TR-177

Construction of Logic Programs Based on Generalized Unfold/Fold Rules

by

Tadashi Kanamori and Kenji Horiuchi

(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

Construction of Logic Programs Based on Generalized Unfold/Fold Rules

Tadashi KANAMORI   Kenji HORIUCHI

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

Abstract

A method to construct logic programs based on generalized unfold/fold rules is described. Though the method itself is not novel, we prove its correctness, that is, when a definite clause program P_N is constructed from a definite clause program P_0 introducing definitions D of new procedures in some class of formulas, the minimum Herbrand model of P_N is identical to that of $P_0 \bigcup D$. This is a generalization of the equivalence preservation theorem for Tamaki-Sato's transformation as well as a partial justification of the method presented by Clark. We also present splitting rules as an example of safe augmenting rules, use of which still preserves minimum Herbrand models even if combined with the unfold/fold rules.

Keywords : Program Synthesis, Program Transformation, Prolog.

Contents

1. Introduction

The unfold/fold rules were advocated as basic transformation rules for functional programs by Burstall and Darlington [3]. It was not clear whether and when combinations of unfolding and folding preserve the equivalence of functional programs, which was later investigated theoretically by Kott [13] and Scherlis [20],[21]. The unfold/fold approach was also extended to Prolog programs by Clark [5]. He permitted more general first order formulas as initial specifications, of which program transformation can be regarded as a special case. The preservation of equivalence in Prolog program transformation (in the sense of the minimum Herbrand model semantics) was investigated by Tamaki and Sato [24].

Suppose we have an initial Prolog program P_0 and some specification D of new procedures in some class of first order formulas and we can well-define the completion ([4],[1]) and the minimum Herbrand model of $P_0 \bigcup D$. In general, construction of a Prolog program is to derive a set of logical consequences P_N from $P_0 \bigcup D$, which is the theoretical basis of the approach by Clark [5] and Hogger [10]. But there can still hold various relations between $P_0 \bigcup D$ and P_N. The tightest relation is the one that the completion of $P_0 \bigcup D$ and that of P_N are logically equivalent. Though such construction still plays an important role, the most interesting optimizations usually loose the equivalence of completions. The loosest relation between $P_0 \bigcup D$ and P_N is the one that we can say nothing more than that $P_0 \bigcup D$ is stronger than P_N. But in such a case, we have to check whether the constructed Prolog program actually computes the specified relation exactly after having constructed it (see [5] p.97,pp.102-105,[8] p.16). The result by Tamaki and Sato [24] is located between them. They proved that every ground atom which is provable from axioms $P_0 \bigcup D$ is also provable from axioms P_N. That is, the minimum Herbrand model of $P_0 \bigcup D$ is identical to that of P_N in their Prolog program transformation, though it does not necessarily preserve the equivalence of completions.

In this paper, we show a construction method based on generalized unfold/fold rules, which includes Tamaki-Sato's transformation and is included in the class of construction presented by Clark [5]. Though the method itself is not novel, we prove its correctness along the same line by Tamaki and Sato. That is, when a definite clause program P_N is constructed from a definite clause program P_0 introducing definitions D of new procedures in some class of formulas, the minimum Herbrand model of P_N is identical to that of $P_0 \bigcup D$. This is a generalization of the equivalence preservation theorem for Tamaki-Sato's transformation as well as a partial justification of Clark's method.

This paper is organized as follows. After preparing preliminary materials in Section 2, we show our construction method using a simplest example in Section 3. In Section 4, we define two notions, rank ordering and rank-consistent proof of ground atoms, based on a well-founded ordering on multisets of formulas in some class. Using them, we prove the equivalence preservation theorem, which is the main purpose of this paper. In Section 5, we show splitting rules as an example of safe augmenting rules, use of which still preserves minimum Herbrand models even if combined with the unfold/fold rules. Finally in Section 6, we discuss the relations to other works.

2. Preliminaries

In the following, we assume familiarity with the basic terminologies of first order logic such as term, atom (atomic formula), positive and negative literals, formula, substitution, most general unifier (m.g.u.) and so on. We also assume knowledge of the semantics of

Prolog such as completion, minimum Herbrand model and transformation T of Herbrand interpretations (see [1],[4],[5],[7],[14]). We follow the syntax of DEC-10 Prolog [17]. As syntactical variables, we use X, Y, Z for variables, s, t for terms, A, B for atoms and $\mathcal{F}, \mathcal{G}, \mathcal{H}$ for formulas, possibly with primes and subscripts. In addition, we use σ, τ for substitutions, $\mathcal{F}_{\mathcal{G}}(\mathcal{H})$ for replacement of *all* occurence of a subformula $\mathcal{G}$ in a formula $\mathcal{F}$ with $\mathcal{H}$ and $\mathcal{F}_{\mathcal{G}}[\mathcal{H}]$ for replacement of an occurence of a subformula $\mathcal{G}$ in a formula $\mathcal{F}$ with $\mathcal{H}$.

2.1. Proof Tree of Ground Goals

A *definite clause program* is a finite set of definite clauses. Variables appearing in the body and not appearing in the head of a definite clause are called *internal variables* of the definite clause. Atoms containing no variable are called *ground atoms*. Finite multisets of (ground) atoms are called *(ground) atom sets*.

Let S^{old} be a definite clause program. (The meaning of the suffix "old" is explained later.) A *proof tree*, or simply *proof*, of ground atom A in S^{old} is a tree T lebelled with ground atoms defined as follows.

(a) T is a proof of A in S^{old} 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^{old}. (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^{old} 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^{old} and $T_1, T_2, \ldots, T_m$ are proofs of $A_1, A_2, \ldots, A_m$ in S^{old} 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.)

Example 2.1. Let *common* and *member* be predicates defined by

```
common(X,L,M) :- member(X,L),member(X,M).
member(U,[U|L]).
member(U,[V|L]) :- member(U,L).
```

Then the tree below is a proof of $common(3,[2,3],[3,5])$ containing 4 nodes.

```
                common(3,[2,3],[3,5])
                   /         \
      member(3,[2,3])    member(3,[3,5])
            |
       member(3,[3])
```

The set of all ground atoms for which proof trees exist is denoted by $M(S^{old})$. It is exactly the minimum Herbrand model of S^{old}. Let T be any proof tree of A in S^{old} which contains the minimum number of nodes among the proofs of A in S^{old}. The definite clause C used at the root of T is going to play a very important role in 4.2.

2.2. Terminating Atom

Let S^{old} be a definite clause program and As be a atom set. Then a *search tree* of As in S^{old} is a tree defined as follows [15].

(a) Each node of the tree is a atom set (possibly empty).

(b) The root node is As.

(c) Let $\{A_1, A_2, \ldots, A_n\}$ be a node in the tree and suppose that A_i is an atom, called a *selected* atom, in it. Then, this node has descendants for each clause "B_0 :- $B_1, B_2, \ldots, B_m$"

in S^{old} such that A_i and B_0 are unifiable, say by an m.g.u. σ. The descendant is $\sigma(\{A_1, \ldots, A_{i-1}, B_1, B_2, \ldots, B_m, A_{i+1}, \ldots, A_n\})$.

(d) Nodes which are the empty atom set have no descendant.

The empty atom sets have no descendant, as is defined in (d) above, and are called *success nodes*. Some non-empty atom sets may have no descendant, for which the selected atom has no clause with a unifiable head in S^{old}, and are called *failure nodes*.

Example 2.2.1. Let *As* be a singleton set { common(2,[2,3],[3,5]) }. Then the tree below is a finite search tree, in which all branches end in failure nodes. The underlines indicate selected atoms.

```
                 { common(2,[2,3],[3,5]) }
                             |
           { member(2,[2,3]),member(2,[3,5]) }
                        /          \
{ member(2,[3,5]) }        { member(2,[3]),member(2,[3,5]) }
          |                              |
 { member(2,[5]) }         { member(2,[ ]),member(2,[3,5]) }
          |
 { member(2,[ ]) }
```

An atom A is said to be *terminating* when there is no search tree of $\{A\}$ containing an infinite branch from the root. In defining the semantics of pure Prolog, we employ a nondeterministic proof procedure in order to avoid the incompleteness due to the specific behavior, i.e. depth-first search with backtracking, of the actual interpreter. When atom A is terminating, such care is unnecessary. The actual interpreter either stops with success or fails finitely for A.

Example 2.2.2. Let *true-or-loop* be a predicate defined by

```
true-or-loop(X) :- is-true.
true-or-loop(X) :- loop(X).
is-true.
loop(X) :- loop(X).
```

Though *true-or-loop*(X) is tautologically true, it is not terminating and there is an infinite branch from the root $\{true\text{-}or\text{-}loop(t)\} \rightarrow \{loop(t)\} \rightarrow \{loop(t)\} \rightarrow \{loop(t)\} \rightarrow \cdots$.

Though there are known several sufficient conditions for guaranteeing that an atom A is terminating, we do not refer the details in order not to make the explanation of the construction rules in Section 3 too complicated.

2.3. Goals

In this section, we generalize usual atoms to *goals*. Now on, we assume about constant, function and predicate symbols as follows.

(a) The set of constant and function symbols is fixed so that we have a fixed Herbrand universe.

(b) The set of predicate symbols is divided into two disjoint sets. One is a set of predicates called *old predicates*. Another is a set of predicates called *new predicates*.

The old predicates are defined by a fixed definite clause program S^{old}. The new predicates are defined by a *definite formula program* S^{new} being introduced in 2.4. Atoms with the old predicates are called *old atoms*, while those with the new predicates are called *new atoms*.

First, we introduce *polarity* of subformulas. The *positive* and *negative subformulas* of a formula $\mathcal{F}$ are defined as follows (see Prawitz [18],Murray [16],Manna and Waldinger [15]).

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

Example 2.3.1. Let $\mathcal{F}$ be $\forall Y(member(Y,L)\supset X < Y)$. Then $member(Y,L)$ is a negative subformula of $\mathcal{F}$.

Let $\mathcal{F}$ be a first order formula. Variables not quantified in $\mathcal{F}$ are called *global variables*. When $\forall X\,\mathcal{G}$ is a positive subformula or $\exists X\,\mathcal{G}$ is a negative subformula of $\mathcal{F}$, X is called *free variable* of $\mathcal{F}$. In other words, free variables are variables quantified universally when $\mathcal{F}$ is converted to prenex normal form.

Example 2.3.2. Let $\mathcal{F}$ be $\forall Y(member(Y,L)\supset X < Y)$. Then X and L are global variables, while Y is a free variable.

Goals, denoted by F,G,H now on, are defined as follows.

(a) A new atom is a goal. Variables in such an atom are *global variables*.
(b) Let $\mathcal{F}$ be a formula which consists of only old atoms and contains no variable other than global variables and free variables. A formula G obtained from such a formula $\mathcal{F}$ by leaving global variable X as it is, replacing free variable Y with $!Y$ and deleting all quantifiers is a *goal*. (Note that $\mathcal{F}$ can be uniquely restorable from G.)

Goals containing no global variable are called *ground goals*. Note that goals in the case (b) consist of only old atoms. Hence if the minimum Herbrand model $M(S^{old})$ is fixed, the set of all ground goals true in $M(S^{old})$, denoted by $\overline{M}(S^{old})$, is also fixed, because we assume a fixed Herbrand universe over which free variables range. Multisets of (ground) goals are called *(ground) goal sets*.

Example 2.3.3. Let *less-than-all* be a new predicate and *list*,*member* and $<$ be old predicates. Then $less\text{-}than\text{-}all(X,L)$ is a goal, where X,L are global variables. $list(L)$ is not only an atom with an old predicate but also a goal, where L is a global variable. In general, usual (ground) atoms are (ground) goals without free variables. $member(!Y,L)\supset X <!Y$ is a goal representing $\forall Y(member(Y,L)\supset X < Y)$. $member(!Y,[5,3])\supset 2 <!Y$ is a ground goal.

2.4. Definite Formulas

In this section, we generalize definite clauses to *definite formulas* and define the new predicates assumed in the previous section by a set of definite formulas S^{new}.

A formula is called *definite formula* when it is of the form ($m\geq 0$)

A :- $G_1, G_2, \ldots, G_m$

where $G_1, G_2, \ldots, G_m$ are goals without common free variables. Definite formulas represent formulas convertible to prenex normal forms

$$\forall X_1,X_2,\ldots,X_\alpha\ \exists Y_1,Y_2,\ldots,Y_\beta\ (\mathcal{G}_1\wedge\mathcal{G}_2\wedge\cdots\wedge\mathcal{G}_m\supset A)$$

where $X_1, X_2, \ldots, X_\alpha$ are all global variables, $Y_1, Y_2, \ldots, Y_\beta$ are all free variables and $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m$ are quantifier-free formulas. A finite set of definite formulas is called *definite formula program*. Variables appearing in the body and not appearing in the head of a definite formula are called *internal variables* of the definite formula.

Example 2.4.1. A formula

less-than-all(X,[Y|L]) :- list(L),X<Y,(member(!Y,L)⊃X<!Y).

is a definite formula representing

$\forall$ X,Y,L (list(L)$\wedge$X<Y$\wedge$($\forall$Y'(member(Y',L) $\supset$X<Y')) $\supset$less-than-all(X,L))

Note that definite formulas include definite clauses as well as general forms of definie clause programs [4]

$$\forall X_1,X_2,\ldots,X_n\ (E_1\vee E_2\vee\cdots\vee E_k\supset p(X_1,X_2,\ldots,X_n))$$

where each E_i is of the form

$\exists\ Y_1,Y_2,\ldots,Y_l\ (X_1{=}t_1\wedge X_2{=}t_2\wedge\cdots\wedge X_n{=}t_n\wedge B_1\wedge B_2\wedge\cdots\wedge B_m$

and $Y_1, Y_2, \ldots, Y_l$ are all variables in $t_1, t_2, \ldots, t_n, B_1, B_2, \ldots, B_m$.

Example 2.4.2. The following definite clauses

less-than-all(X,[ ]).

less-than-all(X,[Y|L]) :- X<Y,less-than-all(X,L).

are definite formulas representing

$\forall$ X less-than-all(X,[ ]).

$\forall$ X,Y,L (X<Y$\wedge$less-than-all(X,L)$\supset$less-than-all(X,[Y|L])).

Example 2.4.3. The general form of the definite clause program of *member*

$\forall$ X,L($\exists X_1,L_1$ (X=$X_1\wedge$L=[X_1|L_1])$\vee$

$\exists X_2,Y_2,L_2$ (X=$X_2\wedge$L=[Y_2|L_2]$\wedge$member(X_2,L_2)) $\supset$member(X,L))

is represented by a definite formula

member(X,L) :- (X=$X_1\wedge$L=[X_1|L_1])$\vee$(X=$X_2\wedge$L=[Y_2|L_2]$\wedge$member(X_2,L_2)).

2.5. Manipulation of Goals

In this section, we introduce three notions about goals, which are used intensively in the sequel.

The first one means intuitively that some subformulas must be true and some must be false when the whole formula is true. The *must-be-true* and *must-be-false subformulas* of a goal F are defined as follows (cf. *positive* and *negative part* by Shütte [22]).

(a) F is a must-be-true subformula of F.

(b) When $\neg G$ is a must-be-true (must-be-false) subformula of F, then G is a must-be-false (must-be-true) subformula of F.

(c) When $G\wedge H$ is a must-be-true subformula of F, then G and H are must-be-true subformulas of F.

(d) When $G\vee H$ is a must-be-false subformula of F, then G and H are must-be-false subformulas of F.

(e) When $G\supset H$ is a must-be-false subformula of F, then G is a must-be-true subformula of F and H is a must-be-false subformula of F.

Those subformulas are related with the polarity, i.e., must-be-true subformulas are always positive and must-be-false subformulas are always negative.

Example 2.5.1. $list(L)$ is a must-be-true subformula of itself. In general, usual atomic goals are always must-be-true subformulas of themselves. $member(!Y, L)$ is neither a must-be-true nor a must-be-false subformula of $member(!Y, L) \supset X <!Y$.

The second one is applications of classes of substitutions. A substitution σ for a goal G is called a *positive substitution* when σ instantiates no free variable in G and $\sigma(X)$ contains no free variable for any global variable X. A substitution σ for G is called a *negative substitution* when σ instantiates no global variable in G.

Example 2.5.2. Let G be the second goal in the body of the definite formula.

less-than-all(X,[Y|L]) :- list(L),(member(!Y,[Y|L])⊃X<!Y).

One of the most general unifier of $member(!Y, [Y|L])$ and the head of the first definite clause defining *member* is a negative substitution $\sigma = < !Y \Leftarrow Y >$. $\sigma(G)$ represents a goal $member(Y, [Y|L]) \supset X < Y$.

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

$$\begin{array}{ll}
\neg true \rightarrow false, & \neg false \rightarrow true, \\
true \wedge G \rightarrow G, & false \wedge G \rightarrow false, \\
G \wedge true \rightarrow G, & G \wedge false \rightarrow false, \\
true \vee G \rightarrow true, & false \vee G \rightarrow G, \\
G \vee true \rightarrow true, & G \vee false \rightarrow G, \\
true \supset G \rightarrow G, & false \supset G \rightarrow true, \\
G \supset true \rightarrow true, & G \supset false \rightarrow \neg G.
\end{array}$$

Example 2.5.3. Let F be $false \supset X <!Y$. Then $F\downarrow$ is *true*. Let G be $true \supset X < Y$. Then $G\downarrow$ is $X < Y$.

3. Unfold/Fold Construction of Logic Programs

3.1. Construction Process

The entire process of our construction proceeds in the completely same way as Tamaki-Sato's transformation [24] as follows.

P_0 := the initial definite clause program ; D_0 := {};
mark every clause in P_0 "foldable";
for i := 1 **to** arbitrary N
  apply any of the construction rules to obtain P_i and D_i from P_{i-1} and D_{i-1};

Figure 1. Construction Process

Example 3.1. Before starting, the initial definite clause program is given, e.g.,

P_0 : C_1. list([ ]).

C_2. list([X|L]) :- list(L).
C_3. 0<suc(Y).
C_4. suc(X)<suc(Y) :- X<Y.
C_5. member(U,[U|L]).
C_6. member(U,[V|L]) :- member(U,L).

and D_0 is initialized to {}. This example is used to illustrate the rules of construction.

3.2. Basic Construction Rules

The basic part of our construction system consists of four rules, i.e., definition, positive unfolding, negative unfolding and folding. In the following, we implicitly assume that a goal is always deleted from the body of definite formulas when it is the logical constant *true* and a definite formula is always deleted from the set of definite formulas when some goal in the body is the logical constant *false*.

Definition : Let C be a definite formula of the form $p(X_1, X_2, \ldots, X_n)$:- $G_1, G_2, \ldots, G_m$ 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 global variables and
(c) predicates of atoms in $G_1, G_2, \ldots, G_m$ all appears in P_0.

Then let P_i be $P_{i-1} \bigcup \{C\}$ and D_i be $D_{i-1} \bigcup \{C\}$. Do not mark C "foldable".

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

Example 3.2.1. Suppose we need a relation meaning that some X is less than any element of a list L. Then we introduce it by the following definition.

C_7. less-than-all(X,L) :- list(L),(member(!Y,L)⊃X<!Y).

Then $P_1 = \{\underline{C_1}, \underline{C_2}, \underline{C_3}, \underline{C_4}, \underline{C_5}, \underline{C_6}, C_7\}$ and $D_1 = \{C_7\}$. The underlines indicate "foldable" clauses.

Positive Unfolding : Let C be a definite formula in P_{i-1} defining a new predicate and A be a positive atom with an old predicate p of some goal G in the body, where

(P_1) it is terminating when all global variables in A are instantiated to ground terms or
(P_2) A is a must-be-true atom of G.

Then

(a) If there is no definite clause with unifiable head, then let C'_1 be the definite formula obtained from C by replacing G with $G_A[false]\downarrow$.
(b) If, for all the definite clauses in P_{i-1} whose heads are unifiable with A, say $C_1, C_2, \ldots, C_k$, they are unifiable with A by positive m.g.u.'s $\sigma_1, \sigma_2, \ldots, \sigma_k$ and the bodies of $\sigma_1(C_1), \sigma_2(C_2), \ldots, \sigma_k(C_k)$ contain no free variable, let G_i be the reduced form of $\sigma_i(G)$ after replacing $\sigma_i(A)$ in $\sigma_i(G)$ with the body of $\sigma_i(C_i)$ and C'_i be the definite formula obtained from $\sigma_i(C)$ by replacing $\sigma_i(G)$ with G_i. (When the body is empty, replace with *true*. New variables introduced from C_i are global variables in G_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}. Mark each C'_i "foldable" unless it is already in P_{i-1}.

Regrettably, the conditions for $\sigma_1, \sigma_2, \ldots, \sigma_k$ are slightly messy. Intuitively, these conditions are for guaranteeing that the resulting formulas fall in the class of definite formulas.

Example 3.2.2. When C_7 is unfolded at its first atom $list(L)$ in the body, we obtain $P_2=\{\underline{C_1}, \underline{C_2}, \underline{C_3}, \underline{C_5}, \underline{C_6}, C_8, C_9\}$ and $D_2=\{C_7\}$ where

C_8. less-than-all(X,[ ]) :- (member(!Y,[ ])$\supset$X<!Y).

C_9. less-than-all(X,[Y|L]) :- list(L), (member(!Y,[Y|L])$\supset$X<!Y).

Negative Unfolding : Let C be a definite formula in P_{i-1} defining a new predicate and A be a negative atom with an old predicate p of some goal G in the body, where

(N) it is terminating when all global variables in A are instantiated to ground terms.

If, for all the definite clauses in P_{i-1} whose heads are unifiable with A, say $C_1, C_2, \ldots, C_k$, they are unifiable by negative m.g.u.'s $\sigma_1, \sigma_2, \ldots, \sigma_k$, let G_0 be the reduced form after replacing A in G with *false* and let G_i be the reduced form after replacing $\sigma_i(A)$ in $\sigma_i(G)$ with the body of $\sigma(C_i)$. (When the body is empty,replace with *true*. New variables introduced from C_i are free variables in G_i.) Then let C' be the definite formula obtained from C by replacing G in the body of C with $G_0, G_1, G_2, \ldots, G_k$. Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'\}$ and D_i be D_{i-1}. Mark C' "foldable" unless it is already in P_{i-1}.

Example 3.2.3. When C_8 is unfolded at its atom $member(!Y, [\,])$ in the body, we obtain $P_3=\{\underline{C_1}, \underline{C_2}, \underline{C_3}, \underline{C_5}, \underline{C_6}, \underline{C'_8}, \underline{C_9}\}$ and $D_3=\{C_7\}$ where

C'_8. less-than-all(X,[ ]) :- (false$\supset$X<!Y)$\downarrow$.

that is,

C'_8. less-than-all(X,[ ]).

because $member(!Y, [])$ is unifiable with no clause defining $member$. Similarly, when C_9 is unfolded at its atom $member(!Y, [Y|L])$ in the body, we obtain $P_4=\{\underline{C_1}, \underline{C_2}, \underline{C_3}, \underline{C_5}, \underline{C_6}, \underline{C'_8}, \underline{C'_9}\}$ and $D_4=\{C_7\}$ where

C'_9. less-than-all(X,[Y|L]) :-
list(L), (false$\supset$X<Y)$\downarrow$, (true$\supset$X<Y)$\downarrow$, (member(!Y,L)$\supset$X<!Y)$\downarrow$.

that is,

C'_9. less-than-all(X,[Y|L]) :- list(L),X<Y, (member(!Y,L)$\supset$X<!Y).

Folding : Let C be a definite formula in P_{i-1} of the form "A :- $F_1, F_2, \ldots, F_n$" defining a new predicate and C_{folder} be a definite formula in D_{i-1} of the form "B :- $G_1, G_2, \ldots, G_m$". Suppose there is a substitution σ and a subset $\{F_{i_1}, F_{i_2}, \ldots, F_{i_m}\}$ of the body of C such that the following conditions hold.

(a) $F_{ij} = \sigma(G_j)$ 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 $\{F_1, F_2, \ldots, F_n\} - \{F_{i_1}, F_{i_2}, \ldots, F_{i_m}\}$ and

(c) C is marked "foldable" or $m < n$.

Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'\}$ and D_i be D_{i-1} where C' is a definite formula with head A and body $(\{F_1, F_2, \ldots, F_n\} - \{F_{i_1}, F_{i_2}, \ldots, F_{i_m}\}) \bigcup \{\sigma(B)\}$. Let C' inherit the mark of C.

Example 3.2.4. By folding the body of C'_9 except $X < Y$ by C_7, we obtain $P_5=\{\underline{C_1}, \underline{C_2}, \underline{C_3}, \underline{C_5}, \underline{C_6}, \underline{C'_8}, \underline{C''_9}\}$ and $D_5=\{C_7\}$ where

C''_9. less-than-all(X,[Y|L]) :- X<Y,less-than-all(X,L).

3.3. Equivalence Preservation Theorem

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

Example 3.3. If we stop the construction process at step 5, we reach the final program and the definition set

P_5 : C_1. list([ ]).
C_2. list([X|L]) :- list(L).
C_3. 0<suc(Y).
C_4. suc(X)<suc(Y) :- X<Y.
C_5. member(U,[U|L]).
C_6. member(U,[V|L]) :- member(U,L).
C'_8. less-than-all(X,[ ]).
C''_9. less-than-all(X,[Y|L]) :- X<Y,less-than-all(X,L).
D : C_7. less-than-all(X,L) :- list(L),(member(!Y,L)⊃X<!Y).

The most important property being proved in Section 4 is the following theorem.

Theorem 3.3. The minimum Herbrand model of the initial program plus the definition set $P_0 \bigcup D$ is identical to that of the final program P_N.

But in the following discussion, it is convinient to assume that all definitions in D are given from the beginning. To pretend it, for any construction sequence (P_0, D_0),(P_1, D_1),..., (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 a *virtual construction sequence*. (This is also due to Tamaki and Sato [24].) In particular $S_0 = P_0 \bigcup D$ and $S_N = P_N$. Since the definition rule is the identity in the virtual construction sequence, it is ignored when treating the virtual construction sequence. Moreover, for simplicity, we have restricted the unfoldings to those on old atoms in definite formulas defining new predicates. Hence definite clauses defining old predicates in S_i are kept fixed during the construction process and the definite formulas defining new predicates is the only changing part. We denote the former by S^{old} and the latter by S_i^{new}.

4. Preservation of Minimum Herbrand Models

4.1. Semantics of Definite Formula Programs

In this section, we show how to give semantics to definite formula program $S^{old} \bigcup S^{new}$, model theoretically and proof theoretically.

Suppose we have a fixed set of constant symbols and function symbols, hence a fixed Herbrand universe H. For a given set of old predicate symbols and a definite clause program S^{old} defining them, we have a minimum Herbrand model $M(S^{old})$, hence a corresponding set of ground goals $\overline{M}(S^{old})$ true in $M(S^{old})$. Now suppose we have a definite formula program S^{new} defining the new predicates. We can consider various Herbrand interpretations I such that I is identical to $M(S^{old})$ as to the old predicates and interpretes the new predicates somehow. Some of them are models of $S^{old} \bigcup S^{new}$, but these models are not necessarily minimum in the general sense.

Example 4.1.1. Let *even* be a predicate defined by
even(0).
even(suc(suc(X))) :- even(X).
even(X) :- even(suc(suc(X))).

and our Herbrand universe H be $\{0, suc(0), suc(suc(0)), \ldots\}$. Because of the third additional definite clause, there exist two Herbrand models

$M_0 = \{$ even(0),even(suc(suc(0))),...,even(suc^{2i}(0)),...$\}$,

$M_1 = \{$ even(0),even(suc(0)),...,even(suc^{i}(0)),...$\}$.

Suppose we have defined a new predicate *conditional-double* by

conditional-double(Y) :- even(X)$\supset$add(X,X,Y).

The Herbrand model corresponding to M_0 is

$M_0 \bigcup \{$conditional-double(suc^{i}(0))$\mid i \in N\}$,

which is not included in the Herbrand model corresponding to M_1

$M_1 \bigcup \{$conditional-double(suc^{2i}(0))$\mid i \in N\}$.

Because of the restriction we imposed on goals, we can still enjoy a kind of *model intersection property*. We call Herbrand models whose interpretations are identical to $M(S^{old})$ as to the old predicates *Herbrand models on* $M(S^{old})$.

Lemma 4.1.1. Let M_1 and M_2 be two Herbrand models of $S^{old} \bigcup S^{new}$ on $M(S^{old})$. Then $M_1 \bigcap M_2$ is also an Herbrand model of $S^{old} \bigcup S^{new}$ on $M(S^{old})$.

Proof. We prove the lemma for a more general case such that goals may include positive new atoms. Consider any ground instantiation σ of all free variables in a ground definite formula $p(t_1, t_2, \ldots, t_n)$:- $G_1, G_2, \ldots, G_m$. Suppose that the interpretation of $\sigma(G_i)$ in $M_1 \bigcap M_2$ is *true*. Because the interpretation by M_1,M_2 and $M_1 \bigcap M_2$ are identical on atoms in $\sigma(G_i)$ except that M_1 or M_2 may includes more (possibly zero) positive new atoms in $\sigma(G_i)$ which is not in $M_1 \bigcap M_2$. Consider all atoms in $\sigma(G_i)$ that has the common interpretation and let F be a formula obtained by assigning *true* or *false* to the atoms according to it. Because the atoms with the different interpretation are all positive in $\sigma(G_i)$ and positive atoms in F are also positive in $F\downarrow$ if they appear in $F\downarrow$, the result of reduction $F\downarrow$ is exactly the logical constant *true*, hence $\sigma(G_i)$ are also true in M_1 and M_2. Then, since this holds for all i and M_1 and M_2 are both Herbrand models of $S^{old} \bigcup S^{new}$, $p(t_1, t_2, \ldots, t_n)$ is included in both M_1 and M_2, hence in $M_1 \bigcap M_2$, when $G_1, G_2, \ldots, G_m$ are true in $M_1 \bigcap M_2$. Because this holds any ground instance of definite formulas, $M_1 \bigcap M_2$ is an Herbrand model of $S^{old} \bigcup S^{new}$.

Collorary 4.1. $S^{old} \bigcup S^{new}$ has a minimum Herbrand model in the class of the Herbrand models on $M(S^{old})$.

Proof. Because $M(S^{old}) \bigcup \{p(t_1, t_2, \ldots, t_n) \mid p$ is a new predicate and $t_1, t_2, \ldots, t_n \in H\}$ is an Herbrand model of $S^{old} \bigcup S^{new}$, there exists at least one Herbrand model of $S^{old} \bigcup S^{new}$ in the class. Then the intersection of all these Herbrand models $\bigcap M$ is the minimum Herbrand model we want.

We can still enjoy a kind of *continuity* as well. Let us define the transformation T of Herbrand interpretations on $M(S^{old})$ as follows.

$$T(I) = M(S^{old}) \bigcup \{p(t_1, t_2, \ldots, t_n) \mid p \text{ is a new predicate, } p(t_1, t_2, \ldots, t_n) \text{ :- } G_1, G_2, \ldots, G_m \text{ is a ground instance of a definite formula in } S^{new} \text{ and all } G_1, G_2, \ldots, G_m \text{ are either in } \overline{M}(S^{old}) \text{ or in } I\}.$$

Lemma 4.1.2. $\bigcup_{i=0}^{\infty} T^i(M(S^{old}))$ is the minimum Herbrand model in the class of the Herbrand models on $M(S^{old})$.

Proof. It is proved similarly to the proof for usual definite clause programs. See [1] or [7].

Let S^{old} and S^{new} be as before. A *proof tree*, or simply *proof*, of a ground goal G in $S^{old} \bigcup S^{new}$ is a tree T lebelled with ground goals defined as follows.

(a) T is a proof of G in $S^{old} \bigcup S^{new}$ when it is a tree consisting of a single node labelled with a ground goal G in $\overline{M}(S^{old})$.

(b) Let $T_1, T_2, \ldots, T_m$ be immediate subtrees of T and $G_1, G_2, \ldots, G_m$ be their root labels. T is a proof of G in $S^{old} \bigcup S^{new}$ when G is a ground new atom A, the root label of T is A, "A :- $G_1, G_2, \ldots, G_m$" is a ground instance of some definite formula in S^{new} and $T_1, T_2, \ldots, T_m$ are proofs of $G_1, G_2, \ldots, G_m$ in $S^{old} \bigcup S^{new}$ respectively. (The definite formula is said to be *used at the root* and $T_1, T_2, \ldots, T_m$ are called *immediate subproofs* of T.)

Example 4.1.2. When *less-than-all* is defined by

```
less-than-all(X,L) :- list(L),member(!Y,L)⊃X<!Y.
```

the tree below is a proof of *less-than-all*(2, [5, 3]).

```
            less-than-all(2,[5,3])
                 /          \
       list([5,3])       member(!Y,[5,3])⊃2<!Y
```

Example 4.1.3. When *less-than-all* is defined by

```
less-than-all(X,[ ]).
less-than-all(X,[Y|L]) :- list(L),X<Y,member(!Y,L)⊃X<!Y.
```

the tree below is a proof of *less-than-all*(2, [5, 3]).

```
            less-than-all(2,[5,3])
                 /     |      \
        list([3])    2<5    member(!Y,[3])⊃2<!Y
```

Example 4.1.4. When *less-than-all* is defined by

```
less-than-all(X,[ ]).
less-than-all(X,[Y|L]) :- X<Y,less-than-all(X,L).
```

the tree below is a proof of *less-than-all*(2, [5, 3]).

```
            less-than-all(2,[5,3])
                   /     \
                2<5   less-than-all(2,[3])
                              /  \
                           2<3   less-than-all(2,[ ])
```

Lemma 4.1.3. The set of all ground atoms that have proofs in $S^{old} \bigcup S^{new}$ is identical to the minimum Herbrand model on $M(S^{old})$.

Proof. Trivial from the continuity of T shown by Lemma 4.1.2.

Before introducing a well-founded ordering, we notice about validity of goals in unfolding.

Lemma 4.1.4.

(a) Let $G_1, G_2, \ldots, G_k$ be goals obtained from a ground goal G by positive unfolding. Then G is in $\overline{M}(S^{old})$ if and only if a ground instance of some G_i $(1 \leq i \leq k)$ is in $\overline{M}(S^{old})$.

(b) Let $G_0, G_1, G_2, \ldots, G_k$ be ground goals obtained from a ground goal G by negative unfolding. Then G is in $\overline{M}(S^{old})$ if and only if all G_i $(0 \leq i \leq k)$ are in $\overline{M}(S^{old})$.

Outline of Proof. Note that $M(S^{old})$ is a model of the completion of S^{old} and that replacement of equivalence with equivalence using the completion of S^{old} keeps validity. Suppose an unfolding is done at a ground old atom A.

As for (a), it is easy to show that G is in $\overline{M}(S^{old})$ if and only if $G_1 \vee G_2 \vee \cdots G_k$ is in $\overline{M}(S^{old})$.

As for (b), G is in $\overline{M}(S^{old})$ if and only if $A \supset G$ and $\neg A \supset G$ are in $\overline{M}(S^{old})$. The former goal is in $\overline{M}(S^{old})$ if and only if $G_1, G_2, \ldots, G_k$ are in $\overline{M}(S^{old})$. The latter goal is in $\overline{M}(S^{old})$ if and only if G_0 is in $\overline{M}(S^{old})$.

4.2. A Well-Founded Ordering on Ground Goal Sets

In this section, we define a slightly complicated ordering $\prec_M$ on ground goal sets true in $M(S^{old})$, i.e., the multiset on $\overline{M}(S^{old})$, which plays a basic role to introduce two important notions, *rank* and *rank ordering*, in the next section.

$\prec$ on $\overline{M}(S^{old})$ is the minimum transitive relation satisfying the following conditions.

(P_1) When G' is a ground instance of a goal obtained from a ground goal G by positive unfolding at a terminating atom A then $G' \prec G$.

(P_2) When G' is a ground instance of a goal obtained from a ground goal G by positive unfolding at a must-be-true atom A using the definite clause used at the root of the minimum proof of A, then $G' \prec G$.

(N) When G' is a ground goal obtained from a ground goal G by negative unfolding at a terminating atom A then $G' \prec G$.

Example 4.2.1. Let S^{old} be P_0 in Example 3.1 defining *list*, *member* and $<$. Then
2<5 $\prec$ member(!Y,[5,3])$\supset$2<!Y,
member(!Y,[3])$\supset$2<!Y $\prec$ member(!Y,[5,3])$\supset$2<!Y,
list([3]) $\prec$ list([5,3]).

Lemma 4.2.1. $\prec$ is a well-founded ordering.
Proof. It is enough to prove that there is no infinite decreasing sequence $F_0 \succ F_1 \succ F_2 \succ \cdots \succ F_n \succ \cdots$. Let us call $\sigma(B_1 \wedge B_2 \wedge \cdots \wedge B_m)$ *descendant of A* when A in F_i is replaced with $\sigma(B_1 \wedge B_2 \wedge \cdots \wedge B_m)$ in F_{i+1}. Note that free variables in such an infinite sequence are instantiated only by negative substitutions in negative unfoldings. Hence,any sequence of descendants of a negative ground atom A in F_0 is a branch of a search tree of $\{A\}$ or part of it. When A is terminating, such a sequence is finite. Hence there occurs only finite number of negative unfoldings in the sequence. Let the result of the last negative unfolding be G_0. Now,it is enough to prove that there is no infinite decreasing sequence $G_0 \succ G_1 \succ G_2 \succ \cdots \succ G_n \succ \cdots$ in which all the $\succ$ relations hold by positive unfoldings in the definition above. Again, any sequence of descendants of a positive ground atom A in G_0 is a branch of a search tree of $\{A\}$ or part of it. (Free variables in A act as if they were new constants.) Because of the conditions of positive unfoldings, we can say again that such a sequence is finite.

$\prec_M$ on the multiset of $\overline{M}(S^{old})$ is the multiset ordering over $\prec$,i.e., the minimum transitive relation satisfying that $Gs' \prec_M Gs$ when a ground goal set Gs' is obtained by replacing some ground goal G in a ground goal set Gs with (possibly zero) ground goals less than G by the ordering $\prec$.

Example 4.2.2. Let S^{old} be as before. Then
$\{list([3]), 2 < 5, member(!Y, [3]) \supset 2 < !Y\}$

$\prec_M \{list([3]), member(!Y, [5,3]) \supset 2 < !Y\}$
$\prec_M \{list([5,3]), member(!Y, [5,3]) \supset 2 < !Y\}$.

Lemma 4.2.2. $\prec_M$ is a well-founded ordering.

Proof. In general,a multiset ordering over a well-founded ordering is always a well-founded ordering. See Dershowitz and Manna [6] p.467.

4.3. Rank and Rank Ordering of Goals

The *rank* is a mapping *rank* from the set of all ground goals true in $M(S^{old} \bigcup S_0^{new})$ to the set of all ground goal sets true in $M(S^{old})$, i.e., $rank : \overline{M}(S^{old} \bigcup S_0^{new}) \mapsto 2^{\overline{M}(S^{old})}$, defind as follows.

(a) $rank(A) = \{G_1, G_2, \ldots, G_m\}$ when A is a ground new atom, where "A :- $G_1, G_2, \ldots, G_m$" is a ground instance of the definite formula defining the new predicate in P_0^{new} used at the root of the minimum proof of A in $S^{old} \bigcup S_0^{new}$.
(b) $rank(G) = \{G\}$ when G is a ground goal consisting of old atoms.

Example 4.3.1. The rank of $2 < 5$ is $\{2 < 5\}$. The rank of *less-than-all*$(2, [5,3])$ is $\{list([5,3]), member(!Y, [5,3]) \supset 2 < !Y\}$.

The *rank ordering* is a well-founded ordering $\ll$ on the set of ground goals $M(S^{old} \bigcup S_0^{new})$. Let A and B be two ground goals. $A \ll B$ is defined as follows.

(a) $A \ll B$ when $rank(A) \prec_M rank(B)$.
(b) $A \ll B$ when $rank(A) = rank(B)$ and the predicate of A is old and that of B is new.

Example 4.3.2. Let S^{old} and S^{new} be as before.Then
less-than-all(2,[3]) $\ll$ less-than-all(2,[5,3]),
because
rank(less-than-all(2,[3]))={ list([3]),member(!Y,[3])$\supset$2<!Y }
$\prec_M$ { list([3]),2<5,member(!Y,[3])$\supset$2<!Y }
$\prec_M$ { list([5,3]),2<5,member(!Y,[3])$\supset$2<!Y }
$\prec_M$ { list([5,3]),member(!Y,[5,3])$\supset$2<!Y }
= rank(less-than-all(2,[5,3])).

4.4. Rank-Consistent Proof

Let $S_i = S^{old} \bigcup S_i^{new}$ be a definite formula program. A proof T of a ground goal G 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 G in S_i when it is a tree consisting of a single node labelled with a ground goal G in $\overline{M}(S^{old})$.
(b) Let $T_1, T_2, \ldots, T_m$ be immediate subproofs of T, $G_1, G_2, \ldots, G_m$ be their root labels and C be the definite formula used at the root of T. T is a rank-consistent proof of G in S_i when (i) G is a ground new atom A, (ii) $rank(A) \succeq_M rank(G_1) \bigcup rank(G_2) \bigcup \cdots \bigcup rank(G_m)$ with equality holding only when C is not marked "foldable" (iii) $G \gg G_k$ for all k $(1 \leq k \leq m)$ and (iv) $T_1, T_2, \ldots, T_m$ are rank-consistent proofs of $G_1, G_2, \ldots, G_m$ respectively.

Example 4.4.1. Let S_5^{new} be a definite formula program
less-than-all(X,[ ]).
less-than-all(X,[Y|L]) :- X<Y,less-than-all(X,L).

Then the proof of *less-than-all*(2, [5, 3]) below

```
                    less-than-all(2,[5,3])
                         /     \
                      2<5    less-than-all(2,[3])
                                  /     \
                               2<3    less-than-all(2,[ ])
```

is rank-consistent, because

rank(less-than-all(2,[5,3])) = { list([5,3]),member(!Y,[5,3])⊃2<!Y }
$\succ_M$ { list([3]),2<5,member(!Y,[3])⊃2<!Y }
= rank(2<5) ⋃ rank(less-than-all(2,[3])),

rank(less-than-all(2,[3])) = { list([3]),member(!Y,[3])⊃2<!Y }
$\succ_M$ { list([ ]),2<3,member(!Y,[ ])⊃2<!Y }
= rank(2<3) ⋃ rank(less-than-all(2,[ ])).

4.5. 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 [24] intentionally in order to emphasize the role of our ordering. We prove the following theorem.

Theorem 4.5. Let $S_1, S_2, \ldots, S_N$ be the virtual transformation sequence. Then $M(S_N) = M(S_0)$.

As was noted, we assumed for simplicity that S^{old} is fixed. Hence we only need to prove the theorem as for new predicates. 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.

Base Case :

The first invariant I1 trivially holds for $i = 0$. As for the second invariant I2, for any ground new atom A in $M(S_0)$, the proof of A is only one using the definition of the new predicate in D, which is obviously rank-consistent. ($S_0 = P_0 \bigcup D$ and the clauses in P_0 are marked "foldable" while those in D are not.)

Induction Step :

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

Lemma 4.5.1. If the invariant I1 holds for S_i, then $M(S_{i+1}) \subseteq M(S_i)$.

Proof. Let A be a ground new 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 formula 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+1} with each T'_j corresponding to T_j. If C is in S_{i+1}, we can immediately construct T' from C and the proofs $T'_1, T'_2, \ldots, T'_n$.

Suppose C is the result of positive unfolding. Then for some j $(1 \leq j \leq n)$, say 1, the root label G_1 of T_1 is a ground instance of a goal obtained from G' by the positive unfolding. Because G' is true in $M(S^{old})$ if G_1 is true in $M(S^{old})$ and G' itself is a proof T'_1, we can construct T' from $T'_1, T'_2, \ldots, T'_n$ using the definite formula C' in S_i of which C is the unfolded result.

Suppose C is the result of negative unfolding. Then for some $j_1, j_2, \ldots, j_m$ $(1 \leq j_k \leq n)$, say $1, 2, \ldots, m$, the root labels $G_1, G_2, \ldots, G_m$ of $T_1, T_2, \ldots, T_m$ are ground goals obtained from G' by the negative unfolding. Because G' is true in $M(S^{old})$ if $G_1, G_2, \ldots, G_m$ are true in $M(S^{old})$ and $G_1, G_2, \ldots, G_m$ themselves are proofs $T'_1, T'_2, \ldots, T'_m$, we can construct T' from $T'_1, T'_2, \ldots, T'_n$ using the definite formula C' 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 goal 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 :- $G_1, G_2, \ldots, G_m$" of some definite formula in D such that $G_1, G_2, \ldots, G_m$ are provable in S_0. Again by I1, $G_1, G_2, \ldots, G_m$ are provable in S_i. Let C' be the clause in S_i of which C is the folded result. Owing to the condition of folding, we can combine the proofs of $G_1, G_2, \ldots, G_m$ and proofs $T'_2, T'_3, \ldots, T'_n$ with C' to obtain T', the proof of A in S_i.

Lemma 4.5.2. If the invariants I1 and I2 hold for S_i, then $M(S_{i+1}) \supseteq M(S_i)$.

Proof. Let A be a ground new goal 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 ovbiously 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 used at the root of T and $T_1, T_2, \ldots, T_n$ $(n \geq 0)$ be the immediate subproofs of T. When a root label G_i of T_i consists of old atoms, G_i itself is a proof T'_i. When G_i is a ground new atom, by the invariant I2, $A \gg G_i$ holds. So by the induction hypothesis there are proofs $T'_1, T'_2, \ldots, T'_n$ of $G_1, G_2, \ldots, G_n$ in S_{i+1}. If C is in S_{i+1}, the construction of T' is immediate.

Suppose C is positively unfolded into $C'_1, C'_2, \ldots, C'_k$ in S_{i+1} and assume that the root label G_1 of T_1 is the instance of the goal at which C is unfolded. Let $G_{11}, G_{12}, \ldots, G_{1k}$ be the ground instances of the goals to which G_1 is unfolded. Because some G_{1l} is true in $M(S^{old})$ if G_1 is true in $M(S^{old})$, G_{1l} is itself a proof T'_{1l} in $M(S_{i+1})$. Combining the proofs $T'_{1l}, T'_2, \ldots, T'_n$ with some C'_l $(1 \leq l \leq k)$, we get a proof T' of A in S_{i+1}.

Suppose C is negatively unfolded into C' in S_{i+1} and assume that the root labels G_1 of T_1 is the instance of the goal at which C is unfolded. Let $G_{10}, G_{11}, \ldots, G_{1k}$ be the ground goals to which G_1 is unfolded. Because all $G_{10}, G_{11}, \ldots, G_{1k}$ are true in $M(S^{old})$ if G_1 is true in $M(S^{old})$, $G_{10}, G_{11}, \ldots, G_{1k}$ are themselves proofs $T'_{10}, T'_{11}, \ldots, T'_{1k}$ in $M(S_{i+1})$. Combining the proofs $T'_{11}, T'_{12}, \ldots, T'_{1k}$, $T'_2, \ldots, T'_n$ with the definite clause C', we get a proof T' of A in S_{i+1}.

Now suppose C is folded into C' in S_{i+1}. Assume that the root labels $G_1, G_2, \ldots, G_k$ of $T_1, T_2, \ldots, T_k$ $(k \leq n)$ are the instances of the folded goals in C. Let B be a goal such that "B :- $G_1, G_2, \ldots, G_k$" is a ground instance of the definite clause in D used in the folding. By definition, $rank(G_1) \bigcup rank(G_2) \bigcup \cdots \bigcup rank(G_k) \succeq_M rank(B)$. By the condition (c) of folding, either C is marked "foldable", which means $rank(A) \succ_M rank(G_1) \bigcup rank(G_2) \bigcup \cdots \bigcup rank(G_k)$, or $k < n$. In either case, $rank(A) \gg rank(B)$ holds. Moreover, by the equivalence of S_i to S_0, B is provable in S_i. 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 4.5.3. If the invariants I1 and I2 hold for S_i, then I2 holds for S_{i+1}.

Proof. We first note that in the proof of Lemma 2, T' is constructed in such a way that it is rank-consistent. Thus every goal in $M(S_i)$ has a rank-consistent proof in S_{i+1}. Because $M(S_{i+1}) \subseteq M(S_i)$ by Lemma 4.5.1, I2 holds for S_{i+1}.

This completes the proof of the theorem.

5. Splitting Rules

In order that our construction system can obtain definite clause programs as its final results, we need several augmenting rules. In this section, we only show the simplest ones for splitting, which are unnecessary in Tamaki-Sato's transformation system but necessary in our system because of our generalization to definite formulas.

5.1. Positive Splitting

We have three splitting rules corresponding to positive subformula of goals of the forms $H_1 \vee H_2 \vee \cdots \vee H_k$, $H_1 \wedge H_2 \wedge \cdots \wedge H_k$ and $H_1 \supset H_2$.

Positive $\vee$ Splitting : Let C be a definite formula in P_{i-1} and G be a goal in the body. When H is a positive subformula of G of the form $H_1 \vee H_2 \vee \cdots \vee H_k$ ($k > 1$) and each free variable $!X$ appearing in H_i appears only in H_i ($1 \leq i \leq k$), let $C'_1, C'_2, \ldots, C'_k$ be the results of replacing H in C with $H_1, H_2, \ldots, H_k$ respectively, i.e., $C_H[H_1], C_H[H_2], \ldots, C_H[H_k]$. Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'_1, C'_2, \ldots, C'_k\}$ and D_i be D_{i-1}. Mark each C'_i "foldable" unless it is already in P_{i-1}.

Example 5.1.1. If the *member* relation is defined by its general form

member(X,L) :- $(X{=}X_1 \wedge L{=}[X_1|L_1]) \vee (X{=}X_2 \wedge L{=}[Y_2|L_2] \wedge \text{member}(X_2,L_2))$.

we can apply the positive $\vee$ splitting to the body and have

member(X,L) :- $X{=}X_1 \wedge L{=}[X_1|L_1]$.

member(X,L) :- $X{=}X_2 \wedge L{=}[Y_2|L_2] \wedge \text{member}(X_2,L_2)$.

Positive $\wedge$ Splitting : Let C be a definite formula in P_{i-1} and G be a goal in the body. When H is a positive subformula of G of the form $H_1 \wedge H_2 \wedge \cdots \wedge H_k$ ($k > 1$), let $G'_1, G'_2, \ldots, G'_k$ be the results of replacing H with $H_1, H_2, \ldots, H_k$ respectively, i.e., $G_H[H_1]$, $G_H[H_2], \ldots, G_H[H_k]$ and C' be the results of replacing G in C with $G_1, G_2, \ldots, G_k$. Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'\}$ and D_i be D_{i-1}. Mark C' "foldable" unless it is already in P_{i-1}.

Example 5.1.2. After the positive $\vee$ splitting in Example 5.1.1, we can apply the positive $\wedge$ splitting and have

member(X,L) :- $X{=}X_1, L{=}[X_1|L_1]$.

member(X,L) :- $X{=}X_2, L{=}[Y_2|L_2]$, member(X_2,L_2).

from which we can obtain the usual definition of *member*

member(U,[U|L]).

member(U,[V|L]) :- member(U,L).

by positive unfoldings on the equations of the bodies.

Positive $\supset$ Splitting : Let C be a definite formula in P_{i-1} and G be a goal in the body. When H is a positive subformula of G of the form $H_1 \supset H_2$ and each free variable appearing in H_i appears only in H_i ($1 \leq i \leq 2$), let C'_1 and C'_2 be the results of replacing H in C with $\neg H_1$ and H_2 respectively, i.e., $C_H[\neg H_1]$ and $C_H[H_2]$. Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'_1, C'_2\}$

and D_i be D_{i-1}. Mark each C'_i "foldable" unless it is already in P_{i-1}.

Example 5.1.3. Let *not-lost* be a predicate defined by
not-lost(Chess-Board) :- be-checked(Chess-Board)$\supset$escapable(Chess-Board).
Then by applying positive $\supset$ splitting,we have
not-lost(Chess-Board) :- $\neg$be-checked(Chess-Board).
not-lost(Chess-Board) :- escapable(Chess-Board).

5.2. Negative Case Splitting

Again we have three splitting rules corresponding to negative subformula of goals of the forms $H_1 \vee H_2 \vee \cdots \vee H_k$, $H_1 \wedge H_2 \wedge \cdots \wedge H_k$ and $H_1 \supset H_2$.

Negative $\vee$ Splitting : Let C be a definite formula in P_{i-1} and G be a goal in the body. When H is a negative subformula of G of the form $H_1 \vee H_2 \vee \cdots \vee H_k$ $(k > 1)$, let $G'_1, G'_2, \ldots, G'_k$ be the results of replacing H with $H_1, H_2, \ldots, H_k$ respectively, i.e., $G_H[H_1]$,$G_H[H_2]$,...,$G_H[H_k]$ and C' be the result of replacing G in C with $G_1, G_2, \ldots, G_k$. Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'\}$ and D_i be D_{i-1}. Mark C' "foldable" unless it is already in P_{i-1}.

Negative $\wedge$ Splitting : Let C be a definite formula in P_{i-1} and G be a goal in the body. When H is an outermost negative subformula of the body of the form $H_1 \wedge H_2 \wedge \cdots \wedge H_k$ $(k > 1)$ and each free variable $!X$ appearing in H_i appears only in H_i $(1 \leq i \leq k)$, let $C'_1, C'_2, \ldots, C'_k$ be the results of replacing H in C with $H_1, H_2, \ldots, H_k$ respectively, i.e., $C_H[H_1]$,$C_H[H_2]$,...,$C_H[H_k]$. Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'_1, C'_2, \ldots, C'_k\}$ and D_i be D_{i-1}. Mark each C'_i "foldable" unless it is already in P_{i-1}.

Negative $\supset$ Splitting : Let C be a definite formulaf in P_{i-1} and G be a goal in the body. When H is an outermost negative subformula of some goal G in the body of the form $H_1 \supset H_2$, let G'_1 and G'_2 be the results of replacing H with H_1 and $\neg H_2$ respectively, i.e., $G_H[\neg H_1]$ and $G_H[H_2]$ and C' be the result of replacing G in C with G'_1 and G'_2. Then let P_i be $(P_{i-1} - \{C\}) \bigcup \{C'\}$ and D_i be D_{i-1}. Mark C' "foldable" unless it is already in P_{i-1}.

5.3. Safety of the Splitting Rules

Tamaki and Sato [24] discussed about various augmenting rules as well. As was noted by them, replacements of goal sets with its equivalent ones do not necessarily preserve minimum Herbrand models when they are combined with the unfold/fold rules. The augmenting rules which preserve minimum Herbrand models are said to be *safe* by them. In this section,we show that our splitting rules are always safe, which suggests a general way to discuss the safety of another augmenting rules.

Theorem 5.3. Splitting rules are safe.

Outline of Proof. Suppose, in general, that a definite formula C is replaced with a definite formula C', where C' is obtained by replacing a goal set Gs in the body of C with another goal set Gs'. Then it is enough to show that $Gs' \prec_M Gs$. As to the splitting rules, we add the following to the definition of $\prec$.

(S) When G' is obtained from G in any of the splitting rules, $G' \prec G$.

Then the proof in 4.5 goes in the completely same way using the new ordering $\lll$.

6. Discussion

Unfold/fold approaches are well-known and have been studied by many researchers. Our new contributions in this paper are the following three.

(i) We have extended the class of formulas permitted as definitions.

Our theorem is a generalization of the equivalence preservation theorem by Tamaki and Sato [24]. They focused their attentions on transformation, where the definition rule is always done by definite clauses. Sato and Tamaki [19] have also studied program synthesis from more general specifications and developed a technique called *double negation* [19]. Our approach unifies their transformation and a part of their synthesis by extending the class of formulas permitted as definitions and generalizing the unfold/fold rules as well as by introducing splitting rules to cover some of their synthesis methods.

Clark [5] permitted a more general class of formulas as definitions, where goals in our paper are any first order formulas. We have restritced the goals to be two-layered, i.e.,containing global and free variables because of two reasons.

One reason is that, even with such a restriction, definite formulas are fairly effective considering its easiness to implement. One might say our definite formulas are too restrictive to use as definitions of predicates to be constructed. But, a fairly lot of examples published in literatures can be defined by definite formulas (with slight modification). In addition, because we have only global and free variables, we only needs distinction of two kind of variables and a little care of unification.

(ii) We have clarified the importance of "terminating" and "must-be-true".

Another reason of our restriction on goals is its simplicity to present the theoretical result without too much complication. Though Clark discussed deduction based construction of Prolog programs from more general class of formulas, it has been open whether and when his construction preserves equivalence. It is not very difficult to extend our goals to full first order formulas and present the method with explicit quantifiers, as was done by Clark, and indeed it will eliminate some unnatural conditions on free and internal variables in our positive unfoldings. We expect that, even if such an extention is done, our discussion still holds. But we are afraid that it makes it slightly hard to see the preservation of minimum Herbrand models.

Here we explain more why these restrictions are necessary by examples. Why must atoms in positive unfoldings be terminating when they are not must-be-true?

Example 6.1. Let *loop,true-and-loop* and *is-true* be predicates defined by

loop(X) :- loop(X).
true-and-loop(X) :- is-true,true-and-loop(X).
is-true.

Supose we have defined *vacantly-true* by

vacantly-true(X) :- loop(X)$\supset$true-and-loop(X).

Then,we obtain

vacantly-true(X) :- loop(X)$\supset$true-and-loop(X).

after positive unfoldings on *true-and-loop*(X) and *is-true*. We should not mark the definite formula "foldable", because, if we did, we would have vacantly-true(X) :- vacantly-true(X).

by folding, whose minimum Herbrand model is different from that of the initial definition of *vacantly-true*. One might think that it works if we adopt a positive unfolding rule such that it does not mark "foldable" and inherit the mark when it is done on non-terminating atoms and mark "foldable" only when it is done on terminating atoms. But the example above is against it.

Why must atoms in negative unfoldings terminating? It is already obvious from Example 6.1. This suggests that finite-failure sets are not preserved in Tamaki-Sato's transformation in general.

Example 6.2. Let us redefine the tautologically true predicate *true-or-loop* in Example 2.2.2 using definite formulas as follows.

true-or-loop(X) :- ¬(is-false ∧loop(X)).

loop(X) :- loop(X).

Of course, *loop* is not terminating. Then by unfolding on $loop(X)$ and folding, we have

true-or-loop(X) :- true-or-loop(X).

for which no ground goal $true\text{-}or\text{-}loop(t)$ succeeds. This example is obtained from the following example due to Tamaki [23] showing that Tamaki-Sato's transformation does not always preserve finite-failure sets.

false-and-loop(X) :- is-false,loop(X).

loop(X) :- loop(X).

One may wonder why any instance of A must be terminating when all global variables are instantiated to ground terms.

Example 6.3. Let our Herbrand universe be $\{0, suc(0), suc(suc(0)), \ldots\}$ and *number* and *is-false* be predicates defined by

number(0).

number(suc(X)) :- number(X).

Suppose we have defined

true-or-not-number(X) :- ¬(is-false∧number(!Y)).

The predicate *true-or-not-number* is intended to be tautologically true. If we had not the condition, we would unfold on $number(!Y)$ as follows.

true-or-not-number(X) :- ¬(is-false∧number(0)), ¬(is-false∧number(!Y)).

The first goal would be reduced to *true* by unfolding *is-false*. Thus, by folding by $true\text{-}or\text{-}not\text{-}number(X)$, we would have

true-or-not-number(X) :- true-or-not-number(X).

for which no ground atom succeeds.

(iii) We have devised a more abstract definition of the rank.

The definition of rank by Tamaki and Sato is more concrete. The rank of a ground atom A in their proof is a mapping $rank : M(S_0) \mapsto \mathrm{N}$ defined as follows.

(a) $rank(A)$ is the minimum size of the proof of A when A has an old predicate.

(b) $rank(A)$ is the minimum size of the proof of A minus one when A has a new predicate.

We generalized it to more abstract one based on the ordering $\prec_M$ on the multiset of $\overline{M}(S^{old})$. The intuitive meaning of our orderings is as follows. When we unfold at an atom in a goal, these unfoldings contribute somehow to know whether the goal is true or not except two cases. One is the case in which whether the goal is true or not does not depend on whether the atom is true or not. Another is the case in which usual one-step SLD-resolutions in execution do not advance us closer to know whether the goal is true or those in the "Negation as Failure" [4] do not advance us closer to know whether the goal is false. The mechanism of marking definite formulas "foldable" or inheritting them, with the conditions of unfolding, guarantees that foldings are done only after we get strictly closer somhow to the consequences.

We expect that this abstract definition of $\prec$ still works even if the definition of our goals and unfolding rules are extended.

7. Conclusions

We have presented a method to construct logic programs based on generalized unfold/fold rules. This method is being used in Argus/C, a system for construction of Prolog programs under development [11],[12].

Acknowledgements

This work is stemed from our experience in using extended execution for first order inference and Tamaki-Sato's transformation [24] for manipulation of computational induction schemes in our verification system Argus/V developed in 1984. Our result is an extension of the result by Tamaki and Sato [24]. The authors would like to express deep gratitude to Dr.T.Sato (Electrotechnical Laboratory) and Prof.H.Tamaki (Ibaraki University) for their stimiulating and perspicuous works.

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

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

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

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

[6] Dershowitz,N.and Z.Manna, "Proving Termination with Multiset Orderings", C.ACM, Vol.22, No.8, pp.465-476, 1979.

[7] 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.

[8] Eriksson,L-H., "Synthesis of A Unification Algorithm in A Logic Programming Calculus", J.Logic Programming, Vol.1. pp.3-18, 1984.

[9] Hansson,A. and S-Å.Tärnlund, "A Natural Programming Calculus", Proc.of 6th International Joint Conference on Artificial Intelligence, pp.348-355, 1979.

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

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

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

[13] Kott,L., "Unfold/Fold Program Transformations", INRIA Research Report, No.155, August, 1982.

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

[15] Manna,Z.and R.Waldinger, "A Deductive Approach to Program Synthesis", ACM Trans. on Programming Languages and System, Vol.2, No.1, pp.90-121, 1980.

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

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

[18] Prawitz,D., "Natural Deduction, A Proof Theoretical Study", Almqvist & Wiksell,Stockholm, 1965.

[19] Sato,T.and H.Tamaki, "Transformational Logic Program Synthesis", Proc.of International Conference on Fifth Generation Computer Systems 1984, pp.195-201, 1984.

[20] Scherlis,W.L., "Expression Procedures and Program Derivation", STAN-CS-80-818, Stanford University, Computer Science Department, 1980.

[21] Scherlis,W.L., "Program Improvement by Internal Specialization", Proc.of 8th Symposium of Princilples of Programming Languages, pp.41-49, 1981.

[22] Schütte,K., "Proof Theory", (translated by J.N.Crossley), Springer Verlag, 1977.

[23] Tamaki,H., private communication, January, 1984.

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