ICOT Technical Report: TR-175

TR-175

Soundness and Completeness of Extended Execution for Proving Properties of Prolog Programs

by
Tadashi Kanamori
(Mitsubishi Electric Corp.)

May, 1986

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

Soundness and Completeness of Extended Execution for Proving Properties of Prolog Programs

Tadashi KANAMORI

Mitsubishi Electric Corporation
Central Research Laboratory
Tsukaguchi-Honmachi 8-1-1
Amagasaki, Hyogo, JAPAN 661

Abstract

Soundness and completeness of extended execution devised for proving properties of Prolog programs are presented. Extended execution is a generalization of the execution of Prolog. It is applied to a class of first order formulas, which is called S-formulas and includes both universal formulas and usual execution goals. It is proved that an S-formula S is provable by extended execution if and only if S is a logical consequence of the completion of program P in the sense of Clark. This result is not only a generalization of the completeness of the SLD-resolution but also that of the "Negation as Failure" rule.

Keywords : Program Verification, Semantics, Prolog, Negation as Failure.

Contents

1. Introduction

The intimacy of Prolog to first order logic is expected to bring advantages to various manipulation of Prolog programs. We have developed an extension of Prolog execution for a class of first order formulas, which is called S-formulas and includes both universal formulas and usual execution goals [9]. It is actually used for first order inference in our verification system for proving properties of Prolog programs with computational induction [10] to show that a specification in S-formula is valid in the minimum Herbrand model of a given program.

In this paper, we prove soundness and completeness of our extended execution. That is, an S-formula S is provable by extended execution [†] if and only if S is a logical consequence of the completion P^* of a given program P in the sense of Clark. This result is not only a generalization of the completeness of the SLD-resolution (van Emden and Kowalski [5], Apt and van Emden [1]) but also that of the "Negation as Failure" rule (Jaffar et al [7]).

After preparing several notions in Section 2 and summarizing extended execution in Section 3, we prove its soundness in Section 4 first. Then in Section 5, after introducing some class of extended execution, we prove the completeness theorem by showing existence of a model of $P^* \bigcup \{\neg S\}$ for any universal formula S not provable by the class of extended execution. Lastly in Section 6, we discuss the implications of the soundness theorem and the completeness theorem.

2. Preliminaries

In the following, we assume familiarity with the basic terminology of first order logic such as term, atom (atomic formula), positive and negative literals, formula, substitution, most general unifier (m.g.u.) and so on. We also assume knowledge of the semantics of Prolog such as completion, minimum Herbrand model and transformation T of Herbrand interpretations (see [5],[1],[4],[13],[3],[7]). We follow the syntax of DEC-10 Prolog [17]. As syntactic variables, we use X, Y, Z for variables, s, t for terms, A, B for atoms, L for literals, C for definite clauses and $\mathcal{F}, \mathcal{G}, \mathcal{H}$ for formulas, possibly with primes and subscripts. In addition, we use $\sigma, \tau, \mu, \nu, \theta$ for substitutions, $\mathcal{F}_{\mathcal{G}}(\mathcal{H})$ for a replacement of *all* occurences of a formula $\mathcal{G}$ in a formula $\mathcal{F}$ with $\mathcal{H}$ and $\mathcal{F}_{\mathcal{G}}[\mathcal{H}]$ for a replacement of an occurence of a formula $\mathcal{G}$ in a formula $\mathcal{F}$ with $\mathcal{H}$.

2.1. Polarity of Subformulas

We generalize the distinction of positive and negative goals. The *positive* and *negative subformulas* of a formula $\mathcal{F}$ are defined as follows (see Manna and Waldinger [15], Murray [16], Prawitz [18]).

(a) $\mathcal{F}$ is a positive subformula of $\mathcal{F}$.
(b) When $\neg\mathcal{G}$ is a positive (negative) subformula of $\mathcal{F}$, then $\mathcal{G}$ is a negative (positive) subformula of $\mathcal{F}$.
(c) When $\mathcal{G} \wedge \mathcal{H}$ or $\mathcal{G} \vee \mathcal{H}$ is a positive (negative) subformula of $\mathcal{F}$, then $\mathcal{G}$ and $\mathcal{H}$ are positive (negative) subformulas of $\mathcal{F}$.
(d) When $\mathcal{G} \supset \mathcal{H}$ is a positive (negative) subformula of $\mathcal{F}$, then $\mathcal{G}$ is a negative (positive)

[†] In our previous papers [9],[10], we used the term "S-formulas" for a larger class of first order formulas and conjectured the completeness of extended execution for it. Though the result of this paper shows that the conjecture is wrong and the completeness holds only for a slightly smaller class of first order formulas, it is still large enough in practice. (See Example 6.2.)

subformula of $\mathcal{F}$ and $\mathcal{H}$ is a positive (negative) subformula of $\mathcal{F}$.

(e) When $\forall X \mathcal{G}$ or $\exists X \mathcal{G}$ is a positive (negative) subformula of $\mathcal{F}$, then $\mathcal{G}_X(t)$ is a positive (negative) subformula of $\mathcal{F}$.

Example 2.1. Let $\mathcal{F}$ be

∀ B,U,A$_1$,V,A (reverse(B,[U|A$_1$])∧append(A$_1$,[V],A) ⊃ ∃ A$_2$ (reverse(B,A$_2$)∧append(A$_2$,[V],[U|A])))

Then $\exists A_2(reverse(B, A_2) \wedge append(A_2, [V], [U|A]))$ is a positive subformula of $\mathcal{F}$, while $reverse(B, [U|A_1])$ is a negative subformula of $\mathcal{F}$.

2.2. S-formulas and Goal Formulas

We introduce a class of first order formulas, which are called S-formulas and includes the class of all universal formulas. S-formulas are represented by goal formulas.

Let $\mathcal{F}$ be a closed first order formula. When $\forall X \mathcal{G}$ is a positive subformula or $\exists X \mathcal{G}$ is a negative subformula of $\mathcal{F}$, X is called a *free variable* of $\mathcal{F}$. When $\forall Y \mathcal{H}$ is a negative subformula or $\exists Y \mathcal{H}$ is a positive subformula of $\mathcal{F}$, Y is called an *undecided variable* of $\mathcal{F}$. In other words, free variables are variables quantified universally and undecided variables are those quantified existentially when $\mathcal{F}$ is converted to prenex normal form.

Example 2.2.1. Let $\mathcal{F}$ be

∀ B,U,A$_1$,V,A (reverse(B,[U|A$_1$])∧append(A$_1$,[V],A) ⊃ ∃ A$_2$ (reverse(B,A$_2$)∧append(A$_2$,[V],[U|A])))

Then B, U, A_1, V and A are all free variables, while A_2 is an undecided variable.

A closed first order formula S is called a *specification formula* (or *S-formula* for short) when

(S_1) no free variable in S is quantified in the scope of quantification of an undecided variable in S and

(S_2) each undecided variable appears only in some positive conjunction of atoms $A_1 \wedge A_2 \wedge \cdots \wedge A_k$ $(k \geq 1)$.

In other words, S-formulas are formulas convertible to prenex normal form $\forall X_1, X_2, \ldots, X_n$ $\exists Y_1, Y_2, \ldots, Y_m \mathcal{F}$ and each Y_i appears in some positive conjunction of $\mathcal{F}$. (Hence none of $Y_1, Y_2, \ldots, Y_m$ appears among the negative atoms of $\mathcal{F}$.) Note that S-formulas include usual execution goals $\exists Y_1, Y_2, \ldots, Y_m$ $(A_1 \wedge A_2 \wedge \cdots \wedge A_k)$.

Example 2.2.2. Let S be

∀ B,U,A$_1$,V,A (reverse(B,[U|A$_1$])∧append(A$_1$,[V],A) ⊃ ∃ A$_2$ (reverse(B,A$_2$)∧append(A$_2$,[V],[U|A])))

Then S is an S-formula, because free variables B, U, A_1, V and A are quantified outside $\exists A_2$, and A_2 appears only in the positive conjunction $reverse(B, A_2) \wedge append(A_2, [V], [U|A])$. An execution goal

∃ C append([1,2],[3],C)

is also an S-formula.

A formula G obtained from an S-formula S by leaving free variable X as it is, replacing undecided variable Y with $?Y$ and deleting all quantifications is called a *goal formula* of S. Note that S can be uniquely restorable from G. In the following, we use goal formulas in stead of original S-formulas. Goal formulas are denoted by F, G, H, I, J.

Example 2.2.3. An S-formula

∀ B,U,A$_1$,V,A (reverse(B,[U|A$_1$])∧append(A$_1$,[V],A) ⊃ ∃ A$_2$ (reverse(B,A$_2$)∧append(A$_2$,[V],[U|A])))

is represented by a goal formula
reverse(B,[U|A_1])∧append(A_1,[V],A) ⊃reverse(B,?A_2)∧append(?A_2,[V],[U|A]).
An execution goal
∃ C append([1,2],[3],C)
is represented by a goal formula
append([1,2],[3],?C).

An S-formula S is called a *universal formula* (or *quantifier-free formula*) when S has no undecided variable. We use universal formulas and their goal formulas indistinguishably.

Example 2.2.4. A universal formula
∀ A,B (reverse(A,B)⊃reverse(B,A))
is an S-formula and represented by a goal formula
reverse(A,B)⊃reverse(B,A).
A goal formula
reverse(B,[U|A_1])∧append(A_1,[V],A) ⊃p(B,V,U,A)
corresponds to a universal formula
∀ B,U,A_1,V,A (reverse(B,[U|A_1])∧append(A_1,[V],A) ⊃p(B,V,U,A)).
This universal formula is obtained from
reverse(B,[U|A_1])∧append(A_1,[V],A) ⊃reverse(B,?A_2)∧append(?A_2,[V],[U|A])
in the example above by introducing a definite clause
p(B,V,U,A) :- reverse(B,A_2)∧append(A_2,[V],[U|A]).
and replacing the conjunction $reverse(B, ?A_2) \wedge append(?A_2, [V], [U|A])$ with $p(B, V, U, A)$. This conversion of an S-formula to a universal formula on the surface is used in the proof of the completeness theorem in 5.5.

2.3. Manipulation of Goal Formulas

Lastly we introduce two manipulations of goal formulas.

One is an application of a class of substitutions. A substitution σ for G is called a *deciding substitution* when σ instantiates no free variable in G. We use σ and μ for deciding substitutions, while τ, ν and θ for instantiation of free variables.

Example 2.3.1. Let S be
∀ A,B,U ((list(A) ⊃∃C append(A,B,C)) ⊃(list(A) ⊃∃C append([U|A],B,C)))
Then the goal formula of S is
(list(A) ⊃append(A,B,C)) ⊃(list(A) ⊃append([U|A],B,?C))
The most general common instance of $append([U|A], B, ?C)$ and the head of the second definite clause for *append* is obtained by a deciding substitution $\sigma = <?C \Leftarrow [U|?C'] >$. $\sigma(G)$ represents an S-formula
∀ A,B,U ((list(A) ⊃∃C append(A,B,C)) ⊃(list(A) ⊃∃C′ append([U|A],B,[U|C′])))

Another manipulation is a *reduction* of goal formulas with the logical constants *true* and *false*. The *reduced form* of a goal formula G, denoted by $G\downarrow$, is the normal form in the reduction system defined as follows.

$\neg true \rightarrow false$, $\neg false \rightarrow true$,
$true \wedge G \rightarrow G$, $false \wedge G \rightarrow false$,
$G \wedge true \rightarrow G$, $G \wedge false \rightarrow false$,
$true \vee G \rightarrow true$, $false \vee G \rightarrow G$,
$G \vee true \rightarrow true$, $G \vee false \rightarrow G$,
$true \supset G \rightarrow G$, $false \supset G \rightarrow true$,
$G \supset true \rightarrow true$, $G \supset false \rightarrow \neg G$.

Example 2.3.2. Let G_1 and G_2 be
(true ⊃reverse(N,L)) ⊃(true ∧append(N,[X],M) ⊃reverse(M,[X|L]))
(false ⊃reverse(N,L)) ⊃(false ∧append(N,[X],M) ⊃reverse(M,[X|L])).
Then $G_1 \downarrow$ is $reverse(N,L) \supset (append(N,[X],M) \supset reverse(M,[X|L]))$ and $G_2 \downarrow$ is *true*.

Lemma 2.3. Let H be a goal formula and A be an atom in H.
(a) When $H_A[true] \downarrow$ is *true*, A is a positive subformula of H.
(b) When $H_A[true] \downarrow$ is *false*, A is a negative subformula of H.
(c) When $H_A[false] \downarrow$ is *false*, A is a positive subformula of H.
(d) When $H_A[false] \downarrow$ is *true*, A is a negative subformula of H.

Proof. By induction on the structure of H (Murray [16] p.72). We prove (a) and (b). Other cases (c) and (d) are proved similarly. Supose $H_A[true] \downarrow$ is either *true* or *false*.

When H is an atom, it must be A and $H_A[true] \downarrow$ is *true*. Hence, the lemma is trivial.

When H is of the form $\neg H'$, $H'_A[true] \downarrow$ is *false* if $H_A[true] \downarrow$ is *true*. By the induction hypothesis, A is a negative subformula of H', hence positive subformula of H. The case $H_A[true] \downarrow$ is *false* is proved similarly.

When H is of the form $H_1 \vee H_2$, $H_A[true] \downarrow$ and $H_{1A}[true] \downarrow$ are *true* if A is in H_1. By the induction hypothesis, A is a positive subformula of H_1, hence positive subformula of H. The case A is in H_2 is proved similarly.

When H is of the form $H_1 \wedge H_2$, $H_A[true] \downarrow$ and $H_{1A}[true] \downarrow$ are *false* if A is in H_1. By the induction hypothesis, A is a negative subformula of H_1, hence negative subformula of H. The case A is in H_2 is proved similarly.

When H is of the form $H_1 \supset H_2$, $H_A[true] \downarrow$ is *true* and $H_{1A}[true] \downarrow$ is *false* if A is in H_1. By the induction hypothesis, A is a negative subformula of H_1, hence a positive subformula of H. Similarly, $H_A[true] \downarrow$ is *true* and $H_{2A}[true] \downarrow$ is *true* if A is in H_2. By the induction hypothesis, A is a positive subformula of H_2, hence a positive subformula of H.

3. An Extension of Execution

Our extension of execution consists of the following seven inference rules. (See the following explanation for their notations.) Each rule says that the subgoals in S-formulas above the line are generated from the goal in S-formulas below the line. We assume that variables in the S-formulas are renamed appropriately so that there occurs no conflict of variable names.

∧-deletion
$$\frac{G_H[H_1] \quad G_H[H_2] \quad \cdots \quad G_H[H_k]}{G_+[H_1 \wedge H_2 \wedge \cdots \wedge H_k]}$$

∨-deletion
$$\frac{G_H[H_1] \quad G_H[H_2] \quad \cdots \quad G_H[H_k]}{G_-[H_1 \vee H_2 \vee \cdots \vee H_k]}$$

$$\supset\text{-deletion} \quad \frac{G_H[\neg H_1] \qquad G_H[H_2]}{G_-[H_1 \supset H_2]}$$

$$\text{DCI} \quad \frac{\sigma_1(G_A[\wedge_{j=1}^{m_1} B_{1j}]) \downarrow}{G_+[A]} \quad \frac{\sigma_2(G_A[\wedge_{j=1}^{m_2} B_{2j}]) \downarrow}{G_+[A]} \quad \cdots \quad \frac{\sigma_k(G_A[\wedge_{j=1}^{m_k} B_{kj}]) \downarrow}{G_+[A]}$$

$$\text{NFI} \quad \frac{G_A[false] \downarrow \quad \tau_1(G_A[\wedge_{j=1}^{m_1} B_{1j}]) \downarrow \quad \cdots \quad \tau_k(G_A[\wedge_{j=1}^{m_k} B_{kj}]) \downarrow}{G_-[A]}$$

$$\text{simplification} \quad \frac{\sigma(G)_A(true) \downarrow \qquad \sigma(G)_A(false) \downarrow}{G}$$

3.1. Case Splitting

$\wedge$-Deletion

Let G be a goal formula. When H is a positive subformula of the form $H_1 \wedge H_2 \wedge \cdots \wedge H_k$ ($k > 1$) and each undecided variable $?X$ appearing in H_i appears only in H_i ($1 \leq i \leq k$), we generate k new AND-goals $G_H[H_1]$,$G_H[H_2]$,...,$G_H[H_k]$, where undecided variables in G and not in H are renamed and not shared between $G_H[H_i]$ and $G_H[H_j]$ ($i \neq j$).

Example 3.1.1. Let S be

∀ A,B,C (append(A,B,C)∧list(C) ⊃∃D reverse(A,D)∧∃E reverse(B,E)).

Then the goal formula of S is

append(A,B,C)∧list(C) ⊃reverse(A,?D)∧reverse(B,?E).

By applying ∧-deletion, we have 2 AND-goals

append(A,B,C)∧list(C) ⊃reverse(A,?D).

append(A,B,C)∧list(C) ⊃reverse(B,?E).

$\vee$-Deletion

Let G be a goal formula. When H is a negative subformula of the form $H_1 \vee H_2 \vee \cdots \vee H_k$ ($k > 1$) and each undecided variable $?X$ appearing in H_i appears only in H_i ($1 \leq i \leq k$), we generate k new AND-goals $G_H[H_1]$,$G_H[H_2]$,...,$G_H[H_k]$, where undecided variables in G and not in H are renamed and not shared between $G_H[H_i]$ and $G_H[H_j]$ ($i \neq j$).

Example 3.1.2. Let S be of the form

∀ S,T ((S=T ∨S<T ∨T<S) ⊃(···)).

Then the goal formula of S is

(S=T ∨S<T ∨T<S) ⊃(···).

By applying ∨-deletion, we have 3 AND-goals

S=T ⊃(···).

S<T ⊃(···).

T<S ⊃(···).

$\supset$-Deletion

Let G be a goal formula. When H is a negative subformula of the form $H_1 \supset H_2$ and each undecided variable $?X$ appearing in H_i appears only in H_i ($1 \leq i \leq 2$), we generate two new AND-goals $G_H[\neg H_1]$ and $G_H[H_2]$, where undecided variables in G and not in H are renamed and not shared between $G_H[H_i]$ and $G_H[H_j]$ ($i \neq j$).

Example 3.1.3. Let S be

$\forall$ U,B,C,D_1,D_2 ((append(B,C,D_2) $\supset$$D_1$=$D_2$) $\supset$(append(B,C,D_2) $\supset$[U|D_1]=[U|D_2])).

Then the goal formula of S is

(append(B,C,D_2) $\supset$$D_1$=$D_2$) $\supset$(append(B,C,D_2) $\supset$[U|D_1]=[U|D_2]).

By applying $\supset$-deletion, we have 2 AND-goals

$\neg$ append(B,C,D_2) $\supset$(append(B,C,D_2) $\supset$[U|D_1]=[U|D_2]).

D_1=D_2 $\supset$(append(B,C,D_2) $\supset$[U|D_1]=[U|D_2]).

3.2. Definite Clause Inference

We generalize the execution of positive goals using polarity.

Definite Clause Inference (DCI)

Let A be a positive atom in a goal formula G and "A_0 :- $A_1, A_2, \ldots, A_m$" be any definite clause in P. When A is unifiable with A_0 by a deciding m.g.u. σ, we generate a new OR-goal $\sigma(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow$. ($A_1 \wedge A_2 \wedge \cdots \wedge A_m$ is *true* when $m = 0$.) All new variables introduced are treated as fresh undecided variables.

Example 3.2.1. Let S be

$\forall$ A,B,U ((reverse(A,B)$\supset$reverse(B,A))$\supset$(reverse(A,[U|B])$\supset$reverse([U|B],A))).

Then the goal formula of S is

(reverse(A,B) $\supset$reverse(B,A)) $\supset$(reverse(A,[U|B]) $\supset$reverse([U|B],A))

We can apply DCI to $reverse([U|B], A)$ and it is replaced with $reverse(A, ?C) \wedge append(?C, [U], B)$. Note that the variable introduced from the body of the definite clause is treated as an undecided variable $?C$.

Example 3.2.2. When S is an existential formula of the form $\exists Y_1 Y_2 \cdots Y_m(A_1 \wedge A_2 \wedge \cdots \wedge A_k)$, i.e., of the form of usual execution goals, the goal formula of S is $?\text{-}A_1, A_2, \ldots, A_k$. (The juxtaposition delimited by "," denotes conjunction and $?\text{-}G$ denotes the goal formula obtained by replacing every variable Y in G with $?Y$.) Then usual execution is applied to $?\text{-}A_1, A_2, \ldots, A_k$.

3.3. "Negation as Failure" Inference

We also generalize the execution of negative goals using polarity.

"Negation as Failure" Inference (NFI)

Let A be a negative atom in a goal formula G. We generate new AND-goals $G_A[false] \downarrow$ and $\tau(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow$ for every definite clause "A_0 :- $A_1, A_2, \ldots, A_m$" in P, whose head A_0 is unifiable with A, say by an m.g.u. τ. ($A_1 \wedge A_2 \wedge \cdots \wedge A_m$ is *true* when $m = 0$.) All new variables introduced are treated as fresh free variables. (Note that A always includes only free variables and τ may be any m.g.u. without restriction.)

Example 3.3.1. Let S be

$\forall$ A,B,U ((reverse(A,B)$\supset$reverse(B,A))$\supset$(reverse([U|A],B)$\supset$reverse(B,[U|A]))).

Then the goal formula of S is

(reverse(A,B) $\supset$reverse(B,A)) $\supset$(reverse([U|A],B) $\supset$reverse(B,[U|A]))

We can apply NFI to $reverse([U|A], B)$. The first goal is trivially *true*. In the second goal, the atom is replaced with $reverse(A, C) \wedge append(C, [U], B)$. Note that the variable introduced from the body of the definite clause is treated as a free variable C.

Example 3.3.2. Let S be an S-formula of the form $\neg A$, where A is a ground atom. Suppose there exist k definite clauses whose heads are unifiable with A by m.g.u.'s $\tau_1, \tau_2, \ldots, \tau_k$. When NFI is applied to A, we have $k+1$ AND-goals

$(\neg false) \downarrow$,
$\tau_1(\neg(A_{11} \wedge A_{12} \wedge \cdots \wedge A_{1m_1})) \downarrow$,
$\tau_2(\neg(A_{21} \wedge A_{22} \wedge \cdots \wedge A_{2m_2})) \downarrow$,
$\vdots$
$\tau_k(\neg(A_{k1} \wedge A_{k2} \wedge \cdots \wedge A_{km_k})) \downarrow$.

The first goal formula is trivially *true*. Other goal formulas are of the form $\forall X_1, X_2, \ldots, X_n$ $\neg(A_1 \wedge A_2 \wedge \cdots \wedge A_m)$, because internal variables introduced from the bodies of the definite clauses are free variables in the generated goal formulas. We can continue applying NFI by selecting atoms in each goal formula. When a selected atom has no unifiable head, the only goal formula generated is the first one, which is always *true*. When all goal formulas are reduced to *true*, $\neg A$ is proved. This is exactly the "Negation as Failure" rule in the usual sense (see Clark [4]).

3.4. Simplification

Simplification

Let G be a goal formula. When $A_1, A_2, \ldots, A_m$ are positive atoms and $A_{m+1}, A_{m+2}, \ldots, A_n$ are negative atoms unifiable to A by a deciding m.g.u. σ, we generate two new AND-goals $\sigma(G)_A(true) \downarrow$ and $\sigma(G)_A(false) \downarrow$.

Example 3.4.1. Let G be a goal formula

(add(X,Y,Z) ⊃add(Y,X,Z)) ⊃(add(X,Y,Z) ⊃add(Y,s(X),s(Z)))

of an S-formula

∀ X,Y,Z ((add(X,Y,Z) ⊃add(Y,X,Z)) ⊃(add(X,Y,Z) ⊃add(Y,s(X),s(Z)))).

Because $\sigma = <>$ is a deciding substitution and unifies the positive atom $add(X, Y, Z)$ and the negative atom $add(X, Y, Z)$, we generate new AND-goals

(true ⊃add(Y,X,Z)) ⊃(true ⊃add(Y,s(X),s(Z))) ↓,
(false ⊃add(Y,X,Z)) ⊃(false ⊃add(Y,s(X),s(Z))) ↓,

i.e., $add(Y, X, Z) \supset add(Y, s(X), s(Z))$ and *true*. This inference corresponds to generating

(Y+X)+1=Y+(X+1)

from

X+Y=Y+X ⊃(X+Y)+1=Y+(X+1)

in functional programs, i.e., using the equation $X+Y = Y+X$ in the premise and throwing it away. This is called *cross-fertilization* in Boyer Moore Theorem Prover (BMTP) [2].

Example 3.4.2. Let G be a goal formula

(reverse(A,B) ⊃reverse(B,A)) ⊃(reverse(A,C) ∧append(C,[U],B) ⊃reverse(B,[U|A]))

of an S-formula

∀ A,B,C,U ((reverse(A,C) ⊃reverse(C,A)) ⊃((reverse(A,C)∧append(C,[U],B))⊃reverse(B,[U|A]))).

Because $\sigma = <>$ is a deciding substitution and unifies the positive atom $reverse(A, C)$ and the negative atom $reverse(A, C)$, we generate new AND-goals

(true ⊃reverse(C,A)) ⊃(true ∧append(C,[U],B) ⊃reverse(B,[U|A])) ↓,
(false ⊃reverse(C,A)) ⊃(false ∧append(C,[U],B) ⊃reverse(B,[U|A])) ↓,

i.e., $reverse(C, A) \supset (append(C, [U], B) \supset reverse(B, [U|A]))$ and *true*. This inference corresponds to generating

reverse(C)=A ⊃reverse(append(C,[U]))=[U|A]

from

reverse(reverse(A))=A ⊃reverse(append(reverse(A),[U]))=[U|A]
in functional programs, i.e., replacement of the special term *reverse*(*A*) with a variable *C*. This is called *generalisation* in BMTP [2].

Remark. The class of S-formulas defined in 2.2 is the minimum class of formulas that includes universal formulas and is closed under extended execution.

4. Soundness of Extended Execution

In this section, we summarize the completion in the sense of Clark first. Then we define conjunctive normal form and disjunctive normal form of goal formulas similarly to usual first order formulas and prepare several lemmas about them. Lastly, we prove the soundness of our extended execution using these lemmas.

4.1. Completion of Prolog Programs

A Prolog program P is a finite set of definite clauses. P states only the "if" part of programmer's intentions. By complementing the "only if" part, we can strengthen P to a theory P^* called *completion* of P. P^* is obtained as follows (Clark [4]).

(a) Transform each definite clause as follows.

$$p(t_1, t_2, \ldots, t_n) \text{ :- } B_1, B_2, \ldots, B_m$$
$$\Downarrow$$
$$X_1 = t_1 \wedge X_2 = t_2 \wedge \cdots \wedge X_n = t_n \wedge B_1 \wedge B_2 \wedge \cdots \wedge B_m \supset p(X_1, X_2, \ldots, X_n)$$
$$\Downarrow$$
$$\exists Z_1 Z_2 \cdots Z_u W_1 W_2 \cdots W_v (X_1 = t_1 \wedge X_2 = t_2 \wedge \cdots \wedge X_n = t_n \wedge B_1 \wedge B_2 \wedge \cdots \wedge B_m) \supset p(X_1, X_2, \ldots, X_n)$$

where $X_1, X_2, \ldots, X_n$ are fresh variables, $Z_1, Z_2, \ldots, Z_u$ are variables appearing in the head $p(t_1, t_2, \ldots, t_n)$ (called *head variables*) and $W_1, W_2, \ldots, W_v$ are variables appearing only in the body $B_1, B_2, \ldots, B_m$ (called *internal variables*).

(b) If the definite clauses with the head predicate p are transformed to

$H_1 \supset p(X_1, X_2, \ldots, X_n)$,
$H_2 \supset p(X_1, X_2, \ldots, X_n)$,
$\vdots$, $H_k \supset p(X_1, X_2, \ldots, X_n)$,

define p^* using the disjunction of the right-hand sides as follows.

$$p^* : \forall X_1, X_2, \ldots, X_n \; [p(X_1, X_2, \ldots, X_n) \equiv H_1 \vee H_2 \vee \cdots \vee H_k].$$

When a predicate symbol p in P has no corresponding definite clause, we stipulate that p^* is $\forall X_1, X_2, \ldots, X_n \; \neg p(X_1, X_2, \ldots, X_n)$. (Such predicates are said to be *undefined*.)

(c) Let P^* be a first order theory whose axiom is the set of all p^* and the following set of axioms EQ for equality.

$X = X$
$X = Y \supset Y = X$
$X = Y \wedge Y = Z \supset X = Z$
$X_1 = Y_1 \wedge X_2 = Y_2 \wedge \cdots \wedge X_n = Y_n \supset f(X_1, X_2, \ldots, X_n) = f(Y_1, Y_2, \ldots, Y_n)$
$X_1 = Y_1 \wedge X_2 = Y_2 \wedge \cdots \wedge X_n = Y_n \supset (p(X_1, X_2, \ldots, X_n) \supset p(Y_1, Y_2, \ldots, Y_n))$
$c \neq c'$ for any pair of distinct constant symbol c and c'
$f(X_1, X_2, \ldots, X_n) \neq g(Y_1, Y_2, \ldots, Y_m)$ for any pair of distinct function symbols f and g

$f(X_1, X_2, \ldots, X_n) = f(Y_1, Y_2, \ldots, Y_n) \supset X_1 = Y_1 \wedge X_2 = Y_2 \wedge \cdots \wedge X_n = Y_n$
$f(X_1, X_2, \ldots, X_n) \neq c$ for any function symbol f and constant symbol c
$t \neq X$ for any non-variable term t containing X

Example 4.1.1. Let *append* be a predicate defined by

```
append([ ],K,K).
append([X|L],M,[X|N]) :- append(L,M,N).
```

Then *append** is

$\forall$ A,B,C (append(A,B,C) $\equiv$
$\exists$K (A=[ ]$\wedge$B=K$\wedge$C=K) $\vee\exists$X,L,M,N (A=[X|L]$\wedge$B=M$\wedge$C=[X|N]$\wedge$append(L,M,N))).

Note that p^* can be expressed in a simpler form when the heads are not unifiable. *append** is simplified to

$\forall$ K append([ ],K,K) $\wedge\forall$X,L,M,N (append([X|L],M,[X|N])$\equiv$ append(L,M,N)).

Example 4.1.2. Let *reverse* be a predicate defined by

```
reverse([ ],[ ]).
reverse([X|L],M) :- reverse(L,N),append(N,[X],M).
```

Then *reverse** is

$\forall$ A,B (reverse(A,B) $\equiv$
(A=[ ]$\wedge$B=[ ]) $\vee\exists$X,L,M,N (A=[X|L]$\wedge$B=M$\wedge$reverse(L,N)$\wedge$append(N,[X],M))).

reverse* can be similarly simplified to

reverse([ ],[ ]) $\wedge\forall$X,L,M,N (reverse([X|L],M)$\equiv$ reverse(L,N)$\wedge$append(N,[X],M)).

4.2. Normal Forms of Goal Formulas

As defined for usual first order formulas, we can define conjunctive and disjunctive normal forms of S-formulas, which correspond conjunctive and disjunctive normal forms of goal formulas. In order to transform goal formula to their normal forms, we use the following transformations, of which (a) and (b) are used commonly, (c) is for conjunctive normal forms and (d) is for disjunctive normal forms.

(a) $G \supset H \rightarrow \neg G \vee H$,
(b) $\neg(G \vee H) \rightarrow \neg G \wedge \neg H$,
$\neg(G \wedge H) \rightarrow \neg G \vee \neg H$,
$\neg(\neg G) \rightarrow G$,
(c) $(F \wedge G) \vee H \rightarrow (F \vee H) \wedge (G \vee H)$,
$F \vee (G \wedge H) \rightarrow (F \vee G) \wedge (F \vee H)$,
(d) $(F \vee G) \wedge H \rightarrow (F \wedge H) \vee (G \wedge H)$,
$F \wedge (G \vee H) \rightarrow (F \wedge G) \vee (F \wedge H)$.

Let G be a goal formula. A goal formula G_{CNF} is called a *conjunctive normal form of* G when it is obtained by applying (a) first as far as possible, (b) next as far as possible and lastly (c) as far as possible. G_{CNF} corresponds to a first order formula of the form

$$\forall X_1, X_2, \ldots, X_n \; \exists Y_1, Y_2, \ldots, Y_m \; ((L_{11} \vee L_{12} \vee \cdots) \wedge (L_{21} \vee L_{22} \vee \cdots) \wedge \cdots \wedge (L_{k1} \vee L_{k2} \vee \cdots))$$

where each $(L_{i1} \vee L_{i2} \vee \cdots)$ is called a *conjunct*.

A goal formula G_{DNF} is called a *disjunctive normal form of* G when it is obtained by applying (a) first as far as possible, (b) next as far as possible and lastly (d) as far as possible. G_{DNF} corresponds to a first order formula of the form

$$\forall X_1, X_2, \ldots, X_n\ \exists Y_1, Y_2, \ldots, Y_m\ ((L_{11} \wedge L_{12} \wedge \cdots) \vee (L_{21} \wedge L_{22} \wedge \cdots) \vee \cdots \vee (L_{k1} \wedge L_{k2} \wedge \cdots))$$

where each $(L_{i1} \wedge L_{i2} \wedge \cdots)$ is called a *disjunct*.

In the transformations above, subformulas in the right-hand side are called *descended subformulas* of the subformula in the left-hand side with the same symbol.

Lemma 4.2.1. Let G be a goal formula, A be an atom in G, G_{CNF} be a conjunctive normal form of G and G_{DNF} be a disjunctive normal form of G.

(a) Let A be a positive atom of G. When some descended atom of A appears in some conjunct of G_{CNF}, it appears positively, i.e., in the form A in the conjunct. When some descended atom of A appears in some disjunct of G_{DNF}, it appears positively, i.e., in the form A in the disjunct.

(b) Let A be a negative atom of G. When some descended atom of A appears in some conjunct of G_{CNF}, it appears negatively, i.e., in the form $\neg A$ in the conjunct. When some descended atom of A appears in some disjunct of G_{DNF}, it appears negatively, i.e., in the form $\neg A$ in the disjunct.

Proof. Because subformulas in each transformation rule have same polarities in the left-hand side and the right-hand side, the descended atoms have the same polarity as the antecedent atom. It holds between G and G_{CNF} and between G and G_{DNF}.

Lemma 4.2.2. Let G be a goal formula and H be a subformula of G.

(a) When H appears positively in G (possibly at several occurrences), G is logically equivalent to $(H \wedge I) \vee J$ for some I and J.

(b) When H appears negatively in G (possibly at several occurrences), G is logically equivalent to $(\neg H \wedge I) \vee J$ for some I and J.

(c) When H appears positively in G (possibly at several occurrences), G is logically equivalent to $(H \vee I) \wedge J$ for some I and J.

(d) When H appears negatively in G (possibly at several occurrences), G is logically equivalent to $(\neg H \vee I) \wedge J$ for some I and J.

where there is no occurrences of H in I or J with the same porality as the original H.

Proof. Suppose H appears positively in G. First we transform G to its disjunctive normal form regarding H as a special atom. Because $H \wedge H$ is logically equivalent to H, it is logically equivalent to a disjunctive normal form with each disjunct containing at most one descended H of the occurrences of H in G. Let the disjuncts containing some descendant of the occurrences of H be

$$(H \wedge I_1) \vee (H \wedge I_2) \vee \cdots \vee (H \wedge I_k)$$

and the disjuncts containing no descendant of H be J. Then it is obviously equivalent to $(H \wedge I) \vee J$, where I is $I_1 \vee I_2 \vee \cdots \vee I_k$. Other cases are proved similarly.

Lemma 4.2.3. Let G be a goal formula.

(a) When a formula H of the form $H_1 \wedge H_2 \wedge \cdots \wedge H_k$ $(k > 1)$ occurs positively in G (possibly at several occurrences) and each undecided variable $?X$ appearing in H_i appears only in H_i $(1 \leq i \leq k)$, G is logically equivalent to $G_H[H_1] \wedge G_H[H_2] \wedge \cdots \wedge G_H[H_k]$.

(b) When a formula H of the form $H_1 \vee H_2 \vee \cdots \vee H_k$ $(k > 1)$ occurs negatively in G (possibly at several occurrences) and each undecided variable $?X$ appearing in H_i appears only in H_i $(1 \leq i \leq k)$, G is logically equivalent to $G_H[H_1] \wedge G_H[H_2] \wedge \cdots \wedge G_H[H_k]$.

(c) When a formula H of the form $H_1 \supset H_2$ occurs negatively in G (possibly at several oc-

currences) and each undecided variable $?X$ appearing in H_i appears only in H_i $(1 \leq i \leq 2)$, G is logically equivalent to $G_H[\neg H_1] \wedge G_H[H_2]$.

where $G_H[H_i]$ is a goal formula obtained by replacing the occurrences of H with H_i. Variables in G are renamed and not shared among $G_H[H_1], G_H[H_2], \ldots, G_H[H_k]$.

Proof. Let G be a goal formula. Suppose a formula of the form $H_1 \wedge H_2 \wedge \cdots \wedge H_k$ appears positively in G (possibly at several occurrences). Then,using Lemma 4.2.2.(a), G is logically equivalent to an S-formula of the form

$$\forall X_1, X_2, \ldots, X_n \; \exists Y_1, Y_2, \ldots, Y_l, Y_{l+1}, Y_{l+2}, \ldots, Y_m \; ((H \wedge I) \vee J),$$

where $Y_1, Y_2, \ldots, Y_l$ are undecided variables not in H and $Y_{l+1}, Y_{l+2}, \ldots, Y_m$ are those in H.

First, we move existential quantifiers for $Y_{l+1}, Y_{l+2}, \ldots, Y_m$ inwards. Because these variables appear just in some H_i, we have

$$\forall X_1, X_2, \ldots, X_n \; \exists Y_1, Y_2, \ldots, Y_l \; (((\exists H_1 \wedge \exists H_2 \wedge \cdots \wedge \exists H_k) \wedge I) \vee J)$$

where each $\exists H_i$ is existential quantified H_i.

Secondly, we move existential quantifications for $Y_1, Y_2, \ldots, Y_l$ inwards. Because $\exists$ is distributive over $\vee$ and these variables do not appear in H, we have

$$\forall X_1, X_2, \ldots, X_n \; (((H_1 \wedge H_2 \wedge \cdots \wedge H_k) \wedge \exists U_1, U_2, \ldots, U_l I') \vee \exists V_1, V_2, \ldots, V_l J')$$

where I' and J' are variants of I and J by renaming $Y_1, Y_2, \ldots, Y_l$ to $U_1, U_2, \ldots, U_l$ and $V_1, V_2, \ldots, V_l$.

Thirdly, we make k variants $\exists I_1, \exists I_2, \ldots, \exists I_k$ of $\exists U_1, U_2, \ldots, U_l I'$ by renaming $U_1, U_2, \ldots, U_l$. Because $\exists I_1 \wedge \exists I_2 \wedge \cdots \wedge \exists I_k$ is logically equivalent to $\exists U_1, U_2, \ldots, U_l I'$, we have

$$\forall X_1, X_2, \ldots, X_n((\exists H_1 \wedge \exists I_1) \wedge (\exists H_2 \wedge \exists I_2) \wedge \cdots \wedge (\exists H_k \wedge \exists I_k)) \vee \exists V_1, V_2, \ldots, V_l J').$$

Fourthly, we distribute $\vee$ over $\wedge$ using de Morgan's law. We have

$$\forall X_1, X_2, \ldots, X_n \; (((\exists H_1 \wedge \exists I_1) \vee \exists J_1) \wedge ((\exists H_2 \wedge \exists I_2) \vee \exists J_2) \wedge \cdots \wedge ((\exists H_k \wedge \exists I_k) \vee \exists J_k))$$

where we have made k variants $\exists J_1, \exists J_2, \ldots, \exists J_k$ of $\exists V_1, V_2, \ldots, V_l J'$ by renaming $V_1, V_2, \ldots, V_l$.

Fifthly, we move universal quantifiers for $X_1, X_2, \ldots, X_n$ inwards. Because $\forall$ is distributive over $\wedge$, we have a conjunction of k formulas

$$\begin{array}{l}
(\forall X_{11}, X_{12}, \ldots, X_{1n} \; ((\exists H_1 \wedge \exists I_1) \vee \exists J_1)) \; \wedge \\
(\forall X_{21}, X_{22}, \ldots, X_{2n} \; ((\exists H_2 \wedge \exists I_2) \vee \exists J_2)) \; \wedge \\
\quad \vdots \\
(\forall X_{k1}, X_{k2}, \ldots, X_{kn} \; ((\exists H_k \wedge \exists I_k) \vee \exists J_k))
\end{array}$$

by renaming $X_1, X_2, \ldots, X_n$.

Lastly, we move existential quantifiers of each conjunct outwards. We have

$$\begin{array}{l}
(\forall X_{11}, X_{12}, \ldots, X_{1n} \; \exists Y_{11}, Y_{12}, \ldots, Y_{1l}, \ldots \; ((\exists H_1 \wedge I_1) \vee J_1)) \; \wedge \\
(\forall X_{21}, X_{22}, \ldots, X_{2n} \; \exists Y_{21}, Y_{22}, \ldots, Y_{2l}, \ldots \; ((\exists H_2 \wedge I_2) \vee J_2)) \; \wedge \\
\quad \vdots \\
(\forall X_{k1}, X_{k2}, \ldots, X_{kn} \; \exists Y_{k1}, Y_{k2}, \ldots, Y_{kl}, \ldots \; ((\exists H_k \wedge I_k) \vee J_k))
\end{array}$$

Now each conjunct G_i is a variant of G except that H is replaced with H_i and has no common variable. Hence it is logically equivalent to $G_H[H_i]$. After all the original goal is logically equivalent to

$$G_H[H_1] \wedge G_H[H_2] \wedge \cdots \wedge G_H[H_k].$$

Other cases are proved similarly using Lemma 4.2.2.(b).

Lemma 4.2.4. Let G be a goal formula.

(a) When H is a negative subformula of the form $H_1 \wedge H_2 \wedge \cdots \wedge H_k$ $(k > 1)$, G is logically equivalent to $G_H[H_1] \vee G_H[H_2] \vee \cdots \vee G_H[H_k]$.

(b) When H is a positive subformula of the form $H_1 \vee H_2 \vee \cdots \vee H_k$ $(k > 1)$, G is logically equivalent to $G_H[H_1] \vee G_H[H_2] \vee \cdots \vee G_H[H_k]$.

(c) When H is a positive subformula of the form $H_1 \supset H_2$, G is logically equivalent to $G_H[\neg H_1] \vee G_H[H_2]$.

where $G_H[H_i]$ is a goal formula obtained by replacing the occurrences of H with H_i. Variables in G are shared among $G_H[H_1], G_H[H_2], \ldots, G_H[H_k]$.

Proof. Let G be a goal formula and H be a negative subformula of the form $H_1 \wedge H_2 \wedge \cdots \wedge H_k$. Then,using Lemma 4.2.2.(d), G is logically equivalent to an S-formula of the form

$$\forall X_1, X_2, \ldots, X_n \; \exists Y_1, Y_2, \ldots, Y_m \; ((\neg H \vee I) \wedge J).$$

First, we make k variants of I. Because $I \vee I \vee \cdots \vee I$ is logically equivalent to I, we have

$$\forall X_1, X_2, \ldots, X_n \; \exists Y_1, Y_2, \ldots, Y_m \; (((\neg H_1 \vee I) \vee (\neg H_2 \wedge I) \vee \cdots \vee (H_k \wedge I)) \wedge J).$$

Secondly, we distribute $\wedge$ over $\vee$ using de Morgan's law. We have

$$\forall X_1, X_2, \ldots, X_n \; \exists Y_1, Y_2, \ldots, Y_m \; (((\neg H_1 \vee I) \wedge J) \vee ((\neg H_2 \vee I) \wedge J) \vee \cdots \vee ((\neg H_k \vee I) \wedge J)).$$

Now each conjunct G_i is identical to G except that H is replaced with H_i. Hence it is logically equivalent to $G_H[H_i]$. After all, the original goal is logically equivalent to

$$G_H[H_1] \vee G_H[H_2] \vee \cdots \vee G_H[H_k].$$

Other cases are proved similarly using Lemma 4.2.2.(c).

Lemma 4.2.5. Let G be a goal formula and A be an atom containing no undecided variable.

(a) When $G_A(true) \downarrow$ is a logical consequence of P^*, then $A \supset G$ is also a logical consequence of P^*.

(b) When $G_A(false) \downarrow$ is a logical consequence of P^*, then $\neg A \supset G$ is also a logical consequence of P^*.

Proof. We prove (a) in 2 steps.

First, it is obvios that, when $G_A(true) \downarrow$ is a logical consequence of P^*, so is $A \supset G_A(true) \downarrow$

Secondly, let G_{CNF} be a conjunctive normal form of G of the form

$$C_1 \wedge C_2 \wedge \cdots \wedge C_k$$

Then $A \supset G$ is logically equivalent to $\neg A \vee G_{CNF}$. By distributing $\vee$ over $\wedge$, it is logically equivalent to

$$(\neg A \vee C_1) \wedge (\neg A \vee C_2) \wedge \cdots \wedge (\neg A \vee C_k).$$

When C_i contains A, $\neg A \vee C_i$ is logically equivalent to $\neg A \vee C_{iA}(true) \downarrow$, i.e., $true$. When C_i contains no A but $\neg A$, then again $\neg A \vee C_i$ is logically equivalent to $\neg A \vee C_{iA}(true) \downarrow$, because $\neg A \vee \neg A \vee \cdots \vee \neg A$ is logically equivalent to $\neg A$. When C_i contains neither A nor $\neg A$, $\neg A \vee C_i$ is obviously logically equivalent to $\neg A \vee C_{iA}(true) \downarrow$. Since this holds for all i, $A \supset G$ is logically equivalent to $A \supset G_A(true) \downarrow$.

After all, when $G_A(true) \downarrow$ is a logical consequence of P^*, so is $A \supset G$. The case (b) is proved similarly.

Lemma 4.2.6. Let G be a goal formula, $[s_1, s_2, \ldots, s_n], [t_1, t_2, \ldots, t_n]$ be two lists of terms, which contain no undecided variable and are unifiable by an m.g.u. τ and $B_1, B_2, \ldots, B_m$ be atoms containing no undecided variable. When $\tau(G)$ is a logical consequence of P^*, so is $s_1 = t_1 \wedge s_2 = t_2 \wedge \cdots \wedge s_n = t_n \wedge B_1 \wedge B_2 \wedge \cdots \wedge B_m \supset G$.

Proof. We prove it in 2 steps.

First, using the set of axioms EQ for $=$, when $\tau(G)$ is a logical consequence of P^*, so is $s_1 = t_1 \wedge s_2 = t_2 \wedge \cdots \wedge s_n = t_n \supset G$.

Secondly, when $s_1 = t_1 \wedge s_2 = t_2 \wedge \cdots \wedge s_n = t_n \supset G$.

is a logical consequence of P^*, so is $s_1 = t_1 \wedge s_2 = t_2 \wedge \cdots \wedge s_n = t_n \wedge B_1 \wedge B_2 \wedge \cdots \wedge B_m \supset G$.

After all, when $r(G)$ is a logical consequence of P^*, so is $s_1 = t_1 \wedge s_2 = t_2 \wedge \cdots \wedge s_n = t_n \wedge B_1 \wedge B_2 \wedge \cdots \wedge B_m \supset G$.

4.3. Proof of the Soundness

Now we prove the soundness of our extended execution.

Lemma 4.3.1. Let ?X be an undecided variable in a goal formula G and $\sigma = <?X \Leftarrow t>$ be a deciding substitution. When $\sigma(G)$ is a logical consequence of P^*, so is G. (All new variables introduced are treated as fresh undecided variables. Now on, we call the rule to generate subgoal $\sigma(G)$ from G *oracle decision*.)

Proof. Trivial.

Lemma 4.3.2. Case splittings are sound.

Proof. It is a special case of Lemma 4.2.3. where we have just one occurrence of H.

Lemma 4.3.3. DCI is sound.

Proof. We prove it in 6 steps. Let A be a positive atom with its predicate symbol p in a goal formula G. Suppose A and the head $p(t_1, t_2, \ldots, t_n)$ of the i-th definite clause for p are unifiable to $p(s_1, s_2, \ldots, s_n)$ by a deciding m.g.u. $\sigma \circ \mu$, where σ and μ are the restrictions of the m.g.u. to undecided variables in G and to head variables of the definite clause, respectively.

First, because oracle decision is sound by Lemma 4.3.1, when $\sigma(G)$ is a logical consequence of P^*, so is G.

Secondly, by instantiating universally quantified variables $X_1, X_2, \ldots, X_n$ in the completion p^* to $s_1, s_2, \ldots, s_n$ respectively, we have

$$p(s_1,s_2,\ldots,s_n) \equiv (H_1 \vee H_2 \vee \cdots \vee H_k)$$

from P^*, where H_i is of the form

$$\exists Y_1,Y_2,\ldots,Y_u,Z_1,Z_2,\ldots,Z_v\ (s_1=t_1 \wedge s_2=t_2 \wedge \cdots \wedge s_n=t_n \wedge B_1 \wedge B_2 \wedge \cdots \wedge B_m),$$

$Y_1, Y_2, \ldots, Y_u$ are head variables and $Z_1, Z_2, \ldots, Z_v$ are internal variables. By replacing equivalence with equivalence, $\sigma(G)$ is a logical consequence of P^* if and only if $\sigma(G)_{p(s_1,s_2,\ldots,s_n)}[H_1 \vee H_2 \vee \cdots \vee H_k]$ is a logical consequence of P^*.

Thirdly, by Lemma 4.2.4, $\sigma(G)_{p(s_1,s_2,\ldots,s_n)}[H_1 \vee H_2 \vee \cdots \vee H_k]$ is logically equivalent to $\sigma(G)_{p(s_1,s_2,\ldots,s_n)}[H_1] \vee \sigma(G)_{p(s_1,s_2,\ldots,s_n)}[H_2] \vee \cdots \sigma(G)_{p(s_1,s_2,\ldots,s_n)}[H_k]$.

Fourthly, it is obvious that, when $\sigma(G)_{p(s_1,s_2,\ldots,s_n)}[H_i]$ is a logical consequence of P^*, so is $\sigma(G)_{p(s_1,s_2,\ldots,s_n)}[H_1] \vee \sigma(G)_{p(s_1,s_2,\ldots,s_n)}[H_2] \vee \cdots \sigma(G)_{p(s_1,s_2,\ldots,s_n)}[H_k]$.

Fifthly, because oracle decision to $\sigma(G)_{p(s_1,s_2,\ldots,s_n)}[H_i]$ by mu in order to instantiate the head variables $Y_1, Y_2, \ldots, Y_u$ is sound by Lemma 4.3.1, when $\sigma(G)_{p(s_1,s_2,\ldots,s_n)}[\mu(H_i)]$ is a logical consequence of P^*, so is $\sigma(G)_{p(s_1,s_2,\ldots,s_n)}[H_i]$, where $\mu(H_i)$ is of the form

$$\exists\ Z_1, Z_2, \ldots, Z_v\ (s_1 = s_1 \wedge s_2 = s_2 \wedge \cdots \wedge s_n = s_n \wedge \mu(B_1 \wedge B_2 \wedge \cdots \wedge B_m)).$$

Sixthly, because $\mu(H_i)$ is logically equivalent to

$$\exists\ Z_1, Z_2, \ldots, Z_v\ \sigma(B_1 \wedge B_2 \wedge \cdots \wedge B_m),$$

$\sigma(G)_{p(s_1,s_2,\ldots,s_n)}[\mu(H_i)]$ is logically equivalent to $\sigma \circ \mu(G_A[\wedge_{j=0}^m B_j])$.

After all, when $\sigma \circ \mu(G_A[\wedge_{j=0}^m B_j])$ is a logical consequence of P^*, so is G.

Lemma 4.3.4. NFI is sound.

Proof. We prove it in 2 steps. Let A be a negative atom $p(s_1, s_2, \ldots, s_n)$ of a goal formula

G containing no undecided variable.

First, it is obvious that G is logically equivalent to $A \supset G$ and $\neg A \supset G$.

Secondly, as for the first goal formula, by instantiating universally quantified variables $X_1, X_2, \ldots, X_n$ in the completion p^* to $s_1, s_2, \ldots, s_n$ respectively, we have $p(s_1, s_2, \ldots, s_n) \equiv (H_1 \vee H_2 \vee \cdots \vee H_l)$
where H_i is of the form

$$\exists\, Y_1, Y_2, \ldots, Y_u, Z_1, Z_2, \ldots, Z_v\ (s_1 = t_1 \wedge s_2 = t_2 \wedge \cdots \wedge s_n = t_n \wedge B_1 \wedge B_2 \wedge \cdots \wedge B_m)$$

By applying Lemma 4.3.3.(a) for A in the antecedant and the negative occurrence in G, $A \supset G$ is logically equivalent to

$$H_1 \supset G_A[H_1],\ H_2 \supset G_H[H_2],\ \ldots,\ H_l \supset G_H[H_l].$$

Suppose the equational part $s_1 = t_1 \wedge s_2 = t_2 \wedge \cdots \wedge s_n = t_n$ are unifiable for $H_1, H_2, \ldots, H_k$, say by m.g.u's $\tau_1, \tau_2, \ldots, \tau_k$ and not unifiable for $H_{k+1}, H_{k+2}, \ldots, H_l$. By Lemma 4.2.6., when

$$\tau_1(G_A[\wedge_{j=0}^{m_1} B_{1j}])\downarrow,\ \tau_2(G_A[\wedge_{j=0}^{m_2} B_{2j}])\downarrow,\ \ldots\ \tau_k(G_A[\wedge_{j=0}^{m_k} B_{kj}])\downarrow$$

are logical consequences of P^*, so are

$$H_1 \supset G_A[H_1],\ H_2 \supset G_H[H_2],\ \ldots,\ H_k \supset G_H[H_k].$$

Other goal formulas $H_{k+1} \supset G_H[H_{k+1}],\ H_{k+2} \supset G_H[H_{k+2}],\ \ldots, H_l \supset G_H[H_l]$ are always logical consequences of P^*. As for the second goal formula, using Lemma 4.3.5.(b), when $G_A[false]\downarrow$ is a logical consequence of P^*, so is $\neg A \supset G$.

After all, when $G_A[false]\downarrow, \tau_1(G_A[\wedge_{j=0}^{m_1} B_{1j}])\downarrow, \tau_2(G_A[\wedge_{j=0}^{m_2} B_{2j}])\downarrow, \ldots, \tau_k(G_A[\wedge_{j=0}^{m_k} B_{kj}])\downarrow$ are logical consequences of P^*, so is G.

Lemma 4.3.5. Simplification is sound.

Proof. We prove it in 3 steps. Let $A_1, A_2, \ldots, A_n$ be atoms in a goal formula G unifiable to A by a deciding m.g.u. σ.

First, because oracle decision is sound by Lemma 4.3.1, when $\sigma(G)$ is a logical consequence of P^*, so is G.

Secondly, it is obvios that $\sigma(G)$ is logically equivalent to $A \supset \sigma(G)$ and $\neg A \supset \sigma(G)$.

Thirdly, by Lemma 4.2.5.(a) for the first goal formula, when $\sigma(G)_A(true)\downarrow$ is a logical consequence of P^*, so is $A \supset \sigma(G)$. Similarly, by Lemma 4.2.5.(b) for the second goal formula, when $\sigma(G)_A(false)\downarrow$ is a logical consequence of P^*, so is $\neg A \supset \sigma(G)$.

After all, when $\sigma(G)_A(true)\downarrow$ and $\sigma(G)_A(false)\downarrow$ are logical consequences of P^*, so is G.

From Lemmas 4.3.2—4.3.5, extended execution is sound for goal formulas.

Theorem 4.3. (Soundness for S-formulas)

Let G be a goal formula and $G_1, G_2, \ldots, G_k$ be goal formulas generated as subgoals by applying some rule of extended execution. When $G_1, G_2, \ldots, G_k$ are all logical consequences of P^*, then the goal formula G is a logical consequence of P^*. That is, extended execution is sound for S-formulas.

5. Completeness of Extended Execution

In this section, we restrict our attention to a class of extended execution sequences, called normal extended execution, whose initial goal formula is a universal formula. (Throughout Section 5 except Corollary 5.5, we assume that the initial goal formula G always contains no undecided variable and use universal formulas and undecided-variable-free goal formulas indistinguishably.) Then, we restrict our attention further to the cases where no case splitting

is applicable to the initial universal formula. After preparing several notions concerning sequences of NFIs, we show that fair application of NFI to such a formula does not lose provability by normal extended execution. Then we introduce a sequence of interpretations associated with fair application of NFI when such a formula is not provable by normal extended execution. Lastly, we prove the completeness theorem using these notions.

5.1. Normal Extended Execution

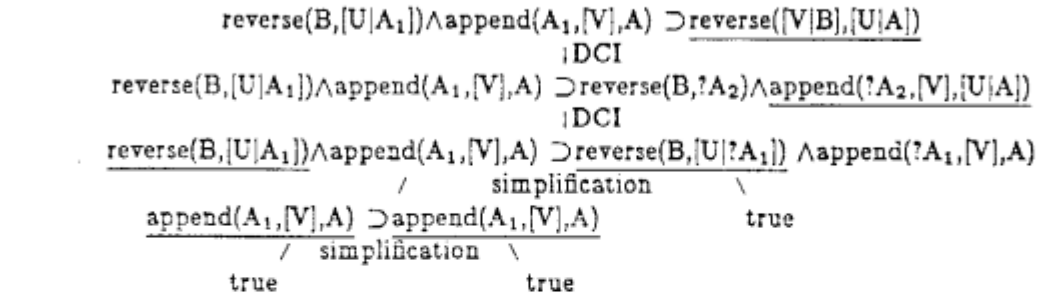
In this section, we consider the trees corresponding to extended execution. Application of a sequence of extended execution rules can be regarded as generation of an AND-tree.

Let P be a program and G be a goal formula. A tree is called an *extended execution tree* of G in P, when it satisfies the following conditions.

(a) Each node of the tree is a goal formula.
(b) The root node is G.
(c) When a case splitting is applied to a subformula H of a node, then this node has k descendants for each AND-goals generated by the case splitting. H and the case splitting rule are called a *selected subformula* and a *selected rule* at this node, respectively.
(d) When a DCI is applied to a positive atom A in a node, then this node has a descendant generated by the DCI. A and DCI are called a *selected atom* and a *selected rule* at this node, respectively.
(e) When an NFI is applied to a negative atom A in a node, then this node has $k+1$ descendants for each AND-gaols generated by the NFI. A and NFI are called a *selected atom* and a *selected rule* at this node, respectively.
(f) When a simplification is applied to atoms $A_1, A_2, \ldots, A_n$ in a node, then this node has two descendants for each AND-gaols generated by the simplification. $A_1, A_2, \ldots, A_n$ and simplification are called *selected atoms* and a *selected rule* at this node, respectively.

A branch from the root G in an extended execution tree is called a *derivation*. A derivation which ends with *true* is called a *proving derivation*. An extended execution tree is called a *proving extended execution tree* when every derivation in the extended execution tree is a proving derivation.

Example 5.1.1. The tree below is a proving execution tree.



The underlined atoms denote the selected atoms.

An extended execution tree of a universal formula G is called a *normal extended execution tree* of G when, for every derivation in the extended execution tree, the sequence of the selected rules on it consists of 4 phases satisfying the following conditions.

(a) The first phase is a (possibly empty) sequence of case splittings. All case splittings must be applied to the universal formulas as far as possible. Among the applicable case

splittings, those to the outer subformulas are prefered.
(b) The second phase is a (possibly empty) sequence of NFIs.
(c) The third phase is a (possibly empty) sequence of DCIs.
(d) The fourth phase is a (possibly empty) sequence of simplifications. The simplifications must be done for positive atoms and one negative atom in the goal formulas.

A normal extended execution tree is said to be *NFI-free* when no NFI is a selected rule in the tree. Note that no fresh free variable is introduced into the succeeding goal formulas after all NFIs were applied in the second phase.

Example 5.1.2. The tree in the example above is an NFI-free normal extended execution tree, where the first phase is empty.

Lemma 5.1.1. Let G be a universal formula and τ be an instantiation of free variables in G. If G is provable by normal exetended execution, so is $\tau(G)$.

Proof. Let T be a normal extended execution tree of G. Then let T' be a tree obtained from T by applying τ to each node and deleting the subtrees whose roots cannot be generated by the corresponding NFIs due to overinstantiation by τ. Then T' is the normal extended execution tree of $\tau(G)$.

Lemma 5.1.2. Let G be a universal formula and $G_1, G_2, \ldots, G_k$ be all the case-splitting-free universal formulas obtained from G by applying case splitting as far as possible. Then G is provable by normal extended execution if and only if all $G_1, G_2, \ldots, G_k$ are provable by normal extended execution.

Proof. Trivial.

Before going on, we show several properties of universal formulas to which no case splitting is applicable.

A universal formula G is said to be *case-splitting-free* when no case splitting is applicable to G, i.e., there is no positive subformula of the form $H_1 \wedge H_2$, no negative subformula of the form $H_1 \vee H_2$ and no negative subformula of the form $H_1 \supset H_2$.

Now, we notice the relation between the positive and negative parts in the sense of Schütte [18] and the positive and negative subformulas in this paper. The *positive* and *negative part* of a formula $\mathcal{F}$ are defined as follows.
(a) $\mathcal{F}$ is a positive part of $\mathcal{F}$.
(b) When $\neg\mathcal{G}$ is a positive (negative) part of $\mathcal{F}$, then $\mathcal{G}$ is a negative (positive) part of $\mathcal{F}$.
(c) When $\mathcal{G} \vee \mathcal{H}$ is a positive part of $\mathcal{F}$, then $\mathcal{G}$ and $\mathcal{H}$ are positive parts of $\mathcal{F}$.
(d) When $\mathcal{G} \wedge \mathcal{H}$ is a negative part of $\mathcal{F}$, then $\mathcal{G}$ and $\mathcal{H}$ are negative parts of $\mathcal{F}$.
(e) When $\mathcal{G} \supset \mathcal{H}$ is a positive part of $\mathcal{F}$, then $\mathcal{G}$ is a negative part of $\mathcal{F}$ and $\mathcal{H}$ is a positive part of $\mathcal{F}$.

Intuitively, a positive part of $\mathcal{F}$ is a subformula of $\mathcal{F}$ such that $\mathcal{F}$ is *true* when it is *true* and a negative part of $\mathcal{F}$ is a subformula of $\mathcal{F}$ such that $\mathcal{F}$ is *true* when it is *false*. Positive parts are always positive subformulas and negative parts are always negative subformulas, but not vice versa in general.

Lemma 5.1.3. Let G be a case-splitting-free universal formula and H be a subformula of G.

Then H is a positive part of G when it is a positive subformula of G and H is a negative part of G when it is a negative subformula of G.

Proof. By induction on the superterm-subterm relation on the set of subformulas of G. (Subformulas are greater than superformulas.)
Base Case : Trivial, because G is not only a positive formula but also a positive part of itself.
Induction Step : Suppose the lemma holds for a positive subformula H of G. When H is either of the form $\neg H', H_1 \vee H_2, H_1 \supset H_2$, the lemma holds obviously for H', H_1, H_2 from the definition. When H is of the form $H_1 \wedge H_2$, it contradicts the fact that no case splitting is applicable to G. The case H is negative is proved similarly.

Lemma 5.1.4. Let G be a case-splitting-free universal formula, A be a negative atom of G and "A_0 :- $A_1, A_2, \ldots, A_m$" be a definite clause ($m \geq 0$) with which an NFI is applicable to A using an m.g.u. τ. Then

(a) $\tau(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m])\downarrow$ is a case-splitting-free universal formula and each $\tau(A_j)$ is a negative part of $\tau(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m])$.
(b) If B is another atom of G, $\tau(B)$ remains in $\tau(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m])\downarrow$.

Proof. When $m > 0$, the lemma is trivial. When $m = 0$, $A_1 \wedge A_2 \wedge \cdots \wedge A_m$ is *true*. Let H be any subformula of G. Because G is case-splitting-free, H must be a positive subformula of the form $H_1 \vee H_2$, $H_1 \supset H_2$, $\neg H'$ or H must be a negative subformula of the form $H_1 \wedge H_2$, $\neg H'$.
Case 1 : When H is a positive subformula of the form $H_1 \vee H_2$ and A is in H_1, A is a negative subformula of H_1. By Lemma 2.3, $H_{1A}[true]\downarrow$ can't be *true*. Hence H_2 reamins in $G_A[true]\downarrow$. The case when A is in H_2 is proved similarly.
Case 2 : When H is a positive subformula of the form $H_1 \supset H_2$ and A is in H_1, A is a positive subformula of H_1. By Lemma 2.3, $H_{1A}[true]\downarrow$ can't be *false*. Hence H_2 reamins in $G_A[true]\downarrow$. The case when A is in H_2 is proved similarly.
Case 3 : When H is a negative subformula of the form $H_1 \wedge H_2$ and A is in H_1, A is a positive subformula of H_1. By Lemma 2.3, $H_{1A}[true]\downarrow$ can't be *false*. Hence H_2 remains in $G_A[true]\downarrow$. The case when A is in H_2 is proved similarly.

Because it holds for any subformula of G, just a subformula of the form $\neg\neg\cdots\neg A$ disappears from G when A is replaced with *true*. Hence $\tau(G_A[true])\downarrow$ is still case-splitting-free and $\tau(B)$ remains in $\tau(G_A[true])\downarrow$ if B is another atom of G.

Lemma 5.1.5. Let G ba a case-splitting-free universal formula and G' be a goal formula obtained from G by applying a sequence of DCIs. Then a negative atom of G' is a negative part of G'.

Proof. Trivial.

5.2. "Negation as Failure" Tree

In this section, we focus our attention to extended execution trees such that only NFIs are applied to case-splitting-free universal formulas.

Let P be a program and G be a case-splitting-free universal formula. A tree is called a *"Negation as Failure" tree* of G in P (or *NFI tree* for short), when it satisfies the following conditions.

(a) Each node of the tree is a case-splitting-free universal formula.
(b) The root node is G.

(c) When G contains a negative atom, let A be a negative atom in G called a *selected* atom. Then this node has $k+1$ descendants for each AND-gaols generated by NFI. (Note that the first goal obtained by replacing A with $false$ is always *true* due to Lemma 5.1.3.)
(d) When G contains no negative atom, this node has no descendant.

A branch from the root G in an NFI tree is called an *NFI derivation* and denoted by a sequence of triples $< G_i, C_i, \theta_i >$, where each G_{i+1} is the result of NFI using a definite clause C_i and an m.g.u. θ_i. An NFI derivation which contains a node provable by some NFI-free normal extended execution is called a *proving NFI derivation*. An NFI tree is called a *proving NFI tree* when every NFI derivation in the NFI tree is a proving NFI derivation.

For a given NFI derivation $< G_i, C_i, \theta_i >$, a negative atom B_k in G_k is called an *instantiated copy* of negative atom B_j in G_j $(j \leq k)$, if there is a sequence of atoms $B_j, B_{j+1}, \ldots, B_k$ with $\theta_j(B_j) = B_{j+1}$, $\theta_{j+1}(B_{j+1}) = B_{j+2}$, $\ldots$, $\theta_{k-1}(B_{k-1}) = B_k$.

For a given NFI derivation $< G_i, C_i, \theta_i >$, an atom in G_i is a *descended atom* of an atom A_0 in G_0 *with level* 0 if it is an instantiated copy of A_0. An atom in G_i is a *descended atom* of atom A_0 in G_0 *with level* n if it is an instantiated copy of an atom which was introduced by applying NFI to a descended atom of A_0 with *level* $n-1$.

Let Th be the set of all case-splitting-free universal formulas provable by normal extended execution. $G \in Th$ is said to be *provable by NFI depth* d when it satisfies the following conditions and denoted by $G \in Th_d$.
(a) If G is provable by NFI-free normal extended execution, $G \in Th_0$.
(b) If there exists an NFI to a negative atom B of G generating subgoals $G_0, G_1, G_2, \ldots, G_k$ such that $G_0, G_1, G_2, \ldots, G_k \in Th_{d-1}$, then $G \in Th_d$. (Such atoms are called *depth-reducing atoms of* G.)

Lemma 5.2.1. Let G be a case-splitting free universal formula provable by normal extended execution, A be a negative atom of G and "A_0 :- $A_1, A_2, \ldots, A_m$" be a definite clause $(m \geq 0)$ whose head A_0 is unifiable with A by an m.g.u. τ. Then $\tau(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow$ is also provable by normal extended execution.

Proof. By induction on the NFI-depth of G. Let T be the proving normal extended execution tree of G. In the following, we construct a proving normal extended execurion tree T' of $\tau(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow$ from T.
Base Case : When $G \in Th_0$, there is an NFI-free tree T.
Case 1 : When the occurrence of A is not selected in the simplification in the fourth phase in T, let T' be the tree obtained by applying τ and replacing $\tau(A)$ with $\tau(A_1 \wedge A_2 \wedge \cdots \wedge A_m)$ in each node in T.
Case 2 : When the occurrence of A is selected in some simplification with positive atoms $B_1, B_2, \ldots$ and the negative atom A, let T' be the tree obtained by applying τ, applying DCIs to $\tau(B_1), \tau(B_2), \ldots$ at the end of the third phase and using new simplifications between positive atoms and negative $\tau(A_i)$'s for each i $(1 \leq i \leq m)$. (See Lemma 5.1.5.).
Induction Step : When $G \in Th_d$, there is a proving normal extended execution tree T of G at the top of which an NFI is applied to a negative atom B to generate $k+1$ case-splitting-free universal formulas
$G_0 : G_B[false] \downarrow$,
$G_1 : \nu_1(G_B[B_{11} \wedge B_{12} \wedge \cdots \wedge B_{1m_1}]) \downarrow$,
$G_2 : \nu_2(G_B[B_{21} \wedge B_{22} \wedge \cdots \wedge B_{2m_2}]) \downarrow$,

$\vdots$

$G_k : \nu_k(G_B[B_{k1} \wedge B_{k2} \wedge \cdots \wedge B_{km_k}]) \downarrow$

such that $G_0, G_1, G_2, \ldots, G_k \in Th_{d-1}$.

Case 1 : When the occurrence of B is identical to that of A, some G_i is identical to $\tau(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow$. Let T' be the immediate subtree of T with its root G_i.

Case 2 : When the occurrence of B is different from that of A, the occurrence of B remains in $\tau(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow$ by Lemma 5.1.4 and the same NFI is applicable to $\tau(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow$ to generate new universal formulas. Because the instantiation of A to $\tau(A)$ may instantiate free variables in B, the number of the generated universal formulas $j+1$ is less than or equal to $k+1$. Say, the first j clauses are used in the NFI with m.g.u.'s $\nu'_1, \nu'_2, \ldots, \nu'_j$. Let τ_i be defined by $\nu'_i \circ \tau = \tau_i \circ \nu_i$. Then the universal formulas generated from $\tau(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow$ are identical to

$G'_0 : true,$

$G'_1 : \tau_1(G_{1\nu_1(A)}[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow,$

$G'_2 : \tau_2(G_{2\nu_2(A)}[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow,$

$\vdots$

$G'_j : \tau_j(G_{j\nu_j(A)}[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow.$

By the induction hypothesis, these $G_{i\nu_i(A)}[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow$ are provable by normal extended execution. Hence, by Lemma 5.1.1, $G'_0, G'_1, \ldots, G'_j$ have normal extended excution trees $T'_0, T'_1, \ldots, T'_j$. Let T' be the tree consisting of them with its root $\tau(G_A[A_1 \wedge A_2 \wedge \cdots \wedge A_m]) \downarrow$.

Lemma 5.2.2. Let G be a case-splitting-free universal formula and G' be a case-splitting-free universal formula which is an immediate descended node of G with its selected atom A.

(a) When $d(G) = 0$, $d(G') = 0$

(b) When $d(G) > 0$, let B be a depth-reducing atom of G. If A and B is an identical occurrence of an atom, then $d(G) > d(G')$. If A and B are different occurrences of atoms, then $d(G') \geq d(G)$, some instantiated copy of B remains in G' and the instantiated copy of B is still a depth-reducing atom of G'.

Proof. It is trivial from Lemma 5.2.1 that, when G is provable by normal extended execution, G' is also provable by normal extended execution. The case (a) is obvious from Base Case there. The former half of the case (b) is obvious from Case 1 in Induction Step and the latter half from Case 2 there.

5.3. Fair "Negation as Failure" Derivation

In this section, we introduce *fairness* of the application of NFIs following Lassez and Maher [12] and show the strong completeness of fair NFI, that is, all fair NFIs generate proving NFI trees if and only if there exists a proving NFI tree.

An NFI derivation is said to be *fair* if, for every atom B in the derivation, either some instantiated copy is selected within a finite number of steps or some instantiated copy is in G_i provable by NFI-free normal extended execution.

A *computational rule* is a rule to choose the selected atom from negative atoms. A computational rule is said to be *fair* if every NFI derivation produced from it is fair.

Lemma 5.3. Let G be a case-splitting-free universal formula. Any fair NFI reduces G to a set of case-splitting-free universal formulas provable by NFI-free normal extended execution

if and only if all fair NFI reduces G to a set of case-splitting-free universal formulas provable by NFI-free normal extended execution.

Proof. By induction on the depth d of G.
Base Case : When $d = 0$, any universal formula obtained from G by applying NFI fairly is with depth 0. Hence the lemma is trivial.
Induction Step : When $d > 0$, let T be a fair NFI tree of G, Br be any derivation in T and B be some depth-reducing atom of G. Because the computation rule is fair, an instantiated copy of B is eventually selected from the case-splitting-free universal formulas on Br and their depths decrease by Lemma 5.2.2. Let G' be a case-splitting-free universal formula on Br whose depth is less than that of G. Then by induction hypothesis, G' has a fair proving NFI tree.

5.4. Models Associated with "Negation as Failure" Derivations

In this section, we define a sequence of models associated with a sequence of case-splitting-free universal formulas in a non-proving NFI derivation.

In the following, we assume a fixed set of constants, funtion symbols and prediate symbols in P. An expressions of the form $p(d_1, d_2, \ldots, d_n)$ is called a *D-atom* when p is an n-ary predicate symbol in P and $d_1, d_2, \ldots, d_n$ are elements of a set D. In general, let D be a fixed domain of interpretation for P with some fixed assignment of constants in P to elemnts of D and functions in P to functions on D and let B_D be the set of all D-atoms.

Depending on the assignments of the predicates of P, we have variety of interpretations for P. Such an interpretation can be identified with some subset of B_D. (When $p(d_1, d_2, \ldots, d_n)$ is in this subset, it is considered true.) All such interpretations naturally form a complete lattice with respect to the partial order of set inclusion. Following Jaffar et al [7], a monotone transformation T from this lattice to itself is defined by

$$T(I) = \{p(d_1, d_2, \ldots, d_n) \mid \text{“}B_0 \text{ :- } B_1, B_2, \ldots, B_m\text{.”} \in P \text{ and}$$
$$\text{there is an assignment of the variables in the definite clause to elements of } D$$
$$\text{such that with this assignment } B_0 \text{ becomes } p(d_1, d_2, \ldots, d_n) \text{ and}$$
$$\text{all } B_1, B_2, \ldots, B_n \text{ are in } I \}.$$

We denote the set of all terms on set of variables $\mathcal{V}$ by $H(\mathcal{V})$ and the set of all $H(\mathcal{V})$-atoms by $B(\mathcal{V})$. ($H(\emptyset)$ and $B(\emptyset)$ are the usual Herbrand universe and Herbrand base.) Subsets of $B(\mathcal{V})$ are considered interpretations with its domain $H(\mathcal{V})$ as subsets of the usual Herbrand base are.

Let $G(= G_0)$ be a case-splitting-free universal formula not provable by normal extended execution and suppose that there is a non-proving fair NFI derivation $BR =< G_i, C_i, \theta_i >$ In order to make the following discussion simple, we assume that each free variable in the universal formulas $G_0, G_1 \ldots$ is renamed so that these formulas have no common free variable. Let $\mathcal{V}_0, \mathcal{V}_1, \mathcal{V}_2, \ldots$ be the sets of all free variables in $G_0, G_1, G_2, \ldots$ respectively and $\mathcal{V}$ be $\bigcup_i \mathcal{V}_i$. Then every free variable in G_i and variable in C_i is instantiated by θ_i to a term containing only free variables in $\mathcal{V}_{i+1}$.

Let $A_0, A_1, A_2, \ldots$ be the sequence of the set of all negative atoms in $G_0, G_1, G_2, \ldots$. We define a sequence $J_0, J_1, J_2, \ldots$, where each J_i is a subset of $B(\mathcal{V}_i)$ as follows.

$J_i = \bigcup_{k=0}^{\infty} T^k(A_i)$.

Lemma 5.4.1. J_i is a model of P with its domain $H(\mathcal{V}_i)$ for all $i \geq 0$.

Proof. It is trivial because $J_i \supseteq T(J_i)$ for all $i \geq 0$.

Lemma 5.4.2. When an atom A is in J_i, a conjunction of atoms can be generated from A by a sequence of DCIs such that some instances of these atoms, say by σ, are all in A_i. (Note that variables in $\mathcal{V}_i$ are treated like constants. σ does not instantiate these variables.)

Proof. By induction on the minimum number l such that $A \in \bigcup_{k=0}^{l} T^k(A_i)$.
Base Case : When $l = 0$, A itself is an atom in A_i.
Induction Step : Suppose that the lemma holds for all atoms in $\bigcup_{k=0}^{l} T^k(A_i)$ and A is in $\bigcup_{k=0}^{l+1} T^k(A_i)$. Then there is an instance of a definite clause "$A \leftarrow A_1, A_2, \ldots, A_m$" such that some instance of $A_1, A_2, \ldots, A_m$ are all in $\bigcup_{k=0}^{l} T^k(A_i)$. By the induction hypothesis, conjunctions of atoms can be generated from the instances of $A_1, A_2, \ldots, A_m$ by sequences of DCIs such that some instances of these atoms are all in A_i. Then, the desired conjunction of atoms can be generated from A by a sequence of DCIs as follows : Apply DCI to A using the definite clause and apply the sequences of DCIs to each A_j.

Lemma 5.4.3. Under the interpretation assigning X as an element of $H(\mathcal{V}_i)$ to each free variable X in G_i, G_i is not valid in J_i for all $i \geq 0$.

Proof. Suppose G_i is valid in J_i under the interpretation. Then some positive atom A in G_i must be in J_i. By Lemma 5.4.2, there is a sequence of DCIs generating a conjunction of atoms from A such that some instances of these atoms are all in A_i i.e., negative atoms in G_i. Apply the same sequence of DCIs to G_i. Then, because A is a positive part of G_i and each negative atoms of G_i is a negative part of G_i, application of simplifications on these atoms always generates *true* from the resultant goal formulas. Hence G_i is provable by normal extended execution, which contradicts the fact that BR is a non-proving NFI derivation.

5.5. Proof of the Completeness

Now we show a proof of the following lemma.

Lemma 5.5. A case-splitting-free universal formula S is provable by normal extended execution if and only if S is a logical consequence of P^*.

We prove the lemma by showing existence of a model of $P^* \bigcup \{\neg S\}$ for any case-splitting-free universal formula S not provable by normal extended execution. Similarly to Jaffar et al [7], we construct a model which is not necessarily isomorphic to Herbrand models. Suppose there is a non-proving NFI derivation $BR = < G_i, C_i, \theta_i >$, where G_0 is the goal formula of S.

5.5.1. Definition of Domain

Here we assume the terminology and notations of term rewriting system such as occurrence, independence of occurrences, subterm of t at occurrence w denoted by t/w and replacement of a subterm of t at occurrence w with term s denoted by $t[w \Leftarrow s]$ (see e.g., Jaffar et al [7] p.503 or Huet [6] pp.807).

Certain binary relations are defined following Jaffar et al [7] p.503. A *rewrite* α is of the form $< w, X, t >$. It defines a mapping from $H(\mathcal{V})$ into itself such that

$$s < w, X, t >= \begin{cases} s[w \Leftarrow t], & w \text{ is an occurrence of a variable } X \text{ in } s; \\ s & \text{otherwise.} \end{cases}$$

Two rewrites $< w_1, X_1, t_1 >$ and $< w_2, X_2, t_2 >$ are said to be *independent* when w_1 and w_2 are independent. A rewrite α is said to be *superfluous* for a term t when $t\alpha = t$.

Let $\mathcal{R}$ be a set of rewites on $H(\mathcal{V})$. Then

(a) $s <_n t$ if n is the smallest natural number such that $s\alpha_1\alpha_2\cdots\alpha_n = t$ for some $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathcal{R}$. $s < t$ if there exists n such that $s <_n t$.

(b) $s\Uparrow_n t$ if n is the smallest natural number such that $u\alpha_1\alpha_2\cdots\alpha_l = s$, $u\beta_1\beta_2\cdots\beta_m = t$ for some $u \in H(\mathcal{V})$, $l + m = n$ and $\alpha_1, \alpha_2, \ldots, \alpha_l\beta_1, \beta_2, \ldots, \beta_m \in \mathcal{R}$. $s\Uparrow t$ if there exists n such that $s\Uparrow_n t$.

(c) $s\Downarrow_n t$ if n is the smallest natural number such that $s\alpha_1\alpha_2\cdots\alpha_l = u$, $t\beta_1\beta_2\cdots\beta_m = u$ for some $u \in H(\mathcal{V})$, $l + m = n$ and $\alpha_1, \alpha_2, \ldots, \alpha_l\beta_1, \beta_2, \ldots, \beta_m \in \mathcal{R}$. $s\Downarrow t$ if there exists n such that $s\Downarrow_n t$.

We now define a set $\mathcal{R}$ of rewrites based on the collection $\{\theta_i\}$:

$\mathcal{R} = \{< w, X, t > |$ the binding $X \Leftarrow t$ appears in some $\theta_i\}$

Lemma 5.5.1. $\Downarrow$ is an equivalence relation on $H(\mathcal{V})$.

Proof. See Jaffar et al [7] p.504.

Let the domain D be the quotient of $H(\mathcal{V})$ by $\Downarrow$ denoted by $H(\mathcal{V})/\Downarrow$.

5.5.2. Definition of Interpretation

Let $[t]$ denote the $\Downarrow$ equivalence class of t. For each constant c in P, we assign to c the equivalence class $[c]$. For each n-ary function f in P, we assign to f the function from D^n to D defined by $([t_1], [t_2], \ldots, [t_n]) \rightarrow [f(t_1, t_2, \ldots, t_n)]$. This assignment is well defined. (See Jaffar et al [7] Proposition 3.2 in p.504.)

Let B_D be the set of all D-atoms. For $=$, we assign the identity relation on D. For other predicates, we define a subset I of B_D from the interpretations $J_0, J_1, \ldots$ associated with BR. Let I_i be a subset of B_D defined as follows.

$$I_i = \{p([t_1], [t_2], \ldots, [t_n]) \mid p(t_1, t_2, \ldots, t_n) \in J_i\}.$$

Lemma 5.5.2.1. Let $I_0, I_1, \ldots$ be as defined. Then $I_0 \subseteq I_1 \subseteq \cdots$.

Proof. Suppose a negative atom A of G_i is in A_i and A is unified with the head of "$A_0 :- A_1, A_2, \ldots, A_m$" by an m.g.u. θ_i and $\theta_i(A)$ in $\theta_i(G_i)$ is replaced with $\theta_i(A_1) \wedge \theta_i(A_2) \wedge \cdots \wedge \theta_i(A_m)$. Because $\theta_i(A_1), \theta_i(A_2), \ldots, \theta_i(A_m) \in B(\mathcal{V}_{i+1})$ are all in A_{i+1} from the definition in 5.4, $\theta_i(A_0)$ is in J_{i+1}. Then because $\theta_i(A) = \theta_i(A_0)$, for all $p(s_1, s_2, \ldots, s_n) \in A_i$, there exist $t_1, t_2, \ldots, t_n \in H(\mathcal{V}_{i+1})$ such that $[s_1] = [t_1], [s_2] = [t_2], \ldots, [s_n] = [t_n]$ and $p(t_1, t_2, \ldots, t_n) \in J_{i+1}$. Hence $I_i \subseteq I_{i+1}$.

Let I be the set of all expressions of the form $p([t_1],[t_2],\ldots,[t_n]) \in D_B$ such that $p(t_1,t_2,\ldots,t_n) \in I_i$ for some i, that is,

$$I = \{p([t_1],[t_2],\ldots,[t_n]) \mid p(t_1,t_2,\ldots,t_n) \in J_i \text{ for some } i\} = \bigcup_i I_i.$$

Lemma 5.5.2.2. I is a fixpoint of T.

Proof. The proof is similar to that of Jaffar et al [7] p.505. Because this lemma is the essential part of the whole proof, we write it down here again.

$I \supseteq T(I)$ is obvious, because $I_i \supseteq T(I_i)$ for each i. (Note that $J_i \supseteq T(J_i)$ for each i.)

$I \subseteq T(I)$ depends on the fact that BR is an NFI derivation in the fair NFI tree. Let $p([t_1],[t_2],\ldots,[t_n])$ be any element in I such that $p(t_1,t_2,\ldots,t_n)$ is in A_i ($i \geq 0$). Because BR is from a fair NFI tree, there exists a $j > 0$ such that $p(s_1,s_2,\ldots,s_n) = \theta_{i+j} \circ \cdots \circ \theta_{i+2} \circ \theta_{i+1}(p(t_1,t_2,\ldots,t_n))$ is in A_{i+j} and $p(s_1,s_2,\ldots,s_n)$ is the selected atom in G_{i+j}. Suppose C_{i+j} takes the form $p(r_1,r_2,\ldots,r_n) :- B_1,B_2,\ldots,B_m$. By the definition of T, $p([\theta_{i+j+1}(r_1)],[\theta_{i+j+1}(r_2)],\ldots,[\theta_{i+j+1}(r_n)]) \in T(I)$. Also, by the abovementioned relationship between substitutions and rewrites and the definition of $\Downarrow$ equivalence class,

$$\begin{aligned}
& p([t_1],[t_2],\ldots,[t_n]) \\
&= p([\theta_{i+j}(t_1)],[\theta_{i+j}(t_2)],\ldots,[\theta_{i+j}(t_n)]) \\
&= p([s_1],[s_2],\ldots,[s_n]) \\
&= p([\theta_{i+j+1}(s_1)],[\theta_{i+j+1}(s_2)],\ldots,[\theta_{i+j+1}(s_n)]) \\
&= p([\theta_{i+j+1}(r_1)],[\theta_{i+j+1}(r_2)],\ldots,[\theta_{i+j+1}(r_n)]),
\end{aligned}$$

so that $p([t_1],[t_2],\ldots,[t_n]) \in T(I)$. Because each J_i is generated from A_i, the above relation holds for any atom in I. Thus $I \subseteq T(I)$.

5.5.3. Invalidity in the Model

Lastly we confirm that I is indeed a model of P^* in which $G(=G_0)$ is not valid.

Lemma 5.5.3.1. I is a model of P^*.

Proof. See Jaffar et al [7] Proposition 3.1 in p.504 and p.506.

Lemma 5.5.3.2. Let $X_1,X_2,\ldots,X_n$ be all the free variables in G. Then $I \not\models \forall X_1,X_2,\ldots,X_n G$.

Proof. We write $E(X_1,X_2,\ldots,X_n)$ to show the set (possibly superset) of all free variables in expression E explicitly. Suppose $I \models \forall X_1,X_2,\ldots,X_n G(X_1,X_2,\ldots,X_n)$. Then $G([X_1],[X_2],\ldots,[X_n])$ must be valid in I. Because all negative atoms in $G([X_1],[X_2],\ldots,[X_n])$ are true in I, it must contain at least one positive atom $A([X_1],[X_2],\ldots,[X_n])$ true in I. This implies that for a sufficiently large M, $\theta_{M-1} \circ \theta_{M-2} \circ \cdots \circ \theta_1 \circ \theta_0(A(X_1,X_2,\ldots,X_n))$ is in J_M. Because the atom is a positive part of G_M, this in turn means that $G_M(Y_1,Y_2,\ldots,Y_m)$ is valid in J_M under the interpretation assigning Y as an element of $H(\mathcal{V}_M)$ to each free variable Y in G_M, which contradicts Lemma 5.4.3.

Thus the proof of Lemma 5.5 is finished.

Theorem 5.5. (Completeness for S-formulas)
Extended execution is complete for S-formulas.

Proof. First, let S be a universal formula which is not necessarily case-splitting-free. By Lemma 5.1.2, S is provable by normal extended execution if and only if all case-splitting-free universal formulas $S_1,S_2,\ldots,S_k$ obtained from G by applying case splitting as far as possible

are provable by normal extended execution. Obviously, S is a logical consequence of P^* if and only if all case-splitting-free universal formulas $S_1, S_2, \ldots, S_k$ are logical consequences of P^*. Hence a universal formula S is provable by normal extended execution if and only if S is a logical consequence of P^*.

Now let S be any S-formula and S_{new} be a universal formula obtained from S by introducing new predicates, adding new definite clauses and replacing several conjuncts in S with atoms with the new predicates, as was done in Example 2.2.5. Let P^*_{new} be the completeion of the original program P plus the added new definite clauses. Then, note that S_{new} is a logical consequence of P^*_{new} if and only if S is a logical consequence of P^*. Moreover, S_{new} is provable by normal extended execution if and only if S is provable by normal extended execution, because the new predicates do not appear in any body of the program P_{new}. Since Lemma 5.5 holds for S_{new} and P^*_{new}, extended execution is complete for S-formulas.

6. Discussion

(1) Completeness of the SLD-Resolution and the "Negation as Failure" Rule

Not only our theorem is a generalization of the completeness of the SLD-refutation [5],[1],[4] where S is a ground positive goal A or more generally a formula of the form $\exists X_1, X_2, \ldots, X_n(A_1 \wedge A_2 \wedge \cdots \wedge A_k)$, but also it is a generalization of the completeness of the "Negation as Failure" rule by Jaffar et al [3],[7],[8],[20] where S is a ground negative goal $\neg A$ or more generally a goal formula of the form $\forall X_1, X_2, \ldots, X_n\ \neg(A_1 \wedge A_2 \wedge \cdots \wedge A_k)$.

Corollary 6.1. $\exists X_1, X_2, \ldots, X_n(A_1 \wedge A_2 \wedge \cdots \wedge A_k)$ is provable by the SLD-resolution if and only if it is a logical consequence of P^*.

Proof. Let G be a goal formula of the form $\exists X_1, X_2, \ldots, X_n(A_1 \wedge A_2 \wedge \cdots \wedge A_k)$. Because there is no negative atom in G, there is no chance to apply NFI and simplification. It is obvious that there exists an SLD-refutation of G if and only if there exists a sequence of DCI proving G. (Indeed the countermodel I in our proof is J_0, which is the usual minimum Herbrand model since $\mathcal{V}_0 = \emptyset$.)

Corollary 6.2. $\forall X_1, X_2, \ldots, X_n \neg(A_1 \wedge A_2 \wedge \cdots \wedge A_k)$ is provable by the "Negation as Failure" rule if and only if it is a logical consequence of P^*.

Proof. Let G be a goal formula of the form $\forall X_1, X_2, \ldots, X_n \neg(A_1 \wedge A_2 \wedge \cdots \wedge A_k)$. Because there is no positive atom in G, there is no chance to apply DCI and simplification. It is obvious that there exists a finite failure tree of G if and only if there exists a sequence of NFI proving G. (Indeed the countermodel I in our proof is identical to the model by Jaffar et all [7], since A_0 is the set of all negative atoms in G.)

We owe very much to the work by Jaffar et al [7]. The use of fairness to overlap interpretations $I_0, I_1, I_2, \ldots$ is due to them, though we have constructed a countermodel without resorting to Knaster-Tarski's fixpoint theorem. The true difficulty existed in clarifying the properties of NFI trees in Section 5.2 and 5.3.

(2) Use of Normal Extended Execution

Our proof of the completeness implies a normal proof theorem.

Corollary 6.1. (Normal Extended Execution Theorem)
An S-formula S is provable by extended execution if and only if S is provable by normal extended execution.

Proof. Suppose an S-formula S is provable by extended execution. By Theorem 4.3, S is a logical consequence of P. By Theorem 5.5, S is provable by normal extended execution. Another direction is trivial.

In our verfication system Argus/V, application of exetended execution is controlled by many BMTP-like heuristics [9],[10],[11],[12]. In particular, we have adopted the following priority rule : Case splitting is prefered to NFI, which is prefered to DCI, which is in turn prefered to simplification. This theorem partly justifies this priority rule in our heuristic control of the inferences.

(3) Roles of the Oracle Decision Rule

Oracle decision is never applied automatically in our verfication system Argus/V. Though Argus/V has the interactive facility to apply oracle decision and direct the rules to be applied next, all theorems proved automatically so far needed no orcle decision, because all these theorems are in S-formulas. But, oracle decision is not completely redundant. Actually, it makes more first order formulas provable.

Example 6.1. Let p be a predicate defined by
$p(Z) :- p_1(Z).$
$p(Z) :- p_2(Z).$
and G be a goal formula
$((p_1(X) \supset q_1(?Z)) \wedge (p_2(X) \supset q_2(?Z))) \vee p(?Z).$
Though this S-formula is a logical consequence of P^*, we can't prove it without oracle decision. (Note that no case splitting is applicable because of the occurrences of $?Z$. If DCI is applied to $p(?Z)$, either $p_1(?Z)$ or $p_2(?Z)$ is lost.) By applying an oracle decision by $\sigma = <?Z \Leftarrow X>$, we have a new goal formula
$((p_1(X) \supset q_1(X)) \wedge (p_2(X) \supset q_2(X))) \vee p(X).$
from which we have
$(p_1(X) \supset q_1(X)) \vee p(X).$
$(p_2(X) \supset q_2(X)) \vee p(X).$
These goal formulas are provable by DCI and simplification.

One might expect that extended execution is complete for a larger class of first order formulas. The following example is a counterexample against it.

Example 6.2. Suppose we have relaxed the condition (S_2) for S-formulas to
(S_2) No undecided variable appears among the negative atoms of S.
We called this class of first formulas S-formulas in [9],[10] and conjectured the completeness. But extended execution is not complete for it even with oracle decision. Let p be a predicate as before and now G be a goal formula
$((p_1(X) \supset q_1(?Z)) \wedge (p_2(Y) \supset q_2(?Z))) \vee p(?Z).$
Though this S-formula is a logical consequence of P^*, we can't prove it even with oracle decision. (Note that both $<?Z \Leftarrow X>$ and $<?Z \Leftarrow Y>$ leaves one goal formula not provable by extended execution.)

7. Conclusions

We have shown soundness and completeness of an extention of execution devised for proving properties of Prolog programs. This work establishes a theoretical foundation of our verification system Argus/V developed between April 1984 and March 1985.

Acknowledgements

Verification system Argus/V is a subproject of the Fifth Generation Computer System (FGCS) "Intelligent Programming System". The authors would like to thank Dr.K.Fuchi (Director of ICOT) for the opportunity of doing this research and Dr.K.Furukawa (Chief of ICOT 2nd Laboratory) and Dr.T.Yokoi (Chief of ICOT 3rd Laboratory) for their advice and encouragement.

References

[1] Apt,K.R. and M.H.van Emden, "Contribution to the Theory of Logic Programming", J.ACM, Vol.29, No.3, pp.841-862, 1982.

[2] Boyer,R.S. and J.S.Moore, "A Computational Logic", Academic Press, 1979.

[3] Clark,K.L., "Negation as Failure", in Logic and Database (H.Gallaire and J.Minker Eds), pp.293-302, 1978.

[4] Clark,K.L., "Predicate Logic as a Computational Formalism", Research Monograph : 79/59, TOC, Imperial College, 1979.

[5] van Emden,M.H. and R.A.Kowalski, "The Semantics of Predicate Logic as a Programing Language", J.ACM, Vol.23, No.4, pp.733-742, 1976.

[6] Huet,G., "Confluent Reduction : Abstract Properties and Applications to Term Rewriting System", J. ACM, Vol.27, No.4, pp.797-821, 1980.

[7] Jaffar,J.,J-L.Lassez and J.Lloyd, "Completeness of the Negation as Failure Rule", Proc.of 8th International Joint Conference on Artificial Intelligence, pp.500-506,1983.

[8] Jaffar,J.,J-L.Lassez and M.J.Maher, "A Theory of Complete Logic Programs with Equality", Proc.of International Conference on Fifth Generation Computer Systems 1984, pp.175-184,1984.

[9] Kanamori,T.and H.Seki, "Verification of Prolog Programs Using An Extension of Execution", ICOT Technical Report, TR-093, 1984. Also Proc.of 3rd International Conference on Logic Programming, 1986.

[10] Kanamori,T.and H.Fujita, "Formulation of Induction Formulas in Verification of Prolog Programs", ICOT Technical Report, TR-094, 1984. Also Proc.of 8th Conference on Automated Deduction, 1986.

[11] Kanamori,T.and K.Horiuchi, "Type Inference in Prolog and Its Applications", ICOT Technical Report, TR-095, 1984. Also Proc.of 9th International Joint Conference on Artificial Intelligence, 1985.

[12] Kanmori,T.,Fujita,H.,Seki,H.,Horiuchi,K.and Maeji,M., "Argus/V : A System for Verification of Prolog Programs", ICOT Technical Report, to appear, 1986. Also Proc.of Fall Joint Computer Conference 86, 1986.

[13] Lassez,J.-L.and M.J.Maher, "Closures and Fairness in the Semantics of Programming Logic", Theoretical Computer Science, Vol.29, pp.167-184, 1984.

[14] Lloyd,J.M., "Foundations of Logic Programming", Technical Report, Department of Computer Science, University of Melbourne, TR/7, 1982.

[15] Manna,Z.and R.Waldinger, "A Deductive Approach to Program Synthesis", ACM Transaction on Programming Language and Systems, pp.90-121, 1980.

[16] Murray,N.V., "Completely Non-Clausal Theorem Proving", Artificial Intelligence, Vol.18, pp.67-85, 1982.

[17] Pereira,L.M.,F.C.N.Pereira and D.H.D.Warren, "User's Guide to DECsystem-10 Prolog", Occasional Paper 15, Dept.of Artificial Intelligence, Edinburgh, 1979.
[18] Prawitz,D., "Natural Deduction, A Proof Theoretical Study", Almqvist & Wiksell,Stockholm, 1965.
[19] Schütte,K., "Proof Theory", (translated by J.N.Crossley), Springer Verlag, 1977.
[20] Wolfram,D.A.,M.J.Maher and J-L.Lassez, "A Unified Treatment of Resolution Strategies for Logic Programs", Proc.of 2nd International Logic Programming Conference, pp.263-276, 1984.