Optative Reasoning with Scenario Semantics

Luís Moniz Pereira
José Júlio Alferes

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

Abstract

The scenario semantics of extended logic programs builds upon simple primitive notions and has been shown to encompass many important logic program semantics. Here we introduce into the scenario semantics paradigm a general flexible notion of optative acceptance of acceptable hypotheses, and proceed to illustrate its application to fault diagnosis, taxonomic reasoning, and declarative logic program debugging.

Whereas the contradiction removal semantics approach has been shown applicable to a wide range of non-monotonic reasoning problems including the above, it is based either on minimally or on sceptically taking back the most primitive (closed world) assumptions contributing to contradiction.

In contrast, we show here how to achieve the same effect, and solve the same range of problems, by means of a contradiction avoiding rule of optative acceptance of hypotheses, easily defined in the present more general approach. Our counterpart to minimality in contradiction removal consists (quasi–complete) scenaria accepting all acceptable optative hypotheses short of contradiction. Our counterpart of contradiction removal scepticism consists in accepting all but only those hypotheses common to all such quasi–complete scenaria.

The close relationship to contradiction removal allows us to rely on our latter's implementation for expeditiously computing results.

Introduction

The scenario semantics paradigm of logic programs [4] has been recently expanded in [1] to encompass important logic programming semantics (including those extended with two kinds of negation - *"explicit negation"* and the traditionally named *"negation as failure"* (NAF)) [6, 8, 15, 18]. The scenario semantics paradigm, recapitulated below, is 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", "evidence contrary to a hypothesis", "admissible scenario", "completeness of a set of hypotheses wrt to a scenario", etc.

The ideal scenario semantics (ISS- cf. below) [1] is a most sceptical one, accepting all and only the accepted hypotheses compatible with all admissible scenaria. However, there is motivation to consider even more sceptical semantics, where some of the acceptable hypotheses might not in fact be accepted.

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, whenever causally deeper faults are to be prefered over hypothesized faults that are simply a consequence of the former: the latter cannot be hypothesized without the first. Moreover, problem specific and user defined preference criteria affecting acceptance of hypotheses may also come to bear. Another case in point is logic program debugging, where one wants to hypothesize about the primitive cause of a bug, and not about the bugginess of some clause if there is the possibility that that clause relies on a still buggy predicate. In general, the clauses 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.

Example 1 Consider this program, describing bycicle behaviour:

$$\begin{aligned}
\neg wobbly_wheel &\leftarrow not\ flat_tyre, not\ broken_spokes \\
flat_tyre &\leftarrow leaky_valve \\
flat_tyre &\leftarrow punctured_tube \\
\neg no_light &\leftarrow not\ faulty_dynamo
\end{aligned}$$

% Observation:
$wobbly\_wheel$

The ISS assigns to it the meaning:

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

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

In this paper, in order to respond to such epistemological requirements as above, we introduce into scenario semantics the general flexible notion of optative acceptance of hypotheses, and proceed to illustrate its application to both the problems of fault diagnosis [16] and declarative debugging of logic programs [7], using the formulations initially devised for them in [1].

Whereas the contradiction removal semantics approach of [10] has been shown applicable to a wide range of non-monotonic reasoning problems [11], including the applications in this paper [12, 13], it is based on minimally or sceptically taking back the most primitive (closed world) assumptions contributing to contradiction.

In contrast, we show how to achieve the same effect, and solve the same range of problems, by means of our contradiction avoiding rule of optative acceptance of hypotheses, easily defined in the present more general approach. Our counterpart to minimality in contradiction removal consists in accepting in (quasi–complete) scenaria all acceptable primal hypotheses short of contradiction. Our counterpart of contradiction removal or belief revision scepticism consists in accepting all but only those hypotheses common to all such quasi–complete scenaria. The close relationship to contradiction removal allows us to rely on our latter's present implementation for expeditiously computing results. In recent work [3, 9] we show the equivalence between contradiction avoidance as defined here and the contradiction removal approach of [10].

1 Scenaria for Extended Logic Programs

Next we review the scenario semantics paradigm of [1].

By extended logic program we mean a set of (ground) rules of the form:

$$L \leftarrow A_1, \ldots, A_n, not\ B_1, \ldots, not\ B_m \quad n, m \geq 0$$

where each of $L, A_1, \ldots, A_n, B_1, \ldots, B_m$ is an objective literal, i.e. an atom P or its explicit negation $\neg P$. A non–extended program does not comprise explicit negation and so reduces to a normal logic program.

1.1 Admissible Scenaria

In [4, 5] a normal logic program is viewed as an abductive framework where literals of the form *not* L (NAF–hypotheses or just hypotheses) can be considered as abducibles, i.e. they can be hypothesized. The set of all ground hypotheses is *not* $\mathcal{H}$, where $\mathcal{H}$ denotes the Herbrand base of the program and *not* prefixed to a set denotes the set obtained by prefixing *not* to each of its elements.

In order to introduce explicit negation, negated objective literals of the form $\neg A$ are treated, like in [1] and [6], as new atomic symbols. The Herbrand base $\mathcal{H}$ is then extended to the set of all such objective literals. Of course, this is not enough to correctly treat explicit negation. Relations among $\neg A$, A, and *not* A, must be established (cf. below). (A study of such relations is made in [2].)

Definition 1.1 (Scenario) *A scenario of an extended logic program* P *is the first order Horn theory* $P \cup H$*, where* $H \subseteq not\ \mathcal{H}$*.*

When introducing explicit negation into logic programs one has to reconsider the notion of hypotheses. As the designation "explicit negation" suggests, when a scenario $P \cup H \vdash \neg L$ it is *explicitly* stating that L is false in that scenario. Consequently the hypothesis *not* L is enforced in the scenario, and cannot optionally be held independently. This is the *"coherence principle"* [8], which relates both negations: i.er. $\neg L$ entails *not* L for any objective literal L.

There is a 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 $\neg\_L$. Intuitively, $\vdash$ is just the standard T_P operator of the Horn propositional programs obtained with the new symbols in place.

Definition 1.2 (Mandatory hypotheses wrt $P \cup H$) *The set of mandatory hypotheses wrt a scenario $P \cup H$ is:*

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

This notion allows us to incorporate the coherence principle into our semantics.

Definition 1.3 (Intended program) *Let P be an extended logic program. The intended program of P is the program obtained by replacing every rule of the form:* $L \leftarrow A_1, \ldots, A_n, not\ B_1, \ldots, not\ B_m \quad n, m \geq 0$
by another: $L \leftarrow A_1, not\ \neg A_1, \ldots, A_n, not\ \neg A_n, not\ B_1, \ldots, not\ B_m$
where $\neg A_i$ denotes the complement of A_i wrt explicit negation.

Note how, in the intended program, any true rule head has the effect of falsifying the body of all rules containing its complement literal wrt explicit negation. This provides the link between complementary literals. From now on, when refering to a program we always mean its intended version.

Definition 1.4 (Consistent scenario) *A scenario $P \cup H$ is consistent iff for all objective literals L such that $P \cup H \cup Mand(H) \vdash L$, neither not $L \in H \cup Mand(H)$ nor $P \cup H \cup Mand(H) \vdash \neg L$.*

Unlike non–extended logic programs, an extended logic program may in general have no consistent scenaria:

Example 2 Program $P = \{\neg p;\ \ p \leftarrow not\ p\}$ has no consistent scenario. Note that $P \cup \{\}$ is not consistent since $Mand(\{\}) = \{not\ p\}$ and $P \cup \{not\ p\} \vdash p$ as well as $\neg p$.

Definition 1.5 (Consistent program) *An extended logic program P is consistent iff $P \cup Mand(\{\})$ is a consistent scenario.*

Henceforth we restrict programs to consistent ones. Any program can be made consistent by adding a new unique hypothesis to each rule body.

Not every consistent scenario specifies a consensual semantics for a program [14]. For example [4] 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 hypothesis *not* L is contradicted only by objective literal L. Evidence for an objective literal L in a program P is any set of hypotheses which, if assumed, would entail L.

Definition 1.6 (Evidence for an objective literal L) *A subset E of not $\mathcal{H}$ is evidence for an objective literal L in a program P iff:*

$$E \supseteq Mand(E) \qquad \text{and} \qquad P \cup E \vdash L.$$

If P is understood and E is evidence for L we write $E \leadsto L$.

As in [1], a hypothesis is acceptable wrt a scenario iff there is no evidence to the contrary, i.e. iff all such evidence is itself defeated by the scenario:

Definition 1.7 (Acceptable hypothesis) *A hypothesis not L is acceptable wrt $P \cup H$, i.e. each evidence for L is defeated by $P \cup H$, iff:*

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

The set of all acceptable hypotheses wrt $P \cup H$ is denoted by $Acc(H)$.

In a consensual semantics we are interested in admitting only consistent scenaria whose hypotheses are either acceptable or mandatory. By definition of mandatory hypotheses any scenario includes its own.

Definition 1.8 (Admissible scenario) *A scenario $P \cup H$ is admissible iff it is consistent and $Mand(H) \subseteq H \subseteq Mand(H) \cup Acc(H)$.*

1.2 Ideal Sceptical and Complete Scenaria Semantics

In [1] it is guaranteed that by considering all admissible scenaria we do not fail to give semantics to consistent programs:

Proposition 1.1 *Any consistent program P has an admissible scenario.*

An admissible scenario refuses all unacceptable hypotheses. One semantics can be defined as the class of all admissible scenaria, the meaning of a program being determined, as usual, by the intersection of all such scenaria. However, since $P \cup Mand(\{\})$ is always the least admissible scenario, this semantics does not include any non–mandatory hypothesis. Consequently this semantics is equivalent to replacing every *not* L by the corresponding objective literal $\neg L$.

Example 3 Let $P = \{\neg p; \quad a \leftarrow not\ b\}$. The least admissible scenario is $P \cup \{not\ p\}$. Thus the literals entailed by the semantics are $\{\neg p, not\ p\}$. Note *not* b is not entailed by this extremely sceptical semantics.

The semantics of admissible scenaria is the most sceptical one for extended logic programs: it contains no hypotheses except for mandatory ones. In order to define more credulous semantics, one defines classes of scenaria based on proper subsets of the class of admissible scenaria, as governed by specific choice criteria. Constraining the set of admissible scenaria reduces undefinedness but may restrict the class of programs with semantics. Here we have particular interest in two of the semantics defined in [1]. One, ISS, is based on a notion of ideal scepticism, and provides semantics for every consistent program. The other, CSS, is a less sceptical semantics, but fails to give meaning to every consistent program.

Suppose the scenario $P \cup H$ represents such an *"ideal"* sceptical semantics, where an "admissible scenario" is one which is admissible for every rational reasoner. Let some such admissible scenario be $P \cup K$. It is clear that $P \cup K \cup H$ should again be admissible, since H must be part of any agent's semantics. This leads to an immediate definition of the *"ideal"* sceptical semantics.

Definition 1.9 (Ideal sceptical semantics) *A set of hypotheses H represents the ideal sceptical semantics, $ISS = P \cup H$, if it is the greatest set satisfying the condition:*

For each admissible scenario $P \cup K$, $P \cup K \cup H$ is again admissible.

If P is consistent then such a set always exists, consequence of the fact that the union of sets satisfying the above condition satisfies it too.

Example 4 Consider program $P = \{a \leftarrow not\ p;\ \neg a \leftarrow not\ q;\ c \leftarrow not\ r\}$. It is not difficult to see that $ISS = P \cup \{not\ r\}$. Hence we conclude c, despite the inconsistency potentially caused by the other rules.

ISS models a reasoner who assumes the program correct and so, whenever confronted with an acceptable hypothesis leading to an inconsistency he cannot accept such a hypothesis; i.e. he prefers to assume the program correct rather than assume that an acceptable hypothesis must be accepted (cf. example 4 above where both *not* p and *not* q are acceptable, but not accepted). It is easy to imagine a less sceptical reasoner who, confronted with an inconsistent scenario, prefers considering the program wrong rather than admitting that an acceptable hypothesis be not accepted. Such a reasoner is more confident in his acceptability criterium: 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 acceptance of each acceptable hypothesis.

In order to obtain a semantics modeling such a reasoner, in [1] a subclass of the admissible scenaria is defined, which directly imposes that acceptable hypotheses be indeed accepted:

Definition 1.10 (Complete scenaria) *A scenario $P \cup H$ is complete iff it is admissible and contains every acceptable hypothesis. In other words, a scenario $P \cup H$ is complete iff $H = Mand(H) \cup Acc(H)$.*

The complete scenaria semantics (CSS) of P is the set of all complete scenaria of P. As usual, the meaning of P is determined by the intersection of all such scenaria.

As expected, and in contradistinction to ISS, complete scenaria may not in general exist, even when P is consistent:

Example 5 $P = \{\neg a \leftarrow not\ b; \quad a \leftarrow not\ c\}$ has several admissible scenaria, resulting from the union of P with each of: $\{\}$, $\{not\, b\}$, $\{not\, c\}$, $\{not\, a, not\, b\}$, and $\{not\ \neg a, not\ c\}$. None is complete. For example, $P \cup \{not\ \neg a, not\ c\}$ is not complete because *not b* is acceptable wrt that scenario.

Definition 1.11 (Contradictory program) *A program is contradictory iff it has no complete scenaria.*

Properties of this semantics, and its relation to other semantics can be found in [1], where CSS is proven equivalent to the well–founded semantics with explicit negation WFSX [8]. Another important result stated there is that the least complete scenario can be obtained as the least fixpoint of an iterative bottom–up construction.

2 Complete Scenaria wrt Optative Hypotheses

In CSS every acceptable hypothesis must be accepted. Consequently some programs might not have a meaning. In ISS some acceptable hypotheses might not be accepted in order to avoid inconsistency. However, as shown in example 1, ISS has no the control over which acceptable hypotheses are not accepted. Conceivably, any acceptable hypotheses may or may not actually be accepted, in a discretionary way.

It is clear from example 1 that we wish to express that only the hypotheses *not broken_spokes*, *not leaky_valve*, *not faulty_dynamo* and *not punctu red_tube* may be optative, i.e. possibly accepted or not, if at all acceptable. The acceptance of hypotheses like *not flat_tyre* is to be determined by other hypotheses; and so we wish them always to be accepted, once acceptable.

Thus we distinguish between the *optative hypotheses*, or optatives, and non–optative ones. That distinction made, we can conceive of scenaria that might not be complete wrt the optatives.

Definition 2.1 (Optative hypotheses) *Those in some $Opt \subseteq not\ \mathcal{H}$.*

In general, when not accepting some optative hypothesis *not L*, i.e. when not assuming the falsity of L, then some otherwise acceptable hypotheses become unacceptable, because of the defined sense below that program models where the optative is true are not ruled out.

Example 6 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 it being true must be considered and so a cannot be assumed false; $P \cup \{b\} \vdash a$ counts as evidence against $not\ a$:

Definition 2.2 (Acceptable hypotheses wrt Opt) *A hypothesis $not\ L$ is acceptable wrt scenario $P \cup H$ and optatives Opt iff $not\ L$ is acceptable both wrt $P \cup H$ and $P \cup H \cup F$, where F is the set of complements of acceptable Opts which were not accepted; i.e.$F = not\ ((Opt \cap Acc(H)) - H)$.*

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

Example 7 In example 6 $Acc_{Opt}(\{not\ p\}) = \{\}$. $not\ b$ is not acceptable because, even though acceptable wrt $P \cup \{not\ p\}$, it is not acceptable wrt $P \cup \{not\ p\} \cup \{b\}$. The same happens with $not\ a$.

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

Definition 2.3 (Complete scenario wrt Opt) *$P \cup H$ is a complete scenario wrt a set of optatives Opt iff it is consistent, and for each $not\ L$:*

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

where (i) and (ii) simply express admissibility.

Let S be a complete scenario wrt Opt. A hypothesis in Opt acceptable wrt $P \cup H$ but leading to an inconsistent scenario if added to it, is not accepted in S so as to preserve consistency. This amounts to contradiction avoidance.

Example 8 Recall the wobbly wheel example 1. If $Opt = \{\}$ there is no complete scenario because of inconsistency. If, with obvious abbreviations, $Opt = \{not\ bs, not\ lv, not\ pt,\ not\ fd\}$ complete scenaria wrt Opt are:

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

Some of these scenaria 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 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 2.4 (Avoidance set) *Let $P \cup H$ be a complete scenario wrt Opt. The avoidance set of $P \cup H$ is $(Opt \cap Acc(H)) - H$.*

Example 9 The avoidance set of the first scenario in example 8 is $\{not\ lv, not\ pt, not\ fd\}$ and of the second one is $\{not\ lv, not\ pt\}$.

In keeping with the sceptic vocation of WFSX, consider those scenaria which are minimal for some given avoidance set.

Definition 2.5 (Base scenario wrt Opt) *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 10 Consider $P = \{a \leftarrow not\ b;\ b \leftarrow not\ a;\ c \leftarrow not\ d;\ \neg c\}$ with $Opt = \{not\ d\}$.

Complete scenaria 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.

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

Definition 2.6 (Quasi–complete scenario wrt Opt) *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 11 In example 8 the quasi–complete scenaria wrt Opt are:
$\{not\ \neg ww, not\ fd, not\ bs,\ not\ lv\}$, $\{not\ \neg ww, not\ fd, not\ bs, not\ pt\}$, and $\{not\ \neg ww, not\ fd, not\ lv, not\ pt, not\ ft\}$.

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*.

Proposition 2.1 *The base scenaria wrt Opt form a lower semi–lattice under set inclusion.*

The quasi–complete scenaria are pairwise incompatible. So the well-founded semantics, being sceptical, is their meet scenario in the semi–lattice of proposition 2.1. Its avoidance set is the union of their avoidance sets.

Definition 2.7 (Well–founded semantics wrt Opt) *The well–founded semantics of a program P is the meet of all quasi–complete scenaria wrt Opt in the semi–lattice of all base scenaria. For short we use WFS_{Opt} to denote it.*

Example 12 In example 8 $WFS_{Opt} = P \cup \{not\ \neg ww, not\ fd\}$. 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 ISS, shown in example 1.

To obtain less sceptical complete scenaria wrt Opt, and in the spirit of partial stable models [15], we introduce:

Definition 2.8 (Partial scenario wrt *Opt***)** *Let the well–founded semantics of a program 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 13 The partial scenaria of P wrt *Opt* in example 8 are the union of P with each of:

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

The first is the WFS_{Opt} (cf. example 12), corresponding to the most sceptical view whereby all possibly relevant faults are assumed. The other extended scenaria 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 CSS 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 [1]).

Theorem 2.1 (Relation to WFSX) *If WFSX is defined for a program P then, for whatever Opt, WFS_{Opt} is the least complete scenario of P.*

For programs without explicit negation WFSX is equivalent [8] to the well–founded semantics of [17].

3 On Primal Optative Reasoning

In the previous section no restriction whatsoever was enforced regarding the optatives of programs. It is possible however that optatives be given by the user along with the program, or particular classes of them desired.

Next we identify a special class of optatives, governed by an important motivation: only hypotheses that do not depend on any other may be optative. 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 $S \subseteq H$ had to be accepted first. Thus *not* L depends on the hypotheses of S, and the latter are more primal than *not* L. As a first approximation, let us define the set of prime optatives as $Acc(\{\})$.

Example 14 Take $P = \{a \leftarrow not\, b;\quad b \leftarrow not\, c;\quad c \leftarrow not\, d\}$.

The only prime optative is *not* d. Hypothesis *not* b is not prime optative since it depends on hypothesis *not* d, i.e. it is only acceptable once *not* d is accepted.

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

Example 15 Let $P = \{a \leftarrow b;\quad b \leftarrow c;\quad p \leftarrow not\ a;\quad \neg p\}$.

$Acc(\{\}) = \{not\ a,\ not\ b, not\ c\}$, and the well–founded semantics 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 for the acceptance of *not* a.

Now, any definition of a semantics based on the notions of scenaria and evidence alone cannot identify the optative primacy of *not* c, because it is insensitive to ungrounded literals.

An assymmetry must be introduced for this, based on a separate new notion, that captures the causal directionality of inference implicit in logic program rules mentioned in the introduction:

Definition 3.1 (Sensitive hypotheses) *A hypothesis* $not\ A \in Acc(\{\})$ *is sensitive to a separate set of hypotheses* $not\, S$ *in* P *iff* $not\ A \notin Acc(P \cup S)$.

Definition 3.2 (Prime Optatives) *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 some element of* $not\ S$ *is sensitive to* $not\ A$.

The set of all prime optatives is denoted by $POpt$.

As a shorthand we refer to the well–founded semantics wrt the set of prime optatives as the *prime optative semantics*, or $\mathcal{POS}$.

Example 16 In example 15 the only prime optative is *not* c. For instance, *not* a is not prime optative since *not* a is sensitive to *not* b and *not* b is not sensitive to *not* a.

Example 17 In the wobbly wheel example $\mathcal{PO}pt = \{not\ bs, not\ pt, not\ lv\}$. It coincides with $Acc(\{\})$, as no hypothesis in it is sensitive to any other.

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

$Acc(\{\}) = \{not\ a, not\ b, not\ d\}$ and all these are prime optatives: $not\ d$ is prime optative because it is insensitive to other hypotheses; $not\ b$ is prime optative 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, even if acceptable, one may fail to give meaning to some consistent programs, because there are less options for avoiding inconsistency.

Example 19 Consider $P = \{e \leftarrow not\ d;\ d \leftarrow not\ c;\ c \leftarrow not\ b;\ \neg e;\ \neg b\}$

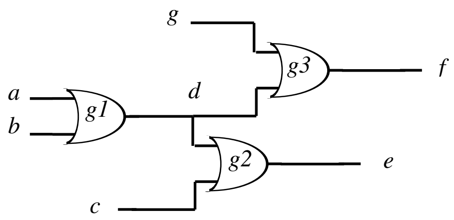
$\mathcal{P}Opt = Acc(\{\}) = \{not\ b\}$. However, no complete scenario wrt $\mathcal{P}Opt$ exists, and thus neither ISS wrt $\mathcal{P}Opt$ nor $WSF_{\mathcal{P}Opt}$ are defined. Note that ISS is $P \cup \{not\ b, not\ e\}$.

4 Applications

Finally we show examples of application of $\mathcal{POS}$ to diagnosis, to taxonomies, and to the debugging of pure Prolog programs.

4.1 Diagnosis

Consider the circuit below comprised of three OR gates:



and represented by the extended logic program:

% Correct behaviour of OR gates:

$gate(or, G, 0, 0, 0) \leftarrow not\ ab(G)$ $\quad$ $gate(or, G, 1, 0, 1) \leftarrow not\ ab(G)$

$gate(or, G, 0, 1, 1) \leftarrow not\ ab(G)$ $\quad$ $gate(or, G, 1, 1, 1) \leftarrow not\ ab(G)$

% Connections among nodes:

$value(d, D) \leftarrow value(a, A), value(b, B), gate(or, g1, A, B, D)$

$value(e, E) \leftarrow value(d, D), value(c, C), gate(or, g2, D, C, E)$

$value(f, F) \leftarrow value(d, D), value(g, G), gate(or, g3, D, G, F)$

% No node can simultaneously have the values 1 and 0:

$\neg value(X, 0) \leftarrow value(X, 1)$

$\neg value(X, 1) \leftarrow value(X, 0)$

To this program we add facts F reporting observations made. Suppose we observe the inputs $\{value(a, 1), value(b, 0), value(c, 0), value(g, 0)\}$. $P \cup F$ is non–contradictory, and thus $\mathcal{POS}$ is equivalent to CSS. In particular all:

$$\{not\ ab(g1), not\ ab(g2), not\ ab(g3), value(d,1), value(f,1), value(e,1)\}$$

are true, and so the behaviour of every gate is normal. The values of the nodes are as expected, and all belong to the consequences of $\mathcal{POS}$, $Con(\mathcal{POS})$.

Suppose an additional factual observation O_1 is made: $value(e,0)$. Then the program $P \cup F \cup O_1$ is contradictory, and so CSS doesn't assign meaning to it. However $\mathcal{POS}$ and partial scenaria exist, and the maximal partial scenaria (XS_1 and XS_2) correspond to minimal diagnoses:

$$\{not\ ab(g2), not\ ab(g3), value(e,0)\} \subseteq Con(XS_1)$$

$$\{not\ ab(g1), not\ ab(g3), value(d,1), value(f,1), value(e,0)\} \subseteq Con(XS_2)$$

where $not\ ab(g1) \notin XS_1$ and $not\ ab(g2) \notin XS_2$.

Finally suppose that yet an additional observation O_2, at node d, is made with result $value(d,0)$. Then the only partial scenario corresponds to the only possible diagnosis:

$$\{not\ ab(g2), not\ ab(g3), value(d,0), value(f,0), value(e,0)\} \subseteq Con(\mathcal{POS})$$

4.2 Taxonomies

Consider the statements: *(i) Normally birds fly. (ii) Normally penguins don't fly. (iii) Penguins are birds. (iv) Tweety is a penguin. (v) Duffy is a bird.*

(iv) and (v) are expressed by the two facts $p(t)$ and $b(d)$, and (iii) is expressed by the rule $b(X) \leftarrow p(X)$. To represent defeasible statements (such as (i) and (ii)) we introduce abnormality predicates negated in the body of rules, i.e. (i) is expressed by $f(X) \leftarrow b(X), not\ ab_1(X)$ and (ii) is expressed by $\neg f(X) \leftarrow p(X), not\ ab_2(X)$. In order to correctly capture the statements above we must also introduce a rule stating that the more specific information overrides the more general one in case of conflict: viz. it is possible to apply (ii) to a penguin then rule (i) should not be applied, i.e. $ab_1(X) \leftarrow p(X), not\ ab_2(X)$. This marks a preference between the defeasible rules. In summary, the program representing the statements above is P :

$$\begin{array}{lll} f(X) \leftarrow b(X), not\ ab_1(X) & b(X) \leftarrow p(X) & b(d) \leftarrow \\ \neg f(X) \leftarrow p(X), not\ ab_2(X) & ab_1(X) \leftarrow p(X), not\ ab_2(X) & p(t) \leftarrow \end{array}$$

The consequences of $\mathcal{POS}$ of P are, as expected:

$$\{b(d), f(d), not\ ab_1(d), not\ ab_2(d),\quad p(t), b(t), \neg f(t), ab_1(t), not\ ab_2(t)\}$$

4.3 Debugging of pure Prolog

In pure Prolog programs, besides looping there are only two other kinds of error, see [7]: wrong solutions or finitely missing solutions. First we apply prime optative semantics to perform debugging of wrong solutions of pure Prolog programs, assuming that a Prolog program stands for its ground version. For an elaboration of these techniques the reader is refered to [12, 13].

Consider the buggy Prolog program P:

$$
\begin{array}{lll}
a(1)\leftarrow & b(2)\leftarrow & c(1,X)\leftarrow \\
a(X)\leftarrow b(X), c(Y,Y) & b(3)\leftarrow & c(2,2)\leftarrow
\end{array}
$$

As you can check, goal $a(2)$ succeeds with the above program. Suppose now that $a(2)$ should not be a conclusion of P, so that $a(2)$ is a wrong solution. What are the minimal causes of this bug?

There are three. First, the obvious one, the second rule for a has a bug; the second is that $b(2)$ should not hold in P; and finally, that neither $c(1,X)$ nor $c(2,2)$ should hold in P.

This type of error (and its causes) is easily detected using prime optative semantics by means of a simple transformation applied to the original program: add $not\ ab_i(X_1, X_2, \ldots, X_n)$ to the body of each i-th rule of P, where n is its arity and $X_1, X_2, \ldots, X_n$ its head arguments.

Applying this to P we obtain P_1:

$$
\begin{array}{lll}
a(1)\leftarrow not\ ab_1(1) & b(2)\leftarrow not\ ab_3(2) & c(1,X)\leftarrow not\ ab_5(1,X) \\
a(X)\leftarrow b(X), c(Y,Y), not\ ab_2(X) & b(3)\leftarrow not\ ab_4(3) & c(2,2)\leftarrow not\ ab_6(2,2)
\end{array}
$$

Now if we have wrong solution $p(X_1, X_2, \ldots, X_n)$ in P just add to program P_1 fact $\neg p(X_1, X_2, \ldots, X_n)$, and to find the possible causes of the wrong solution using as (prime) optatives all the $not\ ab_i$ hypotheses.

For instance, if $a(2)$ is a wrong solution of program P, by adding $\neg a(2)$ to P_1 we obtain three maximal partial scenaria of P_1 : one by not accepting $not\ ab_1(2)$; another by not accepting $not\ ab_3(2)$; and finally another by not accepting both $not\ ab_5(1,1)$ nor $not\ ab_5(2,2)$.

Suppose now a program should not finitely fail on some goal but does so. This is the missing solution problem. Say, for instance, $a(4)$ should succeed in P above. Which are the minimal sets of rules that added to P make $a(4)$ succeed? There are two solutions: either add rule $a(4)$ or rule $b(4)$.

The solution to this type of bug is trickier than the one before, but it suffices to introduce for each predicate p with arity n the following rule:

$$p(X_1, X_2, \ldots, X_n) \leftarrow missing(p(X_1, X_2, \ldots, X_n))$$

Then all that's necessary is to add the fact $\neg\bot$, plus

$$\bot \leftarrow not\ q(X_1, X_2, \ldots, X_n)$$

to state that if predicate q has a missing solution $q(X_1, X_2, \ldots, X_n)$ then a potential contradiction arises, and obtain the complete scenaria of the transformed program wrt $\{not\ missing(A) \mid A\ is\ an\ atom\}$.

The transformed program P_2 obtained is P plus the rules:

$$
\begin{array}{ll}
\neg\bot & b(X) \leftarrow missing(b(X)) \\
a(X) \leftarrow missing(a(X)) & c(X,Y) \leftarrow missing(c(X,Y))
\end{array}
$$

To find the possible causes of the missing solution to $a(4)$ we add rule $\bot \leftarrow not\ a(4)$ and find, as expected, two maximal partial scenaria: one not containing $not\ missing(a(4))$, and another not containing $not\ missing(b(4))$. This means that by adding $a(4)$ or $b(4)$ to P the missing solution is no longer missed.

Acknowledgments

We thank ESPRIT BRA COMPULOG 2 and JNICT Portugal for their support.

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 LPNMR*. MIT Press, 1993.

[2] J. J. Alferes and L. M. Pereira. On logic programs semantics with two kinds of negation. In K. Apt, editor, *IJCSLP*. MIT Press, 1992.

[3] J. J. Alferes and L. M. Pereira. Contradiction: when avoidance equals removal. Part I. In *4th Int. Ws on Extensions of L.P.* University of St. Andrews, 1993.

[4] P. M. Dung. Negation as hypotheses: An abductive framework for logic programming. In K. Furukawa, editor, *8th ICLP*, pages 3–17. MIT Press, 1991.

[5] P. M. Dung, A. C. Kakas, and P. Mancarella. Negation as failure revisited. Technical report, 1992. Preliminary Report.

[6] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In D. Warren and P. Szeredi, editors, *7th ICLP*, pages 579–597. MIT Press, 1990.

[7] J. Lloyd. Declarative error diagnosis. *New Generation Computing*, 5(2), 1987.

[8] L. M. Pereira and J. J. Alferes. Well founded semantics for logic programs with explicit negation. In B. Neumann, editor, *10th ECAI*, pages 102–106. John Wiley & Sons, 1992.

[9] L. M. Pereira and J. J. Alferes. Contradiction: when avoidance equals removal. Part II. In *4th Int. Ws on Extensions of L.P.* University of St. Andrews, 1993.

[10] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Contradiction removal semantics with explicit negation. In *Applied Logic Conf.* ILLC, Amsterdam, 1992.

[11] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Logic programming for nonmonotonic reasoning. Technical report, AI Centre, Uninova, 1992. Submitted.

[12] L. M. Pereira, C. Damásio, and J. J. Alferes. Debugging by diagnosing assumptions. Technical report, AI Centre, Uninova, March 1993.

[13] L. M. Pereira, C. Damásio, and J. J. Alferes. Diagnosis and debugging as contradiction removal. In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. on LPNMR*. MIT Press, 1993.

[14] D. Poole. A logical framework for default reasoning. *AI*, 36:27–47, 1988.

[15] T. Przymusinski. Extended stable semantics for normal and disjunctive programs. In D. Warren and P. Szeredi, editors, *7th ICLP*, pages 459–477. MIT Press, 1990.

[16] R. Reiter. A theory of diagnosis from first principles. *AI*, 32:57–96, 1987.

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

[18] 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.