Theoretical Computer Science 71 (1990) 155-174
North-Holland

SEMANTICS FOR LOGIC PROGRAMS WITHOUT OCCUR CHECK

W. P. WEIJLAND

Centre for Mathematics and Computer Science, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

Abstract. For reasons of efficiency, in almost all implementations of Prolog the *occur check* is left out. This a 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. 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 both CSLD-resolution and the "negation as failure" rule, are established.

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 [12]). For example consider the program

P: $\text{test} \leftarrow p(x, x)$

$p(y, f(y)) \leftarrow .$

Given the goal $\leftarrow$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 [14], 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 reason Prolog II (see [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 [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 in [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 $P \cup \{x=y, y=f(y)\}$. There are still many results left to be established, such as completeness, to develop a complete theory for Prolog II.

In this paper a semantics for logic programs without occur check 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 to it all infinite terms $\{x=f(f(f(\ldots)))\}$ (see [4]). We introduce a new kind of resolution, which will be referred to as CSLD-resolution (*complete* SLD-resolution), which is precisely SLD-resolution but without occur check. Following this idea, we find that both soundness and completeness for CSLD-resolution as well as for the negation as failure rule is obtained. It turns out that due to the new setting, the proof of the completeness theorem for the negation as failure rule becomes much shorter compared with the well-known proofs in [9], [12] and [15]. Then, we conclude as a general result that $\mathrm{comp}(P) \cup \{A\}$ has a "complete" Herbrand model iff it has a model, which indicates that we may expect CSLD-resolution to have some nice extra properties that we do not have for ordinary SLD-resolution.

Independently from this paper, similar results were stated in [7] and [13] on the "Constraint Logic Programming" scheme, in [8] on Prolog II as a Logic Programming Language scheme, and in [10], giving a logical semantics to a language without occur check containing both equations and inequations.

There are a few theoretical differences between these references and the contents of this paper. Consider for example the fact that we will only need a small equational theory for Prolog II, whereas in [6] and [8] this theory contains infinitely many existential formulas, one for every recursive equation. For this reason we do not need to put any constraints on the models of Prolog II programs and the results are more general. Still, apart from the question of whether the main results are new, we believe we have found a rather elegant formalisation of the theory, making the techniques used in this paper of interest by themselves. The concepts and notations in this paper are quite similar to those in [12], which may be considered a contribution to standardising the theory of logic programming without occur check.

2. Complete Herbrand models

We will assume P to be a set of *program clauses* $\forall(B_1 \wedge \cdots \wedge B_k \rightarrow A)$, 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 [12], 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.

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 $[n, 0]$, $[n, 1], \ldots, [n, k]$ as the descendents of the node n for all $n \in T$, $k < \omega$. Now let S be a set of symbols and $\mathrm{ar}: S \to \omega$ be a mapping defining the *arity* of a symbol.

Definition 2.2. A *complete term* (over S) is a function $t: \mathrm{dom}(t) \to S$ such that:

(i) the domain of t, $\mathrm{dom}(t)$, is a non-empty tree

(ii) for all $n \in \mathrm{dom}(t)$, $\mathrm{ar}(t(n)) = |\{i: [n, i] \in \mathrm{dom}(t)\}|$.

In a language L, a *complete atom* is a complete term t such that $t([\,])$ is a predicate symbol.

Definition 2.3. The *depth* $\mathrm{dp}(t)$ of a term t is defined by:

(i) $\mathrm{dp}(t) = \infty$, if t is infinite

(ii) $\mathrm{dp}(t) = 1 + \max\{|n|: n \in \mathrm{dom}(t)\}$, if t is finite.

The tree $\mathrm{dom}(t)$ 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 $\mathrm{dom}(t)$ is finite. Next, we will define a metric on Term_S.

Definition 2.4. Let $s, t \in \mathrm{Term}_S$ and $s \neq t$ then we define $\beta(s, t)$ as being the *least* depth at which s and t differ. Then we define:

(i) $d(s, t) = 0$ if $s = t$

(ii) $d(s, t) = 2^{-\beta(s,t)}$ if $s \neq t$.

Proposition 2.5. (Term_S, d) *is an (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. 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 (nor in Term_S).

Definition 2.6. 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 $\mathrm{dom}(\alpha_n(t))$ is adjusted in the same way, by leaving out all nodes without a label.

Definition 2.7. A metric space (X, d) is *compact* if every sequence in X has a subsequence which converges to a point in X.

Proposition 2.8. (Term_S, d) *is compact iff* S *is finite.*

For a proof of this well known theorem, see [12]. From Definition 2.6 we find: $\mathrm{dp}(\alpha_n(t)) \leqslant n+1$, for all t. Moreover, $d(\alpha_n(t), t) \leqslant 2^{-n}$ and therefore $\lim_{n\to\infty} \alpha_n(t) = t$. Next we will consider complete Herbrand models for a program P, having all possibly infinite terms as its universe.

Definition 2.9. Let P be a program, then the *complete Herbrand universe* CU_P is defined by $\mathrm{Term}_{L(P)}$. A *complete Herbrand model* for L_P is a model $\mathcal{M}$ with domain CU_P, such that

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

(ii) $f^{\mathcal{M}}(t_1, \ldots, t_k) = f(t_1^{\mathcal{M}}, \ldots, t_k^{\mathcal{M}})$, for all functions $f \in L_P$ and complete terms $t_1, \ldots, t_k \in \mathrm{Term}_{L(P)}$.

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

Definition 2.10. The *complete Herbrand base* CB_P, or $\mathrm{CB}_{L(P)}$, of a program P is defined by

$$\mathrm{CB}_P = \{t \in \mathrm{CU}_P : t \text{ is a complete atom}\}.$$

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. Informally, we will write U_P for the set of all finite terms in L_P and B_P for the set of all finite elements from CB_P.

In general any complete Herbrand model for a program P, can be associated with a subset of the complete Herbrand base CB_P: 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 $\mathcal{M}$ and $\models A$ to express that A holds in all (possibly non-Herbrand) models.

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 need a different *unification algorithm,* since we will work in *complete* Herbrand models.

Definition 3.1. Let P be a program. Then the theory $\mathrm{Eq}(L_P)$ (or $\mathrm{Eq}(P)$) consists of the axioms:

(1) $c \neq d$ for all pairs of distinct constants c, d from L_P

(2) $\forall \boldsymbol{x}.\ f(\boldsymbol{x}) \neq g(\boldsymbol{x})$ for all pairs of distinct function symbols f, g from L_P

(3) $\forall \boldsymbol{x}.\ f(\boldsymbol{x}) \neq c$ for all function symbols f and all constants c from L_P

(4) $\forall \boldsymbol{xy}.\ x_1 \neq y_1 \vee \cdots \vee x_k \neq y_k \rightarrow f(\boldsymbol{x}) \neq f(\boldsymbol{y})$ for all function symbols f from L_P

(5) $\forall \boldsymbol{x}.\ x = x$

(6) $\forall \boldsymbol{xy}.\ x_1 = y_1 \wedge \cdots \wedge x_k = y_k \rightarrow f(\boldsymbol{x}) = f(\boldsymbol{y})$ for all function symbols f from L_P

(7) $\forall \boldsymbol{xy}.\ x_1 = y_1 \wedge \cdots \wedge x_k = y_k \rightarrow (P(\boldsymbol{x}) \rightarrow P(\boldsymbol{y}))$ for all predicate symbols P from L_P.

Lemma 3.2. $\mathrm{Eq}(L)$ *holds in all complete Herbrand models for* L.

The axioms of $\mathrm{Eq}(L_P)$ are introduced in [12] to model finite failure: $\mathrm{Eq}(L_P)$ forces any two syntactically different terms to be different in all its models. In [12] we even find an extra axiom:

(8) $\forall \boldsymbol{x}.\ x \neq t[x]$

for all terms t that are unequal to a variable containing 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 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 syntactic operation of binding variables, but directly as equational formulas.

Definition 3.3. A (*recursive*) *specification* in a language L is a set of equations of the form: $\{t_1(\boldsymbol{x}) = s_1(\boldsymbol{x}), \ldots, t_k(\boldsymbol{x}) = s_k(\boldsymbol{x})\}$ for (open) terms t_i, $s_i \in L$ and variables $\boldsymbol{x} \equiv x_1, \ldots, x_n$.

Definition 3.4. An *open* complete term in a language L is obtained by constructing a complete term from $L \cup \{x_i : i \leqslant n\}$, where $\{x_i : i \leqslant n\}$ denotes a finite set of variable symbols with arity zero.

Note that an open complete term only has finitely many variables, called the *free* variables of the term. We will write $t(\boldsymbol{x})$ for the term t which has variables only from $\boldsymbol{x} \equiv x_1, \ldots, x_n$ (but possibly less) and similarly we write $\rho(\boldsymbol{x})$ for a specification with variables only from $\boldsymbol{x} \equiv x_1, \ldots, x_n$. Note that the metric d can simply be generalised to open complete terms, by extending the language L with extra variable symbols.

Proposition 3.5. *Let L be a language with at least one constant, and let $\mathcal{M}$ be a complete Herbrand model for L. Then for all open complete terms $t_1(\boldsymbol{x})$, $t_2(\boldsymbol{x})$ we have:*

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

Definition 3.6. A specification $\rho(\boldsymbol{x})$ is said to be in *reduced form* if it is of the form $\{x_1 = s_1(x_1, \ldots, x_n), \ldots, x_k = s_k(x_1, \ldots, x_n)\}$, where $x_1, \ldots, x_k$ are distinct variables. Moreover, $\rho(\boldsymbol{x})$ *has* a reduced form it 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 $f(t_1, \ldots, t_n) = g(s_1, \ldots, s_n)$ for some distinct symbols a, b or f, g, respectively. Moreover, it *has* a contradictory form if it is logically equivalent to a specification which is in contradictory form.

Definition 3.7. 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 3.8. A specification $\rho(\boldsymbol{x})$ is called *consistent* if $\rho \cup \mathrm{Eq}(\rho)$ is satisfiable in a model.

Theorem 3.9. *Let L be a language. If $\rho(\boldsymbol{x}, \boldsymbol{y})$ is in reduced form, with bound variables $\boldsymbol{x} \equiv x_1, \ldots, x_n$ which are distinct and free variables $\boldsymbol{y}$, then there exist (open) complete terms $t_1(\boldsymbol{y}), \ldots, t_n(\boldsymbol{y})$ with only free variables from ρ, such that in every complete Herbrand model $\mathcal{M}$ for L:*

$$\mathcal{M} \models \forall \boldsymbol{xy}.\ \rho(\boldsymbol{x}, \boldsymbol{y}) \leftrightarrow (x_1 = t_1(\boldsymbol{y}) \wedge \cdots \wedge x_n = t_n(\boldsymbol{y})).$$

Proof. Use the fact that for all equations $x = t(x, \boldsymbol{y})$ in ρ, $\mathcal{M} \models \forall \boldsymbol{xy}.\ x = t(x, \boldsymbol{y}) \leftrightarrow x = z[z/t^{\omega}(\boldsymbol{y})]$, in every complete Herbrand model $\mathcal{M}$ for L_ρ and with $t^{\omega}(\boldsymbol{y}) := t(t(t(\ldots, \boldsymbol{y}), \boldsymbol{y}), \boldsymbol{y})$. □

Theorem 3.10. (i) *A specification in reduced form is consistent.*
(ii) *A specification in contradictory form is inconsistent.*

Proof. (i) Suppose ρ is in reduced form, with language L. Extend L until it contains at least one constant, then L has a complete Herbrand model $\mathcal{M}$. Then from Theorem 3.9 and Lemma 3.2 it follows that: $\mathcal{M} \models \rho \cup \mathrm{Eq}(\rho)$, thus ρ is consistent. Part (ii) follows directly from the definition of $\mathrm{Eq}(\rho)$. □

Corollary 3.11. *Every specification in reduced form with a constant, has a complete Herbrand model.*

Theorem 3.12. (i) *All consistent specifications have a reduced form.*
(ii) *All inconsistent specifications have a contradictory form.*

The proof of Corollary 3.11 follows from the proof of Theorem 3.10. Theorem 3.12 is the reverse of Theorem 3.10. In order to prove it, an algorithm can be

constructed which actually *decides* whether a specification has a reduced form. This algorithm consists of the following five steps, defined in [2] (see also [8]). 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 not larger than s (in number of symbols).
(5) Replace an equation of the form $f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)$ by the equations $t_1 = s_1, \ldots, t_n = s_n$.

It is well known that using these five steps repeatedly, any recursive specification ρ can be reduced into a specification which is either in reduced form or in contradictory form, and equivalent to ρ. This provides us with a proof of Theorem 3.12. 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 $S = \{p(t_1^{(i)}, \ldots, t_n^{(i)})\colon i \leqslant m\}$ then apply the consistency algorithm to $\{t_1^{(1)} = t_1^{(2)}, t_1^{(2)} = t_1^{(3)}, \ldots, t_1^{(m-1)} = t_1^{(m)}, \ldots, t_n^{(1)} = t_n^{(2)}, t_n^{(2)} = t_n^{(3)}, \ldots, t_n^{(m-1)} = t_n^{(m)}\}$.

Corollary 3.13. *A specification ρ is consistent iff it has a reduced form.*

Definition 3.14. Let S be a set of (open) atoms. A specification ρ is called *complete unifier* (cu) for S, if $\models \forall(\rho \rightarrow \bigwedge_{A,B \in S}(A \leftrightarrow B))$. 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 and all complete Herbrand models $\mathcal{M}$, $\mathcal{M} \models \forall(\rho_1 \rightarrow \rho)$.

Proposition 3.15. *For any input set S of atoms, the unification algorithm computes an* mcu *for S.*

Note that Proposition 3.15 does not hold if we change Definition 3.14 by requiring $\forall(\rho_1 \rightarrow \rho)$ to hold in *all* models instead of only complete Herbrand models. For instance if $s = \{p(x), p(f(f(x))\}$ then $\{x = f(f(x))\}$ is computed by the unification algorithm, although $\{x = f(x)\}$ is more general. With respect to complete Herbrand models, however, they are equivalent.

Theorem 3.16. *Let ρ be a (possibly infinite) specification in a language L with at least one constant, then the following are equivalent*:

(i) $\rho \cup \mathrm{Eq}(\rho)$ *has a model,*

(ii) ρ *has a complete Herbrand model,*
(iii) ρ *is satisfiable in all complete Herbrand models for* L.

Proof. (ii)$\Rightarrow$(iii) and (iii)$\Rightarrow$(i) are easy and left to the reader.

(i)$\Rightarrow$(ii): suppose $\rho = \cup_{i\in\omega} \rho_i(\boldsymbol{x}_i)$, where all $\rho_i(\boldsymbol{x}_i)$ are finite subsets of ρ, and define for all n: $\sum_n(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \bigcup_{i\leqslant n} \rho_i(\boldsymbol{x}_i)$. Clearly $\sum_n$ is finite and consistent and hence equivalent to a reduced form $\sum_n'$. From Theorem 3.9 it follows that $\mathcal{M} \models \sum_n'(\boldsymbol{t}_1^n, \ldots, \boldsymbol{t}_n^n)$ for all complete Herbrand models $\mathcal{M}$. Thus, for every natural number n we obtain a sequence $(\boldsymbol{t}_n^i)_{i\geqslant n}$ in CU_ρ. Since CU_ρ is compact every such sequence has a subsequence $(\boldsymbol{t}_n^i)_{i\geqslant n}$ which converges to a limit $\boldsymbol{t}_n$. By topological arguments we find $\mathcal{M} \models \sum_n'(\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n)$ for all $i \geqslant n$, so $\mathcal{M} \models \sum_n(\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n)$ for all n, i.e. $\rho = \bigcup_{n\in\omega} \sum_n$ is satisfiable in all $\mathcal{M}$. □

Theorem 3.17. *Let* L *be a language, and suppose* $A(\boldsymbol{x}), B(\boldsymbol{x}) \in L$ *are unifiable atoms with* mcu $\rho(\boldsymbol{x})$ *then for all complete Herbrand models* $\mathcal{M}$ *for* L *and all* $\boldsymbol{t} \equiv t_1, \ldots, t_k \in |\mathcal{M}|$, *the following are equivalent*:

(i) $\mathcal{M} \models \rho(\boldsymbol{t})$,
(ii) $d(A(\boldsymbol{t}), B(\boldsymbol{t})) = 0$.

Proof. (i)$\Rightarrow$(ii): if $\mathcal{M} \models \rho(\boldsymbol{t})$ for some complete Herbrand model $\mathcal{M}$, then it follows easily from Theorem 3.16 and Definition 3.14 that for all $\mathcal{M}$, we have $\mathcal{M} \models A(\boldsymbol{t}) \leftrightarrow B(\boldsymbol{t})$. Hence $d(A(\boldsymbol{t}), B(\boldsymbol{t})) = 0$.

(ii)$\Rightarrow$(i): if $d(A(\boldsymbol{t}), B(\boldsymbol{t})) = 0$ where: $A = p(u_1(\boldsymbol{x}), \ldots, u_n(\boldsymbol{x}))$, $B = p(v_1(\boldsymbol{x}), \ldots, v_n(\boldsymbol{x}))$ then $d(u_i(\boldsymbol{t}), v_i(\boldsymbol{t})) = 0$ for all i, and hence $\rho' = \{u_1(\boldsymbol{x}) = v_1(\boldsymbol{x}), \ldots, u_n(\boldsymbol{x}) = v_n(\boldsymbol{x})\}$ is consistent. Since ρ' is a cu for $\{A, B\}$, and ρ is an mcu, it follows by Definition 3.14 that $\mathcal{M} \models \rho(\boldsymbol{t})$. □

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 $\sigma(\boldsymbol{x})$, $\rho(\boldsymbol{x})$ with variables $\boldsymbol{x} \equiv x_1, \ldots, x_k$ and for arbitrary complete terms $\boldsymbol{u} \equiv u_1, \ldots, u_k$ we write:

$$\boldsymbol{u} \in \sigma \;:\Leftrightarrow\; \text{for all complete Herbrand models } \mathcal{M} \text{ for } L_P\colon \mathcal{M} \models \sigma[\boldsymbol{u}],$$

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

$$\sigma \leqslant \rho \;:\Leftrightarrow\; \models \forall(\sigma \rightarrow \rho) \quad (\text{'}\sigma \text{ is more specific than } \rho\text{'}),$$

$$\sigma \equiv \bot \;:\Leftrightarrow\; \sigma \text{ is inconsistent} \quad (\bot \text{ stands for the bottom in the ordering } \leqslant),$$

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

Furthermore we often write $\sigma \cdot \rho$ or $\sigma\rho$ instead of $\sigma \cup \rho$.

Proposition 4.2. *One can easily check the following statements*:

(i) $(\emptyset \cdot \sigma) \equiv \sigma$, $(\bot \cdot \sigma) \equiv \bot$, $(\sigma \cdot \rho) \equiv (\rho \cdot \sigma)$, $((\sigma \cdot \rho) \cdot \tau) \equiv (\sigma \cdot (\rho \cdot \tau))$.

(ii) $\sigma \leqslant \rho \Leftrightarrow$ *for some* τ: $\sigma \equiv (\rho \cdot \tau)$.

(iii) *if* $\rho(x)$ *is ground (i.e. has no free variables) and* $\sigma(x, y)$, $\rho(x, z)$ *are specifications with distinct variables* x, y *and* z, *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 4.2 is easy and left to the reader. Taking $\rho = \emptyset$, it follows from (iii) 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) $R(G_i) = A_m$ and $G_{i+1} = (\leftarrow A_1, \ldots, A_{m-1}, B_1, \ldots, B_q, A_{m+1}, \ldots, A_k, \sigma\rho)$

(ii) $\sigma \not\equiv \bot$ and $\rho \equiv \mathrm{mcu}(\{A_m, A\})$.

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 $(G_0, \sigma_0), \ldots, (G_k, \sigma_k), \ldots$, such that for some sequence of program clauses $C_1, \ldots, C_k, \ldots \in P$ with new variables, G_{i+1} is derived from G_i and C_{i+1} by R, for all i. Note that 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 $P \cup (\leftarrow A_1, \ldots, A_k, \rho)$ is a consistent specification σ such that $P \models \forall(\sigma \rightarrow A_1 \wedge \cdots \wedge A_k \wedge \rho)$.

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 and Lloyd [6].

Theorem 4.8 (soundness of CSLD-resolution). *Computed answer specifications are correct.*

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

($n=1$): 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 $\rho \equiv \mathrm{mcu}(\{A_1, A\})$. By Definition 3.14 we have $\models\forall(\rho\rightarrow(A_1\leftrightarrow A))$ and since $\sigma_1 \equiv \sigma_0\rho \leqslant \rho$ (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 $\sigma_1\leqslant\sigma_0$.

($n+1$): Assume $R(G_0)=A_i$, then there is a clause $A\leftarrow B_1,\ldots,B_q\in P$ such that $\sigma_1\equiv\sigma_0\rho$ is consistent, where $\rho\equiv\mathrm{mcu}(\{A_i, A\})$ (see Definitions 4.5 and 4.6). Now $(G_i, \sigma_i)_{1\leqslant i\leqslant n+1}$ is a CSLD-refutation for

$$P\cup\{(\leftarrow A_1,\ldots,A_{i-1}, B_1,\ldots,B_q, A_{i+1},\ldots,A_k,\sigma_1)\}$$

with length n, so by induction we obtain:

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

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

Here (ii) follows from $\sigma_1\leqslant\sigma_0$. By Definition 3.14 it follows that $\models\forall(\rho\rightarrow(A_i\leftrightarrow A))$, and since $A\leftarrow B_1\wedge\cdots\wedge B_q\in P$ we have $P\models\forall(B_1\wedge\cdots\wedge B_q\wedge\rho\rightarrow A_i)$. Therefore from $\sigma_{n+1}\leqslant\sigma_0\rho\leqslant\rho$ and (i) we find that $P\models\forall(\sigma_{n+1}\rightarrow A_i)$. Hence, with (ii), $P\models\forall(\sigma_{n+1}\rightarrow A_1\wedge\cdots\wedge A_k\wedge\sigma_0)$. □

Next we will consider a method, first introduced in [5], to find models for logic programs using fixed points of some monotonic mapping. We assume $A(\boldsymbol{x})$ to be finite for variables $\boldsymbol{x}$.

Definition 4.9. For $X\subseteq \mathrm{CB}_P$ we define

$$\begin{aligned}\mathrm{CT}_P(X) := \{A(\boldsymbol{u})\in\mathrm{CB}_P :\ &\text{there are terms } \boldsymbol{v}\in\mathrm{CU}_P \text{ and a clause } A_1(\boldsymbol{x})\\ &\leftarrow B_1(\boldsymbol{x}),\ldots,B_q(\boldsymbol{x})\in P \text{ such that } d(A(\boldsymbol{u}), A_1(\boldsymbol{v}))=0\\ &\text{and } \{B_1(\boldsymbol{v}),\ldots,B_q(\boldsymbol{v})\}\subseteq X\}.\end{aligned}$$

CT_P is a *continuous* mapping on the complete lattice formed by all subsets of CB_P, with the usual ordering $\subseteq$. CT_P is often called the *one step derivation* map.

Definition 4.10. We will use the following notation:

$$\mathrm{CT}_P\uparrow 0=\emptyset,$$

$$\mathrm{CT}_P\uparrow k+1=\mathrm{CT}_P(\mathrm{CT}_P\uparrow k),\quad k\in\omega,$$

$$\mathrm{CT}_P\uparrow\omega=\bigcup_{k<\omega}\mathrm{CT}_P\uparrow k.$$

A well-known theorem says that $\mathrm{CT}_P\uparrow\omega$ is the *least fixed point* of CT_P, denoted by $\mathrm{lfp}(\mathrm{CT}_P)$. We write T_P for the mapping defined on the powerset of B_P which is CT_P but then restricted to finite atoms.

Proposition 4.11. *Let P be a program and $\mathcal{M} \subseteq \mathrm{CB}_P$ a complete Herbrand model, then $\mathcal{M}$ is a model for $P \Leftrightarrow \mathrm{CT}_P(\mathcal{M}) \subseteq \mathcal{M}$.*

Proposition 4.12. *Let P be a program, then* $\mathrm{lfp}(\mathrm{CT}_P) = \cap\{\mathcal{M} \subseteq \mathrm{CB}_P : \mathrm{CT}_P(\mathcal{M}) \subseteq \mathcal{M}\}$.

Corollary 4.13. (i) *$\mathrm{CT}_P \uparrow \omega$ is a model for P.*
(ii) $\mathrm{CT}_P \uparrow \omega = \cap\{\mathcal{M} \subseteq \mathrm{CB}_P : \mathcal{M}$ *is a model for* $P\}$.

The proof of Proposition 4.11 is easy and left to the reader. Proposition 4.12 is a special instance of a general theorem in the theory of complete partial orders. Its proof is omitted here (see [12]). It follows from Corollary 4.13 that $\mathrm{CT}_P \uparrow \omega$ is the *least complete Herbrand model* for P.

Let us 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 $\mathrm{CT}_P \uparrow \omega$. First we present an important lemma.

Lemma 4.14 (mcu-lemma). *Let G be a goal and assume $P \cup \{G_0, \sigma_0\}$ has a* CSLD-*refutation $(G_i, \sigma_i)_{0 \leq i \leq n}$ 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 $(G'_1, \sigma'_i)_{0 \leq i \leq n}$ of the same length, such that:*
(i) $\sigma'_1 \equiv \sigma_0\theta'_1 \cdots \theta'_i$
(ii) $\theta'_1, \ldots, \theta'_i$ *are most general complete unifiers*
(iii) $\sigma_n \leq \sigma'_n$.

The proof is an analogon of the proof given in [12]. 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. Let us define the complete equivalent of the *success set* (see [1]).

Definition 4.15. The *complete success set* CS_P of a program P is defined by

$$\mathrm{CS}_P = \{A(\boldsymbol{u}) \in \mathrm{CB}_P : P \cup \{(\leftarrow A(\boldsymbol{x}), \emptyset)\} \text{ has a CSLD-refutation with computed answer specification } \sigma(\boldsymbol{x}, \boldsymbol{y}), \text{ such that for some } \boldsymbol{v}\colon (\boldsymbol{u}, \boldsymbol{v}) \in \sigma\}.$$

Proposition 4.16. *CS_P is well defined.*

Proof. Assume $d(A_1(\boldsymbol{u}), A_2(\boldsymbol{w})) = 0$ and $A_1(\boldsymbol{u}) \in \mathrm{CS}_P$, then we prove that $A_2(\boldsymbol{w}) \in \mathrm{CS}_P$. By definition $(\leftarrow A_1(\boldsymbol{x}), \emptyset)$ has a CSLD-refutation $(G_i, \sigma_i)_{0 \leq i \leq n}$ with computed answer specification $\sigma_n(\boldsymbol{x}, \boldsymbol{y})$ such that for some $\boldsymbol{v}\colon (\boldsymbol{u}, \boldsymbol{v}) \in \sigma_n$. Let $\rho(\boldsymbol{x}, \boldsymbol{z}) \equiv \mathrm{mcu}(\{A_1(\boldsymbol{x}), A_2(\boldsymbol{z})\})$, then ρ is consistent and $(\boldsymbol{u}, \boldsymbol{w}) \in \rho$. Clearly $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) \in \rho_n(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ and therefore $\rho\sigma_n$ is consistent and $(G_i, \rho\sigma_i)_{0 \leq i \leq n}$ is a refutation for $P \cup \{\leftarrow A_2(\boldsymbol{z}), \rho\}$ (using the same clauses and the same computation rule). Now consider $(\leftarrow A_2(\boldsymbol{z}), \emptyset)$, $(G_1, \rho\sigma_1)$, $(G_2, \rho\sigma_2), \ldots, (G_n, \rho\sigma_n)$ then clearly this is a refutation for $P \cup \{\leftarrow A_2(\boldsymbol{z}), \emptyset\}$ except that $\rho\sigma_1$ is not *most general.* Thus by Lemma 4.14 there is a

refutation for $P \cup \{\leftarrow A_2(z), \emptyset\}$ with computed answer specification $\sigma(x, y, z) \geqslant \rho\sigma_n(x, y, z)$, so $(u, v, w) \in \sigma$. Hence $A_2(w) \in \mathrm{CS}_P$. □

Theorem 4.17. $\mathrm{CS}_P = \mathrm{CT}_P \uparrow \omega$.

Proof. ($\Rightarrow$): Let $A(u) \in \mathrm{CS}_P$ 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 it follows that $\sigma(x, y)$ is correct for $P \cup \{(\leftarrow A(x), \emptyset)\}$, i.e. $P \models \forall(\sigma(x, y) \rightarrow A(x))$, and hence $\mathcal{M} \models A(u)$ for all complete Herbrand models $\mathcal{M}$ for P, since $(u, v) \in \sigma$. Then by Corollary 4.13 it follows that $A(u) \in \mathrm{CT}_P \uparrow \omega$.

($\Leftarrow$): Let $A(u) \in \mathrm{CT}_P \uparrow \omega$ then $A(u) \in \mathrm{CT}_P \uparrow n$, for some $n < \omega$. Now the proof proceeds by induction.

($n = 1$): Then $A(u) \in \mathrm{CT}_P(\emptyset)$, so there is a unit clause $A_1(y) \leftarrow \in P$, such that for some v, $(u, v) \in \rho(x, y)$ and $\rho(x, y) \equiv \mathrm{mcu}(\{A(x), A_1(y)\})$; 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, $A(u) \in \mathrm{CT}_P(\mathrm{CT}_P \uparrow n)$, so there is a clause $A_1(y) \leftarrow B_1(y), \ldots, B_q(y) \in P$ and v such that: $A(u) \equiv A_1(v)$ and $B_1(v), \ldots, B_q(v) \in \mathrm{CT}_P \uparrow n$. Assume $\rho(x, y) \equiv \mathrm{mcu}(\{A(x), A_1(y)\})$ then $(u, v) \in \rho(x, y)$. 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 $P \cup \{\leftarrow B_1(y), \ldots, B_q(y), \emptyset\}$ with a computed answer specification $\sigma \geqslant \sigma_1 \cdots \sigma_q$. Since $(v, w_1, \ldots, w_q) \in \sigma_1 \cdots \sigma_q$, such a consistent σ exists, and we directly find that $\sigma\rho$ is consistent as well, since $(u, v, w_1, \ldots, w_q) \in \sigma\rho$. Hence $\sigma\rho$ is the intended answer specification. □

Theorem 4.17 is part of a more general result from [10] on a language with equations and inequations.

So far, we found that the correctness theorem can be restored for CSLD-resolution. Moreover, $\mathrm{CT}_P \uparrow \omega$ is the least complete Herbrand model which is the intersection of all complete Herbrand models for P, and equal to the complete success set. Next we will show that we have a completeness theorem as well.

Definition 4.18 (*restriction*). Let $\sigma(x, y)$ and $\rho(x, z)$ be two specifications then we write $\sigma \leqslant_x \rho$ if $\models \forall xy.\ (\sigma(x, y) \rightarrow \exists z.\ \rho(x, z))$. Furthermore we write $\sigma \equiv_x \rho$ if both $\sigma \leqslant_x \rho$ and $\rho \leqslant_x \sigma$.

Proposition 4.19. *Let $\sigma(x)$ be ground then for all specifications ρ: either $\sigma\rho \equiv \bot$ or $\sigma \leqslant_x \rho$.*

Definition 4.18 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. Note that $\sigma \equiv_x \bot \Rightarrow \sigma \equiv \bot$ for all variables x. Proposition 4.19 says that *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. For example, let $\sigma(x) = \{x = f(x, x)\}$ and $\rho(x, y) = \{x = f(x, y), y = x\}$ then $\sigma \leqslant_x \rho$, however not $\sigma \leqslant \rho$ (i.e. $\sigma \leqslant_{xy} \rho$), since ρ has an extra variable y. In fact: $\sigma \equiv_x \rho$.

Lemma 4.20. *Let $A(x)$ be an atom and $\sigma(x)$ correct 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 $u \in \sigma$ then $A(u) \in \mathrm{CT}_P \uparrow \omega$ (by Theorem 3.16, Definition 4.7 and Corollary 4.13). Therefore $A(u) \in \mathrm{CS}_P$ (by Theorem 4.17), hence there is a CSLD-refutation $(G_i, \rho_i)_{0 \leqslant i \leqslant n}$ 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 $(G_i, \sigma\rho_i)_{0 \leqslant i \leqslant n}$ is a refutation for $P \cup \{\leftarrow A(x), \sigma\}$ with computed answer specification $\sigma\rho_n$, and by Proposition 4.2 we have $\sigma\rho_n \leqslant \sigma$. Since σ is ground we find by Proposition 4.19, $\sigma \leqslant_x \sigma\rho_n$. Hence $\sigma \equiv_x \sigma\rho_n$.

Next, assume σ is not ground, and let $x = (y, z)$ where y and $z \equiv z_1, \ldots, z_k$ are the bound and free variables of σ, respectively. Let $a \equiv a_1, \ldots, a_k$ be *new* constants not occurring in P, A or σ, and such that for i, j, $a_i = a_j \Leftrightarrow \sigma \models z_i = z_j$. Next, consider $\sigma'(y, z) \equiv \{z_1 = a_1, \ldots, z_n = a_n\} \cdot \sigma(y, z)$, then 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 4.20 does not hold if we replace $\equiv_x$ by $\equiv$. For example let $P = \{A(y, y) \leftarrow\}$ and 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 $\rho \leqslant \sigma$, however since $\not\models \forall(x = f(x) \rightarrow (x = y \wedge y = f(y)))$, the converse is not true. 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. This problem is overcome, however, by introducing $\leqslant_x$ as a logical notion of restriction.

Lemma 4.21 (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 $\sigma \leqslant \sigma'$ then there is a* CSLD-*refutation for $P \cup \{G, \sigma'\}$ with computed answer specification ρ' of the same length such that $\rho \leqslant \rho'$.*

Theorem 4.22 (completeness of CSLD-resolution). *Let $(G_0(\boldsymbol{x}), \sigma_0(\boldsymbol{x}))$ be a goal and $\sigma(\boldsymbol{x})$ a correct answer specification for $P \cup \{G_0(\boldsymbol{x}), \sigma_0(\boldsymbol{x})\}$ then there exists a computation rule R and a R-computed answer specification $\rho(\boldsymbol{x}, \boldsymbol{y})$ for $P \cup \{G_0(\boldsymbol{x}), \sigma_0(\boldsymbol{x})\}$ such that $\sigma \leqslant_{\boldsymbol{x}} \rho$.*

Proof. Assume $G_0 = (\leftarrow A_1(\boldsymbol{x}), \ldots, A_k(\boldsymbol{x}))$ then $P \models \forall(\sigma \rightarrow A_1(\boldsymbol{x}) \wedge \cdots \wedge A_k(\boldsymbol{x}) \wedge \sigma_0)$ since σ is correct for $P \cup \{G_0, \sigma_0\}$. By Lemma 4.18 there exist refutations for $P \cup \{\leftarrow A_i, \sigma\}$ with ρ_i as computed answer specification, such that $\rho_i \equiv_{\boldsymbol{x}} \sigma$ for all i. These refutations can be combined to obtain a new refutation for $P \cup \{\leftarrow A_1(\boldsymbol{x}), \ldots, A_k(\boldsymbol{x}), \sigma\}$ with answer specification $\rho' \equiv \rho_1 \cdots \rho_k$, so $\sigma \equiv_{\boldsymbol{x}} \rho'$. Since $\sigma \leqslant \sigma_0$, it follows by the lifting lemma that there exists a CSLD-refutation for $P \cup \{\leftarrow A_1(\boldsymbol{x}), \ldots, A_k(\boldsymbol{x}), \sigma_0\}$ with computed answer specification ρ such that $\rho' \leqslant \rho$. Since $\sigma \equiv_{\boldsymbol{x}} \rho'$ we have $\sigma \leqslant_{\boldsymbol{x}} \rho$. □

The proof of Lemma 4.21 is similar to the one in [12]. The completeness theorem presented above, can be obtained from [13] and some additional remarks when applied to the axioms in [8]. It is important to understand how resolution with specifications works. In fact, a computed answer specification can be looked at as an extra condition or *constraint* (see also [7]) 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. The completeness theorem leads almost directly to the following corollary.

Corollary 4.23. *Let $A(\boldsymbol{x})$ be a atom and $\boldsymbol{u} \equiv u_1, \ldots, u_k$ complete terms, then:*

$$A(\boldsymbol{u}) \in \mathrm{CS}_P \Leftrightarrow \textit{for some } \rho(\boldsymbol{x}, y) \textit{ and some } \boldsymbol{v}\colon (\boldsymbol{u}, \boldsymbol{v}) \in \rho \textit{ and } P \models \forall(\rho(\boldsymbol{x}, \boldsymbol{y}) \rightarrow A(\boldsymbol{x})).$$

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 and the root is equal to G

(ii) if $G' = (\leftarrow A_1, \ldots, A_m, \ldots, A_k, \sigma)$ is a node for some consistent σ, and $R(G') = A_m$ then G' has a *successor* $(\leftarrow A_1, \ldots, A_{m-1}, B_1, \ldots, B_q, A_{m+1}, \ldots, A_k, \sigma\rho)$

for every clause $A \leftarrow B_1, \ldots, B_q \in P$, where $\rho \equiv \mathrm{mcu}(\{A_m, A\})$; 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:

$$\mathrm{CT}_P \downarrow 0 = \mathrm{CB}_P,$$

$$\mathrm{CT}_P \downarrow k+1 = \mathrm{CT}_P(\mathrm{CT}_P \downarrow k)), \quad k \in \omega,$$

$$\mathrm{CT}_P \downarrow \omega = \bigcap_{k < \omega} \mathrm{CT}_P \downarrow k.$$

A well-known theorem says that $\mathrm{CT}_P \downarrow \omega$ is the *greatest fixed point*, denoted by $\mathrm{gfp}(\mathrm{CT}_P)$, of the continuous mapping CT_P. Recall that $T_P \downarrow \omega$ does not need to be equal to the greatest fixed point of T_P.

Lemma 5.5. *Let $A(\boldsymbol{x})$ be an atom and $\boldsymbol{u} \equiv u_1, \ldots, u_k$ be complete terms, then*

$$A(\boldsymbol{u}) \in \mathrm{CT}_P \downarrow k+1 \Leftrightarrow \textit{there is a clause } A_1(\boldsymbol{y}) \leftarrow B_1(\boldsymbol{y}), \ldots, B_q(\boldsymbol{y}) \in P, \textit{ such that for some } \boldsymbol{v}\colon (\boldsymbol{u}, \boldsymbol{v}) \in \mathrm{mcu}(\{A(\boldsymbol{x}), A_1(\boldsymbol{y})\}) \textit{ and } B_1(\boldsymbol{v}), \ldots, B_q(\boldsymbol{v}) \in \mathrm{CT}_P \downarrow k.$$

Theorem 5.6. *Suppose R is a fair* c-*rule, i.e. every atom in a goal clause is selected somewhere in any infinite derivation, and assume $(G_i, \sigma_i)_{i \in \omega}$ is an infinite* CSLD-*derivation for $P \cup \{G_0, \sigma_0\}$ by R with $G_0 = (\leftarrow A_1(\boldsymbol{x}), \ldots, A_n(\boldsymbol{x}), \sigma_0)$. If $\boldsymbol{u}$ are complete terms, then*

$$\forall k\, \exists i.\, ((\boldsymbol{u}, \boldsymbol{v}) \in \sigma_i(\boldsymbol{x}, \boldsymbol{y}) \Rightarrow A_1(\boldsymbol{u}), \ldots, A_n(\boldsymbol{u}) \in \mathrm{CT}_P \downarrow k).$$

Proof. Let $k \in \omega$, and $\boldsymbol{u}$ complete terms. Let $(G_i, \sigma_i)_{i < \omega}$ be an infinite derivation for $P \cup (G_0, \sigma_0)$. Then 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 $A(\boldsymbol{y}) \leftarrow B_1(\boldsymbol{y}), \ldots, B_q(\boldsymbol{y})$ such that $\rho(\boldsymbol{x}, \boldsymbol{y}) \equiv \mathrm{mcu}(\{A_j(\boldsymbol{x}), A(\boldsymbol{y})\})$ is consistent and $\sigma_{m+1} \equiv \sigma_m \cdot \rho$. Assume $G_m = (\leftarrow C_1, \ldots, A_j, \ldots, C_r)$ then $G_{m+1} = (\leftarrow C_1, \ldots, B_1, \ldots, B_q, \ldots, C_r)$ 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 induction, let $i' \in \omega$ such that $(\boldsymbol{u}, \boldsymbol{v}) \in \sigma_{i'}(\boldsymbol{x}, \boldsymbol{y}) \Rightarrow C_1(\boldsymbol{u}, \boldsymbol{v}), \ldots, B_1(\boldsymbol{v}), \ldots, B_q(\boldsymbol{v}), \ldots, C_r(\boldsymbol{u}, \boldsymbol{v}) \in \mathrm{CT}_P \downarrow k$, then it follows by definition of CT_P that for all $(\boldsymbol{u}, \boldsymbol{v}) \in \sigma_{i'}(\boldsymbol{x}, \boldsymbol{y}) \Rightarrow A(\boldsymbol{v}) \in \mathrm{CT}_P \downarrow k+1$. Since $\sigma_{i'} \leq \rho$ we have $A_j(\boldsymbol{u}) \in \mathrm{CT}_P \downarrow k+1$. So, for every j such an index i' exists. Now take the maximum of all n indices. $\square$

Lemma 5.7. *If* (G, σ) *has an* ff*-tree with depth* $\leqslant k$*, then* $(G, \sigma\rho)$ *has an* ff*-tree with depth* $\leqslant k$.

Lemma 5.8. *If for some ground specification* $\sigma(\boldsymbol{x})$*, the goal* $(\leftarrow A_1(\boldsymbol{x}), \ldots, A_n(\boldsymbol{x}), \sigma(\boldsymbol{x}))$ *has an* ff*-tree with depth* $\leqslant k$*, then for some* $i \leqslant n$*,* $(\leftarrow A_i(\boldsymbol{x}), \sigma(\boldsymbol{x}))$ *has an* ff*-tree with depth* $\leqslant k$.

Proof. By induction on n:

($n=1$): Immediately.

($n+1$): Assume $A_{n+1}(\boldsymbol{x})$ has no ff-tree with depth $\leqslant k$, then for all c-rules R, $P \cup \{A_{n+1}(\boldsymbol{x}), \sigma(\boldsymbol{x})\}$ has a CSLD-derivation of length $\geqslant k+1$. Let R be a c-rule such that $P \cup \{\leftarrow A_1(\boldsymbol{x}), \ldots, A_{n+1}(\boldsymbol{x}), \sigma(\boldsymbol{x})\}$ has an ff-tree with depth $\leqslant k$, and let $(G_i, \rho_i(\boldsymbol{x}, \boldsymbol{y}_i))_{0 \leqslant i \leqslant k+1}$ be a derivation for $P \cup \{A_{n+1}(\boldsymbol{x}), \sigma(\boldsymbol{x})\}$ via R with length $\geqslant k+1$, then for all derivations $(G_i', \tau_i(\boldsymbol{x}, \boldsymbol{z}_i))_{0 \leqslant i \leqslant m}$ for $P \cup \{\leftarrow A_1(\boldsymbol{x}), \ldots, A_n(\boldsymbol{x}), \sigma(\boldsymbol{x})\}$ we have (by Propositions 4.2(iii) and 4.19) that for all i, j, $\tau_i\rho_j \equiv \bot \Rightarrow \tau_i \equiv \bot \vee \rho_i \equiv \bot$, since $\sigma(\boldsymbol{x})$ is ground and $\tau_0 = \rho_0 = \sigma$. Then, there is a finitely failed derivation $(G_i, \theta_i)_{0 \leqslant i \leqslant q}$ via R for $P \cup \{\leftarrow A_1(\boldsymbol{x}), \ldots, A_{n+1}(\boldsymbol{x}), \sigma(\boldsymbol{x})\}$ such that $\theta_i \geqslant \tau_i\rho_i$ and $q \leqslant k$ (since all derivations via R are finitely failed with length $\leqslant k$) we find that $\tau_q\rho_q \equiv \bot$. Because ρ_q is consistent for all $q \leqslant k$ we have $\tau_q \equiv \bot$.

Therefore, all derivations for $P \cup \{\leftarrow A_1(\boldsymbol{x}), \ldots, A_n(\boldsymbol{x}), \sigma(\boldsymbol{x})\}$ are finitely failed with length $\leqslant k$, hence $P \cup \{\leftarrow A_1(\boldsymbol{x}), \ldots, A_n(\boldsymbol{x}), \sigma(\boldsymbol{x})\}$ has an ff-tree with depth $\leqslant k$. Now by induction. □

Lemma 5.7 is easy to prove. Note that in Lemma 5.8, σ needs to be ground (see also [12]).

Theorem 5.9. *If* $A_1(\boldsymbol{u}), \ldots, A_n(\boldsymbol{u}) \in \mathrm{CT}_P \downarrow k$ *for some complete terms* $\boldsymbol{u} \in \sigma(\boldsymbol{x})$*, then there exists a ground specification* $\rho(\boldsymbol{x}, \boldsymbol{y}) \leqslant \sigma(\boldsymbol{x})$ *such that for* $(\boldsymbol{v}, \boldsymbol{w}) \in \rho(\boldsymbol{x}, \boldsymbol{y})$, $A_1(\boldsymbol{v}), \ldots, A_n(\boldsymbol{v}) \in \mathrm{CT}_P \downarrow k$.

Proof. By induction on k. Assume $\boldsymbol{u} \in \sigma(\boldsymbol{x})$, for some specification σ.

($k=0$): Immediately, since any ground $\rho \leqslant \sigma$ suffices.

($k+1$): Suppose $A_1(\boldsymbol{u}), \ldots, A_n(\boldsymbol{u}) \in \mathrm{CT}_P \downarrow k+1$ then by the definition of CT_P there exist clauses $A^i(\boldsymbol{y}_i) \leftarrow B_1^i(\boldsymbol{y}_i), \ldots, B_q^i(\boldsymbol{y}_i) \in P$ for all $i \leqslant n$, such that with $\rho^i(\boldsymbol{x}, \boldsymbol{y}_i) \equiv \mathrm{mcu}(\{A_i(\boldsymbol{x}), A^i(\boldsymbol{y}_i)\})$, $\rho = \rho^1 \cdots \rho^n$ is consistent. Clearly, $\sigma\rho(\boldsymbol{x}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ is consistent as well. Writing $\boldsymbol{y} \equiv \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ there exist $\boldsymbol{v} \equiv \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ such that $(\boldsymbol{u}, \boldsymbol{v}) \in \sigma\rho(\boldsymbol{x}, \boldsymbol{y})$ and for all $i \leqslant n$: $B_1^i(\boldsymbol{v}_i), \ldots, B_q^i(\boldsymbol{v}_i) \in \mathrm{CT}_P \downarrow k$. It follows by induction that there exists a ground specification $\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \leqslant \sigma\rho(\boldsymbol{x}, \boldsymbol{y})$ such that for $(\boldsymbol{w}, \boldsymbol{w}', \boldsymbol{w}'') \in \tau(\boldsymbol{x}, \boldsymbol{y})$: $B_1^i(\boldsymbol{w}'), \ldots, B_q^i(\boldsymbol{w}') \in \mathrm{CT}_P \downarrow k$. Since $\rho \leqslant \rho^i$ we have $A_i(\boldsymbol{w}) \equiv A^i(\boldsymbol{w}')$ and therefore $A_1(\boldsymbol{w}), \ldots, A_k(\boldsymbol{w}) \in \mathrm{CT}_P \downarrow k+1$, by definition of CT_P. Since $\sigma\rho \leqslant \sigma$ we find $\tau \leqslant \sigma\rho \leqslant \sigma$. □

Theorem 5.10. *If* $\sigma(\boldsymbol{x})$ *is ground and* $(\leftarrow A_1(\boldsymbol{x}), \ldots, A_n(\boldsymbol{x}), \sigma(\boldsymbol{x}))$ *has an* ff*-tree with depth* $\leqslant k$*, then for some* $i \leqslant n$

$$\forall \boldsymbol{u}.\ (\boldsymbol{u} \in \sigma(\boldsymbol{x}) \Rightarrow A_i(\boldsymbol{u}) \notin \mathrm{CT}_P \downarrow k).$$

Proof. By induction on k.

($k=1$): Directly by Lemma 5.5.

($k+1$): Suppose $A_i(\boldsymbol{u}) \in \mathrm{CT}_P \downarrow k+1$, for $\boldsymbol{u} \in \sigma$, then there is a clause $A(\boldsymbol{y}) \leftarrow B_1(\boldsymbol{y}), \ldots, B_q(\boldsymbol{y}) \in P$ such that $\rho(\boldsymbol{x}, \boldsymbol{y}) \equiv \mathrm{mcu}(\{A_i(\boldsymbol{x}), A(\boldsymbol{y})\})$ is consistent and for some $\boldsymbol{v}$: $(\boldsymbol{u}, \boldsymbol{v}) \in \rho$ and $B_1(\boldsymbol{v}), \ldots, B_q(\boldsymbol{v}) \in \mathrm{CT}_P \downarrow k$ (see Lemma 5.5). Clearly $\sigma\rho$ is consistent and $(\boldsymbol{u}, \boldsymbol{v}) \in \sigma\rho(\boldsymbol{x}, \boldsymbol{y})$. Then, by Theorem 5.9 it follows that there is some ground specification $\tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \leqslant \sigma\rho(\boldsymbol{x}, \boldsymbol{y})$ such that: $B_1(\boldsymbol{v}'), \ldots, B_q(\boldsymbol{v}') \in \mathrm{CT}_P \downarrow k$ for all $(\boldsymbol{u}', \boldsymbol{v}', \boldsymbol{w}') \in \tau$. Since τ is ground, it follows by induction that $(\leftarrow B_1(\boldsymbol{y}), \ldots, B_q(\boldsymbol{y}), \tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}))$ has no ff-tree with depth $\leqslant k$, hence $P \cup (\leftarrow A_i(\boldsymbol{x}), \tau(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}))$ has no ff-tree with depth $\leqslant k+1$. Since $\tau \leqslant \sigma$ and both τ and σ are ground, it follows that $\tau \equiv_{\boldsymbol{x}} \sigma$. Hence $P \cup (\leftarrow A_i(\boldsymbol{x}), \sigma(\boldsymbol{x}))$ has no ff-tree with depth $\leqslant k+1$.

Thus, we have proved that if $A_i(\boldsymbol{u}) \in \mathrm{CT}_P \downarrow k+1$, then $P \cup (\leftarrow A_i(\boldsymbol{x}), \sigma)$ has no ff-tree with depth $\leqslant k+1$. Suppose for all $1 \leqslant i \leqslant n$, $P \cup (\leftarrow A_i(\boldsymbol{x}), \sigma)$ has no ff-tree with depth $\leqslant k+1$, then it follows by Lemma 5.8 that $P \cup (\leftarrow A_1(\boldsymbol{x}), \ldots, A_n(\boldsymbol{x}), \sigma(\boldsymbol{x}))$ has no such ff-tree. □

Corollary 5.11. *If* $(\leftarrow A(\boldsymbol{x}), \sigma(\boldsymbol{x}))$ *has an* ff*-tree with depth* $\leqslant k$, *then*

$$\forall \boldsymbol{u}.\ (\boldsymbol{u} \in \sigma(\boldsymbol{x}) \Rightarrow A(\boldsymbol{u}) \notin \mathrm{CT}_P \downarrow k).$$

Note, that in Corollary 5.11, $\sigma(\boldsymbol{x})$ does not need to be ground. Its proof follows from the last three lines of the proof of Theorem 5.10.

Clearly, from a program one cannot derive any negative formulas. As usual, we can consider negation as finite failure of the computation. Consider the following definition.

Definition 5.12. The *complete failure set*, CF_P, of a program P is defined by

$$\mathrm{CF}_P = \{A(\boldsymbol{u}) \in \mathrm{CB}_P : \text{there is a specification } \sigma(\boldsymbol{x}) \text{ such that } \boldsymbol{u} \in \sigma \text{ and } (\leftarrow A(\boldsymbol{x}), \sigma(\boldsymbol{x})) \text{ has an ff-tree}\}.$$

Proposition 5.13. CF_P *is well-defined.*

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

Proposition 5.14. *Suppose* $\sigma_1(\boldsymbol{x}), \ldots, \sigma_k(\boldsymbol{x})$ *are specifications and* $\boldsymbol{u}$ *are complete terms such that* $\boldsymbol{u} \notin \sigma_1(\boldsymbol{x}), \ldots, \boldsymbol{u} \notin \sigma_k(\boldsymbol{x})$, *then there exists a consistent specification* $\rho(\boldsymbol{x})$ *such that* $\boldsymbol{u} \in \rho(\boldsymbol{x})$ *and for all* $1 \leqslant i \leqslant k$, $\rho\sigma_i(\boldsymbol{x}) \equiv \bot$.

Theorem 5.15. $\mathrm{CF}_P = \mathrm{CB}_P/\mathrm{CT}_P\downarrow\omega$.

Proof. ($\subseteq$): Suppose $A(\boldsymbol{u}) \in \mathrm{CF}_P$ then there exists a specification $\sigma(\boldsymbol{x})$, such that $\boldsymbol{u} \in \sigma$ and $P \cup (\leftarrow A(\boldsymbol{x}), \sigma)$ has an ff-tree with depth $\leqslant k$, say. By Corollary 5.11 it follows that $A(\boldsymbol{u}) \notin \mathrm{CT}_P\downarrow k$, hence $A(\boldsymbol{u}) \notin \mathrm{CT}_P\downarrow\omega$.

($\supseteq$): Suppose $A(\boldsymbol{u}) \notin \mathrm{CT}_P\downarrow\omega$ then $A(\boldsymbol{u}) \notin \mathrm{CT}_P\downarrow k$, for some $k \in \omega$, and so by Lemma 5.6 there is no infinite fair derivation $(G_i(\boldsymbol{x}, \boldsymbol{y}_i), \rho_i(\boldsymbol{x}, \boldsymbol{y}_i))_{i\in\omega}$ for $P \cup \{\leftarrow A(\boldsymbol{x}), \emptyset\}$, such that for all i: $\exists \boldsymbol{v}_i.\ (\boldsymbol{u}, \boldsymbol{v}_i) \in \rho_i$. Now, let R be a fair c-rule and let T be the CSLD-tree for $P \cup (\leftarrow A(\boldsymbol{x}), \emptyset)$ with c-rule R. Delete from T all successors of nodes $(G(\boldsymbol{x}, \boldsymbol{y}), \sigma(\boldsymbol{x}, \boldsymbol{y}))$ for which for all $\boldsymbol{v}$: $(\boldsymbol{u}, \boldsymbol{v}) \notin \sigma$. 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 $\boldsymbol{v}$: $(\boldsymbol{u}, \boldsymbol{v}) \in \theta$; otherwise by Theorem 4.17, $A(\boldsymbol{u}) \in \mathrm{CT}_P\downarrow\omega$ which is impossible since $\mathrm{CT}_P\uparrow\omega \subseteq \mathrm{CT}_P\downarrow k$. Therefore in all the leaves $(G'(\boldsymbol{x}, \boldsymbol{y}), \theta'(\boldsymbol{x}, \boldsymbol{y}))$ of T', we have that for all $\boldsymbol{v}$: $(\boldsymbol{u}, \boldsymbol{v}) \notin \theta'$. Since T' has only finitely many leaves, by Proposition 5.14, there exists a specification $\rho(\boldsymbol{x}, \boldsymbol{y})$ such that for some $\boldsymbol{v}$: $(\boldsymbol{u}, \boldsymbol{v}) \in \rho$ and for all leaves $(G'(\boldsymbol{x}, \boldsymbol{y}), \theta'(\boldsymbol{x}, \boldsymbol{y}))$ of T', $\rho\theta' \equiv \bot$. Thus, $(\leftarrow A(\boldsymbol{x}), \rho(\boldsymbol{x}, \boldsymbol{y}))$ has an ff-tree and for some $\boldsymbol{v}$: $(\boldsymbol{u}, \boldsymbol{v}) \in \rho$. $\square$

Theorem 5.15 can be considered as a corollary from results in [10]. It turns out that programs with CSLD-resolution have the nice property that the greatest fixed point of CT_P is precisely the set of all non-failing (complete) atoms. Furthermore we found, that the complete success set is $\mathrm{lfp}(\mathrm{CT}_P)$, so both sets can be described in terms of fixed points of CT_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.

Definition 5.16. The *completion*, $\mathrm{comp}(P)$, of a program P consists of the equational theory $\mathrm{Eq}(P)$, together with the set of all formulas

$$\forall \boldsymbol{x}.\ p(\boldsymbol{x}) \leftrightarrow (\exists \boldsymbol{y}.\ \sigma_1(\boldsymbol{x}, \boldsymbol{y}) \wedge B_1^1(\boldsymbol{y}) \wedge \cdots \wedge B_k^1(\boldsymbol{y})) \vee \cdots$$
$$\vee\ (\exists \boldsymbol{y}.\ \sigma_n(\boldsymbol{x}, \boldsymbol{y}) \wedge B_1^n(\boldsymbol{y}) \wedge \cdots \wedge B_k^n(\boldsymbol{y}))$$

corresponding to the collection of all clauses

$$A_1(\boldsymbol{y}) \leftarrow B_1^1(\boldsymbol{y}), \ldots, B_k^1(\boldsymbol{y}), \ldots, A_n(\boldsymbol{y}) \leftarrow B_1^n(\boldsymbol{y}), \ldots, B_k^n(\boldsymbol{y})$$

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

Theorem 5.17 (soundness of the negation as failure rule)

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

Proof. Suppose $(\leftarrow A_1(\boldsymbol{x}), \ldots, A_n(\boldsymbol{x}), \sigma(\boldsymbol{x}))$ has an ff-tree with depth $\leqslant k$ via c-rule R, say. Now by induction on k.

($k=1$): Suppose $R(\leftarrow A_1(x), \ldots, A_n(x)) = A_i(x)$, P has clauses $C_j(y_j) \leftarrow B_1^j(y_j), \ldots, B_q^j(y_j)$ and define $\rho_j(x, y_j) = \mathrm{mcu}(\{A_i(x), C_j(y_j)\})$. Clearly for all $j \leq n$, $\sigma\rho_j(x, y_j)$ is inconsistent and hence $\mathrm{Eq}(P) \models \forall(\sigma \rightarrow \neg\rho_j)$ for all j. Therefore $\mathrm{comp}(P) \models \forall(\sigma \rightarrow \neg A_i)$ and hence $\mathrm{comp}(P) \models \forall(\sigma \rightarrow \neg(A_1 \wedge \cdots \wedge A_n))$.

($k+1$): Suppose for some j, $\sigma\rho_j(x, y_j)$ is consistent. Then, for every such index j it follows that

$$(\leftarrow A_1(x), \ldots, A_{i-1}(x), B_1^j(y_j), \ldots, B_q^j(y_j), A_{i+1}(x), \ldots, A_n(x), \sigma\rho_j(x, y_j))$$

has an ff-tree with depth $\leq k$. Hence for all $j \leq n$,

$$\mathrm{comp}(P) \models \forall(\sigma\rho_j \rightarrow \neg(A_1 \wedge \cdots \wedge A_{i-1} \wedge B_1^j \wedge \cdots \wedge B_q^j \wedge A_{i+1} \wedge \cdots \wedge A_n)).$$

Since

$$\mathrm{comp}(P) \models \forall x.\, (A_i(x) \leftrightarrow \exists y_1.\, (\rho_1 \wedge B_1^1 \wedge \cdots \wedge B_q^1) \vee \cdots \vee \exists y_n.\, (\rho_n \wedge B_1^n \wedge \cdots \wedge B_q^n))$$

it easily follows that $\mathrm{comp}(P) \models \forall(\sigma \rightarrow \neg(A_1 \wedge \cdots \wedge A_n))$, for all $j \leq n$. □

Theorem 5.17 is a strengthening of a similar theorem from [8] by weakening the equality theory.

Lemma 5.18. *$\mathcal{M} \subseteq \mathrm{CB}_P$ is a model for $\mathrm{comp}(P)$ if and only if $\mathcal{M}$ is a fixed point of CT_P.*

Theorem 5.19 (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 $\mathrm{comp}(P) \models \forall(\sigma \rightarrow \neg(A_1 \wedge \cdots \wedge A_n)) \Rightarrow P \cup (\leftarrow A_1(x), \ldots, A_n(x), \sigma)$ has an ff-tree via R.*

Proof. Suppose $(\leftarrow A_1(x), \ldots, A_n(x), \sigma_0)$ does not have a finitely failed CSLD-tree. It is proved that $\mathrm{gfp}(\mathrm{CT}_P) \models \mathrm{comp}(P) \cup \{\exists(A_1 \wedge \cdots \wedge A_n \wedge \sigma_0)\}$. 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 $P \cup (\leftarrow A_1(x), \ldots, A_n(x), \sigma_0)$ via R. Define $\sum(x, y_1, y_2, \ldots) = \bigcup_{i \in \omega} \sigma_i(x, y_i)$. Since $\sum \cup \mathrm{Eq}(P)$ is finitely satisfiable it follows from the compactness theorem that $\sum \cup \mathrm{Eq}(P)$ has a model. Then, by Theorem 3.16, $\sum$ is satisfiable in every complete Herbrand model and hence $\mathrm{gfp}(\mathrm{CT}_P) \cup \sum$ is satisfiable. Let u be such that $\sum[x/u]$ is satisfiable in $\mathrm{gfp}(\mathrm{CT}_P)$. 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 5.6 that $A_1(u), \ldots, A_n(u) \in \mathrm{CT}_P \downarrow k$ for all k, hence $A_1(u), \ldots, A_n(u) \in \mathrm{CT}_P \downarrow \omega$ and therefore $A_1(u), \ldots, A_n(u) \in \mathrm{gfp}(\mathrm{CT}_P)$, since $\mathrm{CT}_P \downarrow \omega = \mathrm{gfp}(\mathrm{CT}_P)$. By Lemma 5.18 we have $\mathrm{gfp}(\mathrm{CT}_P) \models \mathrm{comp}(P)$, so $\mathrm{gfp}(\mathrm{CT}_P) \models \mathrm{comp}(P) \cup \{\exists(A_1 \wedge \cdots \wedge A_n \wedge \sigma_0)\}$. □

Theorem 5.19 is an immediate consequence of the corresponding theorem in [8]. Its proof is much simpler than the proof in [12] for SLD-resolution, due to the fact that the model $\mathrm{gfp}(\mathrm{CT}_P)$ is a complete Herbrand model. In the case of SLD-resolution, a completeness theorem for negation as failure was proved in [9] and

[15] by constructing a model for $\mathrm{comp}(P) \cup \{\exists(A_1 \wedge \cdots \wedge A_n \wedge \sigma_0)\}$ which is not a Herbrand model. This serious complication in the proof is overcome by the fact that $\mathrm{CT}_P \downarrow \omega = \mathrm{gfp}(\mathrm{CT}_P)$ (for other interpretations of fixed points see for instance [11]).

We have the following corollary which cannot be obtained in ordinary SLD-semantics.

Corollary 5.20. $\mathrm{comp}(P) \cup \{A \wedge \sigma\}$ *has a complete Herbrand model iff it has a model.*

Acknowledgment

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

References

[1] K.R. Apt and M.H. van Emden, Contributions to the theory of logic programming, *J. Assoc. Comput. Mach.* **29** (1982) 841-862.

[2] A. Colmerauer, Prolog and infinite trees, in: Clark and Tarnlund, eds., *Logic Programming*, (Academic Press, New York, 1982) 231-251.

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

[4] B. Courcelle, Fundamental properties of infinite trees, *Theoret. Comput. Sci.* **25** (1983) 95-169.

[5] M.H. van Emden and R.A. Kowalski, The semantics of predicate logic as a programming language, *J. Assoc. Comput. Mach.* **23** (1976) 733-742.

[6] M.H. van Emden and J.W. Lloyd, A logical reconstruction of Prolog II, *J. Logic Programming* **1** (1984) 143-150.

[7] J. Jaffar and L.-L. Lassez, Constraint logic programming, in: *Proc. POPL*, Munich (1987) 111-119.

[8] J. Jaffar, J-L. Lassez and M.J. Maher, Prolog II as an instance of the logic programming language scheme, in: M. Wirsing, ed., *Proc. IFIP Conf.* (North-Holland, Amsterdam, 1987).

[9] J. Jaffar, J-L. Lassez and J.W. Lloyd, Completeness of the negation as finite failure rule, in: *Proc. IJCAI-83*, Karlsruhe (1983) 500-506.

[10] J. Jaffar and P. Stuckey, Semantics of infinite tree logic programming, *Theoret. Comput. Sci.* **46** (1986) 141-158.

[11] G. Levi and C. Palamidessi, Contributions to the semantics of logic perpetual processes, Draft manuscript, Dipartimento di Informatica, Università di Pisa, Italy, 1987.

[12] J.W. Lloyd, *Foundations of Logic Programming* (Springer-Verlag, Heidelberg, 1984).

[13] M.J. Maher, Logic semantics for a class of committed-choice programs, in: *Proc. 4th. Int. Conf. on Logic Programming*, Melbourne (1987) 858-876.

[14] D.A. Plaisted, The occur-check problem in Prolog, in: *Proc. Int. Symp. on Logic Programming*, Atlantic City (1984) 272-280.

[15] D.A. Wolfram, M.J. Maher and J.-L. Lassez, A unified treatment of resolution strategies for logic programs, in: *Proc. Second Int. Logic Programming Conf.*, Uppsala (1984) 263-276.