

Explaining SLDNF Resolution with Non-Normal Defaults

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

Centro Científico Rio - IBM Brasil
Caixa Postal 4624 - 20.001, Rio de Janeiro, RJ

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 semantics nor a radical rewriting of the program clauses that makes it difficult to understand their meaning. Lastly, it extends known results for stratified normal programs to programs that satisfy a condition weaker.

KEYWORDS - negation as finite failure, Default Logic, Closed World Assumption

1. INTRODUCTION

The basis for logic programming systems is a refutation method, called SLD-resolution [3,7], 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) [4]. 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.

We may point out at least three characteristics of the NFF rule: it is easy to implement in Prolog systems; it is a nonmonotonic rule justified on the grounds of the so-called "Closed World Assumption" (CWA) ([10]); it is very difficult to define a semantics for normal programs for which SLDNF-resolution is sound and complete.

This paper addresses the third point by defining a semantics for logic programs by interpreting them as default theories. This semantics offers the following advantages. 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 semantics nor a radical rewriting of the program clauses that makes it difficult to understand their meaning. Lastly, it extends known results for stratified normal programs [9] to programs that satisfy a condition weaker.

Alternative semantics for normal programs that account for SLDNF-resolution have been extensively investigated. Comprehensive surveys can be found in [8,9]. 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 2 reviews the basic concept of default logic and SLDNF-resolution that we need in the paper. Section 3 describes the default logic interpretation we use to explain SLDNF-resolution. Section 4 states the basic results, with the proofs omitted for brevity. Finally, section 5 contains the conclusions.

2. PRELIMINARIES

2.1 Defaults

We review in this section some basic concepts of default logic. A detailed development can be found in [Re.]. A *default* is an expression of the form $(\alpha:\beta_1,\dots,\beta_m/\omega)$ where $\alpha, \beta_1,\dots,\beta_m$ and ω are all first-order formulas. The formulas α and ω are called, respectively, the *pre-requisite* 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 [Re.].

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

$$E_0 = W \text{ and, for } i \geq 0:$$

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

$$\text{Then, } E \text{ is an extension for } \Delta \text{ iff } E = \bigcup_{i=0}^{\infty} E_i.$$

2.2 SLD- and SLDNF-Resolution

In this section we briefly review SLD- and SLDNF-resolution (SLD and SLDNF for short, respectively). The reader familiar with [7,Chap.3] may skip directly to section 3.

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 [7] 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 2: (*Soundness and Completeness of SLD with respect to answers* [7])

- (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 [7] 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 2 above to SLDNF is in fact the major theme of this paper.

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

We begin by emphasizing 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 $\neg p(t)$:

- (a) $\neg p(t)$ 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;
- (b) $\neg p(t)$ 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.

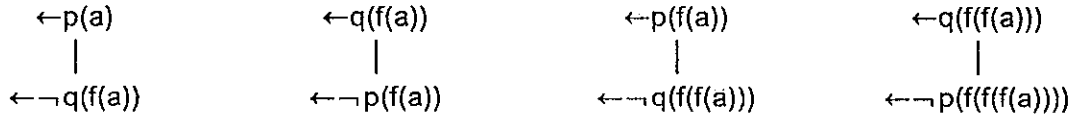
The search for a SLDNF-refutation from a program and a goal may fail for many reasons, such as when an open negative literal is selected at any depth of the recursion, or when the process of cancelling a negative literal diverges either because an infinite branch of a tree is reached or because an infinite 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 $PU\{G\}$ essentially because, if one tries to build a SLDNF-tree from $PU\{\leftarrow p(a)\}$ one will have to build a tree from $PU\{\leftarrow q(f(a))\}$, and from $PU\{\leftarrow p(f(a))\}$, and from $PU\{\leftarrow q(f(f(a)))\}$, and so on:



3. THREE DEFAULT LOGIC INTERPRETATIONS FOR NORMAL PROGRAMS AND GOALS

In section 2.2 we stressed that SLDNF has the flavor of a procedural method that may invoke itself recursively. As a consequence, there is little hope to define a 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 to obtain a semantics which explains (the soundness of) SLDNF-resolution as tightly as possible, but which does 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.

We provide semantics for normal programs and goals indirectly by interpreting them into Default Logic. 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 default logic interpretation for normal programs and goals for which SLDNF is \forall -sound and, hopefully, \exists -complete.

A natural strategy would be to define an interpretation that maps each normal program into the closed normal default theory whose defaults are the CWA defaults and whose set of sentences is the program itself. 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 $\mathbb{C}(P) = (C, P)$, where C is the set of all CWA-defaults over the underlying alphabet, and that maps each conjunction of literals into itself.

We shall show in section 4 that SLDNF is \exists -sound with respect to \mathbb{C} . However, 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 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 non-logical symbols of the underlying first-order language are the constant a and the two unary predicate symbols p and q .

Then, there is a 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 first-order sentence $q(a) \vee p(a)$. Then, $\mathbb{C}(P) = (C, P)$ 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 \mathbb{C} since $\neg p(a)$ does not belong to both extensions.

In view of this example, we introduce another 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 $\bar{\mathbb{C}}$ that maps a normal program P into the default theory $\bar{\mathbb{C}}(P) = (\bar{C}, \bar{P})$, 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.

We will give precise definitions for these concepts in this section and show that barred literals in some sense behave as negated literals in the context of \overline{CWA} -defaults. We shall then show in section 4 that SLDNF is \forall -sound with respect to \overline{C} . 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 \overline{A} , is obtained by adding to A the new symbol \overline{p} as an n -ary predicate symbol, for each n -ary predicate symbol p in A .
- (b) A positive literal M over \overline{A} is a *barred literal* iff M is of the form $\overline{p}(t)$. A positive literal M over \overline{A} is a *positive non-barred literal* iff M is of the form $p(t)$.
- (c) A positive literal M over \overline{A} is the *barred complement* of a positive literal L over A iff M is of the form $\overline{p}(t)$ and L is of the form $p(t)$. The barred complement of L will be denoted by \overline{L} .
- (d) The *barred transform* of a negative literal over A of the form $\neg p(t)$ is the literal $\overline{p}(t)$ over \overline{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 \overline{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 $(:\neg L/\overline{L})$, denote 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 2, illustrates the interpretation \overline{C} .

Example 3:

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

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

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, $\overline{C}(P) = (\overline{C}, \overline{P})$ 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, \overline{P} and \overline{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 C .

Furthermore, observe that, when compared with Example 2, the only change lay in that clause 1 in \overline{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 C , it was first-order equivalent to the sentence $q(a) \vee p(a)$.

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 [7] that takes each Herbrand interpretation I of D into the Herbrand interpretation

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

Theorem 3:

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

Let $T = \{\overline{L} / (\neg L / \overline{L}) \in \overline{C} \wedge L \notin E\}$ 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 .

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 interpretations which are the minimal models of the extensions of $\overline{C}(P)$.

It can also be shown 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 [6].

Finally, given a normal program P , we introduce a third default logic interpretation, \overline{N} , that is much closer to SLDNF than the default logic interpretation \overline{C} , since the former directly encodes the way negative literals are cancelled in SLDNF. We will then prove in section 4 that SLDNF is \forall -sound and \exists -complete with respect to \overline{N} .

Definition 7:

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

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

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

4. RESULTS

We first show in this section that SLDNF is \exists -sound with respect to \mathbb{C} . This result follows from a property of SLDNF posed as a challenge to the reader in [7, ex. 32, Chap.3] and established in the following lemma:

Lemma 1:

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.

Using this lemma we can then prove that SLDNF is \exists -sound with respect to \mathbb{C} .

Theorem 4:

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 $\forall Q\alpha$.

We now proceed to prove the \forall -soundness and \exists -completeness of SLDNF with respect to NFF defaults. The intuition behind the first result towards this end goes as follows. Let P be a normal program and G be a normal goal. On a first approximation, one may say that, if a ground negative literal $\neg p(t)$ is cancelled in a SLDNF-refutation from $PU\{G\}$, then it is consistent with P . This assertion is in principle warranted by the existence of a finitely failed SLDNF-tree T from $PU\{\neg p(t)\}$. However, a close scrutiny reveals that it is false since the construction of T may require cancelling 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 cancelled, then it is consistent with P and with all other ground negative literals that can be cancelled. This recursive statement is not paradoxical and it can be formulated in the context of the default logic interpretation \overline{N} to mean that all defaults belonging to the set \overline{N}_P can be simultaneously fired. This implies that $\overline{N}(P)$ has a unique extension.

Theorem 5:

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

It then follows from this result that SLDNF is \forall -sound and \exists -complete with respect to \overline{N} .

Theorem 6: (*\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 $\forall Q\alpha$ belongs to \overline{E} .

Theorem 7: (*\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 $\forall Q\alpha$ 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 .

Theorem 6 and Theorem 7 above are somewhat unsatisfactory because the definition of \overline{NFF} -defaults refers directly to SLDNF. The next set of results then builds upon them to prove that SLDNF is \forall -sound with respect to \overline{C} (with one proviso). This is a much more satisfactory situation because \overline{C} provides a superior default logic interpretation for normal programs and goals since \overline{CWA} -defaults are defined independently from SLDNF.

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

Theorem 8:

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

From Theorem 6 and Theorem 8 we can then prove the following result:

Corollary 1: (\forall -Soundness of SLDNF with respect to \overline{C})

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

Note that, to apply this theorem, we must first prove that $\overline{C}(P)$ 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 $\overline{C}(P)$ 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 $\delta_p = (\neg p(a)/\overline{p}(a))$ and $\delta_q = (\neg q(a)/\overline{q}(a))$.

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

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

However, there is a SLDNF-refutation R from $PU\{Q\}$ since there is a finitely failed SLDNF-tree for $PU\{\leftarrow q(a)\}$. Note that the fact that there is no SLDNF-tree for $PU\{\leftarrow p(a)\}$ has obviously 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 [5]. We address a different alternative by proving that \overline{C} maps stratified normal programs [1] into default theories that have exactly one extension. Therefore, the proviso of Corollary 1 may be replaced by stratification.

Let $ps(L)$ denote the predicate symbol of the literal L . The following definitions are from [7].

Definition 9:

- (a) 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 $\lambda(s)$.

- (b) 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.

Lemma 2:

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

We then immediately obtain from Theorem 6, Theorem 8 and Lemma 2:

Corollary 2:

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 $\forall \overline{Q}\alpha$.

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{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] Clark, K.L., "Negation as Failure", in *Logic and Databases*, H. Gallaire and J. Minker (eds.), Plenum Press.
- [5] 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).
- [6] 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.
- [7] Lloyd, J.W., *Foundations of Logic Programming*, Springer-Verlag, 2nd (1987).
- [8] 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.
- [9] 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.
- [10] Reiter, R., "On Closed World Databases", in *Logic and Databases*, H. Gallaire and J. Minker (eds.), Plenum Press.
- [11] Reiter, R., "A logic for default reasoning", *Artificial Intelligence* 13 (1980), 81-132.