Logic Programming for Non–monotonic Reasoning

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

January 25, 1993

Abstract

Our purpose is to exhibit a modular systematic method of representing nonmonotonic reasoning problems with the Well Founded Semantics of extended logic programs aumented with eXplicit negation ($\mathcal{WFSX}$), augmented by its Contradiction Removal Semantics ($\mathcal{CRSX}$) when needed. We show how to cast in the language of such logic programs forms of non-monotonic reasoning like defeasible reasoning and hypothetical reasoning and apply them to different domains of knowledge representation, for instance hierarchies and reasoning about actions. We then abstract a modular systematic method of representing non-monotonic problems in logic programming.

1 Introduction

Recently, several authors have stressed and showed the importance of introducing an explicit second kind of negation within logic programs, for use in deductive databases, knowledge representation, and nonmonotonic reasoning [2, 4, 6, 9, 8, 17, 18, 19, 25, 22].

It has been argued [18, 19, 17, 16] that semantics with the well founded property are adequate to capture nonmonotonic reasoning if we interpret the least model provided by the semantics (called the Well Founded Model of a program) as the skeptical view of the world, and the other models (called Extended Stable Models) as alternative enlarged consistent belief sets. A consequence of the well founded property is that intersection of all models identified by the semantics is itself a model belonging to the semantics.

Some proposals for extending logic programming semantics with a second kind of negation has been advanced. One such extension is the Answer Set semantics (AS)[4], which is shown to be an extension of Stable Model ($\mathcal{SM}$) semantics [3] from the class of logic programs [11] to those with a second form of negation. In [9] another proposal for such extension is introduced, based on the $\mathcal{SM}$ semantics, where implicitly a preference between negative information (exceptions) over positive information is assumed. However AS semantics is not well founded. The meaning of the program is defined as the intersection of all answer-sets, and it is known that the computation of this intersection is computationally expensive. Another extension to include a second kind of negation is suggested by Przymusinski in [21]. Although the set of models identified by this extension enjoys the well founded property, it gives some less intuitive results [1] with respect to the coexistence of both forms of negation. Based on the $\mathcal{XSM}$ semantics, Przymusinski [22] also introduces the Stationary semantics where the second form of negation is classical negation. But, classical negation entails that the logic programs under Stationary semantics no longer admit a procedural reading.

On the other hand, Well Founded Semantics with Explicit Negation ($\mathcal{WFSX}$) [13], which we prefer, is an extension to Well Founded Semantics [24] including a second form of negation called *explicit negation*, preserving the well founded property (cf. [1] for a comparison of the above approaches) and procedural reading.

When a second form of negation is introduced contradiction may be present (i.e. l and $\neg l$ hold for some l) and no semantics is given by $\mathcal{WFSX}$[1]. In [14] (also submitted to this conference) the authors define $\mathcal{CRSX}$ extending $\mathcal{WFSX}$ by introducing the notion of removing some contradictions and identifying the models obtained by revising closed world assumptions supporting those contradictions. One unique model, if any such revised model exists, is singled out as the contradiction free semantics. When no contradiction is present $\mathcal{CRSX}$ semantics reduces to $\mathcal{WFSX}$ semantics.

Here, using $\mathcal{CRSX}$ (which is assumed [14]), we show how to cast in the language of logic programs extended with explicit negation different forms of non-monotonic reasoning such as defeasible reasoning and hypothetical reasoning, and apply it to diverse domains of knowledge representation such as hierarchies and reasoning about actions.

Our final purpose is to abstract out a modular systematic method of representing some nonmonotonic reasoning problems with the $\mathcal{CRSX}$ semantics of logic programs.

2 Defeasible Reasoning

In this section we show how to represent defeasible reasoning with logic programs extended with explicit negation. We want to express defeasible reasoning and give a meaning to sets of rules (some of them being defeasible) when contradiction arises from the application of the defeasible rules. For instance, we want to represent defeasible rules such as *birds normally fly* and *penguins normally don't fly*. Given a penguin, which is a bird, we adopt the skeptical point of view and none of the conflicting rules applies. Later on we show how to express preference for one rule over another in case they conflict and both are applicable. Consider for the moment a simpler version of this problem:

Example 1 Consider the statements:

(i) *Normally birds fly.* (ii) *Penguins don't fly.* (iii) *Penguins are birds.* (iv) *a is a penguin.*

represented by the program P (with obvious abreviations, where ab stands for abnormal [2]):

$$\begin{array}{rclcrclc} f(X) & \leftarrow & b(X), \sim ab(X) & \text{(i)} & b(X) & \leftarrow & p(X) & \text{(iii)} \\ \neg f(X) & \leftarrow & p(X) & \text{(ii)} & p(a) & & & \text{(iv)} \end{array}$$

Since there are no rules for $ab(a)$, $\sim ab(a)$ holds and $f(a)$ follows. On the other hand we have $p(a)$ and $\neg f(a)$ follows from rule (ii). Thus the program model $\mathcal{M}_P$ is contradictory. In this case we argue that the first rule gives rise to a contradiction depending on a CWA on $ab(a)$ and so must not conclude $f(a)$. The intended meaning requires $\neg f(a)$ and $\sim f(a)$. We say that in this case a revision occcurs in the CWA of predicate instance $ab(a)$, which must turn to undefined. $\sim f(a)$ follows from $\neg f(a)$ in the semantics.

In this case $\mathcal{CRSX}$ identifies one contradiction removal set $CRS = \{\sim ab(a)\}$. The corresponding contradiction free program is $P \cup \{ab(a) \leftarrow \sim ab(a)\}$, and the corresponding model is $CRWFM$ is $\{p(a), \sim \neg p(a), b(a), \sim \neg b(a), \neg f(a), \sim f(a), \sim \neg ab(a)\}$.

In the example above the revision process is simple and the information to be revised is clearly the CWA about the abnormality predicate, and something can be said about a flying. However this is not always the case, as shown in the following example:

Example 2 Consider the statements represented by P below:

(i) *Normally birds fly.* (ii) *Normally penguins don't fly.* (iii) *Penguins are birds.*

[1] In [15, 16] it is shown how $\mathcal{WFSX}$ relates to default theory.
[2] $\sim$is used for negation instead of the more usual *not* for expressing implicit or default negation (cf. failure to prove).

$$f(X) \leftarrow b(X), \sim ab_1(X)(\mathrm{i}) \qquad \neg f(X) \leftarrow p(X), \sim ab_2(X)(\mathrm{ii}) \qquad b(X) \leftarrow p(X)(\mathrm{iii})$$

Consider a penguin a, a bird b, and a rabbit c which does not fly; i.e. facts $F = \{p(a), b(b), r(c), \neg f(c)\}$.

Remark 2.1 *Facts F above and rule (iii) in P play the rôle of non–defeasible information, and should hold whichever the world view one may choose for the interpretation of P together with those facts.*

- About the bird b everything is well defined and we have:

  $$\{ \ \sim ab_1(b), b(b), f(b), \sim\neg b(b), \sim\neg f(b), \sim\neg ab_1(b), \sim\neg ab_2(b), \sim ab_2(b), \sim p(b), \sim\neg p(b) \ \}$$

  which says that bird b flies, $f(b)$, and it can't be shown it is a penguin, $\sim p(b)$. This is the intuitive result, since we may believe that b flies (because it is a bird) and it is not known to be a penguin, and so rules (i) and (ii) are non–contradictory w.r.t. to bird b.

- About the penguin a use of rules (i) and (ii) provoke a contradiction in $\mathcal{M}_P$: by rule (i) we have $f(a)$ and by rule (ii) we have $\neg f(a)$. Thus nothing can be said for sure about a flying or not, and the only non–ambiguous conclusions we may infer are:

  $$\{ \ p(a), b(a), \sim\neg p(a), \sim\neg b(a), \sim\neg ab_1(a), \sim\neg ab_2(a) \ \}$$

  Note that we are being skeptical w.r.t. to $ab_1(a)$ and $ab_2(a)$, whose negation by CWA would give rise to a contradiction.

- About c rules (i) and (ii) are non-contradiction producing since $\sim p(c)$ and $\sim b(c)$ both hold, and we have:

  $$\{ \ r(c), \neg f(c), \sim p(c), \sim b(c), \sim\neg r(c), \sim f(c), \sim\neg p(c), \sim\neg b(c), \sim\neg ab_1(c), \sim\neg ab_2(c), \sim ab_1(c), \sim ab_2(c) \ \}$$

The view of the world given by the least p–model of $P' = P \cup F$ using $\mathcal{WFSX}$ is[3]:

$$\begin{aligned}\mathcal{M}_{P'} = \{ \quad & p(a), b(a), \sim\neg p(a), \sim\neg b(a), \sim\neg ab_1(a), \sim\neg ab_2(a), \sim ab_2(a), \sim\neg f(a), \sim f(a), \neg f(a), \sim ab_1(a), f(a), \\ & b(b), \sim\neg b(b), \sim\neg p(b), \sim\neg ab_1(b), \sim\neg ab_2(b), \sim ab_1(b), \sim ab_2(b), \\ & \neg f(c), \sim f(c), \sim p(c), \sim\neg p(c), \sim\neg b(c), \sim\neg f(c), \sim b(c), \sim\neg ab_1(c), \sim\neg ab_2(c), \sim ab_1(c), \sim ab_2(c)\}\end{aligned}$$

A contradiction arises about penguin a ($f(a)$ and $\neg f(a)$ both hold) because of the (closed world) assumptions on $ab_1(a)$ and $ab_2(a)$, thus suggesting that the contradiction removal set is $CRS = \{\sim ab_1(a), \sim ab_2(a)\}$. Using $\mathcal{CRSX}$ [4] we can determine formally the CRS above.

2.1 Exceptions

In general we may want to say that a given element is an exception to a normality rule. The notion of exception may be expressed in two different ways.

[3]Note that the difference between the $\mathcal{M}'_P$ model presented and the set of literals considered as the intuitive result in the previous remark differ precisely in the truth valuation of predicate instances $ab_1(a)$, $ab_2(a)$ and $f(a)$.

[4]apa92:crsx

Exceptions to predicates

Example 3 We express that the rule $flies(X) \leftarrow bird(X)$ applies whenever possible but can be defeated by exceptions using the rule:

$$flies(X) \leftarrow bird(X), {\sim}ab(X) \tag{1}$$

If there is a bird b and a bird a which is known not to fly (and we don't know the reason why) we may express it by $\neg flies(a)$. In this case $\neg flies(a)$ establishes an *exception to the conclusion predicate* of the defeasible rule, and the meaning of the program[5] is:

$$\begin{array}{ll} bird(b), {\sim}ab(b), & {\sim}\neg ab(b), {\sim}\neg bird(b), {\sim}\neg flies(b), flies(b) \\ bird(a), & {\sim}\neg ab(a), {\sim}\neg bird(a), \neg flies(a), {\sim}flies(a) \end{array}$$

Note that nothing is said about $ab(a)$, i.e. the CWA on $ab(a)$ is avoided ($\{{\sim}ab(a)\}$ is the CRS) since it would give rise to a contradiction on $flies(a)$. This is the case where we know that bird a is an exception to the *normally birds fly* rule, by observation of the fact that it does not fly: $\neg flies(a)$.

Exceptions to rules

A different way to express that a given animal is some exception is to say that a given rule must not be applicable to the animal. To state that an element is an exception to a specific rule rather than to its conclusion predicate (more than one rule may have the same conclusion), we state that the element is abnormal w.r.t. the rule, i.e. the rule is not applicable to the element: if element a is an exception to the flying birds rule we express it as $ab(a)$.

In general we may want to express that a given X is abnormal under certain conditions. This is the case where we want to express penguins are abnormal w.r.t. to the flying birds rule above, as follows:

$$ab(X) \leftarrow penguin(X) \tag{2}$$

Rule (2) together with the non–defeasible rule $bird(X) \leftarrow penguin(X)$ add that *penguins are birds which are abnormal w.r.t. to flying.* Similarly of dead birds; i.e. $ab(X) \leftarrow bird(X), dead(X)$ adding that *dead birds are abnormal w.r.t. to flying.* Alternatively, given $\neg flies(X) \leftarrow dead(X)$, the non–abnormality of dead bird a w.r.t. to flying, i.e. ${\sim}ab(a)$, may not be consistently assumed since it leads to a contradiction regarding $flies(a)$ and $\neg flies(a)$.

A sstronger form of exception may be used to state that any element of type, say penguin, X is considered an exception unless one knows it explicitly not to be one: $ab(X) \leftarrow {\sim}\neg penguin(X)$. One cannot apply defeasible rule (1) unless X is known not to be a penguin.

2.2 Exceptions to exceptions

In general we may extend the notion of exceptioned rules to exception rules themselves, i.e. exception rules may be defeasible. This will allow us to express an exception to the exception rule for birds to fly, and hence the possibility that an exceptional penguin may fly, or that a dead bird may fly. In this case we want to say that the exception rule is itself a defeasible rule:

$$ab(X) \leftarrow bird(X), dead(X), {\sim}ab\_deadbird(X)$$

2.3 Preferences among rules

We may express now preference between two rules, stating that if one rule may be used, that constitutes an exception to the use of the other rule:

[5]This is a simplified version of example 1.

Example 2 **(cont.)**

$$f(X) \leftarrow b(X), \sim ab_1(X) \text{ (i)} \qquad \neg f(X) \leftarrow p(X), \sim ab_2(X) \text{ (ii)} \qquad b(X) \leftarrow p(X) \text{ (iii)}$$

In some cases we want to apply the most specific information; above, there should be (since a penguin is a specific kind of bird) an explicit preference of the non–flying penguins rule over the flying birds rule:

$$ab_1(X) \leftarrow p(X), \sim ab_2(X) \tag{3}$$

If we have also $penguin(a)$ and $bird(b)$ the unique model is:

$$\{\sim ab_2(b)\ b(b)\ \ \sim p(b)\ \ \sim\neg f(b)\ \ \sim ab_1(b)\ f(b)\ \ \sim ab_2(a)\ p(a)\ b(a)\ ab_1(a)\ \ \sim f(a)\}$$

Rule (3) says that if a given penguin is not abnormal w.r.t. to non–flying then it must be considered abnormal w.r.t. flying. So we infer that b is a flying bird, a being a penguin is also a bird, and that there is no evidence (assume it false) that it flies $\sim f(a)$.

3 Representation of Hierarchical Taxonomies

In this section we illustrate how to represent taxonomies with logic programs with explicit negation. In this representation we wish to express general absolute (i.e. non–defeasible) rules, defeasible rules, exceptions to defeasible rules, as well as exceptions to exceptions, explicitly making preferences among defeasible rules. As we've seen, when defeasible rules contradict each other and no preference rule is present, none of them is considered applicable in the most skeptical reading. We want to be able to express preference for one defeasible rule over another whenever they conflict. In taxonomic hierarchies we wish to express that in the presence of contradictory defeasible rules we prefer the one with most specific information (e.g. for a penguin, which is a bird, we want to conclude that it doesn't fly).

Example 4 The statements, facts and preferences about the domain are:

(1)	Mammals are animals.	(6)	Normally animals don't fly.
(2)	Bats are mammals.	(7)	Normally bats fly.
(3)	Birds are animals.	(8)	Normally birds fly.
(4)	Penguins are birds.	(9)	Normally penguins don't fly.
(5)	Dead animals are animals.	(10)	Normally dead animals don't fly.

(11)	Pluto is a mammal.	(12)	Tweety is a bird.
(13)	Joe is a penguin.	(14)	Dracula is a bat.
(15)	Dracula is a dead animal.		

(16) Dead bats do not fly though bats do.
(17) Dead birds do not fly though birds do.
(18) Dracula is an exception to the above preferences.

Our representation of the hierarchy is the program:

$$\begin{array}{rll}
animal(X) & \leftarrow mammal(X) & (1) \\
mammal(X) & \leftarrow bat(X) & (2) \\
animal(X) & \leftarrow bird(X) & (3) \\
bird(X) & \leftarrow penguin(X) & (4) \\
animal(X) & \leftarrow dead_animal(X) & (5) \\
\neg flies(X) & \leftarrow animal(X), \sim ab_1(X) & (6) \\
flies(X) & \leftarrow bat(X), \sim ab_2(X) & (7) \\
flies(X) & \leftarrow bird(X), \sim ab_3(X) & (8) \\
\neg flies(X) & \leftarrow penguin(X), \sim ab_4(X) & (9) \\
\neg flies(X) & \leftarrow dead_animal(X), \sim ab_5(X) & (10)
\end{array}
\qquad
\begin{array}{rl}
mammal(pluto) & (11) \\
bird(tweety) & (12) \\
penguin(joe) & (13) \\
bat(dracula) & (14) \\
dead_animal(dracula) & (15)
\end{array}$$

with the implicit hierarchical preference rules because of greater specifity:

$$ab_1(X) \leftarrow bat(X), \sim ab_2(X) \qquad ab_1(X) \leftarrow bird(X), \sim ab_3(X) \qquad ab_3(X) \leftarrow penguin(X), \sim ab_4(X)$$

and the explicit problem statement preferences:

$$\begin{array}{ll} ab_2(X) \leftarrow dead\_animal(X), bat(X), \sim ab_5(X) & (16) \\ ab_3(X) \leftarrow dead\_animal(X), bird(X), \sim ab_5(X) & (17) \\ ab_5(dracula) & (18) \end{array}$$

As expected this program has exactly one model (coinciding with the minimal $\mathcal{WFSX}$ model) which is non–contradictory, no choices being possible and everything being defined in the hierarchy.

Thus pluto doesn't fly, and isn't an exception to any of the rules; tweety flies because it's a bird and an exception to the "animals don't fly" rule; joe doesn't fly because it's a penguin and an exception to the "birds fly" rule.

Note that although dracula is a dead animal, which by default don't fly (cf. rule (10)) it is also considered an exception to this very same rule. Furthermore rule (16) saying that "dead bats normally do not fly" is also exceptioned by dracula and thus the "bats fly" rule applies and dracula flies. Note that preferences rules must be present in order to prevent contradiction from arising.

4 Hypothetical Reasoning

In hierarchies everything is defined as seen, leaving no choices available. This is not the case in general hypothetical reasoning situations.

4.1 Hypothetical facts and rules

In some cases we want to be able to hypothesize about the applicability of rules, in the sense that we may consider both the case where some rule may be applied or used to reason with to produce its conclusion, as well as those cases where that very same rule is not applied. Similarly for facts, we want to be able to represent that some unknown considered fact may be hypothesized true as well as false. This is distinct from just not having any knowledge at all about such a fact.

4.1.1 Hypothetical facts

Consider this simple example: *John and Nixon are quakers* and *John is a pacifist.* represented by the program $P_1 = \{quaker(john) \;\; pacifist(john) \;\; quaker(nixon).\}$

The $\mathcal{M}_{P_1}$ (which is the only XM model) is:

$$\left\{ \begin{array}{cccc} quaker(nixon) & \sim pacifist(nixon) & \sim\neg quaker(nixon) & \sim\neg pacifist(nixon) \\ quaker(john) & pacifist(john) & \sim\neg quaker(john) & \sim\neg pacifist(john) \end{array} \right\}$$

and expresses exactly what is intended, i.e. John and Nixon are quakers, John is a pacifist and we have no reason to believe Nixon is a pacifist, in this or any other model (there aren't any others in fact). Now suppose we want to add:

$$Nixon\ might\ be\ a\ pacifist. \tag{4}$$

In our view we wouldn't want in this case to be so strong as to affirm $pacifist(nixon)$, thereby not allowing for the possibility of Nixon not 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. Statement (4) is thus expressed as:

$$pacifist(nixon) \leftarrow \sim\neg pacifist(nixon) \tag{5}$$
$$\neg pacifist(nixon) \leftarrow \sim pacifist(nixon) \tag{6}$$

The first rule states that Nixon is a pacifist if there is no evidence against it. The second rule makes a symmetric statement. Let P_2 be the program P together with these rules. P_2 has a minimal model $\mathcal{M}_{P_2}$ (which is non–contradictory):

$$\left\{ \begin{array}{llll} quaker(nixon) & & \sim\neg quaker(nixon) & \\ quaker(john) & pacifist(john) & \sim\neg quaker(john) & \sim\neg pacifist(john) \end{array} \right\}$$

and two more XMs: $XSM_1 = \mathcal{M}_{P_2} \cup \{pacifist(nixon), \sim\neg pacifist(nixon)\}$ and $XSM_2 = \mathcal{M}_{P_2} \cup \{\neg pacifist(nixon), \sim pacifist(nixon)\}$. which is the result we were seeking. Statements of the form of (4) we dub *unknown possible facts*, and are expressed as by (5) and (6). They can be read as a fact and its negation, each of which can be assumed only if it is consistent to do so.

4.1.2 Hypothetical rules

Consider the well known nixon–diamond example using hypothetical rules instead of defeasible ones.

We represent these rules as named rules (in the fashion of [20]) where the rule name may be present in one model as true, and in others as false.

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.

The corresponding logic program is:

$$\begin{array}{rcll} pacifist(X) & \leftarrow & quaker(X), hypqp(X) & quaker(nixon) \\ hypqp(X) & \leftarrow & \sim\neg hypqp(X) & republican(nixon) \\ hawk(X) & \leftarrow & republican(X), hyprh(X) & quaker(another\_quaker) \\ hyprh(X) & \leftarrow & \sim\neg hyprh(X) & republican(another\_republican) \\ \neg hawk(X) & \leftarrow & pacifist(X) & \\ \neg pacifist(X) & \leftarrow & hawk(X) & \end{array}$$

where the following rules are also added, making each normality instance rule about Nixon hypothetical rather than defeasible (c.f. the representation of defeasible rules in section 2):

$$\begin{array}{rlrl} hypqp(nixon) & \leftarrow \sim\neg hypqp(nixon) & hyprh(nixon) & \leftarrow \sim\neg hyprh(nixon) \\ \neg hypqp(nixon) & \leftarrow \sim hypqp(nixon) & \neg hyprh(nixon) & \leftarrow \sim hyprh(nixon) \end{array}$$

as represented in fig. 1.

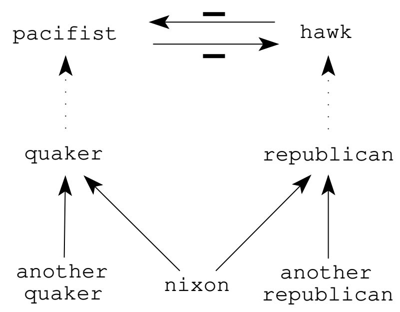

Figure 1: The nixon–diamond

The whole set of models is represented in fig. 2.

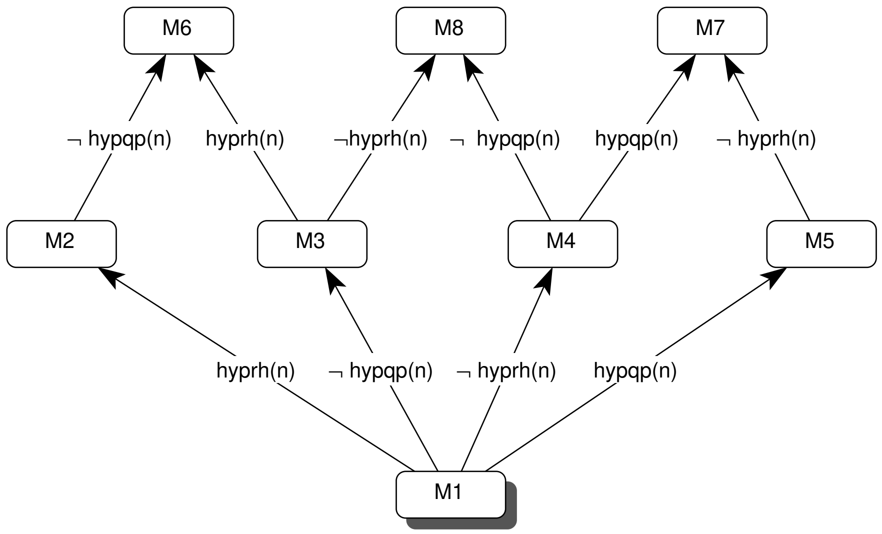

Figure 2: Models of the nixon–diamond problem using hypothetical rules

where the models (with obvious abbreviations) are:

$$
\begin{aligned}
M_1 = \quad & \{qua(n), rep(n), \sim\neg qua(n), \sim\neg rep(n),\\
& qua(a\_qua), \sim\neg qua(a\_qua), \sim rep(a\_qua), \sim\neg rep(a\_qua),\\
& hypqp(a\_qua), \sim\neg hypqp(a\_qua), pac(a\_qua), \sim\neg pac(a\_qua),\\
& hyprh(a\_qua), \sim\neg hyprh(a\_qua), \sim\neg pac(a\_qua), \sim hawk(a\_qua)\\
& rep(a\_rep), \sim\neg rep(a\_rep), \sim qua(a\_rep), \sim\neg qua(a\_rep)\\
& hyprp(a\_rep), \sim\neg hyprp(a\_rep), rep(a\_rep), \sim\neg rep(a\_rep),\\
& hypqp(a\_rep), \sim\neg hypqp(a\_rep), \sim pac(a\_rep), \sim\neg hawk(a\_rep)\}\\
M_2 = \quad & M_1 \cup \{hyprh(n), \sim\neg hyprh(n), hawk(n), \sim\neg hawk(n), \neg pac(n), \sim pac(n)\}\\
M_3 = \quad & M_1 \cup \{\neg hypqp(n), \sim hypqp(n), \sim pac(n), \sim\neg hawk(n)\}\\
M_4 = \quad & M_1 \cup \{\neg hyprh(n), \sim hyprh(n), \sim hawk(n), \sim\neg pac(n)\}\\
M_5 = \quad & M_1 \cup \{hypqp(n), \sim\neg hypqp(n), pac(n), \sim\neg pac(n), \neg hawk(n), \sim hawk(n)\}\\
M_6 = \quad & M_2 \cup \{\neg hypqp(n), \sim hypqp(n), \sim pac(n), \sim\neg hawk(n)\}\\
M_7 = \quad & M_4 \cup \{hypqp(n), \sim\neg hypqp(n), pac(n), \sim\neg pac(n), \sim\neg hawk(n)\}\\
M_8 = \quad & M_3 \cup \{\neg hyprh(n), \sim hyprh(n), \sim hawk(n), \sim\neg pac(n)\}
\end{aligned}
$$

M_1 being the most skeptical one. Edge labels represent the hypothesis being made when going from one model to another.

Note that possible rules are different from defeasible rules. Defeasible rules are applied "whenever possible" unless they lead to a contradiction. Possible rules provide equally plausible alternative extensions. In the most cautious model no hypotheses are made about the applicability of normality rules. A model exists considering the applicability of the *republicans are hawks* normality rule as well as another model (M_3) considering the non–use of it. Note that M_1 and M_3 differ precisely in the way the rule is interpreted. In some sense M_3 is a model where the normality rule is not considered at all, while in M_1 although the rule is considered it is not applied since there are other equally applicable rules which together with it would give rise to an inconsistency.

Remark 4.1 *Note that with this form of representation we might as well add abqp or $\neg abqp$, and thus the treatment of explicit negative information becomes similar to that of positive information. In this case we may now hypothesize about the applicability and non–applicability of each normality rule. However, the most skeptical model (where no hypotheses are made) is still identical to the one where normality rules were interpreted as defeasible rules, the difference being that in this case no revision is enforced since the $\mathcal{M}_P$ model is non–contradictory.*

5 Application to Reasoning about Actions

We now apply the programing methodology used above to some reasoning about action problems and show that it gives correct results. The situation calculus notation [12] is used, where predicate $holds(P, S)$ expresses that property or fluent P holds in situation S; predicate $normal(P, E, S)$ expresses that in situation S, event or action E does not normally affect the truth value of fluent P; the term $result(E, S)$ names the situation resulting from the occurrence of event E in situation S.

5.1 The Yale Shooting Problem

This problem, supplied in [5], will be represented in a form nearer to the one in [9].

Example 5 The problem and its formulation are as follows:
• Initially (in situation $s0$) a person is alive: $holds(alive, s0)$.
• After loading a gun the gun is loaded: $holds(loaded, result(load, S))$.
• If the gun is loaded then after shooting it the person will not be alive:

$$\neg holds(alive, result(shoot, S)) \leftarrow holds(loaded, S).$$

• After an event things normally remain as they were (frame axioms), i.e:
  – properties holding before the event will normally still hold afterwards:
    $holds(P, result(E, S)) \leftarrow holds(P, S), \sim ab(P, E, S)$ (pp)[6]
  – and properties not holding before the event will normally not hold afterwards:
    $\neg holds(P, result(E, S)) \leftarrow \neg holds(P, S), \sim ab(\neg P, E, S)$ (np)[7]

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:

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

With this formulation the $\mathcal{M}_P$ model is the only XS Model. The subset of its elements that match with at least one of the queries is[8]:

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

which means that in situation s3 the gun is loaded and the person is not alive. This result coincides with the one obtained in [8] for *holds*.

5.2 The Stolen Car Problem

The "Stolen Car Problem" [7] is express as follows:

1. I leave my car parked and go away: $holds(pk, s0)$.
2. I return after a while and the car is not there, i.e. after n wait events the car is not parked were I left it: $\neg holds(pk, s_n)$ where s_n represents the expression $r(w, \ldots, r(w, s0)\ldots)$ comprising n wait events.
3. After an event things normally remain as they were, represented as in the previous example.
4. In which situations S, between $s0$ and s_{n-1}, is the car parked and in which is it not?

[6] pp stands for positive persistence.
[7] np stands for negative persistence;the use of $\neg$ as a functor permits a more compact representation of (pp) and (np), with a single *ab* predicate.
[8] Where s3 denotes the term result(shoot,result(wait,result(load,s0))).

Using $CRSX$ the result agrees with common sense, and avoids the strange result that the car is parked in the situation just before I return, given by [7, 10, 23].

The $WFSX$ assigns no meaning to the program since the pseudo WFM contains both $\neg holds(pk, s_n)$ (by 2) and $holds(pk, s_n)$ (by 1 and the frame axioms). The only assumption set of the contradiction is:

$$\{\sim ab(pk, w, s0), \sim ab(pk, w, s_1), \ldots, \sim ab(pk, w, s_{n-1}\}$$

showing that contradiction is supported on assuming persistence at every wait event. So, there are n contradiction removal sets, namely: $CRS_1 = \{\sim ab(pk, w, s0)\}$, ..., $CRS_n = \{\sim ab(pk, w, s_{n-1})\}$. Accordingly:

- The $CRWFM$ is $\{holds(pk, s0), \neg holds(pk, s_n)\}$.
- There are n more $CRXSMs$:

$$\begin{aligned} CRXSM_1 &= \{holds(pk, s0), holds(pk, s_1), \neg holds(pk, s_n)\} \\ &\ldots \\ CRXSM_n &= \{holds(pk, s0), holds(pk, s_1), \ldots, holds(pk, s_{n-1}), \neg holds(pk, s_n)\} \end{aligned}$$

This can be interpreted in the following way:

- In the most skeptical view ($CRWFM$) one only can say for sure that the car was parked when I left it, and not parked when I return. It is undefined whether the car was parked in all other intermediate situations.
- For each intermediate situation there is the possibility (a $CRXSM$) that the car was still parked.

Note how the set of $CRXSMs$ can be read as the disjunct: either the car disappeared in situation s_1 or in situation s_2 or ... or in situation s_{n-1}.

6 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 rule.

- **Definite Rules** *If A then B.* The representation is: $B \leftarrow A$.
- **Definite Facts** *A is true.* The representation is: A. *A is false.* The representation is: $\neg A$.
- **Defeasible (or maximally applicable) Rules** *Normally if A then B.* The representation is:

$$B \leftarrow A, \sim ab.$$

  where $\sim ab$ is a new predicate symbol. As an example consider the rule "Normally birds fly". Its representation is: $fly(X) \leftarrow bird(X), \sim ab(X)$.

  Defeasible Facts are a special case of *Defeasible Rules* where A is absent.

- **Known Exceptions to Defeasible Rules** *Under certain conditions COND there are known exceptions to the defeasible rule $H_1 \leftarrow B_1, \sim ab_1$.*

$$ab_1 \leftarrow COND.$$

  As an example, the representation of the exception "Penguins are exceptions to the "normally birds fly" rule(i.e. rule $f \leftarrow b, \sim abb$)" is: $abb \leftarrow penguin$.

- **Possible Exceptions to Defeasible Rules** *Under certain conditions COND there are no possible exceptions to the defeasible rule* $H_1 \leftarrow B_1, \sim ab_1$.

$$ab_1 \leftarrow \sim\neg COND.$$

As an example, the representation of the exception "Animals not known to be not a penguin are exceptions to the "normally birds fly" rule(i.e. rule $f \leftarrow b, \sim abb$)" is: $abb \leftarrow \sim\neg penguin$.

Preference rules are a special kind of exception to defeasible rules:

- **Preference Rules** *Under conditions COND, prefer to apply the defeasible rule* $H_1 \leftarrow B_1, \sim ab_1$ *instead of the defeasible rule* $H_2 \leftarrow B_2, \sim ab_2$.

$$ab_1 \leftarrow COND, \sim ab_2.$$

As an example consider "For penguins, if the rule that says "normally penguins don't fly" is applicable then inhibit the "normally birds fly" rule". This is represented as:
$ab\_b \leftarrow penguin(X), \sim ab\_penguin(X)$.

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

$$\begin{array}{rcl} F & \leftarrow & \sim\neg F. \\ \neg F & \leftarrow & \sim F. \end{array}$$

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

$$\begin{array}{rcl} B & \leftarrow & A, hyp \\ hyp & \leftarrow & \sim\neg hyp \\ \neg hyp & \leftarrow & \sim hyp \end{array}$$

where hyp is a new predicate symbol. As an example consider the rule "Quakers might be pacifists". Its representation is:

$$pacifist(X) \leftarrow quaker(X), hypqp(X).$$

$$\begin{array}{rcl} hypqp(X) & \leftarrow & \sim\neg hypqp(X). \\ \neg hypqp(X) & \leftarrow & \sim hypqp(X). \end{array}$$

Acknowledgements

We thank ESPRIT BRA projects COMPULOG, COMPULOG II, INIC, JNICT, and Gabinete de Filosofia do Conhecimento for their support.

References

[1] J. J. Alferes and L. M. Pereira. On logic program semantics with two kinds of negation. In *IJCSLP'92*. MIT Press, 1992.

[2] P. M. Dung and P. Ruamviboonsuk. Well founded reasoning with classical negation. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *Workshop on LPNMR*. MIT Press, 1991.

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

[4] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *ICLP*, pages 579–597. MIT Press, 1990.

[5] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics and the frame problem. In *AAAI86*, pages 328–333, 1986.

[6] K. Inoue. Extended logic programs with default assumptions. In K. Furukawa, editor, *8th ICLP*, pages 490–504. MIT Press, 1991.

[7] H. Kautz. The logic of persistence. In *AAAI'86*, page 401, 1986.

[8] R. Kowalski. Problems and promises of computational logic. In J. Lloyd, editor, *Computational Logic Symposium*, pages 1–36. Springer-Verlag, 1990.

[9] R. Kowalski and F. Sadri. Logic programs with exceptions. In Warren and Szeredi, editors, *ICLP*, pages 598–613. MIT Press, 1990.

[10] V. Lifschitz. Pointwise circumscription. In *AAAI'86*, page 406, 1986.

[11] J. W. Lloyd. *Foundations of Logic Programming*. Symbolic Computation. Springer–Verlag, 1984.

[12] J. McCarthy and P. J. Hayes. *Some Philosophical Problems from the Standpoint of Artificial Intelligence*, pages 26–45. Readings in Nonmonotonic Reasoning. M. Kaufmann, Inc, 1987.

[13] L. M. Pereira and J. J. Alferes. Well founded semantics for logic programs with explicit negation. In *ECAI'92*. John Wiley & Sons, Ltd, 1992.

[14] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Contradiction removal semantics with explicit negation. Technical report, AI Centre, Uninova, July 1992. *Submitted*.

[15] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Default theory for well founded semantics with explicit negation. In *JELIA'92*, 1992.

[16] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Well founded semantics with explicit negation and default theory. Technical report, AI Center, Uninova, 1992. *Submitted*.

[17] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Counterfactual reasoning based on revising assumptions. In V. Saraswat and K. Ueda, editors, *ILPS*, pages 566–580. MIT Press, 1991.

[18] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Hypothetical reasoning with well founded semantics. In B. Mayoh, editor, *Third Scandinavian Con. on AI*. IOS Press, 1991.

[19] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Nonmonotonic reasoning with well founded semantics. In K. Furukawa, editor, *ICLP91*, pages 475–489. MIT Press, 1991.

[20] D. L. Poole. A logical framework for default reasoning. *Journal of AI*, 36(1):27–47, 1988.

[21] T. Przymusinski. Extended stable semantics for normal and disjunctive programs. In Warren and Szeredi, editors, *ICLP90*, pages 459–477. MIT Press, 1990.

[22] T. C. Przymusinski. A semantics for disjunctive logic programs. In D. Loveland, J. Lobo, and A. Rajasekar, editors, *ILPS Workshop on Disjunctive Logic Programs*, 1991.

[23] Y. Shoham. *Reasoning about Change: Time and Change from the Standpoint of Artificial Intelligence*. MIT Press, 1988.

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

[25] G. Wagner. A database needs two kinds of negation. In B. Thalheim, J. Demetrovics, and H-D. Gerhardt, editors, *MFDBS'91*, pages 357–371. Springer-Verlag, 1991.