A direct Proof of the Completeness of SLDNF-resolution

ROBERT STÄRK, *Institute of Informatics, University of Fribourg, Rue Faucigny 2, CH–1700 Fribourg, Switzerland.*
E-mail: robert.staerk@unifr.ch

Abstract

We give a direct proof of the following theorem: if a goal $G\sigma$ is a logical consequence of the partial completion of an arbitrary normal logic program P, then each fair, non-floundering SLDNF-tree T for G yields an answer substitution θ which is more general than σ. If the negation $\overline{G}$ is a logical consequence of the partial completion of P, then T is finitely failed. A tree is fair, if each negative main branch ends in failure or each literal in the branch is selected at a certain point. A tree is floundering if it contains a positive node that consists of negative, non-ground literals only.

Keywords: Loigc programming, negation as failure, SLDNF-resolution.

1 Introduction

Most completeness proofs for SLDNF-resolution are of the following kind: if a goal G and a logic program P have the property that each SLDNF-tree for G with respect to P is non-floundering, and if the goal $G\sigma$ is a (two- or three-valued) logical consequence of (the completion of) P, then there exists an SLDNF-tree for G with respect to P which contains a successful branch with an answer substitution that is more general than σ. That all SLDNF-trees for G and P are non-floundering is usually a consequence of syntactic conditions. For example, if G and P are correct with respect to a mode assignment in the sense of [11], then each SLDNF-tree for G and P is non-floundering.

In this paper a direct proof of a stronger theorem is presented. Given a fair, non-floundering SLDNF-tree of G with respect to the normal logic program P we construct a term model of the partial completion of P such that, if $G\sigma$ it is true in the model, then the tree contains a successful branch with an answer substitution that is more general than σ and, if $\overline{G}$ is true in the model, then the tree is finitely failed. Hence we do not require that all SLDNF-trees for G and P are non-floundering, but just the given, fair SLDNF-tree must have the property.

Similar constructions of term models have been used in the proof of the completeness of the negation-as-failure rule for *definite* programs by Wolfram, Maher and Lassez in [14] and in the proof of the completeness of SLDNF-resolution for *stratified, allowed* programs by Cavedon and Lloyd in [5].

What is the difference to Drabent's result in [8]? First, our proof is direct and elementary and does not make a detour via SLDFA-resolution as Drabent's proof does. Second, we consider an extension of SLDNF-resolution. The extension can be characterized by the following two rules:

- $\overline{A}$ succeeds, if A is ground and A fails.

- $\overline{A}$ fails, if A succeeds with the identity substitution.

Drabent requires that the atom A must be ground in the second rule, too.

Buchholz proves in [4] a similar theorem under the stronger assumption that the goal G satisfies an additional condition which ensures that all SLDNF-trees for G are non-floundering. The notion of SLDNF-tree we use in this paper differs from that of Buchholz. We distinguish between positive and negative nodes in SLDNF-trees. A tree is fair if every negative branch is fair, i.e. ends in failure or each literal is selected at a certain point. A tree is floundering if it contains a positive node that consists of negative, non-ground literals only.

2 Basic notions

We assume that for each positive relation symbol R there exists a relation symbol $\overline{R}$ of the same arity. Atomic formulas $R(\vec{t})$ are called *positive literals* or sometimes just *atoms*. They are denoted by A, B. Atomic formulas $\overline{R}(\vec{t})$ are called *negative literals*. The complement of a literal L is defined as follows:

- If $L = R(\vec{t})$, then $\overline{L} := \overline{R}(\vec{t})$.
- If $L = \overline{R}(\vec{t})$, then $\overline{L} := R(\vec{t})$.

Finite conjunctions of literals $L_1 \wedge \ldots \wedge L_n$ are called *goals* and denoted by G, H. The empty goal is $\Box$. If G is the goal $L_1 \wedge \ldots \wedge L_n$, then $\overline{G}$ is the formula $\overline{L_1} \vee \ldots \vee \overline{L_n}$. A *clause* K is a formula $G \to A$. The clause $\Box \to A$ is identified with the atom A. A *normal logic program* P is a finite set of clauses.

We use small greek letters σ, τ, θ, α, β for substitutions. The identity substitution is denoted by ε. We define

- $\mathrm{dom}(\sigma) := \{x \mid x \neq x\sigma\}$,
- $\mathrm{ran}(\sigma) := \bigcup\{\mathrm{vars}(x\sigma) \mid x \in \mathrm{dom}(\sigma)\}$,
- $\mathrm{vars}(\sigma) := \mathrm{dom}(\sigma) \cup \mathrm{ran}(\sigma)$.

We will use the following well-known properties of substitutions:

- A substitution σ is idempotent $[\sigma \circ \sigma = \sigma]$ iff $\mathrm{dom}(\sigma) \cap \mathrm{ran}(\sigma) = \emptyset$.
- If τ is an idempotent, most general unifier of the two atoms A and B [written $\tau = \mathrm{mgu}(A, B)$], then $\mathrm{vars}(\tau) \subseteq \mathrm{vars}(A) \cup \mathrm{vars}(B)$.
- If σ is idempotent, $\tau = \mathrm{mgu}(A\sigma, B)$ and B does not contain variables from $A\sigma$ or $\mathrm{dom}(\sigma)$, then $\sigma\tau$ is idempotent, too.

Let $*$ be a new symbol. *Goal forms* are expressions Γ of the form $G_1 \wedge * \wedge G_2$. The symbol $*$ marks a hole in the goal form. If Γ is the goal form $G_1 \wedge * \wedge G_2$, then $\Gamma[H]$ is the goal $G_1 \wedge H \wedge G_2$; $\Gamma[]$ is the goal $G_1 \wedge G_2$; $\Gamma\sigma$ is the goal form $G_1\sigma \wedge * \wedge G_2\sigma$. For example, $\Gamma\sigma[H\tau]$ is the goal $G_1\sigma \wedge H\tau \wedge G_2\sigma$.

We write $G \geq H$, if there exists a substitution σ such that $G\sigma = H$. If $G \geq H$, then we say that G is more general than H, or that H is an instance of G. We write $F[\vec{x}]$ to indicate that all free variables of the formula F are contained in the list $\vec{x}$. $\forall(F)$ denotes the universal closure of a formula F. For unexplained notions we refer to [1] and [7].

3 The partial completion of logic programs

The partial completion of normal logic programs has been introduced in [9, 12]. It is called *doubled program* in [13]. The difference to Clark's completion in [6] is that the positive relation symbols R are not the logical complements of the relation symbols $\overline{R}$. For example, the axiom $R(\vec{x}\,) \vee \overline{R}(\vec{x}\,)$ does not belong to the partial completion of logic programs. The axiom expresses that $R(\vec{t}\,)$ succeeds or fails and this not true in general. The goal $R(\vec{t}\,)$ can also loop. The partial completion is obtained from Clark's completion by splitting it into two parts, axioms for R and axioms for $\overline{R}$.

The partial completion of P [denoted by *pcomp*(P)] comprises the universal closures of the following axioms:

I. Clark's equational theory CET **for unification:**

(1) $x = x$.
(2) $x = y \to y = x$.
(3) $x = y \wedge y = z \to x = z$.
(4) $x_1 = y_1 \wedge \ldots \wedge x_n = y_n \to f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$. [if f is n-ary]
(5) $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \to x_i = y_i$. [if f is n-ary and $1 \leq i \leq n$]
(6) $f(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_m)$. [if f is n-ary, g is m-ary, and $f \neq g$]
(7) $t(x) \neq x$. [if $t(x)$ is a term, $t(x) \neq x$, and x occurs in $t(x)$]

II. Equality axioms for relation symbols:

(8) $x_1 = y_1 \wedge \ldots \wedge x_n = y_n \wedge R(x_1, \ldots, x_n) \to R(y_1, \ldots, y_n)$
(9) $x_1 = y_1 \wedge \ldots \wedge x_n = y_n \wedge \overline{R}(x_1, \ldots, x_n) \to \overline{R}(y_1, \ldots, y_n)$

III. The clauses in P:

(10) $G \to A$, for each clause $(G \to A) \in P$.

IV. Axioms for the relation symbols $\overline{R}$: Let R be an n-ary relation symbol and assume that the clauses for R in P are

$$L_{i,1}[\vec{u}\,] \wedge \ldots \wedge L_{i,k_i}[\vec{u}\,] \to R(t_{i,1}[\vec{u}\,], \ldots, t_{i,n}[\vec{u}\,]), \quad \text{for } i = 1, \ldots, m.$$

Then we have the following axiom for $\overline{R}$, where $\vec{y}$ is a list of new variables:

$$(11)\quad \left(\bigwedge_{i=1}^{m} \forall \vec{y} \left(\bigwedge_{j=1}^{n} x_j = t_{i,j}[\vec{y}\,] \to \bigvee_{j=1}^{k_i} \overline{L}_{i,j}[\vec{y}\,] \right) \right) \to \overline{R}(x_1, \ldots, x_n).$$

The left hand side of (11) is obtained by negating Clark's completed definition of a predicate and putting it into negation normal form. Axiom (11) can be read as follows: if for all clauses in P, such that the head of the clause matches $R(\vec{x}\,)$, one of the literals in the body fails, then $R(\vec{x}\,)$ fails.

4 SLDNF-resolution

We follow Buchholz' presentation in [4]. We distinguish, however, between positive and negative nodes in SLDNF-trees. Moreover, we separate the current substitution from a goal. This simplifies the notation in the completeness proof. The nodes of an SLDNF-tree are not resultants but so-called *frames*, i.e. triples $\langle S, G, \sigma\rangle$ such that $S \in \{+,-\}$, G is a goal and σ is an idempotent substitution. The sign S indicates whether we are looking for a solution for the goal $G\sigma$ or whether we want the goal $G\sigma$ to fail. The frame is called *positive* or *negative* according to its sign S. A *floundering* frame is a positive frame $\langle +, G, \sigma\rangle$ such that $G\sigma$ consists of negative non-ground literals only.

First we define the notions *resolvent* and *applicable clause.*

Definition 4.1
The frame $\langle S, \Gamma[H], \sigma\tau\rangle$ is called a *resolvent* of $\langle S, G, \sigma\rangle$ with respect to (Γ, K, ι) [in symbols: $\langle S, G, \sigma\rangle \longrightarrow_{(\Gamma,K,\iota)} \langle S, \Gamma[H], \sigma\tau\rangle$], if there exist A and B such that

- $G = \Gamma[A]$,
- $H \to B$ is the variant obtained from clause K by adding the index ι to each variable of K, and
- $\tau = \mathrm{mgu}(A\sigma, B)$ [an idempotent most general unifier].

In the triple (Γ, K, ι) the goal form Γ indicates the position of the selected literal in G, K is the input clause of the resolution step and the index ι is used to uniquely rename the variables of K to make them different from any new variable that is used elsewhere in a computation. If variables with index ι do not occur in G or σ and the substitution σ of the frame is idempotent, then the substitution $\sigma\tau$ of the resolvent is idempotent, too.

Definition 4.2
A clause $H \to B$ is *applicable* to A if there exist σ and τ such that $A\sigma = B\tau$. We set $P(A) := \{K \in P \mid K \text{ is applicable to } A\}$.

An SLDNF-tree is a finitely branching, downward growing (possibly infinite) tree of signed frames which is correct with respect to the rules of Table 1. Before we give a mathematical definition of SLDNF-tree, we explain the rules of Table 1 informally:

T1: If the goal of the frame is empty, then the frame is a leaf node.

T2: (Resolution node) In the frame $\langle S, \Gamma[A], \sigma\rangle$, a positive literal A is selected. The successors of the node are the resolvents using clauses applicable to $A\sigma$. We assume that the variables of the input clause $H_i \to B_i$ are new and are not used elsewhere in the tree for the renaming of input clauses.

T3: (Positive NaF node) In the positive frame $\langle +, \Gamma[\overline{A}], \sigma\rangle$ a negative literal $\overline{A}$ is selected, such that $\overline{A}\sigma$ is ground. By deleting the literal $\overline{A}$ in the frame we obtain the left successor of the node. The right successor is the negative frame $\langle -, A\sigma, \varepsilon\rangle$.

T4: (Negative NaF node) In a negative frame $\langle -, \Gamma[\overline{A}], \sigma\rangle$ the literal $\overline{A}\sigma$ may contain free variables when it is selected. Therefore it is not deleted in the left successor of the node. (In a fair computation it must be selected later again, when it is more instantiated.) The right successor of the node is the positive frame $\langle +, A, \sigma\rangle$.

T5: Floundering frames are leaf nodes in the tree.

Each time when a negative literal is selected the sign switches from plus to minus and vice versa (in the right subtree, only).

TABLE 1. Rules for an SLDNF-tree

T1 :	$\langle S, \Box, \sigma\rangle$	
T2 :	$\begin{array}{ccc} & \langle S, \Gamma[A], \sigma\rangle & \\ \swarrow & & \searrow \\ \langle S, \Gamma[H_1], \sigma\tau_1\rangle & \cdots & \langle S, \Gamma[H_n], \sigma\tau_n\rangle \end{array}$	if $\begin{cases} P(A\sigma) = \{K_1, \ldots, K_n\}, \\ H_i \to B_i \text{ a variant of } K_i, \\ \tau_i = \mathrm{mgu}(A\sigma, B_i). \end{cases}$
T3 :	$\begin{array}{ccc} & \langle +, \Gamma[\overline{A}], \sigma\rangle & \\ \swarrow & & \searrow \\ \langle +, \Gamma[], \sigma\rangle & & \langle -, A\sigma, \varepsilon\rangle \end{array}$	if $\mathrm{vars}(A\sigma) = \emptyset$.
T4 :	$\begin{array}{ccc} & \langle -, \Gamma[\overline{A}], \sigma\rangle & \\ \swarrow & & \searrow \\ \langle -, \Gamma[\overline{A}], \sigma\rangle & & \langle +, A, \sigma\rangle \end{array}$	
T5 :	$\langle +, \overline{A_1} \wedge \ldots \wedge \overline{A_n}, \sigma\rangle$	if $\mathrm{vars}(A_i\sigma) \neq \emptyset$ $(1 \leq i \leq n)$.

DEFINITION 4.3
(Cf. Buchholz [4]) Let P be a logic program. An SLDNF-tree for P is a function T such that

- $\mathrm{dom}(T) \subseteq \{\langle \iota_0, \ldots, \iota_{n-1}\rangle \mid n \in \mathbb{N} \;\&\; \iota_j \in P \cup \{0, 1\} \text{ for } j < n\}$,
- $\mathrm{dom}(T) \neq \emptyset$,
- $\forall \langle \iota_0, \ldots, \iota_n\rangle \in \mathrm{dom}(T)\, (\langle \iota_0, \ldots, \iota_{n-1}\rangle \in \mathrm{dom}(T))$,
- for each $\xi \in \mathrm{dom}(T)$ and $J = \{\iota \mid \xi * \langle\iota\rangle \in \mathrm{dom}(T)\}$, $T(\xi)$ is a signed frame and one of the conditions T1–T5 is satisfied:

T1: $T(\xi) = \langle S, \Box, \sigma\rangle$ and $J = \emptyset$.
T2: $T(\xi) = \langle S, \Gamma[A], \sigma\rangle$, $J = P(A\sigma)$ and
    $T(\xi) \longrightarrow_{(\Gamma, K, \xi * \langle K\rangle)} T(\xi * \langle K\rangle)$ for each $K \in J$.
T3: $T(\xi) = \langle +, \Gamma[\overline{A}], \sigma\rangle$, $\mathrm{vars}(A\sigma) = \emptyset$, $J = \{0, 1\}$,
    $T(\xi * \langle 0\rangle) = \langle +, \Gamma[], \sigma\rangle$ and $T(\xi * \langle 1\rangle) = \langle -, A\sigma, \varepsilon\rangle$.
T4: $T(\xi) = \langle -, \Gamma[\overline{A}], \sigma\rangle$, $J = \{0, 1\}$,
    $T(\xi * \langle 0\rangle) = \langle -, \Gamma[\overline{A}], \sigma\rangle$ and $T(\xi * \langle 1\rangle) = \langle +, A, \sigma\rangle$.
T5: $T(\xi)$ is floundering and $J = \emptyset$.

A node $\xi \in \mathrm{dom}(T)$ is called *positive* or *negative* according to whether the frame $T(\xi)$ is positive or negative. In cases T1 and T5, ξ is called a *leaf.* In case T2, ξ is called a *resolution node* and A is called the *selected atom.* If J is empty in T2, then ξ is called a leaf, too. In cases T3 and T4, ξ is called a *NaF node* and $\overline{A}$ is called the *selected literal.* We set $\Lambda := \langle\rangle$ (the root of the tree).

Given an SLDNF-tree T we can define what it means that a node returns an answer. We define a relation $\xi \Vdash X$ between nodes of T and generalized answers. A *generalized* *answer* is a substitution or the symbol *no*. The relation '$\xi \Vdash \theta$' is read as 'ξ yields computed answer θ'; the relation '$\xi \Vdash$ *no*' is read as 'ξ is finitely failed'. Table 2 makes the following definition more transparent.

Definition 4.4
(Cf. Buchholz [4]) Let T be an SLDNF-tree. The relation '$\Vdash$' is the least relation between nodes and generalized answers satisfying the following conditions for each $\xi \in \mathrm{dom}(T)$ and $J = \{\iota \mid \xi * \langle\iota\rangle \in \mathrm{dom}(T)\}$:

A1: $T(\xi) = \langle +, \Box, \sigma\rangle \implies \xi \Vdash \sigma$.
A2: ξ a positive resolution node, $\exists K \in J\,(\xi * \langle K\rangle \Vdash \theta) \implies \xi \Vdash \theta$.
A3: ξ a negative resolution node, $\forall K \in J\,(\xi * \langle K\rangle \Vdash \mathit{no}) \implies \xi \Vdash \mathit{no}$.
A4: ξ a positive NaF node, $\xi * \langle 0\rangle \Vdash \theta$, $\xi * \langle 1\rangle \Vdash \mathit{no} \implies \xi \Vdash \theta$.
A5: ξ a negative NaF node, $\xi * \langle 0\rangle \Vdash \mathit{no} \implies \xi \Vdash \mathit{no}$.
A6: ξ a negative NaF node, $T(\xi * \langle 1\rangle) = \langle +, A, \sigma\rangle$, $\xi * \langle 1\rangle \Vdash \theta$, $A\theta \geq A\sigma \implies \xi \Vdash \mathit{no}$.

It is easy to see that for any node ξ, if $T(\xi) = \langle +, G, \sigma\rangle$ and $\xi \Vdash \theta$, then there exists a substitution τ such that $\sigma\tau = \theta$. Therefore, in A6, the condition $A\theta \geq A\sigma$ implies that $A\theta$ is a variant of $A\sigma$. Rule A6 thus says: if $A\sigma$ succeeds with answer the identity substitution, then the goal $\Gamma[\overline{A}]\sigma$ fails.

5 Soundness of SLDNF-resolution

The main result of Clark in [6] is that SLDNF-resolution is sound with respect to the completion of a logic program (which is not so complete as its name suggests). Clark's completion of a logic program can be obtained from the partial completion by adding the following axioms for each relation symbol R:

$$\neg(R(\vec{x}\,) \wedge \overline{R}(\vec{x}\,)), \quad R(\vec{x}\,) \vee \overline{R}(\vec{x}\,).$$

The axioms say that $\overline{R}$ is the complement of R. Although the completion is stronger than the partial completion, Clark's proof works for the partial completion, too. This means that SLDNF-resolution is sound for the partial completion as well.

Theorem 5.1
(Soundness) Let T be an SLDNF-tree and $\xi \in \mathrm{dom}(T)$.

(a) If $T(\xi) = \langle +, G, \sigma\rangle$ and $\xi \Vdash \theta$, then $\mathit{pcomp}(P) \models \forall(G\theta)$.
(b) If $T(\xi) = \langle -, G, \sigma\rangle$ and $\xi \Vdash \mathit{no}$, then $\mathit{pcomp}(P) \models \forall(\overline{G}\sigma)$.

The main purpose of this paper is to prove the converse of this theorem for fair, non-floundering SLDNF-trees.

6 The completeness proof

Before we start with the proof we have to say what we mean by fair and non-floundering. We need the notion of a negative main branch. A negative main branch in an SLDNF-tree consists of negative resolution and negative NaF nodes (T2 and T4). In a NaF node (T4), the left successor node is chosen. A negative main branch is infinite or stops at a leaf node.

TABLE 2. Rules for computing answers in an SLDNF-tree

A1 :
$$\langle +, \Box, \sigma\rangle \Vdash \sigma$$

A2 :
$$\begin{tikzcd}[column sep=small] & \langle +, \Gamma[A], \sigma\rangle \Vdash \theta & \\ \cdots & \langle +, \Gamma[H_i], \sigma\tau_i\rangle \Vdash \theta \arrow[u] & \cdots \end{tikzcd}$$

A3 :
$$\begin{tikzcd}[column sep=small] & \langle -, \Gamma[A], \sigma\rangle \Vdash \mathit{no} & \\ \langle -, \Gamma[H_1], \sigma\tau_1\rangle \Vdash \mathit{no} \arrow[ur] & \cdots & \langle -, \Gamma[H_n], \sigma\tau_n\rangle \Vdash \mathit{no} \arrow[ul] \end{tikzcd}$$

A4 :
$$\begin{tikzcd}[column sep=small] & \langle +, \Gamma[\overline{A}], \sigma\rangle \Vdash \theta & \\ \langle +, \Gamma[], \sigma\rangle \Vdash \theta \arrow[ur] & & \langle -, A\sigma, \varepsilon\rangle \Vdash \mathit{no} \arrow[ul] \end{tikzcd}$$

A5 :
$$\begin{tikzcd}[column sep=small] & \langle -, \Gamma[\overline{A}], \sigma\rangle \Vdash \mathit{no} & \\ \langle -, \Gamma[\overline{A}], \sigma\rangle \Vdash \mathit{no} \arrow[ur] & & \cdots \end{tikzcd}$$

A6 :
$$\begin{tikzcd}[column sep=small] & \langle -, \Gamma[\overline{A}], \sigma\rangle \Vdash \mathit{no} & \\ \cdots & & \langle +, A, \sigma\rangle \Vdash \theta \arrow[ul] \end{tikzcd}$$
if $A\theta \geq A\sigma$.

DEFINITION 6.1
(Cf. Buchholz [4]) A *negative main branch* in T is a sequence of negative nodes $(\xi_j)_{j<N}$ such that $0 < N \leq \omega$ and for all $j < N$:

- $j + 1 < N \implies \xi_{j+1} = \xi_j * \langle \iota \rangle$ for some $\iota \in P \cup \{0\}$,
- $j + 1 = N \implies \xi_j$ is a leaf node in T.

DEFINITION 6.2
A negative main branch $(\xi_j)_{j<N}$ is called *fair* if

- it terminates in a leaf node ξ such that $\xi \Vdash \mathit{no}$, or
- for each literal L in it, after finitely many steps a descendent of L is selected.

An SLDNF-tree T is called *fair* if all its negative main branches are fair. The tree T is called *non-floundering* if it does not contain a floundering node.

Let T be a fair, non-floundering SLDNF-tree for P. Our goal is to extract from T a term model $\mathcal{M}$ of *pcomp*(P) with the following properties:

(1) If $T(\Lambda) = \langle +, G, \varepsilon \rangle$ and $\mathcal{M} \models \forall(G\sigma)$, then there exists an answer θ such that $\Lambda \Vdash \theta$ and $G\theta \geq G\sigma$.

(2) If $T(\Lambda) = \langle -, G, \varepsilon \rangle$ and $\mathcal{M} \models \forall(\overline{G})$, then $\Lambda \Vdash$ *no*.

Let I be the set of nodes $\iota \in \mathrm{dom}(T)$ such that $T(\iota) = \langle -, A, \varepsilon \rangle$, $\iota \nVdash$ *no* and ι is the successor of a positive NaF node, i.e. the right child in a configuration T3. (Note that $\mathrm{vars}(A) = \emptyset$ in this case.) If $T(\Lambda) = \langle -, G, \varepsilon \rangle$ and $\Lambda \nVdash$ *no*, then we include Λ into I, too. For each $\iota \in I$ let $(\xi^\iota_j)_{j<N_\iota}$ be a negative main branch in T such that

(3) $\xi^\iota_0 = \iota$,

(4) $\xi^\iota_j \nVdash$ *no* for each $j < N_\iota$,

(5) $T(\xi^\iota_j) = \langle -, G^\iota_j, \sigma^\iota_j \rangle$ for $j < N_\iota$.

Such negative main branches always exist. We show how they can be found. Assume that an initial segment of the branch has been constructed up to ξ^ι_j and that $\xi^\iota_j \nVdash$ *no*. If ξ^ι_j is a negative resolution node, then, by rule A3, there exists a $K \in P$ such that $\xi^\iota_j * \langle K \rangle \nVdash$ *no* and we set $\xi^\iota_{j+1} := \xi^\iota_j * \langle K \rangle$. If ξ^ι_j is a negative NaF node, then, by rule A5, $\xi^\iota_j * \langle 0 \rangle \nVdash$ *no* and we set $\xi^\iota_{j+1} := \xi^\iota_j * \langle 0 \rangle$.

The negative main branches $(\xi^\iota_j)_{j<N_\iota}$ have the following properties:

(6) $(\xi^\iota_j)_{j<N_\iota}$ is a fair branch in T for each $\iota \in I$.

(7) If $\iota, \kappa \in I$ and $\iota \neq \kappa$, then the set $\mathrm{vars}(G^\iota_m) \cup \mathrm{vars}(\sigma^\iota_m)$ is disjoint from the set $\mathrm{vars}(G^\kappa_n) \cup \mathrm{vars}(\sigma^\kappa_n)$ for all $m < N_\iota$ and $n < N_\kappa$.

(8) If $\iota, \kappa \in I$ and $\iota \neq \kappa$, then $\sigma^\iota_m \circ \sigma^\kappa_n = \sigma^\kappa_n \circ \sigma^\iota_m$ for all $m < N_\iota$ and $n < N_\kappa$.

(9) If $m \leq n < N_\iota$, then $\sigma^\iota_m \circ \sigma^\iota_n = \sigma^\iota_n$.

Let SUB be the set of substitutions $\sigma^{\iota_1}_{j_1} \circ \ldots \circ \sigma^{\iota_k}_{j_k}$ so that $k \in \mathbb{N}$, $\iota_1, \ldots, \iota_k$ are pairwise different elements from I and $j_1 < N_{\iota_1}, \ldots, j_k < N_{\iota_k}$. From (8) and (9) it follows that the set of substitutions SUB has the following properties:

(10) If $\sigma \in \mathrm{SUB}$ and $\tau \in \mathrm{SUB}$, then there exists a substitution $\theta \in \mathrm{SUB}$ such that $\sigma \circ \theta = \theta = \tau \circ \theta$.

(11) If $\sigma \in \mathrm{SUB}$ then $\sigma \circ \sigma = \sigma$.

The algebraic part of the model $\mathcal{M}$ is defined in the following way:

(12) The universe $|\mathcal{M}|$ is the set of all terms (with variables).

(13) $f^{\mathcal{M}}(a_1, \ldots, a_n) := f(a_1, \ldots, a_n)$ for $a_1, \ldots, a_n \in |\mathcal{M}|$.

(14) $s =^{\mathcal{M}} t \;:\Longleftrightarrow\; \exists \theta \in \mathrm{SUB}(s\theta = t\theta)$.

Equality is not interpreted as identity but by the equivalence relation $=^{\mathcal{M}}$. For $\vec{s} = s_1, \ldots, s_n$ and $\vec{t} = t_1, \ldots, t_n$ we write $\vec{s} =^{\mathcal{M}} \vec{t}$, if $s_i =^{\mathcal{M}} t_i$ for $i = 1, \ldots, n$.

Lemma 6.3
$\mathcal{M} \models$ CET.

Proof. The set SUB is directed (cf. [1] or [3]). ∎

For $\iota \in I$ we define $\mathrm{LIT}_\iota := \{L \mid \exists j < N_\iota\, (L \text{ occurs in } G^\iota_j)\}$. Moreover, we set

$$\mathrm{LIT} := \bigcup_{\iota \in I} \mathrm{LIT}_\iota.$$

The set LIT is in general not a model of P, since it need not satisfy clauses which are never used in a negative main branch of the tree. It is, however, *supported* by P in the sense of Apt, Blair and Walker [2] as the following lemma shows.

Lemma 6.4
If $R(\vec{s}) \in \mathrm{LIT}$, then there exists an instance $L_1 \wedge \ldots \wedge L_n \to R(\vec{t})$ of a clause of P such that $\vec{s} =^{\mathcal{M}} \vec{t}$ and $\{L_1, \ldots, L_n\} \subseteq \mathrm{LIT}$.

Proof. Assume that $R(\vec{s}) \in \mathrm{LIT}$. There exists a $\iota \in I$ such that $R(\vec{s}) \in \mathrm{LIT}_\iota$. There exists an $m < N_\iota$ such that $R(\vec{s})$ occurs in G^ι_m. Since the negative main branch $(\xi^\iota_j)_{j<N_\iota}$ is fair, there exists a $k \geq m$ such that $R(\vec{s})$ is selected at ξ^ι_k. This means that $T(\xi^\iota_k) = \langle -, \Gamma[R(\vec{s})], \sigma^\iota_k \rangle$ and there exists a variant $H \to R(\vec{t})$ of a clause K of P such that

- $\mathrm{dom}(\sigma^\iota_k) \cap \mathrm{vars}(R(\vec{t})) = \emptyset$,
- $\tau = \mathrm{mgu}(R(\vec{s})\sigma^\iota_k, R(\vec{t}))$,
- $\xi^\iota_{k+1} = \xi^\iota_k * \langle K \rangle$ and
- $T(\xi^\iota_{k+1}) = \langle -, \Gamma[H], \sigma^\iota_k \tau \rangle$.

We have $\sigma^\iota_{k+1} = \sigma^\iota_k \tau$ and $R(\vec{s})\sigma^\iota_{k+1} = R(\vec{t})\sigma^\iota_{k+1}$. Since $\sigma^\iota_{k+1} \in \mathrm{SUB}$, we obtain that $\vec{s} =^{\mathcal{M}} \vec{t}$. Since $G^\iota_{k+1} = \Gamma[H]$, all the literals of the body H belong to LIT. ∎

Since we want the structure $\mathcal{M}$ to have property (2), we have to ensure that, if $R(\vec{t})$ is in LIT, then $\overline{R}(\vec{t})$ is not true in $\mathcal{M}$. This is the motivation for the following interpretation of the relation symbols $\overline{R}$ in $\mathcal{M}$:

(15) $\overline{R}^{\mathcal{M}} := \{\langle \vec{t} \rangle \mid \forall \vec{s} (\vec{t} =^{\mathcal{M}} \vec{s} \implies R(\vec{s}) \notin \mathrm{LIT})\}$.

Since $=^{\mathcal{M}}$ is transitive, we immediately obtain:

(16) $\mathcal{M} \models x_1 = y_1 \wedge \ldots \wedge x_n = y_n \wedge \overline{R}(x_1, \ldots, x_n) \to \overline{R}(y_1, \ldots, y_n)$.

Since, by (11), $\vec{t}\sigma =^{\mathcal{M}} \vec{t}$ for each substitution $\sigma \in \mathrm{SUB}$, we obtain:

(17) If $\langle \vec{t} \rangle \in \overline{R}^{\mathcal{M}}$ and $\sigma \in \mathrm{SUB}$, then $\langle \vec{t}\sigma \rangle \in \overline{R}^{\mathcal{M}}$.

Since we want that $\mathcal{M}$ is a model of the clauses of P, each positive literal, which can be derived from negative, true literals in $\mathcal{M}$ using clauses from P, has to be true in $\mathcal{M}$, too. To make this more clear we need the notion of an implication tree (cf. [2] and [10]).

Definition 6.5
Implication trees (w.r.t. to P and $\mathcal{M}$) are generated as follows:

- If $\langle \vec{t} \rangle \in \overline{R}^{\mathcal{M}}$, then $\overline{R}(\vec{t})$ is an implication tree for $\overline{R}(\vec{t})$.
- If F_j is an implication tree for L_j $(1 \leq j \leq n)$ and $L_1 \wedge \ldots \wedge L_n \to A$ is an instance of a clause of P, then $A(F_1, \ldots, F_n)$ is an implication tree for A.

We say that L has an implication tree, if there exists an implication tree for L.

Let $\mathrm{IMP} := \{L \mid L \text{ has an implication tree}\}$. The set IMP is closed under substitutions from SUB:

(18) If $L \in \mathrm{IMP}$ and $\sigma \in \mathrm{SUB}$, then $L\sigma \in \mathrm{IMP}$.

The interpretations of the relation symbols R in $\mathcal{M}$ are defined by:

(19) $R^{\mathcal{M}} := \{\langle \vec{t} \rangle \mid \exists \sigma \in \mathrm{SUB}\, (R(\vec{t})\sigma \in \mathrm{IMP})\}$.

The structure $\mathcal{M}$ is now fully defined and we have:

Lemma 6.6
$\mathcal{M} \models x_1 = y_1 \wedge \ldots \wedge x_n = y_n \wedge R(x_1, \ldots, x_n) \to R(y_1, \ldots, y_n)$.

Proof. Assume that $\vec{s} =^{\mathcal{M}} \vec{t}$ and $\langle \vec{s} \rangle \in R^{\mathcal{M}}$. According to (19), there exists a substitution $\sigma \in \mathrm{SUB}$ such that $R(\vec{s})\sigma \in \mathrm{IMP}$. Moreover, by (10), there exists a substitution $\theta \in \mathrm{SUB}$ such that $\vec{s}\theta = \vec{t}\theta$ and $\sigma\theta = \theta$. It follows, by (18), that $R(\vec{s})\sigma\theta \in \mathrm{IMP}$. Thus $R(\vec{t})\theta \in \mathrm{IMP}$ and, by (19), $\langle \vec{t} \rangle \in R^{\mathcal{M}}$. ∎

Lemma 6.7
Assume that ξ_0 is an arbitrary node of T and $T(\xi_0) = \langle +, G_0, \sigma_0 \rangle$. Assume that α is a substitution such that every literal of $G_0\sigma_0\alpha$ has an implication tree. Then there exists an answer σ_n such that $\xi_0 \Vdash \sigma_n$ and $G_0\sigma_n \geq G_0\sigma_0\alpha$.

Proof. (See also [10].) Let n be the total number of literals in the implication trees for the literals in $G_0\sigma_0\alpha$. By induction on $i \leq n$, we show that there exists a branch $\xi_0, \ldots, \xi_i$ in T and sequences $G_0, \ldots, G_i$, $\sigma_0, \ldots, \sigma_i$, $\beta_0, \ldots, \beta_i$ such that the following conditions are satisfied:

(a) $T(\xi_i) = \langle +, G_i, \sigma_i \rangle$.
(b) $G_0\sigma_i\beta_i = G_0\sigma_0\alpha$.
(c) Each literal in $G_i\sigma_i\beta_i$ has an implication tree, such that the total number of literals in the trees is equal to $n - i$.
(d) If $0 < i$ and ξ_{i-1} is a NaF node, then $\xi_{i-1} * \langle 1 \rangle \Vdash$ *no* and $\xi_i = \xi_{i-1} * \langle 0 \rangle$.

Assume that $i < n$ and that (a)–(d) are satisfied. We show that there exist suitable ξ_{i+1}, G_{i+1}, σ_{i+1} and β_{i+1}. Since the SLDNF-tree T is not floundered, we have the following two cases:

Case I. ξ_i is a resolution node, i.e. $G_i = \Gamma[A]$ and A is the selected literal in G_i. There exists a clause $K = (H \to B)$ in P and a substitution θ such that $A\sigma_i\beta_i = B\theta$ and each literal in $\Gamma\sigma_i\beta_i[H\theta]$ has an implication tree such that the total number of literals in the implication trees is equal to $n - (i + 1)$. The clause K is applicable to $A\sigma_i$. Let $\xi_{i+1} := \xi_i * \langle K \rangle$. Let $H' \to B'$ be the variant of K that is used in the resolution step from ξ_i to ξ_{i+1}. Let $\tau = \mathrm{mgu}(A\sigma_i, B')$, $G_{i+1} = \Gamma[H']$ and $\sigma_{i+1} = \sigma_i\tau$. Let

$$V := \mathrm{vars}(G_0\sigma_i) \cup \mathrm{vars}(G_i\sigma_i) \cup \mathrm{dom}(\sigma_i).$$

Then we have $\mathrm{vars}(H' \to B') \cap V = \emptyset$, since the variables of $H' \to B'$ carry the index $\xi_i * \langle K \rangle$ whereas the indices of variables in V are initial segments of ξ_i. We can assume that $\mathrm{dom}(\beta_i) \subseteq V$. Let θ' be the substitution with

$$(H \to B)\theta = (H' \to B')\theta' \quad \text{and} \quad \mathrm{dom}(\theta') \subseteq \mathrm{vars}(H' \to B').$$

Let $\beta_{i+1} := \beta_i \cup \theta'$. Then

$$A\sigma_i\beta_{i+1} = A\sigma_i\beta_i = B\theta = B'\theta' = B'\beta_{i+1}.$$

Thus β_{i+1} is a unifier of $A\sigma_i$ and B'. Since τ is an idempotent most general unifier of $A\sigma_i$ and B', we obtain that $\tau\beta_{i+1} = \beta_{i+1}$.

(a) $T(\xi_{i+1}) = \langle +, G_{i+1}, \sigma_{i+1} \rangle$.
(b) $G_0\sigma_{i+1}\beta_{i+1} = G_0\sigma_i\tau\beta_{i+1} = G_0\sigma_i\beta_{i+1} = G_0\sigma_i\beta_i = G_0\sigma_0\alpha$.
(c) $G_{i+1}\sigma_{i+1}\beta_{i+1} = \Gamma[H']\sigma_i\tau\beta_{i+1} = \Gamma\sigma_i\beta_{i+1}[H'\beta_{i+1}] = \Gamma\sigma_i\beta_i[H'\theta']$.

Since ξ_i is not a NaF node, condition (d) is trivially satisfied.

Case II. ξ_i is a NaF node, i.e. $G_i = \Gamma[\overline{A}]$, $\overline{A}$ is the selected literal in G_i and $\text{vars}(A\sigma_i) = \emptyset$. Then

$$T(\xi_i * \langle 0 \rangle) = \langle +, \Gamma[], \sigma_i \rangle \quad \text{and} \quad T(\xi_i * \langle 1 \rangle) = \langle -, A\sigma_i, \varepsilon \rangle.$$

Let $\xi_{i+1} := \xi_i * \langle 0 \rangle$, $G_{i+1} := \Gamma[]$, $\sigma_{i+1} := \sigma_i$ and $\beta_{i+1} := \beta_i$. Then conditions (a)–(c) are satisfied. Assume that $A = R(\vec{t})$. Since $\text{vars}(A\sigma_i) = \emptyset$, we have $A\sigma_i = A\sigma_i\beta_i$. From assumption (c) we obtain that $\overline{R}(\vec{t})\sigma_i$ has an implication tree. By definition, this means that $\langle \vec{t}\sigma_i \rangle \in \overline{R}^{\mathcal{M}}$. Suppose that $\xi_i * \langle 1 \rangle \not\Vdash \textit{no}$. Then $\xi_i * \langle 1 \rangle \in I$ and $R(\vec{t}\sigma_i) \in \text{LIT}$. Thus $\langle \vec{t}\sigma_i \rangle \notin \overline{R}^{\mathcal{M}}$. Contradiction. Thus $\xi_i * \langle 1 \rangle \Vdash \textit{no}$ and condition (d) is satisfied.

Finally, consider the branch $(\xi_i)_{i \le n}$. By (c), the goal G_n must be the empty goal. By (d) and rules A2 and A4 for propagating answers in an SLDNF-tree, we obtain $\xi_i \Vdash \sigma_n$ for all $i \le n$. Hence, $\xi_0 \Vdash \sigma_n$ and, by (b), $G_0\sigma_n\beta_n = G_0\sigma_0\alpha$. ∎

We write $\mathcal{M} \models L$ [id], if the literal L is true in the structure $\mathcal{M}$ under the trivial variable assignment that assigns the element $x \in |\mathcal{M}|$ to each variable x. We have:

(20) $\mathcal{M} \models R(\vec{t})$ [id] $\iff \langle \vec{t} \rangle \in R^{\mathcal{M}} \iff \exists \sigma \in \text{SUB}\,(R(\vec{t})\sigma \in \text{IMP})$.
(21) $\mathcal{M} \models \overline{R}(\vec{t})$ [id] $\iff \langle \vec{t} \rangle \in \overline{R}^{\mathcal{M}} \iff \overline{R}(\vec{t}) \in \text{IMP}$.

Lemma 6.8
$L \in \text{LIT} \implies \mathcal{M} \not\models \overline{L}$ [id].

Proof. *Case I.* L is positive. Assume that $R(\vec{t}) \in \text{LIT}$. By (15), $\langle \vec{t} \rangle \notin \overline{R}^{\mathcal{M}}$. Thus $\mathcal{M} \not\models \overline{R}(\vec{t})$ [id].

Case II. L is negative. Suppose that $\overline{R}(\vec{t}) \in \text{LIT}$ and $\mathcal{M} \models R(\vec{t})$ [id]. By definition, this means that there exists a $\iota \in I$ such that $\overline{R}(\vec{t}) \in \text{LIT}_\iota$ and that there exists a $\tau \in \text{SUB}$ such that $R(\vec{t})\tau \in \text{IMP}$. There exists an $m < N_\iota$ such that $\overline{R}(\vec{t})$ occurs in G^ι_m. By (7), there exists an $n < N_\iota$ such that $R(\vec{t})\tau = R(\vec{t})\sigma^\iota_n$. (If the substitution $\tau \in \text{SUB}$ contains a component with superscript ι, then we take this component, otherwise we take $\sigma^\iota_0 = \varepsilon$.) Consider the negative main branch $(\xi^\iota_j)_{j<N_\iota}$. Since the branch is fair, there exists a $k \ge \max(m,n)$ such that the negative literal $\overline{R}(\vec{t})$ is selected in node ξ^ι_k. This means that

$$T(\xi^\iota_k) = \langle -, \Gamma[\overline{R}(\vec{t})], \sigma^\iota_k \rangle \quad \text{and} \quad T(\xi^\iota_k * \langle 1 \rangle) = \langle +, R(\vec{t}), \sigma^\iota_k \rangle.$$

Since $R(\vec{t})\sigma^\iota_n \in \text{IMP}$, it follows, by (18), that $R(\vec{t})\sigma^\iota_n\sigma^\iota_k \in \text{IMP}$. Since $n \le k$, we obtain, by (9), that $\sigma^\iota_n \circ \sigma^\iota_k = \sigma^\iota_k$ and thus $R(\vec{t})\sigma^\iota_k \in \text{IMP}$. By Lemma 6.7, it follows that there exists an answer θ such that $\xi^\iota_k * \langle 1 \rangle \Vdash \theta$ and $R(\vec{t})\theta \ge R(\vec{t})\sigma^\iota_k$. This implies $\xi^\iota_k \Vdash \textit{no}$ according to rule A6. Hence we have a contradiction to (4). ∎

Lemma 6.9
$\mathcal{M} \models pcomp(P)$.

Proof. Let R be an n-ary relation symbol and assume that the clauses for R in P are

$$L_{i,1}[\vec{u}\,] \land \ldots \land L_{i,k_i}[\vec{u}\,] \to R(t_{i,1}[\vec{u}\,], \ldots, t_{i,n}[\vec{u}\,]) \qquad (*)$$

for $i = 1, \ldots, m$. We have to show that $\mathcal{M}$ is a model of $(*)$ and that

$$\mathcal{M} \models \Bigg(\bigwedge_{i=1}^{m} \forall\vec{y}\,\Big(\bigwedge_{j=1}^{n} x_j = t_{i,j}[\vec{y}\,] \to \bigvee_{j=1}^{k_i} \overline{L}_{i,j}[\vec{y}\,]\Big)\Bigg) \to \overline{R}(x_1, \ldots, x_n). \qquad (**)$$

To show that the clauses $(*)$ are true in $\mathcal{M}$ we assume that $\vec{a} \in |\mathcal{M}|$ and

$$\mathcal{M} \models L_{i,j}[\vec{a}\,]\,[\mathrm{id}] \quad \text{for } j = 1, \ldots, k_i.$$

By (20) and (21), there exist substitutions $\sigma_j \in \mathrm{SUB}$ such that

$$L_{i,j}[\vec{a}\,]\sigma_j \in \mathrm{IMP} \quad \text{for } j = 1, \ldots, k_i.$$

By (10) and (18), there exists a substitution $\tau \in \mathrm{SUB}$ such that

$$L_{i,j}[\vec{a}\,]\tau \in \mathrm{IMP} \quad \text{for } j = 1, \ldots, k_i.$$

By the definition of implication tree, it follows that

$$R(t_{i,1}[\vec{a}\,], \ldots, t_{i,n}[\vec{a}\,])\tau \in \mathrm{IMP}.$$

By (20), we obtain that

$$\mathcal{M} \models R(t_{i,1}[\vec{a}\,], \ldots, t_{i,n}[\vec{a}\,])\,[\mathrm{id}].$$

Thus $\mathcal{M}$ is a model of clause $(*)$.

In order to show $(**)$ we assume that $\mathcal{M} \not\models \overline{R}(a_1, \ldots, a_n)\,[\mathrm{id}]$. We have to show that

$$\mathcal{M} \not\models \bigwedge_{i=1}^{m} \forall\vec{y}\,\Big(\bigwedge_{j=1}^{n} a_j = t_{i,j}[\vec{y}\,] \to \bigvee_{j=1}^{k_i} \overline{L}_{i,j}[\vec{y}\,]\Big)\,[\mathrm{id}].$$

Since $\langle a_1, \ldots, a_n\rangle \notin \overline{R}^{\mathcal{M}}$, by the definition of $\overline{R}$ in (15), there exist $s_1, \ldots, s_n$ such that $a_j =^{\mathcal{M}} s_j$ for $j = 1, \ldots, n$ and $R(s_1, \ldots, s_n) \in \mathrm{LIT}$. By Lemma 6.4, there exists an i and terms $\vec{b}$ such that $1 \le i \le m$ and

$$s_j =^{\mathcal{M}} t_{i,j}[\vec{b}\,] \quad \text{for } j = 1, \ldots, n \quad \text{and} \quad L_{i,j}[\vec{b}\,] \in \mathrm{LIT} \quad \text{for } j = 1, \ldots, k_i.$$

By Lemma 6.8, it follows that

$$\mathcal{M} \not\models \overline{L}_{i,j}[\vec{b}\,]\,[\mathrm{id}] \quad \text{for } j = 1, \ldots, k_i.$$

Hence we have

$$\mathcal{M} \models \bigwedge_{j=1}^{n} a_j = t_{i,j}[\vec{b}\,]\,[\mathrm{id}] \quad \text{and} \quad \mathcal{M} \not\models \bigvee_{j=1}^{k_i} \overline{L}_{i,j}[\vec{b}\,]\,[\mathrm{id}].$$

Thus $(**)$ is shown. ∎

Finally, we can turn to properties (1) and (2).

For property (1) assume that $T(\Lambda) = \langle +, G, \varepsilon\rangle$ and $\mathcal{M} \models \forall(G\sigma)$. Then we have $\mathcal{M} \models G\sigma$ [id]. We can assume that $\mathrm{vars}(G\sigma) \cap \mathrm{dom}(\tau) = \emptyset$ for all $\tau \in \mathrm{SUB}$. By (20) and (21), it follows that each literal of $G\sigma$ has an implication tree. So we can apply Lemma 6.7 and obtain a substitution θ such that $\Lambda \Vdash \theta$ and $G\theta \geq G\sigma$.

To show (2) we assume that $T(\Lambda) = \langle -, G, \varepsilon\rangle$ and $\Lambda \nVdash$ *no*. Then $\Lambda \in I$ and all literals of G belong to LIT. By Lemma 6.8, we obtain that $\mathcal{M} \not\models \overline{L}$ [id] for each literal L of G. Hence, $\mathcal{M} \not\models \overline{G}$ [id] and $\mathcal{M} \not\models \forall(\overline{G})$.

So we have proved the following theorem:

THEOREM 6.10
Let T be a non-floundering, fair SLDNF-tree for G with respect to P. Then we have:

(a) If *pcomp*$(P) \models \forall(G\sigma)$ and $T(\Lambda) = \langle +, G, \varepsilon\rangle$, then there exists a substitution θ such that $\Lambda \Vdash \theta$ and $G\theta \geq G\sigma$.
(b) If *pcomp*$(P) \models \forall(\overline{G})$ and $T(\Lambda) = \langle -, G, \varepsilon\rangle$, then $\Lambda \Vdash$ *no*.

7 Discussion

The model $\mathcal{M}$ constructed in the completeness proof is not very intuitive, since the denotations $R^{\mathcal{M}}$ and $\overline{R}^{\mathcal{M}}$ need not be disjoint in general. It is, however, always possible to add to *pcomp*(P) the axioms

$$\forall\vec{x}\,\neg\big(R(\vec{x}\,) \wedge \overline{R}(\vec{x}\,)\big) \tag{$*$}$$

without increasing its deductive power as far as positive formulas are concerned. The reason is that for each model of the partial completion there exists always a smaller model satisfying the axioms $(*)$ (cf. Theorem 6.1 in [12]).

Why is fairness required in negative branches only? This is because to get the answer *no*, it suffices that one literal fails, and the other literals can be discarded. An unfair computation would never consider this literal. In a positive branch, all literals are considered eventually to reach a solution θ.

Why do we keep $\Gamma[\overline{A}]$ in the left child instead of the more plausible $\Gamma[]$ in rule T4? This is because we have to delay the computation of A until it is instantiated enough. If we do not, then we lose completeness. Consider the program $P := \{R(c)\}$, where c is a constant. Then

$$\mathit{pcomp}(P) \models \forall x\,(\overline{R}(x) \vee R(x)).$$

Hence, by our completeness theorem, for each fair and non-floundering SLDNF-tree T with $T(\Lambda) = \langle -, R(x) \wedge \overline{R}(x), \varepsilon\rangle$ it must be that $\Lambda \Vdash$ *no*. If we changed rule T4 and deleted the negative literal in the left child, then the following would be a fair non-floundering SLDNF-tree:

$$\begin{tikzcd}
& \langle -, R(x) \wedge \overline{R}(x), \varepsilon\rangle \arrow[dl] \arrow[dr] & \\
\langle -, R(x), \varepsilon\rangle \arrow[d] & & \langle +, R(x), \varepsilon\rangle \arrow[d] \\
\langle -, \Box, \{c/x\}\rangle & & \langle +, \Box, \{c/x\}\rangle
\end{tikzcd}$$

For this tree, however, we do not have $\Lambda \Vdash no$, since $R(c) \not\geq R(x)$ and we cannot apply rule A6. Changing rule A6 would destroy the soundness of SLDNF-resolution.

What is the difference to Buchholz' notion of SLDNF-tree in [4]? First, we cannot define resolvents locally without a 'standardizing apart condition' on the variables of input clauses, since in the completeness proof we use the fact that variables occurring in input clauses of different negative main branches of the SLDNF-tree are disjoint. Therefore we rename input clauses by attaching the address of the node where the clause is used as an index to the variables of the clause. This comes close to implementations where renaming of clauses means allocating a new unused block on the stack. The address of the block corresponds to the address of a node in an SLDNF-tree.

In Buchholz' notion of SLDNF-tree there are no signs $S \in \{+, -\}$. In Table 1 he applies rule T3 if $A\sigma$ is ground, and rule T4 otherwise. Rule T5 is therefore not needed in Buchholz' definition. His notion of fairness is stronger, since each main branch of the tree has to be fair, whereas our condition requires only negative main branches to be fair. Nevertheless the following theorem is true for Buchholz' notion of SLDNF-tree.

THEOREM 7.1
Let T be a fair SLDNF-tree for G (in the sense of [4]). A node $\xi \in \mathrm{dom}(T)$ is called positive if the number of 1s in ξ is even, otherwise ξ is called negative. A node $\xi \in \mathrm{dom}(T)$ is called floundering if $T(\xi) = G \to H$ such that G consists of negative non-ground literals only.

(a) If $pcomp(P) \models \forall(G\sigma)$ and no positive node ξ of T is floundering, then there exists a substitution θ such that $T \Vdash \theta$ and $G\theta \geq G\sigma$.
(b) If $pcomp(P) \models \forall(\overline{G})$ and no negative node ξ of T is floundering, then $T \Vdash no$.

Acknowledgement

I am grateful to the anonymous referees for their valuable comments.

References

[1] K. R. Apt. Logic programming. In *Handbook of Theoretical Computer Science, Volume B*, chapter 10, J. van Leeuwen, ed. pp. 495–574. Elsevier, 1990.

[2] K. R. Apt, H. A. Blair and A. Walker. Towards a theory of declarative knowledge. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, ed. pp. 89–148. Morgan Kaufmann, Los Altos, 1987.

[3] H. A. Blair and A. L. Brown. Definite clause programs are canonical (over a suitable domain). *Annals of Mathematics and Artificial Intelligence*, **1**, 1–19, 1990.

[4] W. Buchholz. A note on SLDNF-resolution. *Journal of Logic and Computation*, **8**, 159–168, 1998.

[5] L. Cavedon and J. W. Lloyd. A completeness theorem for SLDNF-resolution. *Journal of Logic Programming*, **7**, 177–191, 1989.

[6] K. L. Clark. Negation as failure. In *Logic and Data Bases*, H. Gallaire and J. Minker, eds. pp. 293–322. Plenum Press, New York, 1978.

[7] K. Doets. *From Logic to Logic Programming.* The MIT Press, Cambridge, MA, 1994.

[8] W. Drabent. Completeness of SLDNF-resolution for non-floundering queries. *Journal of Logic Programming*, **27**, 89–106, 1996.

[9] G. Jäger and R. F. Stärk. A proof-theoretic framework for logic programming. In *Handbook of Proof Theory*, S. R. Buss, ed. pp. 639–682. Elsevier, 1998.

[10] R. F. Stärk. A direct proof for the completeness of SLD-resolution. In *Computer Science Logic, selected papers from CSL '89*, E. Börger, H. Kleine Büning and M. M. Richter, eds. pp. 382–383. Springer-Verlag, Lecture Notes in Computer Science 440, 1990.

[11] R. F. Stärk. Input/output dependencies of normal logic programs. *Journal of Logic and Computation*, **4**, 249–262, 1994.

[12] R. F. Stärk. From logic programs to inductive definitions. In *Logic: From Foundations to Applications, European Logic Colloquium '93*, W. A. Hodges *et al.*, eds. pp. 453–481. Clarendon Press, Oxford, 1996.

[13] A. Van Gelder and J. S. Schlipf. Commonsense axiomatizations for logic programs. *J. of Logic Programming*, **17**, 161–195, 1993.

[14] D. A. Wolfram, M. J. Maher and J.-L. Lassez. A unified treatment of resolution strategies for logic programs. In *Proc. 2nd International Conference on Logic Programming*, pp. 263–276, Uppsala, Sweden, 1984.

Received 11 July 1997