Explaining SLDNF resolution with non-normal defaults

by M. A. Casanova
A. S. Hemerly
R. A. de T. Guerreiro

This paper defines a default logic interpretation for normal programs that has the following major characteristics. First, it directly captures the true nature of SLDNF resolution as an extension of SLD resolution. Second, it is semantically convincing, but it requires neither an elaborated nonstandard interpretation nor a radical rewriting of the program clauses that would make it difficult to understand their meaning. Last, it extends known results for stratified normal programs to programs that satisfy a weaker condition.

1. Introduction

The basis for the vast majority of logic programming systems is a refutation method, called SLD (Structured Logic Design) resolution [1, 2], which accepts only definite programs whose clauses do not admit negative literals in their body. When one relaxes this restriction and switches to so-called normal programs, the refutation method usually adopted becomes SLDNF (SLD with Negation by Failure) resolution, which extends SLD resolution with the negation-by-finite-failure (NFF) rule [3]. Roughly, the NFF rule states that a negative literal $\neg L$ should be canceled from the body of a clause if the query $\leftarrow L$ fails finitely in the presence of the program clauses.

We note at least three important and distinct characteristics of the NFF rule: NFF is easy to implement in Prolog systems; NFF is a nonmonotonic rule justified on the grounds of the so-called *"Closed-World Assumption"* (CWA) [4]; and it is very difficult to define a semantics for normal programs for which SLDNF resolution is sound and complete. The first characteristic is undoubtedly the greatest argument in favor of the adoption of the NFF rule. The second characteristic can be taken either in favor of or against the use of NFF, depending on whether the CWA holds for the application in question.

The third characteristic can best be examined with the help of a very simple example. Consider the question of expressing a disjunction $p \vee q$ as a program clause. If we naively take the symbol $\leftarrow$ to mean (reverse) implication and $\neg$ to mean true negation, then the formula $p \vee q$ and the clauses $p \leftarrow \neg q$ and $q \leftarrow \neg p$ are all equivalent. But this equivalence is false, because negated literals are treated by the NFF rule. Indeed, let $P_1 = \{p \leftarrow \neg q\}$ and $P_2 = \{q \leftarrow \neg p\}$ be two normal programs. Let Q be the query $\leftarrow p$. Then, in an extended Prolog system, the answer of Q to P_1 will be TRUE, whereas the answer of Q to P_2 will be FALSE.

This apparently incorrect behavior will certainly shock naive Prolog users, but it can be explained by resorting, for example, to Clark's theory of program completion [3]. Indeed, denote by comp(S) the completion of a program

S. Then, $comp(P_1) = \{p \Leftrightarrow \neg q, \neg q\}$ and $comp(P_2) = \{q \Leftrightarrow \neg p, \neg p\}$. Hence, p is indeed a logical consequence of $comp(P_1)$ but not of $comp(P_2)$, which correctly *explains* the previous answers, and also *justifies* the bizarre behavior of extended Prolog systems.

From this simple discussion, it becomes clear that, although NFF indeed extends in some sense the expressiveness of pure Prolog systems, it greatly complicates the theory, to the point of raising serious questions with respect to basic soundness.

The goal of this paper is to use defaults to provide a simple and intuitive explanation for SLDNF. The natural choice would be to use CWA defaults, that is, defaults of the form $(:\neg L/\neg L)$, where $\neg L$ is a ground-negative literal. However, CWA defaults create certain problems, as discussed in Section 3, and are then replaced by another class of defaults, called barred CWA defaults, that offer the following advantages. First, they directly capture the true nature of SLDNF as an extension of SLD resolution. Second, they are semantically convincing, as proved at the end of Section 3, but they require neither an elaborated nonstandard semantics nor a radical rewriting of the program clauses that would make it difficult to understand their meaning. Finally, the semantics induced by barred CWA defaults coincides with known results for stratified programs, but it "captures more," since the basic result described in Section 3 requires a condition weaker than stratification.

Alternative semantics for normal programs that account for SLDNF resolution have been extensively investigated. Comprehensive surveys can be found in [5, 6]. The closest approach to ours is that taken in [7], which also maps normal programs into default theories. However, their mapping is far more complex than ours, and they investigate only the case of stratified programs.

This paper is organized as follows. Section 2 reviews the basic concept of default theory and SLDNF resolution that we need in the paper. Section 3 describes the default interpretation we use to explain SLDNF resolution. Section 4 proves the major results of the paper. Finally, Section 5 contains the conclusions.

2. Preliminaries

• *Defaults*

We review in this section some basic concepts of default logic. A detailed development can be found in [8].

A *default* is an expression of the form $(\alpha : \beta_1, \cdots, \beta_m/\omega)$, where α, $\beta_1, \cdots, \beta_m$, and ω are all first-order formulas. The formulas α and ω are called, respectively, the *prerequisite* and the *consequent* of the default, whereas the formulas $\beta_1, \cdots, \beta_m$ are called the *justifications*. A default is *closed* iff α, $\beta_1, \cdots, \beta_m$ and ω are sentences, that is, first-order formulas with no free variables. Otherwise, the default is *open*. A *normal* default is a default of the form $(\alpha : \omega/\omega)$.

For the purposes of this paper, it is also important to recall that a *CWA default* is a default of the form $(:\neg A/\neg A)$, where A is a ground-atomic formula over the first-order language in question.

A *default theory* is a pair $\Delta = (D, W)$, where D is a set of defaults and W is a set of first-order sentences. A default theory is *open, closed,* or *normal* iff D is a set of open, closed, or normal defaults.

A default theory is associated with a set of first-order theories, its *extensions*. The following theorem characterizes extensions as proposed in Theorem 2.1 of [8].

Theorem 1

Let E be a set of sentences and let $\Delta = (D, W)$ be a default theory. Define $E_0 = W$ and, for $i \geq 0$,

$$E_{i+1} = Th(E_i) \cup \left\{ \omega \mid \frac{\alpha : \beta_1, \cdots, \beta_m}{\omega} \in D, \text{ where } \alpha \in E_i \text{ and } \neg\beta_1, \cdots, \neg\beta_m \notin E \right\}.$$

Then, E is an extension for Δ iff

$$E = \bigcup_{i=0}^{\infty} E_i.$$

Given a default theory $\Delta = (D, W)$ and an extension E of Δ, the set of *generating defaults* for E with respect to Δ is

$$Gen(E, \Delta) = \left\{ \frac{\alpha : \beta_1, \cdots, \beta_m}{\omega} \in D \mid \alpha \in E \text{ and } \neg\beta_1, \cdots, \neg\beta_m \notin E \right\}.$$

Given a set of defaults D, we also define $Conseq(D)$ as the set of the consequents and $Prereq(D)$ as the set of the prerequisites of the defaults in D.

Lemma 1

Suppose that E is an extension of a default theory $\Delta = (D, W)$. Then

$$E = Th(W \cup Conseq(Gen(E, \Delta))).$$

Corollary 1

Let $\Delta = (D, W)$ be a default theory.

(a) Δ has an inconsistent extension iff W is inconsistent.
(b) If Δ has an inconsistent extension, then that is its only extension.

Theorem 2 (Minimality of extensions)

Let $\Delta = (D, W)$ be a default theory. If E and F are extensions of Δ and if $E \subseteq F$, then $E = F$.

Theorem 3 (Semimonotonicity of normal default theories) Let $\Delta = (D, W)$ and $\Delta' = (D', W)$ be two normal default theories such that $D' \subseteq D$. For every extension E' of Δ', there is an extension E of Δ such that $E' \subseteq E$.

- *SLD and SLDNF resolution*

In this section we briefly review SLD resolution and SLDNF resolution (abbreviated SLD and SLDNF). The reader familiar with [2, Ch. 3] may skip directly to Section 3.

We first recall some concepts directly related to SLD. An expression of the form $\mathsf{A} \leftarrow \mathsf{B}_1, \cdots, \mathsf{B}_n$ is a *definite clause* iff $\mathsf{A}, \mathsf{B}_1, \cdots, \mathsf{B}_n$ are positive literals. An expression of the form $\leftarrow \mathsf{B}_1, \cdots, \mathsf{B}_n$ is a *goal* iff $\mathsf{B}_1, \cdots, \mathsf{B}_n$ are positive literals. The literals $\mathsf{B}_1, \cdots, \mathsf{B}_n$ are called the *body* of the definite clause or the goal, and the literal A is called the *head* of the definite clause. The empty clause $\square$ is also considered to be a goal. A *program* is a set of definite clauses. Note that a goal $\leftarrow \mathsf{B}_1, \cdots, \mathsf{B}_n$ represents the negation of $\mathsf{B}_1 \wedge \cdots \wedge \mathsf{B}_n$. We refer the reader to [2] for the definitions of *SLD refutation* and *SLD tree.* In particular, we assume through the examples that the selection function always selects the leftmost literal.

In what follows, we denote the universal (or existential) closure of a formula F by $\forall\mathsf{F}$ (or $\exists\mathsf{F}$).

Let P be a program and G be a goal of the form $\leftarrow \mathsf{B}_1, \cdots, \mathsf{B}_n$. An *answer* to G from P is a substitution for the variables occurring in G. An answer α to G from P is *correct* iff $\forall(\mathsf{B}_1 \wedge \cdots \wedge \mathsf{B}_n)\alpha$ is a logical consequence of P. An answer α is *more general* than an answer β iff there is a substitution γ such that β is the composition of α with γ. Finally, an answer α to G from P is *computed* by SLD iff there is an SLD refutation R from $\mathsf{P} \cup \{\mathsf{G}\}$ such that α is the composition of the substitutions used in R, restricted to the variables in G.

SLD correctly computes answers in the following sense.

Theorem 4 (Soundness and completeness of SLD with respect to answers) [2]

(a) If α is an answer to a goal G from a program P which is computed by SLD, then α is correct.
(b) If α is a correct answer to a goal G from a program P, then there is an answer β to G from P computed by SLD which is more general than α.

SLDNF extends SLD to cope with negative literals and, hence, is set in a slightly different context. Briefly, a *program clause* and a *normal goal* are defined similarly to definite clauses and goals, except that the literals in the body may be both positive and negative. A *normal program* is a set of program clauses.

We again refer the reader to [2] for the definitions of *SLDNF refutation* and *SLDNF tree.* We just recall here that the process of constructing an SLDNF refutation or an SLDNF tree from $\mathsf{P} \cup \{\mathsf{G}\}$ will succeed only if, for every negative literal $\neg\mathsf{p(t)}$ that is selected,

- $\neg\mathsf{p(t)}$ is ground.
- There is a finitely failed SLDNF tree for $\mathsf{P} \cup \{\leftarrow \mathsf{p(t)}\}$, in which case $\neg\mathsf{p(t)}$ was canceled, or there is an SLDNF refutation from $\mathsf{P} \cup \{\leftarrow \mathsf{p(t)}\}$, in which case $\neg\mathsf{p(t)}$ was not canceled.

The notions of *answer* and *computed answer* extend directly to the context of SLDNF. However, the notion of *correct answer* does not, because it has long been recognized that the first-order reading of normal programs is not compatible with SLDNF; that is, SLDNF computes incorrect answers if we identify a program clause of the form $\mathsf{A} \leftarrow \mathsf{B}_1, \cdots, \mathsf{B}_n$ with the formula $\forall(\mathsf{B}_1 \wedge \cdots \wedge \mathsf{B}_n \Rightarrow \mathsf{A})$, and similarly for goals. Extending Theorem 4 above to SLDNF is in fact the major theme of this paper.

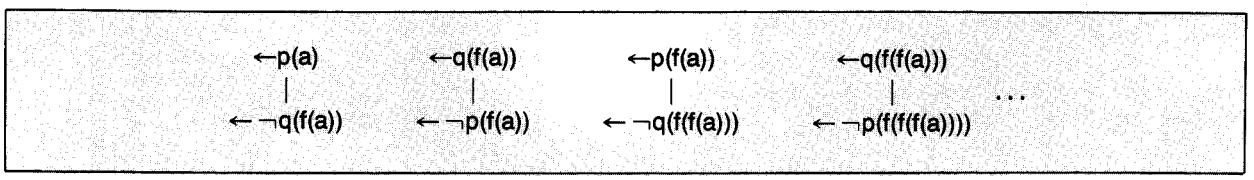
The key advantage of SLDNF is that it is straightforward to implement by extending a processor based on SLD, such as a standard Prolog interpreter. However, the way SLDNF processes negative literals has some special characteristics that we highlight in the rest of this section.

We begin by emphasizing that SLDNF is an inherently recursive process in the sense that, to construct an SLDNF refutation or an SLDNF tree, one may have to build other SLDNF refutations and trees. But the process does not involve self-loops; that is, if an SLDNF tree T (or refutation) is required during the construction of another SLDNF tree T' (or refutation), then T' is not required to build T. Thus, it is possible to define a *rank* for ground-negative literals with respect to a given normal program P as follows. For every ground-negative literal $\neg\mathsf{p(t)}$,

(a) $\neg\mathsf{p(t)}$ has *rank 0* with respect to P iff there is either a finitely failed SLDNF tree or an SLDNF refutation from $\mathsf{P} \cup \{\leftarrow \mathsf{p(t)}\}$ where no negative literal is selected.
(b) $\neg\mathsf{p(t)}$ has *rank* $k+1$ with respect to P iff there is either a finitely failed SLDNF tree or an SLDNF refutation from $\mathsf{P} \cup \{\leftarrow \mathsf{p(t)}\}$ where all negative literals that are selected are ground and have rank less than or equal to k, and at least one has rank k.

Thus, we may consider that the canceling of a negative literal proceeds in stages according to the rank of selected negative literals.

The search for an SLDNF refutation from a program and a goal may fail for many reasons, such as when an open negative literal is selected at any depth of the recursion, or when the process of canceling a negative literal diverges either because an infinite branch of a tree is reached or because an infinite number of trees is required.



We conclude this section with a very simple example of this last phenomenon.

Example 1
Let P be the normal program (clauses 1 and 2) and G be the normal goal (clause 3) below:

1. $p(x) \leftarrow \neg q(f(x))$.
2. $q(x) \leftarrow \neg p(x)$.
3. $\leftarrow \neg p(a)$.

There is no SLDNF refutation from $P \cup \{G\}$ essentially because, if one tries to build an SLDNF tree from $P \cup \{\leftarrow p(a)\}$, one will have to build a tree from $P \cup \{\leftarrow q(f(a))\}$, and from $P \cup \{\leftarrow p(f(a))\}$, and from $P \cup \{\leftarrow q(f(f(a)))\}$, and so on (see the box above).

3. Three default logic interpretations for normal programs and goals

In Section 2 we stressed that SLDNF has the flavor of a procedural method that may invoke itself recursively. As a consequence, there is little hope of defining clean semantics for normal programs and goals for which SLDNF is complete except, perhaps, for special classes of normal programs and goals. Therefore, we direct our efforts toward obtaining semantics which explain (the soundness of) SLDNF resolution as tightly as possible, but which do not directly reflect the procedural nature of the method.

We list the definitions and simple lemmas in this section, leaving the proofs of the more important results to Section 4.

• *Basic definitions*
We provide semantics for normal programs and goals indirectly by interpreting them into default logic. The definitions in this section convey the general idea.

In what follows, if A is a conjunction of literals $A_1 \wedge \cdots \wedge A_n$, let $\exists A$ indicate the existential closure of A and $\leftarrow A$ indicate the goal $\leftarrow A_1, \cdots, A_n$.

Definition 1
A *default logic interpretation* for normal programs and goals is a function Φ that maps each normal program P into a default theory $\Phi(P)$ and each conjunction of literals A into a formula $\Phi(A)$.

Definition 2
Let Φ be a default logic interpretation for normal programs and goals.

(a) SLDNF is $\exists$-*sound* with respect to Φ iff, for every normal program P and every normal goal $\leftarrow Q$, for every answer α to $\leftarrow Q$ from P, if α is computed by SLDNF, then there is an extension of $\Phi(P)$ containing $\forall\Phi(Q)\alpha$.
(b) SLDNF is $\forall$-*sound* with respect to Φ iff, for every normal program P and every normal goal $\leftarrow Q$, for every answer α to $\leftarrow Q$ from P, if α is computed by SLDNF, then every extension of $\Phi(P)$ contains $\forall\Phi(Q)\alpha$.

Definition 3
Let Φ be a default logic interpretation for normal programs and goals.

(a) SLDNF is $\exists$-*complete* with respect to Φ iff, for every normal program P and every normal goal $\leftarrow A$, for every answer α to $\leftarrow Q$ from P, if there is an extension of $\Phi(P)$ that contains $\forall\Phi(Q)\alpha$, then there is an answer β to $\leftarrow Q$ from P computed by SLDNF such that β is more general than α.
(b) SLDNF is $\forall$-*complete* with respect to Φ iff, for every normal program P and every normal goal $\leftarrow A$, for every answer α to $\leftarrow Q$ from P, if every extension of $\Phi(P)$ contains $\forall\Phi(Q)\alpha$, then there is an answer β to $\leftarrow Q$ from P computed by SLDNF such that β is more general than α.

Note that the prefixes $\forall$- and $\exists$- suggest how the extensions are quantified in the above definitions. Also observe that $\forall$-soundness is a more stringent notion than $\exists$-soundness, but $\forall$-completeness is less restrictive than $\exists$-completeness. Hence, one should strive to find a natural default logic interpretation for normal programs and goals for which SLDNF is $\forall$-sound and, hopefully, $\exists$-complete. We leave to later subsections the definition of the interpretations we consider, limiting ourselves in the next subsection to indicating why CWA defaults do not provide an appropriate basis.

• *An interpretation based on CWA defaults*
A natural strategy to explain SLDNF in terms of default logic would be to define an interpretation that maps each

normal program into the closed normal default theory whose defaults are the CWA defaults. More precisely:

Definition 4 (Default logic interpretation $\mathbb{C}$*)*
$\mathbb{C}$ is the interpretation that maps each normal program P into the default theory $\mathbb{C}(P) = (\boldsymbol{C}, P)$, where $\boldsymbol{C}$ is the set of all CWA defaults over the underlying alphabet, and that maps each conjunction of literals into itself.

However, SLDNF is not ∀-sound with respect to $\mathbb{C}$ essentially because, on one hand, default logic requires using the standard first-order semantics when reasoning about the extensions of a default theory, but, on the other hand, SLDNF is incompatible with the first-order reading of normal clauses and goals. The following example illustrates this point well.

Example 2
Let P be the normal program and G be the normal goal defined by the clauses below:

1. $q(a) \leftarrow \neg p(a)$. P
2. $\leftarrow \neg p(a)$. G

Assume that the only nonlogical symbols of the underlying first-order language are the constant a and the two unary predicate symbols p and q.

Then, there is an SLDNF refutation R from $P \cup \{G\}$ consisting of G followed by the empty clause. Moreover, the answer computed by R is the empty substitution ε.

However, if we adopt the standard first-order reading, clause 1 is equivalent to the first-order sentence $q(a) \vee p(a)$. Then, $\mathbb{C}(P) = (\boldsymbol{C}, P)$ has just two extensions, one containing p(a) and $\neg q(a)$, viz., that generated by firing the default $(:\neg q(a)/\neg q(a))$, and one containing q(a) and $\neg p(a)$, viz., that generated by firing the default $(:\neg p(a)/\neg p(a))$.

Hence, SLDNF is not ∀-sound with respect to $\mathbb{C}$, since $\neg p(a)$ does not belong to both extensions.

• *An interpretation based on barred CWA defaults*
We introduce in this subsection a default logic interpretation for normal programs for which SLDNF is ∀-sound. The general idea is to treat a predicate symbol preceded by ¬ as the name of a different predicate symbol. Thus, $\neg p(t)$ in principle does *not* denote the negation of p(t). To emphasize this aspect, we transform every negative literal $\neg p(t)$ into the *barred literal* $\bar{p}(t)$ and propagate this transformation to normal programs and goals. However, a ground-negative literal $\neg p(t)$ and its barred transform $\bar{p}(t)$ remain related by the *$\overline{CWA}$ default* $(:\neg p(t)/\bar{p}(t))$, which intuitively says to accept $\bar{p}(t)$ as valid if it is consistent to assume $\neg p(t)$. Finally, we define the interpretation $\bar{\mathbb{C}}$, which maps a normal program P into the default theory $\bar{\mathbb{C}}(P) = (\bar{\boldsymbol{C}}, \bar{P})$, where $\bar{\boldsymbol{C}}$ is the set of all $\overline{CWA}$ defaults over the underlying alphabet and $\bar{P}$ is the barred transform of P, and which maps each conjunction of literals into its barred transform.

We give precise definitions below for these concepts, and show that barred literals in some sense behave as negated literals in the context of $\overline{CWA}$ defaults. We then show in Section 4 that SLDNF is ∀-sound with respect to $\bar{\mathbb{C}}$. All definitions that follow are relative to a fixed first-order alphabet $\boldsymbol{A}$.

Definition 5

(a) The *barred augmented alphabet* corresponding to $\boldsymbol{A}$, denoted by $\bar{\boldsymbol{A}}$, is obtained by adding to $\boldsymbol{A}$ the new symbol $\bar{p}$ as an n-ary predicate symbol, for each n-ary predicate symbol p in $\boldsymbol{A}$.
(b) A positive literal M over $\bar{\boldsymbol{A}}$ is a *barred literal* iff M is of the form $\bar{p}(t)$. A positive literal M over $\bar{\boldsymbol{A}}$ is a *positive nonbarred literal* iff M is of the form p(t).
(c) A positive literal M over $\bar{\boldsymbol{A}}$ is the *barred complement* of a positive literal L over $\boldsymbol{A}$ iff M is of the form $\bar{p}(t)$ and L is of the form p(t). The barred complement of L is denoted by $\bar{L}$.
(d) The *barred transform* of a negative literal over $\boldsymbol{A}$ of the form $\neg p(t)$ is the literal $\bar{p}(t)$ over $\bar{\boldsymbol{A}}$.
(e) Let P be a program clause (respectively, a normal goal, a conjunction of literals, or a set of program clauses). The *barred transform* of P, denoted by $\bar{P}$, is the definite clause (respectively, the goal, the conjunction of literals, or the set of definite clauses) obtained by replacing each occurrence in P of a negative literal with its barred transform.
(f) A $\overline{CWA}$ default is a closed, non-normal default of the form $(:\neg L/\bar{L})$, denoted by δ_L, where L is a ground-positive nonbarred literal and $\bar{L}$ is the barred complement of L. We also say that $\neg L$ is the *justification* and that $\bar{L}$ is the *consequent* of $(:\neg L/\bar{L})$. We denote the set of all $\overline{CWA}$ defaults by $\bar{\boldsymbol{C}}$.

Definition 6 (Default logic interpretation $\bar{\mathbb{C}}$*)*
$\bar{\mathbb{C}}$ is the interpretation that maps each normal program P into the default theory $\bar{\mathbb{C}}(P) = (\bar{\boldsymbol{C}}, \bar{P})$, where $\bar{\boldsymbol{C}}$ is the set of all $\overline{CWA}$ defaults over the underlying alphabet and $\bar{P}$ is the barred transform of P, and that maps each conjunction of literals A into its barred transform $\bar{A}$.

The following example, which should be compared with Example 2, illustrates the interpretation $\bar{\mathbb{C}}$.

Example 3
Let P and G be as in Example 2. Then, $\bar{P}$ is the (definite clause) program and $\bar{G}$ is the goal defined by the clauses below:

1. $q(a) \leftarrow \bar{p}(a)$. $\bar{P}$
2. $\leftarrow \bar{p}(a)$. G

Recall from Example 2 that p and q are the only predicate symbols and that a is the only constant, by assumption. Also recall that the empty substitution ε is an answer computed by SLDNF [therefore, the universal closure of $\neg p(a)\varepsilon$ is simply $\neg p(a)$].

Now, $\bar{\mathbb{C}}(P) = (\bar{C}, \bar{P})$ has just one extension E, which is generated by firing the default $\delta_p = (:\neg p(a)/\bar{p}(a))$, but not the default $\delta_q = (:\neg q(a)/\bar{q}(a))$. Indeed, nothing prevents the default δ_p from firing, which means that any extension of $\bar{\mathbb{C}}(P)$ must contain $\bar{p}(a)$ and, hence, $q(a)$, in view of clause 1. But this in turn implies that δ_q can never be fired.

Note that, since $\bar{\mathbb{C}}(\neg p(a)) = \bar{p}(a)$, since the universal closure of $\bar{p}(a)\varepsilon$ is equivalent to $\bar{p}(a)$, and since E contains $\bar{p}(a)$, E also contains the universal closure of $\bar{\mathbb{C}}(\neg p(a))\varepsilon$. Therefore, P and G are not a counterexample for the $\forall$-soundness of SLDNF with respect to $\bar{\mathbb{C}}$, whereas they are for the $\forall$-soundness of SLDNF with respect to $\mathbb{C}$.

Furthermore, observe that, when compared with Example 2, the only change is that clause 1 in P, after the transformation induced by $\bar{\mathbb{C}}$, becomes the first-order equivalent of the sentence $q(a) \vee \neg\bar{p}(a)$, whereas, after the transformation induced by $\mathbb{C}$, it was the first-order equivalent of the sentence $q(a) \vee p(a)$.

We now state two properties of $\bar{\mathbb{C}}$ that we will need to prove the main results in Section 4. They follow directly from theorems stated in the subsection on defaults.

Lemma 2
Let P be a normal program.

(a) A set E of sentences over $\bar{A}$ is an extension of $\bar{\mathbb{C}}(P)$ iff $E = Th(\bar{P} \cup T)$, where $T = \{\bar{L} \mid (:\neg L/\bar{L}) \in \bar{C} \wedge L \notin E\}$.
(b) $\bar{\mathbb{C}}(P)$ has no inconsistent extension.

Proof

(a) From Theorem 1, we know that E is an extension of $\bar{\mathbb{C}}(P)$ iff $E = E_0 \cup E_1 \cup E_2$, where $E_0 = \bar{P}$, $E_1 = Th(E_0) \cup T$, and $E_2 = Th(E_1)$, since $Th(E_2) = Th(Th(E_1)) = Th(E_1)$. Therefore, $E = Th(Th(\bar{P}) \cup T) = Th(\bar{P} \cup T)$.
(b) Since $\bar{P}$ is a set of definite clauses, it is always consistent; hence, the lemma follows from Corollary 1. □

The next list of results provides a semantic justification for the transformation $\bar{\mathbb{C}}$.

Given a set of sentences or a set of clauses S, we denote by $\boldsymbol{B}_S$ the Herbrand base for S. Also, given a set of definite clauses D, we denote by $\boldsymbol{M}(D)$ the unique minimal Herbrand model of D and by T_D the mapping that takes each Herbrand interpretation $\boldsymbol{I}$ of D into the Herbrand interpretation

$$T_D(\boldsymbol{I}) = \{A \in \boldsymbol{B}_D \mid A \leftarrow A_1, \cdots, A_n \text{ is a ground instance of a clause in D and } \{A_1, \cdots, A_n\} \subseteq \boldsymbol{I}\}.$$

Theorem 5
Let P be a normal program and E be an extension of $\bar{\mathbb{C}}(P)$. Let

$$T = \{\bar{L} \mid (:\neg L/\bar{L}) \in \bar{C} \wedge L \notin E\} \text{ and } Q = \bar{P} \cup T.$$

(a) E has a unique minimal Herbrand model $\boldsymbol{M}(E)$.
(b) $N \in E$ iff $N \in \boldsymbol{M}(E)$, for any ground-positive literal (barred or not) N.
(c) $B \in T_Q \uparrow \omega$ iff $B \in T$, for any ground-positive barred literal B.
(d) $L \in E$ iff $\bar{L} \notin E$, for any ground-positive nonbarred literal L.
(e) $\boldsymbol{M}(E)$ satisfies $\neg L$ iff $\bar{L} \in \boldsymbol{M}(E)$, for any ground-positive nonbarred literal L.
(f) $\boldsymbol{M}(E)$ is a model of P.

Proof

(a) Q has a unique minimal Herbrand model, since Q is a set of definite clauses, and so does E, since $E = Th(Q)$, by Lemma 2 and the definition of Q.
(b) Let N be a ground-positive literal (barred or not). Then,

$N \in E$	iff $Q \models N$	by Lemma 2 and definition of Q,
	iff $N \in \boldsymbol{M}(Q)$	by Theorem 6.2 of [2], since $N \in \boldsymbol{B}_Q$, and
	iff $N \in \boldsymbol{M}(E)$	since $\boldsymbol{M}(Q) = \boldsymbol{M}(E)$, by Lemma 2 and the definition of Q.

(c) Let B be a ground-positive barred literal. Then, $B \in T_Q \uparrow \omega$ iff $B \in T$, because B cannot be the head of any ground instance of a clause in $\bar{P}$, since the program clauses in P cannot have negative literals as their heads.
(d) Let L be a ground-positive nonbarred literal. Then,

$L \in E$	iff $\bar{L} \notin T$	by definition of T,
	iff $\bar{L} \notin T_Q \uparrow \omega$	by (c),
	iff $\bar{L} \notin \boldsymbol{M}(Q)$	by Theorem 6.5 of [2], and
	iff $\bar{L} \notin E$	by (b) and since $\boldsymbol{M}(Q) = \boldsymbol{M}(E)$, by Lemma 2 and the definition of Q.

(e) Let L be a ground-positive nonbarred literal. Then, $\boldsymbol{M}(E)$ satisfies $\neg L$ iff $L \notin \boldsymbol{M}(E)$, by definition of satisfiability in Herbrand interpretations, iff $L \in \boldsymbol{M}(E)$, by (b) and (d).
(f) First observe that the alphabet $\bar{A}$ contains the alphabet A. Hence, $\boldsymbol{M}(E)$ is in fact a Herbrand interpretation for P. It suffices to show that, for any ground instance B

of a normal clause in $\mathbf{P}$, if $\boldsymbol{M}(E)$ satisfies the body of $\mathbf{B}$, then $\boldsymbol{M}(E)$ also satisfies the head of $\mathbf{B}$.

Let $\mathbf{B}$ be a ground instance of a normal clause in $\mathbf{P}$ and suppose that $\boldsymbol{M}(E)$ satisfies the body of $\mathbf{B}$. Let $\bar{\mathbf{B}}$ be the barred transform of $\mathbf{B}$. Note that if $\neg\mathbf{L}$ is a negative literal in the body of $\mathbf{B}$, then $\boldsymbol{M}(E)$ satisfies $\bar{\mathbf{L}}$, by (e). Hence, directly from our assumption, we obtain that

$\boldsymbol{M}(E)$ satisfies the body of $\bar{\mathbf{B}}$. (*)

But $\bar{\mathbf{B}}$ is also a ground instance of a clause in $\bar{\mathbf{P}}$ and $\boldsymbol{M}(E)$ is a model of $\bar{\mathbf{P}}$, since $\bar{\mathbf{P}}$ is contained in E. Therefore, we also have

$\boldsymbol{M}(E)$ satisfies $\bar{\mathbf{B}}$. (**)

Thus, from (*) and (**), we obtain that $\boldsymbol{M}(E)$ satisfies the head of $\bar{\mathbf{B}}$. Finally, note that the head of $\bar{\mathbf{B}}$ coincides with the head of $\mathbf{B}$, since the head of $\mathbf{B}$ is not a negative literal. Hence, $\boldsymbol{M}(E)$ satisfies the head of $\mathbf{B}$, as was to be shown. □

Note that these simple results follow essentially because *the barred transform of a normal program is a definite clause program* and that, as item (c) shows, *barred literals (representing negative information) can only be generated by firing defaults,* which is consistent with the intuitive idea behind the syntax of normal programs.

Item (a) should be compared with Corollary 3.10 in [7], which shows that every extension of a positivistic default theory has a unique Herbrand model, whereas item (d) should be compared with Lemma 3.1 in [7]. Now, item (e) indicates that a ground-barred literal $\bar{\mathbf{L}}$ indeed behaves as the negation of $\mathbf{L}$ with respect to the unique minimal Herbrand model of each extension of $\bar{\mathbb{C}}(\mathbf{P})$. Finally, item (f) shows that the default logic interpretation $\bar{\mathbb{C}}$ ultimately associates with each normal program $\mathbf{P}$ a set of Herbrand interpretations which are the minimal models of the extensions of $\bar{\mathbb{C}}(\mathbf{P})$.

We can also show that the set of unique minimal Herbrand models of the extensions of $\bar{\mathbb{C}}(\mathbf{P})$ corresponds exactly to the set of stable models of $\mathbf{P}$ [9].

We first recall the definition of stable models. The *GL transformation* is the mapping γ that takes a normal program $\mathbf{P}$ and a Herbrand interpretation $\boldsymbol{I}$ for $\mathbf{P}$ into a new program $\gamma(\mathbf{P},\boldsymbol{I})$, obtained from $\mathbf{P}$ by performing the following two reductions:

- Removing from $\mathbf{P}$ all clauses in which the body contains a negative literal $\neg\mathbf{A}$ such that $\mathbf{A}\in\boldsymbol{I}$.
- Removing from the body of each remaining clause those negative literals $\neg\mathbf{A}$ such that $\mathbf{A}\notin\boldsymbol{I}$.

Note that the new program $\gamma(\mathbf{P},\boldsymbol{I})$ contains only definite clauses and, hence, has a unique minimal Herbrand model $\boldsymbol{M}(\gamma(\mathbf{P},\boldsymbol{I}))$. We then define the *Gelfond–Lifschitz operator* $\Gamma_{\mathbf{P}}$ for $\mathbf{P}$ as $\Gamma_{\mathbf{P}}(\boldsymbol{I}) = \boldsymbol{M}(\gamma(\mathbf{P},\boldsymbol{I}))$. It can be proved that the fixed points of $\Gamma_{\mathbf{P}}$ are minimal models of $\mathbf{P}$ [9]. Finally, we say that a Herbrand interpretation I for $\mathbf{P}$ is a *stable model* of $\mathbf{P}$ iff $\Gamma_{\mathbf{P}}(\boldsymbol{I}) = \boldsymbol{I}$.

We also need some auxiliary definitions. Let $\boldsymbol{A}$ be a first-order alphabet and $\bar{\boldsymbol{A}}$ be the corresponding barred augmented alphabet. If $\boldsymbol{I}$ is a Herbrand structure for $\boldsymbol{A}$, define

$$\alpha(\boldsymbol{I}) = \boldsymbol{I}\cup\{\bar{\mathbf{L}}\mid \mathbf{L} \text{ is a nonbarred ground atom and } \mathbf{L}\notin\boldsymbol{I}\},$$

and, if $\boldsymbol{I}$ is a Herbrand structure for $\bar{\boldsymbol{A}}$, define

$$\beta(\boldsymbol{I}) = \{\mathbf{L}\mid \mathbf{L} \text{ is a nonbarred ground atom and } \mathbf{L}\in\boldsymbol{I}\}.$$

We are now ready to prove the following result.

Theorem 6
Let $\mathbf{P}$ be a normal program. Then,

(a) If E is an extension of $\bar{\mathbb{C}}(\mathbf{P})$ and $\boldsymbol{M}(E)$ is its unique minimal model, then $\beta(\boldsymbol{M}(E))$ is a stable model of $\mathbf{P}$.
(b) If $\boldsymbol{I}$ is a stable model of $\mathbf{P}$, then $\alpha(\boldsymbol{I})$ is the unique minimal model of an extension E of $\bar{\mathbb{C}}(\mathbf{P})$ and $E = Th(\bar{\mathbf{P}}\cup T)$, where $T = \{\bar{\mathbf{L}}\mid(:\neg\mathbf{L}/\bar{\mathbf{L}})\in\bar{\boldsymbol{C}}\wedge\mathbf{L}\notin\boldsymbol{I}\}$.

Proof

(a) Let $T = \{\bar{\mathbf{L}}\mid(:\neg\mathbf{L}/\bar{\mathbf{L}})\in\bar{\boldsymbol{C}}\wedge\mathbf{L}\notin E\}$ and $\mathbf{Q} = \bar{\mathbf{P}}\cup T$. Let $\boldsymbol{M}(\mathbf{Q})$ be the unique minimal model of $\mathbf{Q}$. Let $\mathbf{G} = \gamma(\mathbf{P},\beta(\boldsymbol{M}(E)))$ and $\boldsymbol{M}(\mathbf{G})$ be the unique minimal model of $\mathbf{G}$. Then, we have

$$\begin{aligned}
\beta(\boldsymbol{M}(E)) &= \beta(\boldsymbol{M}(\mathbf{Q})) && \text{since } E = Th(\bar{\mathbf{P}}\cup T) = Th(\mathbf{Q}) \text{ and, thus, } \boldsymbol{M}(E) = \boldsymbol{M}(\mathbf{Q}),\\
&= \beta(T_{\mathbf{Q}}\uparrow\omega) && \text{since } \boldsymbol{M}(\mathbf{Q}) = T_{\mathbf{Q}}\uparrow\omega, \text{ by definition of } T,\\
&= T_{\mathbf{G}}\uparrow\omega && \text{since } \bar{\mathbf{L}}\in T \text{ iff } \mathbf{L}\notin\beta(\boldsymbol{M}(E)),\\
&= \boldsymbol{M}(\mathbf{G}) && \text{by definition of } T, \text{ and}\\
&= \Gamma_{\mathbf{P}}(\beta(\boldsymbol{M}(E))) && \text{by definition of } \Gamma_{\mathbf{P}} \text{ and } \mathbf{G}.
\end{aligned}$$

Hence, $\beta(\boldsymbol{M}(E)) = \Gamma_{\mathbf{P}}(\beta(\boldsymbol{M}(E)))$, which implies that $\beta(\boldsymbol{M}(E))$ is a stable model of $\mathbf{P}$.

(b) Suppose that $\boldsymbol{I}$ is a stable model of $\mathbf{P}$. Then, $\boldsymbol{I} = \Gamma_{\mathbf{P}}(\boldsymbol{I})$. Let $E = Th(\bar{\mathbf{P}}\cup T)$, where $T = \{\bar{\mathbf{L}}\mid(:\mathbf{L}/\bar{\mathbf{L}})\in\bar{\boldsymbol{C}}\wedge\mathbf{L}\notin\boldsymbol{I}\}$. Note that, if $\mathbf{L}$ is a nonbarred ground-positive literal, then $\mathbf{L}\notin\boldsymbol{I}$ iff $\mathbf{L}\notin\alpha(\boldsymbol{I})$ iff $\mathbf{L}\notin E$. Hence, $T = \{\bar{\mathbf{L}}\mid(:\neg\mathbf{L}/\bar{\mathbf{L}})\in\bar{\boldsymbol{C}}\wedge\mathbf{L}\notin E\}$, which implies that E is indeed an extension of $\bar{\mathbb{C}}(\mathbf{P})$.

Let $\boldsymbol{M}(E)$ be the unique minimal model of E, which exists by Theorem 5(a). Let $\mathbf{G} = \gamma(\mathbf{P},\boldsymbol{I})$ and $\boldsymbol{M}(\mathbf{G})$ be the unique minimal model of $\mathbf{G}$. We now show that $\boldsymbol{I} = \boldsymbol{M}(E)$. Indeed,

$$
\begin{aligned}
M(E) &= T_{P\cup T} \uparrow \omega && \text{by definition of } T \text{ and } E,\\
&= \alpha(T_G \uparrow \omega) && \text{since } \bar{L} \in T \text{ iff } L \notin I,\\
&= \alpha(M(G)) && \text{by definition of } T,\\
&= \alpha(\Gamma_P(I)) && \text{by definition of } \Gamma_P, \text{ and}\\
&= \alpha(I) && \text{since } I \text{ is a stable model of } P. \ \square
\end{aligned}
$$

This concludes the preliminary discussion of $\bar{\mathbb{C}}$.

• *An interpretation based on NFF defaults*

Given a normal program P, we now introduce a third default logic interpretation, $\bar{\mathbb{N}}$, which is much closer to SLDNF than the default logic interpretation $\bar{\mathbb{C}}$, since the former directly encodes the way negative literals are canceled in SLDNF. We then prove in Section 4 that SLDNF is ∀-sound and ∃-complete with respect to $\bar{\mathbb{N}}$.

Definition 7

Let P be a normal program. A $\overline{CWA}$ default δ_L is an $\overline{NFF}$ *default* for P iff there is a finitely failed SLDNF tree for $P \cup \{\leftarrow L\}$. We denote the set of all $\overline{NFF}$ defaults for P by $\bar{N}_P$.

Definition 8 (Default logic interpretation $\bar{\mathbb{N}}$)

$\bar{\mathbb{N}}$ is the interpretation that maps each normal program P into the default theory $\bar{\mathbb{N}}(P) = (\bar{N}_P, \bar{P})$, where $\bar{N}_P$ is the set of all $\overline{NFF}$ defaults for P and $\bar{P}$ is the barred transform of P, and that maps each conjunction of literals A into its barred transform $\bar{A}$.

4. Results

We prove in this section results whose major implications (P is a normal program) are as follows:

(a) $\bar{\mathbb{C}}(P)$ may have no extensions, whereas $\bar{\mathbb{N}}(P)$ always has a unique extension.

(b) SLDNF is very strong in the sense that, given a ground-negative literal $\neg L$, if there is a finitely failed SLDNF tree for $P \cup \{\leftarrow L\}$, then δ_L can always be fired in $\bar{\mathbb{C}}(P)$; that is, it belongs to the set of generating defaults of all extensions of $\bar{\mathbb{C}}(P)$, if there is one.

(c) Moreover, given a ground-negative literal $\neg L$, if there is an SLDNF refutation for $P \cup \{\leftarrow L\}$, then δ_L can never be fired in $\bar{\mathbb{C}}(P)$; that is, it does not belong to the set of generating defaults of any extension of $\bar{\mathbb{C}}(P)$, if there is one.

(d) As a consequence, if δ_L belongs to the set of generating defaults of some extension of $\bar{\mathbb{C}}(P)$, but not all, then there is neither a finitely failed SLDNF tree for $P \cup \{\leftarrow L\}$ nor an SLDNF refutation for $P \cup \{\leftarrow L\}$.

(e) However, given a ground-negative literal $\neg L$, there may be neither a finitely failed SLDNF tree nor an SLDNF refutation for $P \cup \{\leftarrow L\}$, and yet δ_L may belong to the set of generating defaults of all extensions of $\bar{\mathbb{C}}(P)$, if there is one.

• *∃-soundness with respect to CWA defaults*

We show in this subsection that SLDNF is ∃-sound with respect to $\bar{\mathbb{C}}$. The result follows from a property of SLDNF posed as a challenge to the reader in [2, Example 32, Ch. 3].

Theorem 7

Let P be a normal program and G be a normal goal. If $P \cup \{G\}$ has a finitely failed SLDNF tree, then $P \cup \{G\}$ is consistent.

Proof

Let P be a normal program and G be a normal goal. Suppose that $P \cup \{G\}$ has a finitely failed SLDNF tree T. Define the *failure support* of T, $FS(T)$, as the set of literals that correspond to the following:

- Let H be a leaf of T and M be the literal selected from H. Then, $FS(T)$ contains all ground instances of M.
- Let H be an internal node of T and M be the literal selected from H. Suppose that M is positive and that $H_1, \cdots, H_r$ are the children of H in T. Then, $FS(T)$ contains
  - All ground instances of M that do not unify with the head of any clause in P.
  - All ground instances of M that unify with the head of a ground instance $A \leftarrow B_1, \cdots, B_n$ and $\{B_1, \cdots, B_n\} \cap FS(T) \neq \emptyset$.

Define $H(P)$ as the set of all positive literals L such that L is the head of a ground instance of a clause in P. We shall prove that $H = H(P) - FS(T)$ is a model for $P \cup \{G\}$.

We first show that H is a model of G by showing that H is a model of all ground instances of G. Suppose that G is of the form $\leftarrow M_1, \cdots, M_q$. Let φ be a substitution such that $(\leftarrow M_1, \cdots, M_q)\varphi$ is ground. First observe that $\{M_1\varphi, \cdots, M_q\varphi\} \cap FS(T) \neq \emptyset$, because otherwise T would not be a failed tree. Let $i_0 \in [1, q]$ be such that $M_{i_0}\varphi \in FS(T)$.

Suppose that $M_{i_0}\varphi$ is negative, that is, of the form $\neg A$. Then $P \cup \{\leftarrow A\}$ has a successful SLDNF tree and, hence, $A \in H(P)$ and $A \notin FS(T)$. Hence, $A \in H$, which implies that H is a model of $\neg(M_1\varphi \wedge \cdots \wedge M_q\varphi)$. Suppose that $M_{i_0}\varphi$ is positive. Then $M_{i_0}\varphi \in FS(T)$ immediately implies that $M_{i_0}\varphi \notin H$. Hence, H is a model of $\neg(M_1\varphi \wedge \cdots \wedge M_q\varphi)$. Therefore, H is a model for all ground instances of G; hence, H is a model for G.

We now show that H is a model of P. Let $C \leftarrow L_1, \cdots, L_n$ be a clause of P. Let φ be a substitution such that $(C \leftarrow L_1, \cdots, L_n)\varphi$ is ground. If $C\varphi \notin FS(T)$, then $C\varphi \in H$ and, hence, H is a model for $(C \leftarrow L_1, \cdots, L_n)\varphi$. If $C\varphi \in FS(T)$, then $\{L_1\varphi, \cdots, L_n\varphi\} \cap FS(T) \neq \emptyset$. As previously proved for the goal, we may conclude that H is a model for $(L_1, \cdots, L_n)\varphi$. Hence, H is a model for

$(C \leftarrow L_1, \cdots, L_n)\varphi$. Therefore, H is a model of all ground instances of all clauses of P. Therefore, H is a model for P. □

Theorem 8

Let P be a normal program and $\leftarrow Q$ be a normal goal. If α is an answer to $\leftarrow Q$ from P computed by SLDNF, then there is an extension of $\mathbb{C}(P)$ that contains $\forall Q\alpha$.

Proof

Let R be an SLDNF refutation from $P \cup \{\leftarrow Q\}$ that computes α. Let $D = \{\neg L_1, \cdots, \neg L_n\}$ be the set of negative literals selected in R. We shall show that $P \cup D$ is consistent.

Indeed, for each $i \in [1, n]$, the SLDNF tree for $P \cup \{\leftarrow L_i\}$ is finitely failed. Hence, the SLDNF tree for $P \cup \{\leftarrow L_1, \cdots, L_n\}$ is finitely failed. Therefore, by the previous theorem, $P \cup \{\leftarrow L_1, \cdots, L_n\}$ is consistent; that is, $P \cup D$ is consistent.

Now, if $P \cup D$ is consistent, then the normal default theory (δ_D, P), where $\delta_D = \{\delta_L \mid \neg L \in D\}$, has exactly one extension. Moreover, R induces a refutation (using linear resolution) from $P \cup D \cup \{\leftarrow Q\}$ that still computes α. Hence, by [1], we know that $P \cup D \models \forall Q\alpha$. Since $E = Th(P \cup D)$, we then have $\forall Q\alpha \in E$. Now, since $D \subseteq C$, by Theorem 3, there is an extension of $\bar{\mathbb{C}}(P) = (C, P)$ that contains $\forall Q\alpha$. □

- *$\forall$-soundness and $\exists$-completeness with respect to NFF defaults*

Let P be a normal program and G be a normal goal. To a first approximation, one may say that, if a ground-negative literal $\neg p(t)$ is canceled in an SLDNF refutation from $P \cup \{G\}$, then it is consistent with P. This assertion is in principle warranted by the existence of a finitely failed SLDNF tree T from $P \cup \{\leftarrow p(t)\}$. However, close scrutiny reveals that it is false, since the construction of T may require canceling other negative literals; that is, it may recursively invoke SLDNF. Hence, a better approximation would be to say that, if a ground-negative literal $\neg p(t)$ is canceled, then it is consistent with P and with all other ground-negative literals that can be canceled. This recursive statement is not paradoxical, and it can be formulated in the context of the default logic interpretation $\bar{\mathbb{N}}$ to mean that all defaults belonging to the set $\bar{N}_P$ can be simultaneously fired. This implies that $\bar{\mathbb{N}}(P)$ has a unique extension, which is our first result.

Theorem 9

Let P be a normal program. Then, $\bar{\mathbb{N}}(P)$ has a unique extension, which is $\bar{E} = Th(\bar{P} \cup Conseq(\bar{N}_P))$.

Proof

We show that, for every default $\delta_L \in \bar{N}_P$, $\leftarrow L$ is consistent with $\bar{P} \cup Conseq(\bar{N}_P)$. Hence, by Lemma 2, $\bar{E} = Th(\bar{P} \cup Conseq(\bar{N}_P))$ is an extension of $\bar{\mathbb{N}}(P) = (\bar{N}_P, \bar{P})$. Therefore, by the minimality of extensions (Theorem 2), since $\bar{N}_P$ is the set of all defaults of $\bar{\mathbb{N}}(P)$, $\bar{E}$ is the only extension.

Suppose that there is $\delta_L \in \bar{N}_P$ such that $\leftarrow L$ is not consistent with $\bar{P} \cup Conseq(\bar{N}_P)$. Since this set contains only definite clauses, there is then an SLD refutation $\bar{R}$ from $P \cup Conseq(\bar{N}_P) \cup \{\leftarrow L\}$. Let R be the sequence of goals obtained by replacing each occurrence of $\bar{M}$ in $\bar{R}$ with $\neg M$, for each barred literal $\bar{M} \in Conseq(\bar{N}_P)$. Observe that $\bar{M} \in Conseq(\bar{N}_P)$ iff M is ground and there is a finitely failed SLDNF tree for $P \cup \{\leftarrow M\}$, by definition of $\bar{N}_P$. Hence, R is an SLDNF refutation for $P \cup \{\leftarrow L\}$. Therefore, there is no failed SLDNF tree for $P \cup \{\leftarrow L\}$, which implies that $\delta_L \notin \bar{N}_P$, a contradiction. □

Theorem 10 ($\forall$-soundness of SLDNF with respect to $\bar{\mathbb{N}}$)

Let P be a normal program and $\leftarrow Q$ be a normal goal. Let $\bar{E}$ be the unique extension of $\bar{\mathbb{N}}(P)$. If α is an answer to $\leftarrow Q$ from P computed by SLDNF, then $\forall \bar{Q}\alpha$ belongs to $\bar{E}$.

Proof

Suppose that α is an answer to $\leftarrow Q$ from P computed by SLDNF. Then, there is an SLDNF refutation R for $P \cup \{\leftarrow Q\}$ such that α is the composition of the substitutions in R, restricted to the variables in Q. Let $\bar{R}$ be the sequence of goals obtained by replacing each occurrence of $\neg M$ in R with $\bar{M}$, for each negated literal $\neg M$. Since R is an SLDNF refutation, if $\neg M$ was canceled in R, then $\neg M$ is ground and there is a finitely failed SLDNF tree for $P \cup \{\leftarrow M\}$. Hence, $\delta_M \in \bar{N}_P$. Therefore, $\bar{R}$ is an SLD refutation from $\bar{P} \cup Conseq(\bar{N}_P) \cup \{\leftarrow \bar{Q}\}$. Moreover, R and $\bar{R}$ perform the same substitutions. Thus, α is also the composition of the substitutions in $\bar{R}$, restricted to the variables in $\bar{Q}$. Hence, we may conclude that $\bar{P} \cup Conseq(\bar{N}_P) \vdash \forall \bar{Q}\alpha$, by Theorem 7.1 of [2] (soundness of SLD resolution). Since $\bar{E} = Th(\bar{P} \cup Conseq(\bar{N}_P))$, by Theorem 9, we finally obtain that $\forall \bar{Q}\alpha$ belongs to $\bar{E}$. □

Theorem 11 ($\exists$-completeness of SLDNF with respect to $\bar{\mathbb{N}}$)

Let P be a normal program and $\leftarrow Q$ be a normal goal. Let $\bar{E}$ be the unique extension of $\bar{\mathbb{N}}(P)$. Let α be an answer to $\leftarrow Q$ from P. If $\forall \bar{Q}\alpha$ belongs to $\bar{E}$, then there is an answer β to $\leftarrow Q$ from P computed by SLDNF such that β is more general than α.

Proof

Suppose that α is an answer to $\leftarrow Q$ from P such that $\forall \bar{Q}\alpha$ belongs to $\bar{E}$. Then, since $\bar{E} = Th(\bar{P} \cup Conseq(\bar{N}_P))$, α is in fact a correct answer to $\leftarrow \bar{Q}$ from $\bar{P} \cup Conseq(\bar{N}_P)$. By Theorem 8.6 of [2] (completeness of SLD resolution), there is an answer β to $\leftarrow \bar{Q}$ from $\bar{P} \cup Conseq(\bar{N}_P)$ computed by SLD such that β is more general than α. We show that β is an answer to $\leftarrow Q$ from P computed by SLDNF.

Let $\bar{R}$ be an SLD refutation for $\bar{\mathsf{P}} \cup \mathit{Conseq}(\bar{N}_{\mathsf{P}}) \cup \{\leftarrow \bar{\mathsf{Q}}\}$ that computes β. As in the proof of Theorem 10, there is then an SLDNF refutation R for $\mathsf{P} \cup \{\leftarrow \mathsf{Q}\}$ that computes β. Therefore, β is an answer to $\leftarrow \mathsf{Q}$ from P computed by SLDNF such that β is more general than α. □

• *∀-soundness with respect to barred CWA defaults*

The results in the preceding subsection are somewhat unsatisfactory because the definition of $\overline{NFF}$ defaults refers directly to SLDNF. The results in this subsection, however, build upon them to prove that SLDNF is ∀-sound with respect to $\bar{\mathbb{C}}$ (with one proviso). This is a much more satisfactory situation, because $\bar{\mathbb{C}}$ provides a superior default logic interpretation for normal programs and goals, since $\overline{CWA}$ defaults are defined independently of SLDNF.

Let $\mathit{Ext}(\Delta)$ denote the set of all extensions of a default theory Δ.

Theorem 12

Let P be a normal program. Let $\bar{E}$ be the unique extension of $\bar{\mathbb{N}}(\mathsf{P})$. Suppose that $\bar{\mathbb{C}}(\mathsf{P})$ has at least one extension. Then, $\bar{E} \subseteq \cap\, \mathit{Ext}(\bar{\mathbb{C}}(\mathsf{P}))$.

Proof

Suppose that $\bar{\mathbb{C}}(\mathsf{P})$ has at least one extension, and let $\bar{F}$ be an extension of $\bar{\mathbb{C}}(\mathsf{P})$. We shall prove that $\bar{N}_{\mathsf{P}} \subseteq \mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))$. Hence, $\bar{E} \subseteq \bar{F}$, which implies that $\bar{E} \subseteq \cap\, \mathit{Ext}(\bar{\mathbb{C}}(\mathsf{P}))$.

In fact, we prove that, if there is a finitely failed SLDNF tree for $\mathsf{P} \cup \{\leftarrow \mathsf{L}\}$, then $\delta_{\mathsf{L}} \in \mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))$ and, if there is an SLDNF refutation for $\mathsf{P} \cup \{\leftarrow \mathsf{L}\}$, then $\delta_{\mathsf{L}} \notin \mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))$. The proof is by induction on the rank (with respect to P) of the ground-negative literals (see Section 2 for the definition of rank).

Basis Let $\neg\mathsf{L}$ be a ground-negative literal, and suppose that $\neg\mathsf{L}$ has rank 0 with respect to P.

- *Case 1:* There is an SLDNF refutation R for $\mathsf{P} \cup \{\leftarrow \mathsf{L}\}$ where no negative literal is selected. Then, R induces an SLD refutation $\bar{R}$ for $\bar{\mathsf{P}} \cup \{\leftarrow \mathsf{L}\}$. Therefore, $\mathsf{L} \in \mathit{Th}(\bar{\mathsf{P}})$ and, hence, $\mathsf{L} \in \bar{F}$, which implies that $\delta_{\mathsf{L}} \notin \mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))$.
- *Case 2:* There is a finitely failed SLDNF tree T for $\mathsf{P} \cup \{\leftarrow \mathsf{L}\}$ where no negative literal is selected. We show that $\mathsf{L} \notin \bar{F}$. Hence, since $\delta_{\mathsf{L}} \subseteq \bar{C}$ and $\bar{F}$ is an extension of $\bar{\mathbb{C}}(\mathsf{P}) = (\bar{C}, \bar{\mathsf{P}})$, by Lemma 2, we have that $\delta_{\mathsf{L}} \in \mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))$.

Suppose by contradiction that $\mathsf{L} \in \bar{F}$. First observe that $\bar{\mathsf{P}} \cup \mathit{Conseq}(\mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P})))$ is a set of definite clauses and recall that $\bar{F} = \mathit{Th}(\bar{\mathsf{P}} \cup \mathit{Conseq}(\mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))))$. Then, by the completeness of SLD resolution for definite clauses, there is an SLD refutation $\bar{R}$ for $\bar{\mathsf{P}} \cup \mathit{Conseq}(\mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))) \cup \{\leftarrow \mathsf{L}\}$. Construct a sequence of normal goals R by replacing each occurrence of $\bar{\mathsf{M}}$ in $\bar{R}$ with $\neg\mathsf{M}$, for each barred literal $\bar{\mathsf{M}} \in \mathit{Conseq}(\mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P})))$. Note that the last goal in R is the empty clause □. Then, by construction of T, there is a branch β of T that corresponds to a prefix of R. Let C be the goal labeling the leaf of β. Since T has failed, $\mathsf{C} \neq \square$; that is, C is not the last goal in R. Let C_1 be the leftmost literal of C. Since $\neg\mathsf{L}$ has rank 0, C_1 is not a negative literal. Moreover, again as T has failed, C_1 does not unify with the head of any clause in P. This is a contradiction, since C is not the last goal in R. Therefore, $\mathsf{L} \notin \bar{F}$, as was to be shown.

Induction step Let $\mathsf{i} > 0$. Suppose that the result holds for ground-negative literals with rank $\mathsf{j} < \mathsf{i}$. Let $\neg\mathsf{L}$ be a ground-negative literal with rank i with respect to P.

- *Case 1:* There is an SLDNF refutation R for $\mathsf{P} \cup \{\leftarrow \mathsf{L}\}$ where all negative literals selected have rank $\mathsf{j} < \mathsf{i}$. Let $\neg\mathsf{M}$ be a negative literal canceled in R. Then, there is a finitely failed SLDNF tree for $\bar{\mathsf{P}} \cup \{\leftarrow \mathsf{M}\}$. Hence, since $\neg\mathsf{M}$ has rank less than i, by the induction hypothesis, $\delta_{\mathsf{M}} \in \mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))$. Then, R induces an SLD refutation $\bar{R}$ for $\bar{\mathsf{P}} \cup \mathit{Conseq}(\mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))) \cup \{\leftarrow \mathsf{L}\}$. Therefore, $\mathsf{L} \in \bar{F}$, which implies that $\delta_{\mathsf{L}} \notin \mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))$.
- *Case 2:* There is a finitely failed SLDNF tree T for $\mathsf{P} \cup \{\leftarrow \mathsf{L}\}$ where all negative literals selected have rank $\mathsf{j} < \mathsf{i}$. We show that $\mathsf{L} \notin \bar{F}$. Hence, since $\delta_{\mathsf{L}} \subseteq \bar{C}$ and $\bar{F}$ is an extension of $\bar{\mathbb{C}}(\mathsf{P}) = (\bar{C}, \bar{\mathsf{P}})$, by Lemma 2, we have that $\delta_{\mathsf{L}} \in \mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))$.

Suppose by contradiction that $\mathsf{L} \in \bar{F}$. First observe that $\bar{\mathsf{P}} \cup \mathit{Conseq}(\mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P})))$ is a set of definite clauses, and recall that $\bar{F} = \mathit{Th}(\bar{\mathsf{P}} \cup \mathit{Conseq}(\mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))))$. Then, by the completeness of SLD resolution for definite clauses, there is an SLD refutation $\bar{R}$ for $\bar{\mathsf{P}} \cup \mathit{Conseq}(\mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P}))) \cup \{\leftarrow \mathsf{L}\}$. Construct a sequence of normal goals R by replacing each occurrence of $\bar{\mathsf{M}}$ in $\bar{R}$ with $\neg\mathsf{M}$, for each barred literal $\bar{\mathsf{M}} \in \mathit{Conseq}(\mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P})))$. Note that the last goal in R is the empty clause □. Then, by construction of T, there is a branch β of T that corresponds to a prefix of R. Let C be the goal labeling the leaf of β. Since T has failed, $\mathsf{C} \neq \square$; that is, C is not the last goal in R.

Let C_1 be the leftmost literal of C. Suppose that C_1 is positive. Then, C_1 can be canceled with the head of some clause in P, since C is not the last goal in R. But, in this case, β can be extended further, contradicting the assumption that T has finitely failed.

Suppose now that C_1 is a negative literal of the form $\neg\mathsf{D}$. Since $\bar{R}$ does not terminate in $\bar{\mathsf{C}}$, the literal $\bar{\mathsf{D}}$ was canceled with the head of some clause in $\bar{\mathsf{P}} \cup \mathit{Conseq}(\mathit{Gen}(\bar{F}, \bar{\mathbb{C}}(\mathsf{P})))$. By construction of $\bar{\mathsf{P}}$, this clause cannot be in $\bar{\mathsf{P}}$. Hence, we have that

$\bar{D} \in Conseq(Gen(\bar{F}, \bar{\mathbb{C}}(P)))$; that is, $\delta_D \in Gen(\bar{F}, \bar{\mathbb{C}}(P))$. On the other hand, since T has finitely failed and C is a leaf of T, there must be an SLDNF refutation from $P \cup \{\leftarrow D\}$. Moreover, $\neg D$ has rank $j < i$. Hence, by the induction hypothesis, $\delta_D \notin Gen(\bar{F}, \bar{\mathbb{C}}(P))$, a contradiction. Hence, $L \notin \bar{F}$, as was to be shown. □

Corollary 2 (∀-soundness of SLDNF with respect to $\bar{\mathbb{C}}$)
Let P be a normal program and ←Q be a normal goal. Suppose that $\bar{\mathbb{C}}(P)$ has at least one extension. If α is an answer to ←Q from P computed by SLDNF, then all extensions of $\bar{\mathbb{C}}(P)$ contain $\forall \bar{Q}\alpha$.

Proof
The proof follows from Theorem 10 and Theorem 12. □

Note that, to apply this theorem, we must first prove that $\bar{\mathbb{C}}(P)$ has at least one extension. This condition is necessary essentially because the construction of an SLDNF refutation is a local process, in the sense that one is required to exhibit a finitely failed SLDNF tree only for the ground-negative literals that are selected. On the other hand, the construction of extensions for $\bar{\mathbb{C}}(P)$ is a global process where all $\overline{CWA}$ defaults are candidates for firing. The next example illustrates this point.

Example 4
Suppose that the only predicate symbols are p and q, and that both are unary. Also, assume that a is the only constant. Then, the $\overline{CWA}$ defaults are $\delta_p = (:\neg p(a)/\bar{p}(a))$ and $\delta_q = (:\neg q(a)/\bar{q}(a))$.

Let $P = \{p(a) \leftarrow \neg p(a)\}$ and $Q = \leftarrow q(a)$.

Then, $\bar{\mathbb{C}}(P) = (\{\delta_p, \delta_q\}, \bar{P})$ has no extension, because δ_p and $p(a) \leftarrow \bar{p}(a)$ block the existence of any extension.

However, there is an SLDNF refutation R from $P \cup \{Q\}$, since there is a finitely failed SLDNF tree for $P \cup \{\leftarrow q(a)\}$. Note that the fact that there is no SLDNF tree for $P \cup \{\leftarrow p(a)\}$ obviously has no effect on the construction of R, since $\neg p(a)$ is never selected.

A possible solution to this problem would be to modify the definition of extension to allow the discarding of defaults when constructing extensions. This alternative is explored in [10]. We address a different alternative in the next subsection.

- *The role of stratification*

We prove in this subsection that $\bar{\mathbb{C}}$ maps stratified normal programs [11] into default theories that have exactly one extension. Therefore, the proviso of Corollary 2 may be replaced by stratification.

Let ps(L) denote the predicate symbol of the literal L. The following definitions are from [2].

Definition 9
A *level mapping* for a normal program P is a mapping λ from the set of predicate symbols of P into the non-negative integers. We refer to the *level* of a predicate symbol s as $\lambda(s)$.

Definition 10
A normal program P is *stratified* iff P has a level mapping such that, for every program clause $A \leftarrow L_1, \cdots, L_m$ in P, for $1 \le i \le m$:

1. $\lambda(ps(L_i)) \le \lambda(ps(A))$, if L_i is a positive literal.
2. $\lambda(ps(L_i)) < \lambda(ps(A))$, if L_i is a negative literal.

Definition 11
Let P be a stratified normal program.

(a) The relation $\prec$ over the predicate symbols occurring in P is defined as follows:

$s_1 \prec s_2$ iff $\lambda(s_1) < \lambda(s_2)$, for every level mapping λ for P.

(b) The *canonical level mapping* for P is the level mapping κ_P defined as

$$\kappa_P(s_2) = \#\{s_1 \mid s_1 \prec s_2 \text{ and } s_1 \text{ is a predicate symbol occurring in } P\},$$

for each predicate symbols s_2 occurring in P.

A relation among predicate symbols on stratified normal programs similar to $\prec$ can be found in [5]. Note that $s_1 \prec s_2$ iff $\kappa_P(s_1) < \kappa_P(s_2)$. Clearly, since P is stratified, the relation $\prec$ and the canonical level mapping for P are well defined. The following lemma follows directly from the definition of stratification.

Lemma 3
Let P be a stratified normal program. Let A and B be sets of barred literals. If there is a nonbarred literal L such that $L \in Th(\bar{P} \cup A) - Th(\bar{P} \cup B)$, then there exists a barred literal $\bar{M}$ such that $\bar{M} \in Th(\bar{P} \cup A) - Th(\bar{P} \cup B)$ and $ps(M) \prec ps(L)$.

Proof
Let L be a nonbarred literal such that $L \in Th(\bar{P} \cup A) - Th(\bar{P} \cup B)$. Thus, we have $\bar{P} \cup A \vdash L$ and $\bar{P} \cup B \not\vdash L$. Then there is a set $A' \subseteq A$ such that $A' \neq \emptyset$ and whose elements are effectively used in the deduction of L. As $\bar{P} \cup B \not\vdash L$, we obtain $A' - B \neq \emptyset$. Then let $\bar{M} \in A' - B$. Clearly, since $\bar{M}$ is used in the deduction of L, we have $ps(M) \prec ps(L)$. □

We are now ready to state the main theorem of this subsection.

Lemma 4
Let P be a stratified normal program. Then, $\bar{\mathbb{C}}(P)$ has exactly one extension.

Proof

We first prove that $\bar{\mathbb{C}}(\mathbf{P})$ has at most one extension. Suppose, by contradiction, that $\bar{\mathbb{C}}(\mathbf{P})$ has two extensions, E and F. Then, by Lemma 2, $E = Th(\bar{\mathbf{P}} \cup \{\bar{\mathbf{L}} \mid \delta_{\mathbf{L}} \in \bar{\boldsymbol{C}} \wedge \mathbf{L} \notin E\})$ and $F = Th(\bar{\mathbf{P}} \cup \{\bar{\mathbf{L}} \mid \delta_{\mathbf{L}} \in \bar{\boldsymbol{C}} \wedge \mathbf{L} \notin F\})$. Since $E \neq F$, we have that $(E - F) \cup (F - E) \neq \emptyset$. Let M be the set of all predicate symbols $\mathbf{s}$ of barred literals $\bar{\mathbf{L}}$ such that $\bar{\mathbf{L}} \in (E - F) \cup (F - E)$. Let $\mathbf{s}_0$ be one of the minimal elements of M with respect to the relation $<$. Let $\bar{\mathbf{L}}_0 \in (E - F) \cup (F - E)$ such that $\mathbf{ps}(\mathbf{L}_0) = \mathbf{s}_0$. Without loss of generality, suppose that $\bar{\mathbf{L}}_0 \in (E - F)$. Then, by Theorem 5(d), $\mathbf{L}_0 \in F$ and $\mathbf{L}_0 \notin E$. By the previous lemma, there is then a barred literal $\bar{\mathbf{L}}_1$ such that $\bar{\mathbf{L}}_1 \in F$ and $\bar{\mathbf{L}}_1 \notin E$, and $\mathbf{ps}(\mathbf{L}_1) < \mathbf{ps}(\mathbf{L}_0) = \mathbf{s}_0$, a contradiction, since we have assumed that $\mathbf{s}_0$ is minimal in M. Hence, since we obtain $\bar{\mathbf{L}} \in E$ iff $\bar{\mathbf{L}} \in F$, we conclude that $E = F$.

We now prove that $\bar{\mathbb{C}}(\mathbf{P})$ has at least one extension. Let

$$\mathbf{C}_0 = Th(\bar{\mathbf{P}});$$

$$\mathbf{C}_{i+1} = Th(\mathbf{C}_i \cup \{\bar{\mathbf{L}} \mid \kappa_{\mathbf{P}}(\mathbf{ps}(\mathbf{L})) = \mathbf{i} \text{ and } \mathbf{L} \notin \mathbf{C}_i\}), \text{ for each } \mathbf{i} \geq 0,$$

and

$$D_i = \{\delta_{\mathbf{L}} \mid \kappa_{\mathbf{P}}(\mathbf{ps}(\mathbf{L})) = \mathbf{i} \text{ and } \mathbf{L} \notin \mathbf{C}_i\}, \text{ for each } \mathbf{i} > 0.$$

Let $E = \bigcup \mathbf{C}_i$, $D' = \bigcup \mathbf{D}_i$, and $D'' = \{\delta_{\mathbf{L}} \mid \delta_{\mathbf{L}} \in \bar{\boldsymbol{C}} \text{ and } \mathbf{L} \notin E\}$.

By a simple induction, we obtain $E = Th(\bar{\mathbf{P}} \cup Conseq(D'))$. Thus, we shall prove that E is an extension of $\bar{\mathbb{C}}(\mathbf{P})$, by showing that $D' = D''$. Indeed, if $D' = D''$, since $E = Th(\bar{\mathbf{P}} \cup Conseq(D'))$, we have that $E = Th(\bar{\mathbf{P}} \cup Conseq(D''))$, which implies that E is an extension of $\bar{\mathbb{C}}(\mathbf{P})$, by Lemma 2(a).

We first prove that $D'' \subseteq D'$. Recall that $\delta_{\mathbf{L}} = (:\neg \mathbf{L}/\bar{\mathbf{L}})$. If $\delta_{\mathbf{L}} \notin D'$, then, taking $\mathbf{i} = \kappa_{\mathbf{P}}(\mathbf{ps}(\mathbf{L}))$, we have that $\mathbf{L} \in \mathbf{C}_i$, which implies that $\mathbf{L} \in E$ and that $\delta_{\mathbf{L}} \notin D''$. Hence, $D'' \subseteq D'$.

We now prove that $D' \subseteq D''$. Suppose by contradiction that $D' - D'' \neq \emptyset$ and let $\delta_{\mathbf{L}} \in D' - D''$. Hence, taking $\mathbf{i} = \kappa_{\mathbf{P}}(\mathbf{ps}(\mathbf{L}))$, we have that $\mathbf{L} \notin \mathbf{C}_i$ and $\mathbf{L} \in E$. Therefore, there exists $\mathbf{j} > \mathbf{i}$ such that $\mathbf{L} \in \mathbf{C}_j$, by definition of E. But we can also prove that

$$\mathbf{C}_j = Th\left(\mathbf{C}_i \cup \bigcup_{k=i}^{j-1} Conseq(D_k)\right).$$

Hence, we have that

$$\mathbf{L} \in Th\left(\mathbf{C}_i \cup \bigcup_{k=i}^{j-1} Conseq(D_k)\right) - Th(\mathbf{C}_i).$$

But, by Lemma 3, there is $\bar{\mathbf{M}}$ such that

$$\bar{\mathbf{M}} \in Th\left(\mathbf{C}_i \cup \bigcup_{k=i}^{j-1} Conseq(D_k)\right) - Th(\mathbf{C}_i)$$

and

$$\kappa_{\mathbf{P}}(\mathbf{ps}(\mathbf{M})) < \kappa_{\mathbf{P}}(\mathbf{ps}(\mathbf{L})) = \mathbf{i},$$

a contradiction, since $\kappa_{\mathbf{P}}(\mathbf{ps}(\mathbf{M})) < \mathbf{i}$ and $\bar{\mathbf{M}} \notin Th(\mathbf{C}_i)$. Then, $D' = D''$. □

We then immediately obtain Corollary 3.

Corollary 3

Let $\mathbf{P}$ be a stratified normal program and $\leftarrow \mathbf{Q}$ be a normal goal. If α is an answer to $\leftarrow \mathbf{Q}$ from $\mathbf{P}$ computed by SLDNF, then the only extension of $\bar{\mathbb{C}}(\mathbf{P})$ contains $\forall \bar{\mathbf{Q}}\alpha$.

Proof

The proof follows from Theorem 10, Theorem 12, and Lemma 4. □

This last result implies that SLDNF is $\forall$-sound with respect to $\bar{\mathbb{C}}$, in a very special sense, for the class of stratified normal programs.

5. Conclusions

We have described in this paper default logic interpretations for normal programs that provide an intuitive justification for SLDNF resolution. We prove that SLDNF resolution is strongly sound ($\forall$-sound) with respect to the interpretation $\bar{\mathbb{C}}$, provided that the resulting default theories had at least one extension. We have also shown that this proviso can be replaced by stratification, the requirement usually adopted in alternative semantics for normal programs.

References

1. M. A. Casanova, F. A. C. Giorno, and A. L. Furtado, *Programação em Lógica e a Linguagem Prolog*, Blücher Publishing, São Paulo, Brazil, 1987.
2. J. W. Lloyd, *Foundations of Logic Programming*, 2nd ed., Springer-Verlag, New York, 1987.
3. K. L. Clark, "Negation as Failure," *Logic and Databases*, H. Gallaire and J. Minker, Eds., Plenum Press, New York, 1978.
4. R. Reiter, "On Closed World Databases," *Logic and Databases*, H. Gallaire and J. Minker, Eds., Plenum Press, New York, 1978.
5. T. C. Przymusinski, "On the Declarative Semantics of Stratified Deductive Databases and Logic Programs," *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed., Morgan Kaufmann, Los Altos, CA, 1988, Ch. 5.
6. H. Przymusinski and T. C. Przymusinski, "Semantic Issues in Deductive Databases and Logic Programs," Technical Report, Department of Computer Science, University of Texas at El Paso.
7. N. Bidoit and C. Froidevaux, "General Logical Databases and Programs: Default Logic Semantics and Stratification," *Technical Report L.R.I. U.A. 410*, CNRS, Université Paris Sud, Paris.
8. R. Reiter, "A Logic for Default Reasoning," *Artif. Intell.* **13**, 81–132 (1980).
9. M. Gelfond and V. Lifschitz, "The Stable Model Semantics for Logic Programming," *Proceedings of the Fifth Logic Programming Symposium*, R. Kowalski and

K. Bowen, Eds., Association for Logic Programming, MIT Press, Cambridge, MA, 1988, pp. 1070–1080.
10. R. A. de T. Guerreiro, M. A. Casanova, and A. S. Hemerly, "Contributions to a Proof Theory for Generic Defaults," *Proceedings of the 9th European Conference on Artificial Intelligence*, Stockholm, Sweden, 1990, pp. 213–218.
11. K. R. Apt, H. A. Blair, and A. Walker, "Towards a Theory of a Declarative Knowledge," *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed., Morgan Kaufmann, Los Altos, CA, 1988, Ch. 2.

Received September 26, 1990; accepted for publication November 6, 1991

Marco Antonio Casanova *IBM Brasil, Rio Scientific Center, Av. Pres. Vargas, 824/22 andar, 20.071-001, Rio de Janeiro, RJ, Brasil (CASANOVA at RIOVMSC, casanova@vnet.ibm.com).* Dr. Casanova received a B.Sc. in electronic engineering from the Military Institute of Engineering of Brazil in 1974, an M.Sc. in computer science from the Pontifical Catholic University of Rio de Janeiro in 1976, and a Ph.D. in applied mathematics from Harvard University in 1979. In November 1982, he joined the IBM Brazil Scientific Center, where he is now a Research Manager. From 1980 to 1982, he was Assistant Professor at the Department of Informatics of the Catholic University in Rio, where he also acted as Graduate Program Coordinator during the year 1981–1982. Dr. Casanova is author of the book *The Concurrency Control Problem for Database Systems*, published by Springer-Verlag; he is a co-author of the books (in Portuguese) *Principles of Distributed Database Systems, Logic Programming and Prolog*, and *Fundamentals of Multimedia Systems.* He has also published several technical papers in international scientific journals. His academic interests include database theory, database management systems, and logic programming.

Andrea Silva Hemerly *IBM Brasil, Rio Scientific Center, Av. Pres. Vargas, 824/22 andar, 20.071-001, Rio de Janeiro, RJ, Brasil (ANDREASI at RIOVMSC, andrea@vnet.ibm.com).* Ms. Hemerly received a B.Sc. in civil engineering from the University of Espirito Santo, Brazil, and an M.Sc. in computer science from the Pontifical Catholic University of Rio de Janeiro; she is currently completing requirements for a D.Sc. at the Catholic University. In December 1988, she joined the IBM Brazil Rio Scientific Center, where she is now a Researcher. Her academic interests include database, logic programming, and artificial intelligence; she has published several technical papers in international scientific journals.

Ramiro Affonso de Tadeu Guerreiro *IBM Brasil, Industria, Maquinas e Services Ltd., Av. Pasteur 138-146, URCA, Rio de Janeiro, RJ, Brasil (RAMIROG at RIOSYS1, ramirog@riosys1.vnet.ibm.com).* Dr. Guerreiro received a B.Sc. in metallurgical engineering from the Military Institute of Engineering, Brazil, in 1981, an M.Sc. in mathematics from the Pure and Applied Mathematics Institute, IMPA, in 1982, and a D.Sc. in computer science from the Pontifical Catholic University of Rio de Janeiro in 1990. In August 1983 he joined IBM Brazil, where he is now a Marketing Specialist for Customer Information Systems–Latin America. From 1987 to 1991, Dr. Guerreiro was a Researcher at the IBM Rio Scientific Center. His academic interests include logic, artificial intelligence, and database theory; he has published several technical papers in international scientific journals.