TR-403

Preservation of Stronger Equivalence in Unfold/Fold Logic Program Transformation (II)

by
T. Kanamori and T. Kawamura(Mitsubishi)

June, 1988

ICOT
Mita Kokusai Bldg. 21F
1-28 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 (II)

Tadashi KANAMORI Tadashi KAWAMURA

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 nor how many times the same answer substitutions are returned. This paper proves that any program obtained from an initial program by applying Tamaki-Sato's transformation can compute the same answer substitutions the same number of times as the initial program for any given top-level goal.

Keywords: Program Transformation, Prolog, Equivalence 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 [7] 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 [6]). Tamaki and Sato [8] [9] [10] proposed the 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 nor how many times the same answer substitutions are returned. For example, consider the following three Prolog programs P_1, P_2 and P_3.

```
P1 : p(X).
     q(a).
P2 : p(a).
     q(a).
P3 : p(a).
     p(X) :- q(X).
     q(a).
```

Because the Herbrand universes of P_1, P_2 and P_3 are {a}, they are equivalent in the sense of the least Herbrand model semantics. However, these programs respond in different manners to a query

```
?- p(X).
```

P_1 returns the empty substitution <>, while P_2 and P_3 returns substitution <X ⇐ a> as its answer. Moreover, P_2 returns the answer substitution only once, while P_3 returns it twice. 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 multiset 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. 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, θ, σ, τ 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 [10].

Definition Program

A *clause* is a pair consisting of a clause identifier and a definite clause. A *program* is a finite set of clauses.

Hereafter, we will assume that no clauses has the same clause identifier so that two definite clauses of the same form are distinguished as different clauses in the program due to the clause identifiers. In the following, we will often use the clause identifiers for referring the definite clauses.

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. (C_1, C_2, C_3 are clause identifiers.)

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 the body of 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 clause* 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 of C_3, 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,[X,Y],N).

these five programs return the same answer substitutions

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

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 [10] [9].

3.1 Proof Tree

Because we need to consider what answer substitutions are returned how many times 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 ("$A = B$", C) where A and B are unifiable atoms and C is a clause identifier. The set of all the equations in 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 :- $B_1, B_2, \ldots, B_n$

such that

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

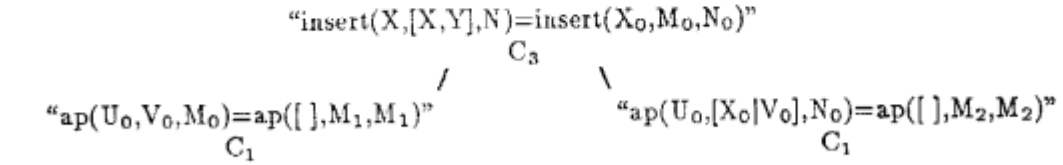
(b) the root node of T is labelled with ("$A = B$", C),

(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

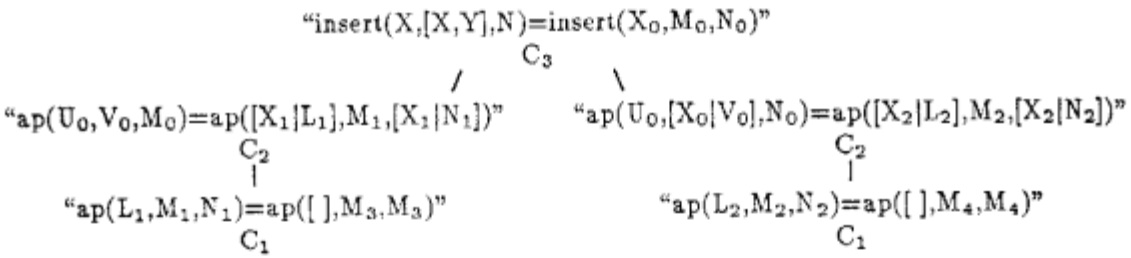
(d) σ is the restriction of an m.g.u. of $\theta, \sigma_1, \sigma_2, \ldots, \sigma_n$ to the variables in A.

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 suffixes.

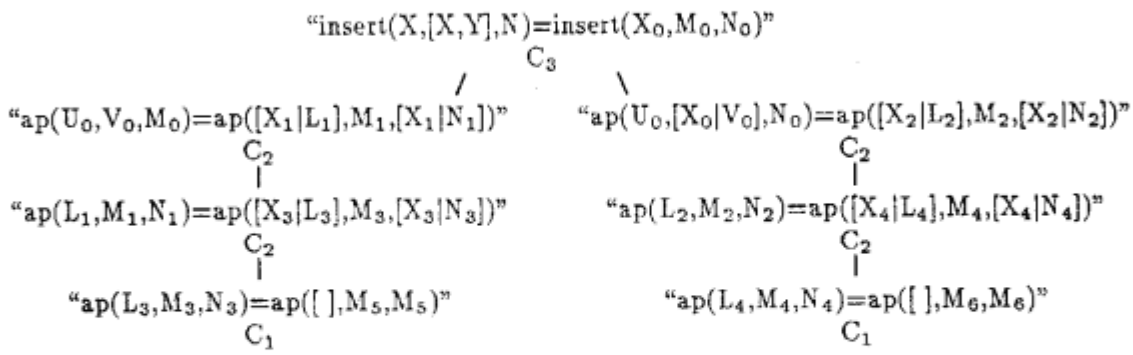
Example 3.1.1 Let P_0 be the program in Example 2.1. Tree T below is a proof tree of 'insert(X,[X,Y],N)' with answer substitution <N ⇐ [X,X,Y]> by P_0.



Tree T_2 below is another proof tree of 'insert(X,[X,Y],N)' with answer substitution <N ⇐ [X,X,Y]> by P_0.



Tree T_3 below is also a proof tree of 'insert(X,[X,Y],N)' with answer substitution <N ⇐ [X,Y,X]> by P_0.



Definition Success Multiset

Let P be a program. The multiset 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 multiset* of P, and denoted by $\mathcal{M}(P)$.

Note that $\mathcal{M}(P)$ is not a set but a multiset so that, if there exist k different proof trees of atom A with the same answer substitution σ by P, then $\mathcal{M}(P)$ includes k atom-substitution pairs (A, σ).

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 equation in the root label 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 Mapping between Proof Tree Sets

Note that the success multiset characterizes Prolog programs more precisely than the least Herbrand model. It is, however, not easy to consider preservation of the success multiset directly so that we will consider the following set:

Definition Proof Tree Set
Let P be a program. The set of all the proof trees by P is called the *proof tree set* of P, and denoted by $\mathcal{T}(P)$.

Note that each atom-substitution pair in the success multiset corresponds to a proof tree in the proof tree set so that two success multisets are identical when there exists a one-to-one correspondence between the corresponding proof tree sets. Hence, we will consider mappings between proof tree sets in the following discussion.

Definition Consistent Mapping between Proof Tree Sets
Let P_i and P_j be programs. A mapping f from $\mathcal{T}(P_i)$ to $\mathcal{T}(P_j)$ is called a *consistent mapping from* $\mathcal{T}(P_i)$ *to* $\mathcal{T}(P_j)$, and denoted by $\mathcal{T}(P_i) \xrightarrow{f} \mathcal{T}(P_j)$, when f maps a proof tree T of atom A with answer substitution σ in $\mathcal{T}(P_i)$ to a proof tree T' of the same atom A with the same answer substitution σ in $\mathcal{T}(P_j)$. A consistent mapping f is said to be *one-to-one* when $f(T_1) = f(T_2)$ if and only if $T_1 = T_2$.

Definition Consistent Mapping Pair between Proof Tree Sets
Let P_i and P_j be programs. A pair of mappings (f, g) is called a *consistent mapping pair between* $\mathcal{T}(P_i)$ *and* $\mathcal{T}(P_j)$, and denoted by $\mathcal{T}(P_i) \underset{g}{\overset{f}{\rightleftharpoons}} \mathcal{T}(P_j)$, when

(a) f is a consistent mapping from $\mathcal{T}(P_i)$ to $\mathcal{T}(P_j)$,
(b) g is a consistent mapping from $\mathcal{T}(P_j)$ to $\mathcal{T}(P_i)$, and
(c) $g \circ f = id_{\mathcal{T}(P_i)}$ and $f \circ g = id_{\mathcal{T}(P_j)}$, where $id_{\mathcal{T}(P_i)}$ and $id_{\mathcal{T}(P_j)}$ are the identity mappings on $\mathcal{T}(P_i)$ and $\mathcal{T}(P_j)$, respectively.

3.3 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 by showing that there exists a one-to-one consistent mappaing from $\mathcal{T}(P_i)$ to $\mathcal{T}(P_0)$, which is the easier direction of stronger equivalence.

Lemma 3.3.1 Let P_i be a program in a transformation sequenece, C be a clause in P_i, C' be a clause obtained from C by permuting the atoms in the body of C, and P_i' be $(P_i - \{C\}) \cup \{C'\}$. Let f be a mapping from $\mathcal{T}(P_i)$ to $\mathcal{T}(P_i')$ such that it maps proof tree T to proof tree T' if and only if T' is 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, and g be the inverse of f. Then (f, g) is a consistent mapping pair between $\mathcal{T}(P_i)$ and $\mathcal{T}(P_i')$.

Proof. 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 the existence of a sequence of consistent mapping pairs between P_0 and P_i.

Lemma 3.3.2 Let P_0 be an initial program, and T be a proof tree of atom $A\theta$ by program P_0. Let T' be the labelled tree obtained from T by replacing $A\theta$ in the left-hand side of the equation in the root label with A. Then T' is a proof tree of atom A by program P_0.

Proof. Obvious.

Lemma 3.3.3 Let P_0 be an initial program, T be a proof tree of atom A with answer substitution σ by program P_0, 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 equation in the root label with $A\theta$. Then T' is a proof tree of atom $A\theta$ by program P_0.

Proof. Obvious.

Lemma 3.3.4 Let $P_0, P_1, \ldots, P_N$ be a transformation sequence. If there exist consistent mapping pairs

$$\mathcal{T}(P_0) \underset{g_1}{\overset{f_1}{\rightleftharpoons}} \mathcal{T}(P_1) \underset{g_2}{\overset{f_2}{\rightleftharpoons}} \mathcal{T}(P_2) \underset{g_3}{\overset{f_3}{\rightleftharpoons}} \cdots \underset{g_i}{\overset{f_i}{\rightleftharpoons}} \mathcal{T}(P_i),$$

then there exists a consistent mapping g_{i+1} from $\mathcal{T}(P_{i+1})$ to $\mathcal{T}(P_i)$ for $i = 0, 1, 2, \ldots, N-1$.

Proof. Let T be a proof tree of A with answer substitution σ by P_{i+1}. By induction on the structure of T, we will define a consistent mapping g_{i+1} such that $g_{i+1}(T)$ is 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

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

$T_{A_1}, T_{A_2}, \ldots, T_{A_n}$ be T's immediate subproofs of $A_1, A_2, \ldots, A_n$. By the inductive definition, $g_{i+1}(T_{A_1}), g_{i+1}(T_{A_2}), \ldots, g_{i+1}(T_{A_n})$ are 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 the proof tree obtained by putting a root node labelled with ("$A = B$", C) over $T'_{A_1}, T'_{A_2}, \ldots, T'_{A_n}$. 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.

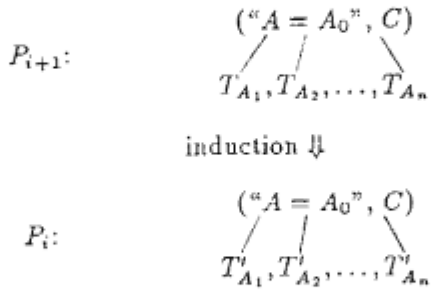


Figure 3.3.1 Definition of g_{i+1} for Case 1

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

From Lemma 3.3.1, 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

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

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 inductive definition, $g_{i+1}(T_{B_1\theta}), \ldots, g_{i+1}(T_{B_m\theta})$, $g_{i+1}(T_{A_2\theta}), \ldots$, $g_{i+1}(T_{A_n\theta})$ are 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.

Second, proof trees $S_{B_1\theta}, \ldots, S_{B_m\theta}, S_{A_2\theta}, \ldots, S_{A_2\theta}$ of $B_1\theta, \ldots, B_m\theta, A_2\theta, \ldots, A_n\theta$ by P_0 with the same answer substitution as $T'_{B_1\theta}, \ldots, T'_{B_m\theta}, T'_{A_2\theta}, \ldots, T'_{A_n\theta}$ are uniquely determined by applying $g_i, \ldots, g_1$ to $T'_{B_1\theta}, \ldots, T'_{B_m\theta}, T'_{A_2\theta}, \ldots, T'_{A_n\theta}$. Let E_2 be the union of the label sets of $S_{B_1\theta}, \ldots, S_{B_m\theta}, S_{A_2}, \ldots, S_{A_n}$ 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 3.3.2, there exist proof trees $S_{B_1}, \ldots, S_{B_m}, S_{A_2}, \ldots, S_{A_n}$ of $B_1, \ldots, B_m, A_2, \ldots, A_n$ by P_0 such that they are identical to $S_{B_1\theta}, \ldots, S_{B_m\theta}, S_{A_2\theta}, \ldots, S_{A_n\theta}$ except the left-hand sides of the equations in the root labels. Let E_3 be the union of the label sets of $S_{B_1}, \ldots, S_{B_m}, S_{A_2}, \ldots, S_{A_n}$ and $\{A = A_0, A_1 = B_0\}$. Then E_2 is identical to $(E_3 - \{A_1 = B_0\})\theta$. 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, 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 with the same answer substitutions as $S_{B_1}, \ldots, S_{B_m}, S_{A_2}, \ldots, S_{A_m}$ are uniquely determined by applying $f_1, \ldots, f_i$ to $S_{B_1}, \ldots, S_{B_m}, S_{A_2}, \ldots, S_{A_m}$. Let T'_{A_1} be the proof tree obtained by putting a root node labelled with ("$A_1 = B_0$", D) over $T'_{B_1}, \ldots, T'_{B_m}$. Let T' be the proof tree obtained by putting a root node labelled with ("$A = A_0$", C') 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 $\{A = A_0, A_1 = B_0\}$. From Lemma 3.1.1, σ is the restriction of an m.g.u. of E' to the variables in A. Hence, T' is a proof tree of A with answer substitution σ by P_i.

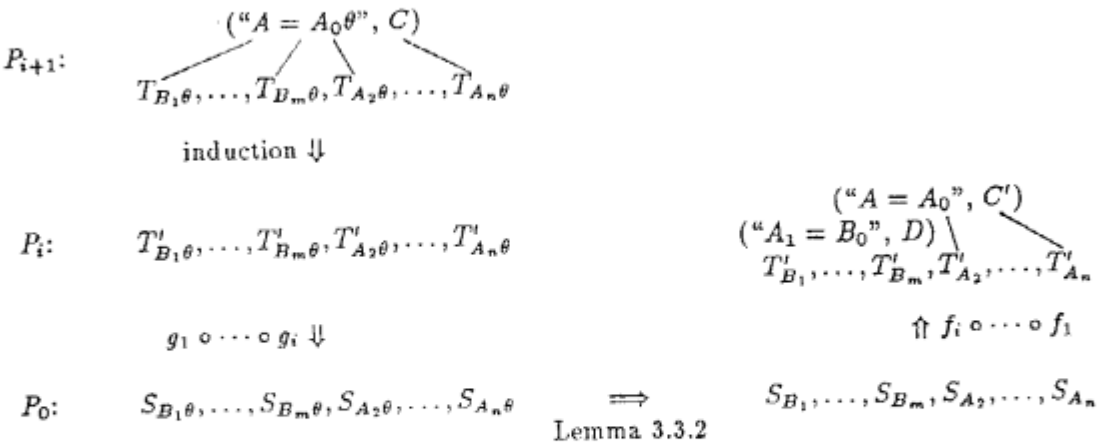


Figure 3.3.2 Definition of g_{i+1} for Case 2

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

and D be the folding clause of the form

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

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

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 inductive definition, $g_{i+1}(T_{B_0\theta}), g_{i+1}(T_{A_{m+1}}), \ldots, g_{i+1}(T_{A_n})$ are 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, a proof tree $S_{B_0\theta}$ of $B_0\theta$ by P_0 with the same answer substitution as $T'_{B_0\theta}$ is uniquely determined by applying $g_i, \ldots, g_1$ to $T'_{B_0\theta}$. 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, from folding condition (c), this clause should be D. Hence, the root label of $S_{B_0\theta}$ is ("$B_0\theta = B_0$", D), 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 $S_{B_1}, \ldots, S_{B_m}, T'_{A_{m+1}}, \ldots, T'_{A_n}$ and $\{A = A_0, B_0\theta = B_0\}$. 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 3.3.3, 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 equations in 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_2 is identical to $(E_3 - \{B_0\theta = B_0\})\theta$. 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, proof trees $T'_{A_1}, \ldots, T'_{A_m}$ of $A_1, \ldots, A_m$ by P_i with the same answer substitutions as $S_{A_1}, \ldots, S_{A_m}$ are uniquely determined by applying $f_1, \ldots, f_i$ to $S_{A_1}, \ldots, S_{A_m}$. Let T' be the proof tree obtained by putting a root node labelled with ("$A = A_0$", C') 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' to the variables in A. Hence, T' is a proof tree of A with answer substitution σ by P_i.

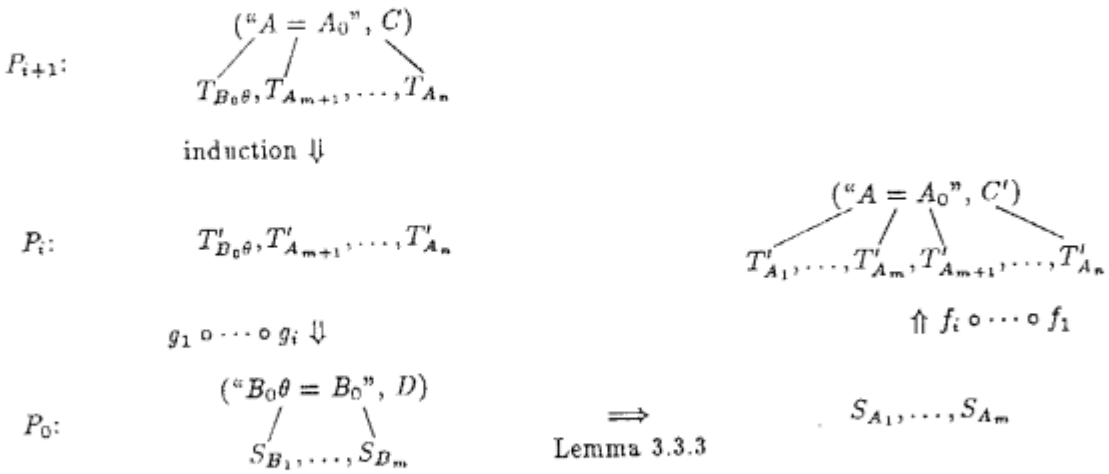


Figure 3.3.3 Definition of g_{i+1} for Case 3

Lemma 3.3.5 Let g_{i+1} be the consistent mapping defined in Lemma 3.3.4. Then g_{i+1} is one-to-one.

Proof. Although this lemma is obvious from Lemma 3.4.8 to be proved later, we will prove it by itself here. Due to space limit, we will show the proof only for the most complicated "Case 3." (See Figure 3.3.3.) The other cases are proved in the same way.

Suppose that $g_{i+1}(T^{(1)}) = g_{i+1}(T^{(2)}) = T'$. We will show that $T^{(1)} = T^{(2)}$ by induction on the structure of $T^{(1)}$ and $T^{(2)}$. Let ("$A = A_0$", C') be the root label of T', C' be the clause in P_i used at the root of T' of the form

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

and $T'_{A_1}, \ldots, T'_{A_m}, T'_{A_{m+1}}, \ldots, T'_{A_n}$ be the immediate subproofs of T'. Suppose that, from P_i to P_{i+1}, clause C' is folded to clause C using instantiation θ. Obviously, the root nodes of $T^{(1)}$ and $T^{(2)}$ are labelled with ("$A = A_0$", C). Let $T^{(1)}_{B_0\theta}, T^{(1)}_{A_{m+1}}, \ldots, T^{(1)}_{A_n}$ be the immediate subproofs of $T^{(1)}$, and $T^{(2)}_{B_0\theta}, T^{(2)}_{A_{m+1}}, \ldots, T^{(2)}_{A_n}$ be the immediate subproofs of $T^{(2)}$. Following the definition of g_{i+1} in the reverse direction, $T'_{A_1}, \ldots, T'_{A_m}$ uniquely determine proof tree $T'_{B_0\theta}$ of $B_0\theta$ by P_i. Then,

$$g_{i+1}(T^{(1)}_{B_0\theta}) = g_{i+1}(T^{(1)}_{B_0\theta}) = T'_{B_0\theta},$$
$$g_{i+1}(T^{(1)}_{A_{m+1}}) = g_{i+1}(T^{(1)}_{A_{m+1}}) = T'_{A_{m+1}},$$
$$\vdots$$
$$g_{i+1}(T^{(1)}_{A_n}) = g_{i+1}(T^{(1)}_{A_n}) = T'_{A_n}.$$

From the induction hypothesis, $T^{(1)}_{B_0\theta} = T^{(2)}_{B_0\theta}, T^{(1)}_{A_{m+1}} = T^{(2)}_{A_{m+1}}, \ldots, T^{(1)}_{A_n} = T^{(2)}_{A_n}$ hold. Hence $T^{(1)} = T^{(2)}$.

3.4 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 by showing that there exists a consistent mapping pair between $\mathcal{T}(P_0)$ and $\mathcal{T}(P_i)$, which is the harder direction of stronger equivalence. First, several definitions are prepared.

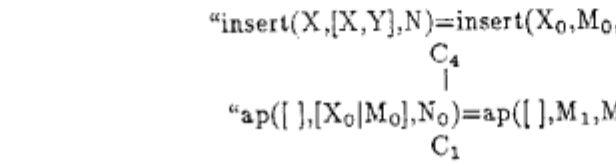
Definition Original Proof Tree

Let $P_0, P_1, \ldots, P_i$ be a transformation sequence such that there exist consistent mapping pairs

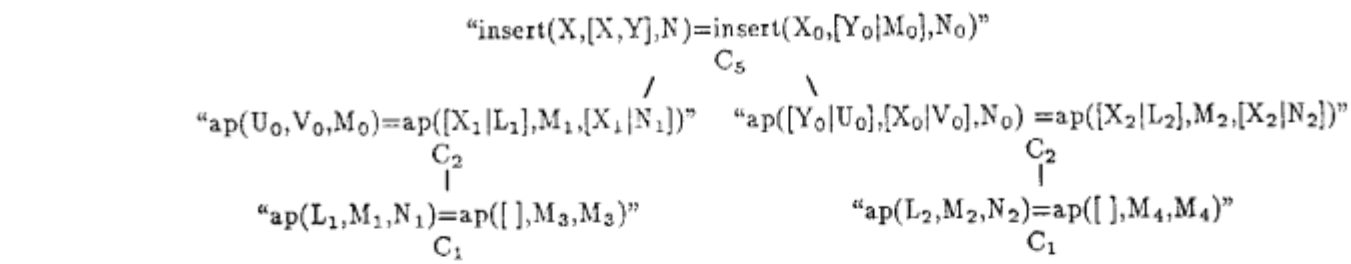
$$\mathcal{T}(P_0) \underset{g_1}{\overset{f_1}{\rightleftharpoons}} \mathcal{T}(P_1) \underset{g_2}{\overset{f_2}{\rightleftharpoons}} \mathcal{T}(P_2) \underset{g_3}{\overset{f_3}{\rightleftharpoons}} \cdots \underset{g_i}{\overset{f_i}{\rightleftharpoons}} \mathcal{T}(P_i).$$

Let T_0 be a proof tree in $\mathcal{T}(P_0)$, and T_i be a proof tree in $\mathcal{T}(P_i)$ obtained by successive application of $f_1, f_2, \ldots, f_i$ to T_0. (Or equivalently, let T_i be a proof tree in $\mathcal{T}(P_i)$, and T_0 be a proof tree in $\mathcal{T}(P_0)$ obtained by successive application of $g_i, \ldots, g_2, g_1$ to T_i.) Then T_0 is called the *original proof tree of* T_i.

Example 3.4.1 Let P_0, P_1 be a transformation sequence in Example 2.2, T_1, T_2 be proof trees in Example 3.1.1. Let T_1' be a proof tree of 'insert(X,[X,Y],N)' with answer substitution <N ⇐ [X,X,Y]> by P_1 depicted below:



Let T_2' be another proof tree of 'insert(X,[X,Y],N)' with answer substitution <N ⇐ [X,X,Y]> by P_1 depicted below:



Let f_1 be a consistent mapping from $\mathcal{T}(P_0)$ to $\mathcal{T}(P_1)$ such that $f_1(T_1) = T_1'$ and $f_1(T_2) = T_2'$, and g_1 be its inverse. Then T_1 is the original proof tree of T_1', and T_2 is the original proof tree of T_2'.

In the following definitions and Lemmas 3.4.1–5, P_i is assumed to be a program in a transformation sequence such that there exists a sequence of consisting mapping pairs $(f_1, g_1), (f_2, g_2), \ldots, (f_i, g_i)$ as above.

Definition Weight of Proof Tree

Let P_i be a program in a transformation sequence, T be a proof tree of atom A by P_i, T_0 be the original proof tree of T, and s be the size of T_0. 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 3.4.2 Let T_1, T_2, T_1', T_2' be proof trees in Example 3.4.1. Then $w(T_1') = w(T_1) = 2$, and $w(T_2') = w(T_2) = 4$.

The following notions, which are generalizations of those in [10], play important roles in the following proof.

Definition Weight Completeness

Let P_i be a program in a transformation sequence. Then P_i is said to be *weight complete* (*w.r.t.* $(f_1, g_1), (f_2, g_2), \ldots, (f_i, g_i)$) when it satisfies the following conditions: let T be a proof tree by program P_i, C be the clause used at the root of T, and $T_1, T_2, \ldots, T_n$ be T's immedaite subproofs. Then

(a) $w(T) \geq w(T_1) + w(T_2) + \cdots + w(T_n)$, and
(b) $w(T) > w(T_1) + w(T_2) + \cdots + w(T_n)$ when C satisfies folding condition (d).

Definition Well-founded Ordiring $\succ$ on Proof Tree Set

Let P_i be a program in a transformation sequence. A well-founded ordering $\succ$ on proof tree set of program P_i is defined as follows: let T be a proof tree of A by P_i, and T' be a proof tree of A' by P_i. Then $T \succ T'$ if and only if

(a) $w(T) > w(T')$, or
(b) $w(T) = w(T')$ and the predicate of A is a new predicate and the predicate of A' is an old predicate.

The next three lemmas are slight extensions of Lemma 3.3.1, 3.3.2 and 3.3.3.

Lemma 3.4.1 Let P_i be a program in a transformation sequenece, C be a clause in P_i, C' be a clause obtained from C by permuting the atoms in the body of C, and P_i' be $(P_i - \{C\}) \cup \{C'\}$. Let f be a mapping from $\mathcal{T}(P_i)$ to $\mathcal{T}(P_i')$ such that it maps proof tree T to proof tree T' if and only if T' is 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, and g be the inverse of f. Then (f, g) is a consistent mapping pair between $\mathcal{T}(P_i)$ and $\mathcal{T}(P_i')$, and P_i is weight complete w.r.t. $(f_1, g_1), (f_2, g_2), \ldots, (f_i, g_i)$ if and only if P_i' is weight complete w.r.t. $(f_1, g_1), (f_2, g_2), \ldots, (f \circ f_i, g \circ g_i)$.

Proof. 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 the existence of a sequence of consistent mapping pairs between P_0 and P_i and weight completeness of P_i.

Lemma 3.4.2 Let P_0 be an initial program, 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 equation in 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 3.4.3 Let P_0 be an initial program, 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 equation in 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 two more lemmas, we will start the proof of total correctness.

Lemma 3.4.4 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 3.4.5 Let P_i be a program in a transformation sequence, T be any proof tree by P_i, and $T_1, T_2, \ldots, T_n$ be the immediate subproofs of T. If P_i is weight complete, then $T \succ T_j$ for $j = 1, 2, \ldots, n$.

Proof. Let C be the clause used at the root of T of the form

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

Then T_j is a proof tree of A_j. From the definition of weight completeness,

$$w(T) \geq w(T_1) + w(T_2) + \cdots + w(T_n),$$

hence, $w(T) \geq w(T_j)$ holds for $j = 1, 2, \ldots, n$, since $w(T_1), w(T_2), \ldots, w(T_n)$ are non-negative numbers. If

$$w(T) > w(T_1) + w(T_2) + \cdots + w(T_n),$$

then $T \succ T_j$ holds. If

$$w(T) = w(T_1) + w(T_2) + \cdots + w(T_n),$$

by condition (b) of weight completeness, C doesn't satisfy folding condition (d). Then, from Lemma 3.4.4, no new predicate appears in $A_1, \ldots, A_n$. Hence $T \succ T_j$ holds for $j = 1, 2, \ldots, n$.

Lemma 3.4.6 The initial program P_0 of a transformation sequence is weight complete.

Proof. Let T be a proof tree by P_0, C be the clause used at the root of T, and $T_1, T_2, \ldots, T_n$ be T's immediate subproofs. Obviously the condition (a) of weight completeness is satisfied. 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 (b) of weight completeness is satisfied. Thus P_0 is weight complete.

Lemma 3.4.7 Let $P_0, P_1, \ldots, P_N$ be a transformation sequence. If there exist consistent mapping pairs

$$\mathcal{T}(P_0) \overset{f_1}{\underset{g_1}{\rightleftharpoons}} \mathcal{T}(P_1) \overset{f_2}{\underset{g_2}{\rightleftharpoons}} \mathcal{T}(P_2) \overset{f_3}{\underset{g_3}{\rightleftharpoons}} \cdots \overset{f_i}{\underset{g_i}{\rightleftharpoons}} \mathcal{T}(P_i),$$

such that $P_0, P_1, \ldots, P_i$ are weight complete, then there exists a consistent mapping f_{i+1} from $\mathcal{T}(P_i)$ to $\mathcal{T}(P_{i+1})$ for $i = 0, 1, \ldots, N-1$.

Proof. Let T be a proof tree of atom A with answer substitution σ by P_i, C be the clause used at the root of T of the form

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

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 induction on the well-founded ordering $\succ$, we will define a consistent mapping f_{i+1} such that $f_{i+1}(T)$ is a proof tree T' of A with answer substitution σ by P_{i+1}.

Case 1 : C is in P_{i+1}.

From Lemma 3.4.5, $T \succ T_{A_j}$ holds for $j = 1, 2, \ldots, n$. By the inductive definition, $f_{i+1}(T_{A_1}), f_{i+1}(T_{A_1}), \ldots, f_{i+1}(T_{A_n})$ are proof trees $T'_{A_1}, T'_{A_2}, \ldots, T'_{A_n}$ of $A_1, A_2, \ldots, A_n$ by P_{i+1} with the same answer substitutions as $T_{A_1}, T_{A_2}, \ldots, T_{A_n}$. Let T' be the proof tree obtained by putting a root node labelled with ("$A = A_0$", C) over $T'_{A_1}, T'_{A_2}, \ldots, T'_{A_n}$. 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+1}.

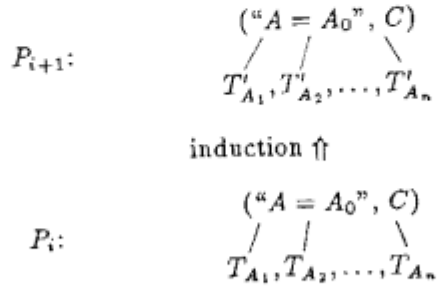


Figure 3.4.1 Definition of f_{i+1} for Case 1

Case 2 : C is unfolded.

From Lemma 3.4.1, without loss of generality, we can assume that A_1 is unfolded. Let D be the clause used at the root of T_{A_1} of the form

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

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

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

First, let $T_{B_1}, \ldots, T_{B_m}$ be T_{A_1}'s immediate subproofs of $B_1, \ldots, B_m$. Then, proof trees $S_{B_1}, \ldots, S_{B_m}$, $S_{A_2}, \ldots, S_{A_n}$ of $B_1, \ldots, B_m$, $A_2, \ldots, A_n$ by P_0 with the same answer substitutions as $T_{B_1}, \ldots, T_{B_m}$, $T_{A_2}, \ldots, T_{A_n}$ are uniquely determined by applying $g_i, \ldots, g_1$ to $T_{B_1}, \ldots, T_{B_m}$, $T_{A_2}, \ldots, T_{A_n}$. Let 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.

Second, from Lemma 3.4.3, there exist proof trees $S_{B_1\theta}, \ldots, S_{B_m\theta}$, $S_{A_2\theta}, \ldots, S_{A_n\theta}$ of $B_1\theta, \ldots, B_m\theta$, $A_2\theta, \ldots, A_n\theta$ by P_0 such that they are identical to $T_{B_1}, \ldots, T_{B_m}$ except the left-hand sides of the equations in the root labels. Let E_2 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_2 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_2 to the variables in A, since θ dose not substitute for the variables in A. Note that, from Lemma 3.4.3, $w(S_{B_1}) = w(S_{B_1\theta}), \ldots, w(S_{B_m}) = w(S_{B_m\theta})$, $w(S_{A_2}) = w(S_{A_2\theta}), \ldots, w(S_{A_n}) = w(S_{A_n\theta})$.

Third, 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 $S_{B_1\theta}, \ldots, S_{B_m\theta}$, $S_{A_2\theta}, \ldots, S_{A_n\theta}$ are uniquely determined by applying $f_1, \ldots, f_i$ to $S_{B_1\theta}, \ldots, S_{B_m\theta}$, $S_{A_2\theta}, \ldots, S_{A_n\theta}$. Let E_3 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_3 to the variables in A.

Last, since P_i is weight complete,

$w(T) \geq w(T_{A_1}) + w(T_{A_2}) + \cdots + w(T_{A_n})$,

$w(T_{A_1}) \geq w(T_{B_1}) + \cdots + w(T_{B_m})$

hold. In addition, if the predicate of B_0 is an old prdicate, D satisfies folding condition (d), and if not, C does from Lemma 3.4.4. Hence, from condition (b) of weight completeness, either

$$w(T) > w(T_{A_1}) + w(T_{A_2}) + \cdots + w(T_{A_n})$$

or

$$w(T_{A_1}) > w(T_{B_1}) + \cdots + w(T_{B_m})$$

holds. Whichever holds, from $w(T_{B_1}) = w(T_{B_1\theta}), \ldots, w(T_{B_m}) = w(T_{B_m\theta})$, $w(T_{A_2}) = w(T_{A_2\theta}), \ldots, w(T_{A_n}) = w(T_{A_n\theta})$,

$$\begin{aligned} w(T) &> w(T_{B_1}) + \cdots + w(T_{B_m}) + w(T_{A_2}) + \cdots + w(T_{A_n}) \\ &= w(T_{B_1\theta}) + \cdots + w(T_{B_m\theta}) + w(T_{A_2\theta}) + \cdots + w(T_{A_n\theta}). \end{aligned}$$

holds. Thus $T \succ T_{B_j\theta}$ holds for $j = 1, \ldots, m$, and $T \succ T_{A_k\theta}$ holds for $k = 2, \ldots, n$. By the inductive definition, $f_{i+1}(T_{B_1\theta}), \ldots, f_{i+1}(T_{B_m\theta}), f_{i+1}(T_{A_2\theta}), \ldots, f_{i+1}(T_{A_n\theta})$ are 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+1} 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 T' be the proof tree obtained by putting a root node labelled with ("$A = A_0\theta$", C') over $T'_{B_1\theta}, \ldots, T'_{B_m\theta}, T'_{A_2\theta}, \ldots, T'_{A_n\theta}$, and E' 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' to the variables in A. Hence, T' is a proof tree of A with answer substitution σ by P_{i+1}.

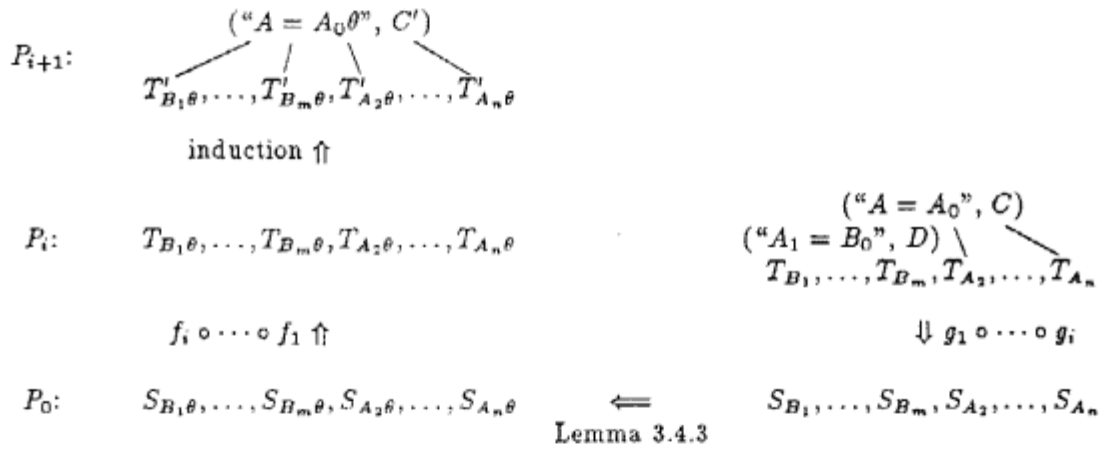


Figure 3.4.2 Definition of f_{i+1} for Case 2

Case 3 : C is folded.

Let D be the folding clause of the form

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

and C' be the result of folding. From Lemma 3.4.1, 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 :- $B_0\theta, A_{m+1}, \ldots, A_n$.

First, proof trees $S_{A_1}, \ldots, S_{A_m}$ of $A_1, \ldots, A_m$ by P_0 with the same answer substitutions as $T_{A_1}, \ldots, T_{A_m}$ are uniquely determined by applying $g_i, \ldots, g_1$ to $T_{A_1}, \ldots, T_{A_m}$. Let E_1 be the union of the label sets of $S_{A_1}, \ldots, S_{A_m}, T_{A_{m+1}}, \ldots, T_{A_n}$ and $\{A = A_0\}$. From Lemma 3.1.1, σ is the restriction of an m.g.u. of E_1 to the variables in A.

Second, from Lemma 3.4.2, 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 equations in the root labels, since $B_1\theta = A_1, \ldots, B_m\theta = A_m$ from folding condition (a). Let E_2 be the

union of the label sets of $S_{B_1},\ldots,S_{B_m}, T_{A_{m+1}},\ldots,T_{A_n}$ and $\{A = A_0, B_0\theta = B_0\}$. Then E_1 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_2 to the variables in A, since θ does not substitute for the variables in A. Note that from Lamma 3.4.3, $w(S_{A_1}) = w(S_{B_1}),\ldots,w(S_{A_m}) = w(S_{B_m})$.

Third, because the predicate of $B_0\theta$ is a new predicate, the clause used at the root of a proof tree of $B_0\theta$ by P_0 is in P_{new}. Further, from folding condition (c), this clause should be D. Let $S_{B_0\theta}$ be the proof tree obtained by putting a root node labelled with ("$B_0\theta = B_0$", D) over $S_{B_1},\ldots,S_{B_m}$. Then, proof tree $T_{B_0\theta}$ of $B_0\theta$ by P_i with the same answer substitution as $S_{B_0\theta}$ is uniquely determined by applying $f_1,\ldots,f_i$ to $S_{B_0\theta}$. Let E_3 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_3 to the variables in A.

Last, since P_i is weight complete and C satisfies folding condition (d),

$$w(T) > w(T_{A_1}) + \cdots + w(T_{A_n})$$

holds. In addition, since the predicate of $B_0\theta$ is a new predicate,

$$w(S_{B_0\theta}) = w(S_{B_1}) + \cdots + w(S_{B_m})$$

holds. Hence, from $w(T_{B_0\theta}) = w(S_{B_0\theta})$ and $w(S_{B_1}) = w(T_{A_1}),\ldots,w(S_{B_m}) = w(T_{A_m})$,

$$\begin{aligned} w(T) &> w(T_{A_1}) + \cdots + w(T_{A_n}) \\ &= w(T_{B_0\theta}) + w(T_{A_{m+1}}) + \cdots + w(T_{A_n}). \end{aligned}$$

holds. Thus $T \succ T_{B_0\theta}$ holds and $T \succ T_{A_j}$ holds for $j = m+1,\ldots,n$. By the inductive definition, $f_{i+1}(T_{B_0\theta}), f_{i+1}(T_{A_{m+1}}),\ldots,f_{i+1}(T_{A_n})$ are 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+1} with the same answer substitution as $T_{B_0\theta}, T_{A_{m+1}},\ldots,T_{A_n}$. Let T' be the proof tree obtained by putting a root node labelled with ("$A = A_0$", C') over $T'_{B_0\theta}, T'_{A_{m+1}},\ldots,T'_{A_n}$, and E' be the label set of T', i.e., the union of the label sets of $T'_{B_0\theta}$, $T'_{A_{m+1}},\ldots,T'_{A_n}$ and $\{A = A_0\}$. Then, from Lemma 3.1.1, σ is the restriction of an m.g.u. of E' to the variables in A. Hence, T' is a proof tree of A with answer substitution σ by P_{i+1}.

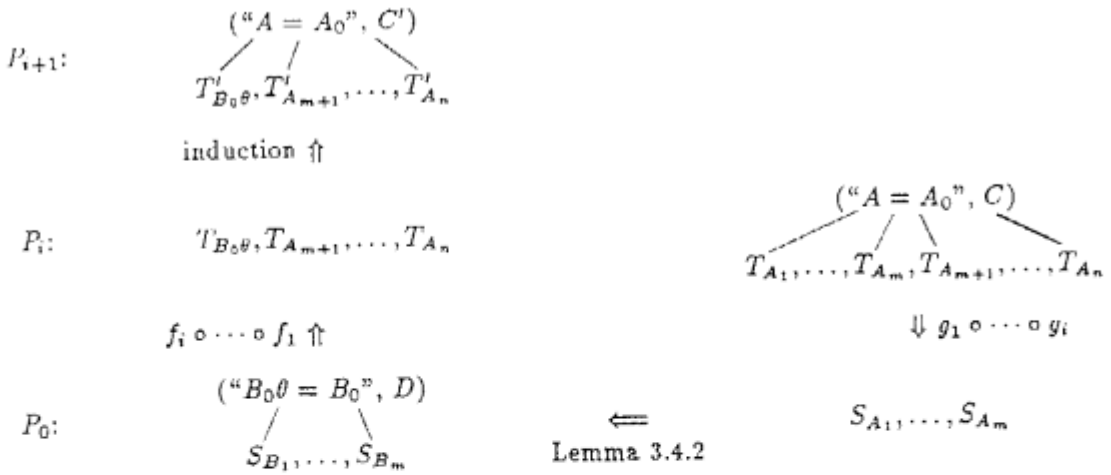


Figure 3.4.3 Definition of f_{i+1} for Case 3

Lemma 3.4.8 Let f_{i+1} be the mapping from $\mathcal{T}(P_i)$ to $\mathcal{T}(P_{i+1})$ defined in Lemma 3.4.7, and g_{i+1} be the mapping from $\mathcal{T}(P_{i+1})$ to $\mathcal{T}(P_i)$ defined in Lemma 3.3.4. Then (f_{i+1}, g_{i+1}) is a consistent mapping pair between $\mathcal{T}(P_i)$ and $\mathcal{T}(P_{i+1})$, and P_{i+1} is weight complete w.r.t. $(f_1, g_1), (f_2, g_2),\ldots,(f_{i+1}, g_{i+1})$.

Proof. Due to space limit, we will show the proof only for the most complicated "Case 3." (See Figure 3.3.3 and 3.4.3.) The other cases are proved in the same way.

The equality $g_{i+1} \circ f_{i+1}(T) = T$ for any proof tree T in $\mathcal{T}(P_i)$ is proved by induction on the well-founded ordering $\succ$. Let ("$A = A_0$", C) be the root label of T, and $T_{A_1}, \ldots, T_{A_m}$, $T_{A_{m+1}}, \ldots, T_{A_n}$ be the immediate subproofs of T. Suppose that, from P_i to P_{i+1}, clause C is folded to clause C' using instantiation θ. Following the definition of f_{i+1} in Figure 3.4.3, $T_{A_1}, \ldots, T_{A_m}$ uniquely determine proof tree $T_{B_0\theta}$. From the induction hypothesis,

$g_{i+1} \circ f_{i+1}(T_{B_0\theta}) = T_{B_0\theta}$,
$g_{i+1} \circ f_{i+1}(T_{A_{m+1}}) = T_{A_{m+1}}$,
$\vdots$
$g_{i+1} \circ f_{i+1}(T_{A_n}) = T_{A_n}$,

because $T \succ T_{B_0\theta}, T \succ T_{A_{m+1}}, \ldots, T \succ T_{A_n}$. This time, following the definition of g_{i+1} in Figure 3.3.3, $T_{B_0\theta}$ uniquely determines proof trees $T_{A_1}, \ldots, T_{A_m}$. Hence, $g_{i+1} \circ f_{i+1}(T) = T$.

The equality $f_{i+1} \circ g_{i+1}(T) = T$ for any proof tree T in $\mathcal{T}(P_{i+1})$ is proved similarly by induction on the structure of proof trees.

The weight completeness of P_{i+1} is easily proved from the definition of f_{i+1} in the proof of Lemma 3.4.7. Let ("$A = A_0$", C) be the clause used at the root of T, and $T_{A_1}, \ldots, T_{A_m}, T_{A_{m+1}}, \ldots, T_{A_n}$ be the immediate subproofs of T. Suppose that, from P_i to P_{i+1}, clause C is folded to clause C' using instantiation θ. Since P_i is weight complete and C satisfies folding condition (d),

$w(T) > w(T_{A_1}) + \cdots + w(T_{A_m}) + w(T_{A_{m+1}}) + \cdots + w(T_{A_n})$.

In addition, since the predicate of $B_0\theta$ is a new predicate,

$w(S_{B_0\theta}) = w(T_{B_1}) + \cdots + w(T_{B_m})$.

Then, from $w(T'_{B_0\theta}) = w(S_{B_0\theta}) = w(T_{A_1}) + \cdots + w(T_{A_m})$ and $w(T'_{A_{m+1}}) = w(T_{A_{m+1}}), \ldots, w(T'_{A_n}) = w(T_{A_n})$,

$w(T') = w(T) > w(T'_{B_0\theta}) + w(T'_{A_{m+1}}) + \cdots + w(T'_{A_n})$.

Theorem 3.4.9 Preservation of Success Multiset

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

Proof. Let $P_0, P_1, \ldots, P_N$ be a transformation sequence. If there exist consistent mapping pairs

$$\mathcal{T}(P_0) \underset{g_1}{\overset{f_1}{\rightleftharpoons}} \mathcal{T}(P_1) \underset{g_2}{\overset{f_2}{\rightleftharpoons}} \mathcal{T}(P_2) \underset{g_3}{\overset{f_3}{\rightleftharpoons}} \cdots \underset{g_i}{\overset{f_i}{\rightleftharpoons}} \mathcal{T}(P_i),$$

such that $P_0, P_1, \ldots, P_i$ are weight complete, from Lemma 3.3.4 and 3.4.7, there exist consistent mappings g_{i+1} from $\mathcal{T}(P_{i+1})$ to $\mathcal{T}(P_i)$ and f_{i+1} from $\mathcal{T}(P_i)$ to $\mathcal{T}(P_{i+1})$. Further, from Lemma 3.4.8, (f_{i+1}, g_{i+1}) is a consistent mapping pair between $\mathcal{T}(P_i)$ and $\mathcal{T}(P_{i+1})$, and P_{i+1} is weight complete w.r.t. $(f_1, g_1), (f_2, g_2), \ldots, (f_{i+1}, g_{i+1})$. From Lemma 3.4.6, initial program P_0 is weight complete. Then, by induction, there exists a consistent mapping pair (f_{i+1}, g_{i+1}) between $\mathcal{T}(P_i)$ and $\mathcal{T}(P_{i+1})$ for every $i = 0, 1, \ldots, N-1$. Hence, $\mathcal{M}(P_i) = \mathcal{M}(P_{i+1})$ holds for every $i = 0, 1, \ldots, N-1$.

The original result by Tamaki and Sato [8] [10] can be derived as a corollary.

Corollary 3.4.10 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 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 3.4.10, $\mathcal{M}(P)$ is preserved. Thus, the least Herbrand model is preserved.

4. Discussion

Preservation of success multiset widens the safe use of the Prolog programs obtained by Tamaki-Sato's transformation, which is not validated by preservation of least Herbrand model (cf. [5]). For example, consider the 'setof' and 'bagof' 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", and a call 'bagof(X,P,S)' means "S is the multiset of all instances of X such that P succeeds". Programs which are equivalent in the sense of the least Herbrand model semantics do not necessarily behave in the same way to the 'setof' and 'bagof' calls. For example, consider the following three programs P_1, P_2, P_3 again.

```
P1 : p(X).
     q(a).
P2 : p(a).
     q(a).
P3 : p(a).
     p(X) :- q(X).
     q(a).
```

Although these three programs are equivalent in the sense of the least Herbrand model semantics, to a query

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

P_2 and P_3 succeeds with answer substitution <X ⇐ a, Y ⇐ [a]>, while P_1 fails. Moreover, to a query

```
?- bagof(X,p(X),Y).
```

P_2 succeeds with answer substitution <X ⇐ a, Y ⇐ [a]>, P_3 succeeds with <X ⇐ a, Y ⇐ [a,a]>, and P_1 fails. However, when the success multisets of programs are identical, they behave in the same way to any 'setof' and 'bagof' calls if the calls stop. (Note that the success multisets of P_1, P_2 and P_3 are not identical.) Hence, we can safely use a predicate as an argument of 'setof' and 'bagof' 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 [9] [10]. 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 can compute the same answer substitutions the same number of times as the initial program for any given top-level goal.

Acknowledgements

This work is based on the result by Tamaki and Sato [8] [9] [10]. 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] Kawamura, T and T.Kanamroi, "Preservation of Stronger Equivalence in Unfold/Fold Logic Program Transformation", ICOT Technical Report, to appear, 1988.

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

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

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

[9] 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.

[10] 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.