ソフトウェア基礎論 28－5
(1989. 3. 10)

論理型プログラムのUnfold/Fold変換における より強い等価性の保存 (II)

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

金森 直　　川村 正
Tadashi KANAMORI　Tadashi KAWAMURA
三菱電機(株)中央研究所
Mitsubishi Electric Corporation Central Research Laboratory

あらまし　本稿では、玉木－佐藤のPrologプログラムの Unfold/Fold変換が、玉木－佐藤の示した最小エルブラン・モデルの等価性よりも強い意味での等価性を保存することを示す。通常、prologプログラムの意味は最小エルブラン・モデルで定義される。しかし、最小エルブラン・モデルはどのような解代入が返されるか、あるいは同じ解が何回返されるかを必ずしも特徴付けるとは限らない。本稿では、玉木－佐藤の変換によって初期プログラムから得られるすべてのプログラムは、どのようなトップレベルのゴールに対しても、初期プログラムと同じ解を同じ回数だけ返すことを証明する。

Abstract　This paper shows that Tamaki-Sato's unfold/fold transformation of Prolog programs preserves equivalence in a stronger sense than that of the usual least Herbrand model semantics, which Tamaki and Sato originally showed. Conventionaly, the semantics of Prolog programs is defined by the least Herbrand model. However, the least Herbrand model does not always characterize what answer substitutions are returned nor how many times the same answer substitutions are returned. This paper proves that any program obtained from an initial program by applying Tamaki-Sato's transformation can compute the same answer substitutions the same number of times as the initial program.

1. はじめに

宣言的に記述されたプログラムより自動的に効率良く動作するプログラムを導き出すことがプログラム変換の目標である。Unfold/Fold変換はそのようなプログラム変換規則の一つであり、最初、BurstallとDarlington[1]によって関数型プログラミング言語に対する変換規則として導入された。また、MannaとWaldinger[5]は同様な規則をプログラム合成の技法として用いた。ところで、プログラム変換の目的は、もとのプログラムと同じ動作をするより効率の良いプログラムを導くことであるから、プログラムの等価性を保存することは変換規則の重要な性質である。ここで、プログラムの等価性はプログラムの意味(Semantics)に基づいて定義されるので、異なる意味によって異なる等価性の概念が定義され得る(cf. Maher[4])。Prologプログラムの意味は、通常、最小エルブラン・モデル(Least Herbrand Model)で定義される。プログラム変換における等価性の保存に関する研究では、玉木と佐藤がPrologプログラムの最小エルブラン・モデルの意味での等価性を保存するUnfold/Fold変換を提案した[6][7][8]。しかし、最小エルブラン・モデルは、プログラムがどのような解代入(answer substitution)を返すか、あるいは同じ解を何回返すかを特徴付け

るとは限らない。例として、次の3つのプログラムP_1、P_2、P_3を考える。

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

P_1、P_2、P_3のエルブラン空間は{a}であるから、これら3つのプログラムは最小エルブラン・モデルの意味で等価である。しかし、ゴール

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

が与えられると、P_1は解として空代入<>を返すのに対し、P_2とP_3は代入<X⇐a>を返す。さらに、P_2はこの解代入を1回だけ返すが、P_3は2回返す。これらのプログラムを区別するためには、より強い等価性の概念が必要である。

本稿では、玉木－佐藤のPrologプログラムのUnfold/Fold変換が最小エルブラン・モデルより強い意味での等価性を保存することを示す。2節では玉木－佐藤のPrologプログラム変換について述べる。3節では、Prologプログラムの意味として、トップレベルのゴールと解代入との対の多重集合(multiset)を導入し、玉木－佐藤の変換がこの意味での等価性も保存することを証明する。

2. Prologプログラムの Unfold/Fold変換

この節では、玉木－佐藤の Unfold/Fold変換[8]について述べる。

定義 プログラム

節は、節識別子と確定節の対から成る。プログラムは、節の有限集合である。

プログラム中のどの節も同じ節識別子を持たないとし、同じ形の節は節識別子によって区別される。

定義 初期プログラム

初期プログラムP_0は、次の条件を満たすプログラムである。

(a) P_0は2つの節の多重集合P_{new}とP_{old}に分けられる。P_{new}で定義される述語は新述語と呼ばれ、P_{old}で定義される述語は旧述語と呼ばれる。

(b) 新述語はP_{new}中の節の頭部(head)にのみ現れ、P_{old}にもP_{new}中の節の本体(body)にも現れない。

例 2.1 $P_0=\{C_1, C_2, C_3\}$を初期プログラムとする。ここで、

```
C1: ap([],M,M).
C2: ap([X|L],M,[X|N]):-ap(L,M,N).
C3: insert(X,M,N)
        :-ap(U,V,M),ap(U,[X|V],N).
```

であり、$P_{old}=\{C_1, C_2\}$、$P_{new}=\{C_3\}$ であるとする。このとき、'ap'は旧述語であり、'insert'は新述語である。（ここで、C_1, C_2, C_3は節識別子である。）

定義 Unfolding

P_iをプログラム、CをP_i中の節、AをCの本体に現れるアトムとする。また、$C_1, C_2, \cdots, C_k$を頭部がCと単一化可能(unifiable)なP_i中の全ての節とし、このときのm.g.u.を$\theta_1, \theta_2, \cdots, \theta_k$とする。ここで、$C_j'$をC中のアトムAを$C_j$の本体で置き換えた後、$\theta_j$を適用して得られた節とする。このとき、$P_{i+1}=(P_i-\{C\})\cup\{C_1', C_2', \cdots, C_k'\}$である。

例 2.2 P_0を上のプログラムとする。C_3をアトム'ap(U,V,M)'でunfoldして、プログラム $P_1=\{C_1, C_2, C_4, C_5\}$が得られる。ここで、

```
C4: insert(X,M,N):-ap([],[X|M],N).
C5: insert(X,[Y|M],N)
        :-ap(U,V,M),ap([Y|U],[X|V],N).
```

である。さらに、C_4とC_5をunfoldすることにより、$P_2=\{C_1, C_2, C_5, C_6\}$と$P_3=\{C_1, C_2, C_6, C_7\}$が得られる。ここで、

```
C6: insert(X,M,[X|M]).
C7: insert(X,[Y|M],[Y|N])
        :-ap(U,V,M),ap(U,[X|V],N).
```

である。

定義 Folding

P_iをプログラム、Cを

```
A0:-A1,A2,…,An(n>0)
```

の形をしたP_i 中の節、Dを

```
B0:-B1,B2,…,Bm(m>0)
```

の形をしたP_{new}中の節とする。ここで、次の条件を満足する代入θが存在するとする。

(a) 異なる自然数$j_1, j_2, \cdots, j_m$に対し、$B_{j_1}\theta=$

$A_{j1}, B_{j2}\theta = A_{j2}, \cdots, B_{jm}\theta = A_{jm}$である。

(b) θはDの本体にのみ現れる変数に、$\{A_0, A_1, \cdots, A_n\} - \{A_{j1}, A_{j2}, \cdots, A_{jm}\}$ に現れない相異なる変数を代入する。

(c) Dは P_{new}中で頭部が$B_0\theta$と単一化可能な唯一の節である。

(d) Cの頭部の述語が旧述語であるか、C はP_0, $P_1, \cdots, P_i$の中で少なくとも一回unfoldされている。

C'を頭部がA_0で本体が$\{B_0\theta\} \cup (\{A_0, A_1, \cdots, A_n\} - \{A_{j1}, A_{j2}, \cdots, A_{jm}\})$であるような節とする。このとき、$P_{i+1} = (P_i - \{C\}) \cup \{C'\}$ である。ここで、Cをfolded節、 Dをfolding節と呼ぶ。

例 2.3　P_3を上のプログラムとする。このとき、C_7の本体をC_3でfoldして、$P_4 = \{C_1, C_2, C_6, C_8\}$が得られる。ここで、

```
C8: insert(X,[Y|M],[Y|N])
        :-insert(X,M,N).
```

である。

定義　変換系列(Transformation Sequence)

P_0を初期プログラム、P_{i+1}をP_iにunfoldingまたはfoldingを適用して得られたプログラムとする(i>0)。このとき、プログラムの系列 $P_0, P_1, \cdots, P_N$をP_0から始まる変換系列と呼ぶ。

例 2.4　例 2.1-2.3で現れた系列P_0, P_1, P_2, P_3, P_4は例 2.1のP_0から始まる変換系列である。ここで、ゴール

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

に対し、これら５つのプログラムは同じ解代入

<N⇐[X,X,Y]>, <N⇐[X,X,Y]>, <N⇐[X,Y,X]>

を返す。

3. より強い等価性の保存

この節では、まず証明木(Proof Tree)に関する定義を行った後、より強い等価性の保存の証明を行う。証明の枠組みは[8][7]での証明と同様である。本稿で記されていない定理や補題の証明については、[3]を参照されたい。

3. 1　証明木

与えられたトップレベルのゴールに対してどのような解代入が返されるかを考える。このとき、導出の際のアトムの選択が非決定的に行われることを考慮しながら証明木を構成すると、その手続きは非常に煩雑になる。この煩雑さを避けるため、以下のように証明木を定義する。

定義　ラベル付き木(Labelled Tree)

ラベル付き木は、節点(node)が(“A=B”, C)の形の表現でラベル付けされた有限木である。ここで、AとBは単一化可能なアトム、C は節識別子である。ラベル付き木 Tの全てのラベルの集合をTのラベル集合と呼ぶ。また、Tの節点の数をTの大きさと呼ぶ。

定義　ラベル付き木のm.g.u.

Tをラベル付き木、 $E = \{A_1 = B_1, A_2 = B_2, \cdots, A_k = B_k\}$をTのラベル集合とする。このとき、i=1, 2, …, kに対し、$A_i\sigma$と$B_i\sigma$が同一になる代入σが存在するならば、T(またはE) は単一化可能であると言う。代入τがそのような代入のうち最も一般的なものであるとき、τをT(またはE)のm.g.u.と呼ぶ。

定義　代入のm.g.u.

$\sigma_1, \sigma_2, \cdots, \sigma_n$を代入とする。ある代入$\sigma$が存在し、この$\sigma$と各$\sigma_i$に対して、$\sigma = \sigma_i\tau_i$を満足するような代入$\tau_i$が存在するとする。このとき、$\sigma_1, \sigma_2, \cdots, \sigma_n$は単一化可能であると言う。代入$\tau$がこのような代入のうち最も一般的なものであるとき、τを代入$\sigma_1, \sigma_2, \cdots, \sigma_n$のm.g.u.と呼ぶ。

定義　証明木

P をプログラム、T をラベル付き木、$T_1, T_2, \cdots, T_n$をTの直接部分木(immediate subtree) とする。P中に

```
B:-B1, B2, …, Bn.
```

の形をした節Cが存在し、これが次の条件を満たすとき、Tをアトム Aと解代入σのPによる証明木と呼ぶ。

(a) AとBはm.g.u.θで単一化可能である。

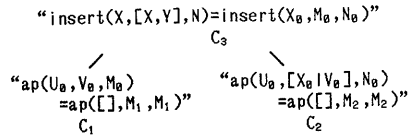
(b) Tの根の節点のラベルは(“A=B”, C)である。

(c) $T_1, T_2, \cdots, T_n$はそれぞれ$B_1, B_2, \cdots, B_n$と解代入$\sigma_1, \sigma_2, \cdots, \sigma_n$のPによる証明木である。

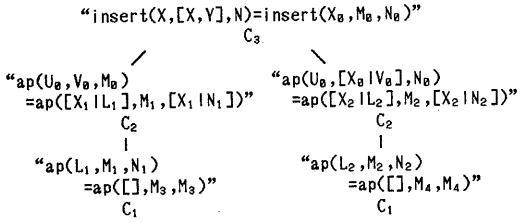
(d) σは$\theta, \sigma_1, \sigma_2, \cdots, \sigma_n$のm.g.u.の、A中の変数への制約である。

ここで、CをTの根で用いられた節、$T_1, T_2, \cdots, T_n$をTの直接部分証明(immediate subproof)と呼ぶ。

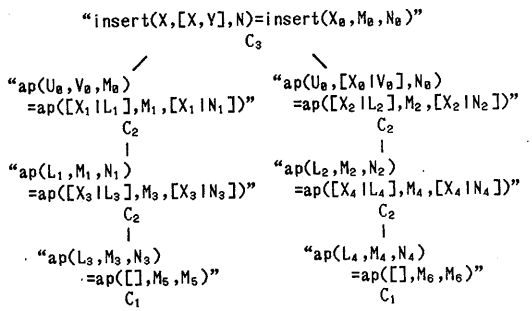
例 3.1.1　P_0を例 2.1のプログラムとする。このとき、'insert(X,[X,Y],N)'と解代入<N⇐[X,X,Y]>のP_0による証明木T_1は次の様になる。



'insert(X,[X,Y],N)'と解代入<N⇐[X,X,Y]>のP_0によるもう1つの証明木T_2は次の様になる。



'insert(X,[X,Y],N)'と解代入<N⇐[X,Y,X]>のP_0による証明木T_3は次の様になる。



定義　成功多重集合(Success Multiset)

Pをプログラムとする。アトムAと解代入σのPによる証明木が存在するような全てのアトム－代入対(A,σ)の多重集合をPの成功多重集合と呼び、M(P)で表す。

M(P)は集合ではなく多重集合であることに留意されたい。即ち、アトムAと解代入σのPによる証明木で相異なるものがk個存在する場合、M(P)はk個のアトム－代入対(A,σ)を含む。

補題 3.1.1　TをアトムAと解代入σの証明木とする。このとき、σはTのラベル集合のm.g.u.中の、A中の変数への制約である。

補題 3.1.2　Eを証明木Tのラベル集合、"A=B"をEの要素、θをAとBのm.g.u.とする。このとき、τが$(E-\{A=B\})\theta$のm.g.u.であるとき、またそのときに限り、$\theta\tau$はEのm.g.u.である。

3．2　証明木集合間の写像

成功多重集合は最小エルブラン・モデルより正確にPrologプログラムを特徴付ける。しかし、成功多重集合の保存を直接取り扱うのは容易ではない。そこで、以下のような概念を定義する。

定義　証明木集合（Proof Tree Set）

Pをプログラムとする。Pによるすべての証明木の集合をPの証明木集合と呼び、$\mathcal{T}(P)$で表す。

成功多重集合中の各アトム－代入対は、証明木集合中の1つの証明木に対応している。従って、2つの成功多重集合が等しいのは対応する証明木集合の間に1対1の対応が存在するときである。そこで、以下の議論では、証明木集合間の写像を考える。

定義　証明木集合間の無矛盾写像

P_iとP_jをプログラム、fを$\mathcal{T}(P_i)$から$\mathcal{T}(P_j)$への写像とする。fが$\mathcal{T}(P_i)$中のアトムAと解代入σの証明木Tを$\mathcal{T}(P_j)$中の同じアトムAと同じ解代入σの証明木T'へ写像するとき、fを$\mathcal{T}(P_i)$から$\mathcal{T}(P_j)$への無矛盾写像と呼び、$\mathcal{T}(P_i) \overset{f}{\pitchfork} \mathcal{T}(P_j)$で表す。ある無矛盾写像fについて、$T_1=T_2$のとき、またそのときに限り$f(T_1)=f(T_2)$が成立するならば、fは1対1であると言う。

定義　証明木集合間の無矛盾写像対

P_iとP_jをプログラムとする。写像の対(f,g)は、以下の条件を満たすとき、$\mathcal{T}(P_i)$と$\mathcal{T}(P_j)$の間の無矛盾写像対であると言い、$\mathcal{T}(P_i) \overset{f,g}{\pitchfork} \mathcal{T}(P_j)$で表す。

(a) fは$\mathcal{T}(P_i)$から$\mathcal{T}(P_j)$への無矛盾写像である。

(b) gは$\mathcal{T}(P_j)$から$\mathcal{T}(P_i)$への無矛盾写像である。

(c) $g \circ f = id_{\mathcal{T}(P_i)}$であり、$f \circ g = id_{\mathcal{T}(P_j)}$である。

ここで、$id_{\mathcal{T}(P_i)}$、$id_{\mathcal{T}(P_j)}$はそれぞれ$\mathcal{T}(P_i)$、$\mathcal{T}(P_j)$上の恒等写像である。

3．3　部分正当性(Partial Correctness)

P_0をプログラム、P_iを変換規則を適用してP_0から得られたプログラムとする。Prologプログラムの変換は、$M(P_0) \supseteq M(P_i)$が成り立つとき、部分正当であると言う。この節では、部分正当性を証明する。証明は、$T(P_i)$から$T(P_0)$への1対1の無矛盾写像の存在を示すことによって行われる。

補題3.3.1　P_iを変換系列中のプログラム、CをP_i中の節とする。また、C'をCの本体のアトムを並べ変えて得られる節、P_i'を$(P_i-\{C\})\cup\{C'\}$とする。今、fを$T(P_i)$から$T(P_i')$への写像とする。fは、TとT'が同一であるか、CからC'への並べ変えに応じて証明木Tの部分証明を並べ変えて証明木T'が得られるとき、またそのときに限りTをT'に写像するものとする。また、gをfの逆写像とする。このとき、(f,g)は$T(P_i)$と$T(P_i')$の間の無矛盾写像対である。

この補題は、プログラムP_i中の節の本体のアトムを並べ変えても、プログラムP_0とP_iの間の無矛盾写像対の系列の有無には影響しないことを示している。

補題3.3.2　P_iを変換系列中のプログラム、Tをアトム$A\theta$と解代入σのP_iによる証明木とする。T'をTの根のラベルの等式の左側の$A\theta$をAで置き換えて得られるラベル付き木とする。このとき、T'はアトムAのP_iによる証明木である。

補題3.3.3　P_iを変換系列中のプログラム、Tをアトム$A\theta$と解代入σのP_iによる証明木、θをA中の変数への代入とし、θとσは単一化可能であるとする。T'をTの根のラベルの等式の左側の$A\theta$を$A\theta$で置き換えて得られるラベル付き木とする。このとき、T'はアトム$A\theta$のP_iによる証明木である。

補題3.3.4　$P_0, P_1, \cdots, P_N$を変換系列とする。i=0,1,…,N-1に対し、無矛盾写像対

$$T(P_0) \leftrightarrows T(P_1) \leftrightarrows T(P_2) \leftrightarrows \cdots \leftrightarrows T(P_i)$$

が存在するならば、$T(P_{i+1})$から$T(P_i)$への無矛盾写像g_{i+1}が存在する。

補題3.3.5　g_{i+1}を補題3.3.4で定義された無矛盾写像とする。このとき、g_{i+1}は1対1である。

3．4　全正当性(Total Correctness)

P_0をプログラム、P_iを変換規則を適用してP_0から得られたプログラムとする。Prologプログラムの変換は、$M(P_0)=M(P_i)$が成り立つとき、全正当であると言う。この節では、全正当性を証明する。

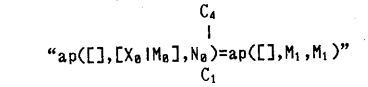
定義　原証明木(Original Proof Tree)

変換系列$P_0, P_1, \cdots, P_i$に対し、無矛盾写像対

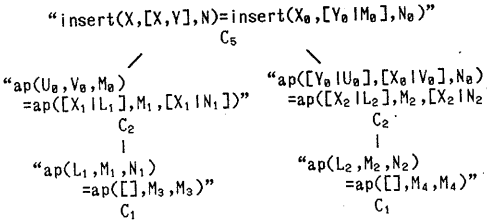
$$T(P_0) \leftrightarrows T(P_1) \leftrightarrows T(P_2) \leftrightarrows \cdots \leftrightarrows T(P_i)$$

が存在するとする。T_0を$T(P_0)$中の証明木、T_iを$T(P_i)$中の証明木で$f_1, f_2, \cdots, f_i$をT_0に続けて適用して得られるものとする。（もしくは、T_iを$T(P_i)$中の証明木、T_0を$T(P_0)$中の証明木で$g_i, \cdots, g_2, g_1$をT_iに続けて適用して得られるものとする、と言い替えてもよい。）このとき、T_0をT_iの原証明木と呼ぶ。

例3.4.1　P_0, P_1を例2.2の変換系列、T_1, T_2を例3.1.1の証明木とする。今、T_1'は'insert(X,[X,Y],N)'と解代入<N⇐[X,X,Y]>のP_1による証明木であり、次の様に描かれるとする。



また、T_2'はもう1つの'insert(X,[X,Y],N)'と解代入<N⇐[X,X,Y]>のP_1による証明木であり、次の様に描かれるとする。



f_1は$T(P_0)$から$T(P_1)$への無矛盾写像であり、$f_1(T_1)=T_1'$、$f_2(T_2)=T_2'$とする。このとき、T_1はT_1'の原証明木であり、T_2はT_2'の原証明木である。

以下の定義と補題3.4.1-5では、P_iは変換系列中のプログラムであり、上のように無矛盾写像

対の列$(f_1, g_1), (f_2, g_2), \cdots, (f_i, g_i)$が存在すると仮定する。

定義　証明木の重み(Weight of Proof Tree)

P_iを変換系列の初期プログラム、Tをアトム$A\sigma$のP_iによる証明木、$T_\emptyset$をTの原証明木、sを$T_\emptyset$の大きさとする。このとき、Tの重み w(T)は、次のように定義される。

$$w(T)=\begin{cases} s-1, & A\sigma\text{の述語が新述語のとき。} \\ s, & A\sigma\text{の述語が旧述語のとき。} \end{cases}$$

例 3.4.2　$P_\emptyset$を例 2.1の初期プログラム、T_1、T_2、T_3を例 3.1.1の証明木とする。このとき、$w(T_1)=2$、$w(T_2)=4$、$w(T_3)=6$である。

以下の概念は、[8] に現れたものの一般化であり、以下の証明で重要な役割を果たす。

定義　重み完全(Weight Completeness)

変換系列中のプログラムP_iは、次の条件を満たすとき、$((f_1, g_1), (f_2, g_2), \cdots, (f_i, g_i))$に関して）重み完全であると言う。

Tを P_iによる証明木、CをTの根で用いられた節、$T_1, T_2, \cdots, T_n$をTの直接部分証明とする。

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

(b) CがFoldingの条件(d)を満たすとき、
$w(T) > w(T_1)+w(T_2)+\cdots+w(T_n)$

定義　証明木集合上の有礎順序$\succ$

P_iを変換系列中のプログラムとする。P_iの証明木集合上の有礎順序$\succ$は以下のように定義される。

Tをアトム$A\sigma$のP_iによる証明木、T'をアトムA'のP_iによる証明木とする。次の(a)(b)のいずれかが成り立つとき、またそのときに限り $T \succ T'$である。

(a) $w(T) > w(T')$

(b) $w(T) = w(T')$、かつAの述語が新述語であり、A'の述語が旧述語である。

次の３つの補題はそれぞれ補題3.3.1、3.3.2、3.3.3の拡張である。

補題 3.4.1　P_iを変換系列中のプログラム、CをP_i中の節とする。また、C'をCの本体のアトムを並べ変えて得られる節、P_i'を$(P_i-\{C\})\cup\{C'\}$とする。今、fを$\mathcal{T}(P_i)$から$\mathcal{T}(P_i')$への写像とする。fは、TとT'が同一であるか、CからC'への並べ変えに応じて証明木Tの部分証明を並べ変えて証明木T'が得られるとき、またそのときに限りTをT'に写像するものとする。また、gをfの逆写像とする。このとき、(f, g)は$\mathcal{T}(P_i)$と$\mathcal{T}(P_i')$の間の無矛盾写像対である。また、P_i'が$(f_1, g_1), (f_2, g_2), \cdots, (f\circ f_i, g\circ g_i)$に関して重み完全であるとき、またそのときに限り、$P_i$は$(f_1, g_1), (f_2, g_2), \cdots, (f_i, g_i)$に関して重み完全である。

この補題は、プログラムP_i中の節の本体のアトムを並べ変えても、プログラム$P_\emptyset$とP_iの間の無矛盾写像対の系列の有無及びP_iの重み完全性には影響しないことを示している。

補題 3.4.2　$P_\emptyset$を変換系列の初期プログラム、Tをアトム$A\theta$と解代入σの$P_\emptyset$による証明木とする。ここで、T'をTの根のラベルの左側の$A\theta$をAで置換えて得られるラベル付き木とする。このとき、T'はアトム$A\sigma$の $P_\emptyset$よる証明木であり、$w(T)=w(T')$が成り立つ。

補題 3.4.3　$P_\emptyset$を変換系列の初期プログラム、Tをアトム$A\sigma$と解代入σの$P_\emptyset$による証明木、θをA中の変数への代入とし、θとσは単一化可能であるとする。ここで、T'をTの根のラベルの左側の$A\sigma$を$A\theta$で置き換えて得られるラベル付き木とする。このとき、T'はアトム$A\theta\sigma$の $P_\emptyset$による証明木であり、$w(T)=w(T')$が成り立つ。

補題 3.4.4　P_iを $P_\emptyset$から始まる変換系列の初期プログラム、Cを P_i中の節とする。Cがfoldingの条件(d)を満足しないならば、Cの本体のアトムの述語はすべて旧述語である。

補題 3.4.5　P_iを変換系列中のプログラム、TをP_iによる任意の証明木、$T_1, T_2, \cdots, T_n$をTの直接部分証明とする。このとき、P_iが重み完全ならば、$j=1, 2, \cdots, n$に対して$T \succ T_j$が成り立つ。

補題 3.4.6　変換系列の初期プログラム $P_\emptyset$は重

み完全である。

補題3.4.7　$P_0, P_1, \cdots, P_N$を変換系列とする。i=0,1,…,N-1に対し、無矛盾写像対

$T(P_0) \frown T(P_1) \frown T(P_2) \frown \cdots \frown T(P_i)$

が存在し、$P_0, P_1, \cdots, P_i$が重み完全であるならば、$T(P_i)$から$T(P_{i+1})$への無矛盾写像f_{i+1}が存在する。

（証明）TをアトムAと解代入σのP_iによる証明木、CをTの根で用いられた節で

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

の形をしたもの、$T_{A_1}, \cdots, T_{A_n}$をTの直接部分証明で、それぞれ$A_1, \cdots, A_n$の証明木であるとする。有礎順序$\succ$上の帰納法により、$f_{i+1}(T)$がAと解代入σのP_{i+1}による証明木T'であるような無矛盾写像f_{i+1}を定義する。P_iからP_{i+1}への変換で用いられた規則により、次の3つに場合分けできる。

(1) CがP_{i+1}中にある場合。
(2) Cがunfoldされた場合。
(3) Cがfoldされた場合。

紙数の制約により、(3)についてのみ証明を示す。

Dを

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

の形をしたfolding節、C'をfoldingの結果得られた節とする。補題3.4.1より、一般性を失う事なく、$A_1, \cdots, A_m$は$B_1, \cdots, B_m$の代入例であり、C'は

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

の形をしていると仮定できる。

まず、$T_{A_1}, \cdots, T_{A_m}$に$g_i, \cdots, g_1$を適用して、$A_1, \cdots, A_m$のP_0による証明木$S_{A_1}, \cdots, S_{A_m}$が唯一に得られる。$g_i, \cdots, g_1$は無矛盾写像であるから、$S_{A_1}, \cdots, S_{A_m}$は$T_{A_1}, \cdots, T_{A_m}$と同じ解代入を持つ。E_1を$S_{A_1}, \cdots, S_{A_m}, T_{A_{m+1}}, \cdots, T_{A_n}$のラベル集合と$\{A=A_0\}$の和集合とする。補題3.1.1より、$\sigma$は$E_1$のm.g.u.の、A中の変数への制約である。

次に、foldingの条件(a)より$B_1\theta = A_1, \cdots, B_m\theta = A_m$であるから、補題3.4.2より、$B_1, \cdots, B_m$の$P_0$による証明木で、根のラベルの左側を除いて$S_{A_1}, \cdots, S_{A_m}$と同じものが存在する。これらを$S_{B_1}, \cdots, S_{B_m}$とする。ここで、$E_2$を$S_{B_1}, \cdots, S_{B_m}, T_{A_{m+1}}, \cdots, T_{A_n}$のラベル集合と$\{A=A_0, B_0\theta = B_0\}$の和集合とする。このとき、$E_1$は$(E_2 - \{B_0\theta = B_0\})\theta$と同一である。$\theta$はA中の変数には代入を行わないので、補題3.1.2より、σはE_2のm.g.u.の、A中の変数への制約である。また、補題3.4.3より$w(S_{A_1}) = w(S_{B_1}), \cdots, w(S_{A_m}) = w(S_{B_m})$である。

次に、$B_0\theta$の述語は新述語であるから、$B_0\theta$のP_0による証明木の根で用いられる節はP_{new}中にある。また、foldingの条件(c)より、この節はDである。$S_{B_0\theta}$を$S_{B_1}, \cdots, S_{B_m}$の上に（"$B_0\theta = B_0$", D)でラベル付けされた根のノードを置いて得られる証明木とする。このとき、$S_{B_0\theta}$に$f_1, \cdots, f_i$を適用して、$B_0\theta$のP_iによる証明木$T_{B_0\theta}$が唯一に得られる。$f_1, \cdots, f_i$は無矛盾写像であるから、$T_{B_0\theta}$は$S_{B_0\theta}$と同じ解代入を持つ。ここで、E_3を$T_{B_0\theta}, T_{A_{m+1}}, \cdots, T_{A_n}$のラベル集合と$\{A=A_0\}$の和集合とする。補題3.1.1より、$\sigma$は$E_3$のm.g.u.の、A中の変数への制約である。

最後に、P_iは重み完全で、Cはfoldingの条件(d)を満たすから、

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

が成立する。また、

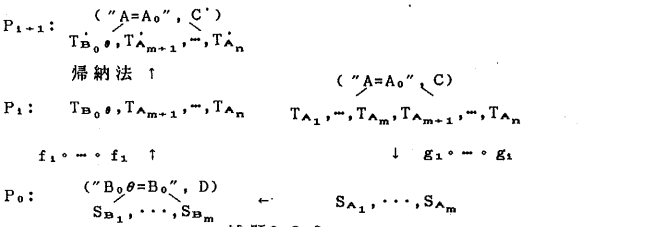
$B_0\theta$の述語は新述語であるから

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

が成立する。これらと$w(T_{B_0\theta}) = w(S_{B_0\theta})$及び$w(S_{B_1}) = w(T_{A_1}), \cdots, w(S_{B_m}) = w(T_{A_m})$より

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

が成立する。従って、$T \succ T_{B_0\theta}$及びj=m+1,…nに対して$T \succ T_{A_j}$が成立する。帰納的定義により、$f_{i+1}(T_{B_0\theta}), f_{i+1}(T_{A_{m+1}}), \cdots, f_{i+1}(T_{A_n})$は$B_0\theta, A_{m+1}, \cdots, A_n$の$P_{i+1}$による証明木$T'_{B_0\theta}, T'_{A_{m+1}}, \cdots, T'_{A_n}$であり、それぞれ$T_{B_0\theta}, T_{A_{m+1}}, \cdots, T_{A_n}$と同じ解代入を持つ。ここで、T'を$T_{B_0\theta}, T_{A_{m+1}}, \cdots, T_{A_n}$の上に（"A=$A_0$", C'）でラベル付けされた根のノードを置いて得られる証明木とする。また、E'をT'のラベル集合、即ち$T_{B_0\theta}, T_{A_{m+1}}, \cdots, T_{A_n}$,



補題3.2.3

図1　f_{i+1}の定義

…, T_{θ_n}のラベル集合と{A=Aθ}の和集合とする。このとき、補題3.1.1より、σはE'のm.g.u.の、A中の変数への制約である。従って、T'はAと解代入σのP_{i+1}による証明木である（図1参照）。よって、$f_{i+1}(T)=T'$と定義できる。

補題3.4.8　f_{i+1}を補題3.4.7で定義された$T(P_i)$から$T(P_{i+1})$への写像、g_{i+1}を補題3.3.4で定義された$T(P_{i+1})$から$T(P_i)$への写像とする。このとき、(f_{i+1}, g_{i+1})は$T(P_i)$と$T(P_{i+1})$の間の無矛盾写像対であり、P_{i+1}は(f_1, g_1), (f_2, g_2), …, (f_{i+1}, g_{i+1})に関して重み完全である。

定理3.4.9（成功多重集合の保存）

初期プログラムP_θから始まる変換系列中の全てのプログラムの成功多重集合は、P_θの成功多重集合と等しい。

系3.4.10（最小エルブラン・モデルの保存）

初期プログラムP_θから始まる変換系列中の全てのプログラムの最小エルブラン・モデルは、P_θの最小エルブラン・モデルと等しい。

4. 考察

成功多重集合を保存する変換規則は、最小エルブラン・モデルのみを保存する変換規則より広い範囲で使用できることが保証される（cf. [2]）。例として、DEC-10 Prologのsetof述語とbagof述語を考える。setof(X,P,S)は「SはPが成功するようなXの代入例の集合である」ことを表し、bagof(X,P,S)は「SはPが成功するようなXの代入例の多重集合である」ことを表す。最小エルブラン・モデルの意味で等価なプログラムはsetof、bagofコールに対して必ずしも同じ振舞いをするとは限らない。例として、1節であげたプログラムP_1、P_2、P_3を考える。これらのプログラムは最小エルブラン・モデルの意味で等価であるが、ゴール

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

に対して、P_2とP_3は解代入<X⇐a,Y⇐[a]>を返すが、P_1は失敗する。さらに、ゴール

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

に対して、P_2は解代入<X⇐a,Y⇐[a]>を、P_3は解代入<X⇐a,Y⇐[a,a]>を返し、P_1は失敗する。しかし、成功多重集合が等しいプログラムは、停止する場合にはsetof及びbagofコールに対して同様に振舞う。（P_1、P_2、P_3の成功多重集合は等しくない。）このことより、玉木－佐藤の変換によって得られたプログラムの述語は、setof及びbagofの引数として用いても差し支えないことが保証される。

5. おわりに

本稿では、玉木－佐藤のPrologプログラムのUnfold/Fold変換が最小エルブラン・モデルよりも強い意味での等価性を保存することを示した。初期プログラムに玉木－佐藤の変換を適用して得られたすべてのプログラムは、トップレベルのゴールに対して初期プログラムと同じ解代入を同じ回数だけ返すことが証明された。

謝辞

本研究は第五世代コンピュータ計画の委託で行われた。研究の機会と激励を頂いた淵一博ICOT研究所長、古川康一同次長、長谷川隆三第一研究室長、伊藤英則同第三研究室長に感謝致します。

参考文献

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

[2] Kawamura,T and T.Kanamori,"Preservation of Stronger Equivalence in Unfold/Fold Logic Program Transformation", Proc. of International Conference on Fifth Generation Computer Systems 1988, pp.413-421,Tokyo, 1988.

[3] Kanamori,T and T.Kawamura,"Preservation of Stronger Equivalence in Unfold/Fold Logic Program Transformation(II)", ICOT Technocal Report TR-403, 1988.

[4] Maher,M.J., "Equivalences in Logic Programs", Proc. of 3rd International Conference on Logic Programming, London, 1986.

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

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

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

[8] 玉木久夫、「論理型言語におけるプログラム変換」、淵一博 監修、古川康一・溝口文雄 共編「プログラム変換」第3章、pp.39-62、共立出版、1987.