

 exception to this exception rule for birds fly, and hence the possibility that

 an exceptional penguin may fly. Let the problem statement now be:

 Normally birds fly. Penguins are birds. Normally penguins don’t fly.

 According to what was said before this problem is represented as:

 \[ f(X) \leftarrow b(X), \neg f(X), bf(X). \quad \text{Normally birds fly} \]

 \[ bf(X) \leftarrow \neg bf(X). \quad \text{This is a default} \]

 \[ \neg f(X) \leftarrow p(X), \neg f(X), pmf(X). \quad \text{Normally penguins don’t fly} \]

 \[ pmf(X) \leftarrow \neg pmf(X) \quad \text{This is a default} \]

 \[ b(X) \leftarrow p(X). \quad \text{Penguins are birds} \]

 where \( p \) stands for penguin and \( pmf \) stands for penguin_not_flies.

 With this program, we consider two birds, one of them being a penguin,

 \( Facts_2 = \{ b(a), p(b) \} \). The WF Model of \( 2 \cup Facts_2 \) is:

 \[ WF(2 \cup Facts_2) = \]

 \[ \{ f(a), \neg f(a), bf(a), \neg bf(a), pmf(a), \neg pmf(a), b(a), \neg p(a), \]

 \[ bf(b), \neg bf(b), pmf(b), \neg pmf(b), b(b), p(b) \} \]

 About \( a \) everything is defined, and we have exactly the same case as

 before. About \( b \) this model only tells us that it is a bird and a penguin,

 leaving the predications flies and \( \neg flies \) undefined for it. This is due to

 the fact that there are two exception rules that might apply, each of which

 inhibits the conclusion of the other. So nothing can be said for sure about \( b \)

 flying or not.

 But the program has two more XS Models:

 \[ XSM_1 = WF \cup \{ f(b), \neg f(b) \} \]

 \[ XSM_2 = WF \cup \{ \neg f(b), f(b) \} \]

 These models can be seen as different models (i.e. consistent beliefs) an

 agent may have of the world. This is quite an intuitive result, since we can

 believe that \( b \) flies (because it is a bird) or that it doesn’t (because it is a

 penguin). The WF Model is the intersection of these two, and intuitively

 gives us our definite belief about the world, the things with are able to believe

 unconditionally. In WF nothing can be said definitely about \( b \) flying or

 not. This topic will be further discussed in the sequel.

 In a hierarchy, however, since we always want to apply the most specific

 information, there should be in this case an explicit preference between the

 rules bird_flies and penguin_not_flies, so as to establish that the property

 of flying is either true or false of any individual in the taxonomy.
By having rules named, this preference is quite easy to represent. What we wish to say is that if we can apply the rule penguin_flies to a penguin then we are in presence of an exception to the bird_flies rule. So the preference rule is simply:

$$\neg bf(X) \leftarrow p(X), pnf(X)^5.$$ 

With this rule the only XS Model is the WF Model:

$$WF = \{ f(a), \neg f(a), bf(a), \neg bf(a), pnf(a), \neg pnf(a), b(a), \neg p(a), 
\neg f(b), \neg f(b), \neg bf(b), \neg bf(b), \neg pnf(b), \neg pnf(b), b(b), p(b) \}$$

which expresses our intuitive notion about the hierarchy, b doesn’t fly and it is an exception to the bird_flies rule.

Next we can represent exceptions to exceptions:

$$Facts_3: \begin{align*}
&f(X) \leftarrow fp(X). \text{ Flying penguins definitely fly} \\
p(X) \leftarrow fp(X). \text{ Flying penguins are penguins} \\
f(c). \text{ c is a flying penguin}
\end{align*}$$

where $fp$ is short for $flying\_penguin$. The WF Model of $2 \cup Facts_3$ is:

$$WFM = \{ f(c), \neg f(c), bf(c), \neg bf(c), pnf(c), \neg pnf(c), b(c), p(c), fp(c) \}$$

Note $c$ is also an exception to the bird_flies rule since its a penguin. Nevertheless it flies, but because of the more specific and absolute rule that flying penguins fly.

3 Possible Worlds

In hierarchies, as seen, everything is defined, leaving no choices available. This is not the case for general hypothetical reasoning problems. In this section we represent hypothetical reasoning problems in the Well Founded Models Semantics and interpret the results.

3.1 The ”Nixon diamond” problem

| Normally quakers are pacifists. | Normally republicans are hawks. |
| Pacifists are non hawks. | Hawks are non pacifists. |
| Nixon is a quaker and a republican. | Pacifists are non hawks. |
| There are other republicans. | There are other quakers. |

Using the representation method of last section this example is expressed as:
\[
\text{pacifist}(X) \leftarrow \text{quaker}(X), \text{quaker}_{\text{pacifist}}(X),
\neg \text{pacifist}(X).
\]

\[
\text{quaker}_{\text{pacifist}}(X) \leftarrow \neg \text{quaker}_{\text{pacifist}}(X).
\]

Normally quakers are pacifists

\[
hawk(X) \leftarrow \text{republican}(X), \text{republican}_{\text{hawk}}(X),
\neg \text{hawk}(X).
\]

\[
\text{republican}_{\text{hawk}}(X) \leftarrow \neg \text{republican}_{\text{hawk}}(X).
\]

Normally republicans are hawks

\[
\neg \text{hawk}(X) \leftarrow \text{pacifist}(X). \quad \text{Pacifists are non hawks}
\]

\[
\neg \text{pacifist}(X) \leftarrow \text{hawk}(X). \quad \text{Hawks are non pacifists}
\]

\[
\text{quaker}(\text{nixon}). \quad \text{Nixon is quaker.}
\]

\[
\text{republican}(\text{nixon}). \quad \text{Nixon is republican.}
\]

\[
\text{quaker}(\text{other-quaker}). \quad \text{There are other quakers.}
\]

\[
\text{republican}(\text{other-republican}). \quad \text{There are other republicans.}
\]

Here there is no preference defined between the rules nor between their conclusions. So for nixon we want to be able to hypothesize him to be a pacifist or a hawk. Let us have a closer look at the extended stable models of this program, and interpret them carefully. The \textit{WFM} is:

\[
\{\text{qua}(n), \text{rep}(n), \text{qp}(n), \text{rh}(n),
\text{qua}(o,q), \text{qp}(o,q), \text{rh}(o,q), \text{pac}(o,q), \neg \text{hawk}(o,q), \neg \text{pac}(o,q), \neg \text{hawk}(o,q), \text{rep}(o,r), \text{qp}(o,r), \text{rh}(o,r), \text{hawk}(o,r), \neg \text{pac}(o,r), \neg \text{hawk}(o,r), \text{pac}(o,r)\}
\]

For nixon, as expected, it is unknown whether he is a pacifist or a hawk. Nevertheless we have \text{quaker}_{\text{pacifist}}(\text{nixon}) and \text{republican}_{\text{hawk}}(\text{nixon}), so that both rules applied. This is not a strange result since rules are maximally applicable, and nixon does not consist an exception to the rules through to their conclusions. Of course, for other quakers and other republicans everything is defined.

Since we have unknown literals not present in the WF Model we might have other extended stable models. In this case the other XS Models are:

\[
\text{XSM}_1 = \text{WFM} \cup \{\text{pac}(n), \neg \text{hawk}(n)\}
\]

\[
\text{XSM}_2 = \text{WFM} \cup \{\neg \text{pac}(n), \text{hawk}(n)\}
\]

These remaining XS Models can be seen as possible extended world views in which some consistent choices of belief have been made. \text{XSM}_1 represents a world view where nixon is a pacifist and a non hawk, and \text{XSM}_2 represents a world view where nixon is a hawk and a non pacifist.
3.2 Unknown application of rules

In the last example, exceptions are not present on the application of rules, but only on their conclusions, and so the rules were maximally applied. But in some cases we would like to choose between applying or not some rule, by keeping the rule name unknown in the WF Model.

Above, a rule of the form \textit{name of rule}(X) \leftarrow \neg \textit{name of rule}(X) expresses that the rule is true if possible. To have the possibility of applying or not the rule, we add a clause stating that the rule name is not true if possible, gaining this way freedom to apply or not the rule. The additional clause is clearly:

\[
\neg \textit{name of rule}(X) \leftarrow \textit{name of rule}(X).
\]

With this formulation rule names, and consequently their conclusions, become unknown in the WF Model, if nothing else is said about their name. XS Models appear: ones with the rule name being true, and others with it being false.

Furthermore, one might want to have the possibility of applying a rule or not, but only under certain conditions, and applying it maximally otherwise. As an example suppose:

Normally quakers are pacifists. (1) Nixon is a quaker.
We are not sure if (1) applies to nixon. (2) There other quakers.

Because of (2), we wish to choose between having or not rule (1) applied to nixon, but to apply it maximally to other individuals. Given (2), rule (1) becomes a rule with a possibly undecided applicability.

Accordingly, our representation of this problem is:

\[
\begin{align*}
pacifist(X) & \leftarrow quaker(X), quaker \cdot pacifist(X), \\
quaker \cdot pacifist(X) & \leftarrow \neg quaker \cdot pacifist(X), \\

\neg quaker \cdot pacifist(nixon) & \leftarrow \neg quaker \cdot pacifist(nixon), \\
quaker(nixon) & \leftarrow \neg quaker \cdot pacifist(nixon), \\
quaker(other \cdot quaker) & \leftarrow \text{There are other quakers.}
\end{align*}
\]

Rule (1) might not apply to nixon

The WF Model of this program is:

\[
WF = \{ quaker(nixon), quaker(other \cdot quaker), \\
pacifist(other \cdot quaker), \neg pacifist(other \cdot quaker) \}
\]

and its consistent XS Models are:

\[
\begin{align*}
XSM_1 & = WF \cup \{ pacifist(nixon), quaker \cdot pacifist(nixon) \} \\
XSM_2 & = WF \cup \{ \neg quaker \cdot pacifist(nixon) \}
\end{align*}
\]
In the WF Model we are not able to conclude anything about nixon being a pacifist because we are not able to apply the rule.

$XSM_1$ represents a world where nixon is a pacifist, by choosing to apply the rule $\text{quaker}_a \rightarrow \text{pacifist}_a$ via its name being true; $XSM_2$ is a world where we choose not to apply the corresponding rule, via its name being false, so that we can prove nothing about nixon being a pacifist.

The interpretation of the above results, in particular in what concerns XSMs where literals not in the WF Model become true, suggested to us a relationship between the XS Models of a program with rules like (1) and the concept of abduction within WF Semantics [9].

3.3 Unknown possible facts

Similarly to rules about which we are undecided regarding their applicability, we might be unsure about some facts. Note that this is different from not having any knowledge at all about such a fact. Consider this simple example:

John and Nixon are quakers. John is a pacifist.

represented by the program:

$$\Pi = \{\text{quaker}(\text{john}), \text{quaker}(\text{nixon}), \text{pacifist}(\text{john})\}.$$  

The $WFM$ (which is the only XS Model) is:

$$\{\text{quaker}(\text{john}), \text{quaker}(\text{nixon}), \text{pacifist}(\text{john}), \neg\text{pacifist}(\text{nixon})\}$$

and expresses exactly what is intended, i.e. john and nixon are quakers, john is a pacifist and we don’t have reason to believe nixon is a pacifist. Now suppose we add: Nixon might be a pacifist (3). In our view, we wouldn’t want in this case to be so strong as to affirm $\neg\text{pacifist}(\text{nixon})$, thereby not allowing for the possibility of nixon being a pacifist. What we are prepared to say is that Nixon might be a pacifist if we don’t have reason to believe he isn’t and, vice-versa, that Nixon might be a non pacifist if we don’t have reason to believe he isn’t one. This is clearly expressed as:

$$\text{pacifist}(\text{nixon}) \leftarrow \neg\text{pacifist}(\text{nixon})$$

$$\neg\text{pacifist}(\text{nixon}) \leftarrow \neg\text{pacifist}(\text{nixon})$$

Program together with these rules has

$$WF = \{\text{quaker}(\text{john}), \text{quaker}(\text{nixon}), \text{pacifist}(\text{john}), \neg\text{pacifist}(\text{nixon})\},$$

and two more XS Models:

$$XSM_1 = WF \cup \{\text{pacifist}(\text{nixon}), \neg\text{pacifist}(\text{nixon})\}$$

$$XSM_2 = WF \cup \{\neg\text{pacifist}(\text{nixon}), \text{pacifist}(\text{nixon})\}$$

which is the result we were seeking. Facts like (3) are what we call unknown possible facts.
4 Summary of our representation method

In this section we summarize and systematize the representation method adopted in all the above examples. The type of rules for which we propose a representation is, in our view, general enough to capture a wide domain of nonmonotonic problems. Each type of rule is described in a subsection by means of a schema in natural language and its corresponding representation schema.

- **Definite Rules** If \( A \) then \( B \). The representation is: \( B \leftarrow A \).

- **Definite Facts** \( A \) is true. The representation is: \( A \).

- **Default (or maximally applicable) Rules** Normally if \( A \) then \( B \). The representation is:
  
  \[
  B \leftarrow A, \text{name}\_A\_B, \neg B, \\
  \text{name}\_A\_B \leftarrow \neg \text{name}\_A\_B.
  \]

  where \( \text{name}\_A\_B \) is a predicate symbol that "names" this rule only. Its arguments are those arguments in \( A \) or \( B \). We will consider Default Facts as a special case of Default Rules where \( A \) is absent. As an example consider the rule "Normally birds fly". Its representation is:

  \[
  \text{fly}(X) \leftarrow \text{bird}(X), \text{bird}\_flies(X), \neg \text{fly}(X), \\
  \text{bird}\_flies(X) \leftarrow \neg \text{bird}\_flies(X).
  \]

- **Possible Rules** Rule "If \( A \) then \( B \)" may or may not apply. Its representation is:

  \[
  B \leftarrow A, \text{name}\_A\_B, \neg B, \\
  \text{name}\_A\_B \leftarrow \neg \text{name}\_A\_B, \\
  \neg \text{name}\_A\_B \leftarrow \neg \text{name}\_A\_B.
  \]

  where \( \text{name}\_A\_B \) is a predicate symbol that "names" the first rule only. Its arguments are all those arguments in \( A \) or \( B \). As an example consider the rule "Quakers might be pacifists". Its representation is:

  \[
  \text{pacifist}(X) \leftarrow \text{quaker}(X), \text{quaker}\_\text{pacifist}(X), \neg \text{pacifist}(X), \\
  \neg \text{quaker}\_\text{pacifist}(X) \leftarrow \neg \neg \text{quaker}\_\text{pacifist}(X), \\
  \neg \text{quaker}\_\text{pacifist}(X) \leftarrow \neg \text{quaker}\_\text{pacifist}(X).
  \]

- **Exceptions to Default Rule** Under certain conditions \( COND \) there are exceptions to the default rule named \( NAME \).

  \[
  \neg NAME \leftarrow COND.
  \]

  As an example, the representation of the exception "Penguins are exceptions to the "normally birds fly" rule" is:

  \[
  \neg \text{bird}\_flies(X) \leftarrow \text{penguin}(X).
  \]
• **Possible Exceptions to Default Rule** Under certain conditions COND there might be exceptions to the default rule named NAME.

\[-\text{NAME} \leftarrow \text{COND, } \neg\text{NAME}.\]

As an example, the representation of the exception "Nixon is a possible exception to the "normally quakers are pacifists" rule" is:

\[-\text{quaker.pacifist}(\text{nixon}) \leftarrow \neg\text{quaker.pacifist}(\text{nixon}).\]

• **Preference Rules** Under conditions COND, prefer to apply the default rule \text{NAME}+ instead of the default rule named \text{NAME}−.

\[-\text{NAME}− \leftarrow \text{COND, } \text{NAME}+.\]

As an example consider "For penguins, if the rule that says that "normally penguins don't fly" is applicable then inhibit the "normally birds fly" rule". This is represented as:

\[-\text{bird.flies}(\text{X}) \leftarrow \text{penguin}(\text{X}), \text{penguins.no.flies}(\text{X}).\]

• **Unknown Possible Fact** F might be true or not (in other words, the possibility of F should be considered).

\[
\begin{align*}
\text{F} & \leftarrow \neg\neg\text{F}. \\
\neg\text{F} & \leftarrow \neg\text{F}.
\end{align*}
\]

5 **Application To Other Problems**

We now apply the programing method described above to some other problems. We will not argue that this method is the best for representing these problems. Instead, we want to show that it gives the right results, so being a general method to represent nonmonotonic problems.

5.1 **The Yale Shooting Problem**

This problem, supplied in [3], will be represented in a form near to the one suggested in [4]. Predicate \text{holds}(P, S) expresses that property P holds in situation S; predicate \text{normal}(P, E, S) expresses that in situation S, event E does not normally affect the truth value of property P; the term \text{result}(E, S) names the situation resulting from the occurrence of event E in situation S.

The problem and its formulation are as follows:

Initially (in situation s0) a person is alive: \text{holds(alive, s0)}.

After loading a gun the gun is loaded: \text{holds(loaded, result(load, S))}.

If the gun is loaded, then after shooting it the person will not be alive. Note that the latter is not a normally-rule:

\[-\text{holds(alive, result(shoot, S))} \leftarrow \text{holds(loaded, S)}.\]
After an event things normally remain as they were, i.e., properties which hold before will normally still hold after the event; likewise, properties which do not hold before the event will normally not hold afterwards as well. Here predicate $\text{normal}(P, E, S)$ will be used to name both these rules:

$$\text{holds}(P, \text{result}(E, S)) \leftarrow \quad \text{holds}(P, S), \text{normal}(P, E, S), \neg \text{holds}(P, \text{result}(E, S)) \quad (\text{pp})^{11}$$

$$\neg \text{holds}(P, \text{result}(E, S)) \leftarrow \quad \neg \text{holds}(P, S), \text{normal}(\neg P, E, S), \neg \text{holds}(P, \text{result}(E, S)) \quad (\text{np})^{12}$$

$$\text{normal}(P, E, S) \leftarrow \quad \neg \text{normal}(P, E, S)$$

In principle we should use different names for each of the rules. In fact $\text{normal}(\neg P, E, S)$ is just a syntactic gimmick to avoid a new name predicate for (np), and thus avoid an additional rule for it.

Consider the question "What holds and what doesn't hold after the loading of a gun, a period of waiting and a shooting?" represented as two queries:

$$\leftarrow \text{holds}(P, \text{result}(\text{shoot, result(wait, result(load, s0))))}$$

$$\leftarrow \neg \text{holds}(P, \text{result}(\text{shoot, result(wait, result(load, s0))))}$$

With this formulation the WF Model is the only XS Model. The subset of its elements that match with at least one of the queries is$^{13}$:

$$\{\text{holds}(\text{loaded, s3}), \neg \text{holds}(\text{loaded, s3}), \neg \text{holds}(\text{alive, s3}), \neg \text{holds}(\text{alive, s3})\}$$

which means that in situation s3 the gun is loaded and the person is not alive. This result coincides with the one obtained in [4] for holds.

To get the result given by circumscription [6] and default logic [14], we must reformulate the problem by adding the sentence: the wait event might not preserve the persistence of the loaded property; in other words, after a wait event the gun might be unloaded. This clearly requires an unknown application of (pp). So the rule to add is:

$$\neg \text{normal}(\text{loaded, wait, S}) \leftarrow \neg \text{normal}(\text{loaded, wait, S})$$

Now the subset of the WF Model is $\{\neg \text{holds}(\text{loaded, s3})\}$. This means that in the WF Model we have no proof that the gun is not loaded. This is acceptable because there is no evidence for it to be unloaded. All other properties are unknown in the WF model. But we have two XS Models, corresponding to the two default extensions. Their subsets of elements that match a query are:

$$\{\text{holds}(\text{alive, s3}), \neg \text{holds}(\text{alive, s3}), \neg \text{holds}(\text{loaded, s3})\}$$

$$\{\neg \text{holds}(\text{alive, s3}), \neg \text{holds}(\text{alive, s3}), \text{holds}(\text{loaded, s3}), \neg \text{holds}(\text{loaded, s3})\}$$

We tested this example with other modifications, for example (1) Considering that having a gun loaded after a wait action is an exception to the (pp) rule. (2) In the initial situation the gun might be loaded.

In applications to some other problems, in an extended version of this paper, our representation method also gives quite intuitive results.
6 Discussion and open work

Here we raise points of discussion, including comparisons with other work.

6.1 Using contrapositives of default rules

One might wish to have contrapositive variants of a default rule. For example, one might want it holds to, given the "normally birds fly" rule, that an individual which doesn’t fly, is normally not a bird.

Given the "birds fly" rule:

\[ flies(X) \leftarrow bird(X), \overline{bird\_flies(X)}, \overline{flies(X)} \ (c1) \]

its sole contrapositive is:

\[ \overline{bird(X)} \leftarrow \overline{flies(X)}, \overline{bird\_flies(X)}, \overline{bird(X)} \ (c2) \]

This has the disadvantage of using the same rule name in both, and being somewhat repetitive. Syntactically, one could write rules allowing contrapositives in a distinct form, for example: \((flies(X) \leftarrow bird(X)) \leftarrow bird\_flies(X)\), which of course would stand for \((c1)\) and \((c2)\), and where \(\leftarrow\) stands for material implication.

Note that the last literal of each of the above rules, is needed only to avoid contradictory well founded models.

For example, consider the program \(P = \{f \leftarrow b, bf \ leftarrow \overline{bf} \ b \ \overline{f}\}\) not having such literals. This program has a contradictory WF Model. However, adding \(\overline{f}\) to the body of the first clause, leads to a noncontradictory WF Model containing \(\overline{f}\). The effect of those literals in the WF Model can be seen as a preference for the use of definite rules over default ones. The next point of discussion suggests a modification to the semantics in order to have that preference without the need for those literals.

6.2 Avoiding contradiction in the WF Model

A literal assumed false\(^{14}\) in the WF Model (Closed World Assumption (CWA) ), can lead to a contradictory WF Model when classical negation is present. If an assumption leads to a contradiction it seems natural to be able to go back on that assumption.

A suggestion is to build a new class of models, defined as being the noncontradictory models, resulting from allowing some literals assumed false by CWA in the WF Model to have the unknown truth value, just in case the WF Model is of contradictory. All these models would be smaller, by Fitting's ordering [1], than the WF Model. For this reason we call them Sub Well Founded Models (SWF Models)[8]. The maximal SWF Models can also provide Sub XSM models.

Note that for programs having a contradiction derived from definite rules or facts (Ex: \(\{p, \overline{p}\}\)) there are no SWF Models. As these models remove
contradiction by changing the truth value only on NAF negated literals, and those literals never appear in definite rules, they reflect preference on the use of these rules over the use of default ones, as proposed above.

6.3 Comparison with ”Logic Programs With Exceptions”[4]

We deal with exceptions to exceptions in a uniform and modular way. Because of its inherent asymmetry, the ”rules with exceptions” approach requires changing previous rules in the program each time an exception to an exception is made, because head literals need to change. For instance, a three level hierarchy of birds, penguins and flying penguins requires rules like

\[
\begin{align*}
\text{fly}(X) & \leftarrow \text{bird}(X) \\
\text{nofly}(X) & \leftarrow \text{penguin}(X) \\
\text{fly}(X) & \leftarrow \text{flying penguin}(X)
\end{align*}
\]

and the exceptions:

\[
\begin{align*}
-\text{fly}(X) & \leftarrow \text{nofly}(X) \\
-\text{nofly}(X) & \leftarrow \text{flying penguin}(X)
\end{align*}
\]

We allow both positive and negative conclusions in rules, inclusively for the same predicate.

The extension of well founded semantics to classical negation provides a (non contradictory) well founded model of definite conclusions in cases where e-answer set semantics provides only alternative models. For instance, in the pacifist/hawk example we obtain a well founded model containing the facts {quaker, republican}, besides the two alternative e-answer sets. By introducing ”name predicates” in rules, we have shown how to express exceptions to rules, rather than exceptions to whole predicates, and how to express preference amongst rules within the language.

6.4 Comparison with ”Answer-set Semantics”[2]

The three-valued semantics subsumes the two-valued semantics of answer-set semantics. The techniques for nonmonotonic reasoning we’ve presented are in part adoptable by the answer-set semantics of extended logic programs.

6.5 Comparison with ”Poole’s naming device”[10]

We deal with preference between rules within the language. Poole does this using his constraints. These constraints are not in the language of logic programs, and are in fact a meta-level model-theoretic construct outside it.

The use of formulas by Poole, instead of rules, includes all contrapositives even if not wanted. Again the pruning effect can only be achieved by the use of constraints. In our method we have only the contrapositives effectively written. If all are wanted all can be included.
Acknowledgements

We thank ESPRIT Basic Research Project COMPULOG (no. 3012), Instituto Nacional de Investigação Científica and Gabinete de Filosofia do Conhecimento for their support. Thanks to Fariba Sadri and Tony Kakas for fruitful discussions.

Notes

1This rule is similar to Reiter's normal defaults [14], where bird(X) is the precondition.

2The symbol ~ stands for the negation as failure operator. ~ stands for classical negation as in [2].

3This is akin to Poole's naming device [10], but used for rules instead of formulas.

4Note that all predicates and their classical negations are defined.

5Poole [10] does this using his constraints, a meta-level model-theoretic construct outside the language of logic programs.

6other_quaker and other_republican can be envisaged as Skolem constants.

7Where qua stands for quaker, rep for republican, pac for pacifist, qf for quaker, pacifist, rh for republican, hawk, n for nixon, or for other_quaker and or for other_republican.

8As these two models are 2-valued we don't show the additional ~ literals, which are implicit.

9Again, as the two models are 2-valued we don't present the additional ~ literals.

10Note that this model contains ~ pacifist(nixon) and ~ pacifist(nixon).

11pp stands for positive persistence.

12np stands for negative persistence.

13Where s stands the term result(shoot, result(wait, result(load,s0))).

14A literal is assumed false in the WFM if it appears under ~there.

References


