Contradiction: when avoidance equals removal
Part I

José Júlio Alferes and Luís Moniz Pereira

CRIA, Uninova and DCS, U. Nova de Lisboa*
2825 Monte da Caparica, Portugal
{jja | lmp}@fct.unl.pt

Abstract. Recently several authors have stressed and illustrated the importance of including a second kind of negation (explicit negation) in logic programs besides *"negation as failure"*, and its use in deductive databases, knowledge representation, and nonmonotonic reasoning.
By introducing explicit negation into logic programs contradiction may appear. In this work we present two approaches for dealing with contradiction, and show their equivalence. One of the approaches consists in avoiding contradiction, and is based on restrictions in the adoption of abductive hypotheses. The other approach consists in removing contradiction, and is based on a transformation of contradictory programs into noncontradictory ones, guided by the reasons for contradiction.
The work is divided into two parts: one is presented in this paper, and comprises the contradiction avoidance approach, and the other in [16] in this volume, comprises the contradiction removal approach and shows the equivalence between the avoidance and removal approaches.

1 Introduction

Recently several authors have stressed and illustrated the importance of including a second kind of negation in logic programs besides *"negation as failure"*, and its use in deductive databases, knowledge representation, and nonmonotonic reasoning [9, 12, 13, 14, 10, 18, 19, 20, 27].

Proposals for extending logic programming semantics with a second negation have been advanced. One is the Answer Sets semantics [9], shown to be an extension of the Stable Model semantics [8] of normal logic programs. In [13] a similar extension proposal was introduced, based also on stable models, where an implicit preference between negative information (exceptions) over positive one is assumed. However, answer sets semantics is not well founded. The meaning of the program, defined as the intersection of all answer sets, is known to be computationally expensive. Yet another extension to include a second negation is suggested by Przymusinski in [24]. Though the intersection of models identified by this extension is a model and enjoys the well founded property, it gives less intuitive results [2] with respect to the coexistence of both forms of negation.

* We thank JNICT and Esprit BR project Compulog 2 (no 6810) for their support.

Well Founded Semantics with Explicit Negation (*WFSX*) [15], which we prefer, is an extension of the Well Founded Semantics [26] to include a second negation $\neg$ called *explicit negation*, that preserves well foundedness and procedural properties. *Explicit negation* is characterized by that, whatever the objective literal L, whenever $\neg L$ holds $not\ L$ does too (*Coherence Principle*), and so L is false, thus avoiding the less intuitive results mentioned.

Once the new negation is introduced contradiction may arise (e.g. when L and $\neg L$ both hold) and no meaning is assigned[2]. While for some programs this seems reasonable (e.g. $P = \{a \leftarrow;\ \neg a \leftarrow\}$), for others this is too strong.

Example 1. Consider the statements: *Birds not shown to be abnormal fly; Tweety is a bird and does not fly; Socrates is a man;* naturally expressed by the program:

$$\begin{array}{ll} fly(X) \leftarrow bird(X), not\ abnormal(X) & \neg fly(tweety) \\ bird(tweety) & man(socrates) \end{array}$$

None of the above mentioned semantics assign a meaning to this program. Intuitively however, we should at least be able to say that $Socrates$ is a man and $tweety$ is a $bird$. It is also reasonable to conclude it doesn't fly, since the fact stating that it doesn't fly makes a stronger statement than the rule concluding it $flies$ because not abnormal. The latter relies on accepting an assumption of non-abnormality, enforced by the closed world assumption treatment of the negation as failure involving the abnormality predicate. Indeed, whenever an assumption supports a contradiction it seems logical to be able to take the assumption back in order to prevent it –*"reductio ad absurdum"* or *"reasoning by contradiction"*.

The scenario semantics paradigm of logic programs [6] has been recently expanded in [1] to encompass extended logic program, including *WFSX*, built upon simple primitive notions, such as those of "scenario" (a program plus a set of NAF-hypotheses), "acceptability of a hypothesis wrt to a scenario" (i.e. without contrary evidence), "evidence contrary to a hypothesis" (i.e. that contradicts it), "admissible scenario" (i.e. all its hypotheses are acceptable), "completeness of a set of hypotheses wrt to a scenario" (i.e. contains all acceptable hypotheses), etc.

[1] presents semantics more sceptical than *WFSX*, thus avoiding contradiction in cases where the latter gives no meaning to a program. For example, the semantics *WFS0*, whose precise details are not relevant here, assigns to the above program the meaning (with obvious abbreviations for constants):

$$\{man(s), \neg fly(t), bird(t), not\ fly(t)\}$$

which corresponds to intuition[3].

[2] Other researchers have defined paraconsistent semantics for contradictory programs [5, 3, 25, 28]. This is not our concern. On the contrary, we wish to remove contradiction whenever it rests on withdrawable assumptions.

[3] For the sake of simplicityt, we omit in the model some literals that are irrelevant for the problem (such as $flies(s)$, $\neg flies(s)$, $bird(s)$, $man(t)$, etc). All these literals are false by default (i.e. $not\ flies(s)$, etc, belong to the model).

Furthermore, there is motivation to consider semantics even more sceptical than *WFS0*, in which some acceptable hypotheses might not be adopted in complete scenarios. For instance, the acceptance of a hypothesis may be conditional upon the equal acceptance of another. This is typical of hypothesizing faults in a device or program debugging, whenever causally deeper faults are to be preferred over faults that are simply a consequence of the deeper ones: the former cannot be hypothesized without the latter [21, 22]. Problem specific and user defined preference criteria affecting acceptance of hypotheses is another instance. In general, the rules of a logic program may be seen as providing a causal directionality of inference, similar to physical causality directionality, so that a distinction can sometimes be drawn about the primacy of one hypothesis over another (cf. [11, 4]).

Example 2. Consider this program, describing bicycle behaviour:

$$\begin{aligned} \neg wobbly\_wheel &\leftarrow not\ flat\_tyre, not\ broken\_spokes \\ flat\_tyre &\leftarrow leaky\_valve \\ flat\_tyre &\leftarrow punctured\_tube \\ \neg no\_light &\leftarrow not\ faulty\_dynamo \end{aligned}$$

plus the factual observation: $wobbly\_wheel$. *WFS0* assigns to it the meaning:

$$\{wobbly\_wheel, not\ faulty\_dynamo, \neg no\_light, not\ no\_light, \\ not\ leaky\_valve, not\ punctured\_tube\}$$

neither accepting the hypothesis $not\ flat\_tyre$ nor $not\ broken\_spokes$, because acceptance of any of them, if the other were accepted too, would lead to a contradiction. Being sceptical *WFS0* accepts neither. However, one would like the semantics in this case to delve deeper into the bicycle model and, again being sceptical, accept neither $not\ leaky\_valve$ nor $not\ punctured\_tube$ as well.

In order to respond to such epistemological requirements, we begin by introducing into the complete scenario semantics of [1] the more flexible notion of optative acceptance of hypotheses. In a complete scenario, optative hypotheses, or optatives, might or might not be accepted even if acceptable. On the other hand, non-optative hypotheses must be accepted if acceptable.

First we make no restriction on the optatives, and consider them provided by the user along with the program. Then we proceed to consider the issue of inferring optative hypotheses from the program, given specific criteria. In particular we show how to infer optatives when the criterion is those hypotheses that do not depend on any other.

As pointed out in [17], these more sceptical semantics model rational reasoners who assume the program absolutely correct and so, whenever confronted with an acceptable hypothesis leading to an inconsistency cannot accept it; i.e. they prefer to assume the program correct rather than assume that an acceptable hypothesis must perforce be accepted.

WFSX models less sceptical reasoners who, confronted with an inconsistent scenario, prefer considering the program wrong rather than admitting that an acceptable hypothesis be not accepted. Such a reasoner is more confident in his acceptability criterion: an acceptable hypothesis is accepted once and for all; if an inconsistency arises then there is certainly a problem with the program, not with the individual acceptance of each acceptable hypothesis. If the problem is with the program then its revision is in order. This view position can be justified if we think of a program as resulting from the assimilation of knowledge into a previous one.

In [12], Kowalski presents a detailed exposition of assimilation processes in various cases, and he claims the notion of integrity constraint is needed in logic programming for knowledge processing, representation, and assimilation. The problem of inconsistency arises from nonsatisfaction of the integrity constraints. If some new knowledge can be shown incompatible with the existing theory and integrity constraints, a revision process is needed to restore satisfaction of those constraints.

In extended logic programming we can view the requirement of non-contradiction as integrity constraint satisfaction, where constraints are denials of the form $\leftarrow L, \neg L$. Consequently we extend logic programs with integrity constraints in the form of denials.

Let's go back to example 1, and view the program as the result of adding to the previous knowledge the fact that tweety doesn't fly. According to *WFSX* the resulting program is inconsistent. One way of restoring consistency would be to add a rule stating that $ab(tweety)$ cannot be false, viz. assuming so would lead directly to contradiction: $ab(tweety) \leftarrow not\ ab(tweety)$. The resulting program is now non-contradictory and its $WFSX$ contains:

$$\{man(s), \neg fly(t), bird(t), not\ fly(t)\}$$

which corresponds to intuition.

This work is divided into two parts. In this part we present a sceptical semantics which avoids contradiction for extended logic programs plus integrity contraints in the form of denials, based on the notion of optative hypotheses –*an abductive approach*. In the second part of this work, in [16], we define a program revision method for removing contradiction from contradictory programs under *WFSX*, based on the notion of revisable hypotheses –*a belief revision approach*– and show the equivalence between the contradiction avoidance semantics and the *WFSX* of revised programs obtained by contradiction removal. Proofs of all theorems are omitted for brevity, but exist in an extended version of this work.

2 Logic Programming with Denials

A program with integrity rules (or constraints) is a set of ground rules:

$$H \leftarrow A_1, \ldots, A_n, not\ B_1, \ldots, not\ B_m, \quad (n, m \geq 0)$$

where H, $A_1, \ldots, A_n, B_1, \ldots, B_m$ are objective literals. An objective literal is an atom A or its explicit negation $\neg A$. A default (or NAF) literal is $not\ L$ where L is an objective literal. Literals are either objective or default ones. The (default) complement of objective literal L is the default literal $not\ L$, and vice-versa. The explicit negation complement of objective literal $\neg L$ is the atom L, and vice-versa. $not\ S$, where S is a set of literals, denotes the set of complements of those in S. $\mathcal{H}$ stands for the set of all objective literals of a program.

An integrity rule is a rule whose head is the reserved atom $\bot$, standing for falsity. Integrity rules must have a non-empty body.

A program P with semantics SEM satisfies the integrity rules iff $P \not\models_{SEM} \bot$.

3 Contradiction Avoidance

Next we present a semantics more sceptical than *WFS0*, based on the notion of scenarios presented in [1]. We begin by briefly reviewing some concepts presented there and needed in the sequel.

Definition 1. A *scenario* of an extended logic program P is the first order Horn theory $P \cup H$, where the set of default literals $H \subseteq not\ \mathcal{H}$ are the scenario hypotheses.

When introducing explicit negation into logic programs one has to consider its relation to the notion of default negation. When a scenario $P \cup H \vdash \neg A$[4] it is *explicitly* stating that A is false in that scenario. Thus the hypothesis $not\ A$ must be enforced in the scenario, and cannot optionally be held independently.

Definition 2. The set of *mandatory hypotheses* wrt a scenario $P \cup H$ is:

$$Mand(H) = \{not\ L \mid P \cup H \cup \{not\ L \leftarrow \neg L \mid L \in \mathcal{H}\} \vdash not\ L\}^5$$

A scenario $P \cup H$ of a program with integrity rules IC is *consistent* iff:

$$P \cup H \cup Mand(H) \cup IC \cup \{\bot \leftarrow L, not\ L \mid L \in \mathcal{H}\} \not\vdash \bot$$

An extended logic program P with integrity constraints IC is consistent iff it has some consistent scenario.

N.B. From now on, unless otherwise stated, we restrict programs to consistent ones only.

4 The rather straightforward formal definition of $\vdash$, where each (ground) $not\ L$ is treated as a new propositional symbol not_L, and each (ground) $\neg L$ is treated as a new propositional symbol $\neg\_L$, can be found in [1]. Intuitively, $\vdash$ is just the standard T_P operator of the Horn propositional programs obtained with the new symbols in place.

5 The rule $not\ L \leftarrow \neg L$ amounts to the *"coherence principle"* of [15].

Not every consistent scenario specifies a consensual semantics for a program [23]. For example [6] the program $P = \{p \leftarrow not\ q\}$ has a consistent scenario $P \cup \{not\ p\}$ which fails to give the intuitive meaning of P. It is not consensual to assume *not* p since there is the possibility of p being true (if *not* q is assumed), and $\neg p$ is not explicitly stated (if this were the case then *not* q could not be assumed). *Intuitively, a hypothesis can be assumed only if there can be no evidence to the contrary.* Clearly a hypotheses *not* L is only contradicted by the objective literal L. Evidence for an objective literal L in a program P is any set of hypotheses which, if assumed in P, would derive L. As in [6], a hypothesis is acceptable wrt a scenario iff any evidence to the contrary is defeated by the scenario:

Definition 3. $E \subseteq not\ \mathcal{H}$ is *evidence for objective literal* L (and against *not* L) in P iff $P \cup E \cup Mand(E) \vdash L$[6], and we say $P \cup E$ defeats *not* L. If P is understood and E is evidence for L we write $E \leadsto L$.

A hypothesis *not* L is *acceptable* wrt the scenario $P \cup H$ iff

$$\forall E : E \leadsto L \Rightarrow \exists not\ A \in E \mid P \cup H \cup Mand(H) \vdash A$$

i.e. in each evidence against *not* L there is a hypothesis defeated by $P \cup H$.

Whenever P is understood, the set of acceptable hypotheses wrt $P \cup H$ is denoted by $Acc(H)$.

In a consensual semantics we are interested in admitting only consistent scenarios whose hypotheses are either acceptable or mandatory.

Definition 4. A scenario $P \cup H$ is *admissible* iff it is consistent and

$$Mand(H) \subseteq H \subseteq Mand(H) \cup Acc(H)$$

Based on this notion, in [1] some more or less sceptical semantics are defined. Here we review the complete scenario semantics, which has been proven equivalent to *WFSX* there.

Definition 5. A scenario $P \cup H$ is *complete* iff it is em consistent, and for each *not* L :

$$\begin{array}{ll} (i) & not\ L \in H \Rightarrow not\ L \in Acc(H) \vee not\ L \in Mand(H) \\ (ii) & not\ L \in Mand(H) \Rightarrow not\ L \in H \\ (iii) & not\ L \in Acc(H) \Rightarrow not\ L \in H \end{array}$$

where (i) and (ii) simply express admissibility. In other words, a scenario $P \cup H$ is complete iff $H = Mand(H) \cup Acc(H)$.

The *complete scenarios semantics* of P is the set of all complete scenarios of P. As usual, the meaning of P is determined by the intersection of all such scenarios.

[6] The consistency of $P \cup E$ is not required (cf. [6]); e.g. $P \cup \{not\ H\} \vdash H$ is allowed.

If every acceptable hypothesis must be accepted some programs might have no meaning (viz. example 1). In *WFS0* some acceptable hypotheses are not accepted in order to avoid inconsistency. However, as shown in example 2, *WFS0* allows no control over which acceptable hypotheses are not accepted. Conceivably, any acceptable hypothesis may or may not actually be accepted, in some discretionary way.

It is clear from example 2 that we wish to express that only the hypotheses $not\ broken\_spokes$, $not\ leaky\_valve$, $not\ faulty\_dynamo$ and $not\ punctured\_tube$ may be optative, i.e. to be possibly accepted or not, if acceptable. The acceptance of hypotheses like $not\ flat\_tyre$ is to be determined solely by the acceptance of other hypotheses, and so we always wish them accepted once acceptable.

Thus we should distinguish between *optative hypotheses* (or optatives) and non-optative ones. Optative hypotheses are those in some pre-defined $Opt \subseteq not\ \mathcal{H}$. That distinction made, we can conceive of scenarios that might not be complete wrt optatives, but are still complete wrt non-optatives: i.e. scenarios which contain all acceptable hypotheses except for possibly optative ones.

In general, when some acceptable optative hypothesis $not\ L$ is not accepted, then some otherwise acceptable hypotheses become unacceptable:

Example 3. Let $P = \{p \leftarrow not\ a;\ a \leftarrow b;\ \bot \leftarrow p\}$ where $Opt = \{not\ b\}$.

In our notion of optative, if $not\ b$ is not accepted then $not\ a$ is unacceptable, i.e. if optative b is not assumed false, the possibility of being true must be considered and so a cannot be assumed false; $P \cup \{b\} \vdash a$ counts as evidence against $not\ a$.

Definition 6. A hypothesis $not\ L$ is *acceptable wrt scenario $P \cup H$ and optatives Opt* iff $not\ L$ is acceptable[7] both wrt $P \cup H$ and $P \cup H \cup F$, where F is the set of facts $not\ ((Opt \cap Acc(H)) - H)$, i.e. F is the set of complements of acceptable Opts wrt H which are not in H (that is which were not accepted).

$Acc_{Opt}(H)$ denotes the set of acceptable hypotheses wrt $P \cup H$ and Opt.

Example 4. In example 3 $Acc_{Opt}(\{not\ p\}) = \{\}$. $not\ b$ is not acceptable because, even though acceptable wrt $P \cup \{not\ p\}$, it is not acceptable wrt $P \cup \{not\ p\} \cup \{b\}$[8]. The same happens with $not\ a$.

With this more general notion of acceptability scenarios may be partially complete; i.e. complete wrt non-optatives, but possibly not complete wrt optatives (condition (iii) below):

Definition 7. A scenario $P \cup H$ is a *complete scenario wrt a set of optatives Opt* iff it is consistent, and for each $not\ L$:

$$
\begin{array}{ll}
(i) & not\ L \in H \Rightarrow not\ L \in Acc_{Opt}(H) \vee not\ L \in Mand(H) \\
(ii) & not\ L \in Mand(H) \Rightarrow not\ L \in H \\
(iii) & not\ L \in Acc_{Opt}(H) \text{ and } not\ L \notin Opt \Rightarrow not\ L \in H
\end{array}
$$

[7] Cf. definition 3.

[8] Note that here $not\ ((Opt \cap Acc(H)) - H) = not\ (\{not\ b\} - \{not\ p\}) = \{b\}$.

Let $S = P \cup H$ be a complete scenario wrt *Opt*. A hypothesis in *Opt* acceptable wrt $P \cup H$ that leads to an inconsistent scenario if added to S, will simply not be accepted in it so as to preserve consistency. This amounts to contradiction avoidance.

Example 5. Recall the wobbly wheel example 2. If *Opt* were $\{\}$ there would be no complete scenario because of inconsistency. If (with obvious abbreviations) $Opt = \{not\ bs, not\ lv, not\ pt,\ not\ fd\}$, complete scenarios wrt *Opt* are :

$$\begin{array}{ccc}
\{not\ \neg ww\} & \{not\ \neg ww, not\ lv\} & \{not\ \neg ww, not\ lv, not\ pt, not\ ft\} \\
\{not\ \neg ww, not\ fd\} & \{not\ \neg ww, not\ pt\} & \{not\ \neg ww, not\ fd, not\ lv\} \\
\{not\ \neg ww, not\ bs\} & \{not\ \neg ww, not\ fd, not\ bs\} & \ldots
\end{array}$$

It is clear some of these scenarios are over-sceptical, in the sense that they fail to accept more optatives than need be to avoid contradiction. For example in the first scenario, in order to avoid contradiction none of the optatives were accepted. This occurs because no condition of maximal acceptance of optatives was enforced.

In order to impose this condition we begin by identifying, for each complete scenario wrt *Opt*, those optatives that though acceptable were not accepted.

Definition 8. Let $P \cup H$ be a complete scenario wrt *Opt*. The *avoidance set* of $P \cup H$ is $(Opt \cap Acc(H)) - H$.

Example 6. The avoidance set of the first scenario in example 5 is $\{not\ lv, not\ pt,$ $not\ fd\}$ and of the second one is $\{not\ lv, not\ pt\}$.

In keeping with the scepticism vocation of *WFSX*, consider those scenarios which, for some given avoidance set, are minimal.

Definition 9. A complete scenario $P \cup H$ wrt *Opt* is a *base scenario* if there exists no scenario $P \cup H'$, with the same avoidance set, such that $H' \subset H$.

Example 7. Consider $P = \{a \leftarrow not\ b;\ b \leftarrow not\ a;\ c \leftarrow not\ d;\ \bot \leftarrow c\}$ with $Opt = \{not\ d\}$.

Complete scenarios wrt *Opt* are $\{\}$, $\{a, not\ b\}$, and $\{b, not\ a\}$.

For all, the avoidance set is $\{not\ d\}$. The corresponding base scenario wrt *Opt* is the first.

Proposition 10. *The base scenarios wrt Opt form a lower semi-lattice under set inclusion.*

Consider now those scenarios comprising as many optatives as possible, i.e. having minimal avoidance sets:

Definition 11. A base scenario $P \cup H$ wrt *Opt*, with avoidance set S, is *quasi-complete* if there is no base scenario $P \cup H'$ wrt *Opt*, with avoidance set S', such that $S' \subset S$.

Example 8. In example 5 the quasi-complete scenarios wrt *Opt* are:

$$\{not\ \neg ww, not\ fd, not\ bs, not\ lv\}$$
$$\{not\ \neg ww, not\ fd, not\ bs, not\ pt\}$$
$$\{not\ \neg ww, not\ fd, not\ lv, not\ pt, not\ ft\}$$

These correspond to minimal faults compatible with the wobbly wheel observation, i.e. the ways of avoiding contradiction (inevitable if *Opt* were $\{\}$) by minimally not accepting acceptable optatives. In the first *not pt* was not accepted, in the second *not lv*, and in the third *not bs*.

As the consequences of these quasi-complete scenarios are pairwise incompatible the well-founded scenario, being sceptical, is their meet in the semi-lattice of proposition 10, so that its avoidance set is the union of their avoidance sets.

Definition 12. The *well-founded scenario of a program P with ICs* is the meet of all quasi-complete scenarios wrt *Opt* in the semi-lattice of all base scenarios. For short we use WFS_{Opt} to denote the well-founded scenario wrt *Opt*.

Example 9. In example 5 $WFS_{Opt} = P \cup \{not\ \neg ww, not\ fd\}$. Thus one can conclude:

$$\{ww, \neg nl, not\ \neg ww, not\ fd\}$$

i.e. no other hypothesis can be assumed for certain; everything is sceptically assumed faulty except for fd. This differs from the result of *WFS0*, shown in example 2.

Example 10. Consider the statements: *Let's go hiking if it is not known to rain; Let's go swimming if it is not known to rain; Let's go swimming if the water is not known to be cold; We cannot go both swimming and hiking.* They render the set of rules P:

$$\begin{array}{ll} \bot \leftarrow hiking, swimming & swimming \leftarrow not\ rain \\ hiking \leftarrow not\ rain & swimming \leftarrow not\ cold\_water \end{array}$$

and let $Opt = \{not\ rain, not\ cold\_water\}$.

Complete scenarios wrt *Opt* are $P \cup \{\}$, and $P \cup \{not\ cold\_water\}$, where the latter is the well founded wrt *Opt*. It entails that *swimming* is true. Note that *not rain* is not assumed because it is optative to do so, and by assuming it contradiction would be unavoidable.

To obtain less sceptical complete scenarios wrt Opt, and in the spirit of partial stable models [24], we introduce:

Definition 13. Let P be an extended logic program with ICs, and let the well-founded scenario of P wrt *Opt* be $P \cup H$.

$P \cup K$ is a *partial scenario of P wrt Opt* iff it is a base scenario wrt *Opt* and $H \subseteq K$.

Example 11. The partial scenarios of P wrt Opt in example 5 are the union of P with each of:

$$\begin{array}{ll} \{not\ \neg ww, not\ fd\} & \{not\ \neg ww, not\ fd, not\ bs, not\ lv\} \\ \{not\ \neg ww, not\ fd, not\ bs\} & \{not\ \neg ww, not\ fd, not\ bs, not\ pt\} \\ \{not\ \neg ww, not\ fd, not\ lv\} & \{not\ \neg ww, not\ fd, not\ lv, not\ pt, not\ ft\} \\ \{not\ \neg ww, not\ fd, not\ pt\} & \end{array}$$

The first is the WFS_{Opt} (cf. example 9), which corresponds to the most sceptical view whereby all possibly relevant faults are assumed. The other extended scenarios represent, in contrast, all other alternative hypothetical presences and absences of faults still compatible with the wobbly wheel observation.

If a program is non-contradictory (i.e. its *WFSX* exists) then no matter which are the optatives, the well-founded semantics wrt *Opt* is always equal to the least complete scenario wrt $\{\}$ (and so, ipso facto, equivalent to the *WFSX*).

Theorem 14. *If* WFSX *is defined for a program* P *with an empty set of ICs then, for whatever* Opt, WFS_{Opt} *is the least complete scenario of* P.

For programs without explicit negation *WFSX* is equivalent to well-founded semantics of [26] (WFS).

4 Primacy in Optative Reasoning

Up to now no restriction was made regarding the optatives of programs. It is possible for optatives to be identified by the user along with the program, or for the user to rely on criteria for specifying the optatives, and expect the system to infer them from the program.

Next we identify a special class of optatives, governed by an important criterion [11, 4]: *Exactly the hypotheses not depending on any others are optative.*

Example 12. Let $P = \{a \leftarrow not\ b;\ b \leftarrow not\ c;\ c \leftarrow not\ d\}$.

Clearly $not\ a$ depends on $not\ b$, $not\ b$ on $not\ c$ and $not\ c$ on $not\ d$. $not\ d$ alone does not depend on any other hypothesis and so is the only optative.

In diagnosis this criterion means hypothesizing as abnormal first the causally deeper faults.

In taxonomies with exceptions this is not the desired preference criterion. To give priority to the most specific default information only a hypothesis on which no other depends should be optative. This way the relinquishing of default hypotheses to avoid contradiction begins with less specific ones.

The subject of defining preference criteria to automatically determine optative hypotheses from programs is complex. It is closely related to that of preference among defaults [7].

How to infer optatives for criteria different from the one above is left open.

Clearly every hypothesis which is not acceptable in $P \cup \{\}$ depends on the acceptance of some other hypothesis. In other words, if a hypothesis $not\ L$ is acceptable in a scenario $P \cup H$, but is not acceptable in $P \cup \{\}$, this means that in order to make $not\ L$ acceptable some other hypotheses $S \subseteq H$ have to be accepted too. Thus $not\ L$ depends on the hypotheses of S, and the latter are more primal than $not\ L$. As a first approximation we define the set of prime optative hypotheses as $Acc(\{\})$.

Example 13. In program P of example 12 $Acc(\{\}) = \{not\ d\}$. So the only prime optative hypothesis is $not\ d$. Hypothesis $not\ b$ is not prime optative because it is only acceptable once $not\ d$ is accepted, otherwise $not\ c$ constitutes evidence to the contrary.

In general, not all hypotheses in $Acc(\{\})$ though are independant of one another. Hence we must refine our first approximation to prime optatives.

Example 14. Let $P = \{a \leftarrow b;\ b \leftarrow c;\ p \leftarrow not\ a;\ \perp \leftarrow p\}$.

Then the optatives are $Acc(\{\}) = \{not\ a, not\ b, not\ c\}$, and the WFS wrt $Acc(\{\})$ is $P \cup \{not\ b, not\ c\}$.

However, it is clear from the program that only $not\ c$ should be prime optative, since the acceptance of $not\ b$ depends on the absence of conclusion c in P, but not vice-versa, and likewise regarding the acceptance of $not\ a$. Any definition of a semantics based on the notions of scenarios and evidence alone cannot distinguish the optative primacy of $not\ c$, because it is insensitive to the groundedness of literals, viz. there being no rules for c, and thus its non-dependence on other hypotheses.

An asymmetry must be introduced, based on a separate new notion, to capture the causal directionality of inference implicit in logic program rules, as mentioned in the introduction:

Definition 15. A hypothesis $not\ A \in Acc(\{\})$ is *sensitive* to a separate set of hypotheses $not\ F$ in P iff $not\ A \notin Acc(P \cup F)$. Note that F is a set of facts.

A hypothesis $not\ A \in Acc(\{\})$ is *prime optative* iff for all $not\ S \subseteq Acc(\{\})$ if $not\ A$ is sensitive to $not\ S$ then an element of $not\ S$ is sensitive to $\{not\ A\}$.

The set of prime optatives is denoted by $\mathcal{POpt}$, and we refer to the well-founded semantics wrt $\mathcal{POpt}$ as the *prime optative semantics*, or $\mathcal{POS}$.

Example 15. In example 14 the only prime optative is $not\ c$. For example, $not\ a$ is not prime optative since $not\ a$ is sensitive to $\{not\ b\}$ and not vice-versa.

Example 16. In example 2, $\mathcal{POpt} = \{not\ bs, not\ pt, not\ lv, not\ fd\}$.

$Acc(\{\}) = \mathcal{POpt} \cup \{not\ ft\}$. However, $not\ ft$ is not prime optative since it is sensitive to both $\{not\ lv\}$ and $\{not\ pt\}$, but not vice-versa.

Example 17. Consider $P = \{p \leftarrow not\ a;\ \neg p;\ a \leftarrow b;\ b \leftarrow a, not\ c;\ c \leftarrow not\ d\}$. Then $Acc(\{\}) = \{not\ a, not\ b, not\ d\}$.

All of these are prime optatives: *not* d because it is insensitive to other hypotheses; *not* b because it is only sensitive to $\{not\ a\}$, but *not* a is sensitive to $\{not\ b\}$; similarly for *not* a.

By insisting on only allowing prime optatives to be possibly accepted, if acceptable, one may fail to give meaning to some consistent programs, as there are fewer options for avoiding inconsistency.

Example 18. Let $P = \{c \leftarrow not\ b;\ b \leftarrow not\ a;\ \neg a;\ \bot \leftarrow not\ c\}$.

In this case $\mathcal{P}\mathcal{O}pt = Acc(\{\}) = \{not\ a\}$, and no complete scenario wrt $\mathcal{P}\mathcal{O}pt$ exists. Thus $\mathcal{POS}$ is not defined.

Note that by making $Opt = \{not\ c\}$, $P \cup \{not\ a\}$ is now complete wrt Opt. In fact this scenario corresponds to the $WFM_{\{not\ c\}}$, expressing that contradiction is avoided by not assuming the optative hypothesis *not* c. It still allows the conclusions $\{\neg a, not\ a, b\}$.

References

1. J. J. Alferes, P. M. Dung, and L. M. Pereira. Scenario semantics of extended logic programs. In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. on Logic Programming and NonMonotonic Reasoning*. MIT Press, 1993.
2. J. J. Alferes and L. M. Pereira. On logic programs semantics with two kinds of negation. In K. Apt, editor, *Int. Joint Conf. and Symp. on Logic Programming*, pages 574–588. MIT Press, 1992.
3. H. Blair and V. S. Subrahmanian. Paraconsistent logic programming. In *Conf. on Foundations of Software Technology and Theoretical Computer Science*, pages 340–360. Springer–Verlag, 1987.
4. G. Brewka and K. Konolige. An abductive framework for generalized logic programs and other nonmonotonic systems. Tech. report, GMD and SRI Int., 1993.
5. N. Costa. On the theory of inconsistency formal system. *Notre Dame J. of Formal Logic*, 15:497–510, 1974.
6. P. M. Dung. Negation as hypotheses: An abductive framework for logic programming. In K. Furukawa, editor, *8th Int. Conf. on Logic Programming*, pages 3–17. MIT Press, 1991.
7. P. Geerts and D. Vermeir. A nonmonotonic reasoning formalism using implicit specificity information. In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. on Logic Programming and NonMonotonic Reasoning*. MIT Press, 1993.
8. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. A. Bowen, editors, *5th Int. Conf. on Logic Programming*, pages 1070–1080. MIT Press, 1988.
9. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In D. H. D. Warren and P. Szeredi, editors, *7th Int. Conf. on Logic Programming*, pages 579–597. MIT Press, 1990.
10. K. Inoue. Extended logic programs with default assumptions. In K. Furukawa, editor, *8th Int. Conf. on Logic Programming*, pages 490–504. MIT Press, 1991.
11. K. Konolige. Using default and causal reasoning in diagnosis. In B. Nebel, C. Rich, and W. Swartout, editors, *3rd Int. Conf. on Knowledge Representation and Reasoning*. Morgan Kaufmann, 1992.

12. R. Kowalski. Problems and promises of computational logic. In John Lloyd, editor, *Computational Logic Symp.*, pages 1–36. Springer-Verlag, November 1990.
13. R. Kowalski and F. Sadri. Logic programs with exceptions. In D. H. D. Warren and P. Szeredi, editors, *7th Int. Conf. on Logic Programming*. MIT Press, 1990.
14. D. Pearce and G. Wagner. Reasoning with negative information I: Strong negation in logic programs. In L. Haaparanta, M. Kusch, and I. Niiniluoto, editors, *Language, Knowledge and Intentionality*, pages 430–453. Acta Philosophica Fennica 49, 1990.
15. L. M. Pereira and J. J. Alferes. Well founded semantics for logic programs with explicit negation. In B. Neumann, editor, *European Conf. on AI*, pages 102–106. John Wiley & Sons, Ltd, 1992.
16. L. M. Pereira and J. J. Alferes. Contradiction: when avoidance equal removal. Part II. In R. Dyckhoff, editor, *4th Int. Ws. on Extensions of Logic Programming*, 1993.
17. L. M. Pereira and J. J. Alferes. Optative reasoning with scenario semantics. In D. S. Warren, editor, *10th International Conference on Logic Programming*, pages 601–615. MIT Press, 1993.
18. L. M. Pereira, J. N. Aparício, and J. J. Alferes. Counterfactual reasoning based on revising assumptions. In Ueda and Saraswat, editors, *Int. Logic Programming Symp.* MIT Press, 1991.
19. L. M. Pereira, J. N. Aparício, and J. J. Alferes. Nonmonotonic reasoning with well founded semantics. In K. Furukawa, editor, *8th Int. Conf. on Logic Programming*, pages 475–489. MIT Press, 1991.
20. L. M. Pereira, J. N. Aparício, and J. J. Alferes. Logic programming for nonmonotonic reasoning. In *Applied Logic Conf.* ILLC, Amsterdam, 1992.
21. L. M. Pereira, C. Damásio, and J. J. Alferes. Diagnosis and debugging as contradiction removal. In L. M. Pereira and A. Nerode, editors, *2nd International Workshop on Logic Programming and NonMonotonic Reasoning*, pages 316–330. MIT Press, 1993.
22. L. M. Pereira, C. Damásio, and J. J. Alferes. Diagnosis and debugging as contradiction removal in logic programs. In L. Damas and M. Filgueiras, editors, *6th Portuguese Artificial Intelligence Conference*. Springer–Verlag, 1993.
23. D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.
24. T. Przymusinski. Extended stable semantics for normal and disjunctive programs. In D. H. D. Warren and P. Szeredi, editors, *7th Int. Conf. on Logic Programming*, pages 459–477. MIT Press, 1990.
25. C. Sakama. Extended well–founded semantics for paraconsistent logic programs. In *Fifth Generation Computer Systems*, pages 592–599. ICOT, 1992.
26. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, pages 221–230, 1990.
27. G. Wagner. A database needs two kinds of negation. In B. Thalheim, J. Demetrovics, and H-D. Gerhardt, editors, *MFDBS*, pages 357–371. Springer-Verlag, 1991.
28. G. Wagner. Reasoning with inconsistency in extended deductive databases. In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. on Logic Programming and NonMonotonic Reasoning*. MIT Press, 1993.