Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

W.P. Weijland

Semantics for logic programs without occur check

Computer Science/Department of Software Technology Report CS-R8740 August

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Semantics for Logic Programs without Occur Check

W.P. Weijland

Centre for Mathematics and Computer Science,

Kruislaan 413, 1098 SJ Amsterdam.

abstract: For reasons of efficiency, in almost all implementations of Prolog the *occur check* is left out. This mechanism should protect the program against introducing *circular bindings of variables*. In practice the occur check is very expensive, however, and it is left to the skills of the user, to avoid these circular bindings in the program.
In this paper a semantics of Prolog without occur check is introduced, by considering circular bindings {x/f(x)} as *recursive equations* {x=f(x)}. The new kind of resolution, i.e.: SLD-resolution without occur check, is referred to as *CSLD*-resolution. Important theorems such as soundness and completeness of CSLD-resolution, are established. Moreover, the *finite failure set* turns out to be precisely the complement of the greatest fixed point of a monotonic mapping T'_P on the complete Herbrand base (see [10]). Soundness and completeness of the negation as failure rule can be obtained in this new setting.

Key words and phrases: Logic Programming, Occur-Check, Infinite trees, fixed point semantics.
1985 Mathematics subject classification : 68Q40; 68Q55.
1982 CR categories: F.3.1; F.3.2; F.4.1;

1. INTRODUCTION.

For reasons of efficiency, in almost all implementations of Prolog the *occur check* is left out, which is a mechanism that should protect the program against introducing *circular bindings of variables*. For instance in a substitution {x/f(x)}, the variable x is bound to a term f(x) containing the variable x again. The problem is, that any such binding endangers the correct behaviour of a Prolog system. In fact, without the occur check we no longer have soundness of SLD-resolution (see LLOYD [10]). For example consider the following program:

example

P: test ← p(x,x)
p(y,f(y)) ←

Given the goal ← test, a Prolog system without occur check will answer 'yes' since p(x,x) will be successfully unified with p(y,f(y)) by the substitution {x/y, y/f(y)}. However, this answer is quite wrong, since test is not a logical consequence of P.

In practice, however, the occur check is very expensive and it is usually left to the skills of the user to avoid these circular bindings in the program. For instance in PLAISTED [11], a method is presented to detect circular bindings more efficiently, by preprocessing Prolog programs.

It would be convenient to develop a theory for SLD-resolution without occur check, and for this

Report CS-R8740
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

reason Prolog II (see COLMERAUER [3]) has been studied quite intensively in the past few years. Roughly speaking, Prolog II is standard Prolog without occur check and can be regarded as a system which manipulates infinite trees (see COLMERAUER [2]).

The question remains, whether or not Prolog II can be thought of as a logic programming language, since the example above shows that Prolog II presents incorrect derivations. This problem was solved by VAN EMDEN & LLOYD [6], by formulating a soundness theorem for Prolog II. In the above example, the computed substitution {x/y,y/f(y)} can be translated to a set of equations {x=y,y=f(y)}, and clearly test is a logical consequence of P∪{x=y,y=f(y)}. There are still many results left to be established, such as completeness for instance, to develop a complete theory for Prolog II.

In this paper a semantics for logic programs without occurcheck is presented by considering circular bindings {x/f(x)} as *recursive equations* {x=f(x)}, and extending the Herbrand universe (consisting of all closed terms) by adding all infinite terms {x=f(f(f(...)))} to it (see COURCELLE [4]). We introduce a new kind of resolution, which will be referred to as CSLD-resolution (*complete* SLD-resolution), which is precisely SLD-resolution without occur check. Following this idea, we find that both soundness and completeness can be obtained.

Independently from this paper, similar results were stated by JAFFAR, LASSER & MAHER [7], although in a somewhat different setting. We will only need a small equational theory for Prolog II, whereas in [6] and [7] this theory contains infinitely many existential formulas, one for every recursive equation. Therefore we do not need to put any constraints on the models of Prolog II programs and the results are more general.

Furthermore, *negation as a failure rule* is considered. It turns out that the complement of the finite failure set is equal to the greatest fixed point of a monotonic mapping T'_P on the complete Herbrand base (see [10]). This is an even better result than we had before for SLD-semantics, and so far, such a result could not be found in the literature although many attempts were made to find an appropriate interpretation for $gfp(T'_P)$ (see for instance LEVI & PALAMIDESSI [9]).

Soundness and completeness of the negation as failure rule is obtained. It turns out that due to the new setting, the proof of the completeness theorem becomes much shorter compared with the well-known proofs for SLD-resolution in [8], [10] and [12]. Then, we may conclude as a general result that comp(P)∪{A} has a 'complete' Herbrand model iff it has a model, which indicates that we may expect CSLD-resolution to have some nice properties extra, that we do not have for ordinary SLD-resolution.

2. COMPLETE HERBRAND MODELS

We will assume P to be a set of *program clauses* $\forall(B_1 \wedge \ldots \wedge B_k \rightarrow A)$, usually written as $A \leftarrow B_1,\ldots,B_k$, where $B_1,\ldots,B_k,A$ are atoms not containing '='. The language of P will be denoted by L(P) or L_P.

In this section we will formally introduce *complete* Herbrand models for P. First we will present a precise definition of a *complete term*, as can be found in [10], and next establish some general model theoretical results.

Let ω^* be the set of all finite sequences of non-negative integers. Such a finite sequence will be written as $[i_1,\ldots,i_k]$, for some $i_1,\ldots,i_k \in \omega$. For all $m,n \in \omega^*$ we write [m,n] for the concatenation of m and n, and for $i \in \omega$ we write [m,i] instead of [m,[i]]. For $X \in \omega^*$ we write $|X|$ for the cardinality of X. Moreover, $|[i_1,\ldots,i_k]|=k$ and $[m,n]=|m|+|n|$.

definition 2.1 $T \subseteq \omega^*$ is called a *tree* if T satisfies the following conditions:

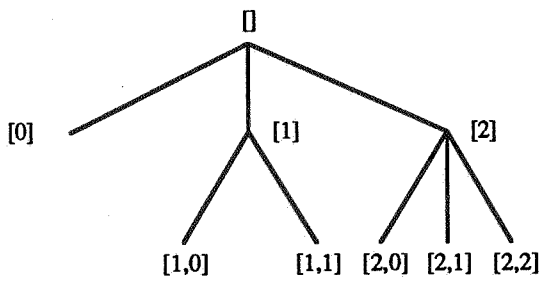
(i) for all $n \in \omega^*$ and $i,j \in \omega$: $[n,i] \in T \wedge j<i \Rightarrow n \in T \wedge [n,j] \in T$

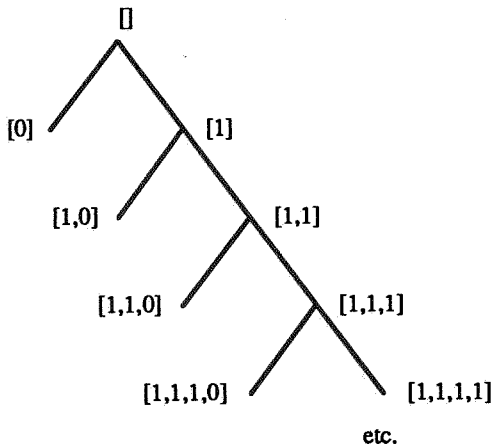
(ii) $|\{i:[n,i] \in T\}|$ is finite for all $n \in T$.

So, by definition 2.1 we can interpret [ ] as the *root* of the tree and [n,0],[n,1],...,[n,k] as the descendents of the node n for all $n \in T$, $k<\omega$.

example

The finite tree {[], [0], [1], [2], [1,0], [1,1], [2,0], [2,1], [2,2]} and the infinite tree {[], [0], [1], [1,0], [1,1], [1,1,0], [1,1,1], [1,1,1,0], [1,1,1,1], ...} can be pictured by:





Now let S be a set of symbols and ar: $S \rightarrow \omega$ be a mapping defining the *arity* of a symbol.

definition 2.2 A *complete term* (over S) is a function t: dom(t)→S such that:
(i) the domain of t, dom(t), is a non-empty tree
(ii) for all n∈dom(t): ar(t(n)) = |{i: [n,i]∈dom(t)}|.
In a language L, a *complete atom* is a complete term t such that t([]) is a predicate symbol.

The tree dom(t) is called the *underlying tree* of t. The set of all complete terms over S is denoted by $Term_S$; these terms can be looked at as (possibly) infinite terms.
By definition a term t is finite if and only if dom(t) is finite.

definition 2.3 The *depth* dp(t) of a term t is defined by:
(i) dp(t) = ∞, if t is infinite
(ii) dp(t) = 1+ max{|n|: n∈dom(t)}, if t is finite.

Next, we will define a metric on $Term_S$.

definition 2.4 Let s,t∈$Term_S$ and s≠t then we define α(s,t) as being the *least* depth at which s and t differ. Then we define

$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ 2^{-\alpha(s,t)} & \text{if } s\neq t. \end{cases}$$

proposition 2.1 ($Term_S$, d) is a (ultra-) metric space.

The proof is simple, and omitted here.
Note that the larger the depth is at which two terms differ, the smaller is their distance. From now on, the closure of a set X⊆$Term_S$ under the topology induced by d, will be denoted by X^c or c(X).
Next, we define the *truncation* of a term, to have finite approximations of infinite terms. Assume Ω to be an extra constant symbol (hence with arity zero), not in S.

definition 2.5 The *truncation at depth n* of a term t, notation $\alpha_n(t)$, can be found from the complete term t by replacing all symbols at depth n by Ω and leaving out all symbols at greater depth.
The underlying tree $\alpha_n(t)$ is adjusted in the same way, by leaving out all nodes without a label.

Clearly we have $dp(\alpha_n(t))\leq n+1$, for all t. Moreover, $d(\alpha_n(t),t)\leq 2^{-n}$ and therefore $\lim_{n\to\infty} \alpha_n(t) = t$.

definition 2.6 A metric space (X,d) is *compact* if every sequence in X has a subsequence which converges to a point in X.

proposition 2.2 ($Term_S$,d) is compact iff S is finite.

For a proof of this well known theorem, see [10].

definition 2.7 For any program P, the *complete Herbrand universe* CU_P, is defined by $Term_{L(P)}$. We will write U_P for its Herbrand universe which consists of all finite closed terms.

Next we will consider models for a program P, having CU_P as its universe.

definition 2.8 Let L be a language.

A *complete Herbrand model* for L is a model $\mathcal{M}$, such that

(i) $a^{\mathcal{M}} = a$, for all constants $a \in L$

(ii) $f^{\mathcal{M}}(t_1{}^{\mathcal{M}},...,t_k{}^{\mathcal{M}}) = f(t_1,...,t_k)$, for all function symbols $f \in L$, and complete terms $t_1,...,t_k \in Term_L$.

A complete Herbrand model for a program P is a complete Herbrand model for L_P which is a model for P.

Note that a program P can have more then one complete Herbrand model, since the interpretation of the relation symbols is still free.

definition 2.9 The *Herbrand base* B_P of a program P, is defined by

$$B_P = \{R(t_1,...,t_k) : R \text{ is a relation symbol in } L_P \text{ and } t_1,...,t_k \in U_P\}.$$

The *complete Herbrand base* CB_P of a program P, is defined by

$$CB_P = \{R(t_1,...,t_k) : R \text{ is a relation symbol in } L_P \text{ and } t_1,...,t_k \in CU_P\}.$$

In case we only have a language L, we simply denote B_L and CB_L respectively.

The elements of CB_P can be represented as trees as well. Moreover, the metric d on CU_P can be extended to CB_P. Note that it directly follows that ${B_P}^c = CB_P$ (clearly $B_P \subseteq CB_P$).
In general any (complete) Herbrand model for a program P, can be associated with a subset of the (complete) Herbrand base B_P or CB_P respectively: such a subset then denotes the complete set of 'ground' atoms, holding in the model. For any ground atom A and Herbrand model $\mathcal{M}$, we will use both notations $A \in \mathcal{M}$ and $\mathcal{M} \models A$, to express that A holds in the model $\mathcal{M}$.

3. RECURSIVE SPECIFICATIONS

In this section we consider so called recursive specifications, which are finite sets of positive equational formulas and will be used later instead of the usual notion of a *substitution*. Returning to Prolog, we will slightly change the usual *unification algorithm*. This is necessary, since from now on we will work in *complete* Herbrand models.

definition 3.1 Let P be a program with language L_P. Then the theory $Eq(L_P)$ (or Eq(P)) is defined by the following axioms:

1. $c \neq d$ for all pairs of distinct constants c,d from L_P
2. $\forall x.\ f(x) \neq g(x)$ for all pairs of distinct function symbols f,g from L_P
3. $\forall x.\ f(x) \neq c$ for all function symbols f and all constants c from L_P
4. $\forall xy.\ x_1 \neq y_1 \vee \ldots \vee x_k \neq y_k \rightarrow f(x) \neq f(y)$ for all function symbols f from L_P
5. $\forall x.\ x = x$
6. $\forall xy.\ x_1 = y_1 \wedge \ldots \wedge x_k = y_k \rightarrow f(x) = f(y)$ for all function symbols f from L_P
7. $\forall xy.\ x_1 = y_1 \wedge \ldots \wedge x_k = y_k \rightarrow (P(x) \rightarrow P(y))$ for all predicate symbols P from L_P.

Clearly the theory $Eq(L_P)$ holds in all (complete) Herbrand models for L_P.
The axioms of $Eq(L_P)$ are introduced in [10] to model finite failure: $Eq(L_P)$ forces any two syntactically different terms to be different in all its models. In [10] we even find an extra axiom:

8. $\forall x.\ x \neq t[x]$

for all terms t that are unequal to a variable and contain the variable x. This axiom is needed to express that the elements in the Herbrand universe consist of all *finite* terms from L_P. Since in the complete Herbrand universe we do have infinite terms as well, we will omit axiom 8 from our equational theory.
It turns out to be convenient to consider substitutions no longer as a syntactical operation of binding variables, but directly as equational formulas.

definition 3.2 A *(recursive) specification* in a language L is a set of equations of the form:
$\{t_1(x_1,\ldots,x_n)=s_1(x_1,\ldots,x_n),\ldots,t_k(x_1,\ldots,x_n)=s_k(x_1,\ldots,x_n)\}$ for (open) terms $t_i, s_i \in L$ and variables $x_1,\ldots,x_n$.

definition 3.3 An *open* complete term in a language L is obtained by constructing a complete term from $L \cup \{x_i : i \leq n\}$, where $\{x_i : i \leq n\}$ denotes a finite set of variable symbols with arity zero.

For instance, f(x,f(x,f(x,...))) is an open complete term obtained from the binary symbol f and the variable symbol x. Note that an open complete term only has finitely many variables, called the *free* variables of the term.
The metric d can simply be generalised to open complete terms, by extending the language L with extra variable symbols.

proposition 3.1 Let L be a language, and let $\mathcal{M}$ be a complete Herbrand model for L. Then for all open complete terms $t_1(x)$, $t_2(x)$ we have:

$$(\forall s \in U_L: \mathcal{M} \models t_1(s) = t_2(s)) \Leftrightarrow d(t_1(x),t_2(x)) = 0.$$

The proof is simple. In fact, the proposition states that open complete terms are syntactically different if and only if they are semantically different in some complete Herbrand model; moreover, this difference can be detected in the model by substituting finite terms $s \equiv s_1,...,s_k$ for the variables.

definition 3.4 A specification $\rho(x)$ is said to be in *reduced form* if it is of the form $\{x_1=s_1(x_1,...,x_n),...,x_k=s_k(x_1,...,x_n)\}$, where $x_1,...,x_k$ are distinct variables. Moreover, $\rho(x)$ *has* a reduced form if it is equivalent to a specification which is in reduced form.

A specification is said to be in *contradictory form* if it contains an equation a=b or $f(t_1,...,t_n)=g(s_1,...,s_n)$ for some distinct symbols a,b or f,g respectively. Moreover, it *has* a contradictory form if it is equivalent to a specification which is in contradictory form.

definition 3.5 A variable x is *bound* in ρ if $\rho \models x=t$ for some term t which is not a variable. Otherwise it is called *free*. A specification is called *ground* if it has no free variables.

example

Let $\sigma(x,y) = \{x=f(x), y=x\}$, then σ has no free variables since $\sigma \models x=f(x)$ and $\sigma \models y=f(x)\}$.

Let $\sigma(x,y,z) = \{x=f(y), y=z\}$, then x is a bound variable in σ, whereas y and z are free.

definition 3.6 A specification $\rho(x)$ is called *consistent* if $\rho \cup Eq(\rho)$ is statisfiable in a model.

The usefulness of open complete terms will become clear in the following theorem.

theorem 3.2 Let L be a language. If $\rho(x,y)$ is in reduced form, with bound variables x and free variables y, then there exist (open) complete terms $t_1(y),...,t_n(y)$ with only *free* variables from ρ, such that in every complete Herbrand model $\mathcal{M}$ for L:

$$\mathcal{M} \models \forall xy.\ \rho(x,y) \leftrightarrow \bigwedge_{i=1}^{n} x_i=t_i(y)$$

where $x \equiv x_1,...,x_n$ are distinct variables.

proof Use the fact that for all $\{x=t(x,y)\} \subseteq \rho$: $\mathcal{M} \models \forall xy.\ x=t(x,y) \leftrightarrow x=z\ [z/t^{\omega}(y)]$, in every complete Herbrand model $\mathcal{M}$ for L_ρ and with $t^{\omega}(y):=t(y,t(y,t(y,...)))$. Note that y are free variables in $\{x=t(x,y)\}$. The rest of the proof is straightforward. □

The following lemma is easy to prove:

lemma 3.3 Eq(L) holds in all complete Herbrand models for L.

theorem 3.4

(i) A specification in reduced form is consistent.

(ii) A specification in contradictory form is inconsistent.

proof (i) Let $\rho(x,y)$ be in reduced form, with language L, where $x \equiv x_1,...,x_n$ are distinct, bound variables of ρ and $y \equiv y_1,...,y_m$ are free in ρ. Let $\mathcal{M}$ be a complete Herbrand model for L. If L does not contain any constants then use $L \cup \{a\}$, for some new constant a.

By theorem 3.2 we find that ρ is satisfiable in $\mathcal{M}$ if and only if $\bigwedge_{i=1}^{n} x_i=t_i(y)$ is.

Since $x \equiv x_1,...,x_n$ are distinct, we have $\mathcal{M} \models \rho(t_1(a,...,a),...,t_n(a,...,a),a,...,a)$ for any constant $a \in L$. So ρ is satisfiable in some complete Herbrand model, and hence in a model. Since $Eq(\rho)$ holds in all complete Herbrand models, $\rho \cup Eq(\rho)$ is satifiable in a model and therefore ρ is consistent.

(ii) Directly from the definition of $Eq(\rho)$. □

corollary 3.5 Every specification in reduced form containing a constant, has a complete Herbrand model.

theorem 3.6

(i) All consistent specifications have a reduced form.

(ii) All inconsistent specifications have a contradictory form.

Theorem 3.6 is the reverse of theorem 3.4. To prove this theorem an algorithm can be constructed, which actually *decides* whether a specification is consistent or not, by calculating an equivalent specification in reduced form (if such specification exists). This algorithm consists of the following five steps, defined in COLMERAUER [2] (see also [7]). Suppose ρ is a specification.

consistency algorithm

(1) Delete from ρ all equations of the form x=x.

(2) If ρ contains an equation x=y, where x and y are different variables, then replace x in all its occurences in ρ by y.

(3) Replace an equation t=x in ρ by x=t, where t is not a variable.

(4) Replace two equations x=t and x=s in ρ by the equations x=t and t=s, where t is the smaller (in number of symbols) of the two terms t and s.

(5) Replace an equation of the form $f(t_1,...,t_n)=f(s_1,...,s_n)$ by the equations $t_1=s_1,...,t_n=s_n$.

It is well-known that, repeatedly using these five steps, any recursive specification ρ can be reduced into either a reduced form or a contradictory form, which is equivalent to ρ. This provides us with a proof of theorem 3.6.
Furthermore, using the consistency algorithm one can define a new kind of unification which precisely coincides with unification in Prolog II. Assume p is a predicate symbol and S is a set of atoms.

unification algorithm

(1) If not all atoms in S start with the same predicate symbol, then S is not unifiable.

(2) Else: if $S = \{p(t^{(i)}_1,...,t^{(i)}_n): i\leq n\}$ then apply the consistency algorithm to $\{t^{(1)}_1=t^{(2)}_1, t^{(2)}_1=t^{(3)}_1,...,t^{(m-1)}_1=t^{(m)}_1,...,t^{(1)}_n=t^{(2)}_n, t^{(2)}_n=t^{(3)}_n,...,t^{(m-1)}_n=t^{(m)}_n\}$.

corollary 3.7 A specification ρ is consistent iff it is equivalent to a simple specification.

definition 3.7 Let S be a set of (open) atoms.
A specification ρ is called *complete unifier* (cu) for S, if: $\models \forall(\rho \rightarrow \bigwedge_{A,B\in S}(A\leftrightarrow B))$.

definition 3.8 Suppose ρ is a cu for S. ρ is called *most general complete unifier* (mcu) for S, if for all complete unifiers ρ_1 for S: $\models \forall(\rho_1\rightarrow\rho)$.

proposition 3.8 For any input set S of atoms, the unification algorithm computes an mcu for S.

theorem 3.9 Let ρ be a (possibly infinite) specification in a language L with at least one constant, then the following are equivalent:

(i) $\rho\cup Eq(\rho)$ has a model

(ii) ρ has a complete Herbrand model

(iii) ρ is satisfiable in all complete Herbrand models for L.

proof (ii)⇒(iii): a complete Herbrand model for ρ is uniquely determined up to the interpretations of the relation symbols; since ρ does not contain any relation symbol ρ is satisfiable in any complete Herbrand model.
(iii)⇒(i): since L contains at least one constant, L has at least one complete Herbrandmodel, and this model satisfies ρ. Since by lemma 3.3 Eq(ρ) holds in all complete Herbrand models, this model satisfies $\rho\cup Eq(\rho)$.
(i)⇒(ii): suppose $\rho = \cup_{i\in\omega}\rho_i(x_i)$, where all $\rho_i(x_i)$ are finite subsets of ρ, and define for all n: $\Sigma_n(x_1,...,x_n) = \cup_{i\leq n}\rho_i(x_i)$. Clearly Σ_n is finite and consistent and hence equivalent to a reduced form Σ'_n. Since L contains a constant, by corollary 3.5 Σ'_n has a complete Herbrand model $\mathcal{M}_n$ such that $\mathcal{M}_n \models \Sigma'_n(t^n_1,...,t^n_n)$, say. From (ii)⇒(iii) it follows that $\mathcal{M} \models \Sigma'_n(t^n_1,...,t^n_n)$ for all complete Herbrand models $\mathcal{M}$. Thus, for every natural number n we obtain a sequence $(t^i_n)_{i\geq n}$ in

CU_ρ. Since CU_ρ is compact every such sequence has a subsequence $(\underline{t}^i_n)_{i\geq n}$ which converges to a limit t_n. Note that $\mathcal{M} \models \Sigma'_n(\underline{t}^i_1,...,\underline{t}^i_n)$ for all $i\geq n$.

Next, assume $y=s(x_1,...,x_n)$ is an equation in Σ'_n, then $\mathcal{M} \models \underline{t}^i_y=s(\underline{t}^i_1,...,\underline{t}^i_n)$ and therefore $d(\underline{t}^i_y,s(\underline{t}^i_1,...,\underline{t}^i_n))=0$ for all $i\geq n$. Suppose $\underline{t}^i_y \rightarrow t_y$ then clearly $d(t_y,s(t_1,...,t_n))=0$ and hence it follows that $\mathcal{M} \models t_y=s(t_1,...,t_n)$ for all $\mathcal{M}$. So we find that $\mathcal{M} \models \Sigma_n(t_1,...,t_n)$ for every natural number n and all $\mathcal{M}$. Therefore $\rho = \cup_{n\in\omega}\Sigma_n$ is satisfiable in all complete Herbrand models $\mathcal{M}$.

□

theorem 3.10 Let L be a language, and suppose $A(x),B(x)\in L$ are unifiable atoms with mcu $\rho(x)$ then for all complete Herbrand models $\mathcal{M}$ for L and all $t_1,...,t_k\in |\mathcal{M}|$, the following are equivalent:

1. $\mathcal{M} \models \rho(x_1,...,x_k)[x_1/t_1,...,x_k/t_k]$
2. $d(A(t_1,...,t_k),B(t_1,...,t_k)) = 0$

proof $1\Rightarrow 2$: suppose $\mathcal{M} \models \rho(t_1,...,t_k)$ for some $t_1,...,t_k\in |\mathcal{M}|$ and some complete Herbrand model $\mathcal{M}$, then we have $\mathcal{M} \models \rho(t_1,...,t_k)$ for all complete Herbrand models $\mathcal{M}$, and so by definition 3.7, for all $\mathcal{M}$, we have $\mathcal{M} \models A(t_1,...,t_k)\leftrightarrow B(t_1,...,t_k)$. Now suppose $d(A(t_1,...,t_k),B(t_1,...,t_k)) \neq 0$, then $\mathcal{M} = \{A(t_1,...,t_k)\}$ does not satisfy $B(t_1,...,t_k)$, which is a contradiction.

$2\Rightarrow 1$: suppose $d(A(t_1,...,t_k),B(t_1,...,t_k)) = 0$;

assume $A=p(u_1(x),...,u_n(x))$ and $B=p(v_1(x),...,v_n(x))$ then $d(u_i(t_1,...,t_k),v_i(t_1,...,t_k))=0$ for all i, and hence $\rho'=\{u_1=v_1,...,u_n=t_n\}$ is consistent (since it has a complete Herbrand model). Since ρ' is a cu for {A,B}, and ρ is an mcu, it follows by definition 3.8 that $\mathcal{M} \models \rho(x_1,...,x_k)[x_1/t_1,...,x_k/t_k]$ □

4. COMPLETE SLD-RESOLUTION

In this section we will assume P to be some fixed program, with at least one constant symbol in its language L_P. Specifications will be denoted by $\sigma,\rho,\tau,\ldots$. For convenience, we present a few notations in the following definition.

definition 4.1 For specifications $\sigma(x),\rho(x)$ with variables $x\equiv x_1,\ldots,x_k$ and for arbitrary complete terms $u \equiv u_1,\ldots,u_k$ we write:

$u\in\sigma :\Leftrightarrow$ for all complete Herbrand models $\mathcal{M}$ for L_P: $\mathcal{M} \models \sigma[u]$

$\sigma\equiv\rho :\Leftrightarrow \models \forall(\sigma\leftrightarrow\rho)$

$\sigma\leq\rho :\Leftrightarrow \models \forall(\sigma\rightarrow\rho)$ ('σ is more specific than ρ')

$\sigma\equiv\bot :\Leftrightarrow$ σ is inconsistent ($\bot$ stands for the bottom in the ordering $\leq$)

$\rho\equiv mcu(\{A,B\}) :\Leftrightarrow$ ρ is an mcu for A and B.

definition 4.2 Let σ and ρ be two specifications. Then we define $\sigma\cdot\rho := \sigma\cup\rho$. Again we write $\sigma\cdot\rho\equiv\bot$ if $\sigma\cdot\rho$ is inconsistent. Moreover, we often write $\sigma\rho$ instead of $\sigma\cdot\rho$.

proposition 4.1 One can easily check the following statements:

(i) $(\emptyset\cdot\sigma) \equiv \sigma$, $(\bot\cdot\sigma) \equiv \bot$

(ii) $(\sigma\cdot\rho) \equiv (\rho\cdot\sigma)$

(iii) $((\sigma\cdot\rho)\cdot\tau) \equiv (\sigma\cdot(\rho\cdot\tau))$

(iv) $\sigma\leq\rho \Leftrightarrow$ for some τ: $\sigma\equiv(\rho\cdot\tau)$.

(v) if $\rho(x)$ is ground (i.e.: has no free variables) and $\sigma(x,y)$, $\rho(x,z)$ are specifications, then: $\rho\sigma\tau(x,y,z) \equiv \bot \Leftrightarrow \rho\sigma(x,y) \equiv \bot \vee \rho\tau(x,z) \equiv \bot$.

The proof of proposition 4.1 is easy and left to the reader. Taking $\rho=\emptyset$, it follows from (v) that $\sigma(y)\cdot\tau(z)\equiv\bot$ if and only if $\sigma(y)\equiv\bot$ or $\tau(z)\equiv\bot$.

Next, we will make straightforward adaptations to some well-known definitions.

definition 4.3 A *goal* is a pair (G,σ) where G is a goal clause and σ is a specification.

definition 4.4 A *computation rule* (c-rule, for short) is a function R from goals $(\leftarrow A_1,\ldots,A_k,\sigma)$ to atoms A, such that $A\in\{A_1,\ldots,A_k\}$.

definition 4.5 Let G_i be the goal $(\leftarrow A_1,\ldots,A_m,\ldots,A_k,\sigma)$ and $C_{i+1}= A\leftarrow B_1,\ldots,B_q$ a clause and R a c-rule. Now, G_{i+1} is *derived* from G_i and C_{i+1} by R and ρ, if:

(i) $R(G_i)=A_m$

(ii) $\rho\equiv mcu(\{A_m,A\})$

(iii) $G_{i+1} = (\leftarrow A_1,\ldots,A_{m-1},B_1,\ldots,B_q,A_{m+1},\ldots,A_k,\sigma\rho)$.

definition 4.6 Let G be a goal and R a c-rule.

A *CSLD-derivation* for $P\cup\{G_0,\sigma_0\}$ is a sequence $(G_0,\sigma_0),...,(G_k,\sigma_k),...$, such that for some sequence of program clauses $C_1,...,C_k,...\in P$ with new variables, G_{i+1} is derived from G_i and C_{i+1} by R, for all i. Moreover, if $\sigma_k\equiv\bot$ then (G_k,σ_k) is the last goal in the derivation.

A *CSLD-refutation* is a CSLD-derivation with $(\square,\rho)$ as the last goal, where $\square$ stands for the empty goal clause, and ρ is a consistent specification which is called the *computed answer specification* for $P\cup\{G_0,\sigma_0\}$ with c-rule R.

definition 4.7 A *correct answer specification* for $P\cup\{\leftarrow A_1,...,A_k,\rho)$ is a consistent specification σ such that : $P\models\forall(\sigma\rightarrow A_1\wedge...\wedge A_k\wedge\rho)$.

CSLD-derivation stands for *complete* SLD-derivation. Note that such derivations are logical derivations as well, as is stated in the next soundness theorem, originally due to VAN EMDEN & LLOYD [6].

theorem 4.2 (soundness of CSLD-resolution)

Computed answer specifications are correct.

proof Let $(G_i,\sigma_i)_{0\le i\le n}$ be a CSLD-refutation for $P\cup(\leftarrow A_1,...,A_k,\sigma_0)$. By induction we prove:
(induction hypothesis) $P\models\forall(\sigma_n\rightarrow A_1\wedge...\wedge A_k\wedge\sigma_0)$.

n=1: Now G_0 is of the form $(\leftarrow A_1)$ and P has a *unit* clause $A\leftarrow$ such that $\sigma_0\rho$ is consistent, where $\rho\equiv mcu(\{A_1,A\})$. By theorem 3.13 we have $\models\forall(\rho\rightarrow(A_1\leftrightarrow A))$ and since $\sigma_1\equiv\sigma_0\rho\le\rho$ (see proposition 4.1) we find $\models\forall(\sigma_1\rightarrow(A_1\leftrightarrow A))$. Since $(A\leftarrow)\in P$, we have $P\models\forall(A)$ and therefore $P\models\forall(\sigma_1\rightarrow A_1)$, hence $P\models\forall(\sigma_1\rightarrow A_1\wedge\sigma_0)$, since $\sigma_1\le\sigma_0$.

n+1: Assume $R(G_0)=A_i$, then there is a clause $A\leftarrow B_1,...,B_q\in P$ such that $\sigma_1\equiv\sigma_0\rho$ is consistent, where $\rho\equiv mcu(\{A_i,A\})$ (see definitions 4.5, 4.6). Now $(G_1,\sigma_1),...,(G_{n+1},\sigma_{n+1})$ is a CSLD-refutation for $P\cup\{(\leftarrow A_1,...,A_{i-1},B_1,...,B_q,A_{i+1},...,A_k,\sigma_1)$ with length n, so by the induction hypothesis we obtain:

(i) $P\models\forall(\sigma_{n+1}\rightarrow A_1\wedge...\wedge A_{i-1}\wedge B_1\wedge...\wedge B_q\wedge A_{i+1}\wedge...\wedge A_k\wedge\sigma_1)$

Since $\sigma_1\le\sigma_0$ we directly find

(ii) $P\models\forall(\sigma_{n+1}\rightarrow A_1\wedge...\wedge A_{i-1}\wedge A_{i+1}\wedge...\wedge A_k\wedge\sigma_0)$.

Since $A\leftarrow B_1\wedge...\wedge B_q\in P$, we have $P\models\forall(B_1\wedge...\wedge B_q\rightarrow A)$. By definition 3.7 it follows that $\models\forall(\rho\rightarrow(A_i\leftrightarrow A))$, hence $P\models\forall(B_1\wedge...\wedge B_q\wedge\rho\rightarrow A_i)$. So from $\sigma_{n+1}\le\sigma_0\rho\le\rho$ and (i) we find that $P\models\forall(\sigma_{n+1}\rightarrow A_i)$. Hence, with (ii): $P\models\forall(\sigma_{n+1}\rightarrow A_1\wedge...\wedge A_k\wedge\sigma_0)$. $\square$

Next we will present a method, first introduced by VAN EMDEN & KOWALSKI [5], to find models for logic programs, using fixed points of some monotonic mapping. As is pointed out in [10], such a mapping can be defined on the complete Herbrand universe as well.
In the following finite atoms are denoted by A(x).

definition 4.8 For $X \subseteq B_P$ we define

$T_P(X) := \{A(u) \in B_P$: there are terms $v \in U_P$ and a clause $A_1(x) \leftarrow B_1(x),...,B_q(x) \in P$ such that $d(A(u),A_1(v))=0$ and $\{B_1(v),...,B_q(v)\} \subseteq X\}$

For $X \subseteq CB_P$ we define

$T'_P(X) := \{A(u) \in CB_P$: there are terms $v \in CU_P$ and a clause $A_1(x) \leftarrow B_1(x),...,B_q(x) \in P$ such that $d(A(u),A_1(v))=0$ and $\{B_1(v),...,B_q(v)\} \subseteq X\}$

T_P and T'_P are *monotonic* mappings on the complete lattices formed by all subsets of B_P and CB_P respectively, with the usual ordering $\subseteq$. $T_P(X)$ (or $T'_P(X)$) stands for the set of all ground (complete) atoms that may be concluded from X, using the rules of P.

definition 4.9 We will use the following notation:

$T_P\uparrow 0 = \emptyset$ $\qquad$ $T'_P\uparrow 0 = \emptyset$

$T_P\uparrow k+1 = T_P(T_P\uparrow k)),\ k \in \omega$ $\qquad$ $T'_P\uparrow k+1 = T'_P(T'_P\uparrow k)),\ k \in \omega$

$T_P\uparrow \omega = \bigcup_{k<\omega} T_P\uparrow k$ $\qquad$ $T'_P\uparrow \omega = \bigcup_{k<\omega} T'_P\uparrow k$

A well-known theorem says that $T_P\uparrow\omega$ and $T'_P\uparrow\omega$ are the *least fixed points* of T_P and T'_P, denoted by $lfp(T_P)$ and $lfp(T'_P)$ respectively. Moreover, it can be proved that for all ground atoms A, $A \in T_P\uparrow\omega$ if and only if A is a logical consequence from P. This theorem can be translated to the complete case as we will see later on.

proposition 4.3 Let P be a program and $\mathcal{M} \subseteq CB_P$ a complete Herbrand model, then:

$\mathcal{M}$ is a model for P $\Leftrightarrow$ $T'_P(\mathcal{M}) \subseteq \mathcal{M}$.

proof Clearly $\mathcal{M}$ is a model for P if and only if for all clauses $A(x) \leftarrow B_1(x),...,B_q(x) \in P$ and all complete terms u we have: $B_1(u),...,B_q(u) \in \mathcal{M}$ implies $A(u) \in \mathcal{M}$, i.e.: $T'_P(\mathcal{M}) \subseteq \mathcal{M}$. □

proposition 4.4 Let P be a program, then $lfp(T'_P) = \cap\{\mathcal{M} \subseteq CB_P$: $T'_P(\mathcal{M}) \subseteq \mathcal{M}\}$.

Proposition 4.4 is a special case of a general theorem in the theory of complete partial orders. Its proof is omitted here (see [10]).

corollary 4.5

(i) $T'_P\uparrow\omega$ is a model for P

(ii) $T'_P\uparrow\omega = \cap\{\mathcal{M} \subseteq CB_P$: $\mathcal{M}$ is a model for P$\}$.

Corollary 4.5 can be proved immediately from the propositions 4.3 and 4.4 respectively. It follows that $T'_P\uparrow\omega$ is the *least complete Herbrand model* for P.

Next we will return to the notion of CSLD-resolution. We will see that the set of complete atoms

with a CSLD-refutation coincides with the least fixed point $T'_P\uparrow\omega$. First we present an important lemma.

lemma 4.6 (mcu-lemma)

Let G be a goal en assume $P\cup\{G_0,\sigma_0\}$ has a CSLD-refutation $(G_i,\sigma_i)_{0\le i\le n}$ with $\sigma_i\equiv\sigma_0\theta_1\cdots\theta_i$, where $\theta_1,\ldots,\theta_i$ are complete unifiers, but not necessarily *most general* complete. Then there is a CSLD-refutation $(G'_i,\sigma'_i)_{0\le i\le n}$ of the same length, such that:

(i) $\sigma'_i\equiv\sigma_0\theta'_1\cdots\theta'_i$

(ii) $\theta'_1,\cdots,\theta'_i$ are most general complete unifiers

(iii) $\sigma_n\le\sigma'_n$.

The proof is an analogon of the proof given in [10]. In fact the mcu-lemma states that refutations with only complete unifiers, can be turned into a more general refutation by using most general complete unifiers. Now we can present the definition of the complete success set (see [1]):

definition 4.10 The *complete success set* CS_P of a program P is defined by

$CS_P = \{A(u)\in CB_P$: $P\cup\{(\leftarrow A(x),\emptyset)\}$ has a CSLD-refutation with computed answer specification $\sigma(x,y)$, such that for some v: $(u,v)\in\sigma\}$.

proposition 4.7 CS_P is well-defined.

proof Suppose $A_1(u)\equiv A_2(w)$ and $A_1(u)\in CS_P$. We will prove that $A_2(w)\in CS_P$.

Since $A_1(u)\in CS_P$, $(\leftarrow A_1(x),\emptyset)$ has a CSLD-refutation $(G_i,\sigma_i)_{0\le i\le n}$ with computed answer specification $\sigma_n(x,y)$ such that for some v: $(u,v)\in\sigma_n$. Let $\rho(x,z)\equiv mcu(\{A_1(x),A_2(z)\})$, then ρ is consistent and $(u,w)\in\rho$. Clearly $(u,v,w)\in\rho\sigma_n(x,y,z)$ and therefore $\rho\sigma_n$ is consistent and $(G_i,\rho\sigma_i)_{0\le i\le n}$ is a refutation for $P\cup\{\leftarrow A_2(z),\rho\}$ (using the same clauses and the same computation rule).

Now consider $(\leftarrow A_2(z),\emptyset)$, $(G_1,\rho\sigma_1)$, $(G_2,\rho\sigma_2)$,..., $(G_n,\rho\sigma_n)$ then clearly this is a refutation for $P\cup\{\leftarrow A_2(z),\emptyset\}$ except that $\rho\sigma_1$ is not a *most general* complete unifier. So by lemma 4.6 there is a refutation for $P\cup\{\leftarrow A_2(z),\emptyset\}$ with computed answer specification $\sigma(x,y,z)\ge\rho\sigma_n(x,y,z)$ and therefore $(u,v,w)\in\sigma$. Hence $A_2(w)\in CS_P$. □

Note that the *success set* S_P of a program can be defined as the set all finite elements of CS_P.

theorem 4.8 $CS_P = T'_P\uparrow\omega$.

proof $\Rightarrow$: Let $A(u)\in CS_P$ then $P\cup\{(\leftarrow A(x),\emptyset)\}$ has a CSLD-refutation with answer specification $\sigma(x,y)$, such that for some v: $(u,v)\in\sigma$. By soundness of CSLD-resolution (see theorem 4.2) it follows that $\sigma(x,y)$ is correct for $P\cup\{(\leftarrow A(x),\emptyset)\}$, therefore $P\models\forall(\sigma(x,y)\rightarrow A(x))$ and hence $\mathcal{M}\models A(x)[x/u]$ for all complete Herbrand models $\mathcal{M}$ for P, since $(u,v)\in\sigma$. Then by corollary

4.5 it follows that $A(u) \in T'_P \uparrow \omega$.

$\Leftarrow$: Let $A(u) \in T'_P \uparrow \omega$, for some atom $A(x)$ and complete terms $u \equiv u_1,...,u_k$, then $A(u) \in T'_P \uparrow n$, for some $n < \omega$. Now by induction:

n=1: then $A(u) \in T'_P(\emptyset)$, so there is a unit clause $A_1(y) \leftarrow \in P$, such that for some v, $(u,v) \in \rho(x,y)$ and $\rho(x,y) \equiv mcu(\{A(x),A_1(y)\})$; since ρ is consistent, $(\leftarrow A(x), \emptyset)$, $(\square, \rho(x,y))$ is a CSLD-refutation for $P \cup \{\leftarrow A(x), \emptyset\}$ with answer specification $\rho(x,y)$.

n+1: now, $A(u) \in T'_P(T'_P \uparrow n)$, so there is a clause $A_1(y) \leftarrow B_1(y),...,B_q(y) \in P$ such that for some v: $A(u) \equiv A_1(v)$ and $B_1(v),...,B_q(v) \in T'_P \uparrow n$. Writing $\rho(x,y) \equiv mcu(\{A(x),A_1(y)\})$ we directly have $(u,v) \in \rho(x,y)$.

By the induction hypothesis, there exist refutations for $P \cup \{\leftarrow B_i(y), \emptyset\}$ with computed answer specification $\sigma_i(y,z_i)$ such that for some w_i: $(v,w_i) \in \sigma_i$. These refutations can be put together to obtain a new refutation for $P \cup \{\leftarrow B_1(y),...,B_q(y), \emptyset\}$ with a computed answer specification $\sigma \geq \sigma_1 \cdots \sigma_q$. Since $(v,w_1,...,w_q) \in \sigma_1 \cdots \sigma_q$, such a consistent σ exists, and we directly find that $\sigma\rho$ is consistent as well, since $(u,v,w_1,...,w_q) \in \sigma\rho$. Hence $\sigma\rho$ is the answer specification we were looking for. $\square$

So far, we found that the correctness theorem can be restored for CSLD-resolution. Moreover, $T'_P \uparrow \omega$ is the least complete Herbrand model which is again the intersection of all complete Herbrand models for P, and equal to the complete success set.
Next we will show we have a completeness theorem for CSLD-resolution as well.

definition 4.11 (restriction)

Let $\sigma(x,y)$ and $\rho(x,z)$ be two specifications, then $\sigma \leq_x \rho$, or $\sigma \leq \rho$ *with respect to the variables x*, if: $\models \forall xy.(\sigma(x,y) \rightarrow \exists z.\rho(x,z))$.

We denote $\sigma \equiv_x \rho$, or say σ and ρ are *equivalent with respect to x*, if both $\sigma \leq_x \rho$ and $\rho \leq_x \sigma$.

Definition 4.11 is needed to indicate that σ is more specific than ρ, although ρ may bind variables not occurring in σ. Moreover, $\sigma \equiv_x \rho$ indicates that σ and ρ are equivalent with respect to the variables x. Finally note that $\sigma \equiv_x \perp \Rightarrow \sigma \equiv \perp$ for all variables x.

proposition 4.9 Let $\sigma(x)$ be ground then for all specifications ρ: either $\sigma\rho \equiv \perp$ or $\sigma \leq_x \rho$.

So, *ground* specifications cannot be further specified with respect to their variables: either $\sigma \cup \rho$ is inconsistent, or σ is more specific than ρ with respect to x. The proof is left to the reader.

example Let $\sigma(x) = \{x = f(x,x)\}$ and $\rho(x,y) = \{x = f(x,y), y = x\}$ then $\sigma \leq_x \rho$, however not $\sigma \leq \rho$ (i.e.: $\sigma \leq_{xy} \rho$), since ρ has an extra variable y. In fact: $\sigma \equiv_x \rho$.

Next we prove two lemmas.

lemma 4.10 Let $A(x)$ be an atom and $\sigma(x)$ a correct specification for $P\cup\{\leftarrow A(x),\emptyset\}$, then there exists a CSLD-refutation for $P\cup\{\leftarrow A(x),\sigma(x)\}$ with $\rho(x,y)$ as computed answer specification, such that $\sigma\equiv_x\rho$.

proof First, assume $\sigma(x)$ is ground. Now, let $u\in\sigma$ for some complete terms $u\equiv u_1,...,u_m$, then $A(u)\in T'_P\uparrow\omega$ (by theorem 3.7 and corollary 4.5). Therefore $A(u)\in CS_P$ (by theorem 4.8), hence there is a CSLD-refutation $(G_i,\rho_i)_{0\le i\le n}$ for $P\cup\{\leftarrow A(x),\emptyset\}$ with computed answer specification $\rho_n(x,y)$ such that for some v: $(u,v)\in\rho_n$. Because $(u,v)\in\sigma\rho_n$, $\sigma\rho_n$ is consistent and therefore $(G_i,\sigma\rho_i)_{0\le i\le n}$ is a refutation for $P\cup\{\leftarrow A(x),\sigma\}$ with computed answer specification $\sigma\rho_n$, and by proposition 4.1 we have $\sigma\rho_n\le\sigma$. Since σ is ground we find by proposition 4.9: $\sigma\le_x\sigma\rho_n$. Hence $\sigma\equiv_x\sigma\rho_n$.
Next, assume σ is not ground, and let $x=(y,z)$ where y and $z\equiv z_1,...,z_k$ respectively are the bound and free variables of σ. Let $a\equiv a_1,...,a_k$ be *new* constants not occurring in P,A or σ, and such that for i,j: $a_i=a_j \Leftrightarrow \sigma\vdash z_i=z_j$. Next, consider $\sigma'(y,z)\equiv\{z_1=a_1,...,z_n=a_n\}\cdot\sigma(y,z)$, then it is easily proved that σ' is consistent and ground. Hence there exists a CSLD-refutation for $P\cup\{\leftarrow A,\sigma'(y,z)\}$ with computed answer specification $\rho'(y,z,z')$ such that $\sigma'\equiv_{yz}\rho'$, or equivalently: $\sigma'\equiv_x\rho'$. Now it is easy to see, that we can find a new refutation for $P\cup\{\leftarrow A,\sigma\}$ by replacing all constants a by new variables, with computed answer specification ρ, such that $\sigma\equiv_x\rho$. □

Note that lemma 4.10 does not hold if we replace $\equiv_x$ by $\equiv$.

example Let P: $A(y,y)\leftarrow$.
Consider the goal clause $G=\leftarrow A(x,f(x))$ then one can easily see that $\sigma(x)=\{x=f(x)\}$ is correct for $P\cup\{G,\emptyset\}$. Indeed, there is a computed answer specification $\rho(x,y)=\{x=\{y,f(y)\}\}$ which is equivalent to $\rho'(x,y)=\{x=y,\ y=f(y)\}$. Clearly $\rho\le\sigma$, however the converse is not true, since: $\not\vdash \forall(x=f(x)\rightarrow(x=y\wedge y=f(y)))$. Hence $\rho\not\equiv\sigma$.

The point is, that in a CSLD-refutation new variables are introduced (input clauses have new variables), and the correct specification we started with cannot impose any constraints upon variables other than its own. In 'common' SLD-resolution, this problem does not occur since computed substitutions are restricted to the goal variables automatically. This can be done, because in SLD-resolution new variables are bound to finite terms (not containing the variable again), hence one can simply carry out the substitution. For instance, restricting all substitutions to the variable x, we have $\{x/f(y)\}\cdot\{y/g(z)\}=\{x/f(g(z))\}$. This procedure fails, however, if the occur check is left out: $\{x/f(y)\}\cdot\{y/g(y)\}$ cannot be restricted to x only. This subsitution problem is overcome, however, by introducing $\le_x$ as a logical notion of restriction.

example Let P: $test\leftarrow p(x,x)$

p(y,f(y)) ←.

Clearly, $\sigma(x,y)=\{x=f(x),y=x\}$ is a computed answer specification for $P\cup\{\leftarrow test,\emptyset\}$ which cannot be restricted to the goal variables. Note, that test is not a logical consequence of P, although it is of $P\cup\sigma$.

lemma 4.11 (lifting lemma)

Let G be a goal clause and σ be a specification.

Assume there exists a CSLD-refutation for $P\cup\{G,\sigma\}$ with computed answer specification ρ. If $\sigma\leq\sigma'$ then there is a CSLD-refutation for $P\cup\{G,\sigma'\}$ with computed answer specification ρ' of the same length such that $\rho\leq\rho'$.

proof Assume $\sigma\equiv\sigma'\tau$, for some specification τ. Suppose the first input clause of the refutation for $P\cup\{G,\sigma\}$ is C_1, the first mcu is θ_1 and G_1 is the goal that results from the first step. Clearly, we can transform this refutation into a new refutation for $P\cup\{G_i,\sigma'\}$, except that the original goal is different, of course, and the complete unifier is $\tau\theta_1$ instead of θ_1. Now use the mcu-lemma (lemma 4.6). □

Lemma 4.11 actually states that once starting a refutation with a more general goal, the computed answer specification will be more general as well. Next, we finally prove the completeness theorem for CSLD-resolution:

theorem 4.12 (completeness of CSLD-resolution)

Let $(G_0(x),\sigma_0(x))$ be a goal and $\sigma(x)$ a correct answer specification for $P\cup\{G_0(x),\sigma_0(x)\}$ then there exists a computation rule R and a R-computed answer specification $\rho(x,y)$ for $P\cup\{G_0(x),\sigma_0(x)\}$ such that $\sigma\leq_x\rho$.

proof Assume $G_0=(\leftarrow A_1(x),...,A_k(x))$ then $P\models\forall(\sigma\rightarrow A_1(x)\wedge...\wedge A_k(x)\wedge\sigma_0)$ since σ is correct for $P\cup\{G_0,\sigma_0\}$. By lemma 4.10 there exist refutations for $P\cup\{\leftarrow A_i,\sigma\}$ with ρ_i as computed answer specification, such that $\rho_i\equiv_x\sigma$ for all i. These refutations can be combined to obtain a new refutation for $P\cup\{\leftarrow A_1(x),...,A_k(x),\sigma\}$ with answer specification $\rho'\equiv\rho_1\cdots\rho_k$, so $\sigma\equiv_x\rho'$. Since $\sigma\leq\sigma_0$, it follows by the lifting lemma that there exists a CSLD-refutation for $P\cup\{\leftarrow A_1(x),...,A_k(x),\sigma_0\}$ with computed answer specification ρ such that $\rho'\leq\rho$. Since $\sigma\equiv_x\rho'$ we have $\sigma\leq_x\rho$. □

It is important to understand how resolution with specifications works. In fact, a computed answer specification can be looked at as an extra condition that needs to be satisfied before a given conclusion may be drawn from P. The completeness theorem simply states that from a logic program all such sufficient conditions can be generated. For instance, consider the following example, taken from [11]:

example Let P: smaller(x,S(x)) ←
smaller(1,0) ← smaller(S(y),y).
Then $\sigma(x)=\{x=S(x)\}$ is correct for $P\cup\{\leftarrow smaller(1,0),\emptyset\}$, since $P\models\forall(x=S(x)\rightarrow smaller(1,0))$. However, σ cannot be computed since any computed answer specification ρ contains at least two variables. Clearly, $\rho(x,y)=\{x=S(y),y=S(x)\}$ can be computed, and $\sigma\equiv_x\rho$.

The example illustrates, that in theorem 4.12, we cannot replace $\leq_x$ by $\leq$, since Prolog will compute answer specifications for $P\cup\{\leftarrow smaller(1,0),\emptyset\}$ having more variables. Note, that P can be considered a set of statements about natural numbers. Here, S stands for the successor function and the predicate smaller(x,y) represents the binary predicate $<$. Both clauses hold in the usual model for natural numbers: the first clause actually 'defines' the predicate, whereas the second states that one could derive $1<0$ if one is able to derive $S(y)<y$ for some y (using induction).
Clearly, $P\cup\{smaller(1,0),\emptyset\}$ has a correct answer specification $\sigma(x,y)=\{x=S(y), y=S(x)\}$, which is ground and consistent and which provides us with a sufficient condition to conclude smaller(1,0) from the program. In the intended model for P however, σ is not satisfiable so we may not conclude smaller(1,0) to hold in this model.
The completeness theorem leads to the following corollary:

corollary 4.13 Let A(x) be a atom and $u\equiv u_1,\ldots,u_k$ complete terms, then:
$A(u)\in CS_P \Leftrightarrow$ for some $\rho(x,y)$ and some v: $(u,v)\in\rho$ and $P\models\forall(\rho(x,y)\rightarrow A(x))$.

proof $\Rightarrow$: Suppose $A(u)\in CS_P$, then there exists some CSLD-refutation for $P\cup\{\leftarrow A(x),\emptyset\}$ with computed answer specification $\rho(x,y)$ such that for some v: $(u,v)\in\rho(x,y)$. By theorem 4.2 ρ is correct for $P\cup\{\leftarrow A(x),\emptyset\}$, hence $P\models\forall(\rho(x,y)\rightarrow A(x))$.
$\Leftarrow$: Suppose for some $\rho(x,y)$ and some v: $(u,v)\in\rho$ we have: $P\models\forall(\rho(x,y)\rightarrow A(x))$. Then $\rho(x,y)$ is a correct answer specification for $P\cup\{\leftarrow A(x),\emptyset\}$, thus by theorem 4.12 there exists a computation rule R and a R-computed answer specification $\sigma(x,y,z)$ for $P\cup\{\leftarrow A(x),\emptyset\}$ such that $\rho\leq_{xy}\sigma$. By definition 4.11 we find $\models\forall xy.(\rho(x,y)\rightarrow\exists z.\sigma(x,y,z))$. Hence, $(u,v,w)\in\sigma$ for some w and therefore by definition 4.10, $A(u)\in CS_P$. □

5. FINITE FAILURE

In this section we will consider the *negation as failure rule*, for CSLD-resolution. It turns out that all 'classical' results can be restored; even better: it seems that working in CSLD-semantics can simplify some theoretical constructions. Let us start with some definitions.

definition 5.1 Let G be a goal. A *CSLD-tree* for $P \cup \{G\}$ with c-rule R, is defined by

(i) every node of the tree is a goal

(ii) G is the root

(iii) if $G'=(\leftarrow A_1,...,A_m,...,A_k,\sigma)$ for some consistent specification σ, and $R(G')=A_m$ then G' has a *successor* $(\leftarrow A_1,...,A_{m-1},B_1,...,B_q,A_{m+1},...,A_k,\sigma\rho)$ for every clause $A\leftarrow B_1,...,B_q \in P$, where $\rho \equiv mcu(\{A_m,A\})$; if σ is inconsistent, then G' has no successor goals.

definition 5.2 A *success branch* in a CSLD-tree is a branch that ends with $(\square,\sigma)$ for some consistent specification σ.

A *failure branch* is a branch $(G_i,\sigma_i)_{0\leq i\leq k}$ such that $\sigma_k \equiv \bot$.

definition 5.3 A *finitely failed CSLD-tree*, or *ff-tree* for short, for $P \cup \{G\}$ is a finite CSLD-tree with only failure branches.

definition 5.4 We will use the following notation:

$$T_P\downarrow 0 = B_P \qquad T'_P\downarrow 0 = CB_P$$

$$T_P\downarrow k+1 = T_P(T_P\downarrow k)),\ k\in\omega \qquad T'_P\downarrow k+1 = T'_P(T'_P\downarrow k)),\ k\in\omega$$

$$T_P\downarrow \omega = \bigcap_{k<\omega} T_P\downarrow k \qquad T'_P\downarrow \omega = \bigcap_{k<\omega} T'_P\downarrow k$$

A well-known theorem says that $T'_P\downarrow\omega$ is the *greatest fixed point*, denoted by $gfp(T'_P)$, of the continuous mapping T'_P. Note, that $T_P\downarrow\omega$ not necessarily is the greatest fixed point of T_P, as can be shown by the following example from [10]:

example Let P: $p(f(x))\leftarrow p(x)$

$q(a)\leftarrow p(x)$.

Then $T'_P\downarrow\omega = \{q(a),p(f^\omega)\} = gfp(T'_P)$. However, $T_P\downarrow\omega = \{q(a)\}$ whereas $gfp(T_P) = \emptyset$.

lemma 5.1 Let A(x) be an atom and $u\equiv u_1,...,u_k$ be complete terms, then

$A(u)\in T'_P\downarrow k+1 \Leftrightarrow$ there is a clause $A_1(y)\leftarrow B_1(y),...,B_q(y)\in P$, such that for some v:

$(u,v)\in mcu(\{A(x),A_1(y)\})$ and $B_1(v),...,B_q(v)\in T'_P\downarrow k$.

proof $\Rightarrow$: Suppose $A(u)\in T'_P\downarrow k+1$, then there is a clause $A_1(y)\leftarrow B_1(y),...,B_q(y)\in P$, such that for some v: $A(u)\equiv A_1(v)$ and $B_1(v),...,B_q(v)\in T'_P\downarrow k$. Hence $(u,v)\in mcu(\{A(x),A_1(y)\})$.

$\Leftarrow$: by definition of T'_P. $\square$

Note that lemma 5.1 does not hold for ordinary substitutions.

example Let P: $A(x,f(x))\leftarrow$.
Then clearly $A(f^{\omega},f^{\omega})\in T'_P{\downarrow}k+1$ for all k, but A(x,f(x)) and A(y,y) cannot be unified because of the occur check.

theorem 5.2 Suppose R is a *fair* c-rule, i.e.: every atom in a goal clause is selected somewhere in the CSLD-tree, and assume $(G_i,\sigma_i)_{i\in\omega}$ is an infinite CSLD-derivation for $P\cup\{G_0,\sigma_0\}$ by R with $G_0=(\leftarrow A_1(x),...,A_n(x),\sigma_0)$. If u are complete terms, then:

$$\forall k\ \exists i.\ (u,v)\in\sigma_i(x,y) \Rightarrow A_1(u),...,A_n(u)\in T'_P{\downarrow}k.$$

proof Let $k\in\omega$, and u complete terms. Let $(G_i,\sigma_i)_{i\in\omega}$ be an infinite derivation for $P\cup(G_0,\sigma_0)$. Now by induction on k.

k=0: Immediately.

k+1: Let $1\leq j\leq n$ and let $m\in\omega$ such that $R(G_m)=A_j$. This m exists because R is fair. Then there is some clause $A(y)\leftarrow B_1(y),...,B_q(y)$ such that $\rho(x,y)\equiv mcu(\{A_j(x),A(y)\})$ is consistent and $\sigma_{m+1}\equiv\sigma_m\cdot\rho$. Assume $G_m=(\leftarrow C_1,...,A_j,...,C_r)$ then $G_{m+1}=(\leftarrow C_1,...,B_1,...,B_q,...,C_r)$ and clearly $(G_{m+i},\sigma_{m+i})_{i\geq 1}$ is an infinite derivation for $P\cup(G_{m+1},\sigma_{m+1})$. By the induction, let $i'\in\omega$ such that $(u,v)\in\sigma_{i'}(x,y) \Rightarrow C_1(u,v),...,B_1(v),...,B_q(v),...,C_r(u,v)\in T'_P{\downarrow}k$, then it follows by definition of T'_P that for all $(u,v)\in\sigma_{i'}(x,y) \Rightarrow A(v)\in T'_P{\downarrow}k+1$. Since $\sigma_{i'}\leq\rho$ and $\rho(x,y)\equiv mcu(\{A_j(x),A(y)\})$, we have $A_j(u)\in T'_P{\downarrow}k+1$. So, for every j such an index i' exists. Now take the maximum of all n indices. □

The following lemmas are easy to prove:

lemma 5.3 If (G,σ) has an ff-tree with depth $\leq k$, then $(G,\sigma\rho)$ has an ff-tree with depth $\leq k$.

lemma 5.4 If for some *ground* specification $\sigma(x)$, the goal $(\leftarrow A_1(x),...,A_n(x),\sigma(x))$ has an ff-tree with depth $\leq k$, then for some $i\leq n$: $(\leftarrow A_i(x),\sigma(x))$ has an ff-tree with depth $\leq k$.

proof By induction on n.

n=1: immediately.

n+1: assume $A_{n+1}(x)$ has no ff-tree with depth $\leq k$, then for all c-rules R, $P\cup\{A_{n+1}(x),\sigma(x)\}$ has a CSLD-derivation of length $\geq k+1$.

Let R be a c-rule such that $P\cup\{\leftarrow A_1(x),...,A_{n+1}(x),\sigma(x)\}$ has an ff-tree with depth $\leq k$, and let $(G_i,\rho_i(x,y_i))_{0\leq i\leq k+1}$ be a derivation for $P\cup\{A_{n+1}(x),\sigma(x)\}$ via R with length $\geq k+1$, then for all derivations $(G'_i,\tau_i(x,z_i))_{0\leq i\leq m}$ for $P\cup\{\leftarrow A_1(x),...,A_n(x),\sigma(x)\}$ we have (by proposition 4.1(v) and 4.9) that for all i,j: $\tau_i\rho_j\equiv\bot \Rightarrow \tau_i\equiv\bot \vee \rho_i\equiv\bot$, since $\sigma(x)$ is ground and $\tau_0=\rho_0=\sigma$. Then, there is a finitely failed derivation $(G_i,\theta_i)_{0\leq i\leq q}$ via R for $P\cup\{\leftarrow A_1(x),...,A_{n+1}(x),\sigma(x)\}$

such that $\theta_i \geq \tau_i\rho_i$ and $q \leq k$ (since all derivations via R are finitely failed with length $\leq k$) we find that $\tau_q\rho_q \equiv \bot$. Because ρ_q is consistent for all $q \leq k$ we have $\tau_q \equiv \bot$.
Therefore, all derivations for $P \cup \{\leftarrow A_1(x),...,A_n(x),\sigma(x)\}$ are finitely failed with length $\leq k$, hence $P \cup \{\leftarrow A_1(x),...,A_n(x),\sigma(x)\}$ has an ff-tree with depth $\leq k$. Now by induction. □

Note that in lemma 5.4 σ needs to be ground, as is shown in the following example:

example Let P: $A(a)\leftarrow$
$B(b)\leftarrow$.
Now $((\leftarrow A(x),B(x)),\emptyset)$ has an ff-tree with depth ≤ 2 whereas $(\leftarrow A(x),\emptyset)$ and $(\leftarrow B(x),\emptyset)$ do not even have an ff-tree.

Next we establish an important theorem.

theorem 5.5 If $A_1(u),...,A_n(u) \in T'_P\downarrow k$ for some complete terms $u \in \sigma(x)$, then there exists a *ground* specification $\rho(x,y) \leq \sigma(x)$ such that for $(v,w) \in \rho(x,y)$: $A_1(v),...,A_n(v) \in T'_P\downarrow k$.

proof By induction on k. Assume $u \in \sigma(x)$, for some specification σ.
k=0: Immediately, since any ground $\rho \leq \sigma$ suffices.
k+1: Suppose $A_1(u),...,A_n(u) \in T'_P\downarrow k+1$ then by definition of T'_P there exist clauses $A^i(y_i)\leftarrow B^i{}_1(y_i),...,B^i{}_q(y_i) \in P$ for all $i \leq n$, such that with $\rho^i(x,y_i) \equiv mcu(\{A_i(x),A^i(y_i)\})$, $\rho \equiv \rho^1 \cdots \rho^n$ is consistent. Clearly, $\sigma\rho(x,y_1,...,y_n)$ is consistent as well.
Writing $y \equiv y_1,...,y_n$ there exist $v \equiv v_1,...,v_n$ such that $(u,v) \in \sigma\rho(x,y)$ and for all $i \leq n$: $B^i{}_1(v_i),...,B^i{}_q(v_i) \in T'_P\downarrow k$. It follows by induction that there exists a ground specification $\tau(x,y,z) \leq \sigma\rho(x,y)$ such that for $(w,w',w'') \in \tau(x,y)$: $B^i{}_1(w'),...,B^i{}_q(w') \in T'_P\downarrow k$. Since $\rho \leq \rho^i$ we have $A_i(w) \equiv A^i(w')$ and therefore $A_1(w),...,A_k(w) \in T'_P\downarrow k+1$, by definition of T'_P. Since (proposition 4.1) $\sigma\rho \leq \sigma$ we find $\tau \leq \sigma\rho \leq \sigma$, which finishes the proof. □

Theorem 5.5 can be used to prove the following theorem:

theorem 5.6 If $\sigma(x)$ is ground and $(\leftarrow A_1(x),...,A_n(x),\sigma(x))$ has an ff-tree with depth $\leq k$, then for some $i \leq n$: $\forall u.\ (\ u \in \sigma(x) \ \Rightarrow\ A_i(u) \notin T'_P\downarrow k\ )$.

proof By induction on k.
k=1: directly by lemma 5.1.
k+1: Suppose $A_i(u) \in T'_P\downarrow k+1$, for $u \in \sigma$, then there is a clause $A(y)\leftarrow B_1(y),...,B_q(y) \in P$ such that $\rho(x,y) \equiv mcu(\{A_i(x),A(y)\})$ is consistent, and such that for some v: $(u,v) \in \rho$ and $B_1(v),...,B_q(v) \in T'_P\downarrow k$ (see lemma 5.1). Clearly $\sigma\rho$ is consistent and $(u,v) \in \sigma\rho(x,y)$. Then, by theorem 5.5 it follows that there is some ground specification $\tau(x,y,z) \leq \sigma\rho(x,y)$ such that for $(u',v',w') \in \tau$: $B_1(v'),...,B_q(v') \in T'_P\downarrow k$. Since τ is ground, it follows by induction that

$(\leftarrow B_1(y),...,B_q(y),\tau(x,y,z))$ has no ff-tree with depth $\leq k$, hence $P\cup(\leftarrow A_i(x),\tau(x,y,z))$ has no ff-tree with depth $\leq k+1$. Since $\tau\leq\sigma$ and both τ and σ are ground, it follows that $\tau\equiv_x\sigma$. Hence $P\cup(\leftarrow A_i(x),\sigma(x))$ has no ff-tree with depth $\leq k+1$.
So we proved that if $A_i(u)\in T'_P\downarrow k+1$, then $P\cup(\leftarrow A_i(x),\sigma)$ has no ff-tree with depth $\leq k+1$. Suppose for all $1\leq i\leq n$, $P\cup(\leftarrow A_i(x),\sigma)$ has no ff-tree with $\leq k+1$, then it follows by lemma 5.4 that $P\cup(\leftarrow A_1(x),...,A_n(x),\sigma(x))$ has no such ff-tree. □

corollary 5.7 If $(\leftarrow A(x),\sigma(x))$ has an ff-tree with depth $\leq k$, then:
$\forall u.\ (\ u\in\sigma(x)\ \Rightarrow\ A(u)\notin T'_P\downarrow k\ )$.

proof Follows from the last three lines of the proof of theorem 5.6. □

Note, that in corollary 5.7 $\sigma(x)$ does not need to be ground.
It is clear that from a program one cannot derive any negative formulas. As usual, we can consider negation as finite failure of the computation, i.e.: as a failing attempt to derive the formula from the program: if σ is a specification and $P\cup\{\leftarrow A,\sigma\}$ has a finitely failed CSLD-tree, we could introduce a *rule*, such as $P\vdash_{ff}\forall(\sigma\rightarrow\neg A)$, which is called the *negation as failure rule*. Then, consider the following definition:

definition 5.5 The *complete failure set*, CF_P, of a program P is defined by:
$CF_P=\{A(u)\in CB_P$: there is a specification $\sigma(x)$ such that $u\in\sigma$ and $(\leftarrow A(x),\sigma(x))$ has an ff-tree$\}$

proposition 5.8 CF_P is well-defined.

proof Assume $A(u)\in CF_P$ and suppose $d(A(u),A_1(v))=0$. Let $u\in\sigma(x)$ such that $(\leftarrow A(x),\sigma(x))$ has an ff-tree. Since $\rho(x,y)\equiv mcu(\{A(x),A_1(y)\})$ is consistent, and $(u,v)\in\rho$ we find that $\sigma\rho(x,y)$ is consistent. Now, it is easy to see that any derivation for $P\cup(\leftarrow A(x),\sigma(x))$ corresponds to a derivation for $P\cup(\leftarrow A_1(y),\sigma\rho(x,y))$, hence $P\cup(\leftarrow A_1(y),\sigma\rho(x,y))$ has an ff-tree. Since $(u,v)\in\sigma\rho(x,y)$, we find $A_1(v)\in CF_P$. □

proposition 5.9 Suppose $\sigma_1(x),...,\sigma_k(x)$ are specifications and u are complete terms such that $u\notin\sigma_1(x),...,u\notin\sigma_k(x)$, then there exists a consistent specification $\rho(x)$ such that:
(i) $u\in\rho(x)$
(ii) for all $1\leq i\leq k$: $\rho\sigma_i(x)\equiv\bot$.

The proof of proposition 5.9 is easy and left to the reader. Next we come to an important result in this chapter:

theorem 5.10 $CF_P=CB_P/T'_P\downarrow\omega$

proof $\subseteq$: Suppose $A(u)\in CF_P$ then there exists a specification $\sigma(x)$, such that $u\in\sigma$ and $P\cup(\leftarrow A(x),\sigma)$ has an ff-tree with depth $\leq k$, say. By corollary 5.7 it follows that $A(u)\notin T'_P\downarrow k$, hence $A(u)\notin T'_P\downarrow\omega$.

$\supseteq$: Suppose $A(u)\notin T'_P\downarrow\omega$ then $A(u)\notin T'_P\downarrow k$, for some $k\in\omega$, and so by lemma 5.2 there is no infinite fair derivation $(G_i(x,y_i),\rho_i(x,y_i))_{i\in\omega}$ for $P\cup\{\leftarrow A(x),\emptyset\}$, such that:

for all i: $\exists v_i.(u,v_i)\in\rho_i$.

Now, let R be a fair c-rule and let T be the CSLD-tree for $P\cup(\leftarrow A(x),\emptyset)$ with c-rule R. Delete from T all successors of nodes $(G(x,y),\sigma(x,y))$ for which for all v: $(u,v)\notin\sigma$. Clearly, the remaining tree T' is finite, since otherwise there would be an infinite derivation as mentioned above. Moreover, T cannot contain any success branches with last node $(\square,\theta)$ such that for some v: $(u,v)\in\theta$; otherwise by theorem 4.8, $A(u)\in T'_P\uparrow\omega$ which is impossible since $T'_P\uparrow\omega\subseteq T'_P\downarrow k$. Therefore in all the leaves $(G'(x,y),\theta'(x,y))$ of T', we have that for all v: $(u,v)\notin\theta'$. Since T' has only finitely many leaves, by proposition 5.9, there exists a specification $\rho(x,y)$ such that for some v: $(u,v)\in\rho$ and for all leaves $(G'(x,y),\theta'(x,y))$ of T': $\rho\theta'\equiv\bot$. Thus, $(\leftarrow A(x),\rho(x,y))$ has an ff-tree and for some v: $(u,v)\in\rho$. $\square$

So it turns out that programs with CSLD-resolution have the nice property that the greatest fixed point of T'_P is precisely the set of all non-failing (complete) atoms. As far as I know, such a result has not yet been established in the literature, although many attempts were made to find some appropriate interpretation for $gfp(T'_P)$ (see for instance LEVI & PALAMIDESSI [9]).
Furthermore we found, that the complete success set is $lfp(T'_P)$, so both sets can be described in terms of fixed points of T'_P. Therefore we have reason to believe that CSLD-semantics for logic programs may have a few nice properties extra that do not exist in 'classical' SLD-semantics.

Finally we will present the completeness theorem for the negation as failure rule. First a definition.

definition 5.6 The *completion*, comp(P), of a program P consists of the equational theory Eq(P), together with the set of all formulas

$\forall x.\ p(x) \leftrightarrow$

$(\exists y.\ \sigma_1(x,y) \wedge B^1{}_1(y)\wedge\cdots\wedge B^1{}_k(y)) \vee\cdots\vee (\exists y.\ \sigma_n(x,y) \wedge B^n{}_1(y)\wedge\cdots\wedge B^n{}_k(y))$

corresponding to the collection of all clauses

$A_1(y) \leftarrow B^1{}_1(y),\ldots,B^1{}_k(y)$

$\ldots.$

$A_n(y) \leftarrow B^n{}_1(y),\ldots,B^n{}_k(y),$

in P, such that $\sigma_i(x,y) = mcu(\{p(x),A_i(y)\})$ and p is a predicate symbol.

theorem 5.11 (soundness of the negation as failure rule)

$(\leftarrow A_1(x),\ldots,A_n(x),\sigma)$ has an ff-tree $\Rightarrow$ $comp(P) \models \forall(\sigma\rightarrow\neg(A_1\wedge\ldots\wedge A_n))$.

proof Suppose $(\leftarrow A_1(x),...,A_n(x),\sigma(x))$ has an ff-tree with depth $\leq k$ via c-rule R, say. Now by induction on k.

k=1: Suppose $R(\leftarrow A_1(x),...,A_n(x))=A_i(x)$, P has clauses $C_j(y_j)\leftarrow B^j{}_1(y_j),...,B^j{}_q(y_j)$ and define $\rho_j(x,y_j) = mcu(\{A_i(x),C_j(y_j)\})$. Clearly for all $j\leq n$, we have that $\sigma\rho_j(x,y_j)$ is inconsistent and hence $Eq(P) \models \neg\exists\sigma\rho_j(x,y_j)$, or equivalently $Eq(P) \models \forall(\sigma\rightarrow\neg\rho_j)$ for all j. Therefore $comp(P) \models \forall(\sigma\rightarrow\neg A_i)$ and hence $comp(P) \models \forall(\sigma\rightarrow\neg(A_1\wedge...\wedge A_n))$.

k+1: Suppose for some j, $\sigma\rho_j(x,y_j)$ is consistent. Then, for every such index j it follows that $(\leftarrow A_1(x),...,A_{i-1}(x),B^j{}_1(y_j),...,B^j{}_q(y_j),A_{i+1}(x),...,A_n(x),\sigma\rho_j(x,y_j))$ has an ff-tree with depth $\leq k$. Hence for all $j\leq n$: $comp(P) \models \forall(\sigma\rho_j\rightarrow\neg(A_1\wedge...\wedge A_{i-1}\wedge B^j{}_1\wedge...\wedge B^j{}_q\wedge A_{i+1}\wedge...\wedge A_n))$.
Since $comp(P) \models \forall x.(A_i(x)\leftrightarrow\exists y_1.(\rho_1\wedge B^1{}_1\wedge...\wedge B^1{}_q) \vee...\vee \exists y_n.(\rho_n\wedge B^n{}_1\wedge...\wedge B^n{}_q))$ it easily follows that $comp(P) \models \forall(\sigma\rightarrow\neg(A_1\wedge...\wedge A_n))$, for all $j\leq n$. □

The following lemma can easily be proved.

lemma 5.12 $\mathcal{M}\subseteq CB_P$ is a model for comp(P) if and only if $\mathcal{M}$ is a fixed point of T'_P.

theorem 5.13 (completeness of the negation as failure rule)
Suppose R is a *fair* computation rule, i.e.: every subgoal is selected in a finite number of steps. Then:
$comp(P) \models \forall(\sigma\rightarrow\neg(A_1\wedge...\wedge A_n)) \Rightarrow P\cup(\leftarrow A_1(x),...,A_n(x),\sigma)$ has an ff-tree via R.

proof Suppose $(\leftarrow A_1(x),...,A_n(x),\sigma_0)$ does not have a finitely failed CSLD-tree. It is proved that $gfp(T'_P) \models comp(P)\cup\{\exists(A_1\wedge...\wedge A_n\wedge\sigma_0)\}$.
Let R be a fair computation rule. Suppose $(G_i,\sigma_i(x,y_i)_{i\in\omega}$ is a non-failed CSLD-derivation for $P\cup(\leftarrow A_1(x),...,A_n(x),\sigma_0)$ via R. Define: $\Sigma(x,y_1,y_2,...) = \cup_{i\in\omega}\sigma_i(x,y_i)$. Since $\Sigma\cup Eq(P)$ is finitely satisfiable it follows from the compactness theorem that $\Sigma\cup Eq(P)$ has a model. Then, by theorem 3.9, Σ is satisfiable in every complete Herbrand model and hence $gfp(T'_P)\cup\Sigma$ is satisfiable.
Let u be such that $\Sigma[x/u]$ is satisfiable in $gfp(T'_P)$. Then for all i, $\sigma_i[x/u]$ is satisfiable, and hence there exist v_i such that for all i: $(u,v_i)\in\sigma_i(x,y_i)$. Especially we find that $u\in\sigma_0(x)$. Since R is fair it follows from theorem 5.2 that $A_1(u),...,A_n(u)\in T'_P\downarrow k$ for all k, hence $A_1(u),...,A_n(u)\in T'_P\downarrow\omega$ and therefore $A_1(u),...,A_n(u)\in gfp(T'_P)$, since $T'_P\downarrow\omega=gfp(T'_P)$.
By lemma 5.12 we have $gfp(T'_P) \models comp(P)$, so $gfp(T'_P) \models comp(P)\cup\{\exists(A_1\wedge...\wedge A_n\wedge\sigma_0)\}$. □

The proof of theorem 5.13 is quite new, in the sense that the model $gfp(T'_P)$ is a Herbrand model. In the case of SLD-resolution, a completeness theorem for negation as failure was proved by JAFFAR, LASSEZ & LLOYD [8] and WOLFRAM, MAHER & LASSEZ [12] by constructing a model for $comp(P)\cup\{\exists(A_1\wedge...\wedge A_n\wedge\sigma_0)\}$ which is not a Herbrand model. This serious complication in the proof is completely overcome by the fact that $T'_P\downarrow\omega=gfp(T'_P)$. We have the following corollary:

corollary 5.14 comp(P)$\cup${A$\wedge\sigma$} has a complete Herbrand model iff it has a model.

proof Directly from 5.11 and 5.13. Note that in the proof of 5.13 gfp(T'_P) is a complete Herbrand model. □

This result cannot be obtained in ordinary SLD-semantics (see for instance [10]).

acknowledgements

At this place I especially want to thank prof. Krzysztof Apt for his clarifying comments, Jos Baeten for pointing out some errors in a draft version and prof. Johan van Benthem for giving his support to this paper.

REFERENCES

[1] Apt, K.R. & van Emden, M.H., *Contributions to the Theory of Logic Programming*, JACM,29,3 (july 1982),841-862.

[2] Colmerauer, A., *Prolog and Infinite Trees*, Clark and Tarnlund (eds.), Logic Programming, Academic Press, New York, 1982, pp.231-251.

[3] Colmerauer, A., *Prolog II Reference Manual and Theoretical Model*, Groupe Intelligence Artificielle, Faculté des Sciences de Luming, Marseille, 1982.

[4] Courcelle, B., *Fundamental Properties of Infinite Trees*, Theoretical Computer Science, 25(2), pp.95-169, March 1983.

[5] van Emden, M.H. & Kowalski, R.A., *The Semantics of Predicate Logic as a Programming Language*, JACM,23,4 (October 1976),733-742.

[6] van Emden, M.H. & Lloyd, J.W., *A Logical Reconstruction of Prolog II*, Journal of Logic Programming, 1(2), pp.143-150, august 1984.

[7] Jaffar, J., Lassez, J-L. & Maher, M.J., *Prolog II as an instance of the Logic Programming Language Scheme*, M. Wirsing (ed.), proc. IFIP conf., North Holland, 1987.

[8] Jaffar, J., Lassez, J-L & Lloyd, J.W., *Completeness of the Negation as Finite Failure Rule*, IJCAI-83, Karlsruhe, 1983, 500-506.

[9] Levi, G. & Palamidessi, C., *Contributions to the Semantics of Logic Perpetual Processes*, draft manuscript, Dipartimento di Informatica, Università di Pisa, Italy, 1987.

[10] Lloyd, J.W., *Foundations of Logic Programming*, Springer Verlag, Heidelberg, 1984.

[11] Plaisted, D.A., *The Occur-Check Problem in Prolog*, Int. Symp. on Logic Programming, Atlantic City, 1984, 272-280.

[12] Wolfram, D.A., Maher, M.J. & Lassez, J-L., *A Unified Treatment of Resolution Strategies for Logic Programs*, Second International Logic Programming Conference, Uppsala, 1984, 263-276.