知識工学と人工知能 63－1
（１９８９．３．１３）

一般プログラムにおける否定の意味論

萱島 信　　中川 裕志

横浜国立大学　工学部　電子情報工学科

論理プログラミングにおいて，否定に関する表現力を拡張するために，複数個の正リテラルを持つ節（一般プログラム節）を含むプログラムに対して，述語にレベル付けを施すことにより，単一のモデルを決定するモデル論的なアプローチが近年盛んに行われている．

本稿では，このアプローチの拡張を行い，従来の方式では扱うことのできなかったプログラムに対してモデルを示すことのできる導出（SLBS-導出）を提案する．これは，ある特別なレベル付けを行ったプログラムに対し，まだ肯定であると証明されていないリテラルを，否定のリテラルであると仮定することによりNAFより強力な否定の概念を実現するものである．さらにこのSLBS-導出の健全性と完全性を証明した．

A Semantics of Nagation on the General Program

Makoto KAYASHIMA　Hiroshi NAKAGAWA
Department of Electrical and Computer Engineering
Yokohama National University, Yokohama 240 Japan

To implement negation, we must deal with general program clauses, but general program may have more than one model. So recently, the notion of the stratified logic program is proposed by Apt, Blair, and Walker.

In this paper, we extend the above notion, and propose a new resolution (called SLBS-resolution) by which we can obtain the unique model for a general program. In this resolution, we consider negative literals which are not recognized yet as positive ones. And further we prove the soundness and completeness of the SLBS-resolution.

0．初めに

論理型言語において，否定を扱うためにいままで色々な試みがなされてきた．その中で，“失敗による否定[1]（Negation as failure：以下ＮＡＦと略記する）”は，否定的な情報を演繹するための重要な規則であり，これに基づいた反駁としてＳＬＤＮＦ－反駁をあげることができる．

しかし，ＳＬＤＮＦ－反駁において束縛がなされるのは，正リテラルの呼出しが成功したときだけであり，負リテラルの呼出しでは束縛が行われない．このため，負リテラルによって解を生成することはできないので，ＮＡＦは単なるテストであり，論理的な否定のかなり不十分な代用ということができる．

そこで，論理的な否定を表現するために，プログラムに出現するリテラルにレベル付けを施すことによって，自然な最小モデルをただ一つ定義することのできるstratified logic program[2][3]というクラスが提案されている．［例１．１］

［例１．１］stratified programとそのモデル

```
p(X)←¬q(X)
q(a)←                ↔        {p(b), q(a), r(b)}
r(b)←
```

また，論理プログラムにおける大きな問題として無限再帰の問題をあげることができる．これにはTight Tree Semantics[4]を用いることにより，そのうちのいくつかの問題を解決することができる．

しかし，再帰的な負リテラルを含むプログラムは，否定を導入するとstratifiedでなくなるため，そのモデルを簡単に作ることはできなくなる．そこで，本稿では，再帰的な負リテラルを含んだ演繹データベースに対して，そのモデルを導出する方法について考察を行うことにする．

1．基本概念

1．1 ホーン節の拡張

プログラム中で，論理的な否定を表現するためにＳＬＤ－導出において基本的な役割を果たしているホーン節を次の様に拡張する．

［定義１．１］一般プログラム

Ｌ：一階述語論理の言語

A, B1, …, Bn, C1, …, Cm：Ｌのアトム

とする．ここで，通常の確定節(definite clause)に加えて“一般節”

$$A \vee \neg B1 \vee \neg Bn \vee \cdots \vee C1 \vee Cm \qquad n, m \geq 0$$

を加えたプログラム（一般プログラム）を考える．一般節は，次のように表すこともできる．

$$A \leftarrow B1, \cdots, Bn, \neg C1, \cdots, \neg Cm$$

（注　Aをhead　B1, …, Bn, ¬C1, …, ¬Cmをbodyと呼ぶ）■

これは直観的には，

「もし，B1, …, Bnが成り立ち，C1, …, Cmが成り立たないときにはAが成立する．」ということを表している．

また，通常のゴール節を拡張して，“一般ゴール”を導入する．これは，

$$\leftarrow L1, \cdots, Ln$$

の形をした節で，L1, …, Lnはリテラルである．

1．2 最小モデルの拡張

一般プログラムに対しては，model intersection propertyが成り立たなくなるので，最小エルブランモデルを用いることができない．そこで，それに替わるperfect modelという概念が導入されている[2]．最初に，それに必要な概念の定義を行う．

［定義１．２］Priority Relation

一般プログラムの節

$$A \leftarrow B1, \cdots, Bn, \neg C1, \cdots, \neg Cm$$

に対して次のようなpriority relationを定義する

(1) $\forall i (Ci < A)$

(2) $\forall i (Bi \leqq A)$ ■

これよりperfect modelは次のように定義される．

［定義１．３］Perfect Model

プログラムに対して，２つの異なったモデルＭ，Ｎを仮定する．Ｎ－Ｍに含まれているground atomをＡ，Ｍ－Ｎに含まれるものをＢとする．ここで，Ａ，Ｂのpriority relationが　Ａ＜Ｂ　である時，ＮはＭよりpreferableであるという．

Ｍよりpreferableなモデルが存在しないとき，モデルＭはperfectであるという．■

1．3 Broadly stratified database

ここでは，一般プログラムのある特別なクラスとしてBroadly stratifiedというクラスを提案する．

［定義１．４］Broadly stratified database

演繹データベースがBroadly stratifiedであるのは，データベース中の節

A←B1,…,Bn,¬C1,…,¬Cm

が，次の条件を満たすときである．

“Aの導出列中に再帰的に出現したAは，どの様な計算規則を用いても，以前に出現したAより簡単な形をしている” ■

ここで“A(t1)は，A(t2)より簡単な形をしている”というのは，“t1とt2に対して f(t1)＝t2 となるようなfunctor fが存在する”ということを表している．

一般プログラムに対して自然なある一つのモデルを決定するために，stratifiedという述語のレベル付けが考えられている．これは再帰的な述語の定義として負リテラルを用いることを許していない．そこで，これを拡張して，プログラム中のground atomに対するレベル付けとしてlocally stratifiedという概念が考えられている．

［定義１．５］locally stratified database[2][3]

データベース中の節

A←B1,…,Bn,¬C1,…,¬Cm

のすべてのground instanceに対して，階層分割化されたエルブラン領域 {H0,…,Hα,…,Hγ,…} を作ることができ，Aのレベルをiとした時，次の条件

(1) 全ての Bi (1 ≦ i ≦ n)は レベルPj {Pj：j≦i} に属する．

(2) 全ての Ci (1 ≦ i ≦ m)は レベルPj {Pj：j＜i} に属する．

を満たす場合locally stratifiedであるという． ■

Broadly stratifiedな演繹データベースは，１．２で定義したpriority relationを用いて，ground atomに対し“Aのレベルはその導出列に再帰的に出現するAのレベルより低い（簡単な方がよりレベルが高い）”というレベル付けを行うとlocally stratifiedとなり，Broadly stratifiedはlocally stratifiedのサブクラスであると考えられる．ここで，locally stratifiedの場合，全てのground atomの演繹列中に出現するリテラルのレベルは，そのground atomよりも高いか，もしくは同レベルのリテラルだけであったので，Broadly stratified databaseでも“すべてのground atomは，自身より上位のground atomだけで決定される”ということに注意する必要がある．

［例１．２］再帰的な負リテラルを持つプログラム

a．Broadly stratifiedである場合

even(0)←

even(s(X))←¬even(X).

b．Broadly stratifiedでない場合

even(0)←

even(X)←¬even(s(X)).

1．4 一般プログラムに対する写像T_P^*

従来のＳＬＤ－導出において，否定情報を演繹するために，ＮＡＦという非単調推論規則がある．これは，AがＳＬＤ有限失敗集合の元のときは¬Aを推論する規則である．この規則を加えた導出をＳＬＤＮＦ－導出と呼ぶが，この導出では計算規則として，次のような“安全な計算規則”を必要とする．

［安全な計算規則Ｒ］

(a) Ｒは基礎状態のときだけ負リテラルを選択する．

(b) あるゴール中の基礎代入の負リテラル¬Aを選択したら，Ｒは残りの計算を続けていく前に←Aを根とする有限失敗ＳＬＤＮＦ－木の生成を終わらせるよう試みる． ■

この計算規則の場合，負リテラルは基礎状態でのみ出現し，その呼出しは成功するか失敗するかの単なるテストに過ぎない．このため，ＮＡＦは論理的な否定のかなり不十分な代用であると言える．そこで負リテラルが基礎状態でないときでも選択できるようにするための考慮を行った写像を新たに定義する．

［定義１．６］一般プログラムに対する写像T_P^*

Lを一般プログラム，Iをエルブラン解釈とするとき，従来の写像T_P

$T_P(I)=\{A\in B_P : A\leftarrow B1,\cdots,Bn$はPの節の基礎代入例で$\{B1,\cdots,Bn\}\subseteq I\}$

を拡張して，次の様な写像T_P^*を考える

$T_P^*(I)=\{A\in B_P ; A\leftarrow B1,\cdots,Bn,\neg C1,\cdots,\neg Cm$はPの節の基礎代入例で

$\{B1,\cdots,Bn\}\subseteq I+$

$\{\neg C1,\cdots,\neg Cn\}\subseteq I-$

$I=I+\cup I-\ \}$

ただし

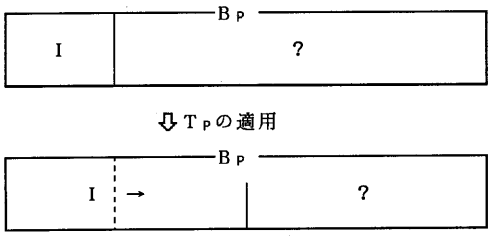
I+はエルブラン解釈の正リテラルによる部分集合

I-はエルブラン解釈の負リテラルによる部分集合

ただし，T_P^*の適用によって$\{\neg C1,\cdots,\neg Cm\}$の要素が減少しないようなI-を選択する

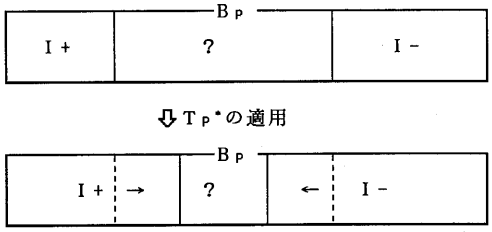
■

写像T_PとT_P^*では，負リテラルの扱いが異なる．写像T_Pに対するエルブラン解釈は正リテラルだけを考えているので，そこに含まれないリテラルの真偽値は偽であり，負リテラルであるとしている．しかし，実際にはエルブラン解釈に含まれていないものは“真偽値が未定”の要素であり，T_Pの場合はエルブラン領域とエルブラン解釈の関係を［図１．１］のように表すことができる．



［図１．１］

写像T_Pは単調なので，エルブラン解釈Iもまた単調に増加する．それに対して写像T_P^*の場合は，負リテラルをエルブラン解釈の要素として考えている．これよりエルブラン領域中の要素は真に対応する“正リテラル”と偽に対応する“負リテラル”だけでなく，“真偽値が未定のリテラル”という３種類の要素より構成されていることになる．［図１．２］



［図１．２］

そこで，写像T_P^*が単調であるためには，I+が単調に増加するだけではなく，“T_P^*の適用の前後でI-の要素が減少してはならない”という条件が外せない．

［例１．３］モデルが作れないケース

p(X)←¬p(X).

q(a)←

$I_1+=\{q(a)\}$とした場合　$B_P-I_1+=\{\neg p(a)\}$になるが，これをI_1-であると仮定し，写像を適用すると

$I_2=T_P^*(I_1)=\{q(a),p(a)\}$

になる．明らかに$I_2=I_2+$であり，$I_2-=\{\}$より

$I_3=T_P^*(I_2)=\{q(a)\}$

となり，単調ではなくなる．

［例１．４］負リテラルによるevenの定義

p(s(X))←¬p(X).

p(0).

ここで，I_1+を次のように設定する

$I_1+=\{p(0)\}$

これよりエルブラン領域でI_1+に属さないものはすべて否定であると仮定すると次のようになる．

$I_1-=\{\neg p(s^n(0)) : n\geq 1\}$

しかし，この全ての要素に写像を適用すると

$I_{2a}+=\{p(s^n(0)) : n\neq 1\}$

となり，T_P^*は単調にならない．

ここで，I_1-と$I_{2a}+$を比較してみると

$\{p(s^n(0)) : n\geq 2\}$

の部分は写像T_P^*の適用により真偽値が変化している．そこで，T_P^*の適用の前後で，真偽値の変化しないリテラルの集合だけが$I_1'-$であるとすると

$I_1'-=\{\neg p(s(0))\}$

になり，$I_{2b}+$が得られる．

$I_{2b}+=\{p(0), p(s(s(0)))\}$

この方式を繰り返し適用してみる．まず，

$B_P - I_{2b}+=\{\neg p(s(0)), \neg p(s^n(0)) : n \geqq 3\}$

から写像の適用により真偽値の変化する部分を取り除くと，I_2'

$I_2'=\{p(0), p(s^2(0))), \neg p(s(0)), \neg p(s^3(0))\}$

を得ることができ，これより

$I_3+=\{p(s^n(0)) : n=0,2,4\}$

が得られ，最終的に

$I_\infty+=\{p(s^n(0)) : n$は偶数$\}$

となる．

最小元からの写像の適用$T_P^* \uparrow$を考えたとき，正リテラルによる基礎節の存在が不可欠である．通常プログラム中にはこの条件を満たす節が存在するが，負リテラルの基礎節については記述されていないケースも考えられる．そこで，エルブラン解釈中の負リテラルによる部分集合 I- は，エルブラン領域に対する正リテラルの部分集合の補集合，すなわち $B_P - I+$ から作ることを考えるが，写像を単調にするため“I+の要素を選んではならない”という条件をつけることになる．しかし，エルブランモデルを求める途中でそのリテラルがI+に含まれるかどうかは決定できない．

そこで，実際にはその時点のエルブラン解釈で肯定であると証明されていないground atomを否定であると仮定して次のような方法で作ることにする．

［I-の作成法］

(1) その述語に使用する定数と関数の部分集合を決定する（関数はプログラムにより判別可能であるが、定数は明示的に示さなければならないケースもある）

(2) 部分集合が定数だけで構成される場合、ground atomとして出現しないものを否定のリテラルであると仮定する

(3) 部分集合が関数を含む場合、ground atomとして現れたものに、部分集合中の関数をすべて一回づつ作用させて、それが正のリテラルであると証明できないものは否定のリテラルであると仮定する

■

この方式ではレベルの高い負リテラルより順に生成することができる．プログラムがBroadly stratifiedである場合，真偽値の決定されたリテラルはそれ以降に真偽値が決定されたリテラルにより，その真偽値を再決定されることはない．すなわち，“上位のレベルのground atomは，下位のレベルのground atomの真偽値に影響を与えない”ので，一度I-に含まれていると考えられた要素はI+に入ることはない．これにより，写像の単調性を維持することができることがわかる．

［定理１．１］

［定理１．１］

PをBroadly stratifiedなプログラムとすると，写像T_P^*は単調である

証明

Xを2^{B_P}の有向部分集合とする．

PはBroadly stratifiedなので，一度I-に含まれた要素がI+に入ることはない．そこで$\exists I : I \in X$に対して，

$\{B1, \cdots, Bn, \neg C1, \cdots, \neg Cm\} \subseteq I \subseteq lub(X)$なので

$A \in T_P^*(lub(X))$

また，

$A \in T_P^*(I)$より　$A \in lub(T_P^*(X))$

よってT_P^*は連続であり，したがって単調である．■

また，この様なI-の作成法を用いることにより，次のような場合でも解を求めることができる．

［例１．５］次のようなデータベース

student(花子)←

student(太郎)←

teacher(一郎)←

に対して，質問

←¬student(X)

を与えることを考える．この場合functorが含まれていないので

I+= {student(花子),
　　student(太郎),
　　teacher(一郎)}

に対して作り出されるI-

I-= {student(一郎),
　　teacher(太郎),
　　teacher(花子)}

はすべて同レベルである．これにより，解として

X=¬student(一郎)

を返すことができる．

2．宣言的意味論

2．1 解代入

1．の定義を用いて，プログラムの解代入の概念を拡張する．

［定義2．1］

Pを一般プログラム，Gを一般ゴール←L1,…,Lnとする．comp(P)の正解代入とは∀((L1∧…∧Ln)θ)がcomp(P)の論理的帰結となるような解代入θのことである．但しcomp(P)はPの完備化された定義をさす

■

そこで次に，この定義がＳＬＤ－導出における正解代入を一般化したものであることを証明する必要がある．

［命題2．1］

Pを一般プログラムとすると，Pはcomp(P)の論理的帰結である．

証明

Mをcomp(P)のモデルとする．そこでMがPのモデルであることを示す．まず，

p(t1,…,tn)←L1,…,Lm

をPの一般プログラム節とし，節中の変数y1,…,ydにある割り当てを行うとL1,…,LmはMで真になると仮定する．またpの完備化された定義を考える

∀x1… ∀xn(p(x1,…,xn)↔E1∨…∨Ek)

Ei= ∀y1… ∀yd((x1=t1)∧…∧(xn=tn))
∧L1∧…∧Lm

ここで変数y1,…,ydに上と同じ割り当てを行い，xjにはtjを割り当てる．すると，L1,…,LmはMで真になり，またMは等号理論の公理1，を満たすのでEiはMで真になる．従ってP(t1,…,tn)はMで真になる．

■

さらに，comp(P)とPとの関係を明らかにするため，次の定理を証明する．

［定理2．1］

Pをプログラム，Gをゴール，θを解代入とする．θがcomp(P)∪{G}の正解代入であることと，θがP∪{G}の正解代入であることは同値

証明

［命題2．1］より

θがP∪{G}の正解代入

→　θはcomp(P)∪{G}の正解代入．

がいえる．そこで，逆向きを示すために，

Pで∀(A1∧…∧Am)がcomp(P)の論理的帰結ならば，∀(A1∧…∧Am)はPの論理的帰結にもなることを示す．

まず，x1,…,xnをA1∧…∧Amの変数とする．ここで∀x1… ∀xn(A1∧…∧Am)がPの論理的帰結であることを示すためには，S＝P∪{¬A1'∧…∧¬Am'}の充足不可能性を示せば良い．ただし，Ai'はAiの中の変数x1,…,xkを適切なスコーレム定数で置き替えたものである．

IがPのモデルならばTP*(I)⊆Iであり，PがBroadly stratifiedならばTP*は単調なので，不動点I'⊆Iが存在し，I'はcomp(P)のモデルとなる．仮定より∀(A1∧…∧Am)はI'で真なので¬A1'∨…∨¬Am'はI'で偽になり，I'⊆Iより¬A1'∨…∨¬Am'はIで偽になる．よってSは充足不可能

■

2．2 perfect model

ここでは，Broadly stratifiedなプログラムに対するperfect modelの不動点による特徴づけを与える．

［定理2．2］

PをBroadly stratifiedなプログラムとすると，perfect model MPに対して

MP＝lfp(TP*)＝TP*↑ω

が成り立つ．

証明

TP*がBroadly stratifiedなプログラムでは単調なことより

MP＝glb{I：IはPのエルブランモデル}
＝glb{I：TP*(I)⊆I}
＝lfp(TP*)
＝TP*↑ω

■

3．手続き的意味論

3．1 否定を含んだプログラムに対する反駁手続き

ここでは，否定を含んだプログラムに対する手続き的意味論を，従来のＳＬＤ－反駁を元に構築する．

一般プログラムに対する反駁は，おおまかには次のような方針

(1) 正リテラルに対しては，通常の反駁を行う．
(2) 負リテラルに対しては，その領域を仮定して反駁を行う．

で行うことにする．

そこで，Broadly stratifiedな一般プログラムに対する新しい導出アルゴリズム（名前を‘ＳＬＢＳ－導出’とする）は，負リテラルが基礎状態になくても選択することができる計算規則を採用する．ここで，負リテラルが基礎状態の場合は，従来のＳＬＤＮＦ－導出と同じくＮＡＦを用いることにする．

［定義３．１］ＳＬＢＳ－導出に対する計算規則の条件
(1) あるゴール中で，基礎状態の負リテラル¬Aを選んだら，Ｒは残りの計算を続けていく前に←Aを根とする有限失敗ＳＬＤＮＦ－木の生成を終わらせるように試みる
(2) あるゴール中で基礎状態にない負リテラル¬Aを選んだ場合，すでに基礎状態になっている正リテラルAと，プログラムのエルブラン空間から負リテラル¬Aを推定する

■

また，ＳＬＢＳ－導出の演繹列は次のように定義される．

［定義３．２］ＳＬＢＳ－導出における演繹列
ＰをBroadly stratified program，Ｇを一般ゴール，ＲをBroadly stratified programに対する計算規則とする．
ＲによるＰ∪{Ｇ}のＳＬＢＳ－演繹列　Ｇ0，Ｇ1，… は次の様に定義される．

(1) Ｇiを←L1，…，Lk，Ｒは正リテラルLmを選択すると仮定する．A←M1，…，Mqを入力節，LmとAの最汎単一化子をθ i+1とする．すると演繹される一般ゴールＧi+1は
←(L1，…，Lm-1，M1，…，Mq，Lm+1，…，Lk)θ i+1
となる．
(2) Ｇiを←L1，…，Lk，Ｒは基礎状態の負リテラルLmを選択すると仮定する．Lmは¬Aとする．すると←Aを根とする有限失敗ＳＬＤＮＦ－木の生成が試みられる．もし←Aが成功したら，副ゴール¬Aは失敗し，ゴールＧiも失敗する．もし，←Aが有限失敗したら，副ゴール¬Aは成功し，演繹される一般ゴールＧi+1は
←L1，…，Lm-1，Lm+1，…，Lk
となる．有限失敗した場合，θ i+1は恒等代入になる
(3) Ｇiを←L1，…，Lk，Ｒは基礎状態でない負リテラルLmを選択すると仮定する．Lmは¬Aとする．この場合，レベル付けされたAのエルブラン領域で，次のようなレベル
『正リテラルであると証明されていないリテラルがまだ存在しているレベルのうち，一番高いレベル』
に属するground atomより，正リテラルであることがまだ証明されていないものを選択し，それを　¬A←とする．
この節とLmの最汎単一化子をθ i+1とすると，演繹される一般ゴールＧi+1は
←(L1，…，Lm-1，Lm+1，…，Lk)θ i+1
となる．

■

3．2 ＳＬＢＳ－導出の健全性

この節では，３．１で定義した計算規則とＳＬＢＳ－演繹列を用いた導出の健全性を証明する．ＳＬＢＳ－導出はＳＬＤＮＦ－導出の拡張版であるので，基礎代入されていない負リテラルに関する部分だけ考察を行えばよいと考えられる．

［定理３．１］ＳＬＢＳ－導出の健全性
ＰをBroadly stratified program，Ｇを一般ゴール，ＲをBroadly stratified programに対する計算規則とすると，Ｐ∪{Ｇ}がＲによる有限失敗ＳＬＤＮＦ－木を持つならば，Ｇはcomp(Ｐ)の論理的帰結になる

証明 ■

G中の基礎状態の負リテラルは，一般プログラムに対するＮＡＦの健全性によりcomp(Ｐ)の論理的帰結であり，その反駁が有限失敗したとき，代入θは恒等代入になる．そこで，G中に基礎状態にない負リテラルがあっても，そのＲ－計算解代入が正解代入であることを示す．

Gをゴール ←A1,…,Ak(ただし A1,…,Akは基礎状態にない負リテラルを含む)また，θ1,…,θnをＲによるＰ∪{G}の反駁で使われた最汎単一化子とする．そこで∀((A1∧…∧Ak)θ1…θn)がcomp(Ｐ)の論理的帰結であることを，ＳＬＢＳ－反駁の長さに関する帰納法で証明する．

まずn＝1と仮定する．これはGが，←L1の形であることを意味する．次の３つの場合に分けて考える．

(a) L1が正のとき
(b) L1が負の基礎代入例であるとき
これらは，ＳＬＤＮＦ－反駁ですでにcomp(Ｐ)の論理的帰結であることが証明されている．
(c) L1が負リテラルで基礎代入例でないとき
この場合，仮定した否定の単位節 ¬A←に対して
L1θ1＝¬Aθ1
が成り立つ．L1θ1はcomp(Ｐ)のある仮定した否定の単位節の代入例より∀(L1θ1)はcomp(Ｐ)の論理的帰結になる．

次に，長さn-1のＳＬＢＳ－反駁から得られるＲ－計算解代入について結果が成り立つとする．
長さnのＰ∪{G}のＳＬＢＳ－反駁で使われる最汎単一化子の列をθ1,…,θnと仮定する．LmをGから選択されるリテラルとする．ここで
(a) Lmが正のとき
(b) Lmが負の基礎代入例であるとき
のときは，comp(Ｐ)の論理的帰結になる．
(c) Lmが負リテラルで基礎代入例でないとき
この場合も入力節は，仮定した否定の単位節 ¬A←によって
Lmθ1＝¬Aθ1
が成り立ち，Lmθ1はcomp(Ｐ)の論理的帰結である．そこで帰納法の仮定より∀((L1∧…∧Lk)θ1…θn)がcomp(Ｐ)の論理的帰結になることが判る

ここで，基礎代入例でない負リテラルが再帰的に定義されている場合は，否定の単位節を仮定するために，より上位のレベルの真偽値を知っている必要がある．このため，基礎代入例でない負リテラルの反駁は，効率が非常に悪くなると考えられる．

また，［定理３．１］より次のことがわかる．

［系３．１］
一般プログラムの成功集合はperfect modelに含まれる．
証明
comp(Ｐ)を一般プログラム，A∈ＢＰとする．
comp(Ｐ)∪{←A}にはBroadly stratified programに対する計算規則Ｒが存在すると仮定する．すると［定理３．１］よりAはcomp(Ｐ)の論理的帰結であり，Ｍpに含まれる．

３．３ ＳＬＢＳ－導出の完全性

本節では，Broadly stratified programを考察の対象として，［定理３．１］の逆が成り立つことを証明する．一般に，Ｐとして一般プログラムを考えるとＳＬＤＮＦ－反駁の完全性は証明することができない．これは，直感的にはＮＡＦによる無限失敗枝の存在がその原因であると考えられる．そこで，まず，Broadly stratified programに対するＮＡＦの完全性を証明する．

［定理３．２］ＮＡＦの完全性
ＰをBroadly stratified program，Ｇを一般ゴール，ＲをBroadly stratified programに対する計算規則とする．
Ｇがcomp(Ｐ)の論理的帰結ならば，comp(Ｐ)∪{Ｇ}はＲによる無限失敗ＳＬＤ－木を持たない．
証明
これは，基礎代入されたゴールＧが成功するか，もしくは有限失敗するかのいずれかであることと等価である．プログラムの導出列が無限枝になる可能性があるのは，再帰的なリテラルが出現したときだけであるから，Ｐの任意の一般節
A←Bi （i＝1,…,n：Biはリテラル）
の導出でBiが再帰的なリテラルであるときのみに注意すれば良い．

さらに，プログラムはBroadly stratifiedなので，termの形によるレベル付けを行うことができ，Biのレベルは常に高い．そこでAのレベルをPjとしてレベルによる帰納法を用いて証明する．

(a) Biが正リテラルの時

j＝2のとき

Biと単一化可能なレベルP1の述語が存在するときは，Biは成功する．これよりAも成功する．

Biと単一化可能なレベルP1の述語が存在しないときは，Biは失敗し，Aも失敗する．

n-1で成り立つと仮定する

j＝nのときBiのレベルはPn-1であり，仮定よりBiは成功，もしくは失敗する．よってAも成功もしくは失敗する．

(b) Biが負の基礎代入例であるとき

j＝2のとき

単一化可能な単位節が存在すれば，¬Biは失敗する，存在しなければ，¬Biは成功する．

n-1で成り立つと仮定する

j＝nのとき¬BiのレベルはP_{n-1}であり，仮定より¬Biは成功，もしくは失敗する．よってAも成功もしくは失敗する．

(c) Biが基礎代入例でないとき．

プログラムはBroadly stratifiedであるから，Biと単一化可能な負の単位節 ¬C← が存在し，¬Biは成功する．

これよりBroadly stratifiedなプログラムでは成功枝と有限失敗枝のみを持つ．

■

［定理３．２］から，プログラムをBroadly stratifiedに制限すると無限失敗枝を排除できることがわかる．これはプログラムが無限ループに落ち込むことを回避する効果がある．

一般プログラムに対し，制限を設けることにより，ある種の無限ループに落ち込むことを回避する試みとしては，Tight Tree Semanticsがあげられる．ここではbounded term sizeとfreedom from recursive negationという条件をつけている．この項の構造に関する条件はBroadly stratifiedよりも緩いが，この方法は基本的にはＮＡＦを用いているのでbindされていない負リテラルを許さない点が異なっている．

最後に，ＳＬＢＳ－導出の完全性は，次の２つの定理により証明される．

［定理３．３］

一般プログラムの成功集合は，そのperfect modelに等しい．

証明

一般プログラムをcomp(Ｐ)とする．［系３．１］が成り立つので，comp(Ｐ)のM_PがＰの成功集合に含まれることを示せばよい．AをM_Pの元とする．［定理２．２］より，ある$n \in \omega$について$A \in T_P^* \uparrow n$が成り立つ．そこでnについての帰納法で，$A \in T_P^* \uparrow n$ならばcomp(Ｐ)∪{←A}には反駁が存在し，Aが成功集合の元であることを示す．

n＝1のとき

$A \in T_P^* \uparrow 1$はAがcomp(Ｐ)の単位節の基礎代入例（仮定した単位節を含む）であることを意味する．明らかにcomp(Ｐ)には反駁が存在する．

n-1の時結果が成り立つとする

$A \in T_P^* \uparrow n$とする．T_P^*の定義と，［定理３．３］より，節B←B1,…,Bk（B1,…,Bkは負リテラルでもよい）の基礎代入例が存在し，ある代入θについてA＝Bθかつ{B1θ,…,Bkθ}$\subseteq T_P^* \uparrow (n-1)$が成立する．帰納法の仮定より各i＝1,…,kについてcomp(Ｐ)∪{←Bi}には反駁が存在する．

各Biθは基礎状態なので，これらの反駁を結合してcomp(Ｐ)∪{←(B1,…,Bk)θ}の反駁が作れる．したがってcomp(Ｐ)∪{←A}には反駁が存在する．

■

［定理３．４］ＳＬＢＳ－導出の完全性

ＰをBroadly stratified program，Ｇを一般ゴール，ＲをBroadly stratified programに対する計算規則とする．

comp(Ｐ)∪{Ｇ}が充足不可能と仮定すると，Ｒによるcomp(Ｐ)∪{Ｇ}のＳＬＢＳ－反駁が存在する．

証明

Ｇを一般ゴール ←A1,…,Akとする．comp(Ｐ)∪{Ｇ}は充足不可能なので，ＧはM_Pで偽になる．従って，Ｇのある基礎代入例Ｇθが，M_Pに関し偽になるので{A1θ,…,Akθ}$\subseteq M_P$が成り立つ．すると［定理３．３］より各i＝1,…,kについてcomp(Ｐ)の反駁が存在する．ところで各Aiθは基礎状態より，これらの反駁を結合すればcomp(Ｐ)∪{Ｇθ}の反駁が作れる．■

4．実際の処理系における問題点

以上よりＳＬＢＳ－導出は，プログラムがBroadly stratifiedである場合uniqueなモデルを持つことがわかった．プログラムがstratifiedであることの判別を行うことは，比較的簡単に行うことができる．しかし，locally stratifiedもしくはBroadly stratifiedであることを判別する方法はかなり難しい．この例を［例４．１］としてあげる．

［例４．１］判別が難しいケース

```
p(X)←q(X,Y),p(Y).          (1)
q(X,f(X))←                 (2)
```

ここで出現するリテラルのうち，(1)のheadに出現するp(X)と，bodyに出現するp(Y)のうち，どちらが高いレベルのリテラルであるかプログラムを見ただけでは判別することができない．

しかし(1)の述語ｑを実行すると(1)'のようになる

```
p(X)←p(f(X)).              (1)'
```

この結果，プログラムがBroadly stratifiedではないことがわかる．

実際に処理系を作る場合は，Broadly stratifiedであるかどうか，識別するために

(a) 他のリテラルを実行することにより，変種間の関係を調べる

ことと，実行効率をあげるために

(b) ＮＡＦをなるべく利用する

が必要であり，このためにはプログラム変換を活用する方法が考えられる．そのなかでも，なるべく高いレベルの述語から実行し，基礎代入されていない負のリテラルの導出は後回しにするためにゴールの順番を入れ換えることが有効である．

また，変種同士のレベルを比較するためには，termを構成する変数とfunctorが同一のものであることが必要である．この形にプログラムを変換するためには，fold,unfold変換をもちいればよいが，問題はどのゴールに対して適用すればよいのかを機械的に判別する方法がないことである．通常はupgoalを行うことにより問題が解決できると思われるが，fold,unfold変換などの技法をもっとうまく利用できる局面について考察を行う必要があるだろう．

さらに，否定情報を使用する局面についてであるが，従来は否定のリテラルをテストとしてだけ使用していた．ＳＬＢＳ－導出により，プログラムがBroadly stratifiedならば解の生成に負リテラルをしようできることが判ったが，これによりプログラムを作る際の自由度をあげることができるが，実際に色々な局面で使用してみる必要があるだろう．

5．終わりに

本稿では，Broadly stratified programという概念を使って，再帰的な負リテラルを含む演繹データベースに対して，そのモデルを導出するためのインタプリタについての宣言的意味論と，手続き的意味論を作り，その手続きであるＳＬＢＳ－導出の健全性と，完全性を証明し，さらにＳＬＢＳ－導出を行う実行系における問題点についての考察を行った．

【参考文献】

(1)Ｊ．Ｗ．ロイド　：論理プログラミングの基礎　産業図書

(2)T.C.Przymusinski：On the Declarative Semantics of Deductive Databases and Logic Programs
FOUNDATIONS OF Diductive Databases and Logic Programs

(3)T.C.Przymusinski：On the Relationship Between Logic Programming and Non-monotonic Reasoning
AAAI88

(4)有川，原口　：述語論理と論理プログラミング　オーム社

(5)中村　：Prologプログラムの等価変換と変換戦略に関する研究