ICOT Technical Report: TR-179

TR-179

Unfold/Fold Transformation of Logic Programs with Counters

by
Tadashi Kanamori and Hiroshi Fujita
(Mitsubishi Electric Corp.)

May, 1986

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

Unfold/Fold Transformation of Logic Programs with Counters

Tadashi KANAMORI Hiroshi FUJITA

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

Abstract

A refinement of Tamaki-Sato's transformation of Prolog programs is presented. When an initial definite clause program S_0 is transformed in sequence $S_0, S_1, \ldots, S_N$, we attach a counter of natural number to each definite clause. Roughly speaking, when execution of a ground atom in the minimum steps in S_i uses some definite clause with counter γ, it guarantees that the minimum number of execution steps in S_i is $\gamma - 1$ less than that in S_0. Using values of counters, we can not only give the condition for folding again but also characterize the class of improved execution, called *rank-consistent proof*, more precisely. We prove that S_0 and S_N are still equivalent in the sense of the minimum Herbrand model semantics in our framework. Then we show several further refinements as well as a slightly relaxed condition for safe use of *goal replacement rule*. We also discuss the source of the reduction of computation steps by program transformation.

Keywords : Program Transformation, Program Improvement, Prolog.

Contents

1. Introduction

In this paper, we present a refinement of Tamaki-Sato's transformation of Prolog programs [11]. When an initial definite clause program S_0 is transformed in sequence $S_0, S_1, \ldots, S_N$ using Tamaki-Sato's transformation, each definite clause in S_i is either marked "foldable" or unmarked, which plays an important role to judge whether folding is applicable or not. Roughly speaking, when execution of a ground atom in the minimum steps in S_i uses some definite clause marked "foldable", it guarantees that the minimum number of execution steps in S_i is less than that in S_0. Instead of the "foldable" marks, we attach a counter of natural number to each definite clause. Roughly speaking, when execution of a ground atom in the minimum steps in S_i uses some definite clause with counter γ, it guarantees that the minimum number of execution steps in S_i is $\gamma - 1$ less than that in S_0. By using values of counters, we can not only give the condition for folding again but also characterize the class of improved execution, called *rank-consistent proofs*, more precisely. We prove that S_0 and S_N are still equivalent in the sense of the minimum Herbrand model semantics in our framework. Then we show several further refinements as well as a slightly relaxed condition for safe use of *goal replacement rule*. We also discuss the sources of the reduction of compuattion steps by program transformation.

This paper is organized as follows. First in Section 2, we give an intuitive explanation of our basic method with least complication using a simple example. Then, we prove its correctness in Section 3. In Section 4, we show further refinements of the basic framework. Then, in Section 5, we show goal replacement rule and the condition whose applications are safe even if combined with the unfold/fold transformation. Lastly in Section 6, we point out that we can only expect $O(n)$ reduction of steps in the basic unfold/fold framework discuss the sources of the reduction of computation steps by program transformation.

In the following, we assume familiarity with the basic terminologies of first order logic such as term, atom (atomic formula or goal in this paper), substitution, most general unifier (m.g.u.) and so on. We also assume knowledge of the semantics of Prolog such as Herbrand interpretations and minimum Herbrand models. (see [1],[3],[4],[9]). We follow the syntax of DEC-10 Prolog [10]. Variables appearing in the body and not in the head of a definite clause are called *internal variables*. As syntactical variables, we use X, Y, Z for variables, $\mathbf{X}$,$\mathbf{Y}$ for sequences of variables, s, t for terms and A, B for atoms, possibly with primes and subscripts. In addition, we use σ, τ for substitutions.

2. Basic Unfold/Fold Transformation with Counters

2.1. Transformation Process

The entire process of our transformation proceeds in the completely same way as Tamaki-Sato's transformation [11] as follows.

P_0 :=the initial definite clause program ; $D_0 := \{\}$;
set each counter of definite clause in P_0 to 1;
for $i := 1$ **to** arbitrary N
    apply any of the transformation rules to obtain P_i and D_i from P_{i-1} and D_{i-1};

Figure 2.1. Transformation Process

Example 2.1. Before starting, the initial definite clause program is given, e.g.,

P_0 : C_1 [[1]]. append([ ],M,M).
C_2 [[1]]. append([X|L],M,[X|N]) :- append(L,M,N).

The numerals in [[ ]] denote the values of the counters. D_0 is initialized to {}. This example is used to illustrate the rules of transformation.

2.2. Basic Transformation Rules

The basic part of the transformation system consists of three rules, i.e., definition, unfolding and folding.

Definition : Let C be a definite clause of the form ($m \geq 0$)

$p(X_1,X_2,\ldots,X_n)$:- $A_1,A_2,\ldots,A_m$.

where

(a) p is an arbitrary predicate appearing neither in P_{i-1} nor in D_{i-1},
(b) $X_1, X_2, \ldots, X_n$ are distinct fresh variables and
(c) predicates of atoms in $A_1, A_2, \ldots, A_m$ all appear in P_0.

Then let P_i be $P_{i-1} \bigcup \{C\}$ and D_i be $D_{i-1} \bigcup \{C\}$. Let C have counter 1.

The predicates introduced by the definition rule are called *new predicates*, while those in P_0 are called *old predicates*.

Example 2.2.1. Suppose we have defined a relation *adjacent* without enough consideration of efficiency as follows.

C_3 [[1]]. adjacent(X,Y,LXYN) :- append(L,[X,Y],LXY),append(LXY,N,LXYN).

Then $P_1 = \{C_1, C_2, C_3\}$ and $D_1 = \{C_3\}$.

Unfolding : Let C be a definite clause in P_{i-1} with counter γ, A be an atom in the body and $C_1, C_2, \ldots, C_k$ be all the definite clauses in P_{i-1} whose heads are unifiable with A, say by m.g.u.'s $\sigma_1, \sigma_2, \ldots, \sigma_k$, and have counters $\gamma_1, \gamma_2, \ldots, \gamma_k$. Let C'_i be the result of replacing $\sigma_i(A)$ in $\sigma_i(C)$ with the body of $\sigma_i(C_i)$. Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'_1, C'_2, \ldots, C'_k\}$ and D_i be D_{i-1}. Let each C'_i have counter $\gamma_i + \gamma$ unless it is already in P_{i-1} with lower counter.

Example 2.2.2. When C_3 is unfolded at its first atom $append(L, [X, Y], LXY)$ in the body, we obtain $P_2 = \{C_1, C_2, C_4, C_5\}$ and $D_2 = \{C_3\}$ where

C_4 [[2]]. adjacent(X,Y,LXYN) :- append([X,Y],N,LXYN).
C_5 [[2]]. adjacent(X,Y,LXYN) :- append(L,[X,Y],LXY),append([Z|LXY],N,LXYN).

Then by unfolding C_4 three times and C_5 once, we obtain $P_6 = \{C_1, C_2, C_6, C_7\}$ and $D_6 = \{C_3\}$ where

C_6 [[5]]. adjacent(X,Y,[X,Y|L]).
C_7 [[3]]. adjacent(X,Y,[Z|LXYN]) :- append(L,[X,Y],LXY),append(LXY,N,LXYN).

Folding : Let C be a definite clause in P_{i-1} of the form

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

with counter γ and C_{folder} be a definite clause of the form

B :- $B_1,B_2,\ldots,B_m$.

such that

(a) C_{folder} is a definite clause in D_{i-1} (with counter 1).
(b) $1 \leq \gamma$.

Suppose there is a substitution σ and a subset $\{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}$ of the body of C such that the following conditions hold.

(a) $A_{i_j} = \sigma(B_j)$ for $j = 1, 2, \ldots, m$,

(b) σ substitutes distinct variables for the internal variables of C_{folder} and moreover those variables occur neither in A nor in $\{A_1, A_2, \ldots, A_n\} - \{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}$ and

(c) $m + 1 < n + \gamma$.

Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'\}$ and D_i be D_{i-1} where C' is a definite clause with head A and body $(\{A_1, A_2, \ldots, A_n\} - \{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}) \bigcup \{\sigma(B)\}$. Let C' have counter $\gamma - 1$.

Example 2.2.3. By folding the body of C_7 by C_3, we obtain $P_7 = \{C_1, C_2, C_6, C'_7\}$ and $D_7 = \{C_3\}$ where

C'7 [[2]]. adjacent(X,Y,[Z|L]) :- adjacent(X,Y,L).

Note that the counter was decremented by 1.

Though we do not prove it in detail, it is easy to see the following fact. Suppose that a definite clause in $P_i \bigcup D_i$ is of the form

A :- A1, A2, ..., An.

and with counter γ. Then $n + \gamma = 1$ holds for unit clauses, $n + \gamma > 1$ holds for other definite clauses and $\gamma \geq 0$ holds for all definite clauses. (It is trivial for $P_0 \bigcup D_0$. For the definition rule, $n + \gamma \geq 1$, since $\gamma = 1$. For the unfolding rule, the value $n + \gamma$ of C is incremented by the value $n_i + \gamma_i - 1$ of C_i. For the folding rule, $(n - m + 1) + (\gamma - 1) > 1$.)

2.3. Equivalence Preservation Theorem

The definite clause program P_0 given first is called the *initial program*. When the transformation process is stopped at some N, the program is transformed to definite clause program P_N and several definitions are accumulated in D_N. Then P_N is called the *final program* and D_N is called the *definition set* of the transformation process, sometimes denoted simply by D.

Example 2.3. If we stop the transformation process at step 6, we reach the final program and the definition set

P6 : C1 [[1]]. append([ ],M,M).
C2 [[1]]. append([X|L],M,[X|N]) :- append(L,M,N).
C6 [[6]]. adjacent(X,Y,[X,Y|L]).
C'7 [[2]]. adjacent(X,Y,[Z|L]) :- adjacent(X,Y,L).
D : C4 [[1]] adjacent(X,Y,LXYN) :- append(L,[X,Y],LXY),append(LXY,N,LXYN).

The most important property to be proved in Section 3 is the following theorem.

Theorem 2.3. The minimum Herbrand model of $P_0 \bigcup D$ is identical to that of P_N.

But in the following discussion, it is convenient to assume that all definitions in D is given from the beginning. To pretend it, for any transformation sequence $(P_0, D_0), (P_1, D_1), \ldots, (P_N, D_N)$, a sequence $S_0, S_1, \ldots, S_N$ is defined by $S_i = P_i \bigcup (D - D_i)$ and called *virtual transformation sequence*. (This is also due to Tamaki and Sato [11].) In particular $S_0 = P_0 \bigcup D$ and $S_N = P_N$. Since the definition rule is the identity transformation in the virtual transformation sequence, it is ignored when treating the virtual transformation sequence.

3. Preservation of Equivalence

In this section, after introducing basic notions in 3.1. and 3.2, we prove the equivalence preservation theorem along the same line as Tamaki and Sato [11].

3.1. Proof, Rank and Rank-Ordering of Ground Atom

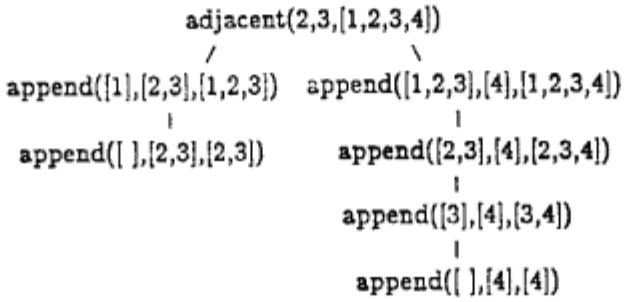
Let S be a definite clause program. A *proof tree*, or simply *proof*, of a ground atom A in S is a tree T labelled with ground atoms defined as follows.

(a) T is a proof of A in S when it is a tree consisting of a single node labelled with A, which is a ground instance of the head of a unit clause in S. (The unit clause is said to be *used at the root*.)

(b) Let $T_1, T_2, \ldots, T_m$ be immediate subtrees of T and $A_1, A_2, \ldots, A_m$ be their root labels. T is a proof of A in S when the root label of T is A, "A :- $A_1, A_2, \ldots, A_m$" is a ground instance of some definite clause in S and $T_1, T_2, \ldots, T_m$ are proofs of $A_1, A_2, \ldots, A_m$ in S, respectively. (The definite clause is said to be *used at the root* and $T_1, T_2, \ldots, T_m$ are called *immediate subproofs* of T.)

The set of all ground atoms that have proofs in S is exactly the minimum Herbrand model of S. We denote it by $M(S)$.

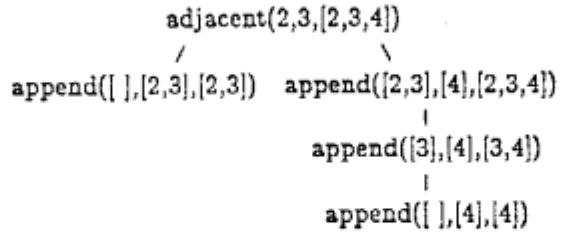
Example 3.1.1. Let *append* and *adjacent* be the predicates defined by $S_0 = P_0 \bigcup D$ in Example 2.2.1–2.2.3. Then $adjacent(2, 3, [1, 2, 3, 4])$ is in $M(S_0)$ and



is a proof in S_0.

Let A be a ground atom in $M(S_0)$. The rank of A is defined by the minimum size of proof of A in S_0 and denoted by $rank(A)$. Note that $rank(A) \geq 1$.

Example 3.1.2. As was shown in the example above, the rank of $adjacent(2, 3, [1, 2, 3, 4])$ is 7. The rank of $adjacent(2, 3, [2, 3, 4])$ is 5, because the following tree is the minimum proof in S_0.



The *rank ordering* is a well-founded ordering $\ll$ on the set of ground atoms $M(S_0)$. Let A and B be two ground atoms in $M(S_0)$. $A \ll B$ when $rank(A) < rank(B)$. One might think that this definition is abuse of notation. This is for the generalizations in Section 4.

Example 3.1.3. $adjacent(2, 3, [2, 3, 4]) \ll adjacent(2, 3, [1, 2, 3, 4])$ holds, because

rank(adjacent(2,3,[2,3,4])) = 5 < 7 = rank(adjacent(2,3,[1,2,3,4])).

3.2. Rank-Consistent Proof

Let S_i be a definite clause program. A proof T of a ground atom A in S_i is said to be *rank-consistent* when it satisfies either of the following conditions.

(a) T is a rank-consistent proof of A in S_i when it is a proof consisting of a single node labelled with a ground atom A, which is an instance of a unit clause in S_i.

(b) Let $T_1, T_2, \ldots, T_m$ be immediate subproofs of T, $A_1, A_2, \ldots, A_m$ be their root labels and C be the definite clause used at the root of T with counter γ. T is a rank-consistent proof of A in S_i when
   (i) $rank(A) = rank(A_1) + rank(A_2) + \cdots + rank(A_m) + \gamma$,
   (ii) $A \gg A_k$ for all k $(1 \leq k \leq m)$ and
   (iii) $T_1, T_2, \ldots, T_m$ are rank-consistent proofs of $A_1, A_2, \ldots, A_m$, respectively.

Note that the condition (ii) is redundant, because

$$\begin{aligned} \mathrm{rank(A)} &\geq \mathrm{rank(A_1)+rank(A_2)+\cdots+rank(A_m)+\gamma} \\ &\geq \mathrm{rank(A_k)+(n-1)+\gamma} \\ &> \mathrm{rank(A_k)}. \end{aligned}$$

This additional condition is for the generalizations in Section 4.

Example 3.2. Let S_6 be the definite clause program in Example 2.3. Then

adjacent(2,3,[1,2,3,4])
|
adjacent(2,3,[2,3,4])

is a rank-consistent proof in S_6, since

rank(adjacent(2,3,[1,2,3,4])) = 7 = 5+2 = rank(adjacent(2,3,[2,3,4])) + 2.

3.3. Proof of the Equivalence Preservation Theorem

In this section, we prove the equivalence preservation theorem. The following proof is, even textually, isomorphic to the one by Tamaki and Sato [11] (except the additional invariant I3) intentionally in order to emphasize the role of the counters.

Now we prove the following theorem.

Theorem 3.3. Let $S_1, S_2, \ldots, S_N$ be the virtual transformation sequence. Then $M(S_N) = M(S_0)$.

The proof of the theorem consists of showing that the following invariants hold for each i $(0 \leq i \leq N)$.

I1. $M(S_i) = M(S_0)$.
I2. For each ground atom A in $M(S_i)$, there is a rank-consistent proof of A in S_i.
I3. For any ground instance "A :- $A_1, A_2, \ldots, A_n$" of a definite clause with counter γ used at the root of a proof in S_i, $rank(A_1) + rank(A_2) + \cdots + rank(A_m) + \gamma \geq rank(A)$.

Base Case :

The first invariant I1 trivially holds for $i = 0$. As for the second invariant I2, for any ground atom A in $M(S_0)$, the smallest proof of A is obviously rank-consistent. (Remember $S_0 = P_0 \bigcup D$ and the counters of the clauses in P_0 and D are 1.) As for the third invariant

I3, note that the rank is the minimum size of proof in S_0.

Induction Step :

The preservation of the invariants is proved in the four lemmas below.

Lemma 3.3.1. If the invariant I1 holds for S_i, then $M(S_{i+1}) \subseteq M(S_i)$.

Proof. Let A be a ground atom in $M(S_{i+1})$ and T be its proof in S_{i+1}. We construct a proof T' of A in S_i by induction on the structure of T.

Let C be the definite clause used at the root of T and $T_1, T_2, \ldots, T_n$ $(n \geq 0)$ be the immediate subproofs of T. By induction hypothesis, we can construct proofs $T'_1, T'_2, \ldots, T'_n$ in S_i with each T'_j corresponding to T_j. If C is in S_i, we can immediately construct T' from C and the proofs $T'_1, T'_2, \ldots, T'_n$.

Suppose C is the result of unfolding. We can construct T' from $T'_1, T'_2, \ldots, T'_n$ using two definite clauses in S_i of which C is the unfolded result.

Suppose C is the result of folding. Then for some j $(1 \leq j \leq n)$, say $j = 1$, the root label A_1 of T_1 is an instance of the folded atom in the body of C. Because A_1 is provable in S_i by T'_1, it is also provable in S_0 by the invariant I1. So there should be a ground instance "$A_1 \text{ :- } B_1, B_2, \ldots, B_m$" of some definite clause such that $B_1, B_2, \ldots, B_m$ are provable in S_0. Again by I1, $B_1, B_2, \ldots, B_m$ are provable in S_i. Let C' be the definite clause in S_i of which C is the folded result. Owing to the condition of folding, we can combine the proofs of $B_1, B_2, \ldots, B_m$ and proofs $T'_2, T'_3, \ldots, T'_n$ with C' to obtain T', the proof of A in S_i.

Lemma 3.3.2. If the invariants I1, I2, and I3 hold for S_i, then $M(S_i) \subseteq M(S_{i+1})$.

Proof. Let A be a ground atom in $M(S_i)$. Then by the invariant I2, there is a rank-consistent proof T of A in S_i. We construct a proof T' of A in S_{i+1} by induction on the well-founded ordering $\gg$.

The base case where A is provable in S_0 itself and A has an old predicate obviously holds because then A should be a ground instance of some unit clause in P_0 which should be in both S_i and S_{i+1}.

Let C be the definite clause in S_i with counter γ used at the root of T and $T_1, T_2, \ldots, T_n$ $(n \geq 0)$ be the immediate subproofs of T. By the invariant I2, for each root label A_i of T_i, $A \gg A_i$ holds. So by the induction hypothesis there are proofs $T'_1, T'_2, \ldots, T'_n$ of $A_1, A_2, \ldots, A_n$ in S_{i+1}. If C is in S_{i+1}, the construction of T' is immediate.

Suppose C is unfolded into $C'_1, C'_2, \ldots, C'_k$ in S_{i+1} and assume that the root label A_1 of T_1 is the instance of the atom at which C is unfolded. Let $T_{11}, T_{12}, \ldots, T_{1s}$ be the immediate subproofs of T_1 and $A_{11}, A_{12}, \ldots, A_{1s}$ be their root labels. Then again by I2 and the induction hypothesis, there are proofs $T'_{11}, T'_{12}, \ldots, T'_{1s}$ of $A_{11}, A_{12}, \ldots, A_{1s}$ in S_{i+1}. Combining the proofs $T'_{11}, T'_{12}, \ldots, T'_{1s}, T'_2, \ldots, T'_n$ with some C'_l $(1 \leq l \leq k)$, we get a proof T' of A in S_{i+1}.

Now suppose C is folded into C' in S_{i+1}. Assume that C and C' have counters γ and $\gamma - \delta (\delta = 1)$ respectively and the root labels $A_1, A_2, \ldots, A_k$ of $T_1, T_2, \ldots, T_k$ $(k \leq n)$ are the instances of the folded atoms in C. Let B be an atom such that "$B \text{ :- } A_1, A_2, \ldots, A_k$" is a ground instance of the definite clause (with counter $\delta = 1$) used in the folding. Because the definite clause used in the folding is in D, $rank(A_1) + rank(A_2) + \cdots + rank(A_k) + \delta \geq rank(B)$ holds by the invariant I3 and B is provable in S_i by the equivalence of S_i to S_0. Because the condition (c) of folding is met, $k + \delta < n + \gamma$, hence

$$\begin{aligned} \text{rank(A)} &\geq \text{rank}(A_1) + \text{rank}(A_2) + \cdots + \text{rank}(A_n) + \gamma \\ &\geq \text{rank(B)} + (\gamma - \delta) + \text{rank}(A_{k+1}) + \text{rank}(A_{k+2}) + \cdots + \text{rank}(A_n) \end{aligned}$$

$\geq \text{rank(B)} + (\gamma - 1) + (n - k)$
$> \text{rank(B)}$

holds, which means $A \gg B$. Therefore by the induction hypothesis, B has a proof T_B in S_{i+1}. Combining the proofs $T_B, T'_{k+1}, \ldots, T'_n$ with the definite clause C', we obtain the proof T' of A in S_{i+1}.

Lemma 3.3.3. If the invariants I1, I2 and I3 hold for S_i, then I2 holds for S_{i+1}.

Proof. We first note that in the proof of Lemma 3.3.2, T' is constructed in such a way that it is rank-consistent. Thus every atom in $M(S_i)$ has a rank-consistent proof in S_{i+1}. Because $M(S_{i+1}) \subseteq M(S_i)$ by Lemma 3.3.1, I2 holds for S_{i+1}.

Lemma 3.3.4. If the invariants I2 and I3 holds for S_i, then I3 holds for S_{i+1}.

Proof. Let "A :- $A_1, A_2, \ldots, A_n$" be a ground instance of a definite clause C with counter γ used at the root of a proof in S_{i+1}. If C is in S_i, the lemma is obvious.

Suppose C is the result of unfolding. Then, by the induction hypothesis, there are two ground instances of definite clauses in S_i

$\text{A} \text{ :- } \text{B}, \text{A}_{k+1}, \text{A}_{k+2}, \ldots, \text{A}_m.$
$\text{B} \text{ :- } \text{A}_1, \text{A}_2, \ldots, \text{A}_k.$

with counter γ_1 and γ_2 such that

$\gamma = \gamma_1 + \gamma_2,$
$\text{rank(B)} + \text{rank}(\text{A}_{k+1}) + \text{rank}(\text{A}_{k+2}) + \cdots + \text{rank}(\text{A}_m) + \gamma_1 \geq \text{rank(A)}$
$\text{rank}(\text{A}_1) + \text{rank}(\text{A}_2) + \cdots + \text{rank}(\text{A}_k) + \gamma_2 \geq \text{rank(B)}$

From these inequalities, I3 holds obviously.

Suppose C is the result of folding. Then, by the induction hypothesis, there are two ground instances of definite clauses in S_i

$\text{A} \text{ :- } \text{A}_{11}, \text{A}_{12}, \ldots, \text{A}_{1k}, \text{A}_2, \ldots, \text{A}_m.$
$\text{A}_1 \text{ :- } \text{A}_{11}, \text{A}_{12}, \ldots, \text{A}_{1k}.$

with counter γ_1 and γ_2 such that

$\gamma = \gamma_1 - \gamma_2,$
$\text{rank}(\text{A}_{11}) + \text{rank}(\text{A}_{12}) + \cdots + \text{rank}(\text{A}_{1k}) + \text{rank}(\text{A}_2) + \cdots + \text{rank}(\text{A}_m) + \gamma_1 \geq \text{rank(A)}$

By the invariant I2, there is a rank-consistent proof of A_1 in S_i. Owing to the condition of folding, we can select the ground instance in such a way that

$\text{rank}(\text{A}_{11}) + \text{rank}(\text{A}_{12}) + \cdots + \text{rank}(\text{A}_{1k}) + \gamma_2 = \text{rank}(\text{A}_1)$

From these inequality and equality, I3 holds obviously.

This completes the proof of the theorem.

4. Several Refinements of the Basic Unfold/Fold Transformation

4.1. Introduction of A Static Ordering on Predicate Symbols

The basic framework presented in Section 2 has several limits in its application of the transformation rules, even if the application does not loose equivalence. One of them is shown by the following example.

Example 4.1. Let *old-p* be a predicate defined by

C_1 [[1]]. old-p(X) :- $\text{A}_1, \text{A}_2, \ldots, \text{A}_n$.

Suppose we have introduced a new predicate *new-p* by

C_2 [[1]]. new-p(X) :- $\text{A}_1, \text{A}_2, \ldots, \text{A}_n$.

Then we can't fold the body of C_1 by C_2.

First, we introduce a fixed ordering on predicate symbols. A predicate symbol p is greater than a predicate symbol q, denoted by $p \succ q$, when p is an old predicate and q is a new predicate. Then, only the folding rule is modified as follows.

Folding : Let C be a definite clause in P_{i-1} of the form

A :- A_1,A_2,...,A_n.

with counter γ and C_{folder} be a definite clause of the form

B :- B_1,B_2,...,B_m.

such that

(a) C_{folder} is a definite clause in D_{i-1} (with counter 1).

(b) $1 \leq \gamma$.

Suppose there is a substitution σ and a subset $\{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}$ of the body of C such that the following conditions hold.

(a) $A_{i_j} = \sigma(B_j)$ for $j = 1, 2, \ldots, m$,

(b) σ substitutes distinct variables for the internal variables of C_{folder} and moreover those variables occur neither in A nor in $\{A_1, A_2, \ldots, A_n\} - \{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}$ and

(c1) $m + 1 < n + \gamma$ or

(c2) $m + 1 = n + \gamma$ and $p \succ q$, where p and q are the predicate symbols of A and B respectively.

Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'\}$ and D_i be D_{i-1} where C' is a definite clause with head A and body $(\{A_1, A_2, \ldots, A_n\} - \{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}) \bigcup \{\sigma(B)\}$. Let C' have counter $\gamma - 1$.

As before, it is easy to see that, for any definite clause in $P_i \bigcup D_i$ of the form

A :- A_1, A_2, ..., A_n.

with counter γ, $n + \gamma \geq 1$ and $\gamma \geq 0$ hold.

The proof of the equivalence preservation theorem goes in the same way except the following points.

(a) The definition of the rank ordering is changed as follows : $A \gg B$ when (i) $rank(A) > rank(B)$ or (ii) $rank(A) = rank(B)$ and $p \succ q$, where p and q are the predicate symbols of A and B, respectively.

(b) The discussion to show $A \gg B$ in the proof of Lemma 3.3.2, i.e., $M(S_i) \subseteq M(S_{i+1})$ when the invariants I1, I2 and I3 hold for S_i, should be modified in accordance with this modification of the definition as follows.

Proof of Lemma 3.3.2. The proof proceeds in the same way as before except the proof of the case C is folded is modified as follows.

Now suppose C is folded into C' in S_{i+1}. Assume that C and C' have counters γ and γ' respectively and the root labels $A_1, A_2, \ldots, A_k$ of $T_1, T_2, \ldots, T_k$ ($k \leq n$) are the instances of the folded atoms in C. Let B be an atom such that "B :- $A_1, A_2, \ldots, A_k$" is a ground instance of the definite clause with counter δ used in the folding ($\gamma - \delta = \gamma'$). Because B is provable in S_i, it has a rank-consistent proof in S_i by the invariant I2. Hence, by the invariant I3, $rank(A_1) + rank(A_2) + \cdots + rank(A_k) + \delta \geq rank(B)$ holds. When the condition (c1) of folding is met, $k + \delta < n + \gamma$, hence

$$\begin{aligned} \text{rank(A)} &\geq \text{rank}(A_1) + \text{rank}(A_2) + \cdots + \text{rank}(A_n) + \gamma \\ &\geq \text{rank(B)} + (\gamma - \delta) + \text{rank}(A_{k+1}) + \text{rank}(A_{k+2}) + \cdots + \text{rank}(A_n) \\ &\geq \text{rank(B)} + (\gamma - \delta) + (n - k) \end{aligned}$$

$>$ rank(B)

holds, which means $A \gtrdot B$. When the condition (c2) of folding is met, $k = n$, $\delta = \gamma$, but the addition of ordering on predicate symbols means $A \gtrdot B$. Therefore by the induction hypothesis, B has a proof T_B in S_{i+1} in either case. Combining the proofs $T_B, T'_{k+1}, \ldots, T'_n$ with the definite clause C', we obtain the proof T' of A in S_{i+1}.

4.2. Introduction of Folding by Programs and A Dynamic Ordering on Predicate Symbols

One might think that it is too restrictive that folding should be done only using definitions, i.e., definite clauses in D_{i-1}.

Example 4.2. Suppose we have defined a predicate *sublist* as an old predicate by

```
sublist(M,LMN) :- append(L,M,LM),append(LM,N,LMN).
```

and the predicate *adjacent* by

```
adjacent(X,Y,LXYN) :- sublist([X,Y],LXYN).
```

Then, after the transformation similar to Example 2.2.1–2.2.3, we would like to fold

```
C7 [[4]]. adjacent(X,Y,[Z|LXYN]) :- append(L,[X,Y],LXY),append(LXY,N,LXYN).
```

by the definite clause above to

```
C'9 [[3]]. adjacent(X,Y,[Z|L]) :- sublist([X,Y],L).
```

In order to permit folding by programs, we generalize the ordering in Section 4.1 to an arbitrary ordering between predicate symbols and keep it during the transformation process.

P_0 :=the initial definite clause program ;
$D_0 := \{\}$;
R_0 :=any ordering on predicate symbols appearing in P_0;
set each counter of definite clause in P_0 to 1;
for $i := 1$ **to** arbitrary N
  apply any of the transformation rules to obtain P_i,D_i and R_i from P_{i-1},D_{i-1} and R_{i-1};

Figure 4.2. Transformation Process

Throughout the transformation process, we have an ordering relation R_i on predicate symbols appearing in P_i and D_i such that $R_0 \subseteq R_1 \subseteq \cdots \subseteq R_i \subseteq \cdots$. We denote them by a commom infix notation $\succ$. The initial ordering R_0 might be empty. The final ordering R_N depends on R_0, the arrangement at user's disposal in the definition rule and the history of the applications of the folding rule. Following this extension, the definition rule and the folding rule are modified as follows.

Definition : Let $C_1, C_2, \ldots, C_k$ be definite clauses of the form ($m \geq 0$)

```
p(t1,t2,...,tn) :- A1,A2,...,Am.
```

where

(a) p is an arbitrary predicate appearing neither in P_{i-1} nor in D_{i-1} and
(b) the heads of $C_1, C_2, \ldots, C_k$ are not unifiable each other.

Then let P_i be $P_{i-1} \bigcup \{C_1, C_2, \ldots, C_k\}$ and D_i be $D_{i-1} \bigcup \{C_1, C_2, \ldots, C_k\}$. Let $C_1, C_2, \ldots, C_k$ have counter 1. Let R_i be $R_{i-1} \bigcup \{\ p \succ q$ or $q \succ p \mid$ you wish to assume it for $q\ \}$ as far as the transitive closure of R_i is irreflexive.

Folding : Let C be a definite clause in P_{i-1} of the form

A :- A_1,A_2,...,A_n.
with counter γ and C_{folder} be a definite clause in $P_{i-1} \bigcup D_{i-1}$ of the form
B :- B_1,B_2,...,B_m.
with counter δ such that

(a1) C_{folder} is a definite clause in D_{i-1} or
(a2) C_{folder} is a definite clause in P_{i-1} for which there is no other definite clause in P_{i-1} whose head is unifiable with B.
(b) $\delta \leq \gamma$.

Suppose there is a substitution σ and a subset $\{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}$ of the body of C such that the following conditions hold.

(a) $A_{i_j} = \sigma(B_j)$ for $j = 1, 2, \ldots, m$,
(b) σ substitutes distinct variables for the internal variables of C_{folder} and moreover those variables occur neither in A nor in $\{A_1, A_2, \ldots, A_n\} - \{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}$ and

(c1) $m + \delta < n + \gamma$ or
(c2) $m + \delta = n + \gamma$ and the transitive closure of $R_{i-1} \bigcup \{p \succ q\}$ is irreflexive, where p and q are the predicate symbols of A and B respectively.

Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'\}$ and D_i be D_{i-1} where C' is a definite clause with head A and body $(\{A_1, A_2, \ldots, A_n\} - \{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}) \bigcup \{\sigma(B)\}$. Let C' have counter $\gamma - \delta$. Let R_i be R_{i-1} when the condition (c1) is met and let R_i be $R_{i-1} \bigcup \{p \succ q\}$ when the condition (c2) is met.

As before, for any definite clause in $P_i \bigcup D_i$ of the form
A :- A_1, A_2, ..., A_n.
with counter γ, $n + \gamma \geq 1$ and $\gamma \geq 0$ hold.

The proof of the equivalence preservation theorem goes in the completely same way as in Section 4.1.

4.3. Introduction of Folding by Previous Programs and Negative Counters

According to the framework so far, we must fold by some definite clause in $P_{i-1} \bigcup D_{i-1}$ and keep counters of definite clauses non-negative.

Example 4.3.1. Let us consider the transformation in Example 4.2 again. Suppose that a predicate *sublist* is defined in P_0 by
sublist(M,LMN) :- append(L,M,LM), append(LM,N,LMN).
and we have defined the predicate *adjacent* by
adjacent(X,Y,LXYN) :- sublist([X,Y],LXYN).
Then by unfolding at $sublist([X, Y], LXYN)$, we have
C' [[2]] adjacent(X,Y,LXYN) :- append(L,[X,Y],LXY), append(LXY,N,LXYN).
in P_2. By continuing the same transformation as in Example 2.2.1–2.2.3, we have
adjacent(X,Y,[Z|LXYN]) :- append(L,[X,Y],LXY), append(LXY,N,LXYN).
If folding by C' in P_2 is allowed, we can immediately have
adjacent(X,Y,[Z|LXYN]) :- adjacent(X,Y,LXYN).

Example 4.3.2. Suppose that an old predicate *old-p* is defined in P_{i-1} by
[[1]] old-p(X) :- A_1, A_2, ..., A_n, B_1, B_2, ..., B_m.
new predicates *pa* and *pb* are defined in D_{i-1} by
[[1]] pa(X) :- A_1, A_2, ..., A_n.
[[1]] pb(X) :- B_1, B_2, ..., B_m.

We can fold the body of *old-p* by $pa(X)$ to get

$[\![0]\!]$ old-p(X) :- pa(X), B_1, B_2, ..., B_m.

But folding by $pb(X)$ is not allowed, because the resulting counter is -1.

A close examination of the proof in Section 3 and Section 4.1 reveals that both the condition that C_{folder} be in $P_{i-1} \bigcup D_{i-1}$ and the condition $\gamma \geq 0$ are not crucial. Because definite clauses in D_{i-1} always appear in some P_j $(j < i)$, the condition of C_{folder} is unified as follows.

Folding : Let C be a definite clause in P_{i-1} of the form

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

with counter γ and C_{folder} be a definite clause in P_j $(j < i)$ of the form

B :- $B_1, B_2, \ldots, B_m$.

with counter δ for which there is no other definite clause in P_j whose head is unifiable with B. Suppose there is a substitution σ and a subset $\{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}$ of the body of C such that the following conditions hold.

(a) $A_{i_j} = \sigma(B_j)$ for $j = 1, 2, \ldots, m$,

(b) σ substitutes distinct variables for the internal variables of C_{folder} and moreover those variables occur neither in A nor in $\{A_1, A_2, \ldots, A_n\} - \{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}$ and

(c1) $m + \delta < n + \gamma$ or

(c2) $m + \delta = n + \gamma$ and the transitive closure of $R_{i-1} \bigcup \{p \succ q\}$ is irreflexive, where p and q are the predicate symbols of A and B respectively.

Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'\}$ and D_i be D_{i-1} where C' is a definite clause with head A and body $(\{A_1, A_2, \ldots, A_n\} - \{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}) \bigcup \{\sigma(B)\}$. Let C' have counter $\gamma - \delta$. Let R_i be R_{i-1} when the condition (c1) is met and let R_i be $R_{i-1} \bigcup \{p \succ q\}$ when the condition (c2) is met.

Now, for any definite clause in $P_i \bigcup D_i$ of the form

A :- A_1, A_2, ..., A_n.

with counter γ, $n + \gamma \geq 1$ holds.

The proof of the equivalence preservation theorem goes in the completely same way as in Section 4.1.

5. Goal Replacement in the Refined Unfold/Fold Transformation

In our unfold/fold transformation system with counters, the goal replacement rule in Tamaki-Sato's framework can be considered a generalization of unfolding or folding. It is applied under a slightly relaxed condition very similar to those for unfolding or folding.

Let S_i be a definite clause program and $\exists\ \mathbf{X}\ (B_1 \wedge B_2 \wedge \cdots \wedge B_n)$ be an existentially quantified conjunction of atoms without free variables. (By $\mathbf{X}$ we represent a vector of variables.) We say that the formula is provable in S_i and write

$S_i \vdash \exists\ \mathbf{X}\ (B_1 \wedge B_2 \wedge \cdots \wedge B_n)$

if there is some ground instantiation θ of $\mathbf{X}$ such that every $\theta(B_i)$ $(1 \leq i \leq n)$ is provable in S_i. By

rank($\exists \mathbf{X}\ (B_1 \wedge B_2 \wedge \cdots \wedge B_n)$)

we represent the minimum of $rank(\sigma(B_1)) + rank(\sigma(B_2)) + \cdots + rank(\sigma(B_n))$ for every ground instantiation σ of $\mathbf{X}$.

Example 5.1. Suppose that *append* is defined as before in S_i. Then

∃ LM (append([1],[2],LM)∧append(LM,[3],[1,2,3]))
is provable in S_i and
rank(∃ LM (append([1],[2],LM)∧append(LM,[3],[1,2,3]))) = 5.

Goal Replacement : Let C be a definite clause in P_{i-1} of the form
A :- A_1,A_2,...,A_k, B_1,B_2,...,B_n.
with counter γ and C' be a definite clause (not in P_{i-1}) of the form
A :- A_1,A_2,...,A_k, B'_1,B'_2,...,B'_m.
Let $\mathbf{X}$ be variables occurring in $B_1, B_2, \ldots, B_n$ and not in $A_1, A_2, \ldots, A_k$, $B'_1, B'_2, \ldots, B'_m$. Similarly, let $\mathbf{Y}$ be variables occurring in $B'_1, B'_2, \ldots, B'_m$ and not in $A_1, A_2, \ldots, A_k, B_1, B_2, \ldots, B_n$. Suppose that, for every ground instantiation θ of $A, A_1, A_2, \ldots, A_k$, the following conditions are satisfied.

(a) $P_{i-1} \bigcup D_{i-1} - \{C\} \vdash \exists\, \mathbf{X}\ \theta(B_1 \wedge B_2 \wedge \cdots \wedge B_n)$ if and only if $P_{i-1} \bigcup D_{i-1} - \{C\} \vdash \exists\, \mathbf{Y}\ \theta(B'_1 \wedge B'_2 \wedge \cdots \wedge B'_m)$,
(b) $rank(\exists \mathbf{X} \theta(B_1 \wedge B_2 \wedge \cdots \wedge B_n)) \geq rank(\exists \mathbf{Y} \theta(B'_1 \wedge B'_2 \wedge \cdots \wedge B'_m)) + \delta$ and
(c) $(\gamma - \delta) + (k + m) > 1$.

Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'\}$ and D_i be D_{i-1}. Let C' have counter $\gamma + \delta$. Let R_i be R_{i-1}.

Example 5.2. Suppose that *append* is defined as before in S_i. Consider the following existential quantified conjunctions.
∃ LM (append(L,M,LM)∧append(LM,N,LMN)),
∃ MN (append(M,N,MN)∧append(L,MN,LMN)).
Then, for any ground instantiation $\theta = < L \Leftarrow t_L, M \Leftarrow t_M, N \Leftarrow t_N, LMN \Leftarrow t_{LMN} >$,
$S_i \vdash$ ∃ LM (append(t_L,t_M,LM)∧append(LM,t_N,t_{LMN})),
if and only if
$S_i \vdash$ ∃ MN (append(t_M,t_N,MN)∧append(t_L,MN,t_{LMN})).
Hence, the condition (a) holds. As for the condition (b),
rank(∃ LM (append(t_L,t_M,LM)∧append(LM,t_N,t_{LMN})))
≥ rank(append(t_M,t_N,MN)∧append(t_L,MN,t_{LMN})) + 0 .
Or, more exactly speaking,
rank(∃ LM (append(t_L,t_M,LM)∧append(LM,t_N,t_{LMN})))
≥ rank(append(t_M,t_N,MN)∧append(t_L,MN,t_{LMN})) + length(t_L).
Consider a definite clause
rev2([X|L],N,M) :- reverse(L,K), append(K,[X],KX),append(KX,N,M).
with counter 2. Because the condition (c) holds ($k = 1, n = 2, m = 2, \gamma = 2, \delta = 0$), we can apply goal replacement to get
rev2([X|L],N,M) :- reverse(L,K), append([X],N,XN),append(K,XN,M).
with counter 2 (or, more exactly speaking, $2 + length(K)$, if such an expression is allowed).

The goal replacement itself preserves the minimum Herbrand model even without the condition (b) and (c). But it was pointed out by Tamaki and Sato [11] that the second invariant I2, hence equivalence, might be lost, when it is used within the unfold/fold system without these conditions. Because of the use of counters, the class of goal replacement considered legal is larger than that in the original Tamaki-Sato's transformation system.

Example 5.3. Let the initial program P_0 be
P_0 : C_1 [[1]]. q(s(X)) :- q(X).
C_2 [[1]]. q(0).
C_3 [[1]]. r(s(X)) :- r(X).

C_4 [1]. r(0).

Suppose we have defined

C_5 [1]. p1(X,Y) :- q(X),r(Y).

C_6 [1]. p2(X,Y) :- q(X),r(Y).

Then we can replace $q(X)$ in the body of C_6 with $q(s(X))$, because

$\{C_1, C_2, C_3, C_4, C_5\} \vdash$ q(X) iff $\{C_1, C_2, C_3, C_4, C_5\} \vdash$ q(s(X)),

rank(q(X)) $\geq$ rank(q(s(X)))-1.

Similarly, we can replace $r(Y)$ in the body of C_5 with $r(s(Y))$. (In our transformation, these goal replacement can be considered folding.) Hence, the following transformation sequence is allowed.

Unfold $q(X)$ in C_5

C_7 [2]. p1(0,Y) :- r(Y).

C_8 [2]. p1(s(X),Y) :- q(X),r(Y).

Replace $r(Y)$ with $r(s(Y))$

C_9 [1]. p1(s(X),Y) :- q(X),r(s(Y)).

(Note that the counter is decremented by 1, because $rank(r(Y)) \geq rank(r(s(Y))) - 1$).

Unfold $r(Y)$ in C_6

C_{10} [2]. p2(X,0) :- q(X).

C_{11} [2]. p2(X,s(Y)) :- q(X),r(Y).

Replace $q(X)$ with $q(s(X))$

C_{12} [1]. p2(X,s(Y)) :- q(s(X)),r(Y).

(Note that the counter is decremented by 1, because $rank(q(X)) \geq rank(q(s(X))) - 1$).

In order to show that unrestricted goal replacement looses equivalence when combined with the unfold/fold rules, Tamaki and Sato folded C_9 and C_{12} here to derive an inequivalent program as follows.

Fold C_9.

C_{13} [0]. p1(s(X),Y) :- p2(X,s(Y)).

Fold C_{12}.

C_{14} [0]. p2(X,s(Y)) :- p1(s(X),Y).

The resulting program contains infinite recursion and is not equivalent to the original one. In Tamaki-Sato's framework, this is because the goal replacement steps destroyed the invariant I2. In our framework, the goal replacement step keeps the invariant I2 and the first folding of C_9 is allowed, because the condition (c2) is met. ($m = 1, n = 1, \gamma = 1, \delta = 1$ and $m + \gamma = n + \delta$ holds. Note that $p1 \succ p2$ is added.) But the second folding of C_{12} is not allowed even in our transformation, because adding $p2 \succ p1$ violates the irreflexivity of $\succ$.

But, because folding by programs is allowed in our framework, additional care is necessary for the folding rule. Application of goal replacement might destroy the invariant I3, which is necessary for the definite clause used as folders. We must restrict the folder to be *goal-replacement-independent*. Intuitively speaking, a definite clause is goal-replacement-indenpendent if it has no relation, either directly or indirectly, with goal replacement. More formally, a definite clause is said to be *goal-replacement-independent* when it is not *goal-replacement-dependent*. A definite clause C is said to be *goal-replacement-dependent* when

(a) C is a result of folding or

(b) C is a result of unfolding applied to a goal-replacement-dependent definite clause or using a goal-replacement-dependent definite clause.

Note that a definite clause in D_{i-1} is always goal-replacement-independent.

Folding : Let C be a definite clause in P_{i-1} of the form

A :- A_1,A_2,...,A_n.

with counter γ and C_{folder} be a goal-replacement-independent definite clause in P_j $(j < i)$ of the form

B :- B_1,B_2,...,B_m.

with counter δ for which there is no other definite clause in P_j whose head is unifiable with B. Suppose there is a substitution σ and a subset $\{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}$ of the body of C such that the following conditions hold.

(a) $A_{i_j} = \sigma(B_j)$ for $j = 1, 2, \ldots, m$,

(b) σ substitutes distinct variables for the internal variables of C_{folder} and moreover those variables occur neither in A nor in $\{A_1, A_2, \ldots, A_n\} - \{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}$ and

(c1) $m + \delta < n + \gamma$ or

(c2) $m + \delta = n + \gamma$ and the transitive closure of $R_{i-1} \bigcup \{p \succ q\}$ is irreflexive, where p and q are the predicate symbols of A and B respectively.

Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'\}$ and D_i be D_{i-1} where C' is a definite clause with head A and body $(\{A_1, A_2, \ldots, A_n\} - \{A_{i_1}, A_{i_2}, \ldots, A_{i_m}\}) \bigcup \{\sigma(B)\}$. Let C' have counter $\gamma - \delta$. Let R_i be R_{i-1} when the condition (c1) is met and let R_i be $R_{i-1} \bigcup \{p \succ q\}$ when the condition (c2) is met.

In general, if our conditions for goal replacement are observed, a rank-consistent proof in S_i can be converted into a rank-consistent proof in S_{i+1}.

6. Source of Optimization in Unfold/Fold Transformation

What is the source of optimization in unfold/fold transformation ? There can be two sources as follows.

(a) One of the most prominent features of Prolog is its ability to describe nondeterminism. Because such description is allowed, we can expect that, programs are once written in nondeterministic way and then transformed to more deterministic efficient one. That is, program transformation reforms the search trees of the Prolog interpreter to the ones with less OR-branching.

(b) Program transformation sometimes reduces the steps to reach the solution even if the initial program is deterministic. That is, program transformation reforms the search trees of the Prolog interpreter to the ones with shorter paths to the solution leaves.

Example 6.1. Suppose that *sort* is given using the typical nondeterministic description in Prolog as follows.

sort(L,M) :- permutation(L,M), ordered(M).

This is a well-known example, in which the nondeterminitic program is converted to the deterministic one. Deriving *insertion-sort* from such description is sometimes cited as a program optimization from $O(n!)$ to $O(n^2)$. But, note that, if the nondeterministic Prolog interpreter always selects correct OR-branches, we can succeed in $O(n^2)$ steps even with the program above. Hence the order of the rank of $sort(s, t)$ is not changed.

Example 6.2. Let us consider the example in Section 2. There, the description of the initial program of *adjacent*

adjacent(X,Y,LXYN) :- append(L,[X,Y],LXY),append(LXY,N,LXYN).

is nondeterministic as well. Note that the transformation sequence there actually reduced the rank, but the order of the rank is $O(n)$ both in P_0 and in P_7.

Now let us focus our attention to the latter source of optimization measured by the rank. The preciseness of the use of counters sheds light on an unexpected fact. As far as we

are within the unfold/fold transformation in Section 2 or Section 4.1, where the predicate symbols are two-layered, i.e., there are only old predicates and new predicates and the new predicates are defined using only the old predicates, the reduction of the computation steps is at most $O(n)$, because the values of counters are constant.

Example 6.3. One might be suspicious of this claim, because the derivation of *rev*2 of $O(n)$ from *reverse* of $O(n^2)$ is a well-known example. Why can we reach the $O(n)$ algorithm from the $O(n^2)$ one if the reduction of steps is $O(n)$? The source is the use of goal replacement

$\exists$ LM (append(L,M,LM)$\wedge$append(LM,N,LMN))
$\Rightarrow \exists$ MN (append(M,N,MN)$\wedge$append(L,MN,LMN)),

where the number of steps proportional to the length of L is reduced. By accumulating these reduced steps, we have the $O(n^2)$ reduction as a whole.

This fact means that the framework with two layers is far from satisfation and we have to remedy it somehow.

One way is dividing the transformation sequence into phases. After the k-th phase is finished, all predicates in the final program $P_{N_k}^{(k)}$ of the k-th phase are considered old predicates in the initial program $P_0^{(k+1)}$ of the $(k+1)$-th phase. Then, starting from the phase 1, the reduction of computation steps in the k-th phase is at most $O(n^k)$ compared with the very beginning $P_0^{(1)}$.

Another way is dividing the predicate symbols into levels as was done by Tamaki and Sato [12]. The predicate symbols in the level k are defined using those with level less than or equal to k. The predicate symbols in the level k are considered greater than those with level less than k. Then, starting from level 0, the reduction of computation steps of the predicate with level k is at most $O(n^k)$.

7. Discussion

The result in this paper is just a refinement of the work by Tamaki and Sato [11]. Our new contributions are the following three points.

First, we generalized the "foldable" marks to the counters of natural numbers so that we can characterize the class of improved execution more precisely. Our evaluation of the number of reduced execution steps is very precise as far as we are within the unfold/fold transformation (without goal replacement). The use of counters enables us to relax the conditions for folding and goal replacement.

Secondly, we introduced several generalizations.

(a) We generalized the rank ordering $\gg$ by introducing orderings on predicate symbols. Intuitively speaking, critical situations to loose equivalence by folding occur only if the folding creates recursions. We keep an ordering R_i to check the possibility of generating critical recursions.
(b) We made it possible to fold using definite clauses in program as far as the head is not unifiable with other heads.
(c) We allowed folding by programs in previous versions and definite clauses with negative counters.

Now, we need no distinction of old predicates and new predicates at all. We believe that almost everything legal is allowed except a few exceptions.

Thirdly, our approach would be a first step toward more precise evaluation of improvement by program transformation. We expect that our definition of the rank is suitable for this purpose. Though one of the purposes of program transformation is improvement in execution efficiency, we have not yet have enough tool to evaluate improvement by transformation. Because the most drastic improvement in execution efficiency usually occurs when recursions are formed by folding, we need further work on analysis of such recursion-formation-time behavior.

8. Conclusions

We have presented a refinement of unfold/fold transformation of logic programs, which takes the number of improved execution steps and the possibilities of forming critical recursions into consideration more precisely. This method is being used in Argus/C, a system for construction of Prolog programs under development [6],[7],[8].

Acknowledgements

This work is an extension of the result by Tamaki and Sato [11]. The authors would like to express deep gratitude to Prof.H.Tamaki (Ibaraki University) for his valuable suggestion.

Our construction system Argus/C under development is a subproject of the Fifth Generation Computer System(FGCS) "Intelligent Programming System". The authors would like to thank Dr.K.Fuchi (Director of ICOT) for the opportunity of doing this research and Dr.T.Yokoi (Chief of ICOT 2nd Laboratory) and Dr.K.Furukawa (Chief of ICOT 1st Laboratory) for their advice and encouragement.

References

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

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

[3] Clark,K.L., "Predicate Logic as A Computational Formalism", Chap.5, Research Monograph : 79/59, TOC, Imperial College, 1979.

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

[5] Hogger,C.J., "Derivation of Logic Programs", J.ACM, Vol.28, No.2, pp.372-392, 1981.

[6] Kanamori,T.and K.Horiuchi, "Construction of Logic Programs Based on Generalized Unfold/Fold Rules", ICOT TR-1??, to appear, 1986.

[7] Kanamori,T.and H.Fujita, "Unfold/Fold Transformation of Logic Programs with Counters", ICOT TR-1??, to appear, 1986.

[8] Kanamori,T.and M.Maeji, "Derivation of Logic Programs from Implicit Definition", ICOT TR-1??, to appear,1986.

[9] Lloyd,J.W., "Foundations of Logic Programming", TR 82/7,Department of Computer Science, University of Melbourne, 1982.

[10] Pereira,L.M.,F.C.N.Pereira and D.H.D.Warren, "User's Guide to DECsystem-10 Prolog", Occasional Paper 15, Dept.of Artificial Intelligence, Edinburgh, 1979.

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

[12] Tamaki,H.and T.Sato, "A Generalized Correctness Proof of the Unfold/Fold Logic Program Transformation", to appear, 1986.