記号処理 18－3
(1982. 3. 25)

Prolog処理系の一改良案(知的後戻り法)

佐藤 泰介
(電子技術総合研究所)

1. はじめに

処理の非決定性をその本質的特徴とするPrologの後戻り(back tracking)をより効率的に行う方法を提案する。この方法はヒヒと完全な(complete)反証法であるOL(ordered linear)反証法及びOLから派生した線形反証法の効率的後戻り法を基礎にしている。

現在のPrologの証明法はtop-down, serialであり、処理が行き詰ると後戻りを開始し、選択肢の残っている地点迄戻り他の道を探す。このような後戻りは単に今迄辿って来た道を逆順に(historical order)戻るだけであり、(単一化の)失敗の原因を分析することなく行うもので、Naive Backtracking、略してNBと呼ばれる。

NBの欠点は左のsampleプログラムを見れば容易に理解される。

```
← A(x,y), B(x), C(y)
      ··· AND goal

A(z,a) ←
A(z,b) ←
B(a) ←
B(b) ←        x,y,zは変数
C(b) ←        a,bは定数
```

図1. プログラム例

AND goal A(x,y), B(x), C(y)に対し、まずA(x,y)がcallされる。すると A(z,a)←と単一化して代入(変数と値a結合, binding) {z/x, a/y}が生じる。次にB(x)がcallされる。xの値はzであるから、実際にcallされるのはB(z)である。このgoalはB(a)←と単一化して成功する。代入{a/z}が生じ、この時点のtotalの代入は{z/x, a/y}・{a/z}となる。次にC(y)がcallされる。実際は代入によりC(a)がcallされる。するとC(a)と単一化可能な入力節がないので後戻りを開始する。NBによると直前の推論ステップ、即ちB(z)とB(a)←を単一化した時点に戻り、これをやり直して他の入力節B(b)←を選ぶ。この選択は失敗に終わる。何故なら、後戻りを開始する原因となったC(y)とC(b)←の単一化の失敗を引き起した変数の値aに何ら影響を及ぼさないので、たとえB(b)←の選択が一時的に成功しても、いずれgoal C(y)を解く段階で再度単一化の失敗が起こるからである。このような無駄を避けるには失敗の原因となった変数yと定数aの結合が生じた時点、即ちgoal A(x,y)とA(z,b)←の単一化を行った時点に戻ってやり直しを行なわなければならない。

以上の考察を一般化したものが単一化の失敗の原因を分析し、分析結果に応じて後戻りを行う"知的後戻り(Intelligent Backtracking, 略してIB)"である。IBは一歩一歩戻るのではなく、何段階か途中の道(証明の探索木のOR分枝)を飛び越して後戻りをする。IBの問題は、飛び越しにより刈り取られた道が果して解へ通ずる可能性のない道であったか否かという点にある。もし刈り取られた道の中に解へ通ずる道があれば、我々は解を見逃す危険性がある。刈り取った道が実際解へ通ずる可能性のない後戻り法をsafeであると言おう。IBに関する研究は既に幾つかあるが、効率的後戻りだけでなくsafeな

後戻りにも目配りをした研究は見当らないようである。本論文では，まず証明の探索空間（探索木）を定め，その上で解の見落しのない safe な知的後戻り法を研究する。

2. 証明の探索木

IB の厳密な議論の為には，まず与えられたプログラム S（horn set，とりあえず 'not'，cut は入っていないものと仮定する）と AND goal に対して，Prolog がどのように振舞うかを記述する必要がある。我々は Prolog の計算過程＝証明過程を AND goal の推移として把握する。計算過程に於ける AND goal は，AND goal の型（template）と代入（substitution，binding）の2項組みにより $<$型，代入$>$の形に常に表わされる。（1節の例を参照のこと）以下 L，M，N，… は goal（リテラル）を，$\alpha, \beta, \gamma, \cdots$ は goal の列，即ち AND goal を，$\theta, \lambda, \mu, \cdots$ は代入を，$E\theta$ は表現 E に代入 θ を適用した結果を示す。

一般的に AND goal の変化を追おう。まず最初の AND goal は与えられた AND goal を α として $<\alpha, \varepsilon>$（ε は null の代入，即ち $\alpha\varepsilon=\alpha$）の形に表わされる。今証明を開始して $i-1$ 回導出を実行した時の AND goal を $<L\alpha, \theta_1\cdots\theta_{i-1}>$ としよう。実際の AND goal は $L\alpha\theta_1\cdots\theta_{i-1}$ であり，次に解くべき goal は $L\theta_1\cdots\theta_i$ である。（goal は左から右へ解かれる）i 回目の導出に $L\theta_1\cdots\theta_{i-1}$ と単一化可能な head M を持つ入力節 $M\leftarrow\beta$ を選んだとする。入力節 $M\leftarrow\beta$ は変数の名前替え（renaming）をしておくものとする。$L\theta_1\cdots\theta_{i-1}$ と M の mgu（most general unifier）を θ_i とおくと i 回目の導出が終了した時の AND goal は $<\beta\alpha, \theta_1\cdots\theta_{i-1}\cdot\theta_i>$ となる。$\theta_1, \cdots, \theta_i$ を代入因子と呼ぶ。最後に AND goal の型が空になり，AND goal の状態が $<\square, \theta>$ となれば証明は成功裡に終了し，答（answer substitution）は θ である。

以上のような証明過程は下図のような（横書きの）証明の探索木の探索過程に対応する。

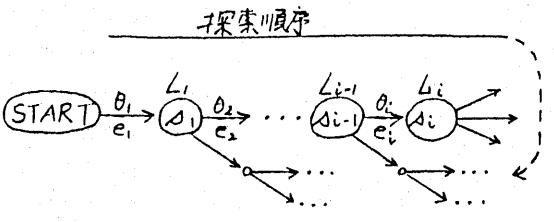


図2. 証明の探索木

各 OR 節点 A_i は AND goal を表わし，その時点で解かれるべき最左端の goal 型でラベル付けされている。節点から出ている有向辺 e_i は導出すべきリテラル pair，即ち goal と入力節の head を指定し，θ_i は e_i を実行する事により得られる mgu を表わしている。左の図で節点 A_i から出ているみつの辺は導出の可能性がある通りあることを表わしている。このような探索木は入力節の集合 S（prolog プログラム）と解くべき AND goal が与えられた時点でそれらの結合グラフ（connection graph）を作り，AND goal を頂点とする木構造に展開することにより得られる。探索は木を左から右へ，上から下へ辿ることにより行なわれ，辺を渡る時に導出し（単一化）を行う。導出が失敗した時は辺に X 印を付けることにする。

AND goalが空である状態に達すれば証明は（他の解を求める場合を除いて）終わる。その節点を空節（null clause）の印□でラベル付けする。探索の開始節点から□でラベル付けされた節点迄の道（path）を解の道と呼ぶ。

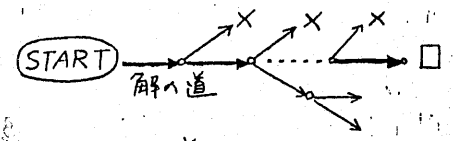


図3. 解の道

さて，ある節点Aを通る解がないことが分かった時後戻りをして他の道を探さなければいけない。次の節から，その後戻りを知的且つdateに行う方法を（後戻りなしに）探索する。なお'木'に関する用語，親節点，子節点，祖先，子孫等々は断わりなしに使う。

3. 幾つかの定義

def. 1

導入点：ある節点Aで導出されるgoal型Lについて，Lが探索を開始して以来初めてAND goalの型の中に現われた節点A'をLの導入点と言い，A'をintro(L)で表わす。

導入点に関し次の2つの事実が成立する。

(F1) Lの導入点を通る解の道の上には必ずLでラベル付けされた節点がある。これは一旦AND goalに導入されたgoal（型）は，そのAND goalが達成される迄に導出により除去されなければならないことから明らかである。

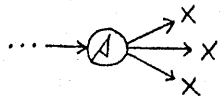
(F2) Lの導入点を祖先に持つLでラベル付けされた節点t, t'…を頂点とする部分探索木はすべて同一である。（modulo renaming）これは探索木はAND goalの型のみにより決まることから明らか。intro(L)に於けるAND goalを$\langle \beta L\alpha, \theta_1 \circ \cdots \circ \theta_i \rangle$とすると，t, t'…はすべて$\langle L\alpha, \theta_1 \circ \cdots \circ \theta_i \circ \nu \rangle$（νは幾つかの代入因子の合成で，t, t'…によって表わる）という形である。

def. 2

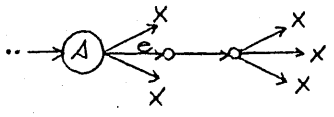
辺の失敗：ある節点から出ている辺eの指示する導出（単一化）が成功しなかった時，辺eは直接失敗したと言い，eの指示する導出が一応成功して次の節点に移ったものの，その後の探索によりその節点を通る解がない事が分かった時，eは間接的に失敗したと言う。

def. 3

節点の失敗：節点Aから出ているすべての辺が直接失敗した時，Aは直接失敗したと言い，Aから出ているある辺が間接的に失敗し，且つ他の辺が直接又は間接に失敗した時Aは間接的に失敗したと言う。証明の探索を開始してから最初に後戻りの必要が生じるのは節点の直接の失敗が起きた時である。図4に節点の失敗の状況を掲げておく。



(節点Aの直接的失敗)



辺eは間接的に失敗している。
(節点Aの間接的失敗)

図4. 節点の失敗

4. 節点が直接失敗した時の知的後戻り

先にも述べたように，最初の後戻りは節点が直接失敗した時に起こる。そこでまずこのような場合の知的且つsafeな後戻り法を考える。

今節点Aから出ているn個の辺$e_1, \cdots, e_n$がすべて直接失敗したとする。e_iを実行する際UA (Unification Algorithm，単一化アルゴリズム)が参照した代入因子を$\{\lambda_1, \cdots, \lambda_k\}$ $(k \geq 0)$とする。これを辺e_iの直接の失敗のagentと呼びagent(e_i)と記す。(以後agentを上のように記す時は$\lambda_1, \cdots, \lambda_k$の順に生成されたものと了解する)

def. 4

辺eの直接の失敗のagent：eが指示する単一化の際UAが参照した代入因子の全体

さてもとに戻り，辺e_iが直接失敗した時，失敗の原因は過去の単一化にあり，過去のある時点で辺e_iの直接の失敗の運命が定っている場合がある。ではe_iの失敗はどの時点で定ったのだろうか？

def. 5

代入因子の集合$\{\lambda_1, \cdots, \lambda_k\}$ $(k \geq 0)$を含む最小prefixとその出現点：証明の探索を開始して以来AND goalの代入に初めて$\{\lambda_1, \cdots, \lambda_k\}$が揃った節点Aの代入を言う。Aを$\{\lambda_1, \cdots, \lambda_k\}$を含む最小prefixの出現点と言う。$k>0$ならAは$\langle \cdots, \theta_1 \cdots \lambda_k \rangle$の形，$k=0$ならAは探索の開始節点である。

def. 6

goal型Lでラベル付けされた節点Aから出ている辺eの直接の失敗の運命点 det(e)：辺eの直接の失敗のagent，agent(e)を含む最小prefixνの出現点とLの導入点intro(L)の内most recent (探索木の頂点から遠い方)の節点をeの失敗の運命点と言いdet(e)と記す。det(e)は$\langle \cdots L \cdots, \nu \circ \nu' \rangle$ (ν'は或る代入)という形をしている。

辺e_iの直接の失敗の運命はdet(e)で既に定っている。理由は次の通り。辺e_iの指示する単一化をLとする入力節のhead Mとの単一化とする。agent(e) $=\{\lambda_1, \cdots, \lambda_k\}$を含む最小prefixを$\nu$とする。$e_i$が直接失敗していることは，$L\lambda_1 \cdots \lambda_k$とMが単一化不能，従って$L\nu$とMが単一化不能であることを意味する。辺$e_i$の失敗の運命点det(e)又はその子孫から探索を行うと(F1)(F2)により，再度辺e_iを渡ることになり，しかもその時のAND-goalは$\langle L\alpha, \nu\circ\nu' \rangle$という形である。辺eは$L\nu\circ\nu'$とMの単一化を指示しているが，これは$L\nu$とMが単一化不能であることにより必ず失敗に終る。

以上の考察より節点Aの直接の失敗の運命が定った時点が定まる。

def. 7

節点 A の失敗の運命点 $det(A)$：A から出ている辺を $e_1, \cdots, e_n$ とする。それらはすべて失敗に終っている。$det(A) = \{det(e_1), \cdots, det(e_n)\}$ の内 most recent な節点

$det(A)$ またはその子孫の節点を通る解の道はない。何故なら (F1)(F2) より $det(A)$ またはその子孫の節点から探索を開始するとやがて（途中で失敗がなければ）$\square$ でラベル付けされた節点に至り，$e_1 \cdots e_n$ のいずれかを渡ろうとするが，$det(A)$ の定義から，どの辺の指示する単一化も失敗に終わるからである。

$det(A)$ の1つ前（親節点）に戻れば，少なくともある e_i の失敗の原因の1つを取り除いたことになるので，探索を再開すると成功する可能性がある。従って，

def. 8

失敗した節点 A の後戻り先 $btk(A)$：節点 A が失敗した時，その失敗の運命点を $det(A)$ とする。$det(A)$ の親節点が A から後戻りすべき節点であり，$btk(A)$ と記す。

次に節点 A が間接的に失敗した時の"知的後戻り"を考えるがその前に定義と幾つかの結果を述べる。

def. 9

失敗した節点 A，及びその失敗の運命点 $det(A)$ の失敗の agent：節点 A から辺 $e_1, \cdots, e_n$ が出ていてすべて失敗したとする。A，及び $det(A)$ の失敗の agent を $agent(A) = agent(det(A)) = agent(e_1) \cup \cdots agent(e_n)$ により定義する。

明らかに $det(A)$ は $agent(A)$ を含む最小 pretix の出現点と A のラベル L の導入点 $intro(L)$ の内の most recent な節点である。今迄の考察により，直接失敗した節点 A，及びその失敗の運命点 $det(A)$（これもやはり失敗している）に関し，

(F3) 失敗した節点 $A = \langle \alpha, V \rangle$ の失敗の agent を $\{\lambda_1, \cdots, \lambda_k\}$ $(k \geq 0)$ とおく。もし他の節点 A' が $\langle \alpha, V' \rangle$，但し V' は $\{\lambda_1, \cdots, \lambda_k\}$ を含む，と書けるならば A' を通る解の道はない。

(F4) 失敗した節点 A の失敗の運命点を $det(A)$，$btk(A)$ を $det(A)$ の1つ前の A からの後戻り先とする。$btk(A)$ から $det(A)$ に移る際得られた代入因子入が A の失敗の agent ($= det(A)$ の失敗の agent) に入っているならば，入はその agent の中で most recent に得られた代入因子であり，$det(A)$ の代入 V は，V' を適当な代入として，$V = V' \circ \lambda$ と書ける。

図5に節点 A の直接の失敗と $det(A)$，$btk(A)$ の関係を図示する。

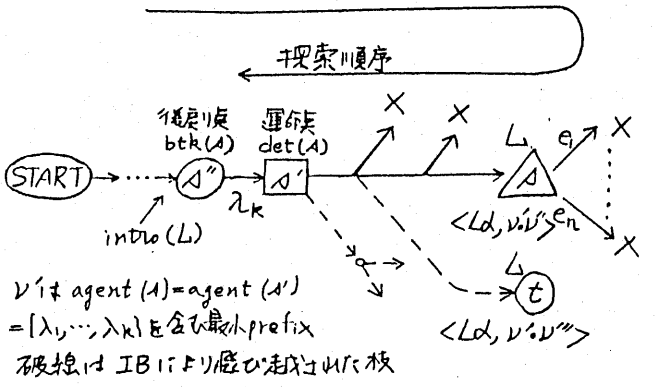


図5. 節点Aとdet(A), btk(A)の関係

5. 節点が間接的に失敗した場合の知的後戻り

紙面の都合上以後簡略に述べる。我々は節点Aが間接的に失敗するに至る迄の，直接又は間接の失敗，その運命矢，その失敗のagent，後戻り矢について(F3)，(F4)が成立していることを仮定する。明らかに(F3)(F4)は最初の節点の失敗し直接の失敗である)について成立している。

まず間接的に失敗した迄の失敗のagent及び失敗の運命矢を定義する。

def. 10

goal型Lでラベルはけされた節点Aから出ている間接的に失敗した迄eの失敗のagent，agent(e)，及び失敗の運命矢det(e)：

eはAから出てA'に入っているとする。節点Aで後戻りが必要となるとすれば，Aはある失敗の後戻り矢で，A'はその失敗の運命矢である。従って我々は節点Aから後戻りする状況に関心があるのだから，A，A'を今迄述べたような節点であると考えることにする。従ってA'についてはその失敗のagentが定義され，(F3)(F4)が成立している。eを実行することにより得られた代入因子をθとする。($A \xrightarrow{e/\theta} A'$)

(i) agent(A') = φ の時。この時はA'のAND goalの型のみがeの失敗の原因である。agent(e) = φ, det(e) = intro(L) と定める。

(ii) agent(A') ≠ φ 且つ agent(A') ∋ θ の時。eの指示する単一化が生み出した代入因子θはA'の失敗に寄与していない。agent(e) = agent(A'), det(e) = agent(A')を含む最小prefixの出現点とintro(L)の内most recentな節点と定める。

(iii) agent(A') ≠ φ 且つ agent(A') ∌ θ の時。agent(A') = {λ1,…,λk}とおく。λ1,…,λkはこの順に作られ，λkが最も最近に作られた代入因子であるとする。(F4)よりθ = λkである。θを作る際にAが参照した代入因子を{μ1,…,μl} (l ≥ 0) とおく。
agent(e) = {λ1,…,λk-1} ∪ {μ1,…,μl}
det(e) = agent(e)を含む最小prefixの出現点とintro(L)の内most recentな方の節点

この定義により与えられるeの失敗の運命矢det(e)又はその子孫の節点から証明の探索を開始した場合，たとえeに達したとしてもeは再度失敗する運命

にある。(iii)の場合それを確かめて見よう。左の図で det(e) またはその子孫 A'' に戻り，他の道（下の道）を選んで辿ったとする。図で ν' は A の失敗の agent {λ1,…,λk} との内 {λ1,…,λk-1} ，及び λk を得る際 UA が参照した代入因子 {μ1,…,μl} をすべて含んでいる。λk は ⟨ν'⟩ とある λ カ節 M←β の head M の mgu である。辺以下の道を辿って行くと [F1] [F2] により図の節点 t の状態に達っする。辺 e は ⟨ν'ν''⟩ と M の単一化を指示している。もし両者の mgu があればそれを δ とおく。⟨ν'ν''⟩δ = Mδ = Mν''δ（M は rename されている）より，ν''δ は ⟨ν'⟩ と M の単一化子である。従って或る代入 δ' に対し ν''δ = λk·δ' と書ける。それ故節点 t から辺 e を渡った先の節点 t' の代入は ν'ν''δ = ν'λk·δ' となり，節点 A' の失敗の agent {λ1,…,λk} をすべて含んでいるので [F3] より t' を通る解はない。従って辺 e を通る解はない。

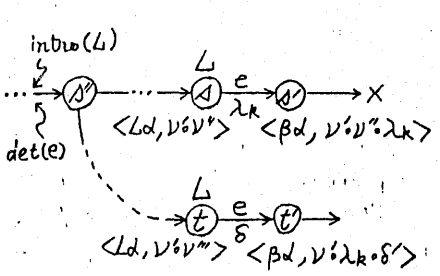


図6. 辺eの間接的失敗の状況

間接的に失敗した辺の失敗の agent，運命矣が定まった。直接失敗した辺の失敗の agent は def. 4 で，運命矣は def. 6 で既に与えられている。これで間接的に失敗した節点 A から出ているすべての辺について，失敗の agent，運命矣が与えられた。今迄の議論から，A の失敗の運命矣 det(A) は def. 7 で，A からの後戻り先 btk(A) は def. 8 で，A 及び det(A) の失敗の agent は def. 9 で与えられることは明らかである。又間接的に失敗した節点 A，及び det(A)，について [F3] が，btk(A)，det(A) について [F4] が再び成立していることも明らかである。

6. 知的後戻り法の具現化 (implementation) に当っての注意点

我々は第4，5節で知的後戻りの基本 algorithm を与えた。実際への応用に際しては幾りかの吟味が必要になる場合がある：

a) 偽の失敗：'cut' による後戻りを失敗の原因の1つとした節点 A の失敗，及び，他の解を求める為の強制的失敗を原因の1つとした節点 A の失敗を偽の失敗と呼ぶ。偽の失敗は単一化の失敗によるものでないので，今迄の議論が適用できない。A の失敗の原因の1つに偽の失敗がある時は，A から NB により後戻りをせざるを得ない。（詳細は省略）

b) 同一の失敗の防止：ここで述べた"知的後戻り"は失敗の原因の1つを取り除いた地点に戻るに過ぎない。従って複数の辺 e1，e2 の失敗が起きて後戻りした時，再度一方の辺を渡ろうとして同じ失敗を繰り返す危険性がある。このようなことを防ぐには，辺 e が失敗した時，失敗の運命矣 det(e) より前に戻らない限り，辺 e を渡らないようにすればよい。（詳細は省略）

c) 一直線の探索木： 探索木が …→○→○→…→○→Ⓐ の形で，Aが失敗した場合，単純にここで述べた"知的後戻り法"を適用すると，一歩一歩後戻りをすることになり能率が悪い。 まとめて"知的後戻り"を行う方法があるが，紙面の都合上省略する。

d) 具現化： 記録しておく必要のあるものは，① goal型Lの導入及し推論ステップ)，② 失敗した辺eの失敗のagentのリスト，③ 成功した辺eの生成した代入因子Bと，Bを作る際UAが参照した代入因子のリストである。 Prologが構造共有(structure sharing)方式で作られている場合，変数セルに値がbindされた推論ステップの番号を付けておく。 例えば，
| x | n | a |
は変数xの値aが第nステップで得られたということを示す。

e) 簡単化： "知的後戻り"を節点が直接失敗した時だけに限れば，記録しておく必要のあるものは直接の失敗のagentの内most recentなものだけである。 又偶々失敗も気にしなくともよい。

d) 一般化： ここで述べた"知的後戻り"の議論の枠組みをそのまま使って，OL反証法(reductionが付け加わる)や，リテラルの選択を自由にしたOL反証法(completeである。AND goalの型が句構造になる)に於ける知的後戻りのalgorithmを導び出すことができる。

7. 結論

Prolog プログラムとそれらしいAND goalから作られる結合グラフを展開した証明の探索木をもとに，知的且つ解の見落しのない効率的後戻り法を述べた。我々の後戻り法は，成功の可能性がある地点に戻るものであり，成功の可能性が高い地点に戻るのではない。 その意味で最低限の"知的後戻り法"である。成功の見込み近者慮に入れた，解の見落しのない効率的後戻り法が次の課題である。 それにはプログラムの変数間の依存関係のfoward analysisが必要とされるであろう。 最後に日頃討論して下さった田中室長はじめ推論機構研究室の皆様に感謝致します。

参考文献：

[1] Bruynooghe, M. "Analysis of Dependencies to Improve the Behavior of Logic Programs", 5th conf. on Automated Deduction, Lec. Note in Comp. Sci. Springer, 19

[2] Chang, C.L. & Lee, R.C.T. "Symbolic Logic & Mechanical Theorem Proving", Academic Press, 1973.

[3] Pereira, L.M. & Porto, A. "Selective Backtracking for Logic Programs", 5th conf. on Automated Deduction, Lec. Note in Comp. Sci. Springer, 1980.

[4] Lasserre, C. & Gallaire, H. "Controlling Backtrack in Horn Clauses Programming", ACM Logic Programming Workshop, Budapest, 1980.