知識工学と人工知能 66 － 5
（１９８９．９．２２）

層状プログラムにおけるUnfold/Fold変換

沈 涵　　中川 裕志
横浜国立大学

玉木－佐藤によって定義された確定節プログラムにおけるunfold／fold変換はプログラム合成など多くの分野で応用されている．確定節プログラムより一般的な否定を許す層状プログラムが整備された意味論を持つので，最近，論理プログラミングと非単調論理の分野で注目を浴びている．この論文では，層状プログラムにおけるunfold／fold変換を提案する．まず反復不動点意味論を導入し，そして，証明木や成功集合などの手続き的な定義を行って，両方が一致するような層状プログラムクラスを見つける．続いて，このような層状プログラムクラスに対して，unfold／fold変換規則を定義する．最後に，定義された変換について，反復不動点意味論で等価性が保存されることを証明する．

UNFOLD/FOLD TRANSFORMATION OF STRATIFIED PROGRAMS

Han SHEN　　Hiroshi NAKAGAWA

Yokohama National University

156, Tokiwadai, Hodogaya-Ku, Yokohama 240, Japan

Unfold/fold transformation rules of definite programs, proposed by Tamaki-Sato have been effectively used in logic programming. Recently, it is realized that stratified programs have satisfactory semantics, so much attention has been paid to such programs in logic programming and nonmonotonic logic.

In this paper, We firstly introduce the iterated fixed points of declarative semantics and define our procedural semantics for stratified programs, then find out the stratified program classes which the two semantics coincides. Nextly, we propose unfold/fold transformation rules for such stratified programs and prove that the transformation preserves equivalence in a sense of iterated fixed point semantics.

1．はじめに

玉木－佐藤によって，定義された確定節プログラムにおけるunfold／fold変換[1]はプログラム合成など多くの分野で応用されている．プログラム変換を行うとき，等価性を保存しなければならないが，等価性はプログラムの意味に基づいて定義される[2]．玉木－佐藤は確定節プログラムにおけるunfold／fold変換について，最小モデル意味で等価性が保存されることを証明した．金森－川村は玉木－佐藤が定義した確定節プログラムに対してunfold／fold変換を行って得られたプログラムが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した[3][4]．

確定節プログラムより一般的な否定を許す層状プログラムが整備された意味論を持つので，最近，論理プログラミングと非単調論理の分野で注目を浴びている．この論文では，まず，反復不動点意味論を導入し，証明木や成功集合など手続き的な定義を行って，両方が一致するような層状プログラムクラスを見つける．そして，このような層状プログラムクラスに対して，unfold／fold変換規則を定義する．最後に，定義された変換について，反復不動点意味論で等価性が保存されることを証明する．

2．層状プログラムの意味論

2．1．基本定義

まず，幾つかの基本的な定義を与える．

【定義2.1.1】

Aを原子式とし，θを代入とし，$L_1, \ldots L_m$をリテラルとする：

$A \leftarrow L_1, L_2, \ldots, L_m$　を節Cと呼ぶ．$L_1, \ldots, L_m$を節Cの本体と呼び，Aを頭部述語と呼ぶ．

本体にのみ現れる変数を内部変数と呼ぶ．

本体に現れるリテラルが全て正リテラルである節を確定節と呼ぶ．確定節の有限集合を確定節プログラムと呼ぶ．

$A\theta \leftarrow L_1\theta, \ldots, L_m\theta$　を節Cのインスタンス節と呼び，$C\theta$と記する．$A\theta, L_1\theta, \ldots, L_m\theta$が全て基礎原子式である場合，$C\theta$を節Cの基底インスタンス節と呼ぶ．

プログラムPに現れる定数と関数から構成できる全ての基礎項の集合をエルブラン空間U_pと呼ぶ．

プログラムPの述語の引数へエルブラン空間の要素を代入して生成される全ての基礎原子式の集合をエルブラン領域B_pと呼ぶ．

【定義2.1.2 層状プログラム】

プログラムPが$P_1; P_2; \ldots; P_m$に分けられるとき，Pを層状プログラムと呼ぶ．ここで，P_1が確定節の集合で、レベル i（$2 \leq i \leq m$）で定義された述語$A : A \leftarrow L_1, \ldots, L_n$において，体部の正リテラル$L_k$（$1 \leq k \leq n$）が全部i以下のレベルで定義され，負リテラル$L_k = \neg q_k$の$q_k$（$1 \leq k \leq n$）が全部i未満のレベルで定義される．また，iを述語Aのレベルと呼び，level(A)と記する（level(A)＝i）．上のレベルの述語が下のレベルの述語によって定義される．レベルは層とも呼ばれる．

2．2．層状プログラムにおける反復不動点意味論

本節では，まず層状プログラムPに対する写像T_pと順序数べき（ordinal powers）を定義し，そのうえ，層状プログラムにおける反復不動点の定義を行う[5]．

【定義2.2.1　写像T_p】

Iをプログラムの解釈とする．

$T_p(I) = \{A : P$に基底インスタンス節$A_1\theta \leftarrow L_1\theta, \ldots, L_n\theta$　が存在して，$A_1\theta = A$かつ正リテラルL_kに対して　$L_k\theta \in I$　$(1 \leq k \leq n)$，負リテラル$L_k = \neg B_k$に対して　$B_k\theta \notin I$　$(1 \leq k \leq n)$が成り立つ$\}$．

【定義2.2.2 T_pの順序数べき】

Iをプログラムの解釈とする．

$$T_p \uparrow 0(I) = I$$

$$T_p \uparrow (n+1)(I) = T_p(T_p \uparrow n(I)) \cup T_p \uparrow n(I)$$

$$\vdots$$

$$T_p \uparrow \omega(I) = \bigcup_{n=0}^{\infty} T_p \uparrow n(I)$$

次に層状プログラムPにおいて、M_pを定義する：

【定義2.2.3　M_p】

ϕを空集合、Pを$P_1; P_2; \ldots; P_m$によって層状化されたプログラムとする．

$M_1 = T_{P_1} \uparrow \omega(\phi)$；$M_2 = T_{P_2} \uparrow \omega(M_1)$；…；$M_m = T_{P_m} \uparrow \omega(M_{m-1})$．$M_p = {}_{df} M_m$．

文献[5]により，$T_p(M_p) = M_p$，しかも，M_pがPの極小モデルである．M_pは層状プログラムPの反復不動点と呼ばれる．直観的に反復不動点M_pは順次にモデル$M_{P_{i-1}}$に対して，P_iに属するプログラム節を使って，モデルM_{P_i}を生成することによって得られる．

我々はこのように定義された反復不動点意味論で，層状プログラムにおけるunfold／fold変換が等価性を保存することを証明する（即ち$M_P = M_Q$のとき，層状プログラムPとQが等価であると見なす）．

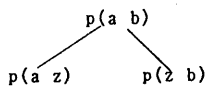
2．3．層状プログラムにおける反復不動点の計算

この節では，層状プログラムの反復不動点を計算するために使われる手続き的な概念 ― 証明木，成功集合を定義し，反復不動点が計算できる（即ち，反復不動点と成功集合が一致する）ような層状プログラムクラスを見つける．

【定義2.3.1 Pによる証明木】
Fを基礎原子式とし，Pを層状プログラムとする．FのPによる証明木が次のように定義される：
(1)証明木の根の節点にラベルFが付けられる；
(2)根から始まった各枝に同じ原子式が一回しか現れない；
(3)木の中の全ての節点vが次のようになる：
プログラムP中の節の基底インスタンス節
$A_1\theta \leftarrow L_1\theta,\ldots,L_u\theta,\neg L_{u+1}\theta,\ldots,\neg L_n\theta$ が存在して，$A = A_1\theta$ しかも$L_1\theta,\ldots,L_u\theta$がそれぞれ証明木$T_1,\ldots,T_u$を持ち，$L_{u+1}\theta,\ldots,L_n\theta$が証明木を持たない．
節点vにラベルAが付けられる．節点vの子節点のラベルがそれぞれ$L_1\theta,\ldots,L_u\theta$，"true",...,"true"である（n＝0のとき，節点vが証明木のleafとなる）．

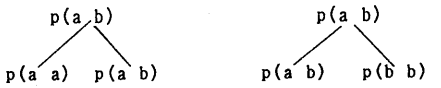
この定義により，任意の基礎原子式のPによる証明木の各節点が基礎原子式あるいは"true"によってラベル付けられる．また，関数記号を含まない層状プログラムの場合，定義2.3.1の条件(2)が無限ループを防ぐことができる．したがって，任意の基礎原子式Aに対して，有限時間内でAの証明木が存在するか否かが分かる．

【例1】プログラムP：
q(a b)
p(x y)←p(x z),p(z y)
において，p(a b)の証明木を構成してみる：



＜図1a＞

Pのエルブラン空間が$U_0 = \{a, b\}$で，その要素を変数zに代入して，次のような二つの証明木の生成を試みる：



＜図1b＞

二つの木両方ともに基礎原子式p(a b)が二回現れる枝が存在して，定義2.3.1の条件(2)を満たさないので，証明木になれない．即ち，p(a b)が証明木を持たない．

次に，証明木に基づいて，層状プログラムの成功集合を定義する：

【定義2.3.2 成功集合succ(P)】
Pを層状プログラムとする．Pによる証明木を持つ全ての基礎原子式の集合を成功集合と呼び，succ(P)と記する．

層状プログラムの場合，確定節プログラムと違って，節の本体に否定述語¬Aが現れるので，Aの証明木が存在しないことも有限時間内で証明しなければならない．次に，どのような条件で，2.2節で定義された反復不動点M_Pが計算できるか（即ち，M_Pと成功集合succ(P)が等しい）について検討する．

【補助定理1】
確定節プログラムの成功集合succ(P)はその反復不動点に等しい．
証明：プログラムPが確定節プログラムであるので，
B ∈ success(P)　(success(P)はSLD反駁によって定義されたPの成功集合である)；
iff　PによるBのSLD反駁が存在する；
iff　定義2.3.1によって定義されたBの証明木が存在する（確定節プログラムの場合，SLD反駁からBの証明木を簡単につくれる．逆に，Bの証明木からBのSLD反駁も簡単に書ける）；
iff　B ∈ succ(P)　が成り立つ．
また，確定節プログラムの場合，文献[6]より最小モデルとsuccess(P)が一致し，反復不動点が最小モデルなので，Pの反復不動点がsucc(P)に等しい．■

【補助定理2】
Pを関数なしの確定節プログラムとし，M_PをPの反復不動点（最小モデルでもある）とし，succ(P)を証明木（定義2.3.1）に基づいて定義された成功集合とする．任意の基礎原子式Aに対して，$A \in M_P$ iff

$A \in succ(P)$が有限時間内で証明できる．

証明：

(a)まず$A \in succ(P)$が証明できることから$A \in M_p$を証明する：

$A \in succ(P)$が証明できるので，Aの証明木が存在しない．即ち，定義2.3.1の条件(2)あるいは条件(3)が満たされない．

case a1:条件(2)が満たされない場合：

節C：$A \leftarrow A, L$ が書ける（ここで，Lはリテラルの連言である）．即ち，$A \vee \neg A \vee \neg L$ が常に真である．$A \in M_p$ が成り立つ．

case a2:条件(3)が満たされない場合：

明らかに，対応するSLD木が有限失敗木で，$A \in M_p$ が成立する．故に，$A \in M_p$ が成立する．

(b)$A \in M_p$ から $A \in succ(P)$を証明する（即ち，任意の$A \in M_p$に対してAの証明木が存在しないことは有限時間内で証明できる）：

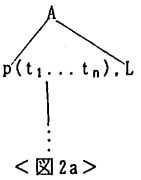
$A \in M_p$ とすると，文献[6]により，PによるAのSLD-木の各枝が有限失敗枝或は無限枝である：

case b1:PによるAのSLD-木の各枝が有限失敗枝である：

明らかに各SLD有限失敗枝と対応する証明木が存在しないから，$A \in succ(P)$ が証明できる．

case b2:PによるAのSLD-木が有限失敗枝と無限枝しか含まない：

無限枝と対応する証明木が存在しないことを示せばよい．任意の一つのSLD無限枝を次のように仮定する：



＜図2a＞

もし，項の組$(t_1 \ldots t_n)$に変数を含むなら，有限個のエルブラン空間の要素（関数を含まないプログラムなので）を変数に代入して（もし先に内部変数なしの論理プログラムへ変換すれば[7][8]効率的になる），SLD無限枝＜図2a＞と対応する証明木の生成を試る：

＜図2b＞

このようにして，エルブラン空間が有限であるから，各木に必ず有限ステップでAが二回現れる枝が生成される．定義2.3.1の証明木の条件(2)が満たさない．故に，証明木になれない（例1を参考されたい）．即ち，任意のSLD無限枝と対応する証明木が存在しないことを有限時間内で証明できる．

よって，$A \in succ(P)$ が導かれる．

(a)(b)をまとめて，$A \in M_p$ iff $A \in succ(P)$ が有限時間内で証明できる．■

【定理1】

Pを関数なしの層状プログラムとする．Pの反復不動点M_pは成功集合$succ(P)$と等しい．

証明：

プログラムPが$P_1; \ldots; P_n$によって層状化されたとする．$Q_i =_{df} P_1 \cup \ldots \cup P_i$.

任意の基礎原子式Aに対して，

$A \in M_p$ iff $A \in succ(P)$ と

$A \in M_p$ iff $A \in succ(P)$ を証明する（ここで，$A \in succ(P)$はPによるAの証明木が存在しないことを有限時間内で証明できるという意味である）．

述語のレベルkに対する帰納法で証明する：

(a)$k = 1$のとき（$Q_1 = P_1$）：

補助定理1より，$B \in M_{P_1}$ iff $B \in succ(P_1)$ が成立する．また，補助定理2により

$B \in M_{P_1}$ iff $B \in succ(P_1)$ が成り立つ．

(b)$k \leq i-1$のとき：

$B \in M_{Q_k}$ iff $B \in succ(Q_k)$ と

$B \in M_{Q_k}$ iff $B \in succ(Q_k)$ が成り立つと仮定する；

(c)$k = i$のとき，$A \in M_{Q_i}$ iff $A \in succ(Q_i)$ と

$A \in M_{Q_i}$ iff $A \in succ(Q_i)$ を証明する（ここで，$level(A) = i$ とする）．

(c1)まず $A \in M_{Q_i}$ iff $A \in succ(Q_i)$ を証明する：

定義によって，$A \in succ(Q_i)$ から $A \in M_{Q_i}$ へはほぼ自明である．

次に，$A \in M_{Q_i}$ から $A \in succ(Q_i)$ を証明する：

$A \in M_{Q_i}$ とする．

Q_i中の節Cの基底インスタンス節$C\theta$：

$A_1\theta \leftarrow B_1\theta, \ldots, B_v\theta, B_{v+1}\theta, \ldots, B_h\theta, \neg B_{h+1}\theta, \ldots, \neg B_m\theta$

が存在する．ここで，$A_1\theta = A$，$level(B_j) = i$ $(j = 1, \ldots v)$，$level(B_j) \leq i-1$ $(j = v+1, \ldots, h)$，$B_j\theta$ $(j = v+1, \ldots, h) \in M_{Q_{i-1}}$，

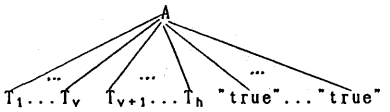
また，層状プログラムの定義によって，述語B_j $(j = h+1, \ldots, m)$ がレベル$i-1$以下で定義されているから，$B_j\theta \in M_{Q_{i-1}}$ $(j = h+1, \ldots, m)$ が成立する．

更に帰納法の仮定により，

$B_j\theta \notin succ(Q_{i-1})$ ($j = h+1, \ldots, m$) と
$B_j\theta \in succ(Q_{i-1})$ ($j = v+1, \ldots, h$) が成り立つ．
$B_j\theta$ ($j = v+1, \ldots, h$) のある証明木を $T_{v+1}, \ldots, T_h$ と仮定する．
節 $C\theta$ がレベル i の正述語だけで定義された確定節
$C'\theta$: $A_1\theta \leftarrow B_1\theta, \ldots, B_v\theta$
と見なせる（同じ理由で，$B_1\theta, \ldots, B_v\theta$ を頭部とする確定節と見なせる節が存在する）．補助定理1によれば節 $C'\theta$ と対応する証明木T'が存在する：

＜図3a 証明木T'＞

更に証明木T'を次のような証明木Tへ拡張する：



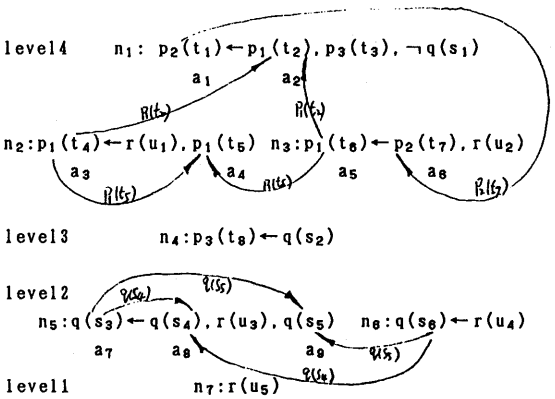
＜図3b 証明木T＞

故に，Aの証明木が存在する．即ち，$A \in M_{Q_i}$ から $A \in succ(Q_i)$ が導かれる．
したがって，$A \in M_{Q_i}$ iff $A \in succ(Q_i)$ が成立する．
(c2) $A \notin M_{Q_i}$ iff $A \notin succ(Q_i)$ を証明する：
以上と似たような方法で，レベルに関して帰納法で補助定理2を利用して証明できる．■

【定義2.3.3 CG(P)(Connection Graph)】
層状プログラムPに対して，次のような二次元組 (N(P) E(P))をPのconnection graphと呼び，CG(P)と記する．ここで，
N(P)：節点の集合で，一つの節点はPの一つの節と対応する；
E(P)：ラベルづけられた有向edgesの集合である．その要素 $<n_1\ n_2\ q(t)>$ はラベルq(t)でづけられた節点 n_1 から n_2 への有向edgeを表す．n_1 の節の頭部述語がq(s)の形で，n_2 の節の体部に現れる正リテラルがq(t)で，述語qのレベルは n_2 の節の頭部述語のレベルと同じである．

CG(P)はプログラムPの各層に現れる再帰関係を表現する．CG(P)の各ループに現れる節の頭部述語は再帰述語である．層状プログラムの定義によって同じ述語が同じレベルで定義され，同じレベルで定義される異なる述語がお互いに再帰的に定義される述語であると規定することができる．また，n_2 の体部にk重再帰述語 $q(t_1), \ldots, q(t_k)$ がk本のedges $<n_1\ n_2\ q(t_1)>, \ldots, <n_1\ n_2\ q(t_k)>$ と対応する（図4のlevel2を参照されたい）．

【例2】層状プログラムP：
n_1: $p_2(t_1) \leftarrow p_1(t_2), p_3(t_3), \neg q(s_1)$
n_2: $p_1(t_4) \leftarrow r(u_1), p_1(t_5)$
n_3: $p_1(t_6) \leftarrow p_2(t_7), r(u_2)$
n_4: $p_3(t_8) \leftarrow q(s_2)$
n_5: $q(s_3) \leftarrow q(s_4), r(u_3), q(s_5)$
n_6: $q(s_6) \leftarrow r(u_4)$
n_7: $r(u_5)$
（ここで，$t_1 \sim t_8, s_1 \sim s_6, u_1 \sim u_5$ は項である）
プログラムPのCG(P)は次のようになる：



＜図4＞

【定義2.3.4 属性値attri(p($t_1 \ldots t_n$))】
項 $t_1, \ldots t_n$ に現れる関数記号の数を $s_1, \ldots, s_n$ とする．$p(t_1 \ldots t_n)$ の属性値attri($p(t_1 \ldots t_n)$)を $(s_1 \ldots s_n)$ と設定する．

この定義では，項 $f^m(a_1 \ldots a_n)$ に現れる関数記号の数をmと見なす（ここで，$a_1, \ldots, a_n$ は定数である）．

【定義2.3.5 項下降性を持つ】
Pを内部変数なしの層状プログラムとする．節点nに到達できる各ループLにおいて（Lにv個の節点 $<n_i\ n_i\ q_i(t_{i1} \ldots t_{im})>$ ($i = 1, \ldots, v$) があると仮定する），n_i の節の頭部述語を $p_i(u_{i1} \ldots u_{im})$ とし，
attri($q_i(t_{i1} \ldots t_{im})$) = $(s_{i1} \ldots s_{im})$
attri($p_i(u_{i1} \ldots u_{im})$) = $(r_{i1} \ldots r_{im})$ とする．
(a)項 t_{ij}, u_{ij} ($i = 1, \ldots, v,\ j = 1, \ldots, m$) に関数記号が

現れない；
或は(b)

$$\sum_{i=1}^{v}(s_{ij}-r_{ij})<0 \quad (j=1,\ldots,m)$$

が成立すれば，節点nは項下降性をもつと呼ぶ．CG(P)の各ループに現れる全ての節点が項下降性を持てば，プログラムPは項下降性を持つと呼ぶ．

同じループの各節の頭部の異なる述語がお互いに再帰的に定義され，しかも，内部変数なしの節であることにより，一つのループ中の各節の頭部述語に現れる項の組の数は同じである（定義2.3.5において，述語$p_i(u_{i1}\ldots u_{im})$と$q_i(t_{i1}\ldots t_{im})$の項の組の数は同じmである）．

【例3】図4において（述語の下に属性値が書いてある），節点n_1に到達できるループは

L_1:{<n_1 n_3 $p_2(t_7)$><n_3 n_1 $p_1(t_2)$>}

L_2:{<n_1 n_3 $p_2(t_7)$><n_3 n_2 $p_1(t_5)$><n_2 n_1 $p_1(t_2)$>}

L_3:{<n_1 n_3 $p_2(t_7)$><n_3 n_2 $p_1(t_5)$><n_2 n_2 $p_1(t_5)$><n_2 n_1 $p_1(t_2)$>}

で，節点n_1に対して，もし

ループL_1において

attri($p_2(t_7)$)-attri($p_1(t_6)$)+attri($p_1(t_2)$)-attri($p_2(t_1)$)<0,

即ち，$a_6-a_5+a_2-a_1 < 0$；

ループL_2において

attri($p_2(t_7)$)-attri($p_1(t_6)$)+attri($p_1(t_5)$)-attri($p_1(t_4)$)+attri($p_1(t_2)$)-attri($p_2(t_1)$)<0,

即ち，$a_6-a_5+a_4-a_3+a_2-a_1 < 0$；

ループL_3において

attri($p_2(t_7)$)-attri($p_1(t_6)$)+attri($p_1(t_5)$)-attri($p_1(t_4)$)+attri($p_1(t_5)$)-attri($p_1(t_4)$)+attri($p_1(t_2)$)-attri($p_2(t_1)$)<0,

即ち，$a_6-a_5+a_4-a_3+a_4-a_3+a_2-a_1 < 0$

が満たされるなら，節点n_1は項下降性を持つ．
更に，節点n_2，n_3，n_5を以上と同じような方法で調べて，もし全て項下降性を持てば，プログラムPは項下降性を持つプログラムである．

次に，関数を含む場合，反復不動点が計算できる層状プログラムクラスを見つける：

【定理２】
項下降性を持つ層状プログラムPにおいて，Pの反復不動点M_pは成功集合succ(P)と同じである．
証明：
二ステップで証明する：
step1:
項下降性を持つ層状プログラムにおいて，任意の基礎原子式A($F^n(\alpha)$)が有限時間内で関数記号を含まない基礎原子式へ変換できる；
step2:
定理１の証明と似たような方法で，反復不動点M_pがsucc(P)と一致することが証明できる．■

3．層状プログラムにおけるUnfold／Fold変換

3．1．層状プログラムにおけるUnfold／Fold変換

本節では，反復不動点が計算できる層状プログラムに対して，unfold／fold変換を定義する．

【定義3.1.1 初期プログラム】
初期プログラムP^0は節の集合P_{new}とP_{old}に分けられ，しかも，反復不動点が計算できる層状プログラムである．P_{old}で定義される述語は旧述語と呼び，P_{new}で定義される述語は新述語と呼ぶ．新述語はP_{new}中の節の頭部にしか現れない．

文献[5]より層状プログラムの各層のわけかたが反復不動点に影響を与えないので，我々は同じレベルで定義される異なる述語がお互いに再帰的に定義される述語に限定できる．

【定義3.1.2 unfold変換】：
A，Bを原子式とし，L，L_1,...,L_nをリテラルの連言とし，P^iを反復不動点が計算できる層状プログラムとする．P^i中の節Cは A←B，L で，原子式Bと単一化可能なP^iの中の全ての節を B_1←L_1 B_2←L_2 ... B_n←L_n とし，それぞれの最汎単一化子をθ_1，θ_2，...，θ_nとする．節Cを $A\theta_1$←$L_1\theta_1$，$L\theta_1$， $A\theta_2$←$L_2\theta_2$，$L\theta_2$，...，$A\theta_n$←$L_n\theta_n$，$L\theta_n$ で置き換えたものを節Cの述語Bに対するunfold変換と定義する．

【定義3.1.3 fold変換】：
A，Bを原子式とし，K，K'，Lをリテラルの連言とし，，P^iを反復不動点が計算できる層状プログラムとする．P^i中の節CはA←K，L で、P_{new}に属する節Dは B←K' である．次のような条件を満たす代入θが存在するとき，節Cを A←Bθ，L で置き換えたものを節Dでの節Cに対するfold変換と定義する．
(1)level(B)≦level(A)；
(2)K'θ＝K；

(3) K'に現れてBに現れない変数を$x_1,\ldots x_m$とするとき、各x_i($1 \leq i \leq m$)はA，Lに現れない変数で，$x_i\theta$は$i \neq k$のとき異なる；
(4) $B\theta$と単一化可能な左辺を持つP_{new}中のルールはDのみである；
(5) Cの左辺の述語は旧述語であるかまたはCは初期プログラムP^0から少なくとも一回unfoldされた節である．

【補助定理３】
定義3.1.2，3.1.3によって定義された層状プログラムにおけるUnfold／fold変換は元のプログラムの層関係を保つ．
証明：
(a)Unfold変換（定義3.1.2）のとき：
層状プログラムの定義によって，
L_iの中の正リテラルL_{ik}において，
$level(L_{ik}) \leq level(B)(i=1,\ldots,n)$，かつ
$level(B) \leq level(A)$　が成立するので，
unfoldされた節の本体の正リテラルL_{ik}において，
$level(L_{ik}) \leq level(A)(i=1,\ldots,n)$が成り立つ．
L_iの中の負リテラルL_{ik}において，
$level(L_{ik}) < level(B)(i=1,\ldots,n)$，かつ
$level(B) \leq level(A)$　が成立するので，
unfoldされた節の本体の負リテラルL_{ik}において，
$level(L_{ik}) < level(A)(i=1,\ldots,n)$が成立する．
よって，unfold変換が元のプログラムの層関係を崩すことはない．
(b)Fold変換（定義3.1.3）のとき：
fold変換の条件(1)により，元のレベル関係を保つことができる．
したがって，定義された層状プログラムにおけるunfold／fold変換は元のプログラムの層関係を崩すことはない．■

【定義3.1.4 変換系列】
P^0を初期プログラム，P^{i+1}（$i=0,\ldots,n-1$）をP^iに対するunfold／fold変換を行って得られたプログラムとする．$P^0, P^1,\ldots, P^n$を変換系列と呼ぶ．

【補助定理４】
初期プログラムP^0に始まる変換系列中のプログラムP^iの反復不動点は計算できる．
証明：この補助定理は簡単に証明できる：
初期プログラムの定義によって，P^0の反復不動点が計算できる．また，定義3.1.2，定義3.1.3によって定義されたunfold／fold変換がその条件を崩さないことにより，帰納法でP^iの反復不動点が計算できることを証明できる．■

３．２．層状プログラムにおけるUnfold／Fold変換の等価性の証明

本節では，反復不動点意味論で，反復不動点が計算できる層状プログラムに対して定義されたunfold／fold変換が等価性を保つことを証明する．２節での定理１，定理２により，関数を含まない層状プログラムあるいは項下降性を持つ層状プログラムでは，その反復不動点と成功集合が一致するので，次の証明では，定義2.3.1の証明木を用いて証明を行う．

まず，証明に使われる幾つかの定義を与える：

【定義3.2.1 重み】
$A, A_1,\ldots, A_n$を原子式とし，Tを定義2.3.1によって定義されるAの証明木のうち節点数が一番少ない証明木とし，Tの節点数をsとする．
Aの重みを

$$weight(A) = \begin{cases} s-1 & : A\text{が新述語；} \\ s & : A\text{が旧述語．} \end{cases}$$

と定義する．
述語の連言（$A_1,\ldots,A_n$）の重みを
$weight(A_1,\ldots,A_n) = weight(A_1)+\ldots+weight(A_n)$
と定義する．

【定義3.2.2 下降インスタンス節】
Aを基礎原子式，P^iをP^0に始まる変換系列中のプログラムとする．P^i中の節Cの基底インスタンス節
$C\theta : A_1\theta \leftarrow L_1\theta,\ldots, L_u\theta, \neg L_{u+1}\theta,\ldots, \neg L_m\theta$
において，次のような条件を満たすとき，$C\theta$をAの下降インスタンス節と定義する：
(a) $A_1\theta = A$，しかも$L_1\theta,\ldots, L_u\theta$が$P^0$で証明できる．$L_{u+1}\theta,\ldots, L_m\theta$が$P^0$で証明できない；
(b) $weight(A) \geq weight(L_1,\ldots,L_u)+(m-u)$；
(c)節Cがfold変換の条件(5)を満たすならば，
$weight(A) > weight(L_1,\ldots,L_u)+(m-u)$．

【定義3.2.3 重み完全】
Aを基礎原子式とし，P^iをP^0に始まる変換系列中のプログラムとする．任意の$A = M_{P^0}$に対して，P^i中の節の基底インスタンス節で，Aの下降インスタンス節 $A \leftarrow L$ が存在すれば，P^iを重み完全と呼ぶ．

【補助定理５】

$P^0,\ldots,P^n$を変換系列とする．もし $M_{Pi}=M_{P^0}$ であれば，$M_{P^{i+1}} \subseteq M_{P^i}$ $(i=0,\ldots,n-1)$ が成り立つ．

証明：

任意の基礎原子式 $A \in M_{P^{i+1}}$ とする．定理1，定理2，補助定理4により，P^{i+1}によるAの証明木Tが存在する．P^iによるAの証明木も存在することを示せばよい．証明木Tの根で使われたP^{i+1}中の節Cの基底インスタンス節を$C\alpha$とする．

case1: CはP^{i}中に存在する．

この場合，明かにTがP^{i}によるAの証明木と見なせる．$A \in M_{P^i}$ が成立する．

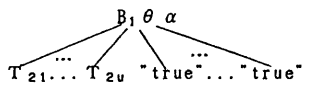
case2: CはP^{i}中の節C'の原子式Bをunfoldして得られた節：

C' : $A_1 \leftarrow B, L_{11},\ldots,L_{1k},\neg L_{1k+1},\ldots,\neg L_{1m}$

C" : $B_1 \leftarrow L_{21},\ldots,L_{2u},\neg L_{2u+1},\ldots,\neg L_{2v}$

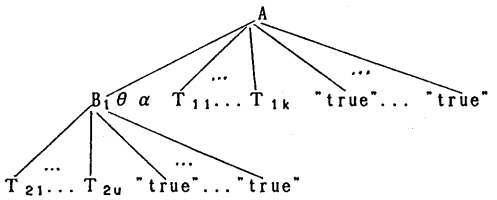
C : $A_1\theta \leftarrow L_{21}\theta,\ldots,L_{2u}\theta,\neg L_{2u+1}\theta,\ldots,\neg L_{2v}\theta, L_{11}\theta,\ldots,L_{1k}\theta,\neg L_{1k+1}\theta,\ldots,\neg L_{1m}\theta$ （ここで，θはBとB_1のmguである）

AがP^{i+1}中の節Cの基底インスタンス節$C\alpha$を根とする証明木Tによって証明されたので，$A_1\theta\alpha=A$ かつ P^{i}による$L_{21}\theta\alpha,\ldots,L_{2u}\theta\alpha,L_{11}\theta\alpha,\ldots,L_{1k}\theta\alpha$の証明木$T_{21},\ldots,T_{2u},T_{11},\ldots,T_{1k}$が存在し，$L_{2u+1}\theta\alpha,\ldots,L_{2v}\theta\alpha,L_{1k+1}\theta\alpha,\ldots,L_{1m}\theta\alpha$の証明木が存在しない．したがって，P^{i}中の節C"によって$B_1\theta\alpha$の証明木T_1が構成できる：



＜図5a　証明木T_1＞

また，P^{i}中の節C'によって，A（$=A_1\theta\alpha$）の証明木T'が構成できる：



＜図5b　証明木T'＞

即ち，AのP^{i}による証明木が存在する．故に，$A \in M_{Pi}$ が成立する（証明木T'には元の層関係が保たれる）．

case3: CはP^{i}中の節C'を節Dでfoldして得られた節：

C' : $A_1 \leftarrow K, L_{11},\ldots,L_{1k},\neg L_{1k+1},\ldots,\neg L_{1m}$

D : $B \leftarrow k_{21},\ldots,k_{2u},\neg k_{2u+1},\ldots,\neg k_{2v}$

C : $A_1 \leftarrow B\theta, L_{11},\ldots,L_{1k},\neg L_{1k+1},\ldots,\neg L_{1m}$

（ここで，$K=(k_{21},\ldots,k_{2u},\neg k_{2u+1},\ldots,\neg k_{2v})\theta$）

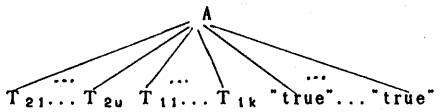
AがP^{i+1}中の節Cの基底インスタンス節$C\alpha$を根とする証明木Tによって証明されたので，$A_1\alpha=A$ かつ P^{i}による$B\theta\alpha,L_{11}\alpha,\ldots,L_{1k}\alpha$の証明木$T_1,T_{11},\ldots,T_{1k}$が存在し，$L_{1k+1}\alpha,\ldots,L_{1m}\alpha$の証明木が存在しない．定理の仮定 $M_{Pi}=M_{P^0}$ により，$B\theta\alpha$のP^{0}による証明木も存在する．また，fold変換の条件(4)より，$B\theta$と単一化可能な左辺を持つP_{new}中の節がDのみであるので，P^{0}による$k_{21}\theta\alpha,\ldots,k_{2u}\theta\alpha$の証明木が存在し，$k_{2u+1}\theta\alpha,\ldots,k_{2v}\theta\alpha$の証明木が存在しない．定理の仮定 $M_{Pi}=M_{P^0}$ によって，P^{i}による$k_{21}\theta\alpha,\ldots,k_{2u}\theta\alpha$の証明木$T_{21},\ldots,T_{2u}$が存在し，$k_{2u+1}\theta\alpha,\ldots,k_{2v}\theta\alpha$の証明木が存在しない．

更に，fold変換の条件(2)より，

$K\alpha=(k_{21}\theta,\ldots,k_{2u}\theta,\neg k_{2u+1}\theta,\ldots,\neg k_{2v}\theta)\alpha$

が成立する．

節C'を使って，P^{i}によるA（$=A_1\alpha$）の証明木T'を構成することができる（fold変換の条件(1)によって，元の層関係が保たれる）．



＜図5c 証明木T'＞ ■

【補助定理6】

P^{i}をP^{0}に始まる変換系列中のプログラムとする．P^{i}中の節Cにおいて，Cがfold変換の条件(5)を満たさなければ，Cの頭部述語が新述語で，本体の述語が全て旧述語である．

証明：fold変換の条件(5)を満たさなければ，明らかに，Cの頭部述語が新述語である．

もし，Cの本体に新述語Bが存在するならば，必ずP^j（$0 \leq j < i$）に節C'が存在して，C'の本体の部分述語の連言が新述語Bを頭部述語とする節によってfoldされる．このとき，節C'がfold変換の条件(5)を満たすので，節Cもfold変換の条件(5)を満たすことになる．これは定理の仮定と矛盾する．

故に，節Cの本体の述語が全て旧述語である．■

【補助定理7】

P^{0}に始まる変換系列中のプログラムP^{i}が重み完全

であるならば，$M_{P^0} \subseteq M_{P^i}$ が成り立つ．

証明：

A, L_k (k=1,...,m) を M_{P^0} に属する基礎原子式とする．P^i中の節の基底インスタンス節

$A \leftarrow L_1,\ldots,L_u,\neg L_{u+1},\ldots,\neg L_m$ に対して，次のような条件を満たすwell-founded ordering $L_k \prec A$ を定義する：

(a) level(A) > level(L_k)

或は(b) level(A) = level(L_k) かつ

weight(A) > weight(L_k) + (m-u).

或は(c) weight(A) = weight(L_k) + (m-u) かつ

Aが新述語で，L_kが旧述語である．

任意の基礎原子式 $A \in M_{P^0}$ に対してP^iが重み完全であるから，P^i中の節Cの基底インスタンス節で，Aの下降インスタンス節C'：

$A \leftarrow L_1,\ldots,L_u,\neg L_{u+1},\ldots,\neg L_m$ が存在する．

層状プログラムの定義より，

level(A) > level(L_k) (k=u+1,...,m) が成り立つ．

故に，$L_k \prec A$ (k=u+1,...m) が成立する．

weight(A) > weight(L_k) + (m-u)のとき，$L_k \prec A$ ($1 \leq k \leq u$) が成立する．

weight(A) = weight(L_k) + (m-u)のとき，下降インスタンス節の条件(c)によって，節Cがfold変換の条件(5)を満たさない．補助定理6より，Aが必ず新述語で，L_kの述語は全部旧述語である．

故に，$L_k \prec A$ ($1 \leq k \leq u$) も成立する．

よって，$L_k \prec A$ (k=1,...,u) が成り立つ．

したがって，C'の本体の任意のL_kにおいて，

$L_k \prec A$ (k=1,..,m) が成り立つ．

更に，$L_k \in M_{P^0}$ (k=1,...,m) に対して，帰納法で順序$\prec$にたどって，P^iによる下降インスタンスを利用して，L_kはP^iで証明できる．故に，AがP^iによって証明できる（P^iによる証明木が存在する）．

即ち，$A \in M_{P^i}$ が成り立つ．■

【補助定理8】

初期プログラムP^0は重み完全である．

証明： 任意の $A \in M_{P^0}$ に対して，AのP^0による証明木のうち，節点数が一番少ない証明木をTとしよう．

P^i中の節Cの基底インスタンス節 $C\theta$：

$A_1\theta \leftarrow L_1\theta,\ldots,L_u\theta,\neg L_{u+1}\theta,\ldots,\neg L_m\theta$

が存在する（$A_1\theta = A$．$L = {}_{df}L_1,\ldots,L_u,\neg L_{u+1},\ldots,\neg L_m$）．

weight(A) ≧ weight($L\theta$) が明かに成り立つ．

節Cがfold変換の条件(5)を満たすのは，Cの左辺の述語が旧述語であるときのみなので，

weight(A) > weight($L\theta$) が成り立つ．

故に，Aの基底インスタンス節$C\theta$は下降インスタンス節でもある．P^0が重み完全である．■

【補助定理9】

変換系列の中のプログラムP^iが重み完全であれば，次のプログラムP^{i+1}も重み完全である．

証明：

$A \in M_{P^0}$ とする．定理の仮定よりP^i中の節Cの基底インスタンス節で，Aの下降インスタンス節$C\alpha$：

$A_1\alpha \leftarrow L_1\alpha$ が存在する（$A_1\alpha = A$）．P^{i+1}にもAの下降インスタンス節が存在するのを示せばよい．

case1：CがP^iから引き継がれた：

この場合，P^{i+1}にAの下降インスタンス節$C\alpha$：

$A_1\alpha \leftarrow L_1\alpha$ が存在する．

case2：節Cの本体の述語Bがunfoldされ，節C_1となる：

$C : A_1 \leftarrow B, L_{11},\ldots,L_{1u},\neg L_{1u+1},\ldots,\neg L_{1n}$

$D : B_1 \leftarrow L_{21},\ldots L_{2v},\neg L_{2v+1},\ldots,\neg L_{2r}$

$C_1 : A_1\theta \leftarrow L_{21}\theta,\ldots L_{2v}\theta,\neg L_{2v+1}\theta,\ldots,$
$\neg L_{2r}\theta, L_{11}\theta,\ldots,L_{1u}\theta,\neg L_{1u+1}\theta,\ldots,$
$\neg L_{1n}\theta$ （ここで，θはBとB_1のmguである）

$C_1\alpha' : A_1\alpha' \leftarrow L_{21}\alpha',\ldots L_{2v}\alpha',\neg L_{2v+1}\alpha',$
$\ldots,\neg L_{2r}\alpha', L_{11}\alpha',\ldots,L_{1u}\alpha',$
$\neg L_{1u+1}\alpha',\ldots,\neg L_{1n}\alpha'$
（$\alpha' = \theta\alpha$ かつ $A_1\alpha' = A_1\alpha = A$）

ここで，節DはBと単一化可能な節のうち，$D\alpha'$が$B\alpha'$の下降インスタンス節となるような節である．

節Cの本体をL_1，節Dの本体をL_2，節$C_1\alpha'$の本体をL'と記する．

$C\alpha' : A_1\alpha' \leftarrow L_1\alpha'$ が$C\alpha$のインスタンス節なので，$C\alpha'$はAの下降インスタンス節である．故に，$B\alpha', L_{11}\alpha',\ldots,L_{1u}\alpha'$が$P^0$で証明でき，$L_{1u+1}\alpha',\ldots,L_{1n}\alpha'$が$P^0$で証明できない．また，$D\alpha'$が$B_1\alpha'$の下降インスタンス節であることより，$L_{21}\alpha',\ldots L_{2v}\alpha'$が$P^0$で証明でき，$L_{2v+1}\alpha',\ldots,L_{2r}\alpha'$が$P^0$で証明できない．

また，下降インスタンス節の条件により

weight(A) ≧ weight($B\alpha'$)+weight($L_{11}\alpha'$)+...+
weight($L_{1u}\alpha'$)+(n-u)

weight($B_1\alpha'$) ≧ weight($L_2\alpha'$) が成立する．

故に，weight(A) ≧ weight(L') が成り立つ．

更に，Bが旧述語のとき，節Dがfold変換の条件(5)を満たすので，weight($B_1\alpha'$) > weight($L_2\alpha'$) が成立し，weight(A) > weight(L') も成り立つ．Bが新述語のとき，補助定理6より，節Cがfold変換の条件(5)を満たすので，

weight(A) > weight($L_1\alpha'$) が成り立つ．

故に，weight(A)＞weight(L')　も成り立つ．
よって，weight(A)＞weight(L')　が成立する．
故に，$C_1\alpha$'がAの下降インスタンス節である．

case3: Cが節Dでfoldされ，節C'となる：

C：$A_1 \leftarrow k_{11},\ldots,k_{1f},\neg k_{1f+1},\ldots,\neg k_{1v},L_{11},\ldots,L_{1u},\neg L_{1u+1},\ldots,\neg L_{1n}$

D：$B \leftarrow k_{11}',\ldots,k_{1f}',\neg k_{1f+1}',\ldots,\neg k_{1v}'$

C'：$A_1 \leftarrow B\theta,L_{11},\ldots,L_{1u},\neg L_{1u+1},\ldots,\neg L_{1n}$
（ここで，$k_{11}'\theta,\ldots,k_{1f}'\theta,\neg k_{1f+1}'\theta,\ldots,\neg k_{1v}'\theta = k_{11},\ldots,k_{1f},\neg k_{1f+1},\ldots,\neg k_{1v}$）

節Cの本体をL_1，節Dの本体をK'，C'の本体をL'と記する．

$C\alpha$がAの下降インスタンス節なので，$k_{11}\alpha,\ldots,k_{1f}\alpha,L_{11}\alpha,\ldots,L_{1u}\alpha$が$P^0$で証明でき，

$k_{1f+1}\alpha,\ldots,k_{1v}\alpha,L_{1u+1}\alpha,\ldots,L_{1n}\alpha$が$P^0$で証明できない．故に，$k_{11}'\theta\alpha,\ldots,k_{1f}'\theta\alpha$が$P^0$で証明でき，$k_{1f+1}'\theta\alpha,\ldots,k_{1v}'\theta\alpha$が$P^0$で証明できない．節Dを使って，$B\theta\alpha$も$P^0$で証明できる．

また，fold変換の条件(4)より，$B\theta$と単一化可能な左辺を持つ節はDのみで，しかもBが新述語であることから，weight($B\theta\alpha$)≦weight($K'\theta\alpha$)が成立する．したがって，

weight(A)≧weight($L_1\alpha$)≧weight($L'\alpha$)　が成立する．

更に，fold変換の条件(5)を満たすとき，

weight(A)＞weight($L_1\alpha$)≧weight($L'\alpha$) を満足するので，$A \leftarrow L'\alpha$　がAの下降インスタンス節である．

以上をまとめて，P^{i+1}も重み完全である．■

補助定理5，7，8，9により次のような重要な定理が得られる：

【定理3】
Pを反復不動点が計算できる層状プログラムとする．P^0に始まるunfold／fold変換系列中の任意のP^iについて，　$M_{P^i}=M_{P^0}$ が成立する．

4．終わりに：

この論文では，まず，反復不動点が計算できる層状プログラムクラスを見つけた．そして，このような層状プログラムクラスに対して，unfold／fold変換を定義した．最後に，定義された変換が反復不動点意味論で等価性を保存することを証明した．層状プログラムにおいて，反復不動点と極小限定[9]や完全モデル[10][11]の一致性を利用して，unfold／fold変換のいろいろな面での応用が期待できる．

参考文献

1. Tamaki, H. and Sato, T.: "Unfold/fold Transformation of Logic Programs", 2nd International Logic Programming Conference, Uppsala, 1984.
2. M.J.Maher: "Equivalences of Logic Programs", Foundations of Deductive Databases and Logic Programming (J.Minker, Ed.), Morgan Kaufman Publishers, Los Altos, CA, 1988.
3. Kawamura, T. and Kanamori, T.: "Preservation of Stronger Equivalence in Unfold/Fold Logic Program Transformation", FGCS 1988, Tokyo, 1988.
4. 金森，川村：" 論理プログラムのUnfold／Fold変換におけるより強い等価性の保存(II)"，情報処理学会研究報告89－SF－28.
5. Apt, K.R., Blair, H., and Walker, A.: "Towards a Theory of Declarative Knowledge", Foundations of Deductive Databases and Logic Programming (J. Minker, Ed.), Morgan Kaufman Publishers, Los Altos, CA., 1988.
6. Lloyd, J.W.: "Foundations of Logic Programming", Springer, Berlin, 1984.
7. 沈，中川："内部変数なしの論理プログラムへの変換(I)"，日本ソフトウェア科学会第五回大会論文集，1989.9.
8. 沈，中川："内部変数なしの論理プログラムへの変換"，人工知能学会誌，Vol.4, No.4.
9. Vladimir Lifschitz: "On the Declarative Semantics of Logic Programs with Negation", Foundations of Deductive Databases and Logic Programming (J. Minker, Ed.), Morgan Kaufman Publishers, Los Altos, CA, 1988.
10. Przymusinski, C.: "On the Relationship Between Logic Programming and Non-monotonic Reasoning", AAAI-88, 1988.
11. Przymusinski, C.: "On the Declarative Semantics of Deductive Databases and Logic Programs", (J. Minker, Ed.), Morgan Kaufman Publishers, Los Altos, CA, 1988.