

PRESERVATION OF STRONGER EQUIVALENCE IN UNFOLD/FOLD LOGIC PROGRAM TRANSFORMATION

Tadashi KAWAMURA and Tadashi KANAMORI

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

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. Conventionally, the semantics of Prolog programs is defined by the least Herbrand model. However, the least Herbrand model does 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.

1. Introduction

The effectiveness of the unfold/fold rules in program transformation was first demonstrated in [1] for functional programs. Manna and Waldinger [8] 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 the semantics of the programs. Different semantics can give different notions of equivalences (cf. [7]). Tamaki and Sato [9, 10, 11] proposed unfold/fold rules for Prolog programs that preserve equivalence in the sense of the least Herbrand model semantics, which is the conventional semantics of Prolog programs. However, the least Herbrand model semantics does 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 $\langle \rangle$, while P_2 returns substitution $\langle X \leftarrow a \rangle$ 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. 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 (mgu) and so on is assumed. A program is a set of definite clauses. The syntax of DEC-10 Prolog is followed. As syntactical variables, A, B are used for atoms, possibly with primes and subscripts. In addition, $\theta, \eta, \sigma, \tau, \lambda$ 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 [9].

Definition 2.1 (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 2.2. Let $P_0 = \{C_1, C_2, C_3\}$ be an initial program, where

$$C_1: \text{ap}([], M, M).$$

$$C_2: \text{ap}([X|L], M, [X|N]) :- \text{ap}(L, M, N).$$

$$C_3: \text{insert}(X, M, N) :- \text{ap}(U, V, M), \text{ap}(U, [X|V], N).$$

and $P_{\text{old}} = \{C_1, C_2\}$, $P_{\text{new}} = \{C_3\}$. Then "ap" is an old predicate, while "insert" is a new predicate.

Definition 2.3 (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, \dots, C_k be all the clauses in P_i whose heads are unifiable with A , say by mgus $\theta_1, \theta_2, \dots, \theta_k$. Let C'_i be the result of applying θ_i after replacing A in C with the body of C_i . Then $P_{i+1} = (P_i - \{C\}) \cup \{C'_1, C'_2, \dots, C'_k\}$. C is called the *unfolded clause* and C_1, C_2, \dots, C_k are called the *unfolding clauses*.

Example 2.4. 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

$$C_4: \text{insert}(X, M, N) :- \text{ap}([], [X|M], N).$$

$$C_5: \text{insert}(X, [Y|M], N) :- \text{ap}(U, V, M), \text{ap}([Y|U], [X|V], N).$$

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

$$C_6: \text{insert}(X, M, [X|M]).$$

$$C_7: \text{insert}(X, [Y|M], [Y|N]) :- \text{ap}(U, V, M), \text{ap}(U, [X|V], N).$$

Definition 2.5 (Folding). Let P_i be a program, C be a clause in P_i of the form

$$A_0 :- A_1, A_2, \dots, A_n \quad (n > 0).$$

and D be a clause in P_{new} of the form.

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

Suppose that there exists a substitution θ satisfying the following conditions:

(a) $B_1\theta = A_{j_1}, B_2\theta = A_{j_2}, \dots, B_m\theta = A_{j_m}$, where j_1, j_2, \dots, 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 $\{B_0\theta, A_0, A_1, \dots, A_n\} - \{A_{j_1}, A_{j_2}, \dots, A_{j_m}\}$.

(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, \dots, P_i .

Let C' be a clause with head A_0 and body $\{B_0\theta\} \cup (\{A_1, A_2, \dots, A_m\} - \{A_{j_1}, A_{j_2}, \dots, A_{j_m}\})$. Then $P_{i+1} = (P_i - \{C\}) \cup \{C'\}$. C is called the *folded clause* and D is called the *folding clause*.

Example 2.6. Let P_3 be the above program. Then, by folding the body of C_7 by C_6 , program $P_4 = \{C_1, C_2, C_6, C_8\}$ is obtained, where

$$C_8: \text{insert}(X, [Y|M], [Y|N]) :- \text{insert}(X, M, N).$$

Definition 2.7 (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, \dots, P_N is called a *transformation sequence starting from P_0* .

Example 2.8. The sequence P_0, P_1, P_2, P_3, P_4 in Examples 2.2, 2.4 and 2.6 is a transformation sequence starting from P_0 in Example 2.2. Note that, for query $?\text{-insert}(X, [Y], N)$, these five programs return the same answer substitutions

$$\langle N \leftarrow [X, Y] \rangle, \quad \langle N \leftarrow [Y, X] \rangle.$$

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, 11].

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 to avoid the complications due to the strategy in nondeterministically selecting atoms to be resolved.

Definition 3.1 (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 3.2 (Most general unifier of a labelled tree). Let T be a labelled tree and $E = \{A_1 = B_1, A_2 = B_2, \dots, A_k = B_k\}$ 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 $i = 1, 2, \dots, k$. A substitution τ is called the *most general unifier* of T (or E) when τ is the most general substitution among such substitutions.

Definition 3.3 (Most general unifier of substitutions). Substitutions $\sigma_1, \sigma_2, \dots, \sigma_n$ are said to be *unifiable* when there exists a substitution σ such that, for each σ_i there exists a substitution τ_i satisfying $\sigma = \sigma_i\tau_i$. A substitution τ is called the *most general unifier* of $\sigma_1, \sigma_2, \dots, \sigma_n$ when τ is the most general substitution among such substitutions.

Definition 3.4 (Proof tree). Let P be a program, T be a labelled tree and T_1, T_2, \dots, 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 $B :- B_1, B_2, \dots, B_n$ such that

- (a) A and B are unifiable, say by an mgu θ ,
- (b) the root node of T is labelled with “ $A = B$ ”,
- (c) T_1, T_2, \dots, T_n are proof trees of B_1, B_2, \dots, B_n with answer substitutions $\sigma_1, \sigma_2, \dots, \sigma_n$ by P respectively, and
- (d) σ is the restriction of an mgu of $\theta, \sigma_1, \sigma_2, \dots, \sigma_n$ to the variables in A .

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

Example 3.5. Let P_0 be the program of Example 2.2. Then proof tree T_1 of “insert($X, [Y], N$)” with answer substitution $\langle N \leftarrow [X, Y] \rangle$ by P_0 is depicted below:

$$\begin{array}{c}
 \text{“insert}(X, [Y], N) = \text{insert}(X_0, M_0, N_0)\text{”} \\
 \swarrow \quad \searrow \\
 \text{“ap}(U_0, V_0, M_0) = \text{ap}([\], M_1, M_1)\text{”} \quad \text{“ap}(U_0, [X_0 | V_0], N_0) = \text{ap}([\], M_2, M_2)\text{”}
 \end{array}$$

Proof tree T_2 of “insert($X, [Y], N$)” with answer substitution $\langle N \leftarrow [Y, X] \rangle$ by P_0

is depicted below:

$$\begin{array}{ccc}
 & \text{"insert}(X, [Y], N) = \text{insert}(X_0, M_0, N_0)\text{"} & \\
 & \swarrow \quad \searrow & \\
 \text{"ap}(U_0, V_0, M_0) = \text{ap}([X_1|L_1], M_1, [X_1|N_1])\text{"} & & \text{"ap}(U_0, [X_0|V_0], N_0) = \text{ap}([X_2|L_2], M_2, [X_2|N_2])\text{"} \\
 \downarrow & & \downarrow \\
 \text{"ap}(L_1, M_1, N_1) = \text{ap}([], M_3, M_3)\text{"} & & \text{"ap}(L_2, M_2, N_2) = \text{ap}([], M_4, M_4)\text{"}
 \end{array}$$

Definition 3.6 (Proof forest). Let P be a program, and T_1, T_2, \dots, T_n be proof trees of atoms A_1, A_2, \dots, A_n with answer substitutions $\sigma_1, \sigma_2, \dots, \sigma_n$ by P . A multiset $F = \{T_1, T_2, \dots, T_n\}$ is called a *proof forest of atom sequence* A_1, A_2, \dots, A_n with answer substitution τ by P when τ is an mgu of $\sigma_1, \sigma_2, \dots, \sigma_n$. Proof trees T_1, T_2, \dots, T_n are called the *component proof trees* of F . Proof forests are denoted by F , possibly with primes and subscripts.

Example 3.7. Let P_0 be the program of Example 2.2. Then proof forest F_1 of atom sequence “ap($U_0, V_0, [Y]$), ap($U_0, [X|V_0], N$)” with answer substitution $\langle U_0 \leftarrow [], V_0 \leftarrow [Y], N \leftarrow [X, Y] \rangle$ by P_0 is depicted below:

$$\text{"ap}(U_0, V_0, [Y]) = \text{ap}([], M_1, M_1)\text{"} \quad \text{"ap}(U_0, [X|V_0], N) = \text{ap}([], M_2, M_2)\text{"}$$

Proof forest F_2 of atom sequence “ap($U_0, V_0, [Y]$), ap($U_0, [X|V_0], N$)” with answer substitution $\langle U_0 \leftarrow [Y], V_0 \leftarrow [], N \leftarrow [Y, X] \rangle$ by P_0 is depicted below:

$$\begin{array}{ccc}
 \text{"ap}(U_0, V_0, [Y]) = \text{ap}([X_1|L_1], M_1, [X_1|N_1])\text{"} & & \text{"ap}(U_0, [X|V_0], N) = \text{ap}([X_2|L_2], M_2, [X_2|N_2])\text{"} \\
 \downarrow & & \downarrow \\
 \text{"ap}(L_1, M_1, N_1) = \text{ap}([], M_3, M_3)\text{"} & & \text{"ap}(L_2, M_2, N_2) = \text{ap}([], M_4, M_4)\text{"}
 \end{array}$$

Definition 3.8 (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 3.9. T is a proof tree of atom A with answer substitution σ if and only if σ is the restriction of an mgu of the label set of T to the variables in A .

Proof. Let “ $A = A_0$ ” be the label of the root node of T , and θ be an mgu of A and B .

The “if” part is proved by induction on the structure of proof trees as follows. Let τ be an mgu of the label set of T , and T_1, T_2, \dots, T_n be T ’s immediate subproofs of A_1, A_2, \dots, A_n . Let σ_i be the restriction of an mgu of T_i to the variables in A_i , for $i = 1, 2, \dots, n$. By the induction hypothesis, T_i is a proof tree of A_i with answer substitution σ_i . Let σ_0 be the restriction of τ to the variables in A, A_0, A_1, \dots, A_n . From the definition of mgu of labelled tree, σ_0 should satisfy $\sigma_0 = \theta\lambda$ and $\sigma_0 = \sigma_i\lambda_i$.

for some substitutions λ and λ_i for $i = 1, 2, \dots, n$. Further, σ_0 is the most general substitution among such substitutions, since τ is most general. Thus, from the definition of mgu of substitutions, σ_0 is an mgu of $\theta, \sigma_1, \sigma_2, \dots, \sigma_n$. Let σ be the restriction of τ to the variables in A . Then σ is the restriction of an mgu of $\theta, \sigma_1, \sigma_2, \dots, \sigma_n$, since σ is also the restriction of σ_0 to the variables in A . Hence, T is a proof tree of A with answer substitution σ .

The “only if” part is also proved by induction on the structure of proof trees as follows. Let T_1, T_2, \dots, T_n be T 's immediate subproofs of A_1, A_2, \dots, A_n with answer substitutions $\sigma_1, \sigma_2, \dots, \sigma_n$. By the induction hypothesis, σ_i is the restriction of an mgu of the label set of T_i to the variables in A_i for $i = 1, 2, \dots, n$. From the definition of proof tree, σ is the restriction of an mgu of $\theta, \sigma_1, \sigma_2, \dots, \sigma_n$ to the variables in A , and the variables in A never appear in the label sets of T_1, T_2, \dots, T_n . Thus, σ is the restriction of an mgu of the label set of T to the variables in A . \square

Lemma 3.10. *Let E be the label set of a proof tree T , “ $A = B$ ” be an element of E , and θ be an mgu of A and B . Then, substitution $\theta\tau$ is an mgu of E if and only if τ is an mgu of $(E - \{A = B\})\theta$.*

Proof. Obvious. \square

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.

Lemma 3.11. *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 $\mathcal{M}(P'_i)$. \square

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 3.12. *Let P_i be a program, and T be a proof tree of atom $A\theta$ by 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 A by P_i .*

Proof. Obvious. \square

Lemma 3.13. *Let P_i be a program, T be a proof tree of atom A with answer substitution σ by 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 $A\theta$ by P_i .*

Proof. Obvious. \square

Lemma 3.14. *Let P_0, P_1, \dots, 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 $i = 0, 1, \dots, N - 1$.*

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 $A_0 :- A_1, A_2, \dots, A_n$ ($n > 0$) and $T_{A_1}, T_{A_2}, \dots, T_{A_n}$ be T 's immediate subproofs of A_1, A_2, \dots, A_n . By the induction hypothesis, there exist proof trees $T'_{A_1}, T'_{A_2}, \dots, T'_{A_n}$ of A_1, A_2, \dots, A_n by P_i with the same answer substitutions as $T_{A_1}, T_{A_2}, \dots, T_{A_n}$. Let T' be a proof tree obtained by putting the root node labelled with " $A = A_0$ " over $T'_{A_1}, T'_{A_2}, \dots, T'_{A_n}$. 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 (see Fig. 1).

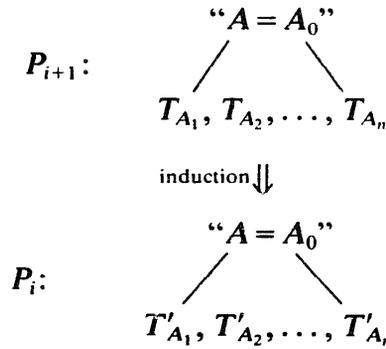


Fig. 1. Construction of proof tree for Case 1.

Case 2: C is the result of unfolding a clause C' in P_i . Let C' be the unfolded clause of the form $A_0 :- A_1, A_2, \dots, A_n$ ($n > 0$) and D be the unfolding clause of the form $B_0 :- B_1, B_2, \dots, B_m$ ($m \geq 0$). From Lemma 3.11, without loss of generality, we can assume that A_1 and B_0 are unifiable, say by an mgu θ , and C is of the form

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

First, let $T_{B_1\theta}, \dots, T_{B_m\theta}, T_{A_2\theta}, \dots, T_{A_n\theta}$ be T 's immediate subproofs of $B_1\theta, \dots, B_m\theta, A_2\theta, \dots, A_n\theta$. By the induction hypothesis, there exist proof trees $T'_{B_1\theta}, \dots, T'_{B_m\theta}, T'_{A_2\theta}, \dots, T'_{A_n\theta}$ of $B_1\theta, \dots, B_m\theta, A_2\theta, \dots, A_n\theta$ by P_i with the same answer substitutions as $T_{B_1\theta}, \dots, T_{B_m\theta}, T_{A_2\theta}, \dots, T_{A_n\theta}$. Let E_1 be the union of the

label set of $T'_{B_1\theta}, \dots, T'_{B_m\theta}, T'_{A_2\theta}, \dots, T'_{A_n\theta}$ and $\{A = A_0\theta\}$. From Lemma 3.9, σ is the restriction of an mgu of E_1 to the variables in A .

Next, from Lemma 3.12, there exist proof trees $T'_{B_1}, \dots, T'_{B_m}, T'_{A_2}, \dots, T'_{A_n}$ of $B_1, \dots, B_m, A_2, \dots, A_n$ by P_i such that they are identical to $T'_{B_1\theta}, \dots, T'_{B_m\theta}, T'_{A_2\theta}, \dots, T'_{A_n\theta}$ except for 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 $T'_{B_1}, \dots, T'_{B_m}$. Let T' be a proof tree obtained by putting a root node labelled with " $A = A_0$ " over $T'_{A_1}, T'_{A_2}, \dots, T'_{A_n}$, and E' be the label set of T' , i.e., the union of the label sets of $T'_{B_1}, \dots, T'_{B_m}, T'_{A_2}, \dots, T'_{A_n}$ and $\{A = A_0, A_1 = B_0\}$. Then E_1 is identical to $(E' - \{A_1 = B_0\})\theta$. From Lemma 3.10, σ is the restriction of an mgu of E' to the variables in A , since θ does not substitute any term for the variables in A . Then, from Lemma 3.9, T' is a proof tree of A with answer substitution σ by P_i (see Fig. 2).

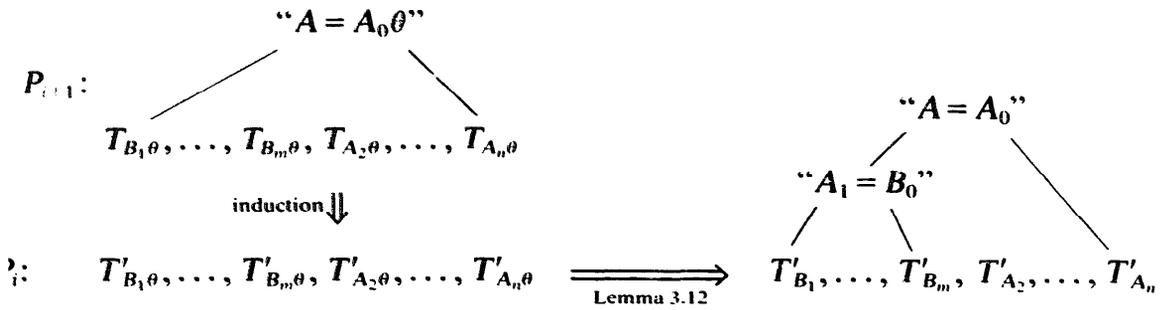


Fig. 2. Construction of proof tree for Case 2.

Case 3: C is the result of folding a clause C' in P_i . Let C' be the folded clause of the form $A_0 :- A_1, A_2, \dots, A_n$ ($n > 0$) and D be the folding clause of the form $B_0 :- B_1, B_2, \dots, B_m$ ($m > 0$). From Lemma 3.11, without loss of generality, we can assume that A_1, \dots, A_m are instances of B_1, \dots, B_m , say by an instantiation θ , and C is of the form

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

First, let $T_{B_0\theta}, T_{A_{m+1}}, \dots, T_{A_n}$ be T 's immediate subproofs of $B_0\theta, A_{m+1}, \dots, A_n$. By the induction hypothesis, there exist proof trees $T'_{B_0\theta}, T'_{A_{m+1}}, \dots, T'_{A_n}$ of $B_0\theta, A_{m+1}, \dots, A_n$ by P_i with the same answer substitutions as $T_{B_0\theta}, T_{A_{m+1}}, \dots, T_{A_n}$. Let E_1 be the union of the label sets of $T'_{B_0\theta}, T'_{A_{m+1}}, \dots, T'_{A_n}$ and $\{A = A_0\}$. From Lemma 3.9, σ is the restriction of an mgu 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}, \dots, S_{B_m} of B_1, \dots, B_m . Let E_2 be the union of the label sets of $S_{B_0\theta}, T'_{A_{m+1}}, \dots, T'_{A_n}$ and $\{A = A_0\}$. Then, from Lemma 3.9, σ is the restriction of an mgu of E_2 to the variables in A .

Third, from Lemma 3.13, there exist proof trees S_{A_1}, \dots, S_{A_m} of A_1, \dots, A_m by P_0 such that they are identical to S_{B_1}, \dots, S_{B_m} except for the left-hand sides of the root labels, since $B_1\theta = A_1, \dots, B_m\theta = A_m$ from folding condition (a). Let E_3 be the union of the label sets of $S_{A_1}, \dots, S_{A_m}, T'_{A_{m+1}}, \dots, T'_{A_n}$ and $\{A = A_0\}$. Then, from folding condition (b), E_3 is identical to $(E_2 - \{B_0\theta = B_0\})\theta$. From Lemma 3.10, σ is the restriction of an mgu of E_3 to the variables in A , since θ does not substitute any term for the variables in A . (Let η be an mgu of A and A_0 , and $\tau_1, \tau_2, \dots, \tau_n$ be the answer substitutions of $S_{A_1}, S_{A_2}, \dots, S_{A_m}, T'_{A_{m+1}}, \dots, T'_{A_n}$. Then σ is the restriction of an mgu of $\eta, \tau_1, \tau_2, \dots, \tau_n$ to the variables in A . This is proved similarly to Lemma 3.9.)

Last, again by the hypothesis $\mathcal{M}(P_i) = \mathcal{M}(P_0)$, there exist proof trees $T'_{A_1}, \dots, T'_{A_m}$ of A_1, \dots, A_m by P_i with the same answer substitutions as S_{A_1}, \dots, S_{A_m} . Let T' be a proof tree obtained by putting a root node labelled with “ $A = A_0$ ” over $T'_{A_1}, \dots, T'_{A_m}, T'_{A_{m+1}}, \dots, T'_{A_n}$ and E' be the label set of T' , i.e., the union of the label sets of $T'_{A_1}, \dots, T'_{A_m}, T'_{A_{m+1}}, \dots, T'_{A_n}$ and $\{A = A_0\}$. Then σ is the restriction of an mgu of E' to the variables in A , since S_{A_1}, \dots, S_{A_m} and $T'_{A_1}, \dots, T'_{A_m}$ are proof trees with the same answer substitutions. Then, from Lemma 3.9, T' is a proof tree of A with answer substitution σ by P_i (see Fig. 3). \square

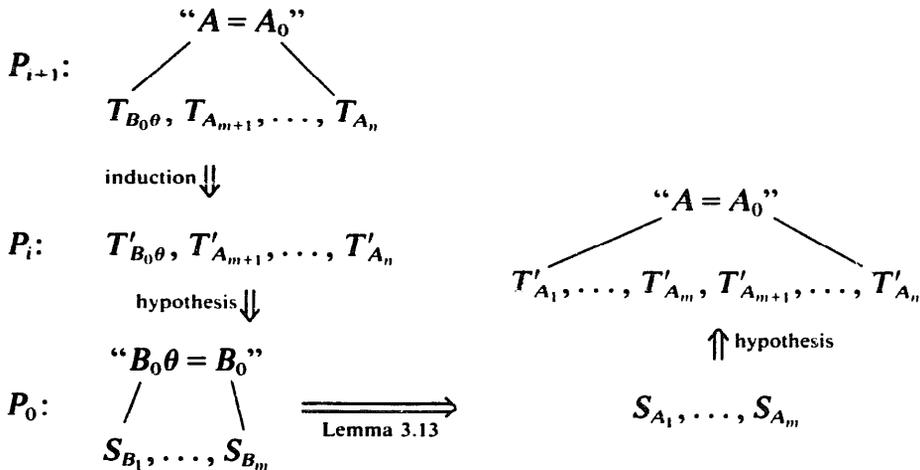


Fig. 3. Construction of proof tree for Case 3.

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. First, several definitions are prepared.

Definition 3.15 (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 3.16. Let P_0 be the initial program in Example 2.2, and T_1, T_2 be proof trees in Example 3.5. Then $w(T_1) = 2$ and $w(T_2) = 4$.

Definition 3.17 (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 3.18. Let P_0 be the program in Example 2.2, and T_1, T_2 be proof trees in Example 3.5. Then

$$w(\text{insert}(X, [Y], Z), \langle Z \Leftarrow [X, Y] \rangle) = 2,$$

because T_1 is the minimum proof tree of “insert($X, [Y], Z$)” with answer substitution $\langle Z \Leftarrow [X, Y] \rangle$ by P_0 . Similarly,

$$w(\text{insert}(X, [Y], Z), \langle Z \Leftarrow [Y, X] \rangle) = 4.$$

Definition 3.19 (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, \dots, 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, \dots, T_n 's weights, i.e., $w(F) = w(T_1) + w(T_2) + \dots + w(T_n)$.

Example 3.20. Let P_0 be the initial program in Example 2.2, and F_1, F_2 be proof forests in Example 3.7. Then $w(F_1) = 2$ and $w(F_2) = 4$.

Definition 3.21 (Weight of atom sequence). Let P_0 be the initial program in a transformation sequence, A_1, A_2, \dots, A_n be an atom sequence, and τ be a substitution. The *weight of A_1, A_2, \dots, A_n* with answer substitution τ , denoted by $w((A_1, A_2, \dots, A_n), \tau)$, is the minimum of the weight of the proof forests of A_1, A_2, \dots, A_n with answer substitution τ .

Example 3.22. Let P_0 be the program in Example 2.2, and F_1, F_2 be proof forests in Example 3.7. Then

$$w((\text{ap}(U_0, V_0, [Y]), \text{ap}(U_0, [X|V_0], N)), \langle U_0 \Leftarrow [], V_0 \Leftarrow [Y], N \Leftarrow [X, Y] \rangle) = 2,$$

because F_1 is the minimum proof forest of “ap($U_0, V_0, [Y]$), ap($U_0, [X|V_0], N$)” with answer substitution $\langle U_0 \Leftarrow [], V_0 \Leftarrow [Y], N \Leftarrow [X, Y] \rangle$ by P_0 . Similarly,

$$w((\text{ap}(U_0, V_0, [Y]), \text{ap}(U_0, [X|V_0], N)), \langle U_0 \Leftarrow [Y], V_0 \Leftarrow [], N \Leftarrow [Y, X] \rangle) = 4.$$

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

Definition 3.23 (Descent clause). Let P_i be a program in a transformation sequence starting from initial program P_0 , A an atom, σ a substitution for the variables in

A , and C a clause in P_i of the form $A_0 :- A_1, A_2, \dots, A_n$ whose head A_0 is unifiable with A , say by an mgu 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, \dots, A_n with answer substitution τ by P_0 such that

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

Definition 3.24 (*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 Lemmas 3.11, 3.12 and 3.13.

Lemma 3.25. *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 weight 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 3.11. When $i > 0$, it is obvious. \square

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 3.26. *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 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 A by P_0 , and $w(T) = w(T')$.*

Proof. Obvious. \square

Lemma 3.27. *Let P_0 be the initial program of a transformation sequence, T be a proof tree of atom A with answer substitution σ by 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 $A\theta$ by P_0 , and $w(T) = w(T')$.*

Proof. Obvious. \square

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

Lemma 3.28. *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 does not 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. In the former case, the lemma obviously holds. In 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. \square

Lemma 3.29. *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 $>$: $(A, \sigma) > (B, \tau)$ 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 $A_0 :- A_1, \dots, A_n$ and η is an mgu of A and A_0 . From the definition of descent clause,

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

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

$$w(A, \sigma) \geq w((A_1, A_2, \dots, A_n), \tau) = w(F) \geq w(T_j) = w(A_j, \sigma_j)$$

holds. If $w(A, \sigma) > w((A_1, \dots, A_n), \tau)$ holds, $(A, \sigma) > (A_j, \sigma_j)$ holds. If $w(A, \sigma) = w((A_1, \dots, A_n), \tau)$ holds, by condition (c) of descent clause, C does not satisfy folding condition (d). Hence, from Lemma 3.28, no new predicate appears in A_1, \dots, A_n . This implies that $(A, \sigma) > (A_j, \sigma_j)$ holds. Hence, whichever holds, $(A, \sigma) > (A_j, \sigma_j)$ holds. Then by induction on $>$, (A_j, σ_j) is in $\mathcal{M}(P_i)$, and there exists a proof forest of A_1, \dots, A_n with answer substitution τ by P_i . Thus (A, σ) is in $\mathcal{M}(P_i)$. \square

Lemma 3.30. *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

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

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 . \square

Lemma 3.31. *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 $A_0 :- A_1, A_2, \dots, A_n$ ($n \geq 0$) and A and A_0 are unifiable, say by an mgu η . We will show that there also exists 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 3.25, without loss of generality, we can assume that C is unfolded at A_1 . Since C is a descent clause, there exists a proof forest of A_1, A_2, \dots, A_n with answer substitution τ by P_0 such that the restriction of an mgu of η and τ to the variables in A is σ . Let F be the minimum proof forest among such proof forests, and $S_{A_1}, S_{A_2}, \dots, S_{A_n}$ be F 's component proof trees with answer substitutions $\sigma_1, \sigma_2, \dots, \sigma_n$ 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 $B_0 :- B_1, \dots, B_m$ ($m \geq 0$) and A_1 and B_0 are unifiable, say by an mgu θ . Let C' be the result of unfolding C using D . Then C' is of the form

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

First, since D is a descent clause, there exists a proof forest of B_1, \dots, B_m with answer substitution τ_1 by P_0 such that the restriction of an mgu 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}, \dots, S_{B_m} be F_1 's component proof trees, and E_1 be the union of the label sets of $S_{B_1}, \dots, S_{B_m}, S_{A_2}, \dots, S_{A_n}$ and $\{A = A_0, A_1 = B_0\}$. From Lemma 3.9, σ is the restriction of an mgu of E_1 to the variables in A .

Next, from Lemma 3.27, there exist proof trees $S_{B_1\theta}, \dots, S_{B_m\theta}$ of $B_1\theta, \dots, B_m\theta$ by P_0 such that they are identical to S_{A_1}, \dots, S_{A_m} except for the left-hand sides of the equations in the root labels. Similarly, from Lemma 3.27, there exist proof trees of $S_{A_2\theta}, \dots, S_{A_n\theta}$ of $A_2\theta, \dots, A_n\theta$ by P_0 such that they are identical to S_{A_2}, \dots, S_{A_n} except for the left-hand sides of the root labels. Let F' be the proof forest consisting of $S_{B_1\theta}, \dots, S_{B_m\theta}, S_{A_2\theta}, \dots, S_{A_n\theta}$, and E' be the union of the label sets of $S_{B_1\theta}, \dots, S_{B_m\theta}, S_{A_2\theta}, \dots, S_{A_n\theta}$ and $\{A = A_0\theta\}$. Then E' is identical to $(E_1 - \{A_1 = B_0\})\theta$. From Lemma 3.10, σ is the restriction of an mgu of E' to the variables in A , since θ does not substitute any term for the variables in A . Let θ' be an mgu of A and $A_0\theta$, and τ' be an mgu of the label set of F' . Then, F' is a proof forest of $B_1\theta, \dots, B_m\theta, A_2\theta, \dots, A_n\theta$ with answer substitution τ' by P_0 such that σ is the restriction of an mgu of θ' and τ' to the variables in A (see Fig. 4).

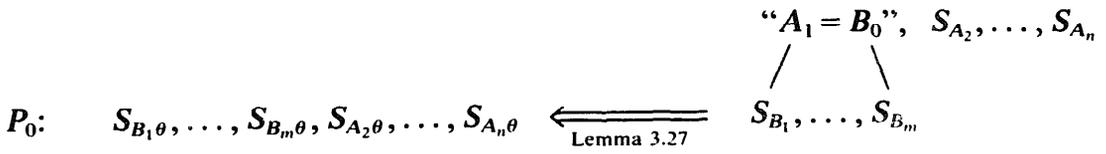


Fig. 4. Construction of proof forest for Case 2.

Last, from Lemma 3.27,

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

hold. Hence

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

holds. Further, if the predicate of B_0 is an old predicate, D satisfies folding condition (d), and if not, C does (from Lemma 3.28). Then, from condition (c) of descent clause, either

$$w(A, \sigma) > w((A_1, \dots, A_n), \tau) \quad \text{or} \quad w(A_1, \sigma_1) > w((B_1, \dots, B_m), \tau_1)$$

holds. Whichever holds,

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

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 $B_0 :- B_1, \dots, B_m$ ($m > 0$) and C' the result of folding. From Lemma 3.25, without loss of generality, we can assume that A_1, \dots, A_m are instances of B_1, \dots, B_m , say by an instantiation θ , and C' is of the form

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

First, since C is a descent clause, there exists a proof forest of A_1, \dots, A_n with answer substitutions τ by P_0 such that the restriction of an mgu of η and τ to the variables in A is σ . Let F be the minimum proof forest among such proof forests, let S_{A_1}, \dots, S_{A_n} be F 's component proof trees of A_1, \dots, A_n with answer substitutions $\sigma_1, \dots, \sigma_n$ by P_0 , and let E_1 be the union of the label sets of S_{A_1}, \dots, S_{A_n} and $\{A = A_0\}$. From Lemma 3.9, σ is the restriction of an mgu of E_1 to the variables in A .

Next, from Lemma 3.26, there exist proof trees S_{B_1}, \dots, S_{B_m} of B_1, \dots, B_m by P_0 such that they are identical to S_{A_1}, \dots, S_{A_m} except for the left-hand sides of the root labels, since $B_1\theta = A_1, \dots, B_m\theta = A_m$ 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}, \dots, S_{B_m} , let F' be the proof forest consisting of $S_{B_0\theta}, S_{A_{m+1}}, \dots, S_{A_n}$, and let E' be the union of the label sets of $S_{B_0\theta}, S_{A_{m+1}}, \dots, S_{A_n}$ and $\{A = A_0\}$. Then, from folding condition (b), E_1 is identical to $(E' - \{B_0\theta = B_0\})\theta$. From Lemma 3.10, σ is the restriction of an mgu of E' to the variables in A , since θ does not substitute any term for the variables in A . Let τ' be an mgu of the union of the label sets of $S_{B_0\theta}, S_{A_{m+1}}, \dots, S_{A_n}$. Then, F' is a proof forest of $B_0\theta, A_{m+1}, \dots, A_n$ with answer substitution τ' by P_0 such that the restriction of an mgu of η and τ' to the variables in A is σ (see Fig. 5).

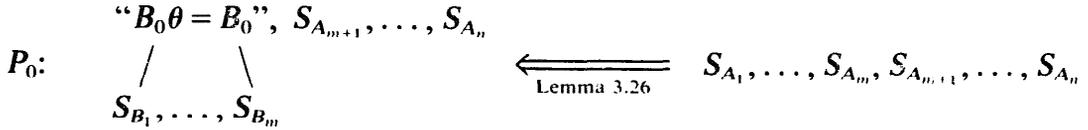


Fig. 5. Construction of proof forest for Case 3.

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}, \dots, S_{B_m} of B_1, \dots, 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,

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

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

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

and from Lemma 3.26,

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

hold. Hence

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

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

Theorem 3.32 (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 Lemmas 3.30 and 3.31, P_{i+1} is weight complete, and then from Lemma 3.29, $\mathcal{M}(P_{i+1}) \supseteq \mathcal{M}(P_0)$ for $i = 0, 1, \dots, N-1$. Further, from Lemma 3.14, $\mathcal{M}(P_{i+1}) = \mathcal{M}(P_0)$ holds for $i = 0, 1, \dots, N-1$. \square

The original result by Tamaki and Sato [9, 10] can be derived as a corollary.

Corollary 3.33 (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 3.32, $\mathcal{M}(P)$ is preserved. Thus, the least Herbrand model is preserved. \square

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 the 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

$$?- \text{setof}(X, p(X), Y),$$

P_2 succeeds with answer substitution $\langle X \Leftarrow a, Y \Leftarrow [a] \rangle$, 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 [9, 10, 11] adopted as one of the basic transformation rules. 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.

Recently, it was proven that any program obtained from an initial program by applying Tamaki-Sato’s transformation returns the same answer substitutions *the same number of times* as the initial program for any given top-level goal (see [4] for the details).

Acknowledgment

This work is based on results of Tamaki and Sato [9, 10, 11]. 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, 5, 6]. 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 (Former Chief of ICOT 3rd Laboratory) for their advice and encouragement. We would also thank Mr. H. Seki (Mitsubishi Electric Corporation) for his valuable suggestions.

References

- [1] R.M. Burstall and J. Darlington, A transformation system for developing recursive programs, *J. ACM* 24 (1977) 44-67.
- [2] T. Kanamori and H. Fujita, Unfold/fold logic program transformation with counters, presented at *U.S.-Japan Workshop on Logic Programs*, Honolulu, 1987; also a preliminary version appeared as ICOT Technical Report TR-179, 1986.

- [3] T. Kanamori and K. Horiuchi, Construction of logic programs based on generalized unfold/fold rules, in: *Proc. 4th Internat. Conf. on Logic Programming*, Melbourne (1987) 744–768; also a preliminary version appeared as ICOT Technical Report TR-177, 1986.
- [4] T. Kanamori and T. Kawamura, Preservation of stronger equivalence in unfold/fold logic program transformation (II), ICOT Technical Report TR-403, 1988.
- [5] T. Kanamori and M. Maeji, Derivation of logic programs from implicit definition, ICOT Technical Report TR-178, 1986.
- [6] T. Kawamura and T. Kanamori, Preservation of stronger equivalence in unfold/fold logic program transformation, in *Proc. Internat. Conf. on Fifth Generation Computer Systems*, Tokyo (1988) 413–421; also a preliminary version appeared as ICOT Technical Report TR-399, 1988.
- [7] M.J. Maher, Equivalences of logic programs, in: *Proc. 3rd Internat. Conf. on Logic Programming*, London (1986), 410–424.
- [8] Z. Manna and R. Waldinger, Synthesis: dreams \Rightarrow programs, *IEEE Trans. Software Engineering* **5** (1979) 294–328.
- [9] H. Tamaki, Program transformation in logic programming (in Japanese), in: K. Fuchi, K. Furukawa and F. Mizoguchi, eds., *Program Transformation* (Kyoritsu Pub. Co., 1987) 39–62.
- [10] H. Tamaki and T. Sato, Unfold/fold transformation of logic programs, in: *Proc. 2nd Internat. Logic Programming Conf.* Uppsala (1984) 127–138.
- [11] H. Tamaki and T. Sato, A generalized correctness proof of the unfold/fold logic program transformation, Department of Information Science TR86-04, Ibaraki University, 1986.