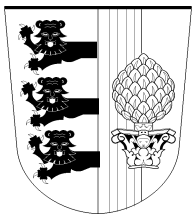
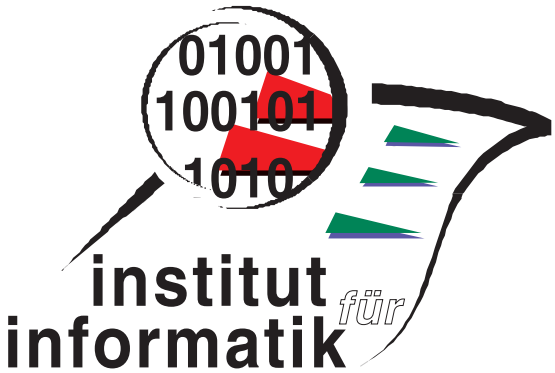
Universität Augsburg



Completeness Result of SLDNF-resolution for a relevant Class of Logic Programs

Ebénézer Ntienjem

Report 1997-04 December, 1997



Institut für Informatik
D-86135 Augsburg

Completeness Result of SLDNF-resolution for a relevant Class of Logic Programs

Ebénézer Ntienjem

December 15, 1997

Abstract

The proof theory of logic programming has been given by the SLDNF-resolution which has been proven complete for the class of arbitrary logic programs when assuming fair selection and non-floundering [4, 15]. To test the non-floundering condition is as hard as to resolve the problem itself. To overcome this assumption we first of all extend the universe to one that contains variables modulo renaming and define the bottom up SLDNF-resolution so that the elimination of this assumption is obvious. We then prove that the so defined SLDNF-resolution is sound and complete for a larger class of logic programs which does obviously contain the classes mentioned above.

Keywords: Logic Programming, Proof Theory, Model Theory, Semantics, Resolution

Contents

1 Introduction

We assume throughout this paper that the reader is acquainted with the basic notions of logic programming. If nothing else is noted, all notations used in the following are borrowed from Apt in [1] or Lloyd in [9]. We say logic program as a short hand for normal respectively general logic program as defined in [9] respectively [1].

We prove the following relevant theorem: let P be a logic program, Γ be a goal, and positive literals be resolved before negative ones whenever they have variables in common. Suppose furthermore that the logic program P is *well-formulated*.

(i) if $\text{pcomp}(P) \models \forall\Gamma\theta$, then the SLDNF-resolution of $P \cup \{\Gamma\}$ succeeds and yields an answer substitution ϑ which is more general that θ.

(ii) if $\text{pcomp}(P) \models \forall \sim\Gamma$, then the SLDNF-resolution of $P \cup \{\Gamma\}$ finitely fails.

By a *well-formulated* logic program we mean one which does not belong to the class of logic programs such that if P is a logic program and Γ is a goal, then $\text{pcomp}(P) \models \forall\Gamma$, but the SLDNF-resolution of $P \cup \{\Gamma\}$ neither succeeds nor finitely fails.

This result generalizes that of Stärk in [15], since non-gound negative literals will be resolved.

1.1 The state of affairs

Recent advances in improving the class of logic programs which is complete using the SLDNF-resolution have been made, but some serious problems still remain. The extension of the SLDNF-resolution in [15] is characterized by the following two rules:

(1) if A is ground and fails, then $\sim A$ succeeds.

(2) if A succeeds with the identity substitution, then $\sim A$ fails.

For the class of arbitrary logic programs, the completeness of the SDLNF-resolution has been proven when assuming that the selection is fair and a goal does not flounder [4] or when assuming that the SLDNF-tree for a goal is fair and does not flounder [15]. In all these cases something flounders if it does contain negative literals in which variables do occur. Non-floundering tries to ensure the SLDNF-resolution of a negative literal which does contain variables not to proceed clumsily.

1.2 Why a new definition of SLDNF-resolution

The test of the non-floundering condition is as hard as to resolve a goal; that lies in the fact that an SLDNF1 prefailed tree [4] respectively an SLDNF-tree [15] has first to be constructed.

We observe that some negative literals in which variables do occur may not proceed clumsily. To illustrate this observation, let us consider the well-known logic program `Append` with the goal

$$\bot \leftarrow \sim \texttt{append}(v, [1|w], [2,3,4]).$$

It is evident that $\texttt{append}(v, [1|w], [2,3,4])$ finitely fails. Hence, the goal

$$\bot \leftarrow \sim \texttt{append}(v, [1,|w], [2,3,4])$$

succeeds. A structure illustrating this fact is shown in figure 2 in subsection 8. The next logic program is interesting, since the formulation is a little different from the well known one in [16] or elsewhere in the literature. The reason why we use this formulation is to fully use the unification as the argument passing mechanism and the SLDNF-resolution as the operational semantics. Note that the unification and the SLDNF-resolution make logic programming more expressive than functional or imperative one. C_i, where $i \geq 1$, denotes a clause number for later reference to that clause if necessary.

$$\begin{array}{lllll}
C_1: & \text{plus}(0, y, y,) & \leftarrow & \\
C_2: & \text{plus}(s(x), 0, s(x),) & \leftarrow & \\
C_3: & \text{plus}(s(x), s(y), s(z)) & \leftarrow & \text{plus}(x, s(y), z) \\
C_4: & \text{times}(0, y, 0) & \leftarrow & \\
C_5: & \text{times}(s(x), 0, 0) & \leftarrow & \\
C_6: & \text{times}(s(x), s(y), s(z)) & \leftarrow & \text{times}(x, s(y), w), \text{plus}(w, s(y), s(z)) \\
C_7: & \text{factorial}(0, s(0)) & \leftarrow & \\
C_8: & \text{factorial}(s(x), s(z)) & \leftarrow & \text{factorial}(x, y), \text{times}(s(x), y, s(z))
\end{array}$$

Let us next consider the program `Factorial` with the goal

$$\bot \leftarrow \sim \texttt{factorial}(v, s(s(s(0)))),$$

where $s(n)$ denotes the successor of $n \in I\!N$. Intuitively,

$$\texttt{factorial}(v, s(s(s(0))))$$

finitely fails. Hence,

$$\bot \leftarrow \sim \texttt{factorial}(v, s(s(s(0))))$$

succeeds. A structure showing this fact is space consuming; we give instead a textual proof in subsection 8.

We furthermore observe that the formulations of the SLDNF-resolution are top down; even that given by Kunen in [8] is based on a top down style of formulation. By a top down formulation one assumes a goal be given and a literal be selected from that goal with respect to a well defined selection function. It is then obvious that the selected literal has to meet some conditions before it is applicable if the underlying method has to give a correct result at all.

1.3 Basic ideas of the new definition of SLDNF-resolution

To overcome these serious and restricting conditions and then still have a method which gives a correct result, we formulate the SLDNF-resolution bottom up. Let then first of all consider an extension of the SLDNF-resolution which is new since it has not been formulated elsewhere in the literature before (cf. [2]), and which differs from the above one used in [15]. We propose an extension of the SLDNF-resolution with respect to a logic program, say P. Let us be simple and characterize our extension by the following two rules:

NaFF1 if the body of each clause in P fails or A and the head of each clause in P are not unifiable, then $\sim A$ succeeds.

NaFF2 if the body of some clause in P succeeds with the renaming substitution and the unificator of A and the head of this clause is a renaming substitution, then $\sim A$ fails.

To ensure that our SLDNF-resolution is sound, it is necessary that positive literals be resolved before negative literals whenever the set of variables occurring in positive literals and the set of variables occurring in negative literals are not disjoint.

It is obvious that this extension of the SLDNF-resolution does include that in [15] which is characterized by the two rules (1) and (2) above, and is sound. To illustrate this claim, let us consider the program `Positive`

$$\begin{aligned} \texttt{positive}(x) &\leftarrow \sim \texttt{zero}(x) \\ \texttt{zero}(0) &\leftarrow \end{aligned}$$

with the goal $\bot\leftarrow \texttt{positive}(y)$. Using rule **NaFF2** one concludes that $\sim\texttt{zero}(y)$ does not fail finitely, since the body of the second clause obviously succeeds with a renaming substitution and the unificator of the head of the second clause and $\texttt{zero}(y)$ is not a renaming substitution. Hence, $\texttt{positive}(y)$ fails finitely. Suppose now any term t different from 0. Then $\texttt{zero}(t)$ fails finitely using rule **NaFF1**, and therefore $\texttt{positive}(t)$ succeeds. This also proves that the rules **NaFF1** and **NaFF2** are sound.

1.4 Organization of the work

This paper is organized as follows: in section 2 we briefly fix the syntax of our language, some basic notions and the partial completion. Section 4 is concerned with the semantics while in subsection 5 we discuss a natural inference system which behaves like a bottom up SLDNF-resolution. In section 6 we define an extended bottom up SLDNF-resolution which allows the selection of non-ground negative literals and prove that it is sound. In section 7 we discuss the completeness result, illustrate our result with some examples, and compare our result with some recent ones in the literature. We then in section 10 suggest some future works in progress.

2 Syntax and basic notions

We assume that our language $\mathcal{L}$ for first order predicate logic is fixed in advance, and does contain, for each $n \geq 0$ a countably infinite set of function symbols $\mathcal{F}_n$, for each $n \geq 0$ a countably infinite set of n-ary predicate symbols $\mathcal{P}_n$. Let $\mathcal{V}$ be a countably infinite set of variables. In addition, our language has particular predicate symbols, '$=$', for equality, and '$\cong$', for equality modulo variable renaming; the set $\cup_{n\in\mathbb{N}}\mathcal{P}_n$ does not contain $=$ and $\cong$.

Let then the syntactic categories $\mathcal{F}$ of function symbols, PRED of n-ary predicate symbols be given, $\mathcal{T}_{\mathcal{F},\mathcal{V}}$ of terms, FOR of formulae be defined as usual. Terms are denoted by r, s, t, and atomic formulae by A, B, α, β. In the sequel we use the syntactical symbol ',' to separate literals as a synonym for the logical symbol '$\wedge$'. We suppose that our syntactic category FOR does contain the particular formula $\bot$, *falsehood* which denotes a formula that is false at all or finitely failed. A *literal* is an atomic formula or a negated atomic formula. Literals are denoted by L, λ. A *program clause* or *clause* for short is a formula of the form

$$\alpha \leftarrow \lambda_1, \cdots, \lambda_n,$$

where α is an atomic formula which is also called the *head*, $\lambda_1, \cdots, \lambda_n = \Pi$ is a formula which is also called the *body*, and $n \geq 0$; we write $\alpha \leftarrow \epsilon$, if $n = 0$. A *program goal* or *goal* for short is a clause of the form

$$\bot\leftarrow \lambda_1, \cdots, \lambda_n,$$

where $n \geq 0$; we write ϵ if $n = 0$. A *logic program* or *program* for short is a finite set of clauses. Instead of considering a logic program to be a set of clauses we let

it be the union of the definitions of n-ary predicate symbols, where the definition P_λ of an n-ary predicate symbol of λ is the set of clauses such that this n-ary predicate symbol does occur in the head of each clause occurring in this set.

By an *expression* we mean a term or a formula. Let $vars(E)$ be the set of variables occurring in the expression E. $\forall E$ denotes the universal closure of E, and $\exists E$ the existential closure of E.

A substitution θ is a set $\{t_1/x_1, \ldots, t_n/x_n\}$ of bindings such that the terms t_i are different from the variables x_i for $1 \leq i \leq n$ and x_i is different from x_j for $1 \leq i < j \leq n$. The domain $dom(\theta)$ of the substitution θ is the set of variables $\{x_i \mid 1 \leq i \leq n\}$. The range $ran(\theta)$ of the substitution θ is the set of terms $\{t_i \mid 1 \leq i \leq n\}$. Let $\theta_{\restriction_V}$ denote the restriction of the substitution θ to the set of variables V, $\mathcal{R}$ denote variable renamings and $\varepsilon = \emptyset$ denote an identity substitution. In the following we say that a clause $\beta' \leftarrow \Pi'$ is a *variant* of a clause $(\beta \leftarrow \Pi) \in P$, if there exists $\rho \in \mathcal{R}$ such that

$$\beta' \leftarrow \Pi' = (\beta \leftarrow \Pi)\rho$$

holds.

The application of a substitution to an expression and the relation *more general than* between substitutions is defined in the usual way. A substitution θ is an *unifier* of expressions E and F if $E\theta = F\theta$, and is a *most general unifier* (in short: *mgu*) of E and F if it is an unifier which is more general than all other unifiers of E and F.

We suppose that our language of discourse has sufficiently many terms. One gets sufficiently many terms when assuming as in [7, 8] an infinite universal language in which all programs and goals occur. We assume in the following that the unification of expressions is defined and the properties of substitutions are stated as in [12]. In particular,

(i) if $\theta = \mathrm{mgu}(\lambda_1, \lambda_2)$ is idempotent, then $\mathrm{vars}(\theta) \subseteq (\mathrm{vars}(\lambda_1) \cup \mathrm{vars}(\lambda_2))$.

(ii) if σ is idempotent, $\theta = \mathrm{mgu}(\lambda_1, \lambda_2\sigma)$ and

$$\mathrm{vars}(\lambda_1) \cap (\mathrm{vars}(\lambda_2\sigma) \cup \mathrm{dom}(\sigma)) = \emptyset,$$

then $\sigma\theta$ is idempotent.

Let us for simplicity write $\vec{x}$ for $x_1, \ldots, x_n$ with $n \geq 0$, say expression for term or formula and write $\tau[\vec{x}\,]$ for an expression τ with all its actual variables among $\vec{x}$. Let Π or Δ or Γ be a short hand for $\lambda_1, \cdots, \lambda_n$ with $n \geq 1$, and $\sim$ be a new logical symbol which denotes "*finitely failed*" and acts like $\neg$.

3 Partial completion of logic programs

The *partial completion* of logic programs is introduced by Jäger in [6] and Stärk in [13]. The aim of the partial completion is to make the completion of a logic program not become inconsistent at all.

Let $\bar{q}$ be a new n-ary predicate symbol whenever q is an n-ary predicate symbol belonging to PRED. Then consider in the sequel the language $\bar{\mathcal{L}} = \mathcal{L} \cup \{\bar{q} \mid q \in$ PRED$\}$. A formula of the form $q(\vec{t})$ is a *positive literal* and a formula of the form $\bar{q}(\vec{t})$ is a *negative literal*. Note that for formulae $\sim$ is in general not $\neg$. If a literal λ is of the form $q(\vec{t})$, then $\sim\lambda := \bar{q}(\vec{t})$. If a literal λ is of the form $\bar{q}(\vec{t})$, then $\sim\lambda := q(\vec{t})$.

The *partially completed* definition of an n-ary predicate symbol is briefly obtained as follows: let the definition of an n-ary predicate symbol q consists of m clauses of the form $q(t_1, \ldots, t_n) \leftarrow \Pi$ and $x_1, \ldots, x_n$ be new variables. We write $q(t_{1,i}, \ldots, t_{n,i}) \leftarrow \Pi_i$ to refer to the i-th clause $1 \leq i \leq m$. Then the partially completed definition of q is

$$q(x_1, \ldots, x_n) \leftarrow \bigvee_{i=1}^{m} \exists\vec{y}\,((\bigwedge_{j=1}^{n}(x_{j,i} \cong t_{j,i}[\vec{y}])) \wedge \bigwedge_{l=1}^{k(i)} \lambda_l[\vec{y}]) \tag{1}$$

$$\bar{q}(x_1, \ldots, x_n) \leftarrow \bigwedge_{i=1}^{m} \forall\vec{y}\,((\bigvee_{j=1}^{n} \sim(x_{j,i} \cong t_{j,i}[\vec{y}])) \vee \bigvee_{l=1}^{k(i)} \sim\lambda_l[\vec{y}]). \tag{2}$$

Note that the formula (1) resp. (2) is a compact representation of a set of formulae of the form $q(\vec{t}) \leftarrow \Pi$ resp. $\bar{q}(\vec{t}) \leftarrow \sim\Pi$. Let P_λ resp. $\bar{P}_\lambda$ be a shorthand for (1) resp. (2) in the sequel. Let the Clark's Equational Theory, CET for short, be the equational axioms of the partially completed program. Note that CET does not depend on a program.

The *partial completion*, $pcomp(P)$, of P consists of the theory CET plus for each n-ary predicate symbol q of $\bar{\mathcal{L}}$

$$(\vec{x} = \vec{y}) \wedge q(\vec{x}\,) \rightarrow q(\vec{y}\,)$$

and plus the partially completed definition of each predicate symbol q of $\bar{\mathcal{L}}$.

Notice that for each n-ary predicate symbol $q \in$ PRED the new n-ary predicate symbol $\bar{q}$ is not the complement of q, that means the axiom $q(\vec{x}\,) \vee \bar{q}(\vec{x}\,)$ does not belong to $pcomp(P)$.

In the sequel $E \neq F$ stands for $\sim(E = F)$, $E \ncong F$ stands for $\sim(E \cong F)$ and $\sim\Pi, \sim\Delta, \sim\Gamma$ is a shorthand for $\sim\lambda_1 \vee \cdots \vee \sim\lambda_n$ with $n \geq 0$. A formula of the form $\sim\Gamma$ also denotes a goal. In the sequel ∇ denotes a goal of the form either Δ or $\sim\Delta$, and $D_q[\vec{x}\,]$ a formula of the form

$$\bigvee_{i=1}^{m} \exists\vec{y}\,((\bigwedge_{j=1}^{n}(x_{j,i} \cong t_{j,i}[\vec{y}])) \wedge \bigwedge_{l=1}^{k(i)} \lambda_l[\vec{y}]).$$

4 Semantics

Let $\mathcal{L}$ be a first order language for predicate logic as defined in section 2 above. When we are formulating an axiom system in $\mathcal{L}$, we have a particular structure, say $\mathcal{I}$, for $\mathcal{L}$ in mind. The structure we have in mind does contain variables modulo renaming. We discuss in this section the structure, the model semantics and a natural inference system. The natural inference system shall be important by the proof of the completeness of the SLDNF-resolution.

4.1 Structure

Let $\mathbb{B}$ be the set of boolean values. A *structure* $\mathcal{I}$ for a language consists of a nonempty set, the domain of discourse D, together with an assignment of a semantic object on D for each of the function symbols and predicate symbols of the language.

- Whenever f is an n-ary function symbol with $n \geq 1$, $\mathcal{I}(f)$ is a function from D^n into D; if $n = 0$, then $\mathcal{I}(f) \in D$.
- Whenever q is an n-ary predicate symbol other than '$=$' and '$\cong$' with $n \geq 1$, $\mathcal{I}(q)$ is a function from D^n into $\mathbb{B}$; if $n = 0$, then $\mathcal{I}(q) \in \mathbb{B}$.
- $\mathcal{I}(=)$ is the true identity.
- $\mathcal{I}(\cong)$ is the true identity modulo variable renaming.

If $\tau[\vec{t}\,]$ is a term, then we define $\mathcal{I}(\tau)\colon D^n \to D$ in the obvious way. Likewise, if $\varphi[\vec{t}\,]$ is a formula, then we define $\mathcal{I}(\varphi)\colon D^n \to \mathbb{B}$ in the obvious way.

We only do consider structures which satisfy CET. Such structures are characterized by the following three conditions:

cet1 $\mathcal{I}(f)$ is an injective function for each n-ary function symbol with $n \geq 1$.

cet2 $\mathcal{I}(f)$ and $\mathcal{I}(g)$ have disjoint ranges whenever f and g are distinct function symbols.

cet3 whenever x_i actually occurs in the term $\tau[\vec{x}\,]$ and $\vec{s} \in D^n$, it holds $s_i \neq \mathcal{I}(\tau)[\vec{s}\,]$.

4.2 Model semantics

The basic idea for the definition of our notion of Herbrand interpretation is to allow variables as elements in the domain of discourse. A term containing variables represents a set of elements whose structure is partially determined. Since in $\mathcal{T}_{\mathcal{F},\mathcal{V}}$ there are different terms that represent the same set, for example $f(x, y)$ and $f(v, w)$, it is adequate to consider $\mathcal{T}_{\mathcal{F},\mathcal{V}}$ modulo variable renaming. Hence, we define on $\mathcal{T}_{\mathcal{F},\mathcal{V}}$ an equivalence relation $\cong$ as follows:

$t \cong r$ if and only if there exist variable renamings ρ and σ such that $t = r\rho$ and $r = t\sigma$.

Let $\mathcal{T}_{\mathcal{F},\mathcal{V}}/\!\cong$ be the set of equivalence classes of $\mathcal{T}_{\mathcal{F},\mathcal{V}}$ with respect to the equivalence relation $\cong$. Assume $t \in \mathcal{T}_{\mathcal{F},\mathcal{V}}$ and $r \in \mathcal{T}_{\mathcal{F},\mathcal{V}}$. Then the relation $\leq$ on $\mathcal{T}_{\mathcal{F},\mathcal{V}}$ such that $t \leq r$ if and only if there exists a substitution θ such that $r \cong t\theta$ holds defines a partial order on $\mathcal{T}_{\mathcal{F},\mathcal{V}}$. It is clear that the order $\leq$ on $\mathcal{T}_{\mathcal{F},\mathcal{V}}$ induces an order relation $\leq$ on $\mathcal{T}_{\mathcal{F},\mathcal{V}}/\!\cong$.

A *Herbrand structure* $\mathcal{H}$ is a structure whose domain of discourse is the quotient set $\mathcal{T}_{\mathcal{F},\mathcal{V}}/\!\cong$. A *Herbrand base* $\mathcal{B}$ is the set of all formulae $q(\vec{t}\,)$ such that $q \in \mathrm{PRED}$ and $\vec{t} \in (\mathcal{T}_{\mathcal{F},\mathcal{V}})^n/\!\cong$. It is obvious that the order $\leq$ on $\mathcal{T}_{\mathcal{F},\mathcal{V}}/\!\cong$ induces an order

relation $\leq$ on $\mathcal{B}$. Usually, the notion of truth coincides with the one of being an element of; Herbrand interpretations are subsets of the Herbrand base. Since our Herbrand base does contain variables, this notion of truth is no longer correct. With respect to the ordering $\leq$ on $\mathcal{T}_{\mathcal{F},\mathcal{V}}$ and hence on $\mathcal{T}_{\mathcal{F},\mathcal{V}}/\cong$ two definitions of the notion of truth arise:

truth1 a formula $q(\vec{t})$ with $q \in \text{PRED}$ is *true* if there exists a formula $q(\vec{s})$ such that $q(\vec{s})$ is already true and $q(\vec{s}) \leq q(\vec{t})$ holds.

truth2 a formula $q(\vec{t})$ with $q \in \text{PRED}$ is *true* if there exists a variable renaming ρ such that $q(\vec{t})\rho$ is already true .

It is evident that the notion of truth with respect to **truth1** is more general than the notion of truth with respect to **truth2**. A detailed discussion of these two notions of truth with respect to a domain of discourse containing variables is given in [5]. We are interested in applying an appropriate kind of these notions of truth to logic programming. Without loss of generality, we do consider in the following the notion of truth according to **truth2**.

We now define the model relation $\models$ on a structure $\mathcal{M}$. We write $\not\models$ for the negation of $\models$. Let $|\mathcal{M}|$ denotes the domain of discourse of the structure $\mathcal{M}$. $\mathcal{M} \models \varphi$ is inductively defined on the structure of the formula φ as follows:

$$\mathcal{M} \models \epsilon. \tag{3}$$

$$\mathcal{M} \not\models \bot\ . \tag{4}$$

$$\mathcal{M} \models R(t_1, \ldots, t_n) \overset{\text{def}}{\iff} (\, t_1\rho, \ldots, t_n\rho\,) \in |\mathcal{M}|^n \text{ for some } \rho \in \mathcal{R}. \tag{5}$$

$$\mathcal{M} \models {\sim}R(t_1, \ldots, t_n) \overset{\text{def}}{\iff} (\, t_1\rho, \ldots, t_n\rho\,) \notin |\mathcal{M}|^n \text{ for all } \rho \in \mathcal{R}. \tag{6}$$

$$\mathcal{M} \models (\varphi \to \psi) \overset{\text{def}}{\iff} \text{if } \mathcal{M} \models \varphi, \text{ then } \mathcal{M} \models \psi. \tag{7}$$

$$\mathcal{M} \models (\varphi \wedge \psi) \overset{\text{def}}{\iff} \mathcal{M} \models \varphi \text{ and } \mathcal{M} \models \psi. \tag{8}$$

$$\mathcal{M} \models (\varphi \vee \psi) \overset{\text{def}}{\iff} \mathcal{M} \models \varphi \text{ or } \mathcal{M} \models \psi. \tag{9}$$

$$\mathcal{M} \models (\forall x\varphi) \overset{\text{def}}{\iff} \forall t \in |\mathcal{M}| \text{ it holds } \mathcal{M} \models \varphi\{\ t/x\ \}. \tag{10}$$

$$\mathcal{M} \models {\sim}\varphi \overset{\text{def}}{\iff} \mathcal{M} \not\models \varphi. \tag{11}$$

Notice that, since the universe of discourse may contain variables modulo renaming, the parentheses in rules (5) and (6) are necessary. To see this, assume $f(x) \cong f(h(1,y))$ and $g(x) \cong g(f(0,z))$. Then it does not hold that $(f(x), g(x)) \cong (f(h(1,y)), g(f(0,z)))$.

Let $\varphi = R(\vec{t})$ be a formula and $|\mathcal{M}|$ denote the domain of $\mathcal{M}$. A *positive instance* of φ is a formula $\psi = R(\vec{s})$ such that $\vec{s} \in |\mathcal{M}|^n$ and $\vec{t} \cong \vec{s}$. A *negative instance* of φ is a formula $\psi = R(\vec{s})$ such that $\vec{s} \notin |\mathcal{M}|^n$ and $\vec{t} \cong \vec{s}$. If φ is a formula, then an *instance* of φ is a formula ψ such that $\varphi \cong \psi$ and each formula of the form $R(\vec{s})$ occurring in ψ is a positive respectively negative instance of a corresponding formula $R(\vec{t})$ occurring in φ.

The structure $\mathcal{M}$ is a model of the completed program if and only if all the sentences of the completed program have truth value true in $\mathcal{M}$.

Let us now turn to our axiom system which will help recognize valid sentences of our first order language for predicate logic.

5 A natural inference system

Following Shoenfield in [11] one ensures that a sentence, that is also a theorem, of the axiom system is valid in a structure $\mathcal{I}$ when it is required that

(i) the axioms be valid in $\mathcal{I}$ and

(ii) the rules be such that the validity of the conclusion follows from that of the hypotheses.

Two kinds of axioms result from this requirement: the axioms that are valid because of the meaning of the logical symbols; let us call them *logical axioms*. The axioms, called *non-logical axioms*, that are valid because of particular properties of the structure.

The rules also divide into two classes: *logical rules*, that are rules in which the conclusion is valid because of the meaning of logical symbols. *Non-logical rules*, that are rules in which the conclusion is the consequence of the hypotheses only because of particular properties of the structure.

We only defined logical axioms and logical rules in the sequel. Let q be an n-ary predicate symbol of $\mathcal{L}$, $\vec{s}, \vec{r}$ and $\vec{t}$ be terms.

Logical axioms are:

- formulae of the form ϵ.
- formulae of the form $q(\vec{s})$.
- formulae of the form $\sim q(\vec{t})$.
- formulae of the form $t = t$.
- formulae of the form $t = s \rightarrow s = t$.
- formulae of the form $t = s \rightarrow s = r \rightarrow t = r$.
- formulae of the form $\vec{s} = \vec{t} \rightarrow f(\vec{s}) = f(\vec{t})$.
- formulae of the form $f(\vec{s}) = f(\vec{t}) \rightarrow \vec{s} = \vec{t}$.
- formulae of the form $f(\vec{s}) \neq g(\vec{t})$ if f and g are different.
- formulae of the form $x \neq t$ if t is different from x and $x \in vars(t)$.
- formulae of the form $(\vec{s} = \vec{t}) \wedge q(\vec{s}) \rightarrow q(\vec{t})$.
- formulae of the form $(\vec{s} = \vec{t}) \wedge \sim q(\vec{s}) \rightarrow \sim q(\vec{t})$.

Logical rules are:

$(\vee^+)$ $\frac{\varphi_i}{\varphi_1 \vee \varphi_2}(i = 1, 2)$ infer $\varphi_1 \vee \varphi_2$ from φ_i, where $i = 1, 2$.

$(\wedge^+)$ $\frac{\varphi \quad \psi}{\varphi \wedge \psi}$ infer $\varphi \wedge \psi$ from φ and ψ.

$(\exists^+)$ $\frac{\varphi[\vec{t}]}{\exists \vec{x}\, \varphi[\vec{x}]}(\vec{x} \notin \mathrm{vars}(\vec{t}))$ infer $\exists \vec{x}\, \varphi[\vec{x}]$ from $\varphi[\vec{t}]$, where $\vec{x} \notin \mathrm{vars}(\vec{t})$.

$(\forall^+)$ $\frac{\varphi[\vec{y}]}{\forall \vec{x}\, \varphi[\vec{x}]}(\vec{x} \notin \mathrm{vars}(\vec{y}))$ infer $\forall \vec{x}\, \varphi[\vec{x}]$ from $\varphi[\vec{y}]$, where $\vec{x} \notin \mathrm{vars}(\vec{y})$.

(cut) $\frac{\psi \vee \phi \quad \sim\phi \vee \varphi}{\psi \vee \varphi}$ infer $\psi \vee \varphi$ from $\psi \vee \phi$ and $\sim\phi \vee \varphi$.

for n-ary predicate symbols

(cl^+) $\frac{D_q[\vec{s}]}{q(\vec{s})}$ infer $q(\vec{s})$ from $D_q[\vec{s}]$.

(cl^-) $\frac{\sim D_q[\vec{s}]}{\sim q(\vec{s})}$ infer $\sim q(\vec{s})$ from $\sim D_q[\vec{s}]$.

We notice that in case $m = 0$ it holds with (cl^+) respectively (cl^-) that

(cl^+) $\frac{\epsilon}{q(\vec{s})}$ infer $q(\vec{s})$ from ϵ,

(cl^-) $\frac{\bot}{\sim q(\vec{s})}$ infer $\sim q(\vec{s})$ from $\bot$,

which are logical axioms.

One can now prove by simple induction that the rules of the inference system are sound. Let us denote in the sequel the inference system by $\mathit{inf}(\mathcal{L})$.

The inference system $\mathrm{inf}(\mathcal{L})$ implies a *derivation structure*, that is a graph structure, for a formula of the language $\mathcal{L}$. We assume that the notion of substructure, in this case subgraph, is defined as usual. A derivation substructure, say τ', of a derivation structure, say τ, is denoted by $\tau' \leq \tau$. We assume in the following simultaneous inductive definition of a derivation structure that τ_n respectively τ_m is a substructure of $\tau_{\max(n,m)+1}$. Independently from any formula of the language $\mathcal{L}$, we define a derivation structure by simultameous induction as follows:

1. any logical axiom builds a leaf of a derivation structure τ_0 at height 0.
2. let φ_i with $1 \leq i \leq m$ be premises of a logical rule and nodes of a derivation structure $\tau_{n(i)}$ at height $n(i) \geq 0$. Then the conclusion is a node of a derivation structure $\tau_{\max_{1 \leq i \leq m}(n(i))+1}$ at height $\max_{1 \leq i \leq m}(n(i)) + 1$.

One can construct a derivation structure which is finite or infinite. It is obvious from the construction of the derivation structure that for any $n, m \in \mathbb{N}$ if $\tau_m \leq \tau_n$ holds and τ_m has logical axioms as its leaves, then τ_n also has logical axioms as its leaves.

Definition 5.1 *A formula, say* Γ*, is* provable *in the inference system* $\mathit{inf}(\mathcal{L})$ *if there exists a finite derivation structure* τ *such that* Γ *is the root of the structure* τ *and the formulae at the leaves of the structure* τ *are logical axioms.*

By induction on the height of the derivation structure τ, considering an instance of a formula and using the remark mentioned above when proving the induction step from n to $n+1$, since it holds $\tau_n \leq \tau_{n+1}$, one proves the following lemma.

Lemma 5.1 *Let Γ be a formula and $\mathcal{M}$ be a model. If Γ is provable in the inference system inf($\mathcal{L}$), then it holds $\mathcal{M} \models \Gamma$.*

Let us next discuss the converse of this lemma.

Lemma 5.2 *Let Γ be a formula and $\mathcal{M}$ be a model. If it holds that $\mathcal{M} \models \Gamma$, then Γ is provable in the inference system inf($\mathcal{L}$).*

Proof by contradiction. Suppose Γ is not provable in inf($\mathcal{L}$). Then there exists no finite derivation structure τ such that Γ is the root of τ. Assume hence that the derivation structure τ is infinite. Then τ has a derivation substructure, say τ', such that for the root Γ' of τ' it holds that $\Gamma' \leq \Gamma$. That means there exists a substitution σ such that $\Gamma = \Gamma'\sigma$. Since $|\mathcal{M}|$ does contain variables modulo renaming, it holds with rule $(\forall^+)$ that Γ is provable in inf($\mathcal{L}$). This contradicts the hypothesis. Hence, Γ is not true in $\mathcal{M}$. □

The following theorem is an immediate consequence of lemma 5.1 and lemma 5.2.

Theorem 5.1 *Let Γ be a formula and $\mathcal{M}$ be a model. $\mathcal{M} \models \Gamma$ if and only if Γ is provable in the inference system inf($\mathcal{L}$).*

The semantics and the inference system necessary to formally determine the validity of a sentence of our language for first-order predicate logic is now fixed. Let us next discuss an operational method to realize it.

6 A bottom up definition of an extended SLDNF-resolution

Our aim in this section and with this extended definition is to eliminate the condition which states that a selected negative literal has to be closed in the well-known definition of the SLDNF-resolution as given in [3, 8, 9, 10, 14] respectively the assumption of non-floundering in the completeness theorem of SLDNF-resolution given in [4, 15]. The suggestion to work in this direction goes back to Kunen in [8] who supposes that one might get better completeness result by strengthening the SLDNF-resolution to compute more answers.

To ensure soundness it will be necessary to resolve positive literals before negative literals whenever the set of variables occurring in positive literals and the set of variables occurring in negative literals are not disjoint. This corresponds to a weak condition for the delay of negative literals.

As noted in the introduction, a goal respectively an SLDNF-tree flounders if the SLDNF-prefailed tree respectively the SLDNF-tree does contain negative literals in which variables do occur. Hence, we simply speak of a negative literal in

which variables do occur in the sequel. We first observe that all goals of the form $\leftarrow\sim q(\vec{t})$ with $\mathrm{vars}(\vec{t}) \neq \emptyset$ should not proceed clumsily. The examples in the introduction better illustrate this fact. We argue that the elimination of this restrictive condition is best done by a bottom up definition of the SLDNF-resolution. Since the SLDNF-resolution is constructed bottom up, the elimination of this restrictive condition is obvious and natural. This bottom up definition differs from that given by Kunen in [8] and refined in [3, 14] in two significant points:

(a) that a negative literal finitely fails is proven bottom up;

(b) in rules (R2) and (F1) it is not required that negative literals be ground.

Let P be a logic program. Like Kunen in [8] we define $\mathbf{Q}(P)$ to be the set of all goals with respect to P and $\mathbf{RES}(P)$ to be the set of all pairs $(\Gamma, \theta\restriction_{\mathrm{vars}(\Gamma)})$, where $\Gamma \in \mathbf{Q}(P)$ is a goal and $\theta\restriction_{\mathrm{vars}(\Gamma)}$ is an idempotent substitution acting on the variables occurring in Γ. Let furthermore $\mathbf{N}(P) \subseteq \mathbf{Q}(P)$ be the set of all goals which fail. Since $\mathbf{N}(P)$ and $\mathbf{RES}(P)$ are related and do contain more elements as needed, we inductively defined two subsets $\mathbf{R}(P)$ of $\mathbf{RES}(P)$ and $\mathbf{F}(P)$ of $\mathbf{N}(P)$. $\mathbf{F}(P)$ is a subset of $\mathbf{N}(P)$ of those goals that finitely fail. $\mathbf{R}(P)$ is a subset of $\mathbf{RES}(P)$ obtainable by SLDNF. We suppose in the sequel that $(\beta \leftarrow \Pi) \in P$ is a variant of a clause and that $\theta = \mathrm{mgu}(\beta, \lambda)$ also denotes the fact that λ and β are not unifiable as well. In case that λ and β are not unifiable it holds that $\theta = \sim\mathrm{mgu}(\beta, \lambda)$ and $\Pi\theta = \bot$ or $\sim\Pi\theta = \bot$. We recall that the formula (1), that is P_λ, resp. (2), that is $\bar{P}_\lambda$, is a compact representation of a set of formulae of the form $q(\vec{t}) \leftarrow \Pi$ resp. $\bar{q}(\vec{t}) \leftarrow \sim\Pi$. We write $\Gamma\mathbf{R}(P)\theta$ instead of $(\Gamma, \theta) \in \mathbf{R}(P)$. $\mathbf{R}(P)$ and $\mathbf{F}(P)$ are the least sets that satisfy the following closure properties:

(R0) $\epsilon\mathbf{R}(P)\varepsilon$.

(F0) $\bot \in \mathbf{F}(P)$.

(R1) If $(\Delta \wedge \lambda, \theta) \in \mathbf{RES}(P)$, λ is a positive literal, $(\beta \leftarrow \Pi) \in P_\lambda$, $\sigma = \mathrm{mgu}(\beta, \lambda\theta)$ and $(\Delta \wedge \Pi)\sigma\mathbf{R}(P)\vartheta$, then $(\Delta \wedge \lambda)\mathbf{R}(P)\sigma\vartheta$.

(R2) If $\Delta\mathbf{R}(P)\theta$, λ is a negative literal and for each $(\beta \leftarrow \Pi) \in \bar{P}_\lambda$ it holds that $\sim\Pi\mathrm{mgu}(\beta, \lambda\theta) \in \mathbf{F}(P)$, then $(\Delta \wedge \lambda)\mathbf{R}(P)\theta$.

(F1) If $(\sim\Delta, \theta) \in \mathbf{RES}(P)$, λ is a negative literal and for each $(\beta \leftarrow \Pi) \in \bar{P}_\lambda$ it holds that $\sim\Pi\mathrm{mgu}(\beta, \lambda\theta) \in \mathbf{F}(P)$, then $(\sim\Delta \vee \lambda) \in \mathbf{F}(P)$.

(F2) If $(\sim\Delta, \theta) \in \mathbf{RES}(P)$, λ is a positive literal and for some $(\beta \leftarrow \Pi) \in P_\lambda$ it holds that $\Pi\mathrm{mgu}(\beta, \lambda\theta)\mathbf{R}(P)\vartheta$ with $\mathrm{mgu}(\beta, \lambda\theta)\vartheta$ a variable renaming of λ, then $(\sim\Delta \vee \lambda) \in \mathbf{F}(P)$.

Let P be a program and Γ be a goal. A substitution θ is a *computed answer substitution* of $P \cup \{\ \Gamma\ \}$ if and only if $\Gamma\mathbf{R}(P)\theta$ holds. We write $P \vdash \forall\Gamma\theta$ if $\Gamma\mathbf{R}(P)\theta$ holds, and $P \vdash \forall \sim\Gamma$ if $\Gamma \in \mathbf{F}(P)$ holds.

Before discussing the soudness of our SLDNF-resolution let us first state an useful property of the sets $\mathbf{R}(P)$ and $\mathbf{F}(P)$.

Lemma 6.1 *The sets* $\mathbf{R}(P)$ *and* $\mathbf{F}(P)$ *are closed under substitutions.*

We better prove the generalization of this Lemma in the next section.

Theorem 6.1 (Soundness) *Let* P *be a program,* Γ *be a goal and* θ *be a substitution.*

(i) If $P \vdash \forall\Gamma\theta$, *then* $pcomp(P) \models \Gamma\theta$.

(ii) If $P \vdash \forall \sim\Gamma$, *then* $pcomp(P) \models \sim\Gamma$.

Proof: by simultaneous induction on the definition of $\mathbf{R}(P)$ and $\mathbf{F}(P)$. Notice that a positive literal λ is resolved before a negative literal L whenever $\mathrm{vars}(\lambda) \cap \mathrm{vars}(L) \neq \emptyset$ holds.

The cases (R0) and (F0) are evident. Let $\Gamma = \Delta$ not be an empty goal or the falsehood and λ be a literal in the sequel. Remember that $\mathrm{mgu}(\lambda, \beta)$ is $\perp$ if λ and β are not unifiable.

`Case (R1):` suppose now that $(\Delta \wedge \lambda, \sigma) \in \mathbf{RES}(P)$, that λ is a positive literal such that $\lambda\sigma$ is of the form $p(\vec{s})$ with p an n-ary predicate symbol, and that $\beta \leftarrow \Pi$ is a variant of a clause in P such that p occurs in the head β. Suppose further that from $\mathrm{pcomp}(P)$ we have

$$p(x_1, \ldots, x_n) \leftarrow \bigvee_{i=1}^{m} \exists\vec{y}\Big(\bigwedge_{j=1}^{n}(x_j \cong t_{i,j}[\vec{y}]) \wedge \bigwedge_{l=1}^{k(i)} \lambda_{i,l}[\vec{y}]\Big),$$

and that for some $1 \leq i \leq m$, such that $\theta = \mathrm{mgu}(p(\vec{s}), p(\vec{x}))$, it holds that

$$\Big(\Delta \wedge \bigwedge_{j=1}^{n}(s_j \cong t_{i,j}[\vec{y}]) \wedge \lambda_{i,1}[\vec{y}] \wedge \cdots \wedge \lambda_{i,k(i)}[\vec{y}]\Big)\theta\mathbf{R}(P)\chi.$$

Then from the induction hypothesis we have

$$\mathrm{pcomp}(P) \models \Big(\Delta \wedge \bigwedge_{j=1}^{n}(s_j \cong t_{i,j}[\vec{y}]) \wedge \lambda_{i,1}[\vec{y}] \wedge \cdots \wedge \lambda_{i,k(i)}[\vec{y}]\Big)\theta\chi.$$

Then with (8) in subsection 4.2 we have

$$\mathrm{pcomp}(P) \models \Big(\bigwedge_{j=1}^{n}(s_j \cong t_{i,j}[\vec{y}]) \wedge \lambda_{i,1}[\vec{y}] \wedge \cdots \wedge \lambda_{i,k(i)}[\vec{y}]\Big)\theta\chi$$

and

$$\mathrm{pcomp}(P) \models \Delta\theta\chi.$$

Then with (7) in subsection 4.2 we have

$$\mathrm{pcomp}(P) \models \Big(\bigwedge_{j=1}^{n}(s_j \cong t_{i,j}[\vec{y}]) \wedge \lambda_{i,1}[\vec{y}] \wedge \cdots \wedge \lambda_{i,k(i)}[\vec{y}]\Big)\theta\chi \rightarrow \lambda\sigma\theta\chi.$$

Let then $\vartheta = (\sigma\theta\chi)\restriction_{\mathrm{vars}(\Delta\wedge\lambda)}$. Then it holds $\mathrm{pcomp}(P) \models \lambda\vartheta$. It also holds that $\Delta\mathbf{R}(P)\vartheta$ and therefore $\mathrm{pcomp}(P) \models \Delta\vartheta$. Then with (8) in subsection 4.2 we have $\mathrm{pcomp}(P) \models (\Delta \wedge \lambda)\vartheta$.

`Case (R2):` suppose that Δ is a goal such that for some substitution χ it holds $\Delta\mathbf{R}(P)\chi$ and that λ is a negative literal such that $\lambda\chi$ is of the form $\sim p(\vec{s})$. From $\mathrm{pcomp}(P)$ we have

$$\bar{p}(x_1, \ldots, x_n) \leftarrow \bigwedge_{i=1}^{m} \forall\vec{y}\Big(\bigvee_{j=1}^{n} \sim(x_j \cong t_{i,j}[\vec{y}]) \vee \bigvee_{l=1}^{k(i)} \sim\lambda_{i,l}[\vec{y}]\Big).$$

Suppose further that for each $1 \leq i \leq m$ it holds

$$\Big(\bigvee_{j=1}^{n} \sim(x_j \cong t_{i,j}[\vec{y}]) \vee \sim\lambda_{i,1}[\vec{y}] \vee \cdots \vee \sim\lambda_{i,k(i)}[\vec{y}]\Big)\mathrm{mgu}(\bar{p}(\vec{x}), \bar{p}(\vec{s})) \in \mathbf{F}(P).$$

Then with the induction hypothesis it holds for each $1 \leq i \leq m$ that

$$\mathrm{pcomp}(P) \models \sim\Big(\lambda_{i,1}[\vec{y}] \wedge \cdots \wedge \lambda_{i,k(i)}[\vec{y}]\Big)\mathrm{mgu}(\bar{p}(\vec{x}), \bar{p}(\vec{s})),$$

and that $\mathrm{pcomp}(P) \models \Delta\chi$ since $\Delta\mathbf{R}(P)\chi$ holds. Then it holds with (7) in subsection 4.2 for each $1 \leq i \leq m$ that

$$\mathrm{pcomp}(P) \models \Big(\sim p(\vec{s}) \leftarrow \sim(\bigwedge_{l=1}^{k(i)} \lambda_{i,l})\Big)$$

and $\mathrm{pcomp}(P) \models \Delta\chi$. Then it holds that $\mathrm{pcomp}(P) \models \lambda\chi$. Then with (8) in subsection 4.2 it holds that $\mathrm{pcomp}(P) \models (\Delta \wedge \lambda)\chi$.

`Case (F1)`: suppose that $(\sim\Delta, \theta) \in \mathbf{RES}(P)$ is a goal and $\lambda\theta$ is a negative literal of the form $\bar{p}(\vec{s})$. From $\mathrm{pcomp}(P)$ we have

$$\bar{p}(x_1, \ldots, x_n) \leftarrow \bigwedge_{i=1}^{m} \forall\vec{y}\Big(\bigvee_{j=1}^{n} \sim(x_j \cong t_{i,j}[\vec{y}]) \vee \bigvee_{l=1}^{k(i)} \sim\lambda_{i,l}[\vec{y}]\Big).$$

Suppose further that for each $1 \leq i \leq m$ it holds

$$\Big(\bigvee_{j=1}^{n} \sim(x_j \cong t_{i,j}[\vec{y}]) \vee \sim\lambda_{i,1}[\vec{y}] \vee \cdots \vee \sim\lambda_{i,k(i)}[\vec{y}]\Big)\mathrm{mgu}(\bar{p}(\vec{x}), \bar{p}(\vec{s})) \in \mathbf{F}(P).$$

Then with the induction hypothesis it holds for each $1 \leq i \leq m$ that

$$\mathrm{pcomp}(P) \models \sim\Big(\lambda_{i,1}[\vec{y}] \wedge \cdots \wedge \lambda_{i,k(i)}[\vec{y}]\Big)\mathrm{mgu}(\bar{p}(\vec{x}), \bar{p}(\vec{s})).$$

Then it holds with (7) in subsection 4.2 for each $1 \leq i \leq m$ that

$$\mathrm{pcomp}(P) \models \Big(\sim p(\vec{s}) \leftarrow \sim(\bigwedge_{l=1}^{k(i)} \lambda_{i,l})\Big).$$

Then it holds that $\mathrm{pcomp}(P) \models \sim\lambda$. Then it holds with (9) in subsection 4.2 that $\mathrm{pcomp}(P) \models \sim(\Delta \wedge \lambda)$.

`Case (F2)`: suppose that $(\sim\Delta, \theta) \in \mathbf{RES}(P)$ is a goal and $\lambda\theta$ is a positive literal of the form $p(\vec{s})$. From $\mathrm{pcomp}(P)$ we have

$$p(x_1, \ldots, x_n) \leftarrow \bigvee_{i=1}^{m} \exists\vec{y}\Big(\bigwedge_{j=1}^{n}(x_j \cong t_{i,j}[\vec{y}]) \wedge \bigwedge_{l=1}^{k(i)} \lambda_{i,l}[\vec{y}]\Big).$$

Suppose further that for some $1 \leq i \leq m$ it holds

$$\Big(\bigwedge_{j=1}^{n}(x_j \cong t_{i,j}[\vec{y}]) \wedge \lambda_{i,1}[\vec{y}] \wedge \cdots \wedge \lambda_{i,k(i)}[\vec{y}]\Big)\mathrm{mgu}(p(\vec{x}), p(\vec{s}))\mathbf{R}(P)\chi$$

and $\mathrm{mgu}(p(\vec{x}), p(\vec{s}))\chi$ is a variable renaming of λ. Then with the induction hypothesis it holds for some $1 \leq i \leq m$ that

$$\mathrm{pcomp}(P) \models \Big(\lambda_{i,1}[\vec{y}] \wedge \cdots \wedge \lambda_{i,k(i)}[\vec{y}]\Big)\mathrm{mgu}(p(\vec{x}), p(\vec{s}))\chi.$$

Then it holds with (7) in subsection 4.2 that

$$\mathrm{pcomp}(P) \models \Big(p(\vec{s}) \leftarrow \bigwedge_{l=1}^{k(i)} \lambda_{i,l}\Big).$$

Since $\sim\lambda = p(\vec{s})$ and $p(\vec{s}) = p(\vec{s})\text{mgu}(p(\vec{x}), p(\vec{s}))\chi$ it follows that $\text{pcomp}(P) \models \sim\lambda$ and hence $\text{pcomp}(P) \models \sim(\Delta \wedge \lambda)$ using (9) in subsection 4.2. □

In the introduction we briefly explain what we mean by a *well-formulated* program. To simply understand the reason for this restriction of the class of program, let us reconsider the program `Factorial` and replace clause C_8 with the following clause

$$C_{8'}: \quad \texttt{factorial}(s(n), m) \leftarrow \texttt{factorial}(n, p), \texttt{times}(s(n), p, m).$$

It is obvious that the SLDNF-resolution of the so formulated program `Factorial` with the goal $\bot\leftarrow$ `factorial`$(v, 0)$ or $\bot\leftarrow \sim$ `factorial`$(v, 0)$ neither succeeds nor finitely fails. But from the declarative semantics it is obvious that $\bot\leftarrow$ `factorial`$(v, 0)$ is not a logical consequence of `Factorial` and $\bot\leftarrow \sim$ `factorial`$(v, 0)$ is a logical consequence of `Factorial`.

Definition 6.1 *A program is* well-formulated *if the procedural semantics finds an answer substitution for that program with a goal whenever such an answer substitution does exists with respect to the declarative semantics.*

Note that the fact that a program is well-formulated does not imply that the SLDNF-resolution of this program with any goal terminates. To be convinced of this claim, just reconsider the program `Factorial` and replace clause C_6 with the following clause

$$C_{6'}: \quad \texttt{times}(s(n), m, p) \leftarrow \texttt{times}(n, m, w), \texttt{plus}(m, w, p).$$

It is obvious that the program `Times` is well-formulated. But the SLDNF-resolution of this program with the goal $\bot\leftarrow$ `times`$(v, w, 0)$ does not terminate, since clause $C_{6'}$ is always applicable.

We now move forward and discuss the completeness result of our SLDNF-resolution.

7 Completeness result

The objective of this section is obvious, namely the proof of the completeness of the SLDNF-resolution for the class of arbitrary programs with an arbitrary goal. To do this, we first of all generalize the definition of $\mathbf{R}(P)$ to $\mathbf{Y}(P) \subseteq \mathbf{RES}(P)$ which is the set of all pairs $(\Gamma, \theta_{\restriction_{\text{vars}(\Gamma)}})$ such that $\Gamma\theta$ is true. The definition of the sets $\mathbf{Y}(P)$ and $\mathbf{N}(P)$ is similar to that of $\mathbf{YES}(P)$ and $\mathbf{NO}(P)$ given by Stärk in [13]. We then define an universal search structure using a technique which goes back to Hintikka, Beth and Schütte, and is described in [10], and prove some useful properties of this search structure. Next, we establish a relationship between the derivation structure of the inference system $\text{inf}(\mathcal{L})$ defined in section 5 and the universal search structure. The completeness result follows then immediately from the relationship between these two structures.

Let us write (Γ, θ) in the sequel and mean $(\Gamma, \theta_{\restriction_{\text{vars}(\Gamma)}})$ with θ idempotent. $\mathbf{Y}(P)$ and $\mathbf{N}(P)$ are the least sets that satisfy the following closure properties:

(Y0) $(\epsilon, \varepsilon) \in \mathbf{Y}(P)$.

(N0) $\bot \in \mathbf{N}(P)$.

(Y1) If $(\Delta \wedge \lambda, \theta) \in \mathbf{RES}(P)$, λ is a positive literal, $(\beta \leftarrow \Pi) \in P_\lambda$, σ is a substitution such that $\beta\sigma \cong (\lambda\theta)\sigma$ and $(\Delta \wedge \Pi, \sigma\chi) \in \mathbf{Y}(P)$, then $(\Delta \wedge \lambda, \theta\sigma\chi) \in \mathbf{Y}(P)$.

(Y2) If $(\Delta, \chi) \in \mathbf{Y}(P)$, λ is a negative literal, for all clauses $(\beta \leftarrow \Pi) \in \bar{P}_\lambda$, for all substitutions θ and σ it holds $\beta\sigma \cong (\lambda\chi)\theta$ and $\sim\Pi\sigma \in \mathbf{N}(P)$, then $(\Delta \wedge \lambda, \chi) \in \mathbf{Y}(P)$.

(N1) If $(\sim\Delta, \vartheta) \in \mathbf{RES}(P)$, λ is a negative literal, for all clauses $(\beta \leftarrow \Pi) \in \bar{P}_\lambda$, for all substitutions θ and σ it holds $\beta\sigma \cong \lambda\vartheta\theta$ and $\sim\Pi\sigma \in \mathbf{N}(P)$, then $(\sim\Delta \vee \lambda) \in \mathbf{N}(P)$.

(N2) If $(\sim\Delta, \vartheta) \in \mathbf{RES}(P)$, λ is a positive literal, for some clause $(\beta \leftarrow \Pi) \in P_\lambda$, for some substitutions σ and θ it holds $\beta\sigma \cong \lambda\vartheta\theta$ and $(\Pi, \sigma\chi) \in \mathbf{Y}(P)$, then $(\sim\Delta \vee \lambda) \in \mathbf{N}(P)$.

We first discuss an useful property of the sets $\mathbf{Y}(P)$ and $\mathbf{N}(P)$. It holds that $\mathbf{R}(P) \subseteq \mathbf{Y}(P)$ and $\mathbf{F}(P) \subseteq \mathbf{N}(P)$. We sketch the proof of this claim by example. For this reconsider the program `Factorial` given in the introduction; and replace clause C_8 by

$$C_{8'}: \ \text{factorial}(s(x), z) \leftarrow \text{factorial}(x, y), \text{times}(s(x), y, z).$$

It is obvious that $\texttt{factorial}(v, 0) \in \mathbf{N}(Factorial)$. Hence, $\sim\texttt{factorial}(v, 0) \in \mathbf{Y}(Factorial)$. But it does not hold that $\texttt{factorial}(v, 0) \in \mathbf{F}(Factorial)$, since the SLDNF-resolution of $\texttt{Factorial} \cup \{\leftarrow \texttt{factorial}(v, 0)\}$ neither succeeds nor finitely fails. Hence, $(\sim\texttt{factorial}(v, 0), \varepsilon) \notin \mathbf{R}(Factorial)$.

The following Lemma is a generalization of Lemma 6.1 stated in the previous section.

Lemma 7.1 *The sets* $\mathbf{Y}(P)$ *and* $\mathbf{N}(P)$ *are closed under substitutions.*

Proof by simultaneous induction on the definition of $\mathbf{Y}(P)$ and $\mathbf{N}(P)$.

The cases (Y0) and (N0) are trivial.

`Case (Y1):` Let $(\Delta \wedge \lambda, \chi) \in \mathbf{Y}(P)$ with λ a positive literal and σ be a substitution. Then there exists a clause $(\beta \leftarrow \Pi) \in P$ such that $(\Pi, \chi) \in \mathbf{Y}(P)$. Since it holds that $(\Delta, \chi) \in \mathbf{Y}(P)$, it follows with the induction hypothesis that $(\Delta, \chi\sigma) \in \mathbf{Y}(P)$ and $(\Pi, \chi\sigma) \in \mathbf{Y}(P)$. Hence, it holds $(\Delta \wedge \lambda, \chi\sigma) \in \mathbf{Y}(P)$.

`Case (Y2):` Let $(\Delta \wedge \lambda, \chi) \in \mathbf{Y}(P)$ with λ a negative literal and σ be a substitution. For all clauses $(\beta \leftarrow \Pi) \in P$ and for all substitutions θ, ϑ such that $\beta\theta \cong \lambda\vartheta$ it holds that $\sim\Pi\theta \in \mathbf{N}(P)$. From the induction hypothesis it holds that $\sim\Pi\theta\sigma \in \mathbf{N}(P)$, and hence $\lambda\vartheta\sigma \in \mathbf{N}(P)$. Since it holds with the induction hypothesis that $(\Delta, \chi\sigma) \in \mathbf{Y}(P)$ it also holds that $(\Delta \wedge \lambda, \chi\sigma) \in \mathbf{Y}(P)$.

`Case (N1):` Let $\Delta \wedge \lambda \in \mathbf{N}(P)$ with λ a negative literal and σ be a substitution. For all clauses $(\beta \leftarrow \Pi) \in P$ and for all substitutions θ, ϑ such that $\beta\theta \cong \lambda\vartheta$ it

holds $\sim\Pi\theta \in \mathbf{N}(P)$. From the induction hypothesis it holds that $\sim\Pi\theta\sigma \in \mathbf{N}(P)$, and hence $\lambda\vartheta\sigma \in \mathbf{N}(P)$. Hence, it holds $\Delta\sigma \vee \lambda\sigma \in \mathbf{N}(P)$.

`Case (N2):` Let $\Delta \wedge \lambda \in \mathbf{N}(P)$ with λ a positive literal and σ be a substitution. For some clause $(\beta \leftarrow \Pi) \in P$ and for some substitution θ such that $\beta\theta \cong \lambda\theta$ it holds $(\Pi, \theta\chi) \in \mathbf{Y}(P)$. From the induction hypothesis it holds that $(\Pi, \theta\chi\sigma) \in \mathbf{Y}(P)$. Hence, it holds that $\Delta\sigma \vee \lambda\sigma \in \mathbf{N}(P)$. □

Like in section 5 we construct an universal search structure $\mathcal{U}$ that does not depend on a given formula Γ. The idea is based on a method of Beth, Hintikka and Schütte which is described in [10]. It will then follows from this universal search structure that for a given formula Γ and a well-formulated program P it holds that for all Herbrand structures either $P \vdash \forall\Gamma\theta$ for some substitution θ or $P \vdash \forall\ \sim\Gamma$.

Note that $\perp$ in the sequel also denotes equations which are not unifiable. Let E^Y respectively E^N denote a set of equations which are obtained when for a goal Γ it holds that $(\Gamma, \mathrm{mgu}(E)) \in \mathbf{Y}(P)$ respectively $\Gamma \in \mathbf{N}(P)$, whereby E is defined to be $E^Y \uplus E^N$, that means $E = E^Y \cup E^N$ with $E^Y \cap E^N = \emptyset$. Let E_1 and E_2 be two such sets of equations. Then we define $E_1 \cup E_2$ to be

$$(E_1^Y \cup E_2^Y) \uplus (E_1^N \cup E_2^N),$$

$E_1 \subseteq E_2$ to hold if

$$E_1^Y \subseteq E_2^Y \ \text{ or } \ E_1^N \subseteq E_2^N,$$

and $\mathrm{mgu}(E)$ to be

$$\mathrm{mgu}(E^Y) \uplus \mathrm{mgu}(E^N).$$

For the constuction of the universal search structure we assume a countably infinite set of formulae Γ, substitutions θ and equations E of the underlying language such that

(i) $\theta = \mathrm{mgu}(E)$ and

(ii) either $(\Gamma, \theta) \in \mathbf{Y}(P)$ or $\Gamma \in \mathbf{N}(P)$.

Let us denote the triple formula, equation and substitution by η, that means $\eta = \langle \Gamma, E, \theta \rangle$. η is then a node of the universal search structure. We write $(\beta \leftarrow \Pi)_i = \beta_i \leftarrow \Pi_i$ for the i-th clause $\beta \leftarrow \Pi$ with $i \geq 0$ and $\beta \doteq \lambda$ as a shorthand for the set of equations resulting from the unification of β with λ. By induction on the height n of the universal search structure $\mathcal{U}$ we define the nodes as follows:

`case` $n = 0$: $\eta_{0,j} = \langle \epsilon, \emptyset, \varepsilon \rangle$ and $\eta_{0,j'} = \langle \perp, E, \perp \rangle$ for all $j, j' \geq 0$ with $j \neq j'$. It is evident that in case $\eta_{0,j} = \langle \epsilon, \emptyset, \varepsilon \rangle$ it holds $(\epsilon, \varepsilon) \in \mathbf{Y}(P)$, and $\varepsilon = \mathrm{mgu}(\emptyset)$, and that in case $\eta_{0,j'} = \langle \perp, E, \perp \rangle$ we have $\perp = \sim\mathrm{mgu}(E)$ and $\perp \in \mathbf{N}(P)$.

`Case` $n > 0$: let λ be a literal,

$$(\beta \leftarrow \Pi)_i \in (P_\lambda \uplus \bar{P}_\lambda)$$

with $1 \leq i \leq m$ be m clauses such that the same k-ary predicate symbol occurs in λ and β_i and

$$\eta_{n,j} = \langle \nabla_{n,j}, E_{n,j}, \theta_{n,j} \rangle$$

be a node with $j \geq 0$, where $\nabla_{n,j}$ stands for either $\Gamma_{n,j}$ or $\sim\Gamma_{n,j}$.

(a) Assume that $(\nabla_{n,j} = \Gamma_{n,j} \wedge \lambda, \theta_{n,j}) \in \mathbf{RES}(P)$, and that λ is a positive literal. If for some $1 \leq i \leq m$ it holds that $(\Gamma_{n,j} \wedge \Pi_i)\mathrm{mgu}(\beta_i, \lambda\theta_{n,j}) \in \mathbf{Y}(P)$, that means there exists some $n(i)$ such that

$$\eta_{n(i),j(i)} = \langle \begin{array}{l} (\Gamma_{n,j} \wedge \Pi_i)\mathrm{mgu}(\beta_i, \lambda\theta_{n,j}), \\ E_{n(i),j(i)}, \\ \theta_{n(i),j(i)} \end{array} \rangle,$$

then

$$\eta_{\max_{1\leq i\leq m}(n,n(i))+1,j} = \langle \begin{array}{l} \Gamma_{n,j} \wedge \lambda, \\ E_{n,j} \cup E_{n(i),j(i)} \cup \{\beta_i \doteq \lambda\theta_{n,j}\}, \\ \theta_{\max_{1\leq i\leq m}(n,n(i))+1,j} \end{array} \rangle.$$

If for each $1 \leq i \leq m$ it holds that

$$(\Gamma_{n,j} \wedge \Pi_i)\mathrm{mgu}(\beta_i, \lambda\theta_{n,j}) \in \mathbf{N}(P),$$

that means there exists $n(i)$ such that

$$\eta_{n(i),j(i)} = \langle \begin{array}{l} (\Gamma_{n,j} \wedge \Pi_i)\mathrm{mgu}(\beta_i, \lambda\theta_{n,j}), \\ E_{n(i),j(i)}, \\ \theta_{n(i),j(i)} \end{array} \rangle,$$

then

$$\eta_{\max_{1\leq i\leq m}(n,n(i))+1,j} = \langle \begin{array}{l} \Gamma_{n,j} \wedge \lambda, \\ E_{n,j} \cup \bigcup_{i=1}^{m}(E_{n(i),j(i)} \cup \{\beta_i \doteq \lambda\theta_{n,j}\}), \\ \theta_{\max_{1\leq i\leq m}(n,n(i))+1,j} \end{array} \rangle.$$

(b) Asume that $(\nabla_{n,j} = \Gamma_{n,j}, \theta_{n,j}) \in \mathbf{Y}(P)$, that all positive literals which have had variables in common with the literal λ do occur in $\Gamma_{n,j}$ and that λ is a negative literal. If for each $1 \leq i \leq m$ it holds that

$$\sim\Pi_i\mathrm{mgu}(\beta_i, \lambda\theta_{n,j}) \in \mathbf{N}(P),$$

that means there exists $n(i)$ such that

$$\eta_{n(i),j(i)} = \langle \begin{array}{l} \sim\Pi_i\mathrm{mgu}(\beta_i, \lambda\theta_{n,j}), \\ E_{n(i),j(i)}, \\ \theta_{n(i),j(i)} \end{array} \rangle,$$

then

$$\eta_{\max_{1\leq i\leq m}(n,n(i))+1,j} = \langle \begin{array}{l} \Gamma_{n,j} \wedge \lambda, \\ E_{n,j} \cup \bigcup_{i=1}^{m}(E_{n(i),j(i)} \cup \{\beta_i \doteq \lambda\theta_{n,j}\}), \\ \theta_{\max_{1\leq i\leq m}(n,n(i))+1,j} \end{array} \rangle.$$

If for some $1 \leq i \leq m$ it holds

$$\sim\Pi_i\mathrm{mgu}(\beta_i, \lambda\theta_{n,j}) \in \mathbf{Y}(P),$$

that means there exists $n(i)$ such that

$$\eta_{n(i),j(i)} = \langle \begin{array}{l} \sim\Pi_i \mathrm{mgu}(\beta_i, \lambda\theta_{n,j}), \\ E_{n(i),j(i)}, \\ \theta_{n(i),j(i)} \end{array} \rangle,$$

then

$$\eta_{\max_{1\leq i\leq m}(n,n(i))+1,j} = \langle \begin{array}{l} \Gamma_{n,j} \wedge \lambda, \\ E_{n,j} \cup E_{n(i),j(i)} \cup \{\beta_i \doteq \lambda\theta_{n,j}\}, \\ \theta_{\max_{1\leq i\leq m}(n,n(i))+1,j} \end{array} \rangle.$$

(c) Asume that $(\nabla_{n,j}, \theta_{n,j}) \in \mathbf{RES}(P)$ and that λ is a negative literal. If for each $1 \leq i \leq m$ it holds that

$$\sim\Pi_i \mathrm{mgu}(\beta_i, \lambda\theta_{n,j}) \in \mathbf{N}(P),$$

that means there exists $n(i)$ such that

$$\eta_{n(i),j(i)} = \langle \begin{array}{l} \sim\Pi_i \mathrm{mgu}(\beta_i, \lambda\theta_{n,j}), \\ E_{n(i),j(i)}, \\ \theta_{n(i),j(i)} \end{array} \rangle,$$

then

$$\eta_{\max_{1\leq i\leq m}(n,n(i))+1,j} = \langle \begin{array}{l} \nabla_{n,j} \vee \lambda, \\ E \cup \bigcup_{i=1}^{m}(E_{n(i),j(i)} \cup \{\beta_i \doteq \lambda\theta_{n,j}\}), \\ \mathrm{mgu}(E_{\max_{1\leq i\leq m}(n,n(i))+1,j}) \end{array} \rangle.$$

If for some $1 \leq i \leq m$ it holds that

$$\sim\Pi_i \mathrm{mgu}(\beta_i, \lambda\theta_{n,j}) \in \mathbf{Y}(P),$$

that means there exists $n(i)$ such that

$$\eta_{n(i),j(i)} = \langle \begin{array}{l} \sim\Pi_i \mathrm{mgu}(\beta_i, \lambda\theta_{n,j}), \\ E_{n(i),j(i)}, \\ \theta_{n(i),j(i)} \end{array} \rangle,$$

then

$$\eta_{\max_{1\leq i\leq m}(n,n(i))+1,j} = \langle \begin{array}{l} \nabla_{n,j} \vee \lambda, \\ E \cup E_{n(i),j(i)} \cup \{\beta_i \doteq \lambda\theta_{n,j}\}, \\ \mathrm{mgu}(E_{\max_{1\leq i\leq m}(n,n(i))+1,j}) \end{array} \rangle.$$

(d) Assume that $(\nabla_{n,j}, \theta_{n,j}) \in \mathbf{RES}(P)$ and that λ is a positive literal. If for some $1 \leq i \leq m$ it holds that

$$\Pi_i \mathrm{mgu}(\beta_i, \lambda\theta_{n,j}) \in \mathbf{Y}(P),$$

that means there exists some $n(i)$ such that

$$\eta_{n(i),j(i)} = \langle \begin{array}{l} \Pi_i \mathrm{mgu}(\beta_i, \lambda\theta_{n,j}), \\ E_{n(i),j(i)}, \\ \theta_{n(i),j(i)} \end{array} \rangle,$$

then

$$\eta_{\max_{1\leq i\leq m}(n,n(i))+1,j} = \langle \begin{array}{l} \nabla_{n,j} \vee \lambda, \\ E \cup \{\beta_i \doteq \lambda\theta_{n,j}\} \cup E_{n(i),j(i)}, \\ \mathrm{mgu}(E_{\max_{1\leq i\leq m}(n,n(i))+1,j}) \end{array} \rangle.$$

If for each $1 \leq i \leq m$ it holds that

$$\Pi_i \mathrm{mgu}(\beta_i, \lambda\theta_{n,j}) \in \mathbf{N}(P),$$

that means there exists $n(i)$ such that

$$\eta_{n(i),j(i)} = \langle \begin{array}{l} \Pi_i \mathrm{mgu}(\beta_i, \lambda\theta_{n,j}), \\ E_{n(i),j(i)}, \\ \theta_{n(i),j(i)} \end{array} \rangle,$$

then

$$\eta_{\max_{1\leq i\leq m}(n,n(i))+1,j} = \langle \begin{array}{l} \nabla_{n,j} \vee \lambda, \\ E \cup \bigcup_{i=1}^{m}(E_{n(i),j(i)} \cup \{\beta_i \doteq \lambda\theta_{n,j}\}), \\ \mathrm{mgu}(E_{\max_{1\leq i\leq m}(n,n(i))+1,j}) \end{array} \rangle.$$

It follows immediately from the construction that for each $j \geq 0$ we have $\theta_{n+1,j} = \mathrm{mgu}(E_{n+1,j})$. We now prove that for each $j \geq 0$ it holds that either $(\Gamma_{n+1,j}, \theta_{n+1,j}) \in \mathbf{Y}(P)$ or $\Gamma_{n+1,j} \in \mathbf{N}(P)$. Let us then consider an j be given.

Suppose from the induction hypothesis that $(\Gamma_{n,j}, \theta_{n,j})$ is in $\mathbf{RES}(P)$. If for some n' and j' we have

$$(\Gamma'_{n',j'}, \theta_{n',j'}) = (\Gamma_{n,j} \wedge \Pi_i, \mathrm{mgu}(\beta_i, \lambda\theta_{n,j})\chi) \in \mathbf{Y}(P)$$

with $1 \leq i \leq m$ and λ is a positive literal, then it holds with (Y1) that

$$(\Gamma_{\max(n,n')+1,j}, \theta_{\max(n,n')+1,j}) = (\Gamma_{n,j} \wedge \lambda, \theta_{n,j}\mathrm{mgu}(\beta_i, \lambda\theta_{n,j})\chi) \in \mathbf{Y}(P).$$

If $(\Gamma_{n,j}, \theta_{n,j})$ is in $\mathbf{Y}(P)$, all positive literals which have had variables in common with the literal λ occur in $\Gamma_{n,j}$, and for all $1 \leq i \leq m$ we have $n'(i)$ and $j'(i)$ such that

$$\Gamma'_{n'(i),j'(i)} = \sim\Pi_i \mathrm{mgu}(\beta_i, \lambda\theta_{n,j})) \in \mathbf{N}(P)$$

and λ is a negative literal, then it holds with (Y2) that

$$(\Gamma_{\max_{1\leq i\leq m}(n,n'(i))+1,j}, \theta_{n,j}) = (\Gamma_{n,j} \wedge \lambda, \theta_{n,j}) \in \mathbf{Y}(P).$$

Suppose now that $\langle \nabla_{n,j}, E_{n,j}, \theta_{n,j}\rangle$ is a node. If for all $1 \leq i \leq m$ we have $n'(i)$ and $j'(i)$ such that

$$\Gamma'_{n'(i),j'(i)} = \sim\Pi_i \mathrm{mgu}(\beta_i, \lambda\theta_{n,j}) \in \mathbf{N}(P)$$

and λ is a negative literal, then it holds with (N1) that

$$\Gamma_{\max_{1\leq i\leq m}(n,n'(i))+1,j} = \nabla_{n,j} \vee \lambda \in \mathbf{N}(P).$$

Suppose now that $\langle \nabla_{n,j}, E_{n,j}, \theta_{n,j}\rangle$ is a node. If

$$(\Gamma'_{n'(i),j'(i)}, \theta_{n'(i),j'(i)}) = (\Pi_i, \mathrm{mgu}(\beta_i, \lambda\theta_{n,j})\chi) \in \mathbf{Y}(P)$$

holds for some $1 \leq i \leq m$ and λ is a positive literal, then it holds with (N2) that

$$\Gamma_{\max(n,n'(i))+1,j} = \nabla_{n,j} \vee \lambda \in \mathbf{N}(P).$$

Remark that in cases (c) and (d) it is not required that $\nabla_{n,j}$ be in $\mathbf{N}(P)$ or that $(\nabla_{n,j}, \theta_{n,j})$ be in $\mathbf{Y}(P)$. We are now able to look at some useful properties of the universal search structure $\mathcal{U}$. Let $\eta_{n,j} = \langle \Gamma_{n,j}, E_{n,j}, \theta_{n,j}\rangle$ and $\eta_{m,j} = \langle \Gamma_{m,j}, E_{m,j}, \theta_{m,j}\rangle$ be two nodes in the structure $\mathcal{U}$. If $\eta_{n,j}$ and $\eta_{m,j}$ are on a path and $m \leq n$, then it holds

(1) $E_{m,j} \subseteq E_{n,j}$ and

(2) $\theta_{n,j} \cong \theta_{m,j}\theta_{n,j}$.

Since $E_{m,j} \subseteq E_{n,j}$, it holds $\theta_{n,j} = \mathrm{mgu}(E_{n,j}) = \mathrm{mgu}(E_{m,j} \cup E)$. That means $\theta_{n,j} \cong \theta_{m,j}\sigma$. Since $\mathrm{mgu}(E')$ is idempotent for any equation E', it holds that $\theta_{n,j} \cong \theta_{m,j}\theta_{m,j}\sigma \cong \theta_{m,j}\theta_{n,j}$.

Lemma 7.2 *If $\eta_{n,j}$ and $\eta_{m,j}$ are on a path, $m \leq n$ and $P \vdash \forall\Gamma_{m,j}\theta_{m,j}$ or $P \vdash \forall \sim\Gamma_{m,j}$, then it holds $P \vdash \forall\Gamma_{n,j}\theta_{n,j}$ or $P \vdash \forall \sim\Gamma_{n,j}$*

Proof by induction on the definition of the universal search structure $\mathcal{U}$. □

The universal search structure $\mathcal{U}$ may be finite or infinite; that means the height of $\mathcal{U}$ may be finite or infinite. Suppose the height of $\mathcal{U}$ is finite. Then for each node $\eta = \langle \Gamma, E, \theta\rangle$ occurring in $\mathcal{U}$ it holds with Lemma 7.2 that either $P \vdash \forall\Gamma\theta$ or $P \vdash \forall \sim\Gamma$.

Note that the search structure for the program

$$\begin{aligned} q(x) &\leftarrow q(f(x)) \\ q(g(x) &\leftarrow \end{aligned}$$

with the goal $\leftarrow q(h(w))$, where x and w are variables, is infinite; one also gets infinite search structures when considering the modified programs `Factorial` and `Times` at the end of section 6 with the goals $\bot\leftarrow\sim$ `factorial`$(v, 0)$ and $\bot\leftarrow$ `times`$(v, w, 0)$.

Suppose now that the height of $\mathcal{U}$ is infinite and hence the structure $\mathcal{U}$ does contain infinite paths such that Γ does not occur in any node on these paths. Let the goal occurring in a node η be writen in the form $\nabla \star \lambda$ such that λ is the literal which is last applied using rules (a)–(d) above to obtain that goal and $\star \in \{\wedge, \vee\}$. Let then

$$\mathcal{E} = \{\ \eta\ \mid\ \eta = \langle \nabla \star \lambda, E, \theta\rangle \text{ and } \eta \text{ occurs on an infinite path of } \mathcal{U}\ \}$$

and $\mathbb{H}$ denotes the set of $n \in \omega$ such that there exists a substructure of $\mathcal{U}$ of height n. Suppose m, n are two heights of $\mathcal{U}$ and η_m, η_n are two nodes belonging to $\mathcal{E}$. We write $\eta_m \leq \eta_n$, if $m \leq n$ and the sturcture with root node η_m is a substructure of that with root node η_n. It holds that

(1) $\forall n \in I\!H \exists j$ such that $\eta_{n,j} \in \mathcal{E}$.

(2) $\forall \eta \forall \eta' \, \eta' \leq \eta \ \wedge \ \eta' \in \mathcal{E} \rightarrow \eta \in \mathcal{E}$.

(3) $\forall \eta \in \mathcal{E} \forall \eta' \in \mathcal{E} \, \eta \leq \eta'$ or $\eta' \leq \eta$.

Let us now define

$$\mathcal{T}_{\mathcal{E}} = \{ \ \nabla \ \mid \ \eta = \langle \nabla, E, \theta \rangle \in \mathcal{E} \ \} \quad \text{and} \quad \mathcal{T}_{\mathcal{U}} = \bigcup_{\mathcal{E}} \mathcal{T}_{\mathcal{E}}.$$

Lemma 7.3 *Let Γ be a formula, P be a program and θ be a substitution.*

(i) If either $P \vdash \forall \Gamma\theta$ or $P \vdash \forall \ \sim\Gamma$, then either $\Gamma\theta \in \mathcal{T}_{\mathcal{U}}$ or $\sim\Gamma \in \mathcal{T}_{\mathcal{U}}$.

(ii) For any goal Γ it does not hold that $\Gamma \in \mathcal{T}_{\mathcal{U}}$ and $\sim\Gamma \in \mathcal{T}_{\mathcal{U}}$.

Proof Let Γ be a goal, P be a program, θ be a substitution.

(i) Suppose that it either holds $P \vdash \forall \Gamma\theta$ or $P \vdash \forall \ \sim\Gamma$. Then there exists a node in $\mathcal{U}$, say at height m, such that $\eta_{m,j} = \langle \Gamma, E, \theta \rangle$ for some $j \geq 0$. Since for each m' with $m' \leq m$ such that $\eta_{m',j} \leq \eta_{m,j}$ it holds that either $P \vdash \forall \Gamma'\theta$ or $P \vdash \forall \ \sim\Gamma'$, we have then either $\Gamma\theta \in \mathcal{T}_{\mathcal{U}}$ or $\sim\Gamma \in \mathcal{T}_{\mathcal{U}}$.

(ii) Suppose any goal Λ such that $\Lambda \in \mathcal{T}_{\mathcal{U}}$ and $\sim\Lambda \in \mathcal{T}_{\mathcal{U}}$. That means there is a program P such that $P \vdash \forall \Lambda\theta$ for some substitution θ and $P \vdash \forall \ \sim\Lambda$. Since the goal is arbitrary chosen, we let the goal Γ be such a goal. Then it holds with (i) that $\Gamma \in \mathcal{T}_{\mathcal{U}}$; that is a contradiction to the construction of $\mathcal{U}$ and hence of $\mathcal{T}_{\mathcal{U}}$. □

Assuming the set $\mathcal{T}_{\mathcal{U}}$ is defined, we now construct the Herbrand structure $\mathcal{H}$ as follows:

- $\mathcal{T}_{\mathcal{F},\mathcal{V}}/\cong$ is the domain of discourse;
- for each $f \in \mathcal{F}$ it holds $\mathcal{I}(f)(\vec{t}) = f(\vec{t})$;
- for each $q \in$ PRED it holds

$$\mathcal{I}(q)(\vec{t}) = \{\vec{t} \in (\mathcal{T}_{\mathcal{F},\mathcal{V}})^n/\cong \mid q(\vec{t})\rho \in \mathcal{T}_{\mathcal{U}} \text{ for some } \rho \in \mathcal{R}\}.$$

Lemma 7.4 *Let Γ be a goal. Then*

$$\mathcal{H} \models \Gamma \iff \textit{for some } \rho \in \mathcal{R} \ \textit{ it holds } \Gamma\rho \in \mathcal{T}_{\mathcal{U}}.$$

Proof by induction on the structure of the goal Γ.

`Case` $\Gamma = q(\vec{t})$: $\mathcal{H} \models \Gamma$ for some $\rho \in \mathcal{R}$ means according to the definition of $\mathcal{H}$ that $\Gamma\rho \in \mathcal{T}_{\mathcal{U}}$.

`Case` $\Gamma = \sim q(\vec{t})$: $\mathcal{H} \models \sim q(\vec{t})$ for all $\rho \in \mathcal{R}$ means according to the definition of $\mathcal{H}$ that $\sim q(\vec{t})\rho \in \mathcal{T}_{\mathcal{U}}$.

`Case` $\Gamma = \Delta \wedge \lambda$: $\mathcal{H} \models \Delta \wedge \lambda \iff \mathcal{H} \models \Delta$ and $\mathcal{H} \models \lambda$. From the induction hypothesis it holds $\Delta \in \mathcal{T}_{\mathcal{U}}$ and $\lambda \in \mathcal{T}_{\mathcal{U}}$. From (a) or (b) in the construction of $\mathcal{U}$ it holds $\Delta \wedge \lambda \in \mathcal{T}_{\mathcal{U}}$ with ρ the identity substitution.

`Case` $\Gamma = \beta \leftarrow \Pi$: '$\Longleftarrow$'. Suppose it holds $(\beta \leftarrow \Pi) \in \mathcal{T}_{\mathcal{U}}$. Then we have to show that $\mathcal{H} \models (\beta \leftarrow \Pi)$. Assume now that $\Pi \in \mathcal{T}_{\mathcal{U}}$ holds. Then from the induction hypothesis it holds $\mathcal{H} \models \Pi$. Since it holds that $(\beta \leftarrow \Pi) \in \mathcal{T}_{\mathcal{U}}$ we get $\mathcal{H} \models \beta$. Hence, from the induction hypothesis it holds $\mathcal{H} \models (\beta \leftarrow \Pi)$.

'$\Longrightarrow$'. Suppose $\mathcal{H} \models (\beta \leftarrow \Pi)$ for some formula of this form. Let then n be an height of $\mathcal{U}$ such that $\Gamma = \beta \leftarrow \Pi$ occurs in a node at this height and Π occurs in a node at an height m with $m \leq n$. Then $\Pi \in \mathcal{T}_{\mathcal{U}}$. Hence, $(\beta \leftarrow \Pi) \in \mathcal{T}_{\mathcal{U}}$ from the construction of $\mathcal{U}$ and hence of $\mathcal{T}_{\mathcal{U}}$. □

Let us now establish a relationship between the universal search structure and the derivation structure defined in section 5 and then prove the completeness of the SLDNF-resolution for an arbitrary program with an arbitrary goal. A negative literal in which variables do occur is subject of selection.

Lemma 7.5 *Let Γ be a goal and P be a program. Γ is provable in the inference system inf($\mathcal{L}$) with respect to P if and only if Γ is an element of $\mathcal{T}_{\mathcal{U}}$.*

Proof: by induction on the structure of the goal Γ.

`case` $\Gamma \in \{\ \epsilon, \bot, R(\vec{t}), \sim R(\vec{s})\ \}$: is obvious, since it follows with the logical axioms.

`case` $\Gamma = \Delta \wedge \lambda$: $\Delta \wedge \lambda$ is provable in inf($\mathcal{L}$) if and only if Δ is provable in inf($\mathcal{L}$) and λ is provable in inf($\mathcal{L}$). Then it holds with the induction hypothesis that $\Delta \in \mathcal{T}_{\mathcal{U}}$ and $\lambda \in \mathcal{T}_{\mathcal{U}}$. Hence, it holds $\Delta \wedge \lambda \in \mathcal{T}_{\mathcal{U}}$.

`case` $\Gamma = \beta \leftarrow \Pi$: '$\Longrightarrow$'. Suppose that $\beta \leftarrow \Pi$ is provable in inf($\mathcal{L}$). Let then n be the height at which Γ is a node in the derivation structure. It is obvious from the construction of the derivation structure that Π is a node at height, say m, with $m < n$ or β is a node at height, say m' with $m' < n$. From the induction hypothesis it follows that $\Pi \in \mathcal{T}_{\mathcal{U}}$ or that $\beta \in \mathcal{T}_{\mathcal{U}}$. Then it holds that $(\beta \leftarrow \Pi) \in \mathcal{T}_{\mathcal{U}}$.

'$\Longleftarrow$'. Suppose that $(\beta \leftarrow \Pi) \in \mathcal{T}_{\mathcal{U}}$. Then there exists an height n such that $\beta \leftarrow \Pi$ occurs in a node at this height. Then from the construction of $\mathcal{T}_{\mathcal{U}}$ there exist $m < n$ and $m' < n$ such that $\sim\Pi$ occurs at height m in a node of $\mathcal{T}_{\mathcal{U}}$ and β occurs at height m' in a node of $\in \mathcal{T}_{\mathcal{U}}$. From the induction hypothesis it holds that Π is provable in inf($\mathcal{L}$) and that β is provable in inf($\mathcal{L}$). Hence, it holds that $\beta \leftarrow \Pi$ is provable in inf($\mathcal{L}$). □

Let us now state and prove the main result of this work.

Theorem 7.1 (Completeness) *Let P be a well-formulated program and Γ be a goal.*

(i) If pcomp(P) $\models \Gamma\theta$, then $P \vdash \forall\Gamma\vartheta$ with ϑ more general than θ.

(ii) If pcomp(P) $\models \sim\Gamma$, then $P \vdash \forall\ \sim\Gamma$.

Proof: (i) Let Γ be a goal and pcomp(P) $\models \Gamma\theta$ for some substitution θ. Then it holds with Lemma 5.2 that $\Gamma\theta$ is provable in inf($\mathcal{L}$). Then $\Gamma\theta$ has a finite derivation structure. From Lemma 7.5 it then holds that $\Gamma\theta \in \mathcal{T}_{\mathcal{U}}$. Hence, there

exists a node η at height, say n, such that a more general instance of $\Gamma\theta$ occurs in η. Then with Lemma 7.2 and ϑ more general than θ it holds that $P \vdash \forall\Gamma\vartheta$.

(ii) Let Γ be a goal and $\mathrm{pcomp}(P) \models \sim\Gamma$. Then it holds with Lemma 5.2 that $\sim\Gamma$ is provable in $\inf(\mathcal{L})$. Then $\sim\Gamma$ has a finite derivation structure. From Lemma 7.5 it then holds that $\sim\Gamma \in \mathcal{T}_{\mathcal{U}}$. Hence, there exists a node η at height, say n, such that $\sim\Gamma$ occurs in η. Then with Lemma 7.2 it holds that $P \vdash \forall\ \sim\Gamma$. □

8 Illustrating examples

8.1 Example 1

Let us look at two sample examples to illustrate the universal search structure $\mathcal{U}$, where we just show those parts of the structure that are interesting for us. Let us write for the sake of simplicity `mbr` for `member`, `djs` for `disjointset`, $E \cong F$ to denote that the expressions E and F are unifiable, that is $\theta = \mathrm{mgu}(E, F)$, and $E \ncong F$ to denote the fact that the expressions E and F are not unifiable, that is $\bot = \sim\mathrm{mgu}(E, F)$ in the graphical representation of the search structure.

$$
\begin{array}{lllll}
C_0: & \mathrm{member}(x,[x|xs]) & \leftarrow & \\
C_1: & \mathrm{member}(x,[y|ys]) & \leftarrow & \mathrm{member}(x,ys) \\
C_2: & \mathrm{append}([],y,y) & \leftarrow & \\
C_3: & \mathrm{append}([x|xs],y,[x|zs]) & \leftarrow & \mathrm{append}(xs,y,zs) \\
C_5: & \mathrm{disjointset}([],[y|ys]) & \leftarrow & \\
C_6: & \mathrm{disjointset}([x|xs],[y|ys]) & \leftarrow & \sim\mathrm{member}(x,[y|ys]), \mathrm{disjointset}(xs,[y|ys])
\end{array}
$$

The structure in figure 1 shows the use of rules (N1), (Y2) and (Y1). The structure in figure 2 does better illustrate the search structure for a negative literal in which variables do occur.

8.2 Example 2

Let us illustrate our approach with first this simple program P

$$
\begin{array}{rll}
p(x) & \leftarrow & \sim q(x,y) \\
q(1,z) & \leftarrow &
\end{array}
$$

and this goal $\Gamma = \leftarrow p(2)$. It holds with (R0) that $\epsilon\mathbf{R}(P)\varepsilon$. Since $\sim q(2,v) \in \mathbf{F}(P)$ it holds with (R2) that $\epsilon\wedge \sim q(2,v)\mathbf{R}(P)\varepsilon$. Using now (R1) it holds that $p(2)\mathbf{R}(P)\varepsilon$.

8.3 Example 3

Consider next the program `Factorial` with the goal

$$\Gamma = \leftarrow \sim \mathtt{factorial}(v, s(s(s(0))))).$$

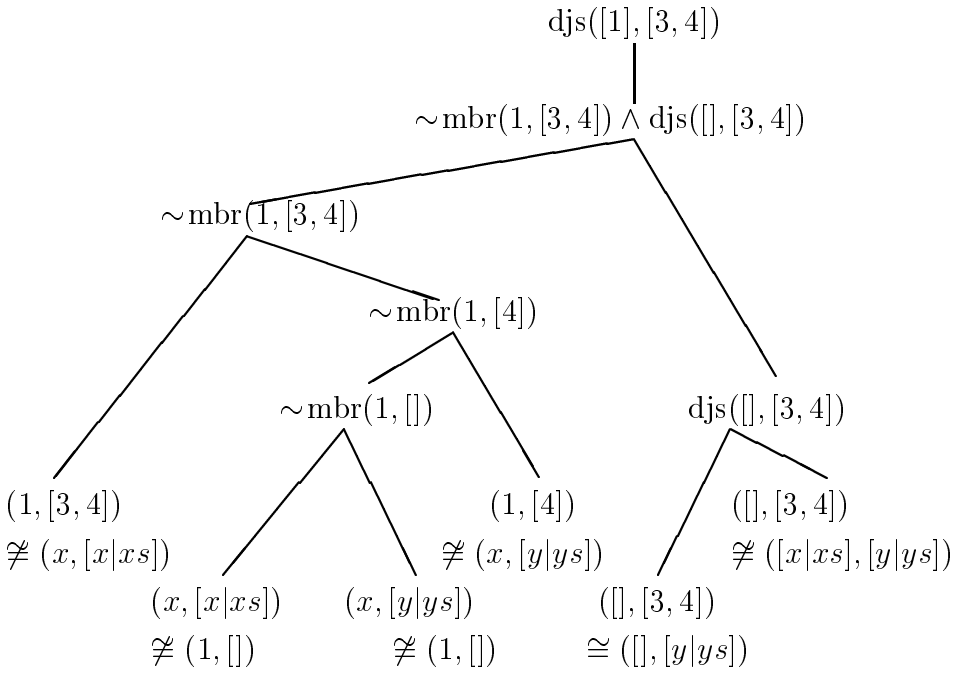


Figure 1: Sample universal search structure for disjointset([1], [3, 4])

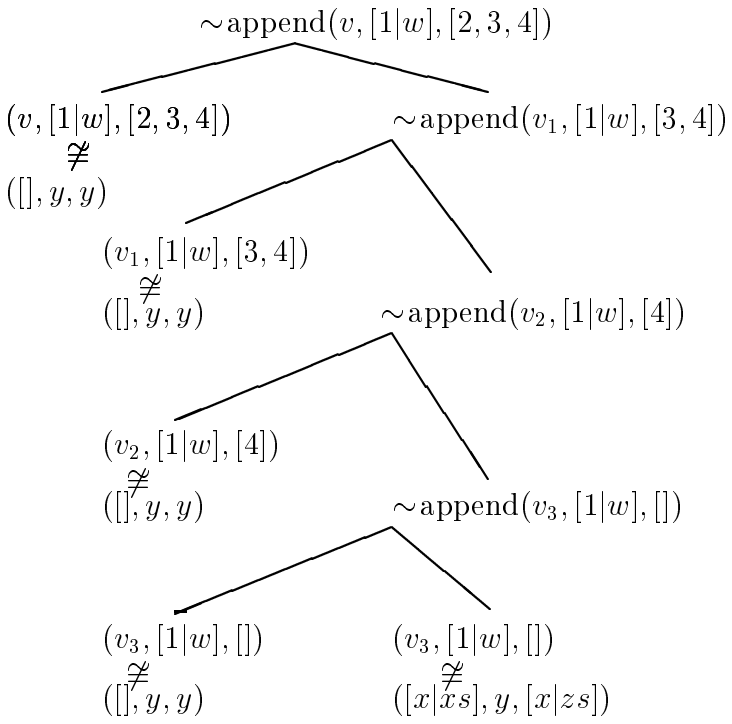


Figure 2: Sample universal search structure for ∼append(v, [1|w], [2, 3, 4])

Three steps are needed to conclude that

$$\texttt{Factorial} \vdash \sim\texttt{factorial}(v, s(s(s(0)))).$$

These three steps result from the fact that

$$\forall(x, y) \in \{(0, s(s(0))), (s(0), s(0)), (s(s(0)), 0)\}$$

it holds that

$$\texttt{Plus} \vdash \texttt{plus}(x, s(y), s(s(s(0)))).$$

Therefore, $\texttt{plus}(x, s(y), s(s(s(0)))) \notin \mathbf{F}(\texttt{Plus})$.

Step 1: Using rule (F1) it is obvious that

$$\sim\texttt{times}(v_2, s(s(0)), s(0)) \in \mathbf{F}(P).$$

Hence,

$$\sim\texttt{times}(v_2, s(s(0)), s(0)) \vee \sim\texttt{plus}(s(0), s(s(0)), s(s(s(0)))) \in \mathbf{F}(P).$$

Hence,

$$\sim\texttt{times}(s(v_2), s(s(0)), s(s(s(0)))) \in \mathbf{F}(P).$$

Hence,

$$\sim\texttt{factorial}(v_2, s(s(0))) \vee \sim\texttt{times}(s(v_2), s(s(0)), s(s(s(0)))) \in \mathbf{F}(P).$$

Step 2: It is obvious that

$$\sim\texttt{times}(s(s(0)), s(0), s(s(0))) \notin \mathbf{F}(P).$$

Hence,

$$\sim\texttt{times}(s(s(0)), s(0), s(s(0)))) \vee \sim\texttt{plus}(s(s(0)), s(0), s(s(s(0)))) \notin \mathbf{F}(P).$$

Hence,

$$\sim\texttt{times}(s(s(s(0))), s(0), s(s(s(0)))) \notin \mathbf{F}(P).$$

Since

$$\sim\texttt{factorial}(s(s(0)), s(0)) \in \mathbf{F}(P)$$

it holds that

$$\sim\texttt{factorial}(s(s(0)), s(0)) \vee \sim\texttt{times}(s(s(s(0))), s(0), s(s(s(0)))) \in \mathbf{F}(P).$$

Step 3: It is obvious that

$$\sim\texttt{times}(0, s(s(s(0))), 0) \notin \mathbf{F}(P).$$

Hence,

$$\sim\texttt{times}(0, s(s(s(0))), 0) \vee \sim\texttt{plus}(0, s(s(s(0))), s(s(s(0)))) \notin \mathbf{F}(P).$$

Hence,

$$\sim\texttt{times}(s(0), s(s(s(0))), s(s(s(0)))) \notin \mathbf{F}(P).$$

Since

$$\sim\texttt{factorial}(0, s(s(s(0)))) \in \mathbf{F}(P)$$

it holds that

$$\sim\texttt{factorial}(0, s(s(s(0))))\vee \sim\texttt{times}(s(0), s(s(s(0))), s(s(s(0)))) \in \mathbf{F}(P).$$

Putting now steps 1 – 3 together with

$$(v_1, w_1) \in \{(0, s(s(s(0)))), (s(s(0)), s(0)), (y, s(s(0)))\},$$

where y is a variable, one gets, that is the body of clause C_8,

$$\sim\texttt{factorial}(v_1, w_1)\vee \sim\texttt{times}(s(v_1), w_1, s(s(s(0)))) \in \mathbf{F}(P).$$

Since it also holds with clause C_7 that

$$\sim\texttt{factorial}(v, s(s(s(0)))) \in \mathbf{F}(P),$$

it is obvious with (R2) that

$$\sim\texttt{factorial}(v, s(s(s(0))))\mathbf{R}(P)\varepsilon$$

9 Comparison with other recent results

Let us illustrate the comparison by the simple examples given in the introduction. For the sake of simplicity the following presentation is not self-contained; therefore, we just refer to the definition in [4] or in [15]. Let us in this context first consider the program `append` with the goal

$$\Gamma = \leftarrow \sim \texttt{append}(v, [1|w], [2, 3, 4]).$$

Following Stärk in [15] and referring to definition 4.3 or table 1 it is clear that $\langle +, \Gamma, \varepsilon\rangle$ flounders since $\text{vars}(\Gamma) \neq \emptyset$.

Following Drabent in [4] and referring to definition 4.1 it is obvious that the SLDNF1 prefailed tree for Γ does not exist since Γ is not ground.

Consider now the program `factorial` with the goal

$$\Gamma = \leftarrow \sim \texttt{factorial}(v, s(s(s(0)))).$$

Using similar arguments as above one also follows that

$$\langle +, \Gamma, \varepsilon\rangle \text{ flounders since } \text{vars}(\Gamma) \neq \emptyset$$

following Stärk and that the SLDNF1-prefailed tree for Γ does not exist since Γ is not ground following Drabent.

10 Related works and conclusion

The objective of this work is to enlarge the class of programs for which the SLDNF-resolution is proven complete when (1) eliminating the restrictive condition on the selection of a negative literal and (2) keeping logic programming away from any notion of mode.

10.1 Related works

As mentioned in the introduction, the proof of the completeness of the SLDNF-resolution for a relevant class of programs has been tackled by many researchers. We just cite two recent works in this area, namely that of Drabent in [4] and that of Stärk in [15]. These recent results on this topic are based on a top down definition of SLDNF-resolution, which is strategically different from the bottom up definition given in this paper. As long as a top down definition has to derive an , say empty, goal from a given, say non-empty, goal, some restrictive conditions are needed to ensure

(1) the termination of the derivation or

(2) the correctness of a derivation step.

It is then well-known that the class of languages handled by a bottom up definition is larger than that handled by a top down definition. Hence, the class of logic programs for which the bottom up SLDNF-resolution is proven complete is larger than that for which the top down SLDNF-resolution is proven complete.

10.2 Conclusion

As stated by Kunen in [8] one might get better completeness result by strengthening the SLDNF-resolution to compute more answers. One way, perhaps not the easiest, to reach this claim is

(i) to eliminate the condition which states that a selected negative literal has to be ground in the definition of the SLDNF-resolution and

(ii) to consider a universe which does contain variables modulo renaming.

To successfully eliminate this condition in the definition of the SLDNF-resolution and prove the SLDNF-resolution complete, we use a bottom up definition of the SLDNF-resolution.

Some further interesting problems still remain. The partial program completion introduced by Jäger in [6] and Stärk in [13] is not practical enough by the implementation of the SLDNF-resolution. Hence, if the SLDNF-resolution is of interest, then a program completion which is simple and meets the behavior of the SLDNF-resolution and which is consistent is required. Works in this direction are in progress.

Acknowledgment

I thank K. R. Apt for fruitful comments and suggestions on an earlier version of this paper.

References

[1] Krzysztof R. Apt. Logic programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 10, pages 495–574. Elsevier, 1990.

[2] Krzysztof R. Apt and Roland Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 19/20:9–71, 1995.

[3] Krzysztof R. Apt and Kees Doets. A new definition of sldnf-resolution. *Journal of Logic Programming*, 18:177–190, 1994.

[4] Wlodzimierz Drabent. Completeness of sldnf-resolution for nonfloundering queries. *Journal of Logic Programming*, 27(2):89–106, 1996.

[5] Moreno Falaschi, Giorgio Levi, Catuscia Palamidessi, and Maurizio Martelli. Declarative modeling of the operational behavior of logic languages. *Journal of Theoretical Computer Science*, 69:289–318, 1989.

[6] Gerhard Jäger. Some proof-theoretic aspects of logic programming. In Friedrich L. Bauer, Wilfried Brauer, and Helmut Schwichtenberg, editors, *logic and algebra of specification*, pages 113–142. Springer, Berlin, 1993.

[7] Kenneth Kunen. Negation in logic programming. *Journal of Logic and Computation*, 4:289–308, 1987.

[8] Kenneth Kunen. Signed data dependencies in logic programs. *Journal of Logic Programming*, 7:231–245, 1989.

[9] John W. Lloyd. *Foundations of Logic Programming*. Springer, 1987.

[10] Helmut Schwichtenberg. Logikprogrammierung. Vorlesungsausarbeitung, Ludwig-Maximilians-Universität München, Wintersemester, 1993.

[11] J. R. Shoenfield. *Mathematical Logic*. Addison-Wesley, Reading, Massachusetts, 1967.

[12] Wayne Snyder and Jean H. Gallier. Higher-order unification revisited: Complete sets of transformations. *Journal of Symbolic Computation*, 8:101–140, 1989.

[13] Robert F. Stärk. From logic programs to inductive definitions. Technical report, Mathematisches Institut, Universität München, 1993. for Logic Colloquium 1993, Keele, Great Britain.

[14] Robert F. Stärk. Input/output dependencies of normal logic programs. *Journal of Logic and Computation*, 4(3):249–262, 1994.

[15] Robert F. Stärk. A direct proof of the completeness of sldnf-resolution. Technical report, Institute of Informatics, University of Fribourg, June 1997.

[16] Leon Sterling and Ehud Shapiro. *The Art of Prolog Advanced Programming Techniques*. The MIT Press, Cambridge, Massachusetts, 1986.