# The Acceptability Semantics for Logic Programs

**A. C. Kakas**
Dept. of Computer Science
University of Cyprus
Kallipoleos 75
Nicosia, Cyprus
antonis@jupiter.cca.ucy.cy

**P. Mancarella**
Dipartimento di Informatica
Università di Pisa
Corso Italia, 40
56125 Pisa, Italy
paolo@di.unipi.it

**Phan Minh Dung**
Division of Computer Science
Asian Institute of Thechnology
GPO Box 2754
Bangkok 10501, Thailand
dung@cs.ait.ac.th

## Abstract

We present a simple yet powerful semantics for Negation as Failure (NAF) in logic programming, called the acceptability semantics. This is based on the idea that NAF literals represent possible extensions of a given logic program, provided that these satisfy an appropriate criterion, namely the acceptability criterion. The importance of this semantics and the way it is formulated lies in the fact that it allows us to abstract away NAF from the object-level syntax of our representation language. This has two significant consequences. First, it introduces a new more general, yet simpler, style of logic programming which is closer to the logical specification of non-monotonic problems, with the same basic computational paradigm of logic programming. Additionally, the understanding of the NAF principle through acceptability provides us with a general encapsulation of this non-monotonic reasoning principle that can be applied to other, richer in language, representation frameworks.

# 1 Introduction

This paper is concerned with the semantics of Negation as Failure (NAF) in Logic Programming [2] and the extension of this non-monotonic reasoning principle to more general representation frameworks. It proposes a way of understanding NAF that can be adopted more generally to provide a simple and natural representation framework close to the logical specification of problems requiring non-monotonic reasoning.

The basic motivation behind this work is the view that NAF is first and foremost a reasoning principle rather than a form of object-level negation. We are thus interested in providing a direct formalization of the intuitive principle "*not* $p$ holds iff $p$ fails to hold". The resulting semantics is called the acceptability semantics. We will show that this encompasses most of the existing semantics for NAF and thus it can help to unify and simplify the study of NAF in logic programming (LP, for short). More importantly, this formalization of the NAF principle provides a more general encapsulation of the principle that can be applied to other (richer in language) representation frameworks outside normal LP. In fact, many of the existing non-monotonic reasoning frameworks can be understood via this general NAF principle. In LP itself this allows us to remove NAF from the object-level syntax of the language, and thus the problem of the existence of two typesof negation (explicit negation and NAF) in the language disappears: NAF is elevated into the semantics.

The work in this paper follows a series of recent works on related ideas. It builds on the basic idea originating in [6] that NAF can be regarded as hypotheses that can be added to a given logic program, provided that they satisfy appropriate criteria. Whereas in [6] the criteria were expressed as integrity constraints, in later work [3] the basic criterion takes the form of acceptance relative to other possible conflicting extensions. The new approach of [3] was extended in [12]. The complexity aspects of this new approach have been studied in [22]. In particular, the paper [12] ends with the suggestion of the acceptability criterion as a simple and general semantics for NAF that can encompass other previously studied semantics. The current paper aims at presenting a proper formalization of the acceptability semantics for NAF in LP and then at applying it to define more general non-monotonic reasoning frameworks. The connection of these works to the notion of acceptability has been studied in [11], where an argumentation theoretic description of acceptability is proposed which we will be adopting in this paper. Also, in [5] a weaker notion of acceptability in LP is studied and shown that it encompasses various semantics for NAF such as the well-founded model and partial stable model semantics. The possibility of extending the acceptability semantics outside LP has been studied in [10] using the argumentation theoretic description of [11]. In particular it is shown that Default Logic [18] can be understood in this way. Similar results of understanding Default Logic and other existing non-monotonic frameworks

in terms of different criteria for accepting hypotheses which are closely related to the acceptability have been obtained in [1]. An abstract framework for argumentation is proposed in [4], in which it is possible to place these previous approaches, although the criteria proposed in [4] are different from the acceptability criterion used in this paper.

# 2 Acceptability Semantics in Logic Programming

In this section we will present and study the acceptability semantics for normal LP, as proposed in [12], using the argumentation theoretic view suggested in [11]. Our study starts from the object-level realization of NAF in LP, which will point out the need of elevating back the NAF principle at the semantics level, aiming at developing a framework where NAF is not explicit in the language.

## 2.1 Motivation

In the LP framework, NAF is basically a realization of the Closed World Assumption. Informally, it can be explained by the statement that, given a logic program $P$, "*not* $p$ holds (in $P$) iff $p$ fails to hold (in $P$)". In many examples of its use in LP we can indeed see that NAF is not an object level negation. Consider the rule

$fly \leftarrow bird, not\ abnormal.$

Here the negative condition *not abnormal* is used as a test that *abnormal* fails. It is a realization of the statement *unless abnormal* rather than a representation of an object-level negative condition needed for *fly* to hold. As a result, this negative condition would be absent in any framework that aims to be closer to the natural representation of this default rule.

On the other hand, there are examples where NAF is used in place of an object-level negation and therefore negative conditions should need to be present in the representation. Examples of this are the following programs for the even numbers and game playing, respectively:

$even(0) \leftarrow$
$even(s(x)) \leftarrow not\ even(x).$

$win(x) \leftarrow move(x,y), not\ win(y)$

Using NAF in this way, we cannot represent fully complex problems. For example, in the *even* program we do not capture that $s(d)$, for any $d$ which is not a natural number, is also not even. On the contrary we can conclude (incorrectly) $even(s(d))$. Also, we the above *game* rule we can not represent games in which there are draw positions. If $y$ is a draw position, and so $not\ win(y)$ holds, we will be able to derive (incorrectly) that any position $x$ from which a move to $y$ exists is a winning position. The problem here stems from the fact that NAF, which is to be understood as a form of CWA to form negative assumptions, is used in place of the explicit object-level negation $\neg win(y)$. Although it is possible to modify the program suitably

to handle these cases, our aim is to study how this can be done by remaining as close as possible to the natural representation of the problem.

This discussion indicates that, even in the case of normal LP, there is the need for two types of negations, which must be properly separated one at the object level and the other (NAF) at the metal-level. In section 4 we will see one way to achieve this.

## 2.2 Acceptability Semantics of Normal Logic Programs

A series of recent works [3, 5, 6, 12] have studied NAF as a form of default hypothesis that can be used to extend a given logic program. Thus, given a normal logic program $P$ we regard the set of negation as failure literals (naf literals, for short)

$\mathcal{B}^* = \{not\ p \mid p \in$ Herbrand Base of $P\}$

as a set of (default) hypotheses with which we can extend the program $P$. (We will assume that a logic program containing variables is a representation of all its variable-free instances over its Herbrand Universe.)

In an extension $P \cup H$, $H \subseteq \mathcal{B}^*$, we reason using definite Horn logic, where the naf literals $not\ p$ appearing in $P$ and $H$ are treated as positive atoms. The question of whether a subset $H$ of $\mathcal{B}^*$ can be accepted as an extension of $P$ depends on whether it obeys the NAF principle "$not\ p$ holds if and only if $p$ fails to hold".

The acceptability semantics, as suggested in [12], can be seen as a proposal for the formalisation of this principle. The first thing to notice is that the acceptability of the hypothesis $not\ p$ depends on the possibility of deriving (or not) the contrary information $p$. Hence $not\ p$ is in conflict with any other set $A$ of naf literals that, together with $P$, allows one to conclude $p$. We say that such a set *attacks* $not\ p$. More generally, the notion of attack between two sets of naf literals is the following.

**Definition 2.1** Let $P$ be a normal logic program and $A, H \subseteq \mathcal{B}^*$. Then $A$ attacks $H$ iff there exists $not\ p \in H$ such that $P \cup A \vdash p$. □

In the above definition, $\vdash$ stands for the usual provability relation of Horn clause logic (recall that here naf literals are viewed as positive atoms). Then, the acceptability of a set $H \subseteq \mathcal{B}^*$ depends on the ability or not to derive conflicting information from $\mathcal{B}^*$, i.e. it depends on the possible attacks against it. There are two cases for which it is easy to determine whether a hypothesis $not\ p$ is acceptable. When the program contains no rules for $p$, there are no attacks against $not\ p$. Hence, $\{not\ p\}$ is acceptable. Conversly, if $p$ is provable from the rules of the program without any further negative assumptions, i.e. $A = \{\}$ is an attack against $not\ p$, then $\{not\ p\}$ is not acceptable. In fact, with this interpretation of "$p$ holds" as "there is an acceptable attack against $\{not\ p\}$" and further with the interpretation of "$p$ fails to hold" not simply as "there is no attack to $\{not\ p\}$" but rather

as "any attack against $\{not\ p\}$ is not acceptable", we arrive at a recursive interpretation of the NAF principle as acceptability given by

"$H$ is acceptable iff any attack $A$ against $H$ is not acceptable."

This interpretation has been formalised and studied in [5] in terms of the fix points of an associated monotonic operator. This work has shown that the least fix point of this operator corresponds to the well-founded model [23] and the greatest fix points correspond to the preferred extensions [3] (and therefore to any other semantics equivalent to this, e.g. partial stable models [19].) This formalization can be generalised to cover extensions of these semantics while, at the same time, avoiding the need to consider greatest fix points. In fact, we can define the semantics fully in terms of the least fix point of a more general acceptability operator, that defines an acceptability relation satisfying the following specification.

**Definition 2.2** (*Specification of Acceptability for LP*)
Let $P$ be a normal logic program and $H_0, H \subseteq \mathcal{B}^*$. Then:

$Acc(H, H_0)$ iff for any attack $A$ against $(H \setminus H_0)$ : $\neg Acc(A, H \cup H_0)$. □

It is important to notice that acceptability is a binary relation on $\mathcal{B}^*$, i.e. that the central notion is "$H$ is acceptable w.r.t. $H_0$", where $H_0$ is regarded as a given choice or context of hypotheses. The basic idea is to consider the acceptability of a set $H$ as a property relative to itself, i.e. in the context where $H$ is to be assumed. This context is defined non-deterministically when we choose a set $H$ among the whole set $\mathcal{B}^*$of hypotheses. We can then use the hypotheses in $H$ to justify themselves, i.e. to make themselves acceptable by rendering attacking sets of hypotheses not acceptable.

Notice also that the attacks $A$ against $H$ that must be considered are only those that attack the new part of $H$, namely $H \setminus H_0$. This subtraction of $H_0$ is important as it is needed to ensure that attacks are not against hypotheses in $H_0$, which we are in fact trying to adopt. In other words, the notion of attack must be *relative to* a given set of hypotheses (namely "$A$ attacks $H$ rel. to $H_0$") that limits the attacks only to those that are against the new hypotheses in $H$ (a more detailed discussion regarding this issue can be found in [12].) Before defining formally the acceptability relation, we illustrate it with an example.

**Example 2.3**

$P$: $\quad p \leftarrow not\ q$
$\quad\quad q \leftarrow not\ p$

$H_1 = \{not\ p\}$ is an acceptable extension of $P$, since any attack against $H_1$ must contain $\{not\ q\}$. But any set containing $not\ q$ is counter-attacked by $H_1$, which is trivially acceptable to itself. This shows how a set of hypotheses is used as its own defence to render itself acceptable. Similarly, $H_2 = \{not\ q\}$ is also an acceptable extension of $P$. □

## 2.3 A fix point definition of Acceptability

The acceptability relation is defined formally as the least fix point of a suitable operator $\mathcal{F}$, obtained by unfolding its recursive specification above.

**Definition 2.4**
Let $P$ be a logic program and $R$ be the set of binary relations on $2^{\mathcal{B}^*}$. Then $\mathcal{F} : R \rightarrow R$ is defined as follows, for any $Acc \in R$ and $H, H_0 \in 2^{\mathcal{B}^*}$:

$\mathcal{F}(Acc)(H, H_0)$ iff for any attack $A$ against $(H \setminus H_0)$
there exists an attack $D$ against $(A \setminus (H \cup H_0))$
s.t. $Acc(D, A \cup H \cup H_0)$.

□

It is easy to show that this operator is monotonic with respect to $\subseteq$.

**Definition 2.5** (*Acceptability relation and acceptable extensions*)
Let $P$ be a logic program and $\mathcal{F}$ the operator as in Def. 2.4. Then:

(i) the acceptability relation $Acc$ of $P$ is the least fix point of $\mathcal{F}$

(ii) given $H \subseteq \mathcal{B}^*$, $P \cup H$ is an *acceptable extension* of $P$ iff $Acc(H, \{\})$. □

**Proposition 2.6** [1] *Let $P$ be a normal logic program and $Acc$ its acceptability relation. Then:*

*(i) there exists at least one acceptable extension of $P$*

*(ii) if $H \subseteq H_0$ then $Acc(H, H_0)$*

*(iii) if $Acc(H, \{\})$ then $P \cup H$ is consistent, i.e. for no $p$ both not $p \in H$ and $P \cup H \vdash p$.* □

Proposition 2.6(ii) shows how Def. 2.5 effects the desired property of taking a set of hypotheses $H_0$ as an a-priori given set under which we want to investigate the acceptability of some other set of hypotheses. Proposition 2.6(iii) shows that inconsistency is understood as self-attack. Let us illustrate the above definitions with an example that emphasizes the recursive nature of acceptability.

**Example 2.7**

$P$: $p \leftarrow not\ q$ $\quad$ $r \leftarrow not\ p$
$q \leftarrow not\ r$ $\quad$ $s \leftarrow r$

[1] All the proofs are omitted and are reported in [13].

We note here that a program like the previous one can be generated by the stable-marriage problem presented and discussed in [4] (see also [13]). The set $H = \{not\ p\}$ is not acceptable since, in the context of $H$, its attack $\{not\ q\}$ becomes acceptable. In fact, any attack against $\{not\ q\}$ must contain $\{not\ r\}$ and therefore can itself be attacked by $H$ (since it derives $r$) which is acceptable to itself. Now the set $\{not\ s\}$ is acceptable, since its attacks must contain $H = \{not\ p\}$ and hence are not acceptable. □

**Theorem 2.8** *Let $P$ be a logic program. Then any stable model [8], partial stable model [19], stationary expansion [17], preferred extension [3] and stable theory [12] corresponds to an acceptable extension of $P$.* □

Furthermore, there are programs for which the acceptability semantics gives additional extensions which are not captured by these other semantics, such as example 2.7. Theorem 2.8 deals with the credulous semantics for logic programs. Turning to the skeptical semantics, it can be shown (see [12]) that the well-founded model [23] of any logic program corresponds to an acceptable extension. More importantly, we can define a strong form of acceptability that captures and extends the well-founded semantics.

**Definition 2.9** Let $P$ be a logic program and $H \subseteq \mathcal{B}^*$. Then $H$ is defined to be *strongly acceptable* to $P$, (denoted by $Acc_{ST}(H)$) if:

$Acc_{ST}(H)$ iff for any attack $A$ of $H$ : $\neg Acc(A, \{\})$,

where $Acc$ is the (credulous) acceptability relation of Def. 2.5. □

Essentially, a set $H$ is strongly acceptable if there is no possible non-deterministic choice of hypotheses that is acceptable and attacks $H$.

**Definition 2.10** The skeptical acceptable semantics of $P$ is given by its maximal strongly acceptable extension. □

Example 2.7 shows that the skeptical acceptability semantics can be seen as an extension of the well-founded semantics. Its skeptical acceptability semantics is given by $H = \{not\ s\}$ whereas the well-founded semantics does not assign a value to $s$ or $not\ s$. Note that to capture the well-founded semantics exactly (see [5]) we need to restrict our notion of strongly acceptable as follows:

$Acc_{WF}(H)$ iff for any attack $A$ of $H$, $\neg Acc_{WF}(A)$.

We note here that in many cases it is sufficient to work with approximations of the acceptability semantics given by the various iterations (or approximations of these iterations) of the fix point operator defined in Def. 2.4. An important example of such an approximation is given by admissibility [3]. This will be used in Section 4 to show and motivate the connection between ordinary LP with NAF in the object-level syntax and the new framework of LP that we are proposing, where NAF is absent from the language.

# 3 General Theory of Acceptability

The acceptability semantics for normal LP can be applied to more general representation frameworks. To do this we need to abstract some of the central notions of the semantics, as applied to LP, and then apply the same ideas referring to a different underlying representation framework. The central notions in the LP case are the notions of attack and acceptability. In our generalization, we keep the notion of acceptability fixed and simply use the attacking relation as a parameter on which acceptability depends.

**Definition 3.1** A *non-monotonic reasoning framework* is given by a monotonic background logic $\mathcal{L}$ along with a binary attacking relation $attack(T, T')$ between sets of sentences (theories) in $\mathcal{L}$. □

Given a theory $\mathcal{T}$ in a non-monotonic reasoning framework, we have the following defining axioms for acceptability.

**Definition 3.2** *(Axioms of acceptability)*
Let $\mathcal{T}$ be a theory in a non-monotonic reasoning framework. Then the acceptability relation $Acc$ on $\mathcal{T}$ is specified via the following axioms. For any $T, T_0 \subseteq \mathcal{T}$ :

(a1) $Acc(T, T_0)$ if $T \subseteq T_0$

(a2) $Acc(T, T_0)$ if for any attack $T'$ against $T$ rel. to $T_0$, $\neg Acc(T', T \cup T_0)$.

□

Note that (a2) requires the notion of attack *relative to* a given subtheory $T_0$ of sentences. This is a generalization of the subtraction $(H \setminus H_0)$ that we have in the Def. 2.4 for the LP case, where the sentences in $H$ and $H_0$ are simple assertions of NAF literals. This notion captures the fact that $T_0$ should be considered as given, and consequently any attack $T'$ against $T$ should not at the same time be an attack against $T_0$. In other words $T'$ should not attack $T_0$ in exactly the same way it attacks $T$. To see the importance of this notion consider $Acc(T, \{\})$. For $T$ to be acceptable to $\{\}$, it is necessary, for any attack $T'$ against $T$ (rel. to $\{\}$), that $T$ counter-attacks $T'$ (rel. to $T$). Now, if $T'$ contains statements that belong to $T$, the required counter-attack against $T'$ must be a genuine attack against $T'$, that is it can not be isolated as an attack against the part of $T'$ that also belongs to $T$.
We will thus require that this notion of relative attack has the following properties:

(1) if $T'$ attacks $T$ and $T'$ does not attack $T_0$ then $T'$ attacks $T$ rel. to $T_0$

(2) if $T \subseteq T_0$ then there exists no $T'$ s.t. $T'$ attacks $T$ rel. to $T_0$.

The existence of the acceptability relation and semantics in a general non-monotonic reasoning framework follows in exactly the same way as for the special case of normal LP in Proposition 2.6. The associated fix point operator is defined in the same way as in Def. 2.4, where we replace the specific form of relative attack with its general form.

**Definition 3.3** Let $\mathcal{T}$ be a theory in a non-monotonic reasoning framework and $R$ be the set of binary relations on $2^{\mathcal{T}}$. Then $\mathcal{F} : R \rightarrow R$ is defined as follows, for any $Acc \in R$ and $T, T_0 \in 2^{\mathcal{T}}$:

$\mathcal{F}(Acc)(T, T_0)$ iff for any attack $T'$ against $T$ rel. to $T_0$,
there exists an attack $T''$ against $T'$ rel. to $(T \cup T_0)$
s.t. $Acc(T'', T' \cup T \cup T_0)$.

□

**Definition 3.4** The (credulous) semantics for a theory $\mathcal{T}$ is the set of acceptable extensions of $\mathcal{T}$, i.e. the set of $T \subseteq \mathcal{T}$ s.t. $Acc(T, \{\})$, $Acc$ being the least fix point of the operator $\mathcal{F}$ of Def. 3.3. □

The skeptical semantics of a non-monotonic reasoning framework can be defined as in the LP case through a strong acceptability relation, as in Def. 2.9 and 2.10. As for LP, the definition of acceptability captures the following informal abstraction of the NAF principle. Given a theory $\mathcal{T}$ that may typically contain incompatible information, the acceptability relation gives us suitable subsets $T$ of $\mathcal{T}$ such that $Acc(T, \{\})$, which we can reason with. Such a set $T$ has the property that it can defend itself from any subset of $\mathcal{T}$ that would render it incompatible, i.e. it can defend itself from any attack. The acceptability semantics for a general non-monotonic reasoning framework is based on an attacking relation among theories. It is important to relate this attacking relation to the background logic of the framework or, in other words, to use an attacking relation that is *naturally* derived from the background logic itself. For the case of LP, as analysed in the previous section, the background logic is definite Horn logic and the notion of attack is the one of Def. 2.1, that is it is defined through the conflict between *not* $p$ and $p$. Like in LP, we often have some natural notion of conflicting information and then, roughly speaking, we can say that a theory $T'$ attacks another theory $T$ iff they derive conflicting information. The typical example of such conflicting information is of course the case of a sentence and its explicit (or classical) negation.

**Definition 3.5** *(Complements/Consistency)*
Let $\mathcal{L}$ be a background logic, $\phi$ a wff formula and $\phi^c$ a complement of $\phi$. We say that $\phi$ and $\phi^c$ are in conflict. A theory $T$ is inconsistent (or incompatible) iff $T \vdash \phi, \phi^c$ for some formula $\phi$. Otherwise, we say that $T$ is consistent (or conflict-free). □

We thus assume that a non-monotonic framework comes with some notion of complements and consistency of its theories. We then require that the attacking relation obeys the following properties.

**Property 3.6** (Properties of Attacks)
*Let $T'$ and $T$ be two theories in $\mathcal{L}$:*

*(i) if $T'$ attacks $T$ then there exists a wff $\phi$ such that $T \vdash \phi$ and $T' \vdash \phi^c$;*

*(ii) if $T'$ attacks $T$ then $T'$ attacks any superset of $T$;*

*(iii) if $T$ is inconsistent, then $T$ attacks $T$.* □

Property 3.6(iii) ensures that any acceptable extension of a given theory $T$ is consistent.

**Theorem 3.7** *Let $\mathcal{T}$ be a theory and $T \subseteq \mathcal{T}$ such that $Acc(T, \{\})$. Then $T$ is consistent.* □

Another important observation is that, if the attacking relation is symmetric, then the acceptability semantics always reduces to consistency as the following result shows. This can occur for example if the complement relation is symmetric, i.e. $(\phi^c)^c = \phi$.

**Proposition 3.8** *Let $\mathcal{T}$ be a theory in a non-monotonic reasoning framework whose attacking relation is symmetric. Then $T \subseteq \mathcal{T}$ is acceptable to $\{\}$ iff $T$ is consistent.* □

The most interesting cases of the acceptability semantics occurs when the attacking relation is not symmetric. Informally, in order to allow $T'$ to attack $T$ we need to localise the incompatibility of $T$ and $T'$ within $T$. For example, in the case of LP, where the given theory $\mathcal{T} = P \cup \mathcal{B}^*$, the attacking relation is not symmetric. We can understand this breaking of the symmetry in different ways. One simple way is to regard the complement relation as non-symmetric, namely the complement of *not p* is $p$ but the reverse does not hold. This understanding is not completely satisfactory in view of the fact that it can not be applied to other frameworks where the conflicts are given through explicit or classical negation (a symmetric form of complement.) Another way of understanding this asymmetric attacking relation of LP is based on the separation of any given theory $\mathcal{T}$ into the program $P$ and the set $\mathcal{B}^*$. This is defined as follows.

**Definition 3.9** *(Attack for logic programming)*
Let $P$ be a normal logic program and $<$ the priority relation on $P \cup \mathcal{B}^*$ that assigns any sentence in $\mathcal{B}^*$ lower priority than any sentence in $P$. Then, for any $H, A \subseteq P \cup \mathcal{B}^*$, $A$ attacks $H$ iff there exist a literal $L$ and $A' \subseteq A$, $H' \subseteq H$ such that:

(i) $H' \vdash_{min} L$ and $A' \vdash_{min} L^c$, and

(ii) there is no rule $\phi$ in $A'$ such that $\phi < \psi$ for some rule $\psi$ in $H'$

where $T \vdash_{min} L$ denotes the fact that $T \vdash L$ and no proper subset $T'$ of $T$ is such that $T' \vdash_{min} L$. □

Notice that $L$ stands for a positive or negative literal ($p$ or *not* $p$ resp.) and $L^c$ denotes its symmetric complement (*not* $p$ or $p$ resp.). It is easy to see that this attacking relation is equivalent to the one in Def 2.1.

This relatively simple alternative view of the attacking relation in LP is very important as it will motivate the definition of a general non-monotonic reasoning framework that can extend LP and, at the same time, allows us to remove NAF from the object-level language.

Before presenting these issues, we mention here that that the general theory of acceptability developed in this section can be applied in a straightforward way (see [13]) to give a semantics to extensions of LP such as disjunctive LP and extended LP with classical negation.

# 4 Logic Programming without negation as failure

In this section we present a concrete framework for non-monotonic reasoning, based on the general ideas and theory developed in section 3, which encompasses and extends the frameworks of normal LP and its various extensions. We saw in section 3 that, in order to specify a non-monotonic reasoning framework, we need to define its background logic, complements and attacking relation.

**Definition 4.1** *(Background logic)*
Formulae in the language of the framework are defined as $L \leftarrow L_1, \ldots, L_n$, where $L, L_1, \ldots, L_n$ are positive or explicit negative literals. The only inference rule is the modus ponens rule

$$\frac{L \leftarrow L_1, \ldots, L_n \qquad\qquad L_1, \ldots, L_n}{L} \qquad (n \geq 0) \qquad \square$$

We assume that, together with the set of sentences $\mathcal{T}$, we are given a priority relation $<$ on these sentences (where $\phi < \psi$ means that $\phi$ has lower priority than $\psi$). The role of the priority relation is to encode locally the relative strength of rules in the theory, typically between contradictory rules. We will require that $<$ is irreflexive and antysimmetric.

**Definition 4.2** *(Non-Monotonic Theory or Program)*
A theory $(\mathcal{T}, <)$ is a set of sentences $\mathcal{T}$ in $\mathcal{L}$ together with a priority relation $<$ on the sentences of $\mathcal{T}$. □

Let us now proceed to define an appropriate notion of attack on these theories. The only source of conflict that we have in a theory $\mathcal{T}$ is between a literal $L$ and its explicit negation $\neg L$, which is a symmetric form of complement. The presence of the priority relation $<$ on $\mathcal{T}$ allows us to define a notion of attack which is in general non-symmetric.

**Definition 4.3** *(Attacks)*
Let $(\mathcal{T}, <)$ be a theory and $T, T' \subseteq \mathcal{T}$. Then $T'$ attacks $T$ iff there exists $L$, $T_1 \subseteq T'$ and $T_2 \subseteq T$ such that

(i) $T_1 \vdash_{min} L$ and $T_2 \vdash_{min} \neg L$

(ii) $(\exists r' \in T_1, r \in T_2 \text{ s.t. } r' < r) \Rightarrow (\exists r' \in T_1, r \in T_2 \text{ s.t. } r < r')$. □

$T \vdash_{min} L$ means that $T \vdash L$ and that $L$ can not be derived from any proper subset of $T$. This definition is a generalization of the corresponding notion of attack for LP as given in Def. 3.9. Notice also that the property "if $T$ is inconsistent then $T$ attacks $T$" is trivially satisfied by this attacking relation.

In this way, we have completed the definition of our non-monotonic reasoning framework according to Def. 3.1. Its semantics is given by acceptability within the argumentation theoretic framework of section 3. As mentioned in section 3, we can work with any suitable approximation of the acceptability semantics. In this spirit, we will now consider the approximation given by *admissibility* and show how logic programs with object-level NAF can be equivalently understood as a specific type of theories in the above non-monotonic reasoning framework.

**Definition 4.4** *(Admissibility)*
Let $(\mathcal{T}, <)$ be a theory and $T \subseteq \mathcal{T}$. Then $T$ is *admissible* iff for any $T' \subseteq T$ if $T'$ attacks $T$ then $T$ attacks $T'$ rel. to $T$. □

This can be expressed equivalently as "$T$ is admissible iff $T$ is consistent and for any $T' \subseteq \mathcal{T}$ if $T'$ attacks $T$ then $T$ attacks $T'$" which is a form closer to the original definition in [3].

Given a logic program, $P$, we define a corresponding theory $\mathcal{D}(P)$. This transformation is motivated (see section 2) from the interpretation of *not* $p$ as *unless* $p$. For example, if we have a rule "$p \leftarrow q, not\ r$" then this is understood as "$p$ holds if $q$ holds unless $r$ holds, in which case this way of deriving $p$ can not apply". The rule is then transformed into two sentences "$p \leftarrow q$"and "$\neg p \leftarrow r$", and the second is assigned higher priority than the first.

**Definition 4.5** Let $P$ be a logic program and $r$ be a rule in $P$ of the form $A \leftarrow L_1, \ldots, L_n$. Then $\mathcal{D}(r)$, the default theory associated to $r$, is $(T_r, <_r)$ defined as follows. $T_r$ contains only the sentences generated by 1,2 and 3 below:

(1) $A \leftarrow A_r$ belongs to $T_r$;

(2) $A_r \leftarrow A_1, \ldots, A_m$ belongs to $T_r$, where $A_1, \ldots, A_m$ is the conjunction of all the positive atoms in $L_1, \ldots, L_n$;

(3) for each *not* $B$ in the conjunction $L_1, \ldots, L_n$, the rule $\neg A_r \leftarrow B$ belongs to $T_r$

where $A_r$ is a new propositional letter. The priority relation, $<_r$, contains only the pairs $r' <_r r''$, where $r'$ is the rule introduced in (2) and $r''$ is any rule introduced in (3). □

**Definition 4.6** Let $P$ be a normal logic program. Then the corresponding theory, $\mathcal{D}(P)$, is the non-monotonic theory $(\mathcal{T}_P, <)$ defined as follows:

(1) $\mathcal{T}_P$ is the union of $T_r$, for every rule $r$ in the program $P$

(2) for any $r', r''$ in $\mathcal{T}_P$, $r' < r''$ iff $r' <_r r''$ for some rule $r$ in $P$. □

We assume that the new propositional symbols introduced in $\mathcal{D}(P)$ associated with two different rules are distinct, even when these two rules have the same head. In fact, this is the reason for introducing these new symbols so that we can separate different rules in $P$ for the same atom.

**Example 4.7** The theory $\mathcal{D}(P)$ associated with the program $P$ of example 2.3 is the following:

$\mathcal{D}(P)$: $p \leftarrow$ $\qquad$ $q \leftarrow$

$\neg p \leftarrow q$ $\qquad$ $\neg q \leftarrow p$

$<$: $\{(p \leftarrow) < (\neg p \leftarrow q),\ (q \leftarrow) < (\neg q \leftarrow p)\}$

where we have not introduced new symbols here since there is only one rule for each propositional symbol of the original program. □

We note that a related transformation has been proposed in [14] to transform extended logic programs with explicit negation and NAF to normal logic programs containing only NAF. With this transformation of logic programs into non-monotonic theories we can show that admissibility in the two frameworks coincides.

**Theorem 4.8** *Let $P$ be a normal logic program, $\mathcal{D}(P) = (\mathcal{T}_P, >)$ its corresponding theory, and $a$ any positive atom in the original language of $P$. Then for each admissible extension $P \cup H$ there exists an admissible subtheory $T$ of $\mathcal{T}_P$ such that*

*(i) $P \cup H \vdash a$ iff $T \vdash a$ and (ii) if not $a \in H$ then $T \nvdash a$.*

*Conversely, for each admissible subtheory $T$ of $\mathcal{T}_P$ there exists an admissible extension $P \cup H$ such that:*

*(i) $T \vdash a$ iff $P \cup H \vdash a$ and (ii) if $T \nvdash a$ then not $a \in H$.* □

In the above example 4.7 the admissible set of hypotheses $\{not\ p\}$ corresponds to the admissible subtheory $\{(q \leftarrow), (\neg q \leftarrow p)\}$ of $\mathcal{T}_P$.
Hence LP with NAF in the object language can be simulated exactly in terms of a subclass of non-monotonic theories that contain only explicit negation. We also note that we can apply a Closed World Assumption within an admissible (or more generally acceptable) subtheory $T$ of $\mathcal{D}(P)$ by assuming the negation of any atom $a$ that does not follow from $T$. The resulting theory will always be consistent.

This subclass of theories corresponding to ordinary logic programs is very specific, since it requires that all conditions of the rules in a theory are positive atoms and the priority relation on the theory is of the form given in Def. 4.6. In the new style of LP that we are advocating, both restrictions can be lifted thus providing a more general representation framework.

Let us illustrate this new framework with a few examples. The usual non-monotonic problem of "flying birds" can be represented by the theory

$r_1 : fly \leftarrow bird$ $\qquad r_1 < r_2$
$r_2 : \neg fly \leftarrow penguin$ $\qquad r_2 < r_3$
$r_3 : fly \leftarrow superpenguin$
$r_4 : bird \leftarrow penguin$
$r_5 : penguin \leftarrow superpenguin$

This theory has acceptable extensions that can derive separately *fly* or $\neg$*fly*. If we add to the theory "$r_6 : superpenguin \leftarrow$" then there is no acceptable extension that derives $\neg$*fly*.
The theory for representing the even numbers will now be written as

$r_1 : even(0) \leftarrow$
$r_2 : even(s(x)) \leftarrow \neg even(x)$
$r_3 : \neg even(s(x)) \leftarrow even(x)$

with an empty priority relation. This does not suffer from the same problems encountered in the program with NAF as discussed in section 2. The previous examples can be handled satisfactorily within the admissibility approximation of the acceptability semantics. There are however problems where this approximation is not sufficient. One such example is the game problem with draw positions, as discussed in section 2. The game program is now represented as

$r_1 : win(x) \leftarrow move(x, y), \neg win(y)$
$r_2 : \neg win(x) \leftarrow move(x, y), win(y)$

with extra rules for *move* and an empty priority relation. Again this does not suffer from the problems discussed in section 2. Other examples can be found in [13].

# 5 Conclusions

We have proposed a framework for non-monotonic reasoning based on ageneral encapsulation, within an argumentation theoretic set up, of the NAF

principle in LP. The semantics of these frameworks is given via the acceptability relation. One of the results of adopting this approach is a new non-monotonic framework based on LP where NAF is elevated to the semantics with only explicit negation needed in the object-level language. The acceptability relation and the corresponding semantics have a naturally associated proof theory that stems directly from their definition. This has been developed in [21] and applied to the special case of LP with object-level NAF as defined in section 2. This proof procedure shares all the basic characteristics of the computational model of ordinary LP with NAF.

We believe that a suitable domain of application of the new LP framework is the area of legislation [20]. This stems from the fact that the specification of problems in this domain, where some of the rules (laws) act as exceptions to other rules (laws), has naturally the form of the non-monotonic theories in the proposed framework.

The framework we have proposed relies heavily on the existence of a priority relation on its theories. The use of priorities and preference relations for non-monotonic reasoning has been the subject of study of many other previous works, e.g [7, 9, 15, 16] and further work is needed to study the relation between our work and these earlier works.

### Acknowledgements

The authors have benefited from discussions with R. A. Kowalski and F. Toni, and from comments and suggestions of the anonymous referees. This research was partly supported by the ESPRIT BRA 6810 (Compulog 2), and PFI Sistemi Informatici e Calcolo Parallelo under grant no. 92.01564.PF69. The second and third author acknwoledge support from the EEC activity KIT011 - LPKRR. The third author was partly supported by the abduction group at Imperial College, under a grant from Fujitsu.

## References

[1] Bondarenko A., Toni F. and R.A. Kowalski. An assumption-based Framework for Non-monotonic Reasoning. *Proc. 2nd Int. Workshop on Logic Programming and Non-Monotonic Reasoning*, Lisbon, 1993.

[2] Clark K.L. Negation as Failure. *Logic and Databases* (H. Gallaire and J. MInker, eds.), Plenum, New York, 1978.

[3] Dung P. M. Negation as Hypothesis: An Abductive Foundation for Logic Programming. *Proc. 8th ICLP*, MIT Press, Paris, 1991.

[4] Dung P. M. On the Acceptability of Arguments and its fundamental role in Non-Monotonic Reasoning and logic programming. *Proc. 13th IJCAI*, 1993.

[5] Dung P. M., Kakas A.C. and P. Mancarella. Negation as Failure Revisited. Technical Report, University of Pisa, 1992.

[6] Eshghi K. and R. A. Kowalski. Abduction Compared with Negation by Failure. *Proc. 6th ICLP*, Lisbon, (1989) 234-255.

[7] Geffner H. *Default Reasoning: Causal and Conditional Theories.* MIT Press, 1992.

[8] Gelfond M. and V. Lifschitz. The Stable Model Semantics for Logic Programming. *Proc. 5th ICLP*, MIT Press, Seattle, 1988.

[9] Hunter A. Using priorities in non-monotonic proof theory. Technical report, Imperial College, London, 1993.

[10] Kakas A.C. Default Reasoning via Negation as Failure *Proc. ECAI'92 Workshop on The theoretical foundations of Knowledge Representation and Reasoning*, (G. Lakemeyer and B. Nebel, eds), 1993.

[11] Kakas A.C., Kowalski R.A. and F. Toni. Abductive Logic Programming. *Journal of Logic and Computation*, 2(6):719-770, 1993.

[12] Kakas A.C. P. Mancarella. Stable theories for Logic Programs. *Proc. Int. Symp. of Logic Programming*, MIT Press, San Diego, 1991.

[13] Kakas A.C., Mancarella P. and P.M. Dung. The Acceptability Semantics for Logic Programming. Technical Report, University of Cyprus, 1994.

[14] Kowalski R.A. and F. Sadri. Logic Programs with Exceptions. *Proc. 7th ICLP*, MIT Press, Jerusalem, 1990.

[15] Laenens E., Saccà D. and D. Vermeir. Extending Logic Programming. *Proc. ACM SIGMOD Conference*, Atlantic City, 1990.

[16] Pereira L.M., Aparicio J.N. and J.J. Alferes. Non-Monotonic Resoning with well-founded semantics. *Proc. 8th ICLP*, Paris, 1991.

[17] Przymusinski T.C. Stationary Semantics for Disjunctive Logic Programs and Deductive Databases *Proc. NACLP 90* (S. Deabray and M. Hermenegildo, eds.), MIT Press, 40-60, 1990.

[18] Reiter R. A Logic for Default Reasoning. *Journal of AI*, 13:81-132, 1980.

[19] Saccà D. and C. Zaniolo. Stable Models and Non-Determinism for Logic Programs with Negation. *Proc. ACM SIGMOD-SIGACT Symp. on Principles of Database Systems*, 205-217, 1990.

[20] Sergot M.J. et al. The British Nationality Act as a Logic Program, *Communication of the ACM*, 29, 1986.

[21] Toni F. and A. C. Kakas. Computing the Acceptability Semantics. Technical report, Imperial College, 1993.

[22] A. Torres. Negation as Failure to Support. *Proc. 2nd Int. Workshop on Logic Programming and Non-Monotonic Reasoning*, Lisbon, 1993.

[23] Van Gelder A., Ross K.A. and J. S. Schlipf. Unfounded sets and the well-founded Semantics for General Logic Programs. *Proc. ACM SIGMOD-SIGACT Symp. on Principles of Database Systems*, 221-230, 1988.