Scenario Semantics of Extended Logic Programs

José Júlio Alferes[*]
Phan Minh Dung[†]
Luís Moniz Pereira[*]
[*]CRIA Uninova and DCS, U.Nova de Lisboa
2825 Monte da Caparica, Portugal
({jja,lmp}@fct.unl.pt)
[†]Division of C.S., A.I.T., GPO Box 2754,
Bangkok 10501, Thailand,
(dung@cs.ait.ac.th)

Abstract

We present a coherent, flexible, unifying, and intuitive framework for the study of explicit negation in logic programs, based on the notion of admissible scenaria and the "coherence principle". With this support we introduce, in a simple way, a proposed "ideal sceptical semantics", as well as its well–founded counterpart.

Another result is a less sceptical "complete scenaria semantics", and its proof of equivalence to the well–founded semantics with explicit negation (WFSX). This has the added benefit of bridging complete scenaria to default theory via WFSX, defined here based on Gelfond–Lifschitz Γ operator..

Finally, we characterize a variety of more and less sceptical or credulous semantics, including answer–sets, and give sufficient conditions for equivalence between those semantics.

Introduction

In general, approaches to semantics follow two major intuitions: scepticism and credulity [30]. In logic programming, the credulous approach includes such semantics as stable semantics [7] and preferred extensions [3], while well–founded semantics [31] is the sole representative of scepticism [3].

Recently, several authors have stressed and shown the importance of including a second kind of negation in logic programs, for use in deductive databases, knowledge representation, and non–monotonic reasoning [8, 9, 10, 11, 13, 21, 22, 23, 24, 32].

Different semantics for logic programs extended with an explicit negation (extended logic programs) have appeared [6, 8, 11, 15, 17, 19, 26, 27, 28, 32]. Many of these semantics are either a generalization of stable models semantics [7] or of well–founded semantics (WFS) [31] (cf. [1] for a comparison).

Others are based on constructive logic [12, 13, 14].

While generalizations of stable models semantics are clearly credulous in their approach, no semantics whatsoever has attempted to seriously explore the sceptical approach. A closer look at the works generalizing well-founded semantics [6, 15, 17, 19, 26, 27, 28] shows these generalizations to be rather technical in nature, where the different techniques introduced to characterize the well–founded semantics of normal logic programs are slightly modified in some way to become applicable to the more general case.

Our first contribution is the presentation of a coherent, flexible, unifying, and more intuitive framework for the study of an explicit second kind of negation in logic programs, based on the notion of admissible scenaria. This framework extends the approach proposed in [3] for normal logic programs, and adopts the *"coherence principle"* of [15]. This principle is easily illustrated with an example:

Example 1 Consider a program containing the rules:

$$\neg driversStrike \leftarrow \qquad\qquad tryBus \leftarrow not\ \ driversSrike$$

advising to plan a trip by bus if there is no reason to assume the bus drivers are on strike, and bus drivers are not on strike. No matter what the rest of the program is (assuming it is consistent on the whole), it is clear that a rational agent has no reason to believe the drivers are on strike, and of course he plans his trip by bus.

In general, the coherence principle states that, for any objective literal L, if $\neg L$ follows within a semantics then $not\, L$ must also be entailed by that semantics[1].

Approaches not taking this principle into account [6, 26, 28] try to transform the program into an "equivalent" normal program version, and then discard any "contradictory" models brought about by explicit negation. The connexion between both kinds of negation is totally absent.

The second contribution is the presentation, in a simple way, of a proposed ideal sceptical semantics and its well–founded (or grounded) part; in fact an entirely declarative semantics able to handle programs like:

$$a \leftarrow not\, p \qquad b \leftarrow not\, r \qquad \neg a \leftarrow not\, q$$

and assigning it the semantics $\{b, not\, r\}$.

Most semantics cited above cannot deal with such programs because, as neither p nor q have rules, they all assume both $not\, p$ and $not\, q$ without regard to the ensuing contradiction, except as an after–the–fact filter. Others [11, 17, 19, 28] treat these programs as contradictory ones, upon which they proceed to excise contradiction.

In our ideal sceptical semantics this program is not contradictory at all. Indeed, the assumptions ($not\, p$ and $not\, q$) are not even accepted, let alone removed.

A third, multiple, result is the introduction of complete scenaria semantics, a less sceptical semantics than the previous one, and its formal equivalence to WFSX [15]. This has the added benefit result of bridging complete scenaria semantics to default theory via WFSX [20]. This equivalence is especially interesting inasmuch as it shows quite different ways of handling the same semantics, and as WFSX is here first defined as the fixpoint of a Gelfond–Lifschitz Γ–like operator.

The final result is the characterization of a variety of more and less sceptical or credulous semantics, including answer–sets [8]. We also give sufficient conditions for the equivalence between these semantics. The notion of "evidence to the contrary" is pervasive, in that it can be tuned to offer such a variety.

1 Admissible Scenaria for Extended Programs

In this section we generalize the notions of scenario and evidence for normal logic programs given in [3], to those extended with explicit negation. They are reminiscent of the notions of scenario and extension of [25].

By extended logic program we mean a set of (ground) rules of the form $L \leftarrow A_1, \ldots, A_n, not\, B_1, \ldots, not\, B_m$ $(n, m \geq 0)$ 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.

In [3, 5] a normal logic program is viewed as an abductive framework where literals of the form $not\ L$ (NAF-hypotheses) can be considered as abducibles, i.e they must be hypothesized. The set of all ground NAF-hypotheses is $not\,\mathcal{H}$, where $\mathcal{H}$ denotes the Herbrand base of the program, as usual, and *not* prefixed to a set denotes the set obtained by prefixing *not* to each of its elements.

In order to introduce explicit negation we first consider negated objective literals of the form $\neg A$ as new symbols (as in [7]). The Herbrand base is now 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, as in the definitions below.

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 NAF-hypotheses. As the designation "explicit negation" suggests, when a scenario $P \cup H$ entails $\neg A$, it is *explicitly* stating that A is false in that scenario. Thus the NAF-hypothesis $not\, A$ is enforced in the scenario, and cannot optionally be held independently (cf. example 1). This is the *"coherence principle"* [15], which relates both negations.

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

$$Mand(H) = \{not\ L \mid P \cup H \cup \{not\ L \leftarrow \neg L \mid L \in \mathcal{H}\} \vdash not\ L\}^2$$

Alternatively, the set of mandatory hypotheses wrt $P \cup H$ is the smallest set $Mand(H)$ such that $Mand(H) = \{not\ L \mid P \cup H \cup Mand(H) \vdash \neg L\}$.

Example 2 Let $P = \{q \leftarrow not\,r;\ \neg r \leftarrow not\,p;\ \neg p\}$. Then:

$$Mand(\{\}) = \{not\,p, not\,r, not\,\neg q\}.$$

Example 3 Consider now a program containing the rules:

$$\neg driversStrike \leftarrow \qquad newsAboutStrike \leftarrow driversStrike$$

stating that newspapers publish news about the strike if the drivers are on strike, and that the bus drivers are definitely not on strike. For a rational reasoner the second rule should not provide a pretext for newspapers to publish news about a strike by possibly assuming it, since indeed the first rule (or some other) may actually state or conclude the contrary of that assumption.

In other words, any objective literal L in the body of a rule is to be considered shorthand for the conjunction $L, not\,\neg L$. This allows for technical simplicity in capturing the relation between $\neg L$ and $not\ L$:

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.

From now on, whenever refering to a program we always mean its intended version. In all examples we expressly use the intended program. Note how, in the intended program, any true rule head has the effect of falsifying the body of rules containing its complement literal wrt explicit negation.

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 the case of non–extended logic programs, an extended logic program may in general have no consistent scenaria.

Example 4 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 $\neg p$.

A notion of program consistency is needed. Intuitively, a program is consistent iff it has some consistent scenario. Because, for any H, if $P \cup H$ is consistent then $P \cup \{\} \cup Mand(\{\})$ is also consistent, we define:

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

From now on, unless otherwise stated, all programs are consistent.

Not every consistent scenario specifies a consensual semantics for a program [25]. For example [3] 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, what we wish to express is that a NAF-hypothesis can be assumed only if there can be no evidence to the contrary.

Clearly a NAF-hypothesis $not\ L$ is only contradicted by the objective literal L. Evidence for an objective literal L in a program P is a set of NAF-hypotheses which, if assumed in P together with its mandatories, 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:*

$$E \supseteq Mand(E) \qquad \text{and} \qquad P \cup E \vdash L^3.$$

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

As in [3] a NAF-hypothesis is acceptable wrt a scenario iff there is no evidence to the contrary, i.e. iff all evidence to the contrary is itself defeated by the scenario:

Definition 1.7 (Acceptable NAF-hypothesis) *A NAF-hypothesis $not\, L$ is acceptable wrt $P \cup H$ iff:*

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

i.e. each evidence for L is defeated by $P \cup H$.

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

In a consensual semantics we are interested only in admitting consistent scenaria whose NAF-hypotheses are either acceptable or mandatory. By definition of mandatory NAF-hypotheses it is clear that any scenario includes all its mandatory hypotheses.

Definition 1.8 (Admissible scenario) *A scenario $P \cup H$ is admissible iff it is consistent and:*

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

We must guarantee that by considering all admissible scenaria we do not fail to give semantics to consistent programs, i.e.:

Proposition 1.1 *Any consistent program has at least an admissible scenario.*

The notion of admissible scenario discards all NAF-hypotheses which are unacceptable, whatever the semantics of extended logic programs to be defined. One semantics based on that notion can be defined as the class of all admissible scenaria, where the meaning of a program being, as usual, determined 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 NAF-hypothesis. Consequently it is equivalent to replacing every $not\, L$ by the corresponding objective literal $\neg L$.

Example 5 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[4]. 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 having a semantics. In the next sections we define a sequence of semantics which, by restricting the set of admissible scenaria, are more credulous but give meaning to narrower classes of programs.

2 A Sceptical Semantics for Extended Programs

Several attempts, mentioned in the introduction, have been made to generalize well–founded semantics to logic programs with explicit negation. But on closer look these generalizations are of a rather technical nature, where different techniques introduced to characterize the well–founded semantics for normal logic programs are modified to become applicable to the more general case. So it would not be surprising if tomorrow some new "well–founded" semantics for programs with explicit negation were to be presented. So which of them is really "well–founded"? And what is the essential difference between them? How many "well–founded" semantics are we going to have? After all, what makes a semantics "well–founded"? Certainly not just because it is in some way "technically" similar to one or other presentation of the well–founded semantics of Van Gelder et al. [31][5].

The intuition behind well–founded semantics is scepticism. So it is natural and important to ask the question of what is an ideally sceptical semantics for explicit negation, *i.e. one which would be part of the semantics of every rational reasoner.*

Suppose that $P \cup H$ is this *"ideal"* sceptical semantics. In the previous section, we have introduced and argued that an "admissible scenario" represents a scenario which is admissible for a rational reasoner. Let one such admissible scenario be $P \cup K$. It is clear that $P \cup K \cup H$ is again admissible since H must be part of this agent's semantics. This leads to an immediate definition of the *"ideal"* sceptical semantics.

Definition 2.1 (Ideal sceptical semantics) *A set of NAF-hypotheses H is called the ideal sceptical semantics, ISS, if it is the greatest set satisfying the condition:* "For each admissible scenario $P \cup K$, $P \cup K \cup H$ is again admissible".

It is clear that if P is consistent such a set exists, consequence of the fact that the union of sets satisfying the above condition satisfies it too.

Example 6 For $P = \{a \leftarrow not\, p;\ \neg a \leftarrow not\, q;\ c \leftarrow not\, r\}$, $ISS = \{not\, r\}$. So we are able to conclude c despite the potential inconsistency.

The most distinguishing feature of ISS is its striking simplicity, which may seem nearly trivial. Such simplicity is clearly a sure sign that our semantics naturally captures the intuitions behind scepticism.

A well–founded semantics is next construable as the *grounded* part of the ideal sceptical semantics. Indeed, in the case of normal programs, the truly or ideally sceptical semantics is determined as the greatest lower bound of all preferred extensions [3], well-founded semantics being the grounded part of that ideal sceptical semantics. This corroborates the intuitions of other related fields, where a distinction is made between restricted scepticism and ideal scepticism [29][6]. In this context, in order to define the well–founded semantics for programs with explicit negation all we need is introduce the grounded part of ideal scepticism:

Definition 2.2 (Well–founded semantics for extended programs) *Let P be an extended logic program whose ideal sceptical semantics is $P \cup H$. First define a transfinite sequence $\{K_\alpha\}$ of sets of NAF-hypotheses of P :*

$$\begin{array}{rcl} K_0 & = & \{\} \\ K_{\alpha+1} & = & K_\alpha \cup (H \cap MA(K_\alpha)) \end{array}$$

where $MA(K_\alpha)$ denotes $Mand(K_\alpha) \cup Acc(K_\alpha)$. The well–founded (sceptical) semantics of P, WFS0, is defined as $P \cup K$, where $K = \bigcup_\alpha K_\alpha$.

NAF-hypotheses belonging to WFS0 belong perforce to ISS, since that is imposed at each step of the above process, and are also grounded in the sense that they are obtained by this bottom–up process, starting from $\{\}$.

Example 7 Let $P = \{a \leftarrow not\, a;\quad a \leftarrow not\, b;\quad b \leftarrow not\, a\}$. $ISS = \{not\, b\}$ and $WFS0 = \{\}$. $not\, b$ is not grounded.

Theorem 2.1 *WFS0 is defined uniquely for every consistent program.*

3 The Semantics of Complete Scenaria

In this section we present a semantics less sceptical than WFS0. We call it *"Complete scenaria semantics"* (CSS for short). Then we exhibit and prove some properties of CSS.

For non–extended programs every acceptable hypothesis can be accepted. In extended programs an acceptable hypotheses may fail to be accepted, because of verified contradiction.

Example 8 Let $P = \{\neg a; \quad a \leftarrow not\, b\}$. The NAF-hypothesis $not\, b$ is acceptable wrt every scenario of P. However, by accepting $not\, b$ the program becomes inconsistent. Thus $not\, b$ can never be accepted. In a semantics like WFS0 such NAF-hypotheses are not accepted.

ISS and WFS0 model 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 6 where both $not\, p$ and $not\, q$ are acceptable, but not accepted). We can also view this reasoner as one who has a more global notion of acceptability. For him, as usual, an hypothesis can only be acceptable if there is no evidence to the contrary, but if by accepting it (along with others) a contradiction arises, then that counts as evidence to the contrary.

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. This position is justified by the stance that acceptance be grounded on the absence of specific contrary evidence rather than on the absence of global non–specific evidence to the contrary[7].

In order to define a semantics modeling such a reasoner we begin by defining a subclass of the admissible scenaria which directly imposes that acceptable NAF-hypotheses be indeed accepted.

Definition 3.1 (Complete scenario) *A scenario $P \cup H$ is complete iff is consistent, and $H = Mand(H) \cup Acc(H)$.*

Example 9 The only complete scenario of $P = \{\neg b \leftarrow;\ b \leftarrow not\, c;\ c \leftarrow not\, c;\ a \leftarrow b, not\, \neg b\}$ is $P \cup \{not\, a, not\, b\}$. In fact: the mandatory hypotheses of that scenario are $\{not\, b\}$; $not\, a$ is acceptable because $not\, \neg b$ belongs to every evidence for a, and $\neg b$ is entailed by the scenario; $not\, c$ is not acceptable because $\{not\, c\}$ is evidence for c. Since every acceptable or mandatory hypothesis is in the scenario, and every hypothesis in the scenario is either acceptable or mandatory, the scenario is complete. Remark that if $not\, \neg b$ were not part of the last rule, as required by definition 1.3 of intended program, then *not a* would not be acceptable.

As expected, and in contradistinction to WFS0, complete scenaria may not in general exist, even when P is consistent.

Example 10 $P = \{\neg a \leftarrow not\, b;\ a \leftarrow not\, c\}$ has several admissible scenaria: $\{\}$, $\{not\, b\}$, $\{not\, c\}$, $\{not\, a, not\, b\}$, and $\{not\, \neg a, not\, c\}$. None is complete.

Definition 3.2 (Contradictory program) *A program is contradictory iff it has no complete scenaria.*

Definition 3.3 (Complete scenaria semantics) *Let P be a non–contradictory program. 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.*

3.1 Properties of Complete Scenaria

Next we study properties and present a fixpoint operator for this semantics.

Theorem 3.1 *Let $CS_P \neq \{\}$ be the set of all complete scenaria of P. Then:*

1. *CS_P is a downward–complete semilattice, i.e. each nonempty subset of CS_P has a greatest lower bound.*
2. *There exists a least complete scenario.*
3. *In general, CS_P is not a complete partial order*[8].

Definition 3.4 (Well founded complete scenario) *Let P be non–contradictory. The well founded complete scenario, $WF(P)$, is the least complete scenario of P.*

An operator over scenaria exists such that every fixpoint of it is a complete scenario:

Definition 3.5 (V_P operator) *Given a program P and a set of NAF-hypotheses H we define $V_P(H) =_{def} H \cup Mand(H) \cup Acc(H)$ just in case $P \cup V_P(H)$ is a consistent scenario; otherwise $V_P(H)$ is not defined.*

Lemma 3.2 *$P \cup H$ is a complete scenario iff $H = V_P(H)$.*

Theorem 3.3 *If P is noncontradictory then V_P is monotonic and*

$$lfp(V_P) = WF(P).$$

Theorem 3.4 (Iterative construction of the WF complete scenario) *In order to obtain a constructive bottom–up iterative definition of the WF scenario of a non–contradictory program P, we define the following transfinite sequence $\{H_\alpha\}$ of sets of NAF-hypotheses of P:*

$$\begin{array}{rcl} H_0 & = & \{\} \\ H_{\alpha+1} & = & V_P(H_\alpha) \\ H_\delta & = & \bigcup\{H_\alpha \mid \alpha < \delta\} \quad \textit{for a limit ordinal } \delta \end{array}$$

By theorem 3.3, there exists a smallest λ such that H_λ is a fixpoint of V_P. The WF complete scenario is $P \cup H_\lambda$.

This constructive definition obliges one to know *a priori* whether a program is contradictory. This prerequisite is not needed:

Theorem 3.5 *A program P is contradictory iff in the sequence of the H_α there exists a λ such that $P \cup V_P(H_\lambda)$ is an inconsistent scenario.*

Thus, in order to compute the $WF(P)$ start building the above sequence. If, at some step i, H_i introduces a pair of complementary objective literals then end the iteration and P is contradictory. Otherwise iterate until the least fixpoint of V_P, which is the $WF(P)$.

4 Complete Scenaria and WFSX

In this section we establish the complete scenaria semantics CSS for extended logic programs and the semantics WFSX set forth in [15] are the same.

We first recap WFSX differing from the original presentation [15], but in a straightforwardly equivalent way given the results in [20]. WFSX can be construed as an appropriate generalization of Baral et al.'s [2] Γ^2–operator to programs with explicit negation, where Γ is the Gelfond–Lifschitz operator [8].

Definition 4.1 (Seminormal version of a program) *The seminormal version of a program P is the program P_s obtained from P by adding to the (possibly empty) Body of each rule $L \leftarrow Body$, a default literal not $\neg L$, where $\neg L$ is the complement wrt explicit negation of L. When P is understood from context, we use $\Gamma_s(S)$ to denote $\Gamma_{P_s}(S)$.*

Definition 4.2 (Extended stable models) *Let P be an extended program and S a set of objective literals such that: (1) $S = \Gamma\Gamma_s S$; and (2) $S \subseteq \Gamma_s S$.*

Then $M = S \cup \{not\, L \mid L \notin \Gamma_s S\}$ is called an extended stable model (XSM for short) of P, and S is called the generator of M. Members of $\Gamma_s S$ not in S are said undefined *in truth–value.*

Example 11 Some programs, like $\{a \leftarrow, \ \ \neg a \leftarrow\}$, have no XSMs.

Definition 4.3 (WFSX–contradictory program) *An extended logic program P is called WFSX–contradictory iff it has no extended stable models.*

Lemma 4.1 *For WFSX–noncontradictory programs the operator $\Gamma\Gamma_s$ is monotonic wrt set inclusion.*

Lemma 4.2 *Let S_1 and S_2 be two fixpoints of $\Gamma\Gamma_s$ for program P such that $S_1 \subseteq S_2$. If S_2 generates a XSM of P then S_1 also generates a XSM of P.*

By monotonicity of the $\Gamma\Gamma_s$ operator and this last lemma there follows:

Theorem 4.3 *If a program P has an XSM then P has a least XSM (wrt $\subseteq$). Moreover the generator of the least XSM is the least fixpoint of $\Gamma\Gamma_s$.*

Definition 4.4 (Well–founded model) *The well founded model (WFM) of a WFSX–noncontradictory extended program is the least XSM of P.*

Analogously to theorems 3.4 and 3.5, an iterative construction of WFM can be defined where $\Gamma\Gamma_s$ replaces V_P.

In this approach the iteration of the fixpoint operator $\Gamma\Gamma_s$ ends up with the set of objective literals which are true in the WFM, and false literals in the WFM are then obtainable from them.

This is the opposite of the approach taken in complete scenaria semantics. There the iteration of the fixpoint operator V_P ends up with the set of objective literals which are false in the WFM, and true literals in the WFM are then obtainable from them.

The theorem below states the equivalence between WFSX and CSS.

Theorem 4.4 (Equivalence) *If S is a XSM of a program P then $P \cup \{not\,L \mid L \notin \Gamma_s S\}$ is a complete scenario.*

If $P \cup H$ is a complete scenario, $\{L \mid P \cup H \vdash L\}$ generates a XSM.

5 More Credulous Semantics

Along the same lines of complete scenaria semantics, we can continue restricting the set of admissible scenaria, and defining in this way more credulous semantics. The most immediate semantics more credulous than CSS is the one obtained by considering only maximal (wrt $\subseteq$) complete scenaria. We call this semantics "preferred extensions" following the tradition of non–extended programs [3].

Definition 5.1 (Preferred extensions semantics) *The preferred extensions semantics of P is the set of its maximal complete scenaria.*

Example below shows that maximal elements might not exist for a collection of complete scenaria, hence preferred extensions are defined for less programs than CSS. Another straightforward result is that this semantics is in general more credulous than CSS.

Example 12 Consider program P :

$$\begin{array}{rclcrclcrcl} a & \leftarrow & not\, b & \quad & p(X) & \leftarrow & not\, q(X) & \quad & b & \leftarrow & not\, p(X) \\ \neg a & \leftarrow & not\, b & & q(X) & \leftarrow & not\, p(X) & & & & \end{array}$$

with Herbrand base $\mathcal{H} = \{0, 1, 2, 3, \ldots\}$.

Every scenario of the form $S_i = P \cup \{not\, q(k) \mid k \leq i\}$ is complete but there exists no complete scenario containing $\bigcup_i S_i$.

A reasoner can even be more credulous by considering only preferred extensions that are two valued (or total), i.e. extensions that whenever L is not a consequence of it $not\, L$ is assumed in it.

Definition 5.2 (Total scenaria semantics) *The total scenaria semantics of an extended program P is the set of its total complete scenaria.*

Theorem 5.1 (Answer–sets) *The total scenaria semantics coincides with the answer–sets semantics of [8].*

Clearly answer-sets semantics is defined for less programs than the previous semantics, since such total scenaria may in general not exist. The typical program for which answer-sets semantics is not defined but CSS is defined is $P = \{a \leftarrow not\, a\}$. This program has only one complete scenario, $\{not\, \neg a\}$, and it is not total. With explicit negation new problems regarding the existence of answer–sets appear. Example 12 shows that the computing of an answer–set cannot in general be made by finite approximations.

5.1 Comparison between the semantics presented

From the definition 2.2 of WFS0 and the iterative construction of the WF complete scenario of CSS (theorem 3.4) it follows almost directly that:

Theorem 5.2 (WFS0 is more sceptical than CSS) *For any non–contradictory program P, $WFS0(P) \subseteq CSS(P)$.*

Example 13 Consider program P :

$$p \leftarrow not\, q \qquad \neg p \leftarrow a \qquad \neg p \leftarrow b \qquad a \leftarrow not\, b \qquad b \leftarrow not\, a$$

whose CSS is $\{not\, q\}$ (apart from irrelevant literals such as $not\, \neg a$).

Since $P \cup \{not\, q, not\, \neg p\}$, $P \cup \{not\, a, not\, p\}$, and $P \cup \{not\, a, not\, p\}$ are admissible scenaria (though not all), and neither $not a$ nor $not b$ can be added to the first scenario, and $not\, q$ cannot be added neither to the second nor to the third scenario above, then $ISS = \{\}$. Thus $WFS0 = \{\}$.

Interesting questions are: *When do all these semantics coincide? Can we state sufficient conditions guaranteeing such an equivalence?*

In order to answer the second question we introduce the notion of semantically normal (s–normal for short) programs; i.e. those whose admissible scenaria can all be completed.

Definition 5.3 (S–normal program) *An extended program is s–normal iff for each admissible scenario $P \cup H$, $P \cup H \cup Acc(H)$ is consistent.*

Lemma 5.3 *Let P be a s–normal program, $P \cup H$ be an admissible scenario, and let $not\ A$, $not\ B$ be acceptable wrt $P \cup H$. Then $P \cup H \cup \{not\ A\}$ is admissible and $not\ B$ is acceptable wrt $P \cup H \cup \{not\ A\}$.*

From this lemma it follows immediately that the set of all admissible scenarios (wrt set inclusion) forms a complete partial order for s–normal programs. Hence, each admissible scenario can be extended into a complete scenario. Thus, for s–normal programs, ISS is contained in a complete scenario.

On the other side, it is easy to see that for each admissible scenario $P \cup H$, $P \cup H \cup CSS(P)$ is again admissible. Therefore:

Theorem 5.4 *Let P be a s–normal program. Then:*

- *The set of complete scenaria of P forms a complete semilattice.*
- *ISS coincides with the intersection of preferred extensions.*
- $WFS0(P) = CSS(P) \subseteq ISS(P)$.

To define larger classes of programs also guaranteeing these comparability results is an open problem. Also of special interest, and subject of future investigation by the authors as well, is to determine syntatic conditions over programs, guaranteeing the equivalence between answer-sets and CSS, in the spirit of the work in [4] regarding well founded semantics of non–extended programs and stable models.

However, for non–extended logic programs, since acceptable NAF-hypotheses can never lead to an inconsistency, both WFS0 and CSS coincide.

Theorem 5.5 (Relation to the WFS of normal programs) *If P is a normal (non–extended) program then CSS, WFS0 and the well–founded semantics of [31] coincide.*

Example 7 shows this equivalence cannot be extended to ISS. There CSS coincides with WFS0 and with WFS and is $\{\}$. ISS is $\{not\, b\}$.

Acknowledgements

The first and third authors thank ESPRIT COMPULOG 2 project, and JNICT Portugal, for their support. The second author is partially supported by the Abduction Group at Imperial College, London, under a grant from FUJITSU.

Notes

[1]As shown in [1], answer–sets comply with the coherence principle.

[2]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 the extended version of this paper. Intuitively, $\vdash$ is just the standard T_P operator of the Horn propositional programs obtained with the new symbols in place.

[3]The consistency of $P \cup E$ is not required; e.g. $P \cup \{not\,H\} \vdash H$ is allowed.

[4]This semantics is equivalent to one which only accepts NAF-hypotheses if it is explicitly negated in the program that there is evidence to the contrary. Hence it contains only the mandatory literals.

[5]Dung [4] has shown that stable model semantics can also be viewed as well–founded semantics, since it can be defined a similar way.

[6]One existing example of such restricted scepticism in logic programming is CRSX [19], which is more sceptical then the well–founded semantics afore. [16] presents a definition of CRSX based on the scenario framework.

[7]Another possibility, not explored here, is to refine the criteria by which acceptable hypotheses are not accepted by virtue of inconsistency, though distinguishing their specific contribution to the inconsistency, as in [18].

[8]However, for normal programs CS_P is a complete partial order.

References

[1] J. J. Alferes and L. M. Pereira. On logic programs semantics with two kinds of negation. In K. Apt, editor, *9th ICLP*. MIT Press, 1992.

[2] C. Baral and V. S. Subrahmanian. Dualities between alternative semantics for logic programming and nonmonotonic reasoning. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LPNMR*. MIT Press, 1991.

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

[4] P. M. Dung. On the relations between stable and well–founded models. *Theoretical Computer Science*, 105:7–25, 1992.

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

[6] P. M. Dung and P. Ruamviboonsuk. Well founded reasoning with classical negation. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LPNMR*, pages 120–132. MIT Press, 1991.

[7] 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.

[8] 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.

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

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

[11] R. Kowalski and F. Sadri. Logic programs with exceptions. In D. Warren and P. Szeredi, editors, *7th ICLP*. MIT Press, 1990.

[12] D. Pearce. Reasoning with negative information II: Hard negation, strong negation and logic programs. In D. Pearce and H. Wansing, editors, *Nonclassical Logics and Information Processing*, pages 63–79. Springer-Verlag, 1990.

[13] 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.

[14] D. Pearce and G. Wagner. Logic programming with strong negation. In P. Schroeder-Heister, editor, *Extensions of Logic Programming*, pages 311–326. Springer-Verlag, 1991.

[15] 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.

[16] L. M. Pereira and J. J. Alferes. Optative reasoning with scenario semantics. In *10th ICLP*. MIT Press, 1993. To appear.

[17] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Contradiction Removal within Well Founded Semantics. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LPNMR*, pages 105–119. MIT Press, 1991.

[18] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Adding closed world assumptions to well founded semantics. In *FGCS*, pages 562–569. ICOT, 1992.

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

[20] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Default theory for well founded semantics with explicit negation. In D. Pearce and G. Wagner, editors, *JELIA*, pages 339–356. Springer-Verlag, 1992.

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

[22] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Nonmonotonic reasoning with well founded semantics. In K. Furukawa, editor, *8th ICLP*, pages 475–489. MIT Press, 1991.

[23] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Logic programming for nonmonotonic reasoning. In *Applied Logic Conf.* ILLC, Amsterdam, 1992.

[24] 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 Logic Programming and NonMonotonic Reasoning*. MIT Press, 1993.

[25] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.

[26] 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.

[27] T. Przymusinski. A semantics for disjunctive logic programs. In Loveland, Lobo, and Rajasekar, editors, *ILPS'91 Ws. in Disjunctive L.P.*, 1991.

[28] C. Sakama. Extended well–founded semantics for paraconsistent logic programs. In *FGCS*, pages 592–599. ICOT, 1992.

[29] L. J. Stein. Skeptical inheritance: computing the intersection of credulous extensions. In *IJCAI*, pages 1153–1158. Morgan Kaufmann Publishers, 1989.

[30] D. S. Touretzky, J. F. Horty, and R. H. Thomason. A clash of intuitions: the current state of nonmonotonic multiple inheritance systems. In *IJCAI*. Morgan Kaufmann Publishers, 1987.

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

[32] 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.