A complete axiomatization of the three-valued completion of logic programs

Robert F. Stärk
Institut für Informatik und angewandte Mathematik, Universität Bern
Längassstrasse 51, CH-3012 Bern, Switzerland, <staerk@iam.unibe.ch>

Abstract

We prove the completeness of extended SLDNF-resolution for the new class of ε-programs with respect to the three-valued completion of a logic program. Not only the class of allowed programs but also the class of definite programs are contained in the class of ε-programs. To understand better the three-valued completion of a logic program we introduce a formal system for three-valued logic in which one can derive exactly the three-valued consequences of the completion of a logic program. The system is proof theoretically interesting, since it is a fragment of Gentzen's sequent calculus LK.

Keywords: Logic programming; three-valued logic; negation as failure; SLDNF-resolution; sequent calculus.

1 Introduction

Negation as failure does not have a simple logical explanation. Procedurally, it can easily be described by the two rules

> the literal $\neg A$ succeeds if A fails,
> the literal $\neg A$ fails if A succeeds with answer the identity substitution.

Formally it is defined as SLDNF-resolution which is SLD-resolution plus negation as failure. Clark has introduced in [3] the completion of a logic program as a declarative semantics for SLDNF-resolution (or his query evaluation procedure). He proved the result that

> if the query `?-` $L_1, \ldots, L_n$ succeeds from a program P with answer θ then $comp(P) \models \forall(L_1 \wedge \ldots \wedge L_n)\theta$ and
> if the query `?-` $L_1, \ldots, L_n$ fails from P then $comp(P) \models \neg\exists(L_1 \wedge \ldots \wedge L_n)$.

Appeared in: *J. of Logic and Computation*, 1(6):811–834, 1991.

To have a satisfactory declarative semantics for SLDNF-resolution one wants also the opposite directions of these statements. But this is not possible in general since the completion of a program can be inconsistent or a query can flounder.

Kunen has defined in [8] the three-valued completion of a logic program. This seems to be a good semantics for negation as failure, since the soundness results above of Clark remain valid if one replaces the classical consequence relation '$\models$' by the three-valued relation '$\models_3$' and, on the other hand, SLDNF-resolution is complete with respect to this semantics for the class of allowed programs. However, the condition of allowedness is very stringent since it excludes many common Prolog constructs. For example the following fragment of a program is not allowed:

$$\begin{array}{l} \mathrm{member}(X,[X|L]). \\ \mathrm{member}(X,[Y|L]) \texttt{ :- } \mathrm{member}(X,L). \\ \quad \vdots \\ \mathrm{free}(L,X) \texttt{ :- } \mathrm{good}(X,L),\ \mathrm{not\ member}(X,L). \\ \quad \vdots \\ \mathrm{good}(X,L) \texttt{ :- } \ldots. \end{array}$$

Thus the problem is to characterize a class of logic programs which includes common Prolog constructs and for which SLDNF-resolution is still complete. In Section 4 we introduce the new class of ε-programs. ε stands for the empty substitution or equivalently for the answer 'yes'. We prove that ESLDNF-resolution, a save extension of SLDNF-resolution, is complete for this class with respect to the three-valued completion. The difference between SLDNF- and ESLDNF-resolution is, that in ESLDNF-resolution a non-ground negative literal $\neg A$ may be chosen, and it succeeds and fails according to the rules above.

The class of ε-programs contains the class of allowed programs. It also contains the class of definite programs. Therefore our completeness result extends the results of Shepherdson in [13], Cavedon and Lloyd in [2] and Kunen in [9]. In the program above, the ε property means that if the goal $\texttt{?-}\,\mathrm{good}(s,t)$ succeeds with answer ε, then the terms s and t have to be closed or the goal $\texttt{?-}\,\mathrm{member}(s,t)$ has to fail.

Having this large class of programs for which ESLDNF-resolution is complete in three-valued logic the question is, if there exists a sound and complete formalization of the three-valued logic used in the completeness proof. The answer is yes. In Section 5 we introduce a new formal system for three-valued logic in which one can derive exactly the three-valued consequences of the completion of a logic program. Thus three-valued logic becomes more perspicuous. The three-valued completion is only an approximation for ESLDNF-resolution. The real formal counterpart to it is the system presented in Section 5. The system is proof theoretically interesting, since it is a fragment of Gentzen's sequent calculus LK for classical logic.

The plan of this paper is as follows. In Section 2 we summarize the three-valued completion $comp(P)$ of a logic program and we introduce a new weak completion $comp^{-}(P)$ of a program, which has the same three-valued consequences as $comp(P)$. Section 3 is concerned with the theory of ESLDNF-resolution. In Section 4 we introduce the class of ε-programs and prove that ESLDNF-resolution is complete for it. In Section 5 we present the new formal system LK(P) and prove that it is a sound and complete axiomatization of the three-valued completion of a logic program.

2 Three-valued logic

Let $\mathcal{L}$ be a fixed first order language with equality. We do not make any assumptions about the number of function symbols or predicate symbols of $\mathcal{L}$, but we assume throughout this paper that all programs and goals are written in this language. The terms $r, s, t, \ldots$ and the formulas $A, B, C, F, \ldots$ of $\mathcal{L}$ are defined as usual. The literals $L, \ldots$ of $\mathcal{L}$ are the atomic and negated atomic formulas of $\mathcal{L}$. We write $A[\vec{x}\,]$ to indicate that all free variables of A are from the list $\vec{x}$; analogous, $t[\vec{x}\,]$ stands for a term with no variables different from $\vec{x}$. An expression of the form

$$A \texttt{ :- } L_1, \ldots, L_n$$

where A is atomic, $0 \leq n$ and the L_i's are literals is called a *program clause*. The atom A is the *head* of the clause and the sequence $L_1, \ldots, L_n$ is the *body* of the clause. A *program* is a finite set of program clauses. A *goal* is an expression of the form

$$\texttt{?- } L_1, \ldots, L_n.$$

We assume that in programs and goals the equality symbol does not occur. Capital greek letters $\Gamma, \Delta, \Lambda, \Pi, \ldots$ denote finite lists of literals. Thus clauses will be denoted by $A \texttt{ :- } \Pi$ and goals simply by Γ (we omit the '?-' sign). $\emptyset$ is the empty goal. Small greek letters $\alpha, \beta, \gamma, \theta, \varphi, \chi, \ldots$ denote substitutions, ε is the empty substitution.

We summarize now the definition of the *completion* of a logic program. Let P be a program and r be a n-ary predicate symbol. We assume that there are m clauses in P which define r and that the i-th clause is of the form

$$r(t_{i,1}[\vec{y}\,], \ldots, t_{i,n}[\vec{y}\,]) \texttt{ :- } L_{i,1}[\vec{y}\,], \ldots, L_{i,k(i)}[\vec{y}\,].$$

Then one defines the *defining formula* for r as

$$D_r[x_1, \ldots, x_n] := \bigvee_{i=1}^{m} \exists \vec{y} \, \Big(\bigwedge_{j=1}^{n} x_j = t_{i,j}[\vec{y}\,] \wedge \bigwedge_{j=1}^{k(i)} L_{i,j}[\vec{y}\,] \Big)$$

and the *completed definition* of r as the formula

$$\forall \vec{x}\,(r(\vec{x}\,) \leftrightarrow D_r[\vec{x}\,]).$$

The cases $m = 0$ or $k(i) = 0$ are treated in a natural way. The empty disjunction is the constant $\bot$ and the empty conjunction is the constant $\top$. The completion $comp(P)$ is obtained from P by taking all completed definitions of all predicates of $\mathcal{L}$ and the following equality and freeness axioms for $\mathcal{L}$, the so called theory *CET* (Clark's equality theory).

(1) $x = x$

(2) $x = y \rightarrow y = x$

(3) $x = y \wedge y = z \rightarrow x = z$

(4) $x_1 = y_1 \wedge \ldots \wedge x_n = y_n \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$

(5) $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \rightarrow x_i = y_i \qquad$ (for $1 \leq i \leq n$)

(6) $f(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_m) \qquad$ (if $f \neq g$)

(7) $t \neq x \qquad$ (if t is a term different from x and $x \in \mathrm{var}(t)$)

The axiom $x_1 = y_1 \wedge \ldots \wedge x_n = y_n \wedge r(x_1, \ldots, x_n) \rightarrow r(y_1, \ldots, y_n)$ is not needed because it is derivable from $comp(P)$.

Following Kunen in [8] and [9] we use the three-valued logic of Kleene for the interpretation of $comp(P)$. In this logic the three truth values are $\mathbf{t}$ (true), $\mathbf{f}$ (false) and $\mathbf{u}$ (undefined) with the partial ordering defined by $\mathbf{u} < \mathbf{t}$ and $\mathbf{u} < \mathbf{f}$. Then $x \leq y$ is equivalent to the two statements

(1) if $x = \mathbf{t}$ then $y = \mathbf{t}$ and

(2) if $x = \mathbf{f}$ then $y = \mathbf{f}$.

The formula $A \wedge B$ is true iff both A and B are true and it is false iff one of A and B is false and undefined otherwise. The formula $A \vee B$ is true iff A or B is true and $A \vee B$ is false iff A and B are false and $A \vee B$ is undefined otherwise. The formula $\neg A$ is true iff A is false and it is false iff A is true and undefined otherwise.

A three-valued structure $\mathcal{A}$ is a non-empty set $|\mathcal{A}|$ of objects together with interpretations of the function symbols and the equality relation in a two-valued manner and interpretations of the n-ary predicates as functions from $|\mathcal{A}|^n$ into $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$. The quantifiers $\forall$ and $\exists$ are treated as infinite conjunctions and disjunctions. So $\mathcal{A}(\forall x A[x, \vec{a}\,]) = \mathbf{t}$ iff for all $b \in |\mathcal{A}|$: $\mathcal{A}(A[b, \vec{a}\,]) = \mathbf{t}$ and $\mathcal{A}(\forall x A[x, \vec{a}\,]) = \mathbf{f}$ iff there exists a $b \in |\mathcal{A}|$ such that $\mathcal{A}(A[b, \vec{a}\,]) = \mathbf{f}$. $\mathcal{A}(\exists x A[x, \vec{a}\,]) = \mathbf{t}$ iff there exists a $b \in |\mathcal{A}|$ such that $\mathcal{A}(A[b, \vec{a}\,]) = \mathbf{t}$ and $\mathcal{A}(\exists x A[x, \vec{a}\,]) = \mathbf{f}$ iff for all $b \in |\mathcal{A}|$: $\mathcal{A}(A[b, \vec{a}\,]) = \mathbf{f}$.

We write $\mathcal{A} \models_3 A[\vec{x}\,]$ iff for all $\vec{a} \in |\mathcal{A}|$: $\mathcal{A}(A[\vec{a}\,]) = \mathbf{t}$ and for theories T we write $T \models_3 A$ iff for all structures $\mathcal{A}$: $\mathcal{A} \models_3 T$ implies $\mathcal{A} \models_3 A$.

Kunen has given to the connective '$\leftrightarrow$' in the completed definition the following interpretation: $A \leftrightarrow B$ is true in a model if and only if A and B have the same truth value. This information about '$\leftrightarrow$' suffices, since we are only interested in the relation $comp(P) \models_3 F$, where the formula F contains $\neg, \wedge, \vee, \exists$ and $\forall$. Such formulas are called *Kleene formulas*. We will show below that one can replace '$\leftrightarrow$' by a weaker connective.

On three-valued structures the following $\leq$ relation is defined. Let $\mathcal{A} \leq \mathcal{B}$ iff $\mathcal{A}$ and $\mathcal{B}$ have the same universe and the same interpretation of functions and equality and for all predicates r, for all $\vec{a} \in |\mathcal{A}|$: $\mathcal{A}(r(\vec{a}\,)) \leq \mathcal{B}(r(\vec{a}\,))$. It is easy to see that this is a partial ordering. If $\mathcal{A} \leq \mathcal{B}$ and $F[\vec{x}\,]$ is a Kleene formula then $\mathcal{A}(F[\vec{a}\,]) \leq \mathcal{B}(F[\vec{a}\,])$ for all $\vec{a} \in |\mathcal{A}|$.

In [5] Fitting has defined for a program P an operator Φ_P which is the three-valued analogue of the 'immediate consequence operator' T_P of logic programming. The operator Φ_P assigns to a three-valued structure $\mathcal{A}$ a new structure $\Phi_P\,\mathcal{A}$ defined by

$$\begin{aligned} \Phi_P\,\mathcal{A}(r(\vec{a}\,)) = \mathbf{t} \quad &:\Longleftrightarrow \quad \mathcal{A}(D_r[\vec{a}\,]) = \mathbf{t}, \\ \Phi_P\,\mathcal{A}(r(\vec{a}\,)) = \mathbf{f} \quad &:\Longleftrightarrow \quad \mathcal{A}(D_r[\vec{a}\,]) = \mathbf{f}, \end{aligned}$$

where the completed definition of r in P is $\forall \vec{x}\,(r(\vec{x}\,) \leftrightarrow D_r[\vec{x}\,])$. The operator Φ_P is monotonic, since $\mathcal{A} \leq \mathcal{B}$ implies $\Phi_P\,\mathcal{A} \leq \Phi_P\,\mathcal{B}$. A structure $\mathcal{A}$ satisfying *CET* is a fixpoint of Φ_P iff $\mathcal{A}$ is a model of $comp(P)$.

Now we introduce the *weak completion* of a logic program. We denote by $comp^-(P)$ the theory which is obtained from $comp(P)$ if one replaces the equivalence '$\leftrightarrow$' in the completed definitions by a new connective '$\Leftarrow$' such that they are of the form

$$\forall \vec{x}\,(r(\vec{x}\,) \Leftarrow D_r[\vec{x}\,]).$$

The interpretation of '$\Leftarrow$' is as follows: $A \Leftarrow B$ is true in a model iff the truth value of B is less than or equal to the truth value of A and false otherwise. Hence we have that a structure $\mathcal{A}$ is a model of $comp^-(P)$ iff $\mathcal{A}$ is closed under Φ_P. In the case A, B have values among $\mathbf{t}$, $\mathbf{f}$ $A \Leftarrow B$ has the same truth value as $A \leftrightarrow B$. The difference between $comp(P)$ and $comp^-(P)$ can be expressed by the following two equivalences:

$$\begin{aligned} \Phi_P\,\mathcal{A} = \mathcal{A} \quad &\Longleftrightarrow \quad \mathcal{A} \models_3 comp(P), \\ \Phi_P\,\mathcal{A} \leq \mathcal{A} \quad &\Longleftrightarrow \quad \mathcal{A} \models_3 comp^-(P). \end{aligned}$$

We want now to show that $comp^-(P)$ has the same logical consequences as $comp(P)$. To prove this we need the following lemma.

Lemma 1 If $\Phi_P \mathcal{A} \leq \mathcal{A}$ then there exists a structure $\mathcal{B} \leq \mathcal{A}$ such that $\Phi_P \mathcal{B} = \mathcal{B}$.

Proof. If $\Phi_P \mathcal{A} \leq \mathcal{A}$ then Φ_P is monotonic on the cpo $\{\mathcal{B} \mid \mathcal{B} \leq \mathcal{A}\}$, so it must have a fixed point there. □

Lemma 2 If F is a Kleene formula then $comp(P) \models_3 F$ iff $comp^-(P) \models_3 F$.

Proof. The direction from right to left is trivial. For the direction from left to right we assume that $comp(P) \models_3 F$ and that $\mathcal{A} \models_3 comp^-(P)$. Then we have $\Phi_P \mathcal{A} \leq \mathcal{A}$ and by Lemma 1 there exists a $\mathcal{B} \leq \mathcal{A}$ with $\Phi_P \mathcal{B} = \mathcal{B}$. Now $\mathcal{B} \models_3 comp(P)$ and $\mathcal{B} \models_3 F$ and since F is a Kleene formula $\mathcal{A} \models_3 F$. □

Lemma 1 and 2 will essentially be used in the proofs of the main theorems of Section 4 and 5. The next section is about the theory of extended SLDNF-resolution.

3 Theory of extended SLDNF-resolution

In this section we give a formal definition of extended SLDNF-resolution and prove some basic facts about it. We will use these technical lemmas in the completeness proof of Section 4. In Shepherdson's papers [12] and [13] some of them are proved in detail, but the version of SLDNF-resolution that he uses is weaker than our version. We do not require that negative selected literals have to be closed. So we will prove the lemmas here again but every time when there is a correspondence to [12] and [13] we will indicate it. The most interesting new results of this section are Lemma 14 and Corollary 15.

One serious problem of the negation as failure rule is its nested use. Negation as failure is easily implemented. Unfortunately it is much more difficult to describe and to understand what happens in a query evaluation process. The following two definitions are an adaptation of the definitions in §15 of Lloyd [10].

Definition 3 A goal Γ' is derived from Γ using (the selected atom) A, (the input clause) C and (the most general unifier) θ if Γ is of the form Δ_0, A, Δ_1 and C of the form $B \text{ :- } \Pi$ and θ is a most general unifier[1] of A and B and $\Gamma' = (\Delta_0, \Pi, \Delta_1)\theta$.

Let P be a program and Γ be a goal. One defines by recursion on k the two notions

(a) $\Gamma_0, \Gamma_1, \ldots, \Gamma_n, \theta_1, \ldots, \theta_n$ is an ESLDNF proof of P/Γ of rank k with answer θ and

[1] We assume in this paper that a most general unifier is idempotent, i. e. $\theta\theta = \theta$. This implies especially that $\mathrm{var}(\theta) \subset \mathrm{var}(A) \cup \mathrm{var}(B)$.

(b) T is a finitely failed ESLDNF tree for P/Γ of rank k.

Definition 4 (a) $\Gamma_0, \Gamma_1, \ldots, \Gamma_n, \theta_1, \ldots, \theta_n$ is an ESLDNF proof of P/Γ of rank k with answer θ, if $\Gamma_0 = \Gamma$ and $\Gamma_n = \emptyset$ (empty goal) and $\theta = \theta_1 \cdots \theta_n \restriction \mathrm{var}(\Gamma)$ and for every $i < n$ there is in Γ_i a literal L (the selected literal), i. e. Γ_i is of the form Δ_0, L, Δ_1, such that

(a+) if L is positive then there is a variant C of a clause of P, in which no variables occur from $\Gamma_0\theta_1 \cdots \theta_i$ or Γ_i, and Γ_{i+1} is derived from Γ_i using L, C and θ_{i+1},

(a−) if L is the negative literal $\neg A$ then there is a finitely failed ESLDNF tree for P/A of rank less than k and $\Gamma_{i+1} = \Delta_0, \Delta_1$ and $\theta_{i+1} = \varepsilon$.

(b) T is a finitely failed ESLDNF tree for P/Γ of rank k, if T is a finite tree with root Γ consisting of non empty goals such that in every node Δ of T there is a literal L (the selected literal), i. e. Δ is of the form Δ_0, L, Δ_1, such that

(b+) if L is positive then there is for every clause of P, which has a variant unifying L, exactly one variant C with no variables in common to Δ and a child Δ' derived from Δ using L, C and some most general unifier, and Δ has no other children,

(b−) if L is the negative literal $\neg A$ then there is an ESLDNF proof of P/A of rank less than k with answer ε and Δ has no children.

This definition is more general than the usual definition of SLDNF-resolution as presented for example in Lloyd [10]. Normally in the steps (a−) and (b−) the literal $\neg A$ has to be closed. But we think that there is no reason to require this. The ESLDNF derivation procedure corresponds to the implementation of negation in IC-Prolog [4].

We say that a goal Γ *succeeds with answer* θ (from P) or equivalently θ *is a computed answer for* Γ if there exists a $k \geq 0$ and an ESLDNF proof of P/Γ of rank k with answer θ. A goal Γ *is finitely failed* (from P) if there exists a $k \geq 0$ and a finitely failed ESLDNF tree for P/Γ of rank k.

If T is a finitely failed ESLDNF tree for P/Γ of rank k with less than or equal to n nodes then we say that Γ is finitely failed of rank (k, n). We take the lexicographical ordering on pairs of natural numbers to compare the ranks of finitely failed goals.

If $k \leq k'$ then a proof (finitely failed tree) of rank k is also a proof (finitely failed tree) of rank k'.

The following lemma is a version of the lifting lemma which we think is the most appropriate and the most useful tool for proving theorems about ESLDNF-resolution.

Lemma 5 (Lifting) If Γ is the goal Δ_0, A, Δ_1 and C the clause $B \,:\!-\, \Pi$ and if σ and τ are substitutions such that $A\sigma = B\tau$ then the head of every variant C' of C which has no variables common to Γ is unifiable with A, and if Γ' is the goal derived from Γ using A, C' and some most general unifier θ then there exists a substitution α such that one obtains $\Gamma'\alpha$ from $\Gamma\sigma$ if one replaces $A\sigma$ by the body $\Pi\tau$ of clause $C\tau$, i. e. $\Gamma'\alpha = \Delta_0\sigma, \Pi\tau, \Delta_1\sigma$.

Proof. Assume that $A\sigma = B\tau$ and that C' is a variant of C such that $\mathrm{var}(C') \cap \mathrm{var}(\Gamma) = \emptyset$. Then there is a permutation of variables η such that $C' = C\eta$. Let χ be the substitution $\sigma \restriction \mathrm{var}(\Gamma) \cup (\eta^{-1}\tau) \restriction \mathrm{var}(C')$. The substitution χ acts on Γ like σ and on C' like $\eta^{-1}\tau$. Then

$$A\chi = A\sigma = B\tau = B\eta(\eta^{-1}\tau) = (B\eta)\chi$$

and A and $B\eta$ are unifiable. Let θ be a most general unifier of A and $B\eta$. Now there exists a substitution α such that $\chi = \theta\alpha$. Thus $\Gamma' = (\Delta_0, \Pi\eta, \Delta_1)\theta$ and $\Delta_i\theta\alpha = \Delta_i\chi = \Delta_i\sigma$ and $(\Pi\eta\theta)\alpha = \Pi\eta\chi = \Pi\eta(\eta^{-1}\tau) = \Pi\tau$. Altogether we obtain $\Gamma'\alpha = \Delta_0\sigma, \Pi\tau, \Delta_1\sigma$. $\square$

The next lemma corresponds to the Lemmas 2 and 3 of [12].

Lemma 6 Let Γ be a goal and φ be a substitution.

(a) If $\Gamma_0, \Gamma_1, \ldots, \Gamma_n, \theta_1, \ldots, \theta_n$ is an ESLDNF proof of Γ of rank k with answer θ then there exists an ESLDNF proof of $\Gamma\theta\varphi$ of rank k with answer ε.

(b) If Γ is finitely failed of rank (k, n) then $\Gamma\varphi$ is finitely failed of rank (k, n).

Proof. The two statements (a) and (b) are proved simultaneously by induction on k.
(a) Let $\Gamma_0, \Gamma_1, \ldots, \Gamma_n, \theta_1, \ldots, \theta_n$ be an ESLDNF proof of Γ of rank k with answer θ. Using the induction hypothesis we may assume that there is a $j \leq n$ such that in all Γ_i, $i < j$, only positive literals are selected and Γ_j consists entirely of negative literals $\neg A$ where A is finitely failed of rank less than k. Let $\Gamma' := \Gamma\theta\varphi$. We construct recursively on $i \leq j$ an ESLDNF derivation $\Gamma'_0, \Gamma'_1, \ldots, \Gamma'_i, \theta'_1, \ldots, \theta'_i$ of Γ' and substitutions α_i and β_i such that

(i) $\Gamma'\theta'_1 \cdots \theta'_i\alpha_i = \Gamma'$,

(ii) $\Gamma'_i\alpha_i = \Gamma_i\theta_{i+1} \cdots \theta_n\varphi$,

(iii) $\Gamma_i\beta_i = \Gamma'_i$.

If $i = 0$ then we put $\Gamma'_0 := \Gamma'$, $\alpha_0 := \varepsilon$ and $\beta_0 := \theta_1 \cdots \theta_n\varphi$. Now we assume that $i < j$ and that we have already constructed the derivation up to i. The goal Γ_i is of the form Δ_0, A, Δ_1 and $\Gamma'_i = \Delta'_0, A', \Delta'_1$ and there is a variant $B :\!- \Pi$ of a clause such that $\Gamma_{i+1} = (\Delta_0, \Pi, \Delta_1)\theta_{i+1}$. We put $\tau := \theta_{i+1} \cdots \theta_n\varphi$ and then $A'\alpha_i = A\tau = B\tau$ and by Lemma 5 applied to the enlarged goal $(\Gamma'_0\theta'_1 \cdots \theta'_i, \Gamma'_i)$ one can continue the derivation to $\Gamma'_{i+1} = (\Delta'_0, \Pi\eta, \Delta'_1)\theta'_{i+1}$ for some renaming η, and there is a substitution α_{i+1} such that $(\Gamma'\theta'_1 \cdots \theta'_i\theta'_{i+1}, \Gamma'_{i+1})\alpha_{i+1}$ is obtained from $(\Gamma'_0\theta'_1 \cdots \theta'_i, \Gamma'_i)\alpha_i = (\Gamma', \Gamma_i\tau)$ by replacing $A\tau$ by $\Pi\tau$, i. e.

(i) $\Gamma'\theta'_1 \cdots \theta'_i\theta'_{i+1}\alpha_{i+1} = \Gamma'$,

(ii) $\Gamma'_{i+1}\alpha_{i+1} = \Delta_0\tau, \Pi\tau, \Delta_1\tau = \Gamma_{i+1}\theta_{i+2} \cdots \theta_n\varphi$.

In order to obtain the substitution β_{i+1} we will again use Lemma 5. Since $A\beta_i\theta'_{i+1} = A'\theta'_{i+1} = B\eta\theta'_{i+1}$ there is a substitution β_{i+1} such that $\Gamma_{i+1}\beta_{i+1}$ is obtained from $\Gamma'_i\theta'_{i+1}$ by replacing $A'\theta'_{i+1}$ by $\Pi\eta\theta'_{i+1}$, i. e.

(iii) $\Gamma_{i+1}\beta_{i+1} = \Delta'_0\theta'_{i+1}, \Pi\eta\theta'_{i+1}, \Delta'_1\theta'_{i+1} = \Gamma'_{i+1}$.

Now $\Gamma_j\beta_j = \Gamma'_j$ and Γ'_j consists therefore of negative literals $\neg A$ where A is finitely failed of rank less than k. Since $\Gamma\theta\varphi\theta'_1 \cdots \theta'_j\alpha_j = \Gamma' = \Gamma\theta\varphi$ it follows that $\Gamma\theta\varphi\theta'_1 \cdots \theta'_j$ is a variant of $\Gamma\theta\varphi$ and therefore the computed answer is ε.
(b) We prove by induction on n that if Γ is finitely failed of rank (k, n) then $\Gamma\varphi$ is finitely failed of rank (k, n).
(b$-$) Γ is of the form $\Delta_0, \neg A, \Delta_1$ and A has a proof of rank less than k with answer ε: By the main induction hypothesis on k the atom $A\varphi$ has a proof of rank less than k with answer ε too and $\Gamma\varphi$ is finitely failed of rank (k, n).
(b+) Γ is of the form Δ_0, A, Δ_1 and all goals derived from Γ using A are finitely failed of rank (k, m) with $m < n$: Using Lemma 5 we see that if a variant of some clause unifies with $A\varphi$ then the goal derived from $\Gamma\varphi$ using $A\varphi$ is an instance of a child of Γ and is therefore finitely failed of rank less than (k, n). Putting all together we obtain that $\Gamma\varphi$ is finitely failed of rank (k, n). □

The next lemma is a weakening of the previous one, since we do not consider the ranks of the proofs and finitely failed trees.

Lemma 7 Let Γ be a goal and φ be a substitution.

(a) If Γ succeeds with answer θ then $\Gamma\theta\varphi$ succeeds with answer ε.

(b) If Γ is finitely failed then $\Gamma\varphi$ is finitely failed.

The notion of implication trees, which we shall now introduce, seems to be very useful, because in implication trees the non-determinism in selecting the literals is missing. It is important to note that implication trees are not computations of an ideal logic programming machine like ESLDNF proofs. They are only a tool for proving properties of computations. Using implication trees one can give a very short proof for the completeness of SLD-resolution for definite programs (see Stärk [15]). Closed implication trees were first introduced by Apt, Blair and Walker in [1].

Definition 8 Let L be a literal and P be a program. An *implication tree for* L with respect to P of rank k is a finite tree T whose nodes are literals and whose root is L such that

(a+) if A is a positive node of T then there exists a clause B :- Π in P and a substitution θ such that $A = B\theta$ and the children of A in T are exactly the literals of $\Pi\theta$,

(a−) if $\neg A$ is a negative node of T then A is finitely failed of rank less than k and $\neg A$ has no children.

Using Lemma 6 one sees that if T is an implication tree for L of rank k and σ is a substitution then also $T\sigma$ is an implication tree for $L\sigma$ of rank k.

Lemma 9 If the goal Γ has an ESLDNF proof of rank k with answer θ then every literal in $\Gamma\theta$ has an implication tree of rank k.

Proof. By induction on the length of an ESLDNF proof. □

The reverse of Lemma 9 in not true in general. Take for example the program consisting of the two clauses $r(X)$:- $\neg q(X)$ and $q(0)$ and let $\theta := \{X := 1\}$. Then $r(X)\theta$ has an implication tree but $r(X)$ does not succeed with any answer. Later we will introduce the class of ε-programs for which the reverse of Lemma 9 holds. The next lemma will help us to prove that a goal cannot succeed *and* fail.

Lemma 10 If the goal Γ is finitely failed of rank (k, n) and θ is a substitution then it is not possible that every literal in $\Gamma\theta$ has an implication tree of rank k.

Proof. The proof is by induction on (k, n). Let Γ be finitely failed of rank (k, n). Case−: Suppose that $\Gamma = \Delta_0, \neg A, \Delta_1$ and that A has a proof of rank less than k with answer ε. If we assume that $\neg A\theta$ has an implication tree of rank k then $A\theta$ is finitely failed of rank less than k. But $A\theta$ has by Lemma 6 a proof of rank less than k with answer ε too and by Lemma 9 an implication tree of rank less than k which contradicts the induction hypothesis.

Case+: Suppose that $\Gamma = \Delta_0, A, \Delta_1$ and every derived goal from Γ using A is finitely failed of rank less than (k, n). If we assume that every literal in $\Gamma\theta$ has an implication tree of rank k then in particular $A\theta$ has one. This means that there exists a clause B :- Π and a substitution τ such that $A\theta = B\tau$ and every literal in $\Pi\tau$ has an implication tree of rank k. Let Γ' be the goal derived from Γ using A and (a variant of) B :- Π. By Lemma 5 such a goal exists and there is a substitution σ such that $\Gamma'\sigma = \Delta_0\theta, \Pi\tau, \Delta_1\theta$. Since Γ' is finitely failed of rank less than (k, n) this contradicts the induction hypothesis. □

The next lemma follows from Lemma 9 and Lemma 10 and it corresponds to Shepherdson's Theorem 4 of [12]. It justifies ESLDNF-resolution.

Lemma 11 If Γ is finitely failed then Γ does not succeed with any answer.

The following lemma is exactly Lemma 6 of [12]. We omit its proof.

Lemma 12 If the goal Γ, Δ is finitely failed and if Γ and Δ have no variables in common then either Γ or Δ is finitely failed.

One can also prove something like a 'cut rule' for ESLDNF-resolution. This is not the same kind of cut that is used in the formal system of Section 5.

Lemma 13 If the goal Γ, Δ is finitely failed of rank (k, n) and every literal in Δ has an implication tree then Γ is finitely failed of rank (k, n).

Proof. The proof is by induction on (k, n).
Case−: Suppose that the selected literal in Γ, Δ is $\neg A$ and A succeeds with answer ε. Then $\neg A$ cannot be in Δ since then A would be finitely failed and this would contradict Lemma 11. Therefore $\neg A$ is in Γ and Γ is finitely failed of rank (k, n).
Case+: Suppose that the selected literal is A and every goal derived from Γ, Δ using A is finitely failed of rank less than (k, n).
Case+ 1: A is in Γ: All goals derived from Γ, Δ using A are of the form $\Lambda, \Delta\sigma$ where Λ is derived from Γ. Since every literal in $\Delta\sigma$ has an implication tree too, every Λ is finitely failed by induction hypothesis and therefore Γ is finitely failed of rank (k, n).
Case+ 2: A is in Δ: Then Δ is of the form Δ_0, A, Δ_1 and there is a clause B :- Π and a substitution τ such that $A = B\tau$ and every literal in $\Pi\tau$ has an implication tree. By Lemma 5 there is a goal Λ derived from Γ, Δ and a substitution α such that $\Lambda\alpha = \Gamma, \Delta_0, \Pi\tau, \Delta_1$. Since Λ is finitely failed of rank less than (k, n) by Lemma 6 $\Lambda\alpha$ is finitely failed of rank less than (k, n) too and by induction hypothesis Γ is finitely failed of rank (k, n). □

From Lemma 9 and Lemma 13 we obtain

Lemma 14 (Cut rule) If the goal Γ, Δ is finitely failed and Δ succeeds with answer ε then Γ is finitely failed.

Corollary 15 If the goal Γ, Δ is finitely failed and Δ succeeds with answer θ then $\Gamma\theta$ is finitely failed.

Proof. If Δ succeeds with answer θ then by Lemma 7 the goal $\Delta\theta$ succeeds with answer ε. If Γ, Δ is finitely failed then by Lemma 7 the goal $\Gamma\theta, \Delta\theta$ is finitely failed and by Lemma 14 the goal $\Gamma\theta$ is finitely failed. □

4 Completeness of extended SLDNF-resolution

Until this point all properties of ESLDNF-resolution were proved for any given program. We now define the class of ε-programs and prove that ESLDNF-resolution is complete for it.

Definition 16 Let Γ be a goal and P be a program. Γ is an *ε-goal* for P if for every substitution θ:

if every positive literal in $\Gamma\theta$ succeeds with answer ε
then
every negative literal $\neg A\theta$ in $\Gamma\theta$ is closed or $A\theta$ is finitely failed.

P is an *ε-program* if for every clause $A \text{ :- } \Gamma$ of P the body Γ is an ε-goal for P.

It is easy to see that the ε-property is not decidable. But it is even worse. I am grateful to the referee for bringing the following argument to my attention which shows that the set of (finite) ε-programs is a complete Π^0_2 set.

Proof: It is obviously Π^0_2. Let $\{k \mid \forall n \exists m(k, n, m) \in A\}$ be a complete Π^0_2 set, with A primitive recursive. Let Q be a definite program which defines the complement of A in the sense that $\texttt{?-}\, b(s^k(0), s^n(0), s^m(0))$ succeeds if $(k, n, m) \notin A$ and fails finitely if $(k, n, m) \in A$. Let P_k be the program Q together with:

$g \text{ :- } p(N),\ \neg q(s^k(0), N, Y).$
$p(0).$
$p(s(N)) \text{ :- } p(N).$

$q(K, N, Y) \text{ :- } r(K, N, 0).$
$r(K, N, M) \text{ :- } b(K, N, M),\ r(K, N, s(M)).$

Then every clause body of P_k is an ε-goal except possibly $p(N),\ \neg q(s^k(0), N, Y)$. Since $p(N)$ succeeds only with the substitutions $\{N := s^n(0)\}$, P_k is an ε-program iff for all n, $r(s^k(0), s^n(0), 0)$ is finitely failed. Note that $r(s^k(0), s^n(0), 0)$ never succeeds, and is finitely failed iff for some m, $b(s^k(0), s^n(0), s^m(0))$ fails. Thus P_k is an ε-program iff $\forall n \exists m(k, n, m) \in A$.

However, there are some well known subclasses of ε-programs which are defined purely syntactically.

Remarks 17 1° If P is *definite* then P is an ε-program, since in P there are no negated atoms.
2° A clause is called *allowed* if every variable of the clause occurs also in a positive literal of the body of the clause. A program is allowed if every clause of it is allowed. A goal is allowed if every variable of the goal occurs also in a positive literal of the goal. Now if P is allowed then P is an ε-program, and if P and Γ are allowed then Γ is an ε-goal for P.
3° P is *quasi-definite* if for every negative literal $\neg A$ in the body of a clause of P the atom A does not unify with the head of any clause in P. Now if P is quasi-definite then P is an ε-program.
4° The programs which are *safe for negation* of Van Gelder in [16] are ε-programs.

Definition 18 A *weak implication tree* T for L with respect to P is defined like an implication tree for L (Definition 8) but clause (a−) is replaced by

(a−) if $\neg A$ is a negative node of T then there exists a substitution σ such that $A\sigma$ is finitely failed and $\neg A$ has no children.

The notion of weak implication trees is only an ad hoc notion. Every implication tree is also a weak implication tree. For ε-programs one can prove now the reverse of Lemma 9.

Lemma 19 Suppose that Γ is an ε-goal for the ε-program P, that φ is a substitution and that every literal in $\Gamma\varphi$ has a weak implication tree. Then there exists a computed answer θ of Γ and a substitution α such that $\Gamma\theta\alpha = \Gamma\varphi$.

Proof. By induction on the total number of nodes of the weak implication trees of $\Gamma\varphi$. Assume that there are n positive and k negative nodes in the weak implication trees of $\Gamma\varphi$. Then using Lemma 5 one can construct an ESLDNF derivation $\Gamma_0, \Gamma_1, \ldots, \Gamma_n$, $\theta_1, \ldots, \theta_n$ of Γ and a substitution α such that $\Gamma_0\theta_1 \cdots \theta_n\alpha = \Gamma\varphi$ and in Γ_i $(i < n)$ only positive literals are selected and $\Gamma_n\alpha$ consists of the k negative literals of the weak implication trees. If we can prove that for every literal $\neg B$ of Γ_n the atom B is finitely failed we are done.

Assume that $\neg B$ is in the body Π of a clause which was used in the resolution step from Γ_i to Γ_{i+1} (or that $\neg B$ is in Γ_0 and $i = 0$). Then $\neg B\theta_{i+1} \cdots \theta_n$ is in Γ_n. Every positive literal of $\Pi\theta_{i+1} \cdots \theta_n$ (or $\Gamma_0\theta_1 \cdots \theta_n$) has a weak implication tree with less than $n + k$ nodes and since a positive literal is trivially an ε-goal for P, by induction hypothesis, it succeeds with answer ε. Since P is an ε-program (and Γ is an ε-goal for P) $\neg B\theta_{i+1} \cdots \theta_n$ is closed or $B\theta_{i+1} \cdots \theta_n$ is finitely failed. If $\neg B\theta_{i+1} \cdots \theta_n$ is closed then $B\theta_{i+1} \cdots \theta_n$ is finitely failed since it is a leave of a weak implication tree. □

As an immediate consequence we have:

Lemma 20 If P is an ε-program and A :- Γ a clause of P and θ a substitution, and if every positive literal of $\Gamma\theta$ succeeds with answer ε and every negative literal $\neg B\theta$ in $\Gamma\theta$ is such that $B\theta$ is finitely failed, then $A\theta$ succeeds with answer ε.

There is an alternative way to characterize the class of ε-programs.

Remark 21 If one denotes the sequence of all positive literals of a goal Γ by Γ^+ and the sequence of all negative literals by Γ^- then one can define the notions of an ε-goal for an ε-program in the following equivalent way: Γ is called a *regular goal* for P if for every computed answer θ of Γ^+ every literal $\neg B\theta$ in $\Gamma^-\theta$ is closed or $B\theta$ is finitely failed. P is a *regular program* if for every clause A :- Γ of P the body Γ is a regular goal for P.
One can prove Lemma 19 in an analogous way for regular programs, and using this fact it is easy to see that P is an ε-program iff P is a regular program, and if this is the case then Γ is an ε-goal for P iff it is a regular goal for P.

We come to the main theorem of this section. It corresponds to the completeness of the negation as failure rule for definite programs which was proved by Jaffar, Lassez and Lloyd in [7].

Theorem 22 Let P be an ε-program. If the goal ?- $L_1, \ldots, L_q$ is not finitely failed then there exists a countable three-valued structure $\mathcal{M}$ with

(1) $\mathcal{M}$ is a model of $comp(P)$,

(2) $\neg\exists(L_1 \wedge \ldots \wedge L_q)$ is not true in $\mathcal{M}$,

(3) if an atom A is true in $\mathcal{M}$ then some instance of A succeeds with answer ε,

(4) if a closed atom A is false in $\mathcal{M}$ then A is finitely failed.

Proof. Note that $\neg\exists(L_1 \wedge \ldots \wedge L_q)$ is not true in a model $\mathcal{M}$ iff there is a variable assignment α such that $\mathcal{M}(L_i, \alpha) \neq \mathbf{f}$ for $i = 1, \ldots, q$.

Let $A_0, A_1, \ldots$ be an enumeration of all atoms of $\mathcal{L}$ such that every atom occurs infinitely many often in the enumeration. Let $\Gamma_0 := L_1, \ldots, L_q$. We will construct by recursion a sequence $\Gamma_0, \Gamma_1, \ldots$ of non-finitely failed goals and a sequence $\theta_1, \theta_2, \ldots$ of substitutions. This sequence will be something like a generalized infinite fair SLD-derivation of Γ_0.

We assume that $\Gamma_0, \Gamma_1, \ldots, \Gamma_n$ and $\theta_1, \ldots, \theta_n$ are already constructed and that Γ_n is not finitely failed. We consider two cases.
Case $n = 2i$: Assume that $\Gamma_n = \Delta_0, A, \Delta_1$ and that A is the leftmost positive literal

in Γ_n. Since Γ_n is not finitely failed there is a variant C of a clause of P of the form B :- Π and a most general unifier θ_{n+1} of A and B such that $(\Delta_0, \Delta_1, \Pi)\theta_{n+1}$ is not finitely failed. We put $\Gamma_{n+1} := (\Delta_0, \Delta_1, \Pi)\theta_{n+1}$. If there is no positive literal in Γ_n then we put $\Gamma_{n+1} := \Gamma_n$ and $\theta_{n+1} := \varepsilon$.
Case $n = 2i+1$: If $\Gamma_n, (A_i\theta_1 \cdots \theta_n)$ is not finitely failed then $\Gamma_{n+1} := \Gamma_n, (A_i\theta_1 \cdots \theta_n)$ else $\Gamma_{n+1} := \Gamma_n$. In every case $\theta_{n+1} := \varepsilon$.

It is easy to see that all compositions $\theta_1 \cdots \theta_n$ are idempotent (i. e. $\theta_1 \cdots \theta_n = \theta_1 \cdots \theta_n\theta_1 \cdots \theta_n$) and that for every Γ_n there exists a Γ'_n such that $\Gamma_n = \Gamma'_n\theta_1 \cdots \theta_n$. (We assume that in step $n = 2i$ the clause C has no variables affected by $\theta_1 \cdots \theta_n$.) Let Γ be the set of all literals which occur in some Γ_n. We define now a three-valued structure $\mathcal{A}$. Let the universe $|\mathcal{A}|$ of $\mathcal{A}$ be the set of all terms of $\mathcal{L}$ and $f^{\mathcal{A}}(t_1, \ldots, t_n) := f(t_1, \ldots, t_n)$. Like in Theorem 16.1 of Lloyd [10] a binary relation is defined on $|\mathcal{A}|$ by

$$s \sim t : \Longleftrightarrow \text{ there exists an } n \in \mathbb{N} \text{ such that } s\theta_1 \cdots \theta_n = t\theta_1 \cdots \theta_n.$$

It is easy to see that '$\sim$' satisfies the equality and freeness axioms. The interpretation of the predicates is as follows

$$\mathcal{A}(A) := \begin{cases} \mathbf{t}, & \text{if there ex. an } n \in \mathbb{N} \text{ such that } A\theta_1 \cdots \theta_n \text{ succeeds with answer } \varepsilon; \\ \mathbf{f}, & \text{if for all } n \in \mathbb{N}\text{: } A\theta_1 \cdots \theta_n \notin \Gamma; \\ \mathbf{u} & \text{otherwise.} \end{cases}$$

The definition makes sense since it is not possible that an atom A is $\mathbf{t}$ and $\mathbf{f}$: Assume that $A\theta_1 \cdots \theta_n$ succeeds with answer ε. Fix $i \geq n$ such that A is A_i, and let $m = 2i + 1$. By Lemma 7 the atom $A\theta_1 \cdots \theta_m$ succeeds with answer ε too. If $\Gamma_m, A\theta_1 \cdots \theta_m$ would be finitely failed then by Lemma 14 Γ_m would be finitely failed. Since this is not the case $A\theta_1 \cdots \theta_m$ is in $\Gamma_{m+1} \subseteq \Gamma$.

Now we show that if $\mathcal{A}(A) = \mathbf{f}$ and $A\theta_1 \cdots \theta_p$ is closed then $A\theta_1 \cdots \theta_p$ is finitely failed: Let $\mathcal{A}(A) = \mathbf{f}$. Fix $i \geq p$ such that $A = A_i$ and let $m = 2i + 1$. The atom $A\theta_1 \cdots \theta_m$ is tried in step m. But $A\theta_1 \cdots \theta_m = A\theta_1 \cdots \theta_p$ and since $\mathcal{A}(A) = \mathbf{f}$ we have $A\theta_1 \cdots \theta_p \notin \Gamma$ and the goal $\Gamma_m, A\theta_1 \cdots \theta_p$ is finitely failed. By Lemma 12 the goal Γ_m or the atom $A\theta_1 \cdots \theta_p$ is finitely failed. Since Γ_m is not finitely failed $A\theta_1 \cdots \theta_p$ is finitely failed.

In a next step we prove that if $L \in \Gamma$ then $\mathcal{A}(L) \neq \mathbf{f}$: If L is positive then this is clear. Assume now that $L \in \Gamma$ and $L = \neg A$ and $\mathcal{A}(\neg A) = \mathbf{f}$. Then $\mathcal{A}(A) = \mathbf{t}$ and there exists an $n \in \mathbb{N}$ such that $A\theta_1 \cdots \theta_n$ succeeds with answer ε. There exists also an $i \in \mathbb{N}$ such that $\neg A$ is in Γ_i and $A = A'\theta_1 \cdots \theta_i$. If $m := \max(i, n)$ then $\neg A'\theta_1 \cdots \theta_m$ is in Γ_m and $A\theta_1 \cdots \theta_m = A'\theta_1 \cdots \theta_m$ and $A\theta_1 \cdots \theta_m$ succeeds with answer ε. But then Γ_m is finitely failed which is not the case. Therefore $\mathcal{A}(\neg A) \neq \mathbf{f}$.

Now we show that $\Phi_P\,\mathcal{A} \leq \mathcal{A}$. Let r be a predicate symbol with completed definition $\forall\vec{x}\,(r(\vec{x}\,) \leftrightarrow D_r[\vec{x}\,])$ where

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

We show in a first step that if $\mathcal{A}(D_r[\vec{s}\,]) = \mathbf{t}$ then $\mathcal{A}(r(\vec{s}\,)) = \mathbf{t}$. Assume that $\mathcal{A}(D_r[\vec{s}\,]) = \mathbf{t}$. Then there exist terms $\vec{r}$ and an i $(1 \leq i \leq m)$ such that

$$\mathcal{A}\Big(\bigwedge_{j=1}^{n} s_j = t_{i,j}[\vec{r}\,] \wedge \bigwedge_{j=1}^{k(i)} L_{i,j}[\vec{r}\,]\Big) = \mathbf{t}.$$

There exists a $p \in \mathbb{N}$ such that $s_j\theta_1\cdots\theta_p = t_{i,j}[\vec{r}\,]\theta_1\cdots\theta_p$ $(j = 1,\ldots,n)$ and if $L_{i,j}[\vec{r}\,]$ is positive then $L_{i,j}[\vec{r}\,]\theta_1\cdots\theta_p$ succeeds with answer ε. Since P is an ε-program, if $L_{i,j}[\vec{r}\,]\theta_1\cdots\theta_p = \neg A$ then A is closed or A is finitely failed. If A is closed then A is finitely failed. By Lemma 20 the head $r(\vec{s}\,)\theta_1\cdots\theta_p$ succeeds with answer ε and thus $\mathcal{A}(r(\vec{s}\,)) = \mathbf{t}$.

In a next step we show that if $\mathcal{A}(r(\vec{s}\,)) \neq \mathbf{f}$ then $\mathcal{A}(D_r[\vec{s}\,]) \neq \mathbf{f}$. If $\mathcal{A}(r(\vec{s}\,)) \neq \mathbf{f}$ then by definition of $\mathcal{A}$ there exists an $n \geq 0$ such that $r(\vec{s}\,)\theta_1\cdots\theta_n$ is in Γ. There exists an $i \geq 0$ such that $r(\vec{s}\,)\theta_1\cdots\theta_n$ is in Γ_i and of the form $r(\vec{s'}\,)\theta_1\cdots\theta_i$. The atom $r(\vec{s}\,)\theta_1\cdots\theta_n$ is selected later and hence there is a $j > i$ and a variant $r(\vec{t'}\,)$:- Π' of a clause $r(\vec{t}\,)$:- Π such that $r(\vec{t'}\,)\theta_j = r(\vec{s'}\,)\theta_1\cdots\theta_j$ and $\Pi'\theta_j \subseteq \Gamma_j \subseteq \Gamma$. Since $\vec{s} \sim \vec{s}\,\theta_1\cdots\theta_n = \vec{s'}\,\theta_1\cdots\theta_i \sim \vec{s'}\,\theta_1\cdots\theta_j = \vec{t'}\,\theta_j$ we get $\vec{s} \sim \vec{t'}\,\theta_j$ and $\mathcal{A}(D_r[\vec{s}\,]) \neq \mathbf{f}$.

Since $\Phi_P\,\mathcal{A} \leq \mathcal{A}$ by Lemma 1 there is a structure $\mathcal{M} \leq \mathcal{A}$ such that $\Phi_P\,\mathcal{M} = \mathcal{M}$. Then $\mathcal{M} \models_3 \mathit{comp}(P)$ and $\mathcal{M} \not\models_3 \neg\exists(L_1 \wedge \ldots \wedge L_q)$, since $\mathcal{A} \not\models_3 \neg\exists(L_1 \wedge \ldots \wedge L_q)$. If $\mathcal{M} \models_3 \forall(A)$ then $\mathcal{A} \models_3 \forall(A)$ and $\mathcal{A}(A) = \mathbf{t}$ and there exists an $n \in \mathbb{N}$ such that $A\theta_1\cdots\theta_n$ succeeds with answer ε. If A is closed and $\mathcal{M} \models_3 \neg A$ then $\mathcal{A} \models_3 \neg A$ and $\mathcal{A}(A) = \mathbf{f}$ and A is finitely failed. □

An immediate consequence of this theorem is the completeness of finite failure for ε-programs.

Theorem 23 (Completeness of ESLDNF-resolution for negative queries)
Let P be an ε-program.
If $\mathit{comp}(P) \models_3 \neg\exists(L_1 \wedge \ldots \wedge L_n)$ then the goal ?- $L_1,\ldots,L_n$ is finitely failed.

Note that there is no ε-condition on the goal ?- $L_1,\ldots,L_n$. In the case of success there is the following condition.

Theorem 24 (Completeness of ESLDNF-resolution for positive queries)
Let P be an ε-program and let ?- $L_1,\ldots,L_n$ be an ε-goal for P.

If $comp(P) \models_3 \forall(L_1 \wedge \ldots \wedge L_n)\sigma$ then the goal ?- $L_1, \ldots, L_n$ succeeds with answer θ including σ, i. e. there exists a substitution α such that $(L_1 \wedge \ldots \wedge L_n)\theta\alpha = (L_1 \wedge \ldots \wedge L_n)\sigma$.

Proof. Assume that P is an ε-program and that ?- $L_1, \ldots, L_n$ is an ε-goal for P and that $comp(P) \models_3 \forall(L_1 \wedge \ldots \wedge L_n)\sigma$. Let $1 \leq i \leq n$. We have $comp(P) \models_3 \forall(L_i\sigma)$ and therefore $comp(P) \models_3 \neg\exists\neg(L_i\sigma)$. If L_i is positive then by Theorem 23 $\neg L_i\sigma$ is finitely failed, and this implies that $L_i\sigma$ succeeds with answer ε. If L_i is negative and of the form $\neg A_i$ then by Theorem 23 $A_i\sigma$ is finitely failed. By Lemma 19, there is a computed answer θ of the goal ?- $L_1, \ldots, L_n$ and a substitution α such that $(L_1 \wedge \ldots \wedge L_n)\theta\alpha = (L_1 \wedge \ldots \wedge L_n)\sigma$. □

Remark 25 Allowed programs P and an allowed goals Γ have the following properties:

(a) If Γ succeeds with answer θ from P using ESLDNF-resolution then Γ succeeds with answer θ using SLDNF-resolution.

(b) If Γ is finitely failed in ESLDNF-resolution then Γ is finitely failed in SLDNF-resolution.

These statements follow from the facts that for an allowed program P an atom A is ground if it succeeds with answer ε in ELSDNF-resolution; and that in every ESLDNF-derivation which ends in a goal consisting only of negative literals this goal is ground.

5 An axiomatization of three-valued logic

Let P be any given program. We introduce a formal system LK(P) with the property that $comp(P) \models_3 A$ iff LK(P) $\vdash A$.

From now on we will consider only Kleene formulas of $\mathcal{L}$, i. e. formulas with $\neg$, $\wedge$, $\vee$, $\forall$ and $\exists$. The capital greek letters $\Gamma, \Delta, \Lambda, \Pi, \ldots$ will denote finite sequences of such formulas. An expression of the form $\Gamma \supset \Delta$ is called a *sequent*. We say that a sequent $\Gamma[\vec{x}] \supset \Delta[\vec{x}]$ is valid in a three-valued structure $\mathcal{A}$ (written $\mathcal{A} \models_3 \Gamma[\vec{x}] \supset \Delta[\vec{x}]$) if for all $\vec{a} \in |\mathcal{A}|$ there is a formula $A[\vec{a}]$ in $\Gamma[\vec{a}]$ which is false in $\mathcal{A}$ or there is a formula $B[\vec{a}]$ in $\Delta[\vec{a}]$ which is true in $\mathcal{A}$. Thus we have that

$$\mathcal{A} \models_3 A_1, \ldots, A_m \supset B_1, \ldots, B_n$$

is equivalent to

$$\mathcal{A} \models_3 \neg A_1 \vee \ldots \vee \neg A_m \vee B_1 \vee \ldots \vee B_n.$$

The main difference between Gentzen's sequent calculus LK for classical logic and LK(P) is that in LK(P) there are no axioms of the form $A \supset A$, since such sequents are in general not valid in three-valued structures. LK(P) consists of the exchange, weakening, contraction and cut rule, the left and right introduction rules for $\neg, \wedge, \vee, \forall, \exists$ and some initial equality sequents.

LK(P)

structural rules

$$\text{ex}\ \mathbf{l}\quad \frac{\Gamma, A, B, \Lambda \supset \Delta}{\Gamma, B, A, \Lambda \supset \Delta} \qquad \frac{\Gamma \supset \Delta, A, B, \Pi}{\Gamma \supset \Delta, B, A, \Pi}\quad \mathbf{r}\ \text{ex}$$

$$\text{w}\ \mathbf{l}\quad \frac{\Gamma \supset \Delta}{A, \Gamma \supset \Delta} \qquad \frac{\Gamma \supset \Delta}{\Gamma \supset \Delta, A}\quad \mathbf{r}\ \text{w}$$

$$\text{c}\ \mathbf{l}\quad \frac{A, A, \Gamma \supset \Delta}{A, \Gamma \supset \Delta} \qquad \frac{\Gamma \supset \Delta, A, A}{\Gamma \supset \Delta, A}\quad \mathbf{r}\ \text{c}$$

cut

$$\frac{\Gamma \supset \Delta, A \qquad A, \Gamma \supset \Delta}{\Gamma \supset \Delta}$$

logical rules

$$\neg\ \mathbf{l}\quad \frac{\Gamma \supset \Delta, A}{\neg A, \Gamma \supset \Delta} \qquad \frac{A, \Gamma \supset \Delta}{\Gamma \supset \Delta, \neg A}\quad \mathbf{r}\ \neg$$

$$\wedge\ \mathbf{l}\quad \frac{A, B, \Gamma \supset \Delta}{A \wedge B, \Gamma \supset \Delta} \qquad \frac{\Gamma \supset \Delta, A \qquad \Gamma \supset \Delta, B}{\Gamma \supset \Delta, A \wedge B}\quad \mathbf{r}\ \wedge$$

$$\vee\ \mathbf{l}\quad \frac{A, \Gamma \supset \Delta \qquad B, \Gamma \supset \Delta}{A \vee B, \Gamma \supset \Delta} \qquad \frac{\Gamma \supset \Delta, A, B}{\Gamma \supset \Delta, A \vee B}\quad \mathbf{r}\ \vee$$

quantifier rules

$$\forall\ \mathbf{l}\quad \frac{A[x/t], \Gamma \supset \Delta}{\forall x A, \Gamma \supset \Delta} \qquad \frac{\Gamma \supset \Delta, A}{\Gamma \supset \Delta, \forall y A[x/y]}{}^{*}\quad \mathbf{r}\ \forall$$

$$\exists\ \mathbf{l}\quad \frac{A, \Gamma \supset \Delta}{\exists y A[x/y], \Gamma \supset \Delta}{}^{*} \qquad \frac{\Gamma \supset \Delta, A[x/t]}{\Gamma \supset \Delta, \exists x A}\quad \mathbf{r}\ \exists$$

$*$ if $x \notin \text{var}(\Gamma, \Delta)$ and $x = y$ or $y \notin \text{var}(A)$.

program rules

(for a predicate r with completed definition $\forall \vec{x}\,(r(\vec{x}\,) \leftrightarrow D_r[\vec{x}\,])$)

$$\text{r}\ \mathbf{l}\quad \frac{D_r[\vec{s}\,], \Gamma \supset \Delta}{r(\vec{s}), \Gamma \supset \Delta} \qquad \frac{\Gamma \supset \Delta, D_r[\vec{s}\,]}{\Gamma \supset \Delta, r(\vec{s})}\quad \mathbf{r}\ \text{r}$$

equality and freeness sequents

$s = t \supset s = t$
$\supset t = t$
$t_1 = t_2 \supset t_2 = t_1$
$t_1 = t_2, t_2 = t_3 \supset t_1 = t_3$
$s_1 = t_1, \ldots, s_n = t_n \supset f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$
$f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n) \supset s_i = t_i$ (for $1 \leq i \leq n$)
$f(s_1, \ldots, s_n) = g(t_1, \ldots, t_m) \supset$ (if $f \neq g$)
$t[x/s] = s \supset$ (if $x \neq t$ and $x \in \mathrm{var}(t)$)

It is easy to see that the structural rules, the cut rule, the logical rules and the quantifier rules are valid in any three-valued structure $\mathcal{A}$, i. e. if $\mathcal{A} \models_3 S_i$ for all premisses S_i of a rule with conclusion S then $\mathcal{A} \models_3 S$.

The only axioms of LK(P) are the equality and freeness sequents and those are valid in any three-valued structure satisfying *CET*, since equality is always interpreted two-valued.

The proof of the soundness of LK(P) is now routine.

Theorem 26 (Soundness of LK(P)) If a sequent $\Gamma \supset \Delta$ is provable in LK(P) then $\Gamma \supset \Delta$ is valid in any three-valued model of *comp*(P).

The proof of the completeness of LK(P) goes similar to the proof of the completeness of Schütte valuations (see [11]). A version with the same terminology and notions that we use can be found in Girard's book ([6], Theorem 3.1.9, p. 164).

Theorem 27 (Completeness of LK(P)) If a sequent $\Gamma \supset \Delta$ is valid in any three-valued model of *comp*(P) then $\Gamma \supset \Delta$ is provable in LK(P) (with only atomic cuts).

Proof. Assume that $\Gamma_0 \supset \Delta_0$ is not provable in LK(P) with only atomic cuts. We will construct recursively a sequence $S_0, S_1, \ldots$ of non provable sequents from which one can extract a three-valued model of *comp*(P) in which $\Gamma_0 \supset \Delta_0$ is not valid.

Let $t_0, t_1, \ldots$ be an enumeration of all terms of $\mathcal{L}$ and $A_0, A_1, \ldots$ be an enumeration of all atomic formulas (inclusive equations) of $\mathcal{L}$. Let $S_0 := \Gamma_0 \supset \Delta_0$. Consider the following reduction rules.

(1) $\Gamma_0, \neg A, \Gamma_1 \supset \Delta \quad \rightsquigarrow \quad \Gamma_0, \Gamma_1 \supset \Delta, A$

(2) $\Gamma_0, A \wedge B, \Gamma_1 \supset \Delta \quad \rightsquigarrow \quad \Gamma_0, A, B, \Gamma_1 \supset \Delta$

(3) $\Gamma_0, A \vee B, \Gamma_1 \supset \Delta \quad \rightsquigarrow \quad \Gamma_0, A, \Gamma_1 \supset \Delta \quad$ or $\quad \Gamma_0, B, \Gamma_1 \supset \Delta$

(4) $\Gamma_0, \forall x A, \Gamma_1 \supset \Delta \quad \rightsquigarrow \quad \Gamma_0, A[x/t_n], \Gamma_1, \forall x A \supset \Delta$

(5) $\Gamma_0, \exists x A, \Gamma_1 \supset \Delta \quad \rightsquigarrow \quad \Gamma_0, A[x/y], \Gamma_1 \supset \Delta \quad (y \notin \mathrm{var}(\Gamma_0, \Gamma_1, \Delta))$

(6) $\Gamma_0, r(\vec{s}), \Gamma_1 \supset \Delta \quad \rightsquigarrow \quad \Gamma_0, \Gamma_1, D_r[\vec{s}\,] \supset \Delta$

(7) $\Gamma \supset \Delta_0, \neg A, \Delta_1 \quad \rightsquigarrow \quad A, \Gamma \supset \Delta_0, \Delta_1$

(8) $\Gamma \supset \Delta_0, A \wedge B, \Delta_1 \quad \rightsquigarrow \quad \Gamma \supset \Delta_0, A, \Delta_1 \quad \text{or} \quad \Gamma \supset \Delta_0, B, \Delta_1$

(9) $\Gamma \supset \Delta_0, A \vee B, \Delta_1 \quad \rightsquigarrow \quad \Gamma \supset \Delta_0, A, B, \Delta_1$

(10) $\Gamma \supset \Delta_0, \forall x A, \Delta_1 \quad \rightsquigarrow \quad \Gamma \supset \Delta_0, A[x/y], \Delta_1 \quad (y \notin \mathrm{var}(\Gamma, \Delta_0, \Delta_1))$

(11) $\Gamma \supset \Delta_0, \exists x A, \Delta_1 \quad \rightsquigarrow \quad \Gamma \supset \Delta_0, A[x/t_n], \Delta_1, \exists x A$

(12) $\Gamma \supset \Delta_0, r(\vec{s}), \Delta_1 \quad \rightsquigarrow \quad \Gamma \supset \Delta_0, \Delta_1, D_r[\vec{s}\,]$

(13) $\Gamma \supset \Delta \quad \rightsquigarrow \quad A_n, \Gamma \supset \Delta \quad \text{or} \quad \Gamma \supset \Delta, A_n$

We observe that if the left hand side of a reduction rule is not provable with atomic cuts only then also one of the sequents on the right hand side is not provable with atomic cuts only. Now we apply these reduction rules in a fair manner to S_0 and obtain a sequence $S_0, S_1, \ldots$ of non provable sequents.

Let Γ be the set of all formulas which occur on the left hand side of a sequent S_n and let Δ be the set of all formulas which occur on the right hand side of a sequent S_n. Then Γ and Δ have the following properties.

(1) if $\neg A \in \Gamma$ then $A \in \Delta$,

(2) if $A \wedge B \in \Gamma$ then $A \in \Gamma$ and $B \in \Gamma$,

(3) if $A \vee B \in \Gamma$ then $A \in \Gamma$ or $B \in \Gamma$,

(4) if $\forall x A \in \Gamma$ then for all terms t is $A[x/t] \in \Gamma$,

(5) if $\exists x A \in \Gamma$ then there exists a y such that $A[x/y] \in \Gamma$,

(6) if $r(\vec{s}) \in \Gamma$ then $D_r[\vec{s}\,] \in \Gamma$,

(7) if $\neg A \in \Delta$ then $A \in \Gamma$,

(8) if $A \wedge B \in \Delta$ then $A \in \Delta$ or $B \in \Delta$,

(9) if $A \vee B \in \Delta$ then $A \in \Delta$ and $B \in \Delta$,

(10) if $\forall x A \in \Delta$ then there exists a y such that $A[x/y] \in \Delta$,

(11) if $\exists x A \in \Delta$ then for all terms t is $A[x/t] \in \Delta$,

(12) if $r(\vec{s}) \in \Delta$ then $D_r[\vec{s}\,] \in \Delta$,

(13) for each atomic formula A is $A \in \Gamma$ or $A \in \Delta$,

(14) there is no equality or freeness sequent $\Lambda \supset \Pi$ such that $\Lambda \subseteq \Gamma$ and $\Pi \subseteq \Delta$.

A three-valued structure $\mathcal{A}$ is defined as follows. Let $|\mathcal{A}|$ be the set of all terms and let $f^{\mathcal{A}}(t_1, \ldots, t_n) := f(t_1, \ldots, t_n)$. Equality and predicates are defined by

$$\mathcal{A}(s = t) := \begin{cases} \mathbf{t}, & \text{if } (s = t) \in \Gamma; \\ \\ \mathbf{f}, & \text{otherwise.} \end{cases} \qquad \mathcal{A}(r(\vec{t})) := \begin{cases} \mathbf{t}, & \text{if } r(\vec{t}) \notin \Delta; \\ \mathbf{f}, & \text{if } r(\vec{t}) \notin \Gamma; \\ \mathbf{u}, & \text{otherwise.} \end{cases}$$

It is easy to see that $\mathcal{A}$ satisfies the equality axioms. We show only—as an example—the symmetry axiom. Suppose that $s = t \in \Gamma$. Then $t = s \in \Gamma$ or $t = s \in \Delta$. Since $s = t \supset t = s$ is an equality sequent it follows that $t = s \in \Gamma$. So if $\mathcal{A}(s = t) = \mathbf{t}$ then $\mathcal{A}(t = s) = \mathbf{t}$.

In a next step one proves by induction on the length of a formula A that, if A is in Γ then A is not false in $\mathcal{A}$ and if A is in Δ then A is not true in $\mathcal{A}$ under the canonical variable assignment where the value of a variable v_i is the element v_i.

From this it follows that the sequent $\Gamma_0 \supset \Delta_0$ is not valid in $\mathcal{A}$.

Now we claim that $\Phi_P \mathcal{A} \leq \mathcal{A}$. If $r(\vec{s})$ is not true in $\mathcal{A}$ then by definition of $\mathcal{A}$ the atom $r(\vec{s}) \in \Delta$ and therefore $D_r[\vec{s}\,] \in \Delta$ and $D_r[\vec{s}\,]$ is not true in $\mathcal{A}$. A similar observation shows that if $r(\vec{s})$ is not false in $\mathcal{A}$ then $D_r[\vec{s}\,]$ is not false in $\mathcal{A}$.

Now we apply Lemma 1 and obtain a structure $\mathcal{B} \leq \mathcal{A}$ with $\Phi_P \mathcal{B} = \mathcal{B}$. Hence $\mathcal{B} \models_3 \mathit{comp}(P)$ but $\mathcal{B} \not\models_3 \Gamma_0 \supset \Delta_0$. □

One can combine the previous two theorems to get the following one.

Theorem 28 If F is a Kleene formula then $\mathit{comp}(P) \models_3 F$ iff $\mathrm{LK}(P) \vdash F$.

From the soundness and completeness theorems we also obtain the following partial cut elimination theorem for $\mathrm{LK}(P)$.

Lemma 29 (Partial cut elimination) If the sequent $\Gamma \supset \Delta$ is provable in $\mathrm{LK}(P)$ then it is provable with atomic cuts only.

This lemma can also be proved in the usual constructive way. We do not want to go further into the details of the proof theory of $\mathrm{LK}(P)$ in this paper. There are only two technical remarks to mention.

Remarks 30 1° If there is no clause in P which defines the predicate r then the completed definition of r is $\forall \vec{x}\,(r(\vec{x}\,) \leftrightarrow \bot)$. We have two possibilities in this case. The first is to add the axiom $\bot, \Gamma \supset \Delta$ to $\mathrm{LK}(P)$ and to formulate the rules for r as before. The second possibility is without $\bot$. Then the rules for r can be formulated as a single axiom $r(\vec{s}), \Gamma \supset \Delta$.
2° The sequent $s_1 = t_1, \ldots, s_n = t_n, r(s_1, \ldots, s_n) \supset r(t_1, \ldots, t_n)$ is not valid in general, but the following substitution rule is admissible in $\mathrm{LK}(P)$. If $\Gamma[x/s] \supset \Delta[x/s]$ is provable then $s = t, \Gamma[x/t] \supset \Delta[x/t]$ is provable in $\mathrm{LK}(P)$.

In LK(P) only the direction from right to left of the completed definitions is used. If the right hand side is true (false) then the left hand side is true (false). Since in a proof of a sequent the program rules are only used finitely many times the following stronger form of the soundness theorem is not surprising. Consider a structure $\mathcal{A}$ satisfying *CET* with $\mathcal{A} \leq \Phi_P \mathcal{A}$. Then we define ${\Phi_P}^0 \mathcal{A} := \mathcal{A}$ and ${\Phi_P}^{n+1} \mathcal{A} := \Phi_P({\Phi_P}^n \mathcal{A})$. The structure ${\Phi_P}^n \mathcal{A}$ is in general neither a model of *comp*(P) nor of *comp*$^-$(P).

Theorem 31 (Strong soundness of LK(P)) If the sequent $\Gamma \supset \Delta$ is provable in LK(P) then there exists a natural number $n \in \mathbb{N}$ such that for every three-valued structure $\mathcal{A}$ satisfying *CET* and $\mathcal{A} \leq \Phi_P \mathcal{A}$ we have ${\Phi_P}^n \mathcal{A} \models_3 \Gamma \supset \Delta$.

Proof. Easy induction on the length of a proof of $\Gamma \supset \Delta$. One uses the fact that ${\Phi_P}^n \mathcal{A} \leq {\Phi_P}^{n+1} \mathcal{A}$. □

From this theorem we obtain as a corollary the following theorem of Kunen (one direction of Theorem 6.3 in [8]) which he had proved in a purely model theoretic way, using ultra powers.

Corollary 32 If *comp*(P) $\models_3 \forall(F)$ then there exists a natural number $n \in \mathbb{N}$ such that for every three-valued structure $\mathcal{A}$ satisfying *CET* and $\mathcal{A} \leq \Phi_P \mathcal{A}$ we have ${\Phi_P}^n \mathcal{A} \models_3 \forall(F)$.

There are two natural structures over the Herbrand universe of $\mathcal{L}$ with $\mathcal{A} \leq \Phi_P \mathcal{A}$. The first is the everywhere undefined structure, and the second is the success-failure structure in which a closed atom is true iff it succeeds and false iff it fails.

6 Conclusion

We have characterized in this paper a large class of programs for which ESLDNF-resolution is complete with respect to the three-valued completion. Then we have introduced LK(P) and we have proved that it is a sound and complete axiomatization of the three-valued completion of logic programs. But LK(P) is still not complete for negation as failure.

Consider the program P below with the completed definitions for p, q and r on the right hand side.

p :- $\neg q(X)$.
$q(c)$.
$q(X)$:- $\neg r(X)$.
$r(c)$.

$$\begin{array}{rcl} p & \leftrightarrow & \exists X \neg q(X) \\ q(X) & \leftrightarrow & X = c \vee \exists Y (X = Y \wedge \neg r(Y)) \\ r(X) & \leftrightarrow & X = c \end{array}$$

P is not an ε-program. The goal `?-` p does not fail but $\neg p$ is cut-free provable in LK(P):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{X = c \supset X = c}{r(X) \supset X = c}}{r(X) \supset X = c \ \vee \exists Y(X = Y \wedge \neg r(Y))}}{r(X) \supset q(X)}}{\neg q(X) \supset \neg r(X)} \qquad \supset X = X}}{\neg q(X) \supset X = X \wedge \neg r(X)}}{\neg q(X) \supset \exists Y(X = Y \wedge \neg r(Y))}}{\neg q(X) \supset X = c \vee \exists Y(X = Y \wedge \neg r(Y))}}{\neg q(X) \supset q(X)}}{\neg q(X), \neg q(X) \supset}}{\neg q(X) \supset}}{\exists X \neg q(X) \supset}}{p \supset}}{\supset \neg p}$$

One question is now, how can one restrict LK(P) such that the proof above is not possible. More generally: how can one restrict LK(P) such that it is sound and complete for ESLDNF-resolution? One possibility is to exclude axioms of the form $s = t \supset s = t$. But these axioms are needed for the soundness of ESLDNF-resolution.

Another possibility is to restrict the negation rules to have only equational side formulas. They are then written as

$$\dfrac{A, \Gamma \supset}{\Gamma \supset \neg A} \qquad\qquad \dfrac{\Gamma \supset A}{\neg A, \Gamma \supset}$$

with the condition that Γ consists only of equations. Then LK(P) is still sound for ESLDNF-resolution. But what is the semantics of this logic? Some kind of three-valued Kripke structures? Is it simpler than the procedural semantics of negation as failure?

Another possibility is to omit the contraction rules in LK(P) and to use a version of linear logic. But ESLDNF-resolution *has* contraction, since for example if the goal `?-` A, A, Γ is finitely failed then the goal `?-` A, Γ is finitely failed or if `?-` A, A, Γ succeeds with answer ε then `?-` A, Γ succeeds with answer ε.

It would be interesting to compare LK(P) or one of its subsystems with the provability relation $\vdash_{3I}$ of Shepherdson in [14]. The relation $\vdash_{3I}$ is three-valued sound and intuitionistically sound. Therefore it is weaker than LK(P).

Acknowledgment

I am grateful to John Shepherdson and an anonymous referee for helpful comments to the earlier version of the paper.

References

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

[2] L. Cavedon and J. W. Lloyd. A completeness theorem for SLDNF-resolution. *J. of Logic Programming*, 7(3):177–191, 1989.

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

[4] K. L. Clark, F. G. McCabe, and S. Gregory. IC-PROLOG language features. In K. L. Clark and S.-A. Tärnlund, editors, *Logic Programming*, pages 253–266. Academic Press, London, 1982.

[5] M. Fitting. A Kripke-Kleene semantics for logic programs. *J. of Logic Programming*, 2:295–312, 1985.

[6] J.-Y. Girard. *Proof Theory and Logical Complexity*. Bibliopolis, Napoli, 1987.

[7] J. Jaffar, J.-L. Lassez, and J. W. Lloyd. Completeness of the negation as failure rule. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence IJCAI-83*, pages 500–506, Karlsruhe, 1983.

[8] K. Kunen. Negation in logic programming. *J. of Logic Programming*, 4(4):289–308, 1987.

[9] K. Kunen. Signed data dependencies in logic programs. *J. of Logic Programming*, 7(3):231–245, 1989.

[10] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, second edition, 1987.

[11] K. Schütte. *Proof Theory*. Springer-Verlag, Berlin, 1977.

[12] J. C. Shepherdson. Negation as failure: A comparison of Clark's completed data base and Reiter's closed world assumption. *J. of Logic Programming*, 1(1):51–79, 1984.

[13] J. C. Shepherdson. Negation as failure II. *J. of Logic Programming*, 2(3):185–202, 1985.

[14] J. C. Shepherdson. A sound and complete semantics for a version of negation as failure. *Theoretical Computer Science*, 65(3):343–371, 1989.

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

[16] A. Van Gelder. Negation as failure using tight derivations for general logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 149–176. Morgan Kaufmann, Los Altos, 1987.