ソフトウエア 工学 9 - 6
(1979. 3. 30)

構造的反証原理について

謝 章文
(京都産大・計科研)

1.はじめに

一階述語論理の証明問題が準決定可能であることはよく知られている。

一階述語論理の証明問題は，演えき定理により，論理式の恒真性または矛盾性(inconsistensy)の問題に，論理式の矛盾性の問題は，論理節の集合の矛盾性すなわち充足不能性の問題に帰着される。

論理節の集合の充足不能性の問題はエルブランの定理により，その集合より生成される充足不能な基礎論理節の集合をみつける問題に還元するかresolution-refutationにより，論理節の集合より□(空論理節)を導出する問題に還元される。

エルブランの定理に直接基づく方法は基礎論理節という概念を用いて命題算の証明手続きを利用する。

これに対してresolution-refutationはmost general unifier (mgu)を用い，直接一階述語算の論理節を扱う証明手法である。

resolution-refutation法は，それ以前の証明手続きよりは，原理的に効率がよいものであるが，まだまだ実用にはほど遠いものである。

ここでは，高効率な新しいresolution-refutation法である構造反証原理(Structural Refutation Principle)について概説する。

構造反証原理は，与えられる論理節の集合が反映する概念間の構造とすべての作成可能な反証図に関する構造のグラフ的考察およびDavis and Putnam Method (とくにsplitting rule)の拡張に基づくsubgoal approachに基づく。

この原理は，定理の自動証明に基づくプログラムの合成のために開発されたものであり，そのため，すべての異なるモデルを構築することを目的としたものである。

2.基本的考察と基本方針

考察1

述語算と命題算の相違は，つぎの2つの観点から考察される。(図1)

1. P-リテラルの一意性
2. mguの存在条件の有無

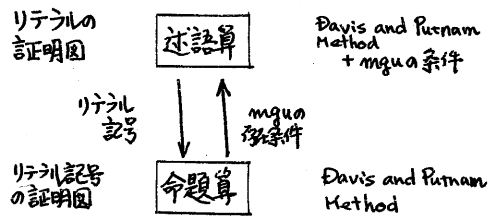


図1 命題算と述語算の関係

1.の観点から，述語算は，引数部を無視すれば，命題算に縮退する。

命題算においては，論理節の充足不能性の問題は，アルゴリズムが存在する。

この考察に基づき，リテラルの引数部分を消去したものをリテラル記号と呼ぶ。命題算の場合，リテラルとリテラル記号は一致する。

方針1

述語算の反証図(refutation schema)の構成において

1° リテラル記号の反証図を構成する

2° リテラル記号の反証図をmguの存在条件で選択する。

リテラル記号の反証図は，命題算の反証図であるが最簡形とは限らない。

考察2

論理節の集合Sが充足不能であるならば，resolution-refutation法により，

empty clause□が生成される。(resolution の completeness and soundness より)

ある論理節から始めて，resolution－refutation 法で□が導出されたならば，その論理節の各リテラルからも□が導出される。これは，その相補リテラルがSから導出されることを意味する。

方針2

リテラル記号の反証図の構成において，

1° Sの要素を1つ定め，始論理節とする。

2° 考察より，始論理節から□が導出されるならば，その各リテラルの相補リテラルがSから導出されねばならず，そのためにはSにその相補リテラルを含む論理節が存在せねばならない。

3° それらの論理節のその相補リテラルを除いた部分から少なくとも1つ□が導出されねばならない。

この手続きを繰返すことにより，すべてのリテラル記号の反証図を網羅する。

方針3

考察1の観点から，命題算の証明手法であるDavis and Putnam Method に mgu の存在条件を導入し，述語算へ拡張する。

特に，splitting rule の拡張から効果的な subgoal approach が可能になる。

方針4

tree search のような総当り法は，論理節の集合Sを充分縮退させた上で行なう。

3. Davis and Putnam Method の拡張

Davis and Putnam Method を簡単に紹介する。これはつぎの4つの規則からなる。

Sを命題算の論理節の集合とする。

I. Tautology Rule

Sから全ての tautology を除いても充足不能性は不変である。

II. One-Lieral Rule

リテラルLがSの中に単独で存在すれば，Lを含む論理節および$\bar{L}$のすべてをSから消去しても，充足不能性は不変である。

III. Pure-Literal Rule

相補リテラルがSの中に存在しないリテラルに対しては，それを含む論理節を消去しても，充足不能性は不変である。

IV. Splitting Rule

Sの充足不能性(goal)をあるリテラルLに着目して，つぎの2つのSの部分集合S_1, S_2の充足不能性(subgoal)に帰着する。

S_1(S_2)は，SからL($\bar{L}$)を含む論理節および$\bar{L}$(L)をすべて除いたものである。

述語算では，リテラル記号がトートロジイ形になる論理節は重要な働きを持つ。構造反証原理では，すべての反証図の構成を目指すので，One-Literal Rule は用いない。Pure-Literal Rule はリテラル記号を考慮して適用する。

Splitting Rule の拡張

S_1はSに論理節Lを加え，One-Literal Rule を適用したものと等しい。

S_1から導出される□は，条件Lの下でのSからの□の導出とみなされる。

これを□ if L または□{$\bar{L}$}と表わす。同様にして，S_2からの□は□{L}である。命題算では，{L, $\bar{L}$}の充足不能性が自明であるので，Splitting Rule が成立する。

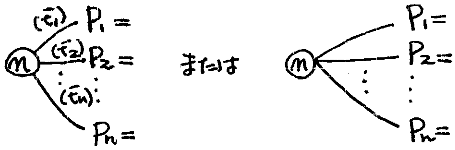
見方を変えると，S_1, S_2では，Lおよび$\bar{L}$以外のリテラルの消去を行ない，Lおよび$\bar{L}$だけの論理節が導出されたとき，つぎのstepとして，それらの論

理節から□を導出することと考えられる。
　述語算に対しても，リテラル記号に着目すれば，この見方に従って適用することができる。ただし，1段目・2段目ともにmguの存在条件が考慮されなくてはならない。

4. 構造的反証原理(その1)

4.1 論理構造グラフ

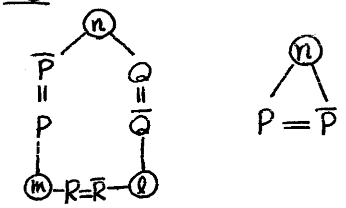
　構造的反証原理は，構造方程式と構造反証グラフからなる。構造方程式は，具体的な反証手続きを定め，構造反証グラフはそのために必要な全体情報から指針を与えるものである。
　また，これらの方法の基礎概念として論理構造グラフを用いる。
　論理節はすべて積表現を用い，番号が付けられているものとする。
　論理節の集合Sおよびその始論理節が与えられたとき，Sに対する論理構造グラフは，つぎの規則によって作成されるものである。
　規則1　Sに出現するすべての述語記号P_iに対して，$P_i=\overline{P}_i$ を書く。
　規則2　番号nの論理節$P_1(\bar{t}_1)P_2(\bar{t}_2)\cdots P_n(\bar{t}_n)$に従って，



と書く。特に，始論理節はⓝで表わす。
　引数部分を記さない論理構造グラフは，Sのリテラル記号に対する論理構造グラフである。引数部分をもつ論理構造グラフは，nodeⓝの各edgeにすべての変数記号に対するλ-baundを施しているものとする。
　これ以後，簡単のため，引数を記さない論理構造グラフを用いる。

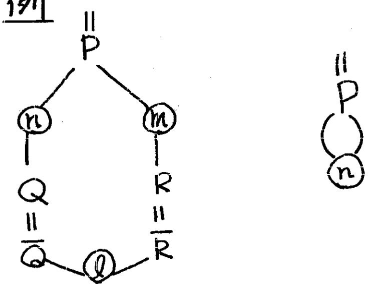
　Iループとは，−=−だけで構成されるループをいう。

例



　Mループとは，−=−以外に一ヶ所−‖−を含むループをいう。

例



−‖−を2ヶ所以上含むループは，反証手続きにおいては重要ではない。

4.2 構造方程式と構造反証グラフ

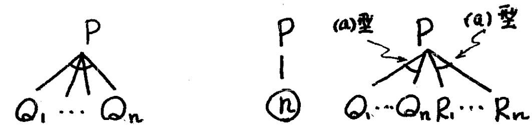
　構造方程式は，ゴールおよび方程式とよばれる部分からなる。
　Cを始論理節とするとき，□[C]をゴールと呼ぶ。また，Pをリテラル記号とするとき，P-方程式はつぎのようなものである。

$$P = C_1 + C_2 + \cdots + C_n \quad n \geq 1$$

　ここで，各C_iは$L_1[L_2\cdots L_k]\{L_{k+1}\cdots L_r\}$の形である。$L_i$はリテラルを表わし，特に$L_1$はP-リテラルである。$L_1$以外の部分は，ない場合がある。また，Pを方程式のidentitierと呼ぶ。
　構造反証グラフは，source nodeに□が，sink nodeにⓝ，{P}，または[P]が，および，他のnodeにリテラル記号が対

応する and/or graph である。ただし、nは節番号、Pはリテラル記号である。

$\bar{P}$-リテラルを含む論理節の集合Sが与えられたとき、各Sの要素に対して、つぎのグラフ(a)を考え、Sに対して(a)の形のグラフをorでまとめた(b)を考える。便宜的に、(b)のグラフをP-グラフと呼ぶ。(a)において、特に$n=0$のときは、節番号を書く。(α')



$\bar{P}(-)Q_1(-)\cdots Q_n(-)\in S$

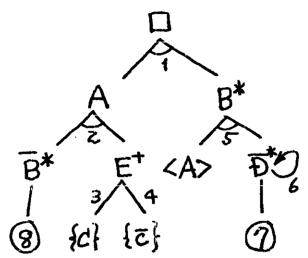
(a)　　(α')　　(b)

構造方程式および構造反証グラフを例をもちいて概説する。

論理節の集合Sを、つぎのものとする。(始論理節は1)

1.	AB	5.	$A\bar{B}\bar{D}$
2.	$\bar{A}\bar{B}E$	6.	$D\bar{D}$
3.	$\bar{E}C$	7.	D
4.	$\bar{E}\bar{C}$	8.	B

Sの構造グラフおよび構造方程式のうち、その展開をdepth-first かつ leftmost-first としたものは、つぎのようになる。



$\square[A\ B^*]^1$

$A=\bar{A}[\bar{B}^*\ E]^2$

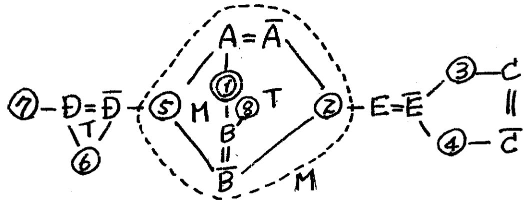
$\bar{B}^*=B^8$

$E^+=\bar{E}\{c\}^3+\bar{E}\{\bar{c}\}^4$

$B^*=\bar{B}[A\bar{D}^*]^5$

$\bar{D}^*=D[\bar{D}^*]^6+D^7$

また、Sに対する論理構造グラフはつぎのようになる。



グラフにおいて、同一リテラル記号Aは、一つを除いて<A>と表わされる。ただし、<A>がAの子孫となるような場合は、$\bar{D}$のようにループで表わし、$\bar{D}^*$とする。

論理節の7,8は、unit clause であるので、グラフでは⑦,⑧によってそれを示す。c, $\bar{c}$は、EにおいてOR-結合しており、この場合、それぞれ$\{c\}$,$\{\bar{c}\}$, E^+と表わされる。

B, $\bar{B}$は□においてAND-結合しており、この場合には、それぞれB^*, $\bar{B}^*$ と表わされる。

{ }内のリテラルを分岐リテラル、*, + のつけられたリテラルをそれぞれ*, +リテラルとよぶ。

方程式において、各identifierは、グラフ中のsource node, sink node以外のリテラル記号に対応する。

各節の右肩の数字は、対応する節番号である。$E(\bar{D})$式に現われる"+"は、$\bar{E}(D)$リテラルを含む論理節が二個以上存在することを示す。

[ ]内のリテラルは対応する方程式を持ち、binary resolution の適用対象となる。分岐リテラルは方程式においても{ }内に記され、binary resolution または binary factoring の適用対象となる。

論理構造グラフのループは構造反証グラフにおいてつぎのように表現される。

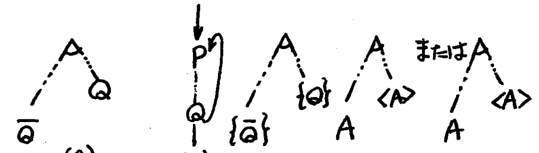
始論理節からの展開で

i. Tループにⓝから入った場合　(a)

ii. TループにPから入った場合　(b)

iii. MループにPから入った場合　(c)

iv. Mループにⓝから入った場合およびその他のループに入った場合　(d)



(a)　(b)　(c)　(d)

グラフによって定められるリテラル記号の半順序(ロに近い方が大きいとする)に従って、それをidentifierとする方程式およびリテラル間に上位、下位の概念を導入する。また、P-方程式内のPリテラルを同位リテラルと呼ぶ。

各節は、[ ]部がない場合に基本節、[ ]部に下位リテラルを含む場合に下位節、[ ]部に同位リテラルを含み、下位リテラルを含まない場合に同位節、[ ]部が上位リテラルのみの場合に上位節という。

4.3 構成手順

構造方程式と構造反証グラフの構成手順の概要

1° 始論理節から、goalおよび構造反証グラフの第1段を作成する。

2° 作成中のグラフのleafにsink node以外のリテラル記号が存在するとき、その内から任意の1つPを選び、$\bar{P}$-リテラルを含む未使用のSの論理節のすべてを用いて、node Pを一段展開する。このとき、同一リテラル記号が存在すれば、< >またはループ(および*)処理を施こす。相補リテラル記号が存在するときは、And結合、OR結合の別に従い、*, { }, +処理を施こす。また、このリテラル記号に対応する方程式を作り、Sから、方程式に用いられた論理節を消去する。

1°,2°の処理を、leafにsink node以外が存在しなくなるか、または、それ以外のleafに対する2°の処理で、対応する未使用の論理節がSに存在しなくなるまで続ける。

以上の手順で作成されたグラフにつぎの処理を施こす。

3° (pure-literal rule) leafにsink node以外のリテラル記号が存在するとき、そのリテラル記号を含むsubgraphでrootがOR-結合で他のsubgraphと結ばれる最小のものを消去する。このようなsubgraphが存在しないときは、その始論理節から始まる反証は不可能である。

4° (+消去) +リテラル記号P^+が存在するとき、対応する$\bar{P}$-リテラルが存在するならば、$\bar{P}$式をたてる。そうでないならば、+印を消去する。

Connecterの導入

特に、この手続において、leaf $\bar{P}$の展開において、Sの要素として$PP\bar{Q}$の形のものがある場合は、新しいリテラルAを導入し、この論理節を$PA(x)$、$\bar{A}(x)P\bar{Q}$なる二つの論理節でおきかえる。$A\bar{A}$はconnecterと呼ばれ、その引数は、同一でPと$P\bar{Q}$の共通変数の並びである。二個以上Pが含まれるときも同様とする。

5. 構造反証原理(その2)

5.1 解法の方針

方針1

リテラル記号に着目し、その消去を行なう。

方針2

拡張されたsplitting ruleを用い、反証手続きを階層化する。(subgoal approach) step1では、分岐リテラル以外の消去を行なう。step2で、残された分岐リテラルに対して、再び構造的反証原理を適用する。

方針3

step1では、*リテラル以外の消去を

優先させる。

方針4

Tree search のような総当りは，必要最少限にとどめる。

5.2 解法手順

解式とは，方程式において identifier が*リテラル記号でなく，すべての節が[]部に*以外の同位，下位リテラル記号を含まないものである。

構造方程式の解法とは，リテラル記号の個数（方程式の数）および，論理節の個数を減らし，必要最少限の総当りによって □{-} を求めることである。

解法手順の大すじを述べる。

0° *リテラル記号以外に対する pure literal rule の適用。このとき，方程式がなくなれば，5°へ。

1° 解式の適用を*方程式（identifier が*リテラル記号であるもの）のみになるまで繰返す。

2° 拡張された pure literal rule の適用を行なう。式がなくなれば，5°へ行く。

3° #消去を行なう。これは，2°において導入される#リテラルに対する処理である。

4° 1°，3°をできるだけ繰返し，ループ処理を行なった後，総当りを行ない，□{-}を求める。

5° Sから方程式に用いられた論理節を除きS'とする。

S'=φなら，反証不能。

S'≠φなら，S'から新しく論理節を選び始論理節とし，S'に対して構造反証原理を適用する。

*リテラル記号以外に対する pure literal rule

対応する方程式のない*以外のリテラル記号を含む節を，すべて消去することである。

解式の適用

解式があれば，それを用いてその式の identifier であるリテラル記号を消すことである。

拡張された pure literal rule

*リテラル記号に関しては，Tautology rule を適用できないので，拡張された pure literal rule を適用する。

1° *方程式に対しては，[]部のない項あれば，そのリテラル記号の*をすべて#におきかえる。

2° []部に#を含む節を少なくとも一つ含む方程式の identifier のリテラル記号の*を#におきかえる。

3° 2°を可能な限り繰り返す，残った*方程式の identifier P^* に対し P^* の方程式以外の方程式から，P^*リテラルを含む節をすべて消去し，P^*方程式を除く。

#消去

i) 方程式の各同位節，

$$P^\# = \bar{P}(t_1)[P^\#(t_2)]$$

および，他の方程式のつぎの形の各節

$$Q = \bar{Q}[\bar{P}^\#(t_1)P^\#(t_2)]$$

のそれぞれに対して t_1，t_2 が共通変数を含み，同一でないとき，#を*におきかえ，そうでないとき，#を消去する。

ii) 方程式の右辺に P^* が少くとも一度出現すれば，$P^\#$方程式の identifier を P^* とおきかえる。また，$P^\#$，P^*，$\bar{P}^{all}$ が現われなければ，対応する $P^\#$方程式の identifier を P とおきかえる。

iii) P方程式のすべての同位節を，それ以外の節を用いて解式にする。

ε効果による Loop 処理

i) #印のある節だけからなる部分方程式を考え，[]部を持たない上位節 $\bar{R}[P^\#]$ を用いて，上位式中の対応する下位節を消去する。これを繰返えす。

ii) $P = \bar{P}(t_1)[P^\#(t_2)]$ が出現し，t_1，t_2 の mgu が ε であるような上位節のすべてで，もとの方程式の同位節を展開し，同位節の消去を行なう。

このように同位節のなくなった節を

*解式と考え，解式の適用を行なない対応する*リテラルを消去する。
(LPSにおけるmsortの例題参照[4])

総当りの処理

このようにまとめた方程式のループは，すべて意味を持つものであるので，本質的に総当りを行なう必要がある。

Loopに関して閉じている構造反証グラフの部分グラフで，source nodeが一つであるようなものに対しては，partial map classifierとなるような新しい関数記号を導入し，非決定性の汎関係方程式を構成することにより，処理の効率をあげる。

LPSにおける，Đ axiomの導入は，この方法に対応する。[2]

5.3 subgoal approachの手順

step 1において，{}部のない□が求められれば，反証としては完結したことになる。ただし，すべてのモデルの構成を必要とするLPSでは，このような場合でも，step 2へ進まなくてはならない。

step 2では，つぎの四つの論理節の集合をまとめ，この中から一つを始論理節として選び出し，これに対して構造反証原理を再び適用する。

i) step 1で求められたすべての{}に対して，その{}部にbinary factoringを施して得られる論理節の集合

ii) step 1の構造方程式で使用されなかったらの論理節の集合、

iii) step 1の□{}の作成に関与しなかった論理節の集合

iv) step 1で導入されたconnecterを含む論理節の集合

6. 例題

始論理節は1番目の節とする。

例1

4.2の例題を解く。
例題の構造方程式を再記する。

1. $\square[A\ B^*]$
2. $A=\bar{A}[\bar{B}^*E]$
3. $\bar{B}^*=B$
4. $E=\bar{E}\{c\}+E\{\bar{c}\}$
5. $B^*=\bar{B}[A\bar{Đ}^*]$
6. $\bar{Đ}^*=Đ[\bar{Đ}^*]+Đ$

0のpure literalがないので，Pの解式の適用を行なう。

ここで，E式は解式であるので，Eを消去し，2から7を得る。

7. $A=\bar{A}[\bar{B}^*]\{c\}+\bar{A}[\bar{B}^*]\{\bar{c}\}$

このとき，Aは解式であるので，Aを消去し，ゴールが8，5式が9となる。

8. $\square[\bar{B}^*B^*]\{c\}+\square[\bar{B}^*B^*]\{\bar{c}\}$
9. $B^*=\bar{B}[\bar{B}^*\bar{Đ}^*]\{c\}+\bar{B}[\bar{B}^*Đ^*]\{\bar{c}\}$

これで，*式のみになったので，拡張されたpure literal ruleの適用を行なう。まず*を#でおきかえる。

10. $\bar{B}^*=B$
11. $\bar{Đ}^\#=Đ[\bar{Đ}^\#]+Đ$
12. $B^\#=\bar{B}[\bar{B}^\#\bar{Đ}^\#]\{c\}+\bar{B}[\bar{B}^\#\bar{Đ}^\#]\{\bar{c}\}$
13. $\square[\bar{B}^\#B^\#]\{c\}+\square[\bar{B}^\#B^\#]\{\bar{c}\}$

この例では，pureな*リテラル記号はない。

つぎに#消去を行なう。

14. $\bar{Đ}^\#=Đ[\bar{Đ}]+Đ$
15. $\square[\bar{B}B]\{c\}+\square[\bar{B}B]\{\bar{c}\}$

ここで，Đの方程式の#印が消される。

16. $\bar{Đ}=Đ[\bar{Đ}]+Đ$
17. $B=\bar{B}[\bar{B}^\#\bar{Đ}]\{c\}+B[\bar{B}^\#\bar{Đ}]\{\bar{c}\}$

これを解式にする。

18. $\square=\square+\square$
$=\square$

これを用いてĐを消去する。12は19になる。

19. $B=\bar{B}[\bar{B}^\#]\{c\}+\bar{B}[\bar{B}^\#]\{\bar{c}\}$

これは解式である。Bを消すと，ゴールは20になる。

20、 $\square[\bar{B}\bar{B}^{\#}]\{\bar{c},c\}+\square[\bar{B}\bar{B}^{\#}]\{\bar{c},c\}$
$+\square[\bar{B}\bar{B}^{\#}]\{\bar{c},\bar{c}\}$

20は $\bar{B}$が無くなったので、21となる。

21、 $\square[\bar{B}\bar{B}]\{c,c\}+\square[\bar{B}\bar{B}]\{c,\bar{c}\}+\cdots$

10式は、22となり、解式になる。

22、 $\bar{B}=B$

$\bar{B}$を消去すると、21はつぎの23となる。

23、 $\square\{c,c\}+\square\{c,\bar{c}\}+\square\{\bar{c},\bar{c}\}$

step2で、この23にbinary factoringをほどこし、つぎの集合で始める。

$\{c, c\bar{c}, c\}$

たとえば、cを始論理節とする。

24、 $\square[c^*]$

25、 $c^* = \bar{c}[c^*]+\bar{c}$

これ以後は、つぎのようになる。

26、 $c^{\#} = \bar{c}[c^{\#}]+\bar{c}$

27、 $\square[c^{\#}]$

28、 $c^{\#} = \bar{c}[c]+\bar{c}$

29、 $c = \bar{c}[c]+\bar{c}$

30、 $c = \bar{c}$

31、 $\square$

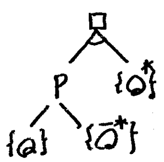
例2

与えられた論理節の集合を

1. PQ　　3. $\bar{P}Q$
2. $\bar{P}\bar{Q}$　　4. $P\bar{Q}$

とし、以下、例1と同様に行なう。

1. $\square[P]\{Q\}$
2. $P=\bar{P}\{Q\}+\bar{P}\{\bar{Q}\}$



この構造方程式は、2が解式であるから、3を得る。

3. $\square\{Q,Q\}+\square\{Q\bar{Q}\}$

3の第2項は、1節がトートロジーなので除く。

step 2

1. Q　　2. $\bar{P}\bar{Q}$　　4. $P\bar{Q}$

$$\square\{Q\}$$
$$\dfrac{Q=\bar{Q}\{\bar{P}\}+\bar{Q}\{P\}}{\square\{\bar{P}\}+\square\{P\}}$$

step3

1. $\bar{P}$　　2. P

$$\square[\bar{P}]$$
$$\dfrac{\bar{P}=P}{\square}$$

この例では、binary resolutionの回数は、4回、binary factoringは1回である。

つぎの反証図では、binary resolutionが3回、binary factoringが5回で等しくなっている。

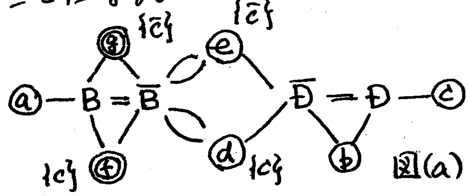
$$\dfrac{\dfrac{\dfrac{\bar{P}Q \quad \bar{P}\bar{Q}}{\bar{P}\bar{P}}}{\bar{P}} \qquad \dfrac{\dfrac{PQ \quad P\bar{Q}}{PP}}{P}}{\square}$$

例3

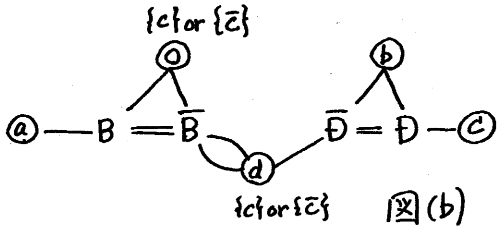
例1において、各リテラル記号が述語算のリテラルであり、#消去で*リテラルに書き換えられるならば、(例えば、$\bar{B}(x)\ B(f(x))$ の場合)、14以下は、つぎのようになり、Loop処理、総当りの対象となる。

14. $\bar{B}^* = B^{a}$
15. $\bar{D}^* = D[\bar{D}^*]^{b}+D^{c}$
16. $\bar{B}^* = \bar{B}[\bar{B}^*\bar{D}^*]\{c\}^{d}+\bar{B}[\bar{B}^*\bar{D}^*]\{\bar{c}\}^{e}$
17. $D[\bar{B}^*B^*]\{c\}^{f}+\square[\bar{B}^*B^*]\{\bar{c}\}^{g}$

この時点での論理構造グラフは、図(a)であり、これはつぎの4つのより簡単な論理構造グラフ(図(b))の問題をとくことになる。



図(a)



図(b)

構成の途中で，つぎの条件をみたすPMC，$\beta(x)$が導入される。

$$\forall x\ \bar{B}(\beta(x))$$

反証手続きは，つぎのようなPMCの記号計算によりなされる。

$$\begin{aligned}
\alpha((kf)^2g^2y) &= k\alpha(fkfg^2y)\\
&= kfk^{-1}\alpha(k^2fg^2y)\\
&= kf\alpha(kfg^2y)\\
&= kfk\alpha(fg^2y)\\
&= kfkf\beta(g^2y)\\
&= kfkfg\beta(gy)\\
&= kfkfg^2\beta(y)\\
&= kfkfg^2by
\end{aligned}$$

この計算は，非決定性の手続きであり，これにおいて，停止し，かつ結果がωでないものが存在すれば，反証がなされたことを示し，その結果が反例である。

例4

つぎに，Partial map classifier(PMC) を用いた例をあげる。

与えられた論理節の集合を，

1、$A(fx)B(x)$　4、$A(fx)\bar{A}(kx)$
2、$A((kf)^2g^2y)$　5、$\bar{B}(gx)B(x)$
3、$A(kx)\bar{A}(x)$　6、$\bar{B}(bx)$

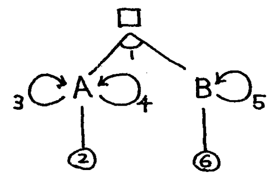
とする。この構造反証グラフは図(a)のようになる。

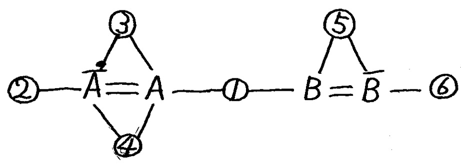
ここでは，構造反証原理により，総当りを考えなくてはならない。このとき，まず論理構造グラフ(b)を参考にし，始論理節を2に変更する。

ゴール□$[\bar{A}((kf)^2g^2y)]$より，PMC,$\alpha(x)$を導入する。$\alpha(x)$は，

$$\forall x\ A(\alpha(x))$$

をみたし，図(c)のように構成される。

(a) 

(b) 

(c)
$$\alpha(x)=\begin{cases} k\alpha(y)\cdot\sigma & \text{if } \exists\sigma=mgu(x,ky) \quad \text{from} \quad A(kx)[\bar{A}(x)]^3\\ fk^{-1}\alpha(ky)\cdot\sigma & \text{if } \exists\sigma=mgu(x,fy) \quad A(fx)[\bar{A}(kx)]^4\\ f\beta(y)\cdot\sigma & \text{if } \exists\sigma=mgu(x,fy) \quad A(fx)[B(x)]^1\\ \omega & \text{otherwise}\end{cases}$$

(d)
$$\beta(x)=\begin{cases} g\beta(y)\cdot\sigma & \text{if } \exists\sigma=mgu(x,gy) \quad B(gx)[B(x)]^5\\ by\cdot\sigma & \text{if } \exists\sigma=mgu(x,by) \quad B(bx)^6\\ \omega & \text{otherwise}\end{cases}$$

(e)
$$\alpha(x)\ :\ fk^{-1}\beta(ky)\cdot\sigma \quad \text{if } \exists\sigma=mgu(x,fy),\ \bar{P}(x) \qquad A(fx)[B(kx)]\{P(x)\}$$

$$A(\alpha(x)) \qquad B(\beta(x))$$

7、おわりに

構造的反証原理は，論理節の集合の全体構造を論理構造グラフで把握し，反証手続きに関する全体構造を，論理構造グラフと密接に関連する構造反証グラフで把握することにより，構造方程式における各論理節に，反証に対して有効な推論の方向性([]部)とその推論に必要な条件部({}部)を導入する。この方向性と条件を機械的にみつけることが，この原理の特徴である。そのため，これを用いることにより，機械ばかりではなく人間にとっても，非常に反証がしやすくなる。

構造的反証証原理の基礎をなす考えはどれもSimpleなものである。

また，unificationに方向性を導入した反証を考えることにより，その構造反証グラフから自然なflowchart programが合成される。このことからも，構造的反証原理がNaturalであることもわかる。

容易に形式化および機械化ができ，他の方法に較べ非常に効率がよいことは，我々のLPS projectにおいて既に実証済みである。

謝辞.

日頃御指導いただく京都産業大学上村義明教授に深謝します。また，原稿の作成に大変協力いただいた，京産大LPS projectのメンバーである，大村伸一，琴野実，東本謙治，馬渡幸夫の各氏に感謝します。

参考文献

1. 謝(1979)：論理的プログラム合成と構造的反証原理.
   情報処理・ソフトウェア工学研究会 3月
2. 謝(1977)：Foundation of Logical Program Synthesis.
   情報処理・ソフトウェア工学研究会 3月
3. 謝(1975)：Resolution-Refutation法によるMechanical Program Synthesis
   信学会AL 74-40
4. 大村,琴野,馬渡,東本(1979)：構造的反証原理に基づく論理的プログラム合成系・Pilot Ⅲ
   情報処理・ソフトウェア工学研究会 3月.