ICOT Technical Report: TR-399

TR-399

Preservation of Stronger Equivalance in Unfold/Fold Logic Program Transformation I

by
T. Kawamura and T. Kanamori(Mitsubishi)

June, 1988

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

Preservation of Stronger Equivalence in Unfold/Fold Logic Program Transformation

Tadashi KAWAMURA Tadashi KANAMORI

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

Abstract

This paper shows that Tamaki-Sato's unfold/fold transformation of Prolog programs preserves equivalence in a stronger sense than that of the usual least Herbrand model semantics, which Tamaki and Sato originally showed. Conventionaly, the semantics of Prolog programs is defined by the least Herbrand model. However, the least Herbrand model dose not always characterize what answer substitutions are returned. This paper proves that any program obtained from an initial program by applying Tamaki-Sato's transformation returns the same answer substitutions as the initial program for any given top-level goal.

Keywords: Program Transformation, Prolog, Eqnivalence of Programs.

Contents

1. Introduction

The effectiveness of the unfold/fold rules in program transformation was first demonstrated by Burstall and Darlington [1] for functional programs. Manna and Waldinger [6] independently proposed a program synthesis method based on similar rules. Because the purpose of program transformation is to mechanically derive programs which perform the same task, one of the important properties of such program transformation rules is preservation of equivalence. An equivalence relation between programs is defined based on a semantics of programs. Different semantics can give different notions of equivalences (cf. Maher [5]). Tamaki and Sato [7] [8] [9] proposed unfold/fold rules for Prolog programs which preserves equivalence in the sense of the least Herbrand model semantics, which is the conventional semantics of Prolog programs. However, the least Herbrand model semantics dose not always characterize what answer substitutions are returned. For example, consider the following two Prolog programs P_1 and P_2.

P_1 : p(X).
q(a).
P_2 : p(a).
q(a).

Because the Herbrand universes of P_1 and P_2 are both {a}, they are equivalent in the sense of the least Herbrand model semantics. However, these two programs respond in different manners to a query

?- p(X).

P_1 returns the empty substitution <>, while P_2 returns substitution <X ⇐ a> as its answer. To make a distinction between these programs, more refined equivalence is required.

This paper shows that Tamaki-Sato's unfold/fold transformation of Prolog programs preserves equivalence in a stronger sense than that of the usual least Herbrand model semantics. First, Section 2 describes Tamaki-Sato's transformation of Prolog programs. Then, Section 3 introduces a set of pairs consisting of a given top-level goal and the answer substitution as the semantics of Prolog programs, and proves that Tamaki-Sato's transformation also preserves equivalence in the sense of this semantics.

In the following, familiarity with the basic terminologies of first order logic such as term, atom, definite clause, substitution, most general unifier(m.g.u.) and so on is assumed. A program is a set of definite clauses. The syntax of DEC-10 Prolog is followed. As syntactical variables, X, Y are used for variables, and A, B for atoms, possibly with primes and subscripts. In addition, $\theta, \eta, \sigma, \tau$ are used for substitutions, and $A\theta$ for the atom obtained from atom A by applying substitution θ.

2. Unfold/Fold Transformation of Prolog Programs

This section describes Tamaki-Sato's unfold/fold transformation following [9].

Definition Initial Program

An initial program P_0 is a program satisfying the following conditions:

(a) P_0 is divided into two disjoint sets of clauses, P_{new} and P_{old}. The predicates defined by P_{new} are called *new predicates*, while those by P_{old} are called *old predicates*.
(b) The new predicates never appear in P_{old} nor in the bodies of the clauses in P_{new}.

Example 2.1 Let $P_0 = \{C_1, C_2, C_3\}$ be an initial program, where

C_1 : ap([ ],M,M).

C_2 : ap([X|L],M,[X|N]) :- ap(L,M,N).
C_3 : insert(X,M,N) :- ap(U,V,M), ap(U,[X|V],N).
and $P_{old} = \{C_1, C_2\}$, $P_{new} = \{C_3\}$. Then 'ap' is an old predicate, while 'insert' is a new predicate.

Definition Unfolding
Let P_i be a program, C be a clause in P_i, A be an atom in the body of C, and $C_1, C_2, \ldots, C_k$ be all the clauses in P_{i-1} whose heads are unifiable with A, say by m.g.u.'s $\theta_1, \theta_2, \ldots, \theta_k$. Let C_i' be the result of applying θ_i after replacing A in C with the body of C_i. Then $P_{i+1} = (P_i - \{C\}) \cup \{C_1', C_2', \ldots, C_k'\}$. C is called the *unfolded caluse* and $C_1, C_2, \ldots, C_k$ are called the *unfolding clauses.*

Example 2.2 Let P_0 be the above program. By unfolding C_3 at atom 'ap(U,V,M)' in the body, program $P_1 = \{C_1, C_2, C_4, C_5\}$ is obtained, where
C_4 : insert(X,M,N) :- ap([ ],[X|M],N).
C_5 : insert(X,[Y|M],N) :- ap(U,V,M), ap([Y|U],[X|V],N).
By unfolding C_4 and C_5 further, program $P_2 = \{C_1, C_2, C_5, C_6\}$ and $P_3 = \{C_1, C_2, C_6, C_7\}$ are obtained, where
C_6 : insert(X,M,[X|M]).
C_7 : insert(X,[Y|M],[Y|N]) :- ap(U,V,M), ap(U,[X|V],N).

Definition Folding
Let P_i be a program, C be a clause in P_i of the form
A_0 :- $A_1, A_2, \ldots, A_n$ $(n > 0)$.
and D be a clause in P_{new} of the form
B_0 :- $B_1, B_2, \ldots, B_m$ $(m > 0)$.
Suppose that there exists a substitution θ satisfying the following conditions:
(a) $B_1\theta = A_{j_1}, B_2\theta = A_{j_2}, \ldots, B_m\theta = A_{j_m}$ where $j_1, j_2, \ldots, j_m$ are different natural numbers.
(b) For each variable appearing only in the body of D, θ substitutes a distinct variable not appearing in $\{A_0, A_1, \ldots, A_n\} - \{A_{j_1}, A_{j_2}, \ldots, A_{j_m}\}$.
(c) D is the only clause in P_{new} whose head is unifiable with $B_0\theta$.
(d) Either the predicate of C's head is an old predicate, or C is unfolded at least once in the sequence $P_0, P_1, \ldots, P_i$.
Let C' be a clause with head A_0 and body $\{B_0\}\theta \cup (\{A_1, A_2, \ldots, A_m\} - \{A_{j_1}, A_{j_2}, \ldots, A_{j_m}\})$. Then $P_{i+1} = (P_i - \{C\}) \cup \{C'\}$. C is called the *folded clause* and D is called the *folding clause*.

Example 2.3 Let P_3 be the above program. Then, by folding the body of C_7 by C_3, program $P_4 = \{C_1, C_2, C_6, C_8\}$ is obtained, where
C_8 : insert(X,[Y|M],[Y|N]) :- insert(X,M,N).

Definition Transformation Sequence
Let P_0 be an initial program, and P_{i+1} be a program obtained from P_i by applying either unfolding or folding for $i \geq 0$. The sequence of programs $P_0, P_1, \ldots, P_N$ is called a *transformation sequence starting from* P_0.

Example 2.4 The sequence P_0, P_1, P_2, P_3, P_4 in Example 2.1–2.3 is a transformation sequence starting from P_0 in Example 2.1. Note that, for query

?- insert(X,[Y],N).

these five programs return the same answer substitutions

$< N \Leftarrow [X, Y] >$,

$< N \Leftarrow [Y, X] >$.

3. Preservation of Stronger Equivalence

This section first introduces several basic notions of proof tree, then proves preservation of equivalence in the stronger sense along the same line as [9] [8].

3.1 Proof Tree

Because we need to consider what answer substitutions are returned for given top-level goals, more refined notions of proof trees are necessary so as to avoid the complications due to the strategy in nondeterministically selecting atoms to be resolved.

Definition Labelled Tree

A *labelled tree* is a finite tree whose nodes are labelled with expressions of the form "$A = B$", where A and B are unifiable atoms. The set of all the labels of labelled tree T is called the *label set* of T. The number of nodes of labelled tree T is called the *size* of T.

Definition Most General Unifier of Labelled Tree

Let T be a labelled tree and $E = \{A_1 = B_1, A_2 = B_2, \ldots, A_k = B_k\}$ be the label set of T. Then T (or E) is said to be *unifiable* when there exists a substitution σ such that $A_i\sigma$ and $B_i\sigma$ are identical for all $i = 1, 2, \ldots, k$. A substitution τ is called the *most general unifier of* T (or E) when τ is the most general substitution among such substitutions.

Definition Most General Unifier of Substitutions

Substitutions $\sigma_1, \sigma_2, \ldots, \sigma_n$ are said to be *unifiable* when there exists a substitution σ such that, for each σ_i, there exists a substituion τ_i satisfying $\sigma = \sigma_i\tau_i$. A substitution τ is called the *most general unifier of* $\sigma_1, \sigma_2, \ldots, \sigma_n$ when τ is the most general substitution among such substitutions.

Definition Proof Tree

Let P be a program, T be a labelled tree and $T_1, T_2, \ldots, T_n$ be its immediate subtrees. The labelled tree T is called a *proof tree of* atom A *with answer substitution* σ *by* P when there exists a clause C in P of the form

$$B \text{ :- } B_1, B_2, \ldots, B_n$$

such that

(a) A and B are unifiable, say by an m.g.u. θ,

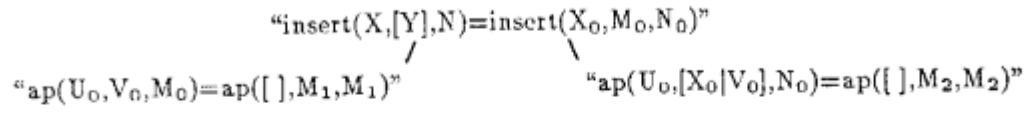
(b) the root node of T is labelled with "$A = B$,"

(c) $T_1, T_2, \ldots, T_n$ are proof trees of $B_1, B_2, \ldots, B_n$ with answer substitutions $\sigma_1, \sigma_2, \ldots, \sigma_n$ by P respectively, and

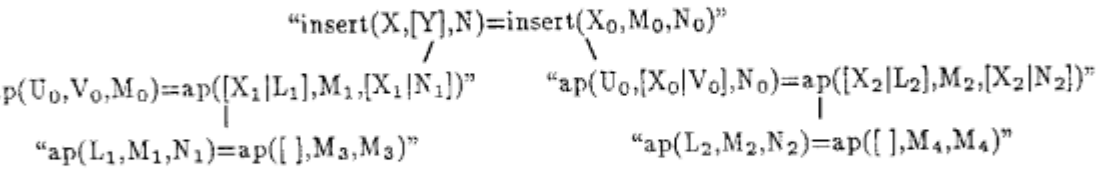
(d) σ is the restriction of an m.g.u. of $\theta, \sigma_1, \sigma_2, \ldots, \sigma_n$ to the variables in A.

The clause C is called the *clause used at the root of* T, and $T_1, T_2, \ldots, T_n$ are called the *immediate subproofs* of T. Proof trees are denoted by T and S, possibly with primes and subscripts.

Example 3.1.1 Let P_0 be the program of Example 2.1. Then proof tree T_1 of 'insert(X,[Y],N)' with answer substitution <N ⇐ [X,Y]> by P_0 is depicted below:



Proof tree T_2 of 'insert(X,[Y],N)' with answer substitution <N ⇐ [Y,X]> by P_0 is depicted below:

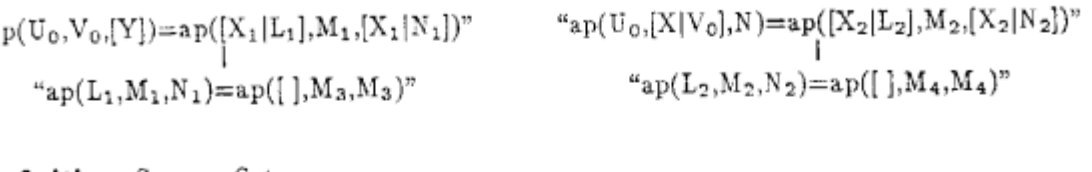


Definition Proof Forest

Let P be a program, and $T_1, T_2, \ldots, T_n$ be proof trees of atoms $A_1, A_2, \ldots, A_n$ with answer substitutions $\sigma_1, \sigma_2, \ldots, \sigma_n$ by P. A multiset $F = \{T_1, T_2, \ldots, T_n\}$ is called a *proof forest of* atom sequence $A_1, A_2, \ldots, A_n$ *with answer substitution* τ *by* P when τ is an m.g.u. of $\sigma_1, \sigma_2, \ldots, \sigma_n$. Proof trees $T_1, T_2, \ldots, T_n$ are called the *component proof trees* of F. Proof forests are denoted by F, possibly with primes and subscripts.

Example 3.1.2 Let P_0 be the program of Example 2.1. Then proof forest F_1 of atom sequence 'ap(U0,V0,[Y]), ap(U0,[X|V0],N)' with answer substitution <U0 ⇐ [ ], V0 ⇐ [Y], N ⇐ [X,Y]> by P_0 is depicted below:

Proof forest F_2 of atom sequence 'ap(U0,V0,[Y]), ap(U0,[X|V0],N)' with answer substitution <U0 ⇐ [Y], V0 ⇐ [ ], N ⇐ [Y,X]> by P_0 is depicted below:



Definition Success Set

Let P be a program. The set of all the atom-substitution pairs (A, σ) such that there exists a proof tree of A with answer substitution σ by P is called the *success set* of P, and denoted by $\mathcal{M}(P)$.

Note that the success set characterizes Prolog programs more precisely than the least Herbrand model. In the following discussion, we consider preservation of the success set in place of the least Herbrand model.

Lemma 3.1.1 If T is a proof tree of atom A with answer substitution σ, then σ is the restriction of an m.g.u. of the label set of T to the variables in A.

Proof. By induction on the structure of proof trees. Let "$A = B$" be the label of the root node of T, θ be an m.g.u. of A and B, and $T_1, T_2, \ldots, T_n$ be T's immediate subproofs of $B_1, B_2, \ldots, B_n$ with answer substitutions $\sigma_1, \sigma_2, \ldots, \sigma_n$. By the induction hypothesis, σ_i is the restriction of an m.g.u. of the label set of T_i to the variables in B_i for $i = 1, 2, \ldots, n$.

From the definition of proof tree, σ is the restriction of an m.g.u. of $\theta, \sigma_1, \sigma_2, \ldots, \sigma_n$ to the variables in A, and the variables in A never appear in the label sets of $T_1, T_2, \ldots, T_n$. Thus σ is the restriction of an m.g.u. of the label set of T to the variables in A.

Lemma 3.1.2 Let E be the label set of a proof tree T, "$A = B$" be an element of E, and θ be an m.g.u. of A and B. Then, substitution $\theta\tau$ is an m.g.u. of E if and only if τ is an m.g.u. of $(E - \{A = B\})\theta$.

Proof. Obvious.

3.2 Partial Correctness

Let P_0 and P_i be Prolog programs such that P_i is obtained from P_0 by applying the transformation rules. A transformation of Prolog program is said to be *partially correct* when $\mathcal{M}(P_0) \supseteq \mathcal{M}(P_i)$ holds. This subsection proves partial correctness, which is the easier direction of stronger equivalence.

Lemma 3.2.1 Let P_i be a program and C be a clause in P_i. Let C' be a clause obtained from C by permuting the atoms in the body of C, and P_i' be $(P_i - \{C\}) \cup \{C'\}$. Then $\mathcal{M}(P_i) = \mathcal{M}(P_i')$.

Proof. Let T be a proof tree by P_i, and T' be a proof tree obtained from T by permuting the subproofs of the atoms in the body of C according to the permutation from C to C' when clause C is used at the node. Then, this correspondence gives a one-to-one correspondence between $\mathcal{M}(P_i)$ and $\mathcal{M}(P_i')$.

This lemma implies that we can arbitrarily rearrange the atoms in the bodies of the clauses in program P_i before applying the next transformation rule while keeping the success set of P_i.

Lemma 3.2.2 Let P_i be a program in a transformation sequence, and T be a proof tree of atom $A\theta$ by program P_i. Let T' be the labelled tree obtained from T by replacing $A\theta$ in the left-hand side of the root label with A. Then T' is a proof tree of atom A by program P_i.

Proof. Obvious.

Lemma 3.2.3 Let P_i be a program in a transformation sequence, T be a proof tree of atom A with answer substitution σ by program P_i, and θ be a substitution for the variables in A such that θ and σ are unifiable. Let T' be the labelled tree obtained from T by replacing A in the left-hand side of the root label with $A\theta$. Then T' is a proof tree of atom $A\theta$ by program P_i.

Proof. Obvious.

Lemma 3.2.4 Let $P_0, P_1, \ldots, P_N$ be a transformation sequence. If $\mathcal{M}(P_i) = \mathcal{M}(P_0)$, then $\mathcal{M}(P_i) \supseteq \mathcal{M}(P_{i+1})$ for $i = 0, 1, \ldots, N-1$.

Proof. Let (A, σ) be an atom-substitution pair in $\mathcal{M}(P_{i+1})$, and T be a proof tree of A with answer substitution σ by P_{i+1}. By induction on the structure of T, we will construct a proof tree T' of A with answer substitution σ by P_i. Let C be the clause used at the root of T.

Case 1 : C is in P_i.

Let C be of the form

A_0 :- $A_1, A_2, \ldots, A_n$ $(n > 0)$

and $T_{A_1}, T_{A_2}, \ldots, T_{A_n}$ be T's immediate subproofs of $A_1, A_2, \ldots, A_n$. By the induction hypothesis, there exist proof trees $T'_{A_1}, T'_{A_2}, \ldots, T'_{A_n}$ of $A_1, A_2, \ldots, A_n$ by P_i with the same answer substitutions as $T_{A_1}, T_{A_2}, \ldots, T_{A_n}$. Let T' be a proof tree obtained by putting the root node labelled with "$A = A_0$" over $T'_{A_1}, T'_{A_2}, \ldots, T'_{A_n}$. Then, from the definition of answer substitution, σ is an answer substitution of T'. Hence T' is a proof tree of A with answer substitution σ by P_i.

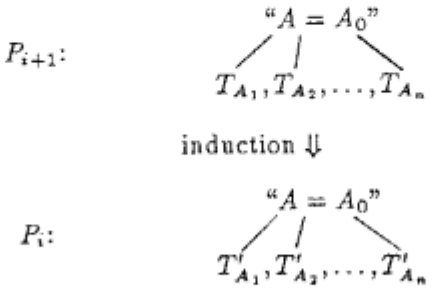


Figure 3.2.1 Construction of Proof Tree for Case 1

Case 2 : C is the result of unfolding a clause C' in P_i.

Let C' be the unfolded clause of the form

A_0 :- $A_1, A_2, \ldots, A_n$ $(n > 0)$

and D be the unfolding clause of the form

B_0 :- $B_1, B_2, \ldots, B_m$ $(m > 0)$.

From Lemma 3.2.1, without loss of generality, we can assume that A_1 and B_0 are unifiable, say by an m.g.u. θ, and C is of the form

$A_0\theta$:- $B_1\theta, \ldots, B_m\theta, A_2\theta, \ldots, A_n\theta$.

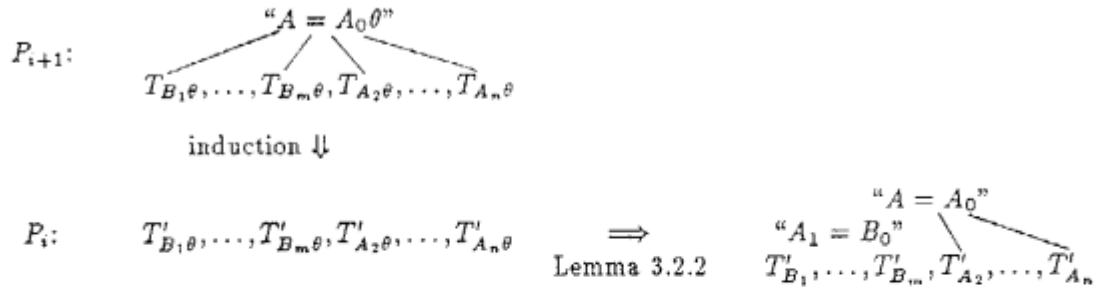


Figure 3.2.2 Construction of Proof Tree for Case 2

First, let $T_{B_1\theta}, \ldots, T_{B_m\theta}, T_{A_2\theta}, \ldots, T_{A_n\theta}$ be T's immediate subproofs of $B_1\theta, \ldots, B_m\theta$, $A_2\theta, \ldots, A_n\theta$. By the induction hypothesis, there exist proof trees $T'_{B_1\theta}, \ldots, T'_{B_m\theta}, T'_{A_2\theta}, \ldots, T'_{A_n\theta}$ of $B_1\theta, \ldots, B_m\theta, A_2\theta, \ldots, A_n\theta$ by P_i with the same answer substitutions as $T_{B_1\theta}, \ldots, T_{B_m\theta}, T_{A_2\theta}, \ldots, T_{A_n\theta}$. Let E_1 be the union of the label sets of $T'_{B_1\theta}, \ldots, T'_{B_m\theta}, T'_{A_2\theta}, \ldots, T'_{A_n\theta}$ and $\{A = A_0\theta\}$. From Lemma 3.1.1, σ is the restriction of an m.g.u. of E_1 to the variables in A.

Next, from Lemma 3.2.2, there exist proof trees $T'_{B_1}, \ldots, T'_{B_m}, T'_{A_2}, \ldots, T'_{A_n}$ of $B_1, \ldots, B_m, A_2, \ldots, A_n$ by P_i such that they are identical to $T'_{B_1\theta}, \ldots, T'_{B_m\theta}, T'_{A_2\theta}, \ldots, T'_{A_n\theta}$ except

the left-hand sides of the root labels. Let T'_{A_1} be a proof tree obtained by putting a root node labelled with "$A_1 = B_0$" over $T'_{B_1}, \ldots, T'_{B_m}$. Let T' be a proof tree obtained by putting a root node labelled with "$A = A_0$" over $T'_{A_1}, T'_{A_2}, \ldots, T'_{A_n}$, and E' be the label set of T', i.e., the union of the label sets of $T'_{B_1}, \ldots, T'_{B_m}, T'_{A_2}, \ldots, T'_{A_n}$ and $\{A = A_0, A_1 = B_0\}$. Then E_1 is identical to $(E' - \{A_1 = B_0\})\theta$. From Lemma 3.1.2, σ is the restriction of an m.g.u. of E' to the variables in A, since θ does not substitute for the variables in A. Hence, T' is a proof tree of A with answer substitution σ by P_i.

Case 3 : C is the result of folding a clause C' in P_i.

Let C' be the folded clause of the form

A_0 :- $A_1, A_2, \ldots, A_n$ $(n > 0)$

and D be the folding clause of the form

B_0 :- $B_1, B_2, \ldots, B_m$ $(m > 0)$.

From Lemma 3.2.1, without loss of generality, we can assume that $A_1, \ldots, A_m$ are instances of $B_1, \ldots, B_m$, say by an instantiation θ, and, from folding condition (b), C is of the form

A_0 :- $B_0\theta, A_{m+1}, \ldots, A_n$.

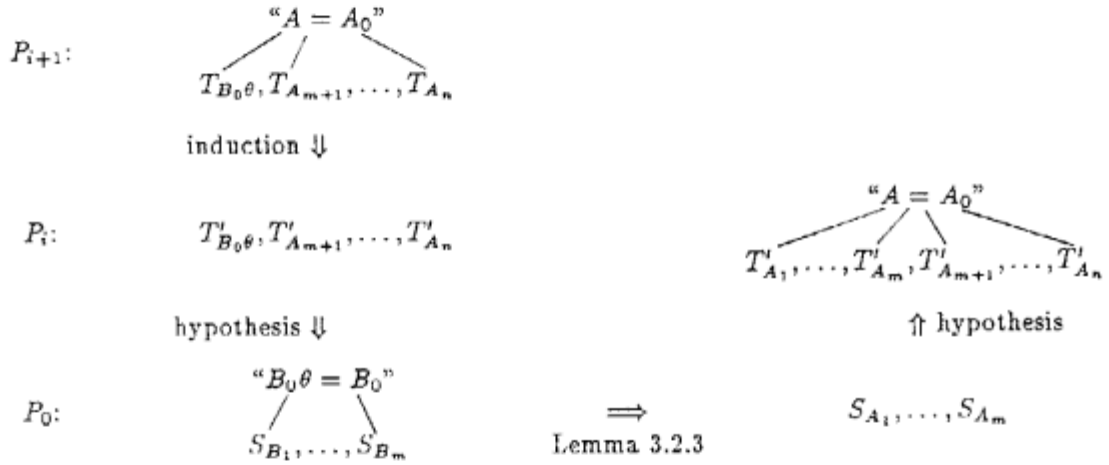


Figure 3.2.3 Construction of Proof Tree for Case 3

First, let $T_{B_0\theta}, T_{A_{m+1}}, \ldots, T_{A_n}$ be T's immediate subproofs of $B_0\theta, A_{m+1}, \ldots, A_n$. By the induction hypothesis, there exist proof trees $T'_{B_0\theta}, T'_{A_{m+1}}, \ldots, T'_{A_n}$ of $B_0\theta, A_{m+1}, \ldots, A_n$ by P_i with the same answer substitutions as $T_{B_0\theta}, T_{A_{m+1}}, \ldots, T_{A_n}$. Let E_1 be the union of the label sets of $T'_{B_0\theta}, T'_{A_{m+1}}, \ldots, T'_{A_n}$ and $\{A = A_0\}$. From Lemma 3.1.1, σ is the restriction of an m.g.u. of E_1 to the variables in A.

Second, by the hypothesis $\mathcal{M}(P_i) = \mathcal{M}(P_0)$, there exists a proof tree $S_{B_0\theta}$ of $B_0\theta$ by P_0 with the same answer substitution as $T'_{B_0\theta}$. Because the predicate of $B_0\theta$ is a new predicate, the clause used at the root of $S_{B_0\theta}$ is in P_{new}. Further, by folding condition (c), this clause should be D. Hence, the root label of $S_{B_0\theta}$ is "$B_0\theta = B_0$," and $S_{B_0\theta}$'s immediate subproofs are proof trees $S_{B_1}, \ldots, S_{B_m}$ of $B_1, \ldots, B_m$. Let E_2 be the union of the label sets of $S_{B_0\theta}, T'_{A_{m+1}}, \ldots, T'_{A_n}$ and $\{A = A_0\}$. Then, from Lemma 3.1.1, σ is the restriction of an m.g.u. of E_2 to the variables in A.

Third, from Lemma 3.2.3, there exist proof trees $S_{A_1}, \ldots, S_{A_m}$ of $A_1, \ldots, A_m$ by P_0 such that they are identical to $S_{B_1}, \ldots, S_{B_m}$ except the left-hand sides of the root labels, since $B_1\theta = A_1, \ldots, B_m\theta = A_m$ from folding condition (a). Let E_3 be the union of the label sets of $S_{A_1}, \ldots, S_{A_m}, T'_{A_{m+1}}, \ldots, T'_{A_n}$ and $\{A = A_0\}$. Then E_3 is identical to $(E_2 - \{B_0\theta = B_0\})\theta$.

From Lemma 3.1.2, σ is the restriction of an m.g.u. of E_3 to the variables in A, since θ does not substitute for the variables in A.

Last, again by the hypothesis $\mathcal{M}(P_i) = \mathcal{M}(P_0)$, there exist proof trees $T'_{A_1}, \ldots, T'_{A_m}$ of $A_1, \ldots, A_m$ by P_i with the same answer substitutions as $S_{A_1}, \ldots, S_{A_m}$. Let T' be a proof tree of A by P_i obtained by putting a root node labelled with "$A = A_0$" over $T'_{A_1}, \ldots, T'_{A_m}, T'_{A_{m+1}}, \ldots, T'_{A_n}$, and E' be the label set of T', i.e., the union of the label sets of $T'_{A_1}, \ldots, T'_{A_m}, T'_{A_{m+1}}, \ldots, T'_{A_n}$ and $\{A = A_0\}$. From Lemma 3.1.1, σ is the restriction of an m.g.u. of E'. Hence, T' is a proof tree of A with answer substitution σ by P_i.

3.3 Total Correctness

Let P_0 and P_i be Prolog programs such that P_i is obtained from P_0 by applying the transformation rules. A transformation of Prolog program is said to be *totally correct* when $\mathcal{M}(P_0) = \mathcal{M}(P_i)$ holds. This subsection proves total correctness, which is the harder direction of stronger equivalence. First, several definitions are prepared.

Definition Weight of Proof Tree

Let P_0 be the initial program in a transformation sequence, T be a proof tree of atom A by P_0, and s be the size of T. Then the *weight* of T, denoted by $w(T)$, is defined as follows:

$$w(T) = \begin{cases} s-1, & \text{if the predicate of } A \text{ is a new predicate ;} \\ s, & \text{if the predicate of } A \text{ is an old predicate.} \end{cases}$$

Example 3.3.1 Let P_0 be the initial program in Example 2.1, and T_1, T_2 be proof trees in Example 3.1.1. Then $w(T_1) = 2$ and $w(T_2) = 4$.

Definition Weight of Atom

Let P_0 be the initial program in a transformation sequence, A be an atom, and σ be a substitution. The *weight* of A with answer substitution σ, denoted by $w(A, \sigma)$, is the minimum of the weight of the proof trees of A with answer substitution σ.

Example 3.3.2 Let P_0 be the program in Example 2.1, and T_1 and T_2 be proof trees in Example 3.1.1. Then

w(insert(X,[Y],Z), <Z ⇐ [X,Y]>) = 2,

because T_1 is the minimum proof tree of 'insert(X,[Y],Z)' with answer substitution <Z ⇐ [X,Y]> by P_0. Similarly,

w(insert(X,[Y],Z), <Z ⇐ [Y,X]>) = 4.

Definition Weight of Proof Forest

Let P_0 be the initial program in a transformation sequence, F be a proof forest by P_0, and $T_1, T_2, \ldots, T_n$ be the component proof trees of F. Then the *weight* of F is defined as the sum of the $T_1, T_2, \ldots, T_n$'s weights, i.e., $w(F) = w(T_1) + w(T_2) + \cdots + w(T_n)$.

Example 3.3.3 Let P_0 be the initial program in Example 2.1, and F_1 and F_2 be proof forests in Example 3.1.2. Then $w(F_1) = 2$ and $w(F_2) = 4$.

Definition Weight of Atom Sequence

Let P_0 be the initial program in a transformation sequence, $A_1, A_2, \ldots, A_n$ be an atom sequence, and σ be a substitution. The *weight* of $A_1, A_2, \ldots, A_n$ with answer substitution

τ, denoted by $w((A_1, A_2, \ldots, A_n), \tau)$, is the minimum of the weight of the proof forests of $A_1, A_2, \ldots, A_n$ with answer substitution τ.

Example 3.3.4 Let P_0 be the program in Example 2.1, and F_1 and F_2 be proof forests in Example 3.1.2. Then

$w((\text{ap(U,V,[Y])},\text{ap(U,[X|V],N)}), \text{<U} \Leftarrow \text{[ ], V} \Leftarrow \text{[Y], N} \Leftarrow \text{[X,Y]>}) = 2,$

because F_1 is the minimum proof forest of 'ap(U,V,[Y]),ap(U,[X|V],N)' with answer substitution <U ⇐ [ ], V ⇐ [Y], N ⇐ [X,Y]> by P_0. Similarly,

$w((\text{ap(U,V,[Y])},\text{ap(U,[X|V],N)}), \text{<U} \Leftarrow \text{[Y], V} \Leftarrow \text{[ ], N} \Leftarrow \text{[Y,X]>}) = 4,$

The following notions, which are generalizations of those in [9], play an important role in the following proof.

Definition Descent Clause

Let P_i be a program in a transformation sequence starting from initial program P_0, A be an atom, σ be a substitution for the variables in A, and C be a clause of the form

$$A_0 \text{ :- } A_1, A_2, \ldots, A_n$$

whose head A_0 is unifiable with A, say by an m.g.u. η. Then clause C is called a *descent clause* of atom A with answer substitution σ in P_i when there exists a proof forest of $A_1, A_2, \ldots, A_n$ with answer substitution τ by P_0 such that

(a) the restriction of an m.g.u. of η and τ to the variables in A is σ,
(b) $w(A, \sigma) \geq w((A_1, A_2, \ldots, A_n), \tau)$, and
(c) $w(A, \sigma) > w((A_1, A_2, \ldots, A_n), \tau)$ when C satisfies folding condition (d).

Definition Weight Completeness

Let P_i be a program in a transformation sequence starting from initial program P_0. Then P_i is said to be *weight complete* if and only if, for any atom-substitution pair (A, σ) in $\mathcal{M}(P_0)$, there exists a descent clause of A with answer substitution σ in P_i.

The next three lemmas correspond to Lemma 3.2.1, 3.2.2 and 3.2.3.

Lemma 3.3.1 Let P_i be a program and C be a clause in P_i. Let C' be a clause obtained from C by permuting the atoms in the body of C, and P_i' be $(P_i - \{C\}) \cup \{C'\}$. Then P_i is wight complete if and only if P_i' is weight complete.

Proof. When $i = 0$, it is proved in the same way as the proof of Lemma 3.2.1. When $i > 0$, it is obvious.

This lemma implies that we can arbitrarily rearrange the atoms in the bodies of the clauses in program P_i before applying the next transformation rule while keeping weight completeness of P_i.

Lemma 3.3.2 Let P_0 be the initial program of a transformation sequence, and T be a proof tree of atom $A\theta$ with answer substitution σ by program P_0. Let T' be the labelled tree obtained from T by replacing $A\theta$ in the left-hand side of the root label with A. Then T' is a proof tree of atom A by program P_0, and $w(T) = w(T')$.

Proof. Obvious.

Lemma 3.3.3 Let P_0 be the initial program of a transformation sequence, T be a proof tree of atom A with answer substitution σ by program P_0, and θ be a substitution such that

θ and σ are unifiable. Let T' be the labelled tree obtained from T by replacing A in the left-hand side of the root label with $A\theta$. Then T' is a proof tree of atom $A\theta$ by program P_0, and $w(T) = w(T')$.

Proof. Obvious.

After proving one more lemma, we will start the proof of total correctness.

Lemma 3.3.4 Let P_i be a program in a transformation sequence starting from initial program P_0, and C be a clause in P_i. If C doesn't satisfy folding condition (d), all the predicates of atoms in the body of C are old predicates.

Proof. By the hypothesis, either C remains as it is during the transformation sequence from P_0 to P_i, or C is introduced by folding. For the former case, the lemma holds obviously. For the latter case, there exists a clause C' in some P_j ($j < i$), and C is the result of folding C'. Then C' satisfied folding condition (d). But, as the condition is not affected by folding, C also satisfies the condition, which contradicts the hypothesis.

Lemma 3.3.5 Let P_i be a program in a transformation sequence starting from initial program P_0. If P_i is weight complete, then $\mathcal{M}(P_i) \supseteq \mathcal{M}(P_0)$.

Proof. The proof is by induction on atom-substitution pairs ordered by the following well-founded ordering $\succ$: $(A,\sigma) \succ (B,\tau)$ if and only if

(a) $w(A,\sigma) > w(B,\tau)$, or

(b) $w(A,\sigma) = w(B,\tau)$ and the predicate of A is a new predicate and the predicate of B is an old predicate.

Let (A,σ) be an atom-substitution pair in $\mathcal{M}(P_0)$. Then there exists a descent clause C of A with answer substitution σ in P_i, where C is a clause in P_i of the form

$$A_0 \text{ :- } A_1,\ldots,A_n$$

and η is an m.g.u. of A and A_0. From the definition of descent clause,

$$w(A,\sigma) \geq w((A_1,\ldots,A_n),\tau)$$

holds, where the restriction of an m.g.u. of η and τ to the variables in A is σ. Let F be the minimum proof forest of $A_1,\ldots,A_n$ with answer substitution τ by P_0 and $T_1,\ldots,T_n$ be its component proof trees of $A_1,\ldots,A_n$ with answer substitutions $\sigma_1,\ldots,\sigma_n$. Then

$$\begin{aligned} w(A,\sigma) &\geq w((A_1,A_2,\ldots,A_n),\tau) \\ &= w(F) \\ &\geq w(T_j) \\ &= w(A_j,\sigma_j) \end{aligned}$$

holds. If

$$w(A,\sigma) > w((A_1,\ldots,A_n),\tau)$$

holds, $(A,\sigma) \succ (A_j,\sigma_j)$ holds. If

$$w(A,\sigma) = w((A_1,\ldots,A_n),\tau)$$

holds, by condition (c) of descent clause, C doesn't satisfy folding condition (d), hence, from Lemma 3.3.4, no new predicate appears in $A_1,\ldots,A_n$, which implies that $(A,\sigma) \succ (A_j,\sigma_j)$ holds. Hence, whichever holds, $(A,\sigma) \succ (A_j,\sigma_j)$ holds. Then by induction on $\succ$, (A_j,σ_j) is in $\mathcal{M}(P_i)$, and there exists a proof forest of $A_1,\ldots,A_n$ with answer substitution τ by P_i. Thus (A,σ) is in $\mathcal{M}(P_i)$.

Lemma 3.3.6 The initial program P_0 of a transformation sequence is weight complete.

Proof. Let (A, σ) be an atom-substitution pair in $\mathcal{M}(P_0)$, T be the minimum proof tree of A with answer substitution σ by P_0, and C be the clause used at the root of T of the form

$$A_0 \ :- \ A_1, A_2, \ldots, A_n.$$

Then, obviously C satisfies conditions (a),(b) of descent clause. In addition, C satisfies folding condition (d) if and only if the predicate of C's head is an old predicate. In that case, obviously condition (c) of descent clause is satisfied. Thus C is a descent clause of A with answer substitution σ.

Lemma 3.3.7 Let P_i be a program in a transformation sequence starting from initial program P_0. If P_i is weight complete, then the next program P_{i+1} in the sequence is also weight complete.

Proof. Let (A, σ) be an atom-substitution pair in $\mathcal{M}(P_0)$. Because P_i is weight complete, there exists a descent clause C of A with answer substitution σ in P_i, where C is a clause of the form

$$A_0 \ :- \ A_1, A_2, \ldots, A_n \ (n \geq 0)$$

and A and A_0 are unifiable, say by an m.g.u. η. We will show that there also exist a descent clause of A with answer substitution σ in P_{i+1}.

Case 1 : C is in P_{i+1}.
C is a descent clause of A with answer substitution σ in P_{i+1}.

Case 2 : C is unfolded.

From Lemma 3.3.1, without loss of generality, we can assume that A_1 is unfolded. Since C is a descent clause, there exists a proof forest of $A_1, A_2, \ldots, A_n$ with answer substitution τ by P_0 such that the restriction of an m.g.u. of η and τ to the variables in A is σ. Let F be the minimum proof forest among such proof forests, and $S_{A_1}, S_{A_2}, \ldots, S_{A_n}$ be F's component trees with answer substitutions $\sigma_1, \sigma_2, \ldots, \sigma_n$ by P_0. Further, since P_i is weight complete, there exists a descent clause D of A_1 with answer substitution σ_1 in P_i, where D is a clause of the form

$$B_0 \ :- \ B_1, \ldots, B_m \ (m \geq 0)$$

and A_1 and B_0 are unifiable, say by an m.g.u. θ. Let C' be the result of unfolding C using D. Then C' is of the form

$$A_0\theta \ :- \ B_1\theta, \ldots, B_m\theta, A_2\theta, \ldots, A_n\theta.$$

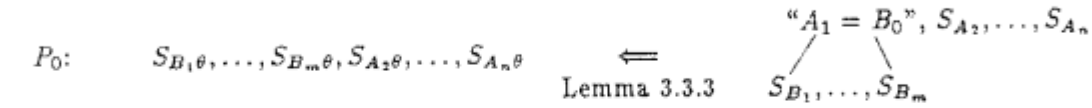


Figure 3.3.1 Construction of Proof Forest for Case 2

First, since D is a descent clause, there exists a proof forest of $B_1, \ldots, B_m$ with answer substitution τ_1 by P_0 such that the restriction of an m.g.u. of θ and τ_1 to the variables in A_1 is σ_1. Let F_1 be the minimum proof forest among such proof forests, $S_{B_1}, \ldots, S_{B_m}$ be F_1's component proof trees, and E_1 be the union of the label sets of $S_{B_1}, \ldots, S_{B_m}, S_{A_2}, \ldots, S_{A_n}$ and $\{A = A_0, A_1 = B_0\}$. From Lemma 3.1.1, σ is the restriction of an m.g.u. of E_1 to the variables in A.

Next, from Lemma 3.3.3, there exist proof trees $S_{B_1\theta}, \ldots, S_{B_m\theta}$ of $B_1\theta, \ldots, B_m\theta$ by P_0 such that they are identical to $S_{A_1}, \ldots, S_{A_m}$ except the left-hand sides of the equations in the roots labels. Similarly, from Lemma 3.3.3, there exist proof trees of $S_{A_2\theta}, \ldots, S_{A_n\theta}$ of

$A_2\theta, \ldots, A_n\theta$ by P_0 such that they are identical to $S_{A_2}, \ldots, S_{A_n}$ except the left-hand sides the root labels. Let F' be the proof forest consisting of $S_{B_1\theta}, \ldots, S_{B_m\theta}, S_{A_2\theta}, \ldots, S_{A_n\theta}$, and E' be the union of the label sets of $S_{B_1\theta}, \ldots, S_{B_m\theta}, S_{A_2\theta}, \ldots, S_{A_n\theta}$ and $\{A = A_0\theta\}$. Then E' is identical to $(E_1 - \{A_1 = B_0\})\theta$. From Lemma 3.1.2, σ is the restriction of an m.g.u. of E' to the variables in A, since θ dose not substitute for the variables in A. Let θ' be an m.g.u. of A and $A_0\theta$, and τ' be an m.g.u. of the label set of F'. Then, F' is a proof forest of $B_1\theta, \ldots, B_m\theta, A_2\theta, \ldots, A_n\theta$ with answer substitution τ' by P_0 such that σ is the restriction of an m.g.u. of θ' and τ' to the variables in A.

Last, from Lemma 3.3.3,

$$
\begin{aligned}
&w(S_{B_1}) = w(S_{B_1\theta}),\\
&\quad\vdots\\
&w(S_{B_m}) = w(S_{B_m\theta}),\\
&w(S_{A_2}) = w(S_{A_2\theta}),\\
&\quad\vdots\\
&w(S_{A_n}) = w(S_{A_n\theta})
\end{aligned}
$$

holds. Hence

$$
\begin{aligned}
w(A,\sigma) &\geq w((A_1, A_2, \ldots, A_n), \tau)\\
&= w(F)\\
&= w(S_{A_1}) + w(S_{A_2}) + \cdots + w(S_{A_n})\\
&= w(A_1, \sigma_1) + w(S_{A_2}) + \cdots + w(S_{A_n})\\
&\geq w((B_1, \ldots, B_m), \tau_1) + w(S_{A_2}) + \cdots + w(S_{A_n})\\
&= w(F_1) + w(S_{A_2}) + \cdots + w(S_{A_n})\\
&= w(S_{B_1}) + \cdots + w(S_{B_m}) + w(S_{A_2}) + \cdots + w(S_{A_n})\\
&= w(S_{B_1\theta}) + \cdots + w(S_{B_m\theta}) + w(S_{A_2\theta}) + \cdots + w(S_{A_n\theta})\\
&\geq w((B_1\theta, \ldots, B_m\theta, A_2\theta, \ldots, A_n\theta), \tau').
\end{aligned}
$$

holds. Further, if the predicate of B_0 is an old prdicate, D satisfies folding condition (d), and if not, C does from Lemma 3.3.4. Then, from condition (c) of descent clause, either

$$w(A,\sigma) > w((A_1 \ldots A_n), \tau)$$

or

$$w(A_1, \sigma_1) > w((B_1, \ldots, B_m), \tau_1)$$

holds. Whichever holds,

$$w(A,\sigma) > w((B_1\theta, \ldots B_m\theta, A_2\theta, \ldots, A_n\theta), \tau')$$

holds. Thus, C' is a descent clause of A with answer substitution σ in P_{i+1}.

Case 3 : C is folded.

Let D be the folding clause of the form

$$B_0 \text{ :- } B_1, \ldots, B_m \ (m > 0)$$

and C' be the result of folding. From Lemma 3.3.1, without loss of generality, we can assume that $A_1, \ldots, A_m$ are instances of $B_1, \ldots, B_m$, say by an instantiation θ, and from folding condition (b), C' is of the form

$$A_0 \text{ :- } B_0\theta, A_{m+1}, \ldots, A_n.$$

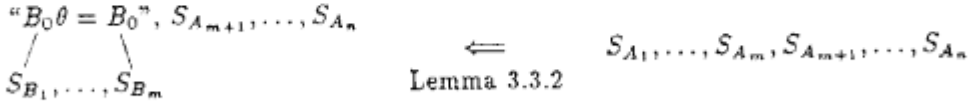


Figure 3.3.2 Construction of Proof Forest for Case 3

First, since C is a descent clause, there exists a proof forest of $A_1, \ldots, A_n$ with answer substitutions τ by P_0 such that the restriction of an m.g.u. of η and τ to the variables in A is σ. Let F be the minimum proof forest among such proof forests, $S_{A_1}, \ldots, S_{A_n}$ be F's component proof trees of $A_1, \ldots, A_n$ with answer substitutions $\sigma_1, \ldots, \sigma_n$ by P_0, and E_1 be the union of the label sets of $S_{A_1}, \ldots, S_{A_n}$ and $\{A = A_0\}$. From Lemma 3.1.1, σ is the restriction of an m.g.u. of E_1 to the variables in A.

Next, from Lemma 3.3.2, there exist proof trees $S_{B_1}, \ldots, S_{B_m}$ of $B_1, \ldots, B_m$ by P_0 such that they are identical to $S_{A_1}, \ldots, S_{A_m}$ except the left-hand sides of the root labels, since $B_1\theta = A_1, \ldots, B_m\theta = A_m$ from folding condition (a). Let $S_{B_0\theta}$ be a proof tree obtained by putting a root node labelled with "$B_0\theta = B_0$" over $S_{B_1}, \ldots, S_{B_m}$, F' be the proof forest consisting of $S_{B_0\theta}, S_{A_{m+1}}, \ldots, S_{A_n}$, and E' be the union of the label sets of $S_{B_0\theta}, S_{A_{m+1}}, \ldots, S_{A_n}$ and $\{A = A_0\}$. Then E_1 is identical to $(E' - \{B_0\theta = B_0\})\theta$. From Lemma 3.1.2, σ is the restriction of an m.g.u. of E' to the variables in A, since θ does not substitute for the variables in A. Let τ' be an m.g.u. of the union of the label sets of $S_{B_0\eta}, S_{A_{m+1}}, \ldots, S_{A_n}$. Then, F' is a proof forest of $B_0\theta, A_{m+1}, \ldots, A_n$ with answer substitution τ' by P_0 such that the restriction of an m.g.u. of θ and τ' to the variables in A is σ.

Last, let σ_0 be the answer substitution of $S_{B_0\theta}$. Because the predicate of $B_0\theta$ is a new predicate, the clause used at the root of any proof tree of $B_0\theta$ by P_0 is in P_{new}. Further, by folding condition (c), this clause should be D. Hence, the root label of such a proof tree is "$B_0\theta = B_0$," and immediate subproofs of such a proof tree are proof trees $S_{B_1}, \ldots, S_{B_m}$ of $B_1, \ldots, B_m$. Since the weight $w(B_0\theta, \sigma_0)$ is the minimum size of such proof trees and the predicate of B_0 is a new predicate,

$$w(B_0\theta, \sigma_0) \leq w(S_{B_1}) + \cdots + w(S_{B_m})$$

holds. In addition, by folding condition (d) and the definition of descent clause,

$$w(A, \sigma) > w((A_1, \ldots, A_n), \tau)$$

and from Lemma 3.3.2,

$$\begin{aligned} w(S_{B_1}) &= w(S_{A_1}), \\ &\vdots \\ w(S_{B_m}) &= w(S_{A_m}) \end{aligned}$$

hold. Hence

$$\begin{aligned} w(A, \sigma) &> w((A_1, \ldots, A_n), \tau) \\ &= w(S_{A_1}) + \cdots + w(S_{A_m}) + w(S_{A_{m+1}}) + \cdots + w(S_{A_n}) \\ &= w(S_{B_1}) + \cdots + w(S_{B_m}) + w(S_{A_{m+1}}) + \cdots + w(S_{A_n}) \\ &\geq w(B_0\theta, \sigma_0) + w(A_{m+1}, \sigma_{m+1}) + \cdots + w(A_n, \sigma_n) \\ &\geq w((B_0\theta, A_{m+1}, \ldots, A_n), \tau') \end{aligned}$$

holds. Thus, C' is a descent clause of A with answer substitution σ in P_{i+1}.

Theorem 3.3.8 Preservation of Success Set

The success set of any program in a transformation sequence starting from initial program P_0 is identical to that of P_0.

Proof. From Lemma 3.3.6 and 3.3.7, P_{i+1} is weight complete, and then from Lemma 3.3.5, $\mathcal{M}(P_{i+1}) \supseteq \mathcal{M}(P_0)$ for $i = 0, 1, \ldots, N-1$. Further, from Lemma 3.2.4, $\mathcal{M}(P_{i+1}) = \mathcal{M}(P_0)$ holds for $i = 0, 1, \ldots, N-1$.

The original result by Tamaki and Sato [7] [9] can be derived as a corollary.

Corollary 3.3.9 Preservation of Least Herbrand Model

The least Herbrand model of any program in a transformation sequence starting from initial program P_0 is identical to that of P_0.

Proof. Let P be a program, $M(P)$ be the set of all the ground atoms $A\sigma$ such that atom-substitution pair (A, σ) is included in $\mathcal{M}(P)$. Then $M(P)$ is the least Herbrand model of P, and from Theorem 3.3.8, $\mathcal{M}(P)$ is preserved. Thus, the least Herbrand model is preserved.

4. Discussion

Preservation of success set widens the safe use of the Prolog programs obtained by Tamaki-Sato's transformation, which is not validated by preservation of least Herbrand model. For example, consider the 'setof' predicate of DEC-10 Prolog. A call 'setof(X,P,S)' means "S is the set of all instances of X such that P succeeds". Two programs which are equivalent in the sense of the least Herbrand model semantics do not necessarily behave in the same way to the 'setof' call. For example, consider again two programs P_1 and P_2 we have shown in Section 1. Although these two programs are equivalent in the sense of the least Herbrand model semantics, to a query

?- setof(X,p(X),Y),

P_2 succeeds with answer substitution <X ⇐ a, Y ⇐ [a]>, while P_1 fails. However, when the success sets of programs are identical, they behave in the same way to any 'setof' call if the call stops. (Note that the success sets of P_1 and P_2 are not identical.) Hence, we can safely use a predicate as an argument of 'setof' when the program for the predicate is obtained by Tamaki-Sato's transformation.

In this paper, we have not mentioned the goal replacement rule, which Tamaki and Sato adopted as one of the basic transformation rules [8] [9]. We expect that, in application of the goal replacement rule, slightly stronger conditions than those by Tamaki and Sato would guarantee the equivalence-preservation in our sense.

5. Conclusions

We have shown that Tamaki-Sato's unfold/fold transformation of Prolog programs preserves equivalence in a stronger sense than that of the usual least Herbrand model semantics, which Tamaki and Sato originally showed. That is, any program obtained from an initial program by applying Tamaki-Sato's transformation returns the same answer substitutions as the initial program for any given top-level goal.

Acknowledgements

This work is based on the result by Tamaki and Sato [7] [8] [9]. The authors would like to express deep gratitude to Mr. H. Tamaki (Ibaraki University) and Dr. T. Sato (Electrotechnical Laboratory) for their perspicuous and stimulative works.

This research was done as a part of the Fifth Generation Computer Systems project of Japan [2] [3] [4]. We would like to thank Dr. K. Fuchi (Director of ICOT) for the opportunity of doing this research, and Dr. K. Furukawa (Vice Director of ICOT), Dr. R. Hasegawa (Chief of ICOT 1st Laboratory) and Dr. H. Ito (Chief of ICOT 3rd Laboratory) for their advice and encouragement.

References

[1] Burstall, R.M and J.Darlington, "A Transformation System for Developing Recursive Programs", J.ACM, Vol.24, No.1, pp.44–67, 1977.

[2] Kanamroi, T and K.Horiuchi, "Construction of Logic Programs Based on Generalized Unfold/Fold Rules", Proc. of 4th International Conference on Logic Programming, pp. 744–768, Melbourne, May 1987. Also a preliminary version appeared as ICOT Technical Report TR-177, 1986.

[3] Kanamroi, T and H.Fujita, "Unfold/Fold Logic Program Transformation with Counters", Presented at U.S-Japan Workshop on Logic of Programs, Honolulu, May 1987. Also a preliminary version appeared as ICOT Technical Report TR-179, 1986.

[4] Kanamroi, T and M.Maeji, "Derivation of Logic Programs from Implicit Definition", ICOT Technical Report TR-178, 1986.

[5] Maher, M.J., "Equivalences of Logic Programs", Proc. of 3rd Internatinal Conference on Logic Programming, London, July 1986.

[6] Manna, Z and R.Waldinger, "Synthesis : Dreams ⇒ Programs", IEEE Trans. on Software Engineering, Vol.5, No.4, pp.294–328, 1979.

[7] Tamaki, H and T.Sato, "Unfold/Fold Transformation of Logic Programs", Proc. of 2nd International Logic Programming Conference, pp.127–138, Uppsala, July 1984.

[8] Tamaki, H and T.Sato, "A Generalized Correctness Proof of the Unfold/Fold Logic Program Transformation", Department of Information Science TR86-04, Ibaraki University, 1986.

[9] Tamaki, H, "Program Transformation in Logic Programming", (in Japanese,) in "Program Transformation", eds. K.Fuchi, K.Furukawa and F.Mizoguchi, Kyoritsu Pub. Co., pp.39–62, 1987.