

Explaining SLDNF Resolution with Non-Normal Defaults (Extended Abstract)

Marco A. Casanova, Andrea S. Hemerly, Ramiro A. de T. Guerreiro

Rio Scientific Center - IBM Brazil
P.O. Box 4624
20.001, Rio de Janeiro, RJ - Brazil

Abstract

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 a extension of SLD-resolution. Second, it is semantically convincing, but it requires neither an elaborated non-standard interpretation nor a radical rewriting of the program clauses that would make it difficult to understand their meaning. Lastly, 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-resolution [3,8], that accepts only definite programs, whose clauses do not admit negative literals in their body. When one relaxes this restriction and works with the so-called normal programs, the refutation method usually adopted becomes SLDNF-resolution, that extends SLD-resolution with the negation by finite failure rule (NFF) [5]. Roughly, the NFF rule says to cancel a negative literal $\neg L$ from the body of a clause if one fails finitely to answer YES to the query $\leftarrow L$ in the presence of the program clauses.

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. We then introduce another class of defaults, called barred CWA defaults, that offer the following advantages. First, they directly capture the true nature of SLDNF as a extension of SLD-resolution. Second, they are semantically convincing, as proved at the end of section 3, but they require neither an elaborated non-standard semantics nor a radical rewriting of the program clauses that makes it difficult to understand their meaning. Finally, the semantics induced by barred CWA defaults coincides with the other results for stratified programs, but it "captures more" since the basic result 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 [9,10]. The closest approach to ours is that taken in [2], 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 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.

Finally, the reader is referred to [4] for the complete proofs of the results, which were omitted here due to space limitations.

2. Preliminaries

2.1 Defaults

We review in this section some basic concepts of default logic. A detailed development can be found in [12]. A *default* is an expression of the form $(\alpha:\beta_1,\dots,\beta_m/\omega)$ where α , β_1,\dots,β_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 β_1,\dots,β_m are called the *justifications*. A default is *closed* iff $\alpha,\beta_1,\dots,\beta_m$ 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 import 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 [12].

Theorem 1: Let E be a set of sentences and let $\Delta=(D,W)$ be a default theory. Define

$$\begin{aligned} E_0 &= W \quad \text{and, for } i \geq 0: \\ E_{i+1} &= Th(E_i) \cup \\ &\quad \left\{ \omega \mid \frac{\alpha:\beta_1,\dots,\beta_m}{\omega} \in D, \text{ where } \alpha \in E_i \text{ and } \neg\beta_1,\dots,\neg\beta_m \notin E_i \right\}. \end{aligned}$$

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:

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

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 for 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 it is its only extension.

Theorem 2: (*Minimality of Extensions*). Let $\Delta = (D, W)$ be a default theory. If E and F are extensions for Δ and if $E \subseteq F$, then $E = F$.

2.2 SLD- and SLDNF-Resolution

We first recall some concepts directly related to SLD. An expression of the form $A \leftarrow B_1, \dots, B_n$ is a *definite clause* iff A, B_1, \dots, B_n are positive literals. An expression of the form $\leftarrow B_1, \dots, B_n$ is a *goal* iff B_1, \dots, B_n are positive literals. The literals B_1, \dots, 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 $\leftarrow B_1, \dots, B_n$ represents the negation of $B_1 \wedge \dots \wedge B_n$. We refer the reader to [8] 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 will denote the universal (or existential) closure of a formula F by $\forall F$ (or $\exists F$).

Let P be a program and G be a goal of the form $\leftarrow B_1, \dots, B_n$. 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 $\forall (B_1 \wedge \dots \wedge B_n)\alpha$ 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 a SLD-refutation R from $PU\{G\}$ 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 3: (*Soundness and Completeness of SLD with respect to answers*) [8]

- (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, it 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 [8] for the definitions of *SLDNF-refutation* and *SLDNF-tree*. We just recall here that the process of constructing a SLDNF-refutation or a SLDNF-tree from $PU\{G\}$ will succeed only if, for every negative literal $\neg p(t)$ that was selected:

- $\neg p(t)$ is ground; and
- there is a finitely failed SLDNF-tree for $PU\{\leftarrow p(t)\}$, in which case $\neg p(t)$ was cancelled, or there is a SLDNF-refutation from $PU\{\leftarrow p(t)\}$, in which case $\neg p(t)$ was not cancelled.

The notions of answer and computed answer extend to the context of SLDNF directly. 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 $A \leftarrow B_1, \dots, B_n$ with the formula $\forall (B_1 \wedge \dots \wedge B_n \Rightarrow A)$, and similarly for goals. Extending Theorem 3 above to SLDNF is in fact the major theme of this paper.

We emphasize that SLDNF is an inherently recursive process in the sense that, to construct a SLDNF-refutation or a SLDNF-tree, one may have to build other SLDNF-refutations and trees. But the process does not involve self-loops, that is, if a 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 $\neg p(t)$:

- $\neg p(t)$ has *rank* 0 with respect to P iff there is either a finitely failed SLDNF-tree or a SLDNF-refutation from $PU\{\leftarrow p(t)\}$ where no negative literal is selected;
- $\neg p(t)$ has *rank* $k+1$ with respect to P iff there is either a finitely failed SLDNF-tree or a SLDNF-refutation from $PU\{\leftarrow p(t)\}$ 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 cancelling of a negative literal proceeds in stages according to the rank of the selected negative literals.

3. Three Default Logic Interpretations for Normal Programs and Goals

3.1 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 $A_1 \wedge \dots \wedge A_n$, let $\exists A$ indicate the existential closure of A and $\leftarrow A$ indicate the goal $\leftarrow A_1, \dots, A_n$.

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.

3.2 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: (The Default Logic Interpretation \mathbb{C})

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

However, we can easily show that SLDNF is not \forall -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 1:

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 non-logical symbols of the underlying first-order lan-

guage are the constant a and the two unary predicate symbols p and q .

Then, there is a SLDNF-refutation R from $PU\{G\}$ 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 first-order sentence $q(a) \vee p(a)$. Then, $C(P) = (C, P)$ 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 \forall -sound with respect to C since $\neg p(a)$ does not belong to both extensions.

3.3 An Interpretation based on Barred CWA Defaults

We introduce in this section a default logic interpretation for normal programs for which SLDNF is \forall -sound. The general idea is to treat a predicate symbol preceded by " \neg " 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 *CWA-default* $(:\neg p(t) / \bar{p}(t))$, which intuitively says to believe in $\bar{p}(t)$ if it is consistent to assume $\neg p(t)$. Finally, we will define the interpretation \bar{C} that maps a normal program P into the default theory $\bar{C}(P) = (\bar{C}, \bar{P})$, where \bar{C} is the set of all *CWA-defaults* over the underlying alphabet and \bar{P} is the barred transform of P , and that maps each conjunction of literals into its barred transform.

All definitions that follow are relative to a fixed first-order alphabet A .

Definition 5:

- (a) The *barred augmented alphabet* corresponding to A , denoted by \bar{A} , is obtained by adding to A the new symbol \bar{p} as an n -ary predicate symbol, for each n -ary predicate symbol p in A .
- (b) A positive literal M over \bar{A} is a *barred literal* iff M is of the form $\bar{p}(t)$. A positive literal M over \bar{A} is a *positive non-barred literal* iff M is of the form $p(t)$.
- (c) A positive literal M over \bar{A} is the *barred complement* of a positive literal L over A iff M is of the form $\bar{p}(t)$ and L is of the form $p(t)$. The barred complement of L will be denoted by \bar{L} .
- (d) The *barred transform* of a negative literal over A of the form $\neg p(t)$ is the literal $\bar{p}(t)$ over \bar{A} .
- (e) Let P be a program clause (resp., 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 (resp., the goal, the conjunction of literals or the set of definite clauses) obtained by replacing in P each occurrence of a negative literal by its barred transform.

- (f) A \overline{CWA} -default is a closed, non-normal default of the form $(:\neg L/\overline{L})$, denoted by δ_L , where L is a ground positive non-barred literal and \overline{L} is the barred complement of L . We also say that $\neg L$ is the *justification* and that \overline{L} is the *consequent* of $(:\neg L/\overline{L})$. We will denote the set of all \overline{CWA} -defaults by \overline{C} .

Definition 6: (*The Default Logic Interpretation \overline{C}*)

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

The following example, which should be compared with Example 1, illustrates the interpretation \overline{C} .

Example 2:

Let \underline{P} and \underline{G} be as in Example 1. Then, \overline{P} is the (definite clause) program and \overline{G} is the goal defined by the clauses below:

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

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

Note that, since $\overline{C}(\neg p(a)) = \overline{p}(a)$, since the universal closure of $\overline{p}(a)\varepsilon$ is equivalent to

$\overline{p}(a)$ and since E contains $\overline{p}(a)$, E also contains the universal closure of $\overline{C}(\neg p(a))\varepsilon$. Therefore, \underline{P} and \underline{G} are not a counter-example for the \forall -soundness of SLDNF with respect to \overline{C} , whereas they are for the \forall -soundness of SLDNF with respect to \underline{C} .

Furthermore, observe that, when compared with Example 1, the only change lay in that clause 1 in \underline{P} , after the transformation induced by \overline{C} , became first-order equivalent to the sentence $q(a) \vee \neg \overline{p}(a)$ whereas, after the transformation induced by \underline{C} , it was first-order equivalent to the sentence $q(a) \vee p(a)$.

We now state two properties of \overline{C} that we will need to prove the main results in section 4. They follow directly from theorems stated in section 2.1.

Lemma 2:

Let P be a normal program.

- (a) A set E of sentences over \overline{A} is an extension for $\overline{C}(P)$ iff $E = Th(\overline{P} \cup T)$, where $T = \{\overline{L}/(:\neg L/\overline{L}) \in \overline{C} \wedge L \notin E\}$.

(b) $\overline{C}(P)$ has no inconsistent extension.

The next list of results provides a semantic justification for the transformation \overline{C} . Given a set of sentences or a set of clauses S , we will denote by B_S the Herbrand base for S . Also, given a set of definite clauses D , we will denote by $M(D)$ the unique minimal Herbrand model of D and by T_D the mapping that takes each Herbrand interpretation I of D into the Herbrand interpretation

$$T_D(I) = \{A \in B_D \mid A \leftarrow A_1, \dots, A_n \text{ is a ground instance of a clause in } D \text{ and } \{A_1, \dots, A_n\} \subseteq I\}$$

Theorem 4:

Let P be a normal program and E be an extension of $\overline{C}(P)$.

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

- (a) E has a unique minimal Herbrand model $M(E)$.
- (b) $N \in E$ iff $N \in M(E)$, 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 $\overline{L} \notin E$, for any ground positive non-barred literal L .
- (e) $M(E)$ satisfies $\neg L$ iff $\overline{L} \in M(E)$, for any ground positive non-barred literal L .
- (f) $M(E)$ is a model of P .

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 [2], 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 [2]. Now, item (e) indicates that a ground barred literal \overline{L} indeed behaves as the negation of L with respect to the unique minimal Herbrand model of each extension of $\overline{C}(P)$. Finally, item (f) shows that the default logic interpretation \overline{C} ultimately associates with each normal program P a set of Herbrand interpretation which are the minimal models of the extensions of $\overline{C}(P)$.

We can also show that the set of unique minimal Herbrand models of the extensions of $\overline{C}(P)$ corresponds exactly to the set of stable models of P [7]. We first recall the definition of stable models. The *GL-transformation* is the mapping γ that takes a normal program P and a Herbrand interpretation I for P into a new program $\gamma(P, I)$, obtained from P by performing the following two reductions:

- removing from P all clauses whose body contain a negative literal $\neg A$ such that $A \in I$;
- removing from the body of all the remaining clauses those negative literals $\neg A$ such that $A \notin I$.

Note that the new program $\gamma(P, I)$ contains only definite clauses and, hence, it has a unique minimal Herbrand model $M(\gamma(P, I))$. We then define the *Gelfond-*

Lifschitz operator Γ_P for P as $\Gamma_P(I) = M(\gamma(P, I))$. It can be proved that the fixed-points of Γ_P are minimal models of P [7]. Finally, we say that a Herbrand interpretation I for P is a *stable model* of P iff $\Gamma_P(I) = I$.

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

$$\alpha(I) = I \cup \{\bar{L} \mid L \text{ is a non-barred ground atom and } L \notin I\}$$

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

$$\beta(I) = \{L \mid L \text{ is a non-barred ground atom and } L \in I\}$$

We are now ready to prove the following result.

Theorem 5:

Let P be a normal program. Then:

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

3.4 An Interpretation based on NFF Defaults

Given a normal program P , we now introduce a third default logic interpretation, \bar{N} , that is much closer to SLDNF than the default logic interpretation \bar{C} , since the former directly encodes the way negative literals are cancelled in SLDNF. We will then prove in section 4.1 that SLDNF is \forall -sound and \exists -complete with respect to \bar{N} .

Definition 7:

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

Definition 8: (The Default Logic Interpretation \bar{N})

\bar{N} is the interpretation that maps each normal program P into the default theory $\bar{N}(P) = (\bar{N}_P, \bar{P})$, where \bar{N}_P is the set of all \bar{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 shall prove in this section results whose major implications are (P is a normal program):

- (a) $\bar{C}(P)$ may have no extensions, whereas $\bar{N}(P)$ 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 $PU\{\leftarrow L\}$, then δ_L can always be fired in $\overline{C}(P)$, that is, it belongs to the set of generating defaults of all extensions of $\overline{C}(P)$, if there is one;
- (c) moreover, given a ground negative literal $\neg L$, if there is a SLDNF-refutation for $PU\{\leftarrow L\}$, then δ_L can never be fired in $\overline{C}(P)$, that is, it does not belong to the set of generating defaults of any extension of $\overline{C}(P)$, if there is one;
- (d) as a consequence, if δ_L belongs to the set of generating defaults of some extension of $\overline{C}(P)$, but not all, then there is neither a finitely failed SLDNF-tree for $PU\{\leftarrow L\}$ nor a SLDNF-refutation for $PU\{\leftarrow L\}$.
- (e) however, given a ground negative literal $\neg L$, there may be neither a finitely failed SLDNF-tree nor a SLDNF-refutation for $PU\{\leftarrow L\}$ and yet δ_L may belong to the set of generating defaults of all extensions of $\overline{C}(P)$, if there is one.

4.1 \exists -Soundness with Respect to CWA Defaults

We shall show in this section that SLDNF is \exists -sound with respect to C . The results follows from a property of SLDNF posed as a challenge to the reader in [8, ex. 32, Chap.3].

Theorem 6:

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

Theorem 7:

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 $C(P)$ that contains $\forall Q\alpha$.

Proof

Let R be a SLDNF-refutation from $PU\{\leftarrow Q\}$ that computes α . Let $D = \{\neg L_1, \dots, \neg L_n\}$ be the set of negative literals selected in R . We shall show that PUD is consistent.

Indeed, for each $i \in [1, n]$, the SLDNF-tree for $PU\{\leftarrow L_i\}$ is finitely failed. Hence, the SLDNF-tree for $PU\{\leftarrow L_1, \dots, L_n\}$ is finitely failed. Therefore, by the previous theorem, $PU\{\leftarrow L_1, \dots, L_n\}$ is consistent, that is, PUD is consistent.

Now, if PUD is consistent, then the normal default theory (D, P) has an extension, call it E . Moreover, R is a refutation (using linear resolution) from $PUDU\{\leftarrow Q\}$ that still computes α . Hence, $PUD = \forall Q\alpha$, by [3]. Since $PUD \subseteq E$, we then have $\forall Q\alpha \in E$. Now, since $D \subseteq C$, by the semi-monotonicity of normal default theories, there is an extension of $\overline{C}(P) = (C, P)$ that contains $\forall Q\alpha$.

□

4.2 \forall -Soundness and \exists -Completeness with Respect to NFF Defaults

Theorem 8:

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

Proof

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

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

Theorem 9: (\forall -Soundness of SLDNF with respect to \overline{N})

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

Proof

Suppose that α is an answer to $\leftarrow Q$ from P computed by SLDNF. Then, there is a SLDNF-refutation R for $PU\{\leftarrow Q\}$ such that α is the composition of the substitutions in R , restricted to the variables in Q . Let \overline{R} be the sequence of goals obtained by replacing each occurrence of $\neg M$ in R by \overline{M} , for each negated literal $\neg M$. Since R is a SLDNF-refutation, if $\neg M$ was cancelled in R , then $\neg M$ is ground and there is a finitely failed SLDNF-tree for $P \cup \{\leftarrow M\}$. Hence, $\delta_M \in \overline{N}_P$. Therefore, \overline{R} is a SLD-refutation from $\overline{P} \cup Conseq(\overline{N}_P) \cup \{\leftarrow Q\}$. Moreover, R and \overline{R} perform the same substitutions. Thus, α is also the composition of the substitutions in \overline{R} , restricted to the variables in Q . Hence, we may conclude that $\overline{P} \cup Conseq(\overline{N}_P) \vdash \forall \overline{Q}\alpha$, by Theorem 7.1 of [8] (Soundness of SLD-resolution). Since $\overline{E} = Th(\overline{P} \cup Conseq(\overline{N}_P))$, by Theorem 8, we finally obtain that $\forall \overline{Q}\alpha$ belongs to \overline{E} . \square

Theorem 10: (\exists -Completeness of SLDNF with respect to \overline{N})

Let P be a normal program and $\leftarrow Q$ be a normal goal. Let \overline{E} be the unique extension of $\overline{N}(P)$. Let α be an answer to $\leftarrow Q$ from P . If $\forall \overline{Q}\alpha$ belongs to \overline{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 [8] (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 shall show that β is an answer to $\leftarrow Q$ from P computed by SLDNF.

Let \bar{R} be a SLD-refutation for $\bar{P} \cup Conseq(\bar{N}_P) \cup \{\leftarrow \bar{Q}\}$ that computes β . As in the proof of Theorem 9, there is then a SLDNF-refutation R for $P \cup \{\leftarrow Q\}$ that computes β . Therefore, β is an answer to $\leftarrow Q$ from P computed by SLDNF such that β is more general than α . □

4.3 \forall -Soundness with Respect to Barred CWA Defaults

The results in section 4.2 are somewhat unsatisfactory because the definition of \overline{NFF} -defaults refers directly to SLDNF. The results in this section then builds upon them to prove that SLDNF is \forall -sound with respect to \bar{C} (with one proviso). This is a much more satisfactory situation because \bar{C} provides a superior default logic interpretation for normal programs and goals since CWA -defaults are defined independently from SLDNF.

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

Theorem 11:

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

Corollary 2: (\forall -Soundness of SLDNF with respect to \bar{C})

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

Proof: Follows from Theorem 9 and Theorem 11. □

Note that, to apply this theorem, we must first prove that $\bar{C}(P)$ has at least one extension. This condition is necessary essentially because the construction of a 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 $\bar{C}(P)$ is a global process where all CWA defaults are candidates for firing.

4.4 The Role of Stratification

We prove in this section that \bar{C} maps stratified normal programs [1] 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 [8].

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 $A \leftarrow L_1, \dots, L_m$ in P , for $1 \leq i \leq m$:

- 1) $\lambda(\text{ps}(L_i)) \leq \lambda(\text{ps}(A))$, if L_i is a positive literal;
- 2) $\lambda(\text{ps}(L_i)) < \lambda(\text{ps}(A))$, if L_i is a negative literal.

We also introduce some auxiliary concepts below. A similar relation among predicate symbols on stratified normal programs is found in [9].

Definition 11:

Let P be a stratified normal program.

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

$s_1 < s_2$ 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:

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

Note that $s_1 < s_2$ iff $\kappa_P(s_1) < \kappa_P(s_2)$. Clearly, since P is stratified, the relation $<$ and the canonical level mapping for P is 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 non-barred literal L such that $L \in \text{Th}(\overline{\neg} \cup A) - \text{Th}(\overline{P} \cup B)$, then there exists a barred literal \overline{M} such that $\overline{M} \in \text{Th}(\overline{P} \cup A) - \text{Th}(\overline{P} \cup B)$ and $\text{ps}(M) < \text{ps}(L)$.

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

Lemma 4:

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

We then immediately obtain:

Corollary 3:

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

Proof: Follows from Theorem 9, Theorem 11 and Lemma 4. □

Therefore, as promised, we proved that SLDNF is \forall -sound with respect to $\overline{\mathbb{C}}$, for the class of stratified normal programs.

5. Conclusions

We described in this paper default logic interpretations for normal programs that provide an intuitive justification for SLDNF-resolution. We proved that SLDNF-resolution is strongly sound (\forall -sound) with respect to the interpretation $\overline{\mathbb{C}}$, 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] Apt, R.K., Blair, H. and Walker, A., "Towards a Theory of Declarative Knowledge", Workshop on Foundations of Deductive Databases and Logic Programming, (1986), 546-628.
- [2] Bidoit, N. and Froidevaux, C., "General Logical databases and Programs: Default Logic Semantics and Stratification", Technical Report, L.R.I. U.A. 410 du CNRS, Université Paris Sud.
- [3] Casanova, M.A., Giorno, F.A.C. and Furtado, A.L., *Programação em Lógica e a Linguagem Prolog*, Ed. Blucher.
- [4] Casanova, M.A., Hemerly, A.S. and Guerreiro, R.A.T., "Explaining SLDNF Resolution with Non-Normal Defaults", Technical Report CCR115, Rio Scientific Center, IBM Brazil (Dec. 1990).
- [5] Clark, K.L., "Negation as Failure", in *Logic and Databases*, H. Gallaire and J. Minker (eds.), Plenum Press.
- [6] Guerreiro, R.A.T., Casanova, M.A. and Hemerly, A.S., "Contributions to a Proof Theory for Generic Defaults", Proc. 9th European Conf. on Artificial Intelligence, Stockholm, Sweden (Aug. 1990).
- [7] Gelfond, M. and Lifschitz, V., "The stable model semantics for logic programming", in *Proceedings of the Fifth Logic Programming Symposium*, R. Kowalski and K. Bowen (eds), Association for Logic Programming, MIT Press (1988), 1070-1080.
- [8] Lloyd, J.W., *Foundations of Logic Programming*, Springer-Verlag, 2nd ed. (1987).
- [9] Przymusiński, T.C., "On the declarative semantics of stratified deductive databases and logic programming", in *Foundations of Deductive Databases and Logic Programming*, J. Minker (ed.), Morgan Kaufmann (1988), 193-216.
- [10] Przymusiński, H. and Przymusiński, T.C., "Semantic Issues in Deductive Databases and Logic Programs", Technical Report, Department of Computer Science, University of Texas at El Paso.
- [11] Reiter, R., "On Closed World Databases", in *Logic and Databases*, H. Gallaire and J. Minker (eds.), Plenum Press.
- [12] Reiter, R., "A logic for default reasoning", *Artificial Intelligence* 13 (1980), 81-132.