文字列領域の問題解決における一階論理表現からのプログラム生成

吉田忠行[†] 赤間清[††] 宮本衛市[†]

問題解決において，正しく，かつ効率的に問題を解くプログラムを作成するのは重要なことであるが，同時に困難なことでもある．本論文では，正しく効率的なプログラムを自動的に作成するための新しい方法を提案する．本論文では，文字列領域における問題の1つとして言語認識問題を取り上げる．一階述語論理表現を用いて定式化した問題仕様から，問題を高速に解くためのプログラムを生成する．ただし，本方法によって生成されるプログラムは，有限オートマトンに相当する単純なプログラムに限定している．本論文では，問題仕様とプログラムを従来の理論よりも一般的な形で関係づけている，等価変換に基づく計算モデル（等価変換モデル）を採用する．等価変換モデルでは，プログラムは等価変換ルールの集合である．本論文のプログラム生成では，問題仕様から等価変換ルールを次々に生成し，その中から効率的なルールだけを選別する方法を用いる．ルール生成は「メタ計算」と呼ばれる計算によって行い，効率的なルールの評価基準に基づき，生成されたルールを選別する．本論文の提案する方法により，一階述語論理表現を用いて自然な形で記述された言語仕様から，その言語に関する認識問題を効率的に解くプログラムを得ることが可能になる．

Program Generation from First-order Logical Expressions for Problem Solving on a String Domain

TADAYUKI YOSHIDA,[†] KIYOSHI AKAMA[††] and EIICHI MIYAMOTO[†]

It is very important but not easy to write correct and efficient programs for solving problems. In this paper, we propose a new method for automatic generation of correct and efficient programs. We assume that we are given a language recognition problem that is formalized by using first-order logical expressions. Correct and efficient programs are then generated from the problem specification. The class of programs generated by the proposed method correspond to the one of finite automatons. In this paper we adopt a computation model based on equivalent transformation (ET model), which formalizes the relation between specifications and programs more generally than other existing models. In the ET model, a program consists of equivalent transformation rules (ET rules). We generate many ET rules from a given specification represented by first-order logical expressions and select "efficient" rules among them. We generate ET rules by "meta computation" and select efficient rules based on an evaluation function. We can obtain correct and efficient programs to solve language recognition problems from their language specifications naturally represented by first-order logical expressions.

1. まえがき

問題解決において，正しく，かつ効率的なプログラムを作成するのは重要なことである．しかし，そのようなプログラムを作成するのは，多くの場合，容易ではない．本論文では，正しく，かつ効率的なプログラムを作成するために，プログラムを理論に基づいて自動生成することを試みる．具体的には，文字列領域における問題のうち言語認識問題を取り上げ，一階述語論理表現を用いて定式化し問題仕様とする．その問題仕様から，問題を効率的に解くためのプログラムを生成する．このプログラムは，有限オートマトンに相当する単純なものである．

本論文では，問題仕様とプログラムを明確に，かつ強力に結び付けるために，等価変換に基づく計算モデル（等価変換モデル）を採用する．等価変換モデルでは，プログラムは等価変換ルールの集合である．したがって，ルールの良し悪しがプログラムの良し悪しを

† 北海道大学大学院システム情報工学専攻
Division of System and Information Engineering, Hokkaido University
†† 北海道大学情報メディア教育研究総合センター
Center of Information and Multimedia Studies, Hokkaido University

決定する．本論文では，一階述語論理表現による問題仕様から等価変換ルールを生成し，その中から本論文における「効率的なルール」を選別する方法を用いる．問題仕様から等価変換ルールを生成するために，「メタ計算」と呼ばれる計算を行う．また，効率的なルールを明確に判定する評価基準を設定する．プログラム生成システムは，次々に生成されたルールの中からその評価基準を満たすルールの集合を出力する．

一階述語論理表現からプログラムを生成する枠組みとしては，論理プログラムに基づく方法も存在する[1]．しかし，論理パラダイムにおけるプログラムは，確定節集合などの論理式であり，アルゴリズムの表現力に乏しく，効率的なプログラムを自然に記述することに限界がある．それに対して等価変換パラダイムでは，手続きは表現力の豊かな等価変換ルールで記述され，広範なアルゴリズムを自然に表現することができる．

また，特に一階述語論理式を扱うときには，論理パラダイムでは，計算方法が，一階述語論理式を，標準形に変換するフェーズとその変換された標準形をSLDNF導出に基づいて推論するという2つのフェーズに分離している[2]など，計算の枠組みが統一的ではない．一方，等価変換の枠組みでは，「計算＝等価変換」という統一した原理がある．このため，計算に関してすべて単一の原理で行うことができ，プログラム生成の研究も見通し良く行えることが期待できる．

等価変換モデルではすでに，確定節から等価変換ルールを生成する理論的な基礎が与えられている[3]．本論文では，その理論を基にして，文字列領域をターゲットとして，表現レベルを確定節から一階述語論理表現を用いることができるところまで拡大する．これは，等価変換の理論においてこれまで並行して行われていた，一階述語論理表現の扱いに関する研究と，純粋な確定節レベルの仕様から等価変換ルールを生成する研究を結び付けるものである．

2. 等価変換による問題解決

2.1 言語認識プログラム作成問題

プログラム作成問題の例として，次の問題を考える．

> Σ をアルファベット $\{a,b,c\}$ とする．Σ 上の言語 L は，次の条件を満たす Σ 上の任意の文字列 δ の集合と定義する．
>
> > δ がもし aa という部分列を含むならば，その aa の右には必ず ba が出現する．
>
> Σ 上の任意の文字列が与えられたとき，それが言語 L に属するかどうかを判定するプログラムを作成せよ．

本論文では，この問題を例として，文字列領域でのプログラム生成の方法を議論する．

2.2 宣言的記述による問題の定式化

$s(x)$ は「x は言語 L の正しい文字列である」を意味するアトムとする．

任意の Σ 上の文字列 δ が与えられたとき，$s(\delta)$ が成り立つか否かを判定する問題は，

$$P = \{\, s(X) \leftarrow \forall\alpha\forall\beta\,([X = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma]).\}$$

$$P' = P \cup \{yes \leftarrow s(\delta)\}$$

として，

$yes \in \mathcal{M}(P')$☆

か否かを判定する問題として定式化できる．ただし，$[X = \alpha aa\beta]$ や $[\beta = ba\gamma]$ は文字列領域上の等式制約，yes は述語である．

2.3 等価変換による問題の解決

P' から P'' に問題を等価変換するとは，$\mathcal{M}(P') = \mathcal{M}(P'')$ を満たす条件のもとで P' を変形し P'' を得ることである．

$yes \in \mathcal{M}(P')$ によって定式化した言語認識問題は，等価変換により次のように解くことができる．

P' を等価変換して，

$$P'' = P \cup \{yes \leftarrow .\}$$

に変換できれば，$yes \in \mathcal{M}(P'')$ と，$\mathcal{M}(P') = \mathcal{M}(P'')$ より，$yes \in \mathcal{M}(P')$ が成り立ち，答えは yes となる．一方，

$$P'' = P$$

となった場合には，$yes \notin \mathcal{M}(P'')$ であるから，$yes \notin \mathcal{M}(P')$ となり，答えは no である．

2.4 等価変換パラダイムにおけるプログラム

等価変換パラダイムにおけるプログラムとは，**等価変換ルール**と**制御**（**優先度記述**）により，構成される．等価変換ルールは，対象となるアトムや制約などを簡単化する書き換えルールであり，制御とは，各ルールにそれぞれ適用の優先度を与えることである．ただし，本論文の例題では，制御に関しては特に言及する必要がないため，以降では，制御の話題は取り上げない．

たとえば，言語認識問題を解くプログラムは，次のような等価変換ルール（とそれらを適用する制御）から構成することができる．

- $s(X)$ を展開するルール
- 等式制約のルール
- 一階論理制約[4]のルール

☆ $\mathcal{M}(P)$ の定義は，付録 A.1 を参照．

- 負制約[4]のルール

このルールから構成されるプログラムを用いれば，文字列 $caaba$ が L に属するかどうか，すなわち $yes \in \mathcal{M}(P')$ かどうかは，以下のような過程で計算される．

p1: $yes \leftarrow s(caaba)$
p2: $yes \leftarrow \forall\alpha,\beta([caaba = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])$
p3: $yes \leftarrow \neg\exists\alpha,\beta\neg([caaba = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])$
p4: $yes \leftarrow [q \notin \mathcal{M}(Q)]$
$Q = \{q \leftarrow \exists\alpha,\beta\neg([caaba = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])\}$
p5: $yes \leftarrow [q \notin \mathcal{M}(Q)]$
$Q = \{q \leftarrow \neg([caaba = YaaZ] \Rightarrow \exists\gamma[Z = ba\gamma])\}$
p6: $yes \leftarrow [q \notin \mathcal{M}(Q)]$
$Q = \{q \leftarrow [caaba = YaaZ], \neg\exists\gamma[Z = ba\gamma]\}$
p7: $yes \leftarrow [q \notin \mathcal{M}(Q)]$
$Q = \{q \leftarrow \neg\exists\gamma[ba = ba\gamma]\}$
p8: $yes \leftarrow [q \notin \mathcal{M}(Q)]$
$Q = \{q \leftarrow [r \notin \mathcal{M}(R)]\}$
$R = \{r \leftarrow \exists\gamma[ba = ba\gamma]\}$
p9: $yes \leftarrow [q \notin \mathcal{M}(Q)]$
$Q = \{q \leftarrow [r \notin \mathcal{M}(R)]\}$
$R = \{r \leftarrow [ba = baU]\}$
p10: $yes \leftarrow [q \notin \mathcal{M}(Q)]$
$Q = \{q \leftarrow [r \notin \mathcal{M}(R)]\}$
$R = \{r \leftarrow\}$
p11: $yes \leftarrow [q \notin \mathcal{M}(Q)]$
$Q = \{\}$
p12: $yes \leftarrow$

これにより，$yes \in \mathcal{M}(P')$ であることが分かり，文字列 $caaba$ は L に属すると結論できる．

この変換過程の各ステップ簡単な説明を以下に示す．

p1 → p2	$s(X)$ の unfold 変換
p2 → p3	$\forall$ の除去
p3 → p4	負制約に書き換え
p4 → p5	$\exists$ の除去
p5 → p6	$\Rightarrow$ の否定の除去
p6 → p7	等式制約の解消（$Y/c, Z/ba$）
p7 → p8	負制約に書き換え
p8 → p9	$\exists$ の除去
p9 → p10	等式制約の解消
p10 → p11	単位節を取り出す
p11 → p12	負制約の解消

3. 本論文の問題設定

3.1 効率的なプログラムの生成

前節のプログラムは，Σ 上の任意の文字列が L に属するか否かを判定することができる．このプログラムは，L に依存しない変換ルールにより構成されており汎用性が高いが，効率という点では問題がある．

そこで，問題仕様として，一階述語論理表現を含む記述を採用し，そこから “効率的な” プログラムを生成する方法を与えるのが，本論文の目的である．

3.2 対象とする問題クラス

本論文では，プログラム生成の対象クラスを任意のアルファベット Σ と，一階論理制約 $FLC(X)$ のペア全体の集合とする．また，$\forall X\, s(X) \Leftrightarrow FLC(X)$ によって定義された述語を s とする．Σ 上の文字列 δ が任意に与えられたとき，$s(\delta)$ が成立するか否かを判定する効率的なプログラムを生成するのが，本論文の問題である．

プログラムに効率性や停止性を保証するために，生成されるプログラムは，以下のような条件を満たす等価変換ルールから構成されるものに限定する．

条件 1 $atom$, $atom'$ をメタアトム（5.1 節参照）として，ルールの形は，
- $atom \rightarrow \langle true\epsilon\rangle$.
- $atom \rightarrow \langle false\epsilon\rangle$.
- $atom \rightarrow atom'$.

のどれかになっていなければならない．

条件 2 ルールの適用前後において，メタアトムのサイズ（メタアトムにおける変数と定数の総出現数）は減少していなければならない．

条件 3 生成されるルールの左辺のメタアトムの引数パターンは，全体で Σ 上の文字列をすべてカバーしていなければならない．

条件 4 生成されるルールの左辺のメタアトムの引数パターンは，互いに素になっていなければならない．こうすることによって，同一のアトムに対して，複数のルールが選択されることはなくなる．この結果，生成されるプログラムは，制御（ルールの優先度）に依存しないものとなる．

3.3 例題において生成されるプログラム

前章で取り上げた例題は，

$\Sigma = \{a, b, c\}$

$FLC(X) = \forall\alpha\forall\beta\,([X = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])$

とした問題である．

このようにして与えた問題仕様から，本論文の方法では次のようなプログラム（等価変換ルールの集合）

が生成される．$\&X$ は任意の文字列を表す変数（&変数）である（5.1 節参照）．このプログラムは，全体として言語 L を認識するオートマトンに相当するアルゴリズムを実現している．

r1: $s(\epsilon) \rightarrow \langle true \rangle$
r2: $s(b\&X) \rightarrow s(\&X)$
r3: $s(c\&X) \rightarrow s(\&X)$
r4: $s(a) \rightarrow \langle true \rangle$
r5: $s(ab\&X) \rightarrow s(\&X)$
r6: $s(ac\&X) \rightarrow s(\&X)$
r7: $s(aa) \rightarrow \langle false \rangle$
r8: $s(aab) \rightarrow \langle false \rangle$
r9: $s(aac\&X) \rightarrow \langle false \rangle$
r10: $s(aaa\&X) \rightarrow \langle false \rangle$
r11: $s(aabb\&X) \rightarrow \langle false \rangle$
r12: $s(aabc\&X) \rightarrow \langle false \rangle$
r13: $s(aaba\&X) \rightarrow s(a\&X)$

ルールの解釈は，次のとおりである．

たとえば，ルール r4 は，$s(a)$ というアトムがボディにあれば，それを除去してよいことを表している．ルール r7 は，$s(aa)$ というアトムがボディにあれば，そのアトムを含む節を除去してよいことを表している．ルール r13 は，$s(aaba\&X)$ という形のアトムがあれば，それを $s(a\&X)$ という形のアトムに書き換えてよいことを表している．それ以外のルールについても同様である．

これらのルールは，前節で定めた条件（条件 1〜条件 4）をすべて満たしている．

4. 等価変換によるプログラム生成の方法

4.1 プログラム生成の要素

本論文におけるプログラム生成の枠組みは，次のような要素から成り立っている．

(1) メタルールの準備
　メタ節を変換するためのメタルールを用意する．
(2) ルール生成ルーチン *generate*
　メタルールを用いて変換を行い，ルールを出力する．
(3) 評価基準 *eval*
　ルールの評価値を与える写像である．
(4) メタルール生成ルーチン *genMR*
　ルール生成時に，ルールからメタルールを作る．
(5) アトムパターン集合の更新ルーチン *specialize*
　ルール生成ルーチンに与えるアトムパターンの集合を更新する．

以降の節では，これらの各要素について説明し，最後にプログラム生成システムを提案する．

4.2 メタルールの準備

メタ節を計算するメタルールをあらかじめ用意する．これらのメタルール群を MR_0 とする．この MR_0 には，以下の種類のメタルールが含まれる．

(1) 宣言的記述 P から得られるメタルール
(2) 等式制約を処理するメタルール
(3) 一階論理制約を処理するメタルール
(4) 負制約を処理するメタルール

4.3 ルール生成ルーチン（*generate*）

等価変換に基づく方法では，ルールを生成するには，

(1) 対象となるアトムのパターン p
(2) ルール生成の計算を行うメタルールの集合 MR

が必要である[3]．

これらを用いて，ルール r が生成されることを，

$$generate(p, MR) = r$$

と表す．また，ルール生成が正常に行われなかったとき（無限ループに陥るなどして，あらかじめ定めた時間内に停止しない場合など）は，null を返すものとする．

4.4 評価基準（*eval*）

アトムパターンとメタルールによって 1 つのルールが生成される．しかし，そのルールが十分効率的であるか否かまでは，保証できない．そこで，生成されたルールに対して，実際に出力するルール群に追加するかどうかを決める「評価基準」が必要になる．

本論文の例では，ルールが 3.2 節で定めた条件 1 と条件 2 に合致したとき，

$$eval(r) = T$$

とし，それ以外のときは，

$$eval(r) = F$$

とする．

4.5 メタルール生成ルーチン（*genMR*）

生成されたルールが評価基準を満たせば，それは最終的に出力されるルール群に加えられる．一方，評価基準を満たさなかったルールは，破棄されるのではなく，**メタルール** に作り直してメタルール群に追加することにより，それ以降のルール生成ルーチンで用いることができる．

genMR は，等価変換ルールに対して，

$$\begin{cases} \&X \Rightarrow *X \\ \#X \Rightarrow \%X \end{cases}$$

という変数の置き換えをして新しいメタルールを得る

操作である．

たとえば，アトムパターン $s(\&X)$ からは

$$r_a : s(\&X) \rightarrow \begin{cases} [q(\&X) \notin \mathcal{M}(Q)] \\ Q = \{q(\#Yaa\#Z) \leftarrow [\#Z \notin rep(ba\#U)]\} \end{cases}$$

というルールが生成される（付録 A.2 参照）．しかし，これら 2 つのルールは，上記の評価基準を満たさないので，出力するルールとしては採用できない．したがって，r_a から次のような 2 つのメタルールを作り，メタルールの集合に追加する．

$$s(*X) \rightarrow \begin{cases} [q(*X) \notin \mathcal{M}(Q)] \\ Q = \{q(\%Yaa\%Z) \leftarrow [\%Z \notin rep(ba\%U)]\} \end{cases}$$

$$\left.\begin{array}{l} [q(*X) \notin \mathcal{M}(Q)] \\ Q = \{q(\%Yaa\%Z) \leftarrow [\%Z \notin rep(ba\%U)]\} \\ \quad \rightarrow s(*X) \end{array}\right\}$$

これらをそれぞれ m_a, m_b とし，この変換操作を，

$$genMR(r_a) = \{m_a, m_b\}$$

と表記する．

4.6 アトムパターンの更新（*specialize*）

specialize は，引数となるアトムパターンの集合の各要素について，3.2 節で述べた条件 3，条件 4 を考慮したうえでパターンを特殊化する操作である．

本論文で取り上げた問題では，対象とするアトム $s(X)$ の引数のパターンは次のように順次生成される．ここでの $\&X$ などは &変数と呼ばれる変数である（5.1 節参照）．

たとえば，プログラム生成の途中で $s(aa\&X)$ というパターンに対するルールを生成し，それが前出の評価基準を満たさないとする．このときプログラム生成システムは，もう少し具体化されたパターンを作り直す必要がある．例題の言語 L は $\Sigma = \{a, b, c\}$ 上の文字列であるから，$s(aa\&X)$ の中の $\&X$ を空文字列，$a\&W$，$b\&W$，$c\&W$ にそれぞれ置き換える．この操作により，

$$s(aa),\ s(aaa\&W),\ s(aab\&W),\ s(aac\&W)$$

というパターンの集合が新たに生成される．これを，次のように表記する．

$$specialize(\{s(aa\&X)\}) = \{s(aa), s(aaa\&W), s(aab\&W), s(aac\&W)\}$$

4.7 プログラム生成システム

これまで定義した，*generate*，*eval*，*genMR*，*specialize* を用いて，本論文で対象としている問題クラスに対するプログラム生成システムを提案する．

ルールの集合，メタルールの集合，アトムパターンの集合をそれぞれ，GR，MR，Pat とする．

プログラム生成システムのアルゴリズムを疑似的な言語で記述する．

［プログラム生成アルゴリズム］

```
GR := ∅, MR := MR0, Pat := {s(&X)};
while(Pat ≠ ∅) {
  Pat' = Pat ;
  foreach p in Pat' {
    r := generate(p, MR);
    if (r ≠ null) {
      if (eval(r) = T) {
        GR := GR ∪ {r};
        Pat := Pat − {p};
      } else {
        MR := MR ∪ genMR(r);
      }
    }
  }
  Pat := specialize(Pat);
}
return GR;
```

まず，GR，MR，Pat をそれぞれ初期化する．そのあと，ターゲットとするアトムパターンがなくなるまで生成の処理を続ける（while ループ）．生成処理は，現在のパターン Pat を Pat' にコピーしたあと，Pat' の各要素（アトムパターン）p に対して，メタルール MR を使ってルールを 1 つ生成する．これを r とする．生成処理がうまくいかなかったとき（r =null）は次のパターンに移る．生成処理がうまくいったときは，そのルール r を評価して，基準を満たしていれば（$eval(r) = T$）r を GR に追加し，生成に用いたアトムパターン p を Pat から除去する．生成したルールが評価基準を満たさないとき（$eval(r) = F$）は，そのルールから作られたすべてのメタルール（$genMR(r)$）を，メタルール集合 MR に追加し，以降のルール生成処理に用いることができるようにする．Pat' のすべての要素を処理したら，その時点でのアトムパターンの集合 Pat を更新し（*specialize*），while ループを繰り返す．

生成するパターンがなくなったとき（$Pat = \emptyset$）は，すべてのパターンが生成できたので，その時点で得られているルール集合 GR を出力して終了する．

このシステムに本論文の例題を入力として与えたときの，GR および Pat の様子は，付録 A.6 で詳説する．

5. ルール生成部の核部分

5.1 メタ表現の導入

本節では，メタ表現における表記の方法[3]について解説する．

$\&A$ のように & がついている変数は「任意の文字列を代入できる」変数を意味し，&変数と呼ぶ．$\#W$ のように # がついている変数は「その節に出現しない任意の変数を代入できる」変数であり，#変数と呼ぶ．この &変数，#変数と Σ の元からなる文字列を「メタ文字列」と呼ぶ．たとえば，$\#Xaa\#Y$，$ab\&U$ などはメタ文字列である．また，メタ文字列を引数に持つアトムを「メタアトム」と呼ぶ．

5.2 メタ節

ルール生成は，「メタ節」と呼ばれる節を変換することによって行う．メタ節は，ヘッドが h のような引数を持たないダミーのアトムで，ボディがメタアトムから構成される

$$h \leftarrow s(aa\&X)$$

のような節である．

5.3 メタルール

等価変換によるルール生成では，メタ節を変換することによりルールを作り出す．メタ節を変換するルールを「メタルール」と呼ぶ．メタルールの表記の中で，$*X$ のように $*$ がついている変数は「任意のメタ文字列を代入できる」変数であり，$*$ 変数と呼ぶ．$\%W$ のように % がついている変数は「その節に出現しない任意の#変数を代入できる」変数であり，%変数と呼ぶ．

本論文の例で用いたメタルールの例を以下に示す．ただし，E，E_1，E_2 は一階論理制約とする．

m1: $s(*X)$ の定義節から導かれるルール

$$s(*X) \rightarrow \forall\alpha\forall\beta([*X = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma]).$$

m2: $\forall$ の除去 $(m \geq 1)$

$$\forall\alpha_1 \cdots \forall\alpha_m E(*X) \rightarrow \neg\exists\alpha_1 \cdots \exists\alpha_m \neg E(*X).$$

m3: $\exists$ の除去 $(m \geq 1)$

$$\exists\alpha_1 \cdots \exists\alpha_m E(*X)[\alpha_1, \cdots, \alpha_m] \rightarrow E(*X)[\%W_1, \cdots, \%W_m].$$

$[\%W_1, \cdots, \%W_m]$ は，対応する $[\alpha_1, \cdots, \alpha_m]$ をそれぞれ置き換えることを表す．

m4: 負制約に書き換え

$$\neg\exists\alpha_1 \cdots \exists\alpha_n E[*X_1, \cdots, *X_m] \rightarrow [q(*X_1, \cdots, *X_m) \notin \mathcal{M}(Q)].(m \geq 1)$$
$$Q = \{q(*X_1, \cdots, *X_m) \leftarrow \exists\alpha_1 \cdots \exists\alpha_n E[*X_1, \cdots, *X_m].\}$$

ただし，$[*X_1, \cdots, *X_m]$ は E の中に自由変数として $*X_1, \cdots, *X_m$ が含まれていることを示す．

m5: $\neg \Rightarrow$ の除去

$$\neg(E_1 \Rightarrow E_2) \rightarrow E_1, \neg E_2.$$

6. ルールの生成例

本章では，ルール生成ルーチン $generate$ の動作を説明する．紙面の都合上，最初に作られるルール r_a と

r4: $s(a) \rightarrow \langle true \rangle$

r10: $s(aaa\&X) \rightarrow \langle false \rangle$

r5: $s(ab\&X) \rightarrow s(\&X)$

という合計4つのルールの生成過程のみを示す．用意したメタルールの集合を MR とする．

6.1 ルール生成過程

6.1.1 生成例1

まず，$s(\&X)$ という最も一般的なパターンを与えた場合，メタルール群 MR を用いた計算で，メタ節 p1 は p12 に変換される（詳細は，付録 A.2 参照）．

p1: $h \leftarrow s(\&X)$

$\cdots$

p12: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Yaa\#Z) \leftarrow [\#Z \notin rep(ba\#U)]\}$

したがって P1 と P12 のボディより，

$$r_a : s(\&X) \rightarrow \begin{cases} h \leftarrow [q(\&X) \notin \mathcal{M}(Q)] \\ Q = \{q(\#Yaa\#Z) \leftarrow [\#Z \notin rep(ba\#U)]\} \end{cases}$$

という等価変換ルールが生成される．すなわち，

$$generate(s(\&X), MR) = r_a$$

である．

しかし，このルールは評価基準を満たさないので，このルールからメタルールを生成する（4.5 節参照）．この生成したメタルールを MR に追加して，MR' とする．以降では，MR' を用いてルール生成を行う．

6.1.2 ルールの生成例2

$s(a)$ というパターンを与えた場合，メタルール群 MR' を用いた計算で，メタ節 p1 は p8 に変換される（詳細は，付録 A.3 参照）．

p1: $h \leftarrow s(a)$

$\cdots$

p8: $h \leftarrow$

したがって p1 と p8 のボディより，

r4: $s(a) \rightarrow \langle true \rangle$

というルールが生成される．すなわち，

$$generate(s(a), MR') = r4$$

である．

6.1.3 ルールの生成 3

$s(aaa\&X)$ というパターンを与えた場合，メタルール群 MR' を用いた計算で，メタ節 p1 は p13 に変換される（詳細は，付録 A.4 参照）．

p1: $h \leftarrow s(aaa\&X)$

$\cdots$

p13: 節が消滅

したがって p1 と p13 のボディより，

r10: $s(aaa\&X) \rightarrow \langle false \epsilon \rangle$

というルールが生成される．すなわち，

$generate(s(aaa\&X), MR') = r10$

である．

6.1.4 ルールの生成 4

$s(ab\&X)$ というパターンを与えた場合，メタルール群 MR' を用いた計算で，メタ節 p1 は p14 に変換される（詳細は，付録 A.5 参照）．

p1: $h \leftarrow s(ab\&X)$

$\cdots$

p13: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Waa\#Z) \leftarrow [\#Z \notin rep(ba\#U)]\}$

p14: $h \leftarrow s(\&X)$

したがって p1 と p14 より，

r5: $s(ab\&X) \rightarrow s(\&X)$

というルールが生成される．すなわち，

$generate(s(ab\&X), MR') = r5$

である．

7. プログラム生成システムの特徴

この章では，本プログラム生成システムの特徴を，生成されるプログラムの性質（正当性，停止性，効率性），生成システム自体の停止性，および生成するプログラムの多様性に関して述べる．

7.1 生成されるプログラムの性質

7.1.1 正 当 性

生成されるプログラムは，等価変換ルールの集合として得られる．生成される個々の等価変換ルールの正当性は，文献 3) と類似の理論で保証することができる．したがって，プログラム全体の正当性も保証することができる．

7.1.2 停 止 性

生成されたプログラム全体の停止性は，次の 2 つの事実から容易に導かれる．

- 条件 4 により，つねにちょうど 1 つのルールだけが適用可能になる．しかも条件 3 により，計算の途中でルールが適用できなくなるようなことはない．
- 条件 2 により，アトムのサイズ（変数と定数の総出現数）は減少列（$\langle true \epsilon \rangle$ または $\langle false \epsilon \rangle$ によってアトムがなくなるときは，サイズは 0 とする）をなす．また，アトムのサイズは非負整数である．

したがって，基礎 s アトムは，ルールを有限回適用することにより，必ず $\langle true \epsilon \rangle$ または $\langle false \epsilon \rangle$ に置き換えられるので，L に関する言語認識問題はすべて本論文で生成したルールだけで有限ステップで解くことができる．

7.1.3 効 率 性

本論文で生成したプログラムを用いて，2.4 節での文字列を変換すると，次のようになる．

p1: $yes \leftarrow s(caaba)$

p2: $yes \leftarrow s(aaba)$

p3: $yes \leftarrow s(a)$

p4: $yes \leftarrow$

この変換過程で用いた等価変換ルールを以下に示す．

p1 → p2　r3: $s(c\&X) \rightarrow s(\&X)$

p2 → p3　r13: $s(aaba\&X) \rightarrow s(a\&X)$

p3 → p4　r4: $s(a) \rightarrow \langle true \epsilon \rangle$

これと，2.4 節の変換の様子とを比較すると，プログラム生成を行うことによって，大幅に変換ステップ数を減らすことが可能になるだけでなく，1 つ 1 つの変換ステップにかかる計算コストも非常に少なくなっていることが分かる．したがって，計算効率が大きく改善されているといえる．また，計算時間においては，2.4 節でのプログラムを用いた場合は，10 数ミリ秒かかるのに対し，生成したプログラムでは，1 ミリ秒以下となり，時間的にも大きく改善されている．

7.2 プログラム生成システムの停止性

本論文で提案するプログラム生成システムは，正常に停止（終了）すれば，オートマトンに相当するプログラムを生成する．したがって，対偶を考えると，対象とする言語認識問題がオートマトンの範囲で実現不可能であれば停止しないのは明らかである．すなわち，すべての言語認識問題を対象とした場合には，生成システム自体の停止性を保証するのは不可能である．

システムに与えた問題が，潜在的にオートマトンでプログラムできる問題ならば，多くの場合生成アルゴリズムは正常に停止し，プログラムを生成できると考えられる．ただし，一階論理表現を用いて記述できる問題のクラスは非常に広いので，停止性を厳密に証明するのは現時点では未解決の事柄である．

我々は停止性に関して次のように考えている．

- 停止性を厳密に保証するには，ある程度表現できる問題クラスを狭くする必要があるかもしれない．

しかし，できる限り問題クラスは狭くしたくない．

- たとえ停止性を保証できても，実際には，プログラム生成が期待している時間内に終了しないことがある．

以上の理由により，本論文では，実用的な観点から停止性の保証は重要視せず，生成アルゴリズムの停止性の証明については議論しない．

7.3 多様なプログラムの出力

プログラム生成の途中で，付録 A.2 にあるように，

$$s(\&X) \rightarrow \begin{cases} [q(\&X) \notin \mathcal{M}(Q)] \\ Q = \{q(\#Yaa\#Z) \leftarrow [\#Z \notin rep(ba\#U)]\} \end{cases}$$

という等価変換ルールが生成されている．

このルールは，引数が任意の文字列を表していて，ルール内で，元の問題で与えられいるアルファベット Σ の情報は用いていない．したがって，このルールだけで，すべてのケースを処理できる．しかし，このルールの実行時には新たに宣言的記述を生成するなど，時間的，空間的ともにコストがかかるという問題がある．

一方，3.3 節で示したプログラムは，アルファベットの情報を用いてパターンを生成し，パターン別のルールの集合となっている．したがって，プログラムとしては大きくなっているが，ルールが単純なので，実行時には高速な処理が可能である．

アルファベットの数が非常に大きくならない限りは，後者のプログラムを出力する評価基準を与えるのが適切と考えられる．したがって，本システムが最終的に出力するルールには，前者のようなルールは含まれていない．

本プログラム生成システムの評価基準（$eval$）は，3.3 節で示したプログラムを生成するように与えている．しかし，評価基準を別のものと取り換えることにより，出力されるプログラムに含まれるルールを容易に変更することができる．

たとえば，本プログラム生成システムの評価基準を適切に設定することによって，与えられた問題仕様（本論文では言語認識問題）から，

(1) アルファベットに依存しないルールによるプログラム
(2) アルファベットの情報を用いて効率化したルールによるプログラム

という 2 種類の（レベルの違う）プログラムを生成することができる．

8. おわりに

本研究は，等価変換の理論においてこれまで行われていた，一階述語論理表現の扱いに関する研究と，純粋な確定節レベルの仕様から等価変換ルールを生成するという研究を結び付けて得られた．

本論文では，生成するプログラムはオートマトンに相当するレベルに限定して議論したが，この方法は，それよりもっと広範な表現力を持つプログラムを生成する目的にも適用することができる．

参考文献

1) Lloyd, J.: *Foundations of Logic Programming*, 2nd ed, Springer-Verlag (1987).
2) Lloyd, J. and Topor, R.W.: Making Prolog More Expressive, Vol.1, No.3, pp.225–240 (1984).
3) 赤間 清，小池英勝，宮本衛市：等価変換ルールの生成方法の理論的基礎，情報処理学会研究報告 (1999).
4) 吉田忠行，赤間 清，宮本衛市：一階論理制約の等価変換を用いた問題解決の枠組，電子情報通信学会研究報告，KBSE98-6, pp.17–24 (1998).

付 録

A.1 $\mathcal{M}(P)$ の定義

宣言的記述 P の意味 $\mathcal{M}(P)$ とは，以下のようにして得られる基礎アトムの集合である．

$$\mathcal{M}(P) \stackrel{\text{def}}{=} [T_P]^1(\emptyset) \cup [T_P]^2(\emptyset) \cup [T_P]^3(\emptyset) \cup \cdots = \bigcup_{n=1}^{\infty} [T_P]^n(\emptyset),$$

ここで，T_P は，以下のように定義された写像である．

任意の基礎アトムの集合を x とする．$T_P(x)$ は，次の条件を満たすすべての g の集合である．

ある代入 θ と，P の中の任意の確定節 C が存在して，

(1) $C\theta$ が基礎節（変数を含まない節）である，
(2) g は $C\theta$ のヘッドである，
(3) $C\theta$ 中のすべてのボディアトムが x に属する，
(4) $C\theta$ 中のすべての一階論理制約および負参照が真となる，

を満たす．

A.2 生成例 1 の詳細

p1: $h \leftarrow s(\&X)$
p2: $h \leftarrow \forall\alpha\forall\beta([\&X = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])$
p3: $h \leftarrow \neg\exists\alpha\exists\beta\neg([\&X = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])$
p4: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\&X) \leftarrow \exists\alpha\exists\beta\neg([\&X = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])\}$

p5: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\&X) \leftarrow \neg([\&X = \#Yaa\#Z] \Rightarrow \exists\gamma[\#Z = ba\gamma])\}$

p6: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\&X) \leftarrow [\&X = \#Yaa\#Z], \neg\exists\gamma[\#Z = ba\gamma])\}$

p7: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Yaa\#Z) \leftarrow \neg\exists\gamma[\#Z = ba\gamma])$

p8: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Yaa\#Z) \leftarrow [r(\#Z) \notin \mathcal{M}(R)]\}$

$R = \{r(\#Z) \leftarrow \exists\gamma[\#Z = ba\gamma])\}$

p9: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Yaa\#Z) \leftarrow [r(\#Z) \notin \mathcal{M}(R)]\}$

$R = \{r(\#Z) \leftarrow [\#Z = ba\&U])\}$

p10: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Yaa\#Z) \leftarrow [r(\#Z) \notin \mathcal{M}(R)]\}$

$R = \{r(ba\&U) \leftarrow\}$

p11: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Yaa\#Z) \leftarrow [r(\#Z) \notin rep(r(ba\#U))]\}$

p12: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Yaa\#Z) \leftarrow [\#Z \notin rep(ba\#U)]\}$

したがって p1 と p12 のボディより，

$$s(\&X) \rightarrow \begin{cases} [q(\&X) \notin \mathcal{M}(Q)] \\ Q = \{q(\#Yaa\#Z) \leftarrow [\#Z \notin rep(ba\#U)]\} \end{cases}$$

というルールを得ることができる．

この変換過程で用いたメタルールを以下に示す．

p1 → p2　m1: $s(*X)$ の展開
p2 → p3　m2: $\forall$ の除去
p3 → p4　m4: 負制約に書き換え
p4 → p5　m3: $\exists$ の除去
p5 → p6　m5: $\Rightarrow$ の否定の除去
p6 → p7　等式制約を解消
p7 → p8　m4: 負制約に書き換え
p8 → p9　m3: $\exists$ の除去
p9 → p10　等式制約を解消
p10 → p11 単位節を取り出す
p11 → p12 共通述語の除去

A.3 生成例 2 の詳細

p1: $h \leftarrow s(a)$

p2: $h \leftarrow \forall\alpha\forall\beta([a = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])$

p3: $h \leftarrow \neg\exists\alpha\exists\beta\neg([a = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])$

p4: $h \leftarrow [q \notin \mathcal{M}(Q)]$

$Q = \{q \leftarrow \exists\alpha\exists\beta\neg([a = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])\}$

p5: $h \leftarrow [q \notin \mathcal{M}(Q)]$

$Q = \{q \leftarrow \neg([a = \#Yaa\#Z] \Rightarrow \exists\gamma[\#Z = ba\gamma])\}$

p6: $h \leftarrow [q \notin \mathcal{M}(Q)]$

$Q = \{q \leftarrow [a = \#Yaa\#Z], \neg\exists\gamma[\#Z = ba\gamma])\}$

p7: $h \leftarrow [q \notin \mathcal{M}(Q)]$

$Q = \{\}$

p8: $h \leftarrow$

したがって p1 と p8 のボディより，

r4: $s(a) \rightarrow \langle true\epsilon\rangle$

というルールを得ることができる．

この変換過程で用いたメタルールを以下に示す．

p1 → p2　m1: $s(*X)$ の展開
p2 → p3　m2: $\forall$ の除去
p3 → p4　m4: 負制約に書き換え
p4 → p5　m3: $\exists$ の除去
p5 → p6　m5: $\Rightarrow$ の否定の除去
p6 → p7　等式制約が偽となり，節を除去
p7 → p8　負制約を解消

A.4 生成例 3 の詳細

p1: $h \leftarrow s(aaa\&X)$

p2: $h \leftarrow \forall\alpha\forall\beta([aaa\&X = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])$

p3: $h \leftarrow \neg\exists\alpha\exists\beta\neg([aaa\&X = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])$

p4: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\&X) \leftarrow \exists\alpha\exists\beta\neg([aaa\&X = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])\}$

p5: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\&X) \leftarrow \neg([aaa\&X = \#Yaa\#Z] \Rightarrow \exists\gamma[\#Z = ba\gamma])\}$

p6: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\&X) \leftarrow [aaa\&X = \#Yaa\#Z], \neg\exists\gamma[\#Z = ba\gamma])\}$

p7: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{\ q(\&X) \leftarrow [\#Y = [a\#W]], [aa\&X = \#Waa\#Z], \neg\exists\gamma[\#Z = ba\gamma])$

$q(\&X) \leftarrow [\#Y = [\,]], [aaa\&X = aa\#Z], \neg\exists\gamma[\#Z = ba\gamma])\}$

p8: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{\ q(\&X) \leftarrow [aa\&X = \#Waa\#Z], \neg\exists\gamma[\#Z = ba\gamma])$

$q(\&X) \leftarrow \neg\exists\gamma[a\&X = ba\gamma])\}$

p9: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{\ q(\&X) \leftarrow [aa\&X = \#Waa\#Z], \neg\exists\gamma[\#Z = ba\gamma])$

$q(\&X) \leftarrow [r(\&X) \notin \mathcal{M}(R)]\}$

$R = \{r(\&X) \leftarrow \exists\gamma[a\&X = ba\gamma]\}\}$

p10: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{\ q(\&X) \leftarrow [aa\&X = \#Waa\#Z],$
$\neg\exists\gamma[\#Z = ba\gamma])$
$q(\&X) \leftarrow [r(\&X) \notin \mathcal{M}(R)]\}$

$R = \{r(\&X) \leftarrow [a\&X = ba\#U])\}$

p11: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{\ q(\&X) \leftarrow [aa\&X = \#Waa\#Z],$
$\neg\exists\gamma[\#Z = ba\gamma])$
$q(\&X) \leftarrow [r(\&X) \notin \mathcal{M}(R)]\}$

$R = \{\}$

p12: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{\ q(\&X) \leftarrow [aa\&X = \#Waa\#Z],$
$\neg\exists\gamma[\#Z = ba\gamma])$
$q(\&X) \leftarrow\}$

p13: 節が消滅

したがって p1 と p13 のボディより，

$s(aaa\&X) \rightarrow \langle false \rangle$

というルールを得ることができる．

この変換過程で用いたメタルールを以下に示す．

p1 → p2 m1: $s(*X)$ の展開
p2 → p3 m2: $\forall$ の除去
p3 → p4 m4: 負制約に書き換え
p4 → p5 m3: $\exists$ の除去
p5 → p6 m5: $\Rightarrow$ の否定の除去
p6 → p7 等式制約を解消
p7 → p8 等式制約を解消
p8 → p9 m4: 負制約に書き換え
p9 → p10 m3: $\exists$ の除去
p10 → p11 等式制約が偽となり節を除去
p11 → p12 負制約を解消
p12 → p13 負制約が偽となり節を除去

A.5 生成例 4 の詳細

p1: $h \leftarrow s(ab\&X)$

p2: $h \leftarrow \forall\alpha\forall\beta([ab\&X = \alpha aa\beta] \Rightarrow \exists\gamma[\beta = ba\gamma])$

p3: $h \leftarrow \neg\exists\alpha\exists\beta\neg([ab\&X = \alpha aa\beta] \Rightarrow$
$\exists\gamma[\beta = ba\gamma])$

p4: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\&X) \leftarrow \exists\alpha\exists\beta\neg([ab\&X = \alpha aa\beta] \Rightarrow$
$\exists\gamma[\beta = ba\gamma])\}$

p5: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\&X) \leftarrow \neg([ab\&X = \#Yaa\#Z] \Rightarrow$
$\exists\gamma[\#Z = ba\gamma])\}$

p6: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\&X) \leftarrow [ab\&X = \#Yaa\#Z],$
$\neg\exists\gamma[\#Z = ba\gamma])\}$

p7: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\&X) \leftarrow [\&X = \#Waa\#Z],$
$\neg\exists\gamma[\#Z = ba\gamma])$

p8: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Waa\#Z) \leftarrow \neg\exists\gamma[\#Z = ba\gamma])$

p9: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Waa\#Z) \leftarrow [r(\#Z) \notin \mathcal{M}(R)]\}$

$R = \{r(\#Z) \leftarrow \exists\gamma[\#Z = ba\gamma])\}$

p10: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Waa\#Z) \leftarrow [r(\#Z) \notin \mathcal{M}(R)]\}$

$R = \{r(\#Z) \leftarrow [\#Z = ba\&U])\}$

p11: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Waa\#Z) \leftarrow [r(\#Z) \notin \mathcal{M}(R)]\}$

$R = \{r(ba\&U) \leftarrow\}$

p12: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Waa\#Z) \leftarrow [r(\#Z) \notin rep(r(ba\#U))]\}$

p13: $h \leftarrow [q(\&X) \notin \mathcal{M}(Q)]$

$Q = \{q(\#Waa\#Z) \leftarrow [\#Z \notin rep(ba\#U)]\}$

p14: $h \leftarrow s(\&X)$

したがって p1 と p14 より，

$s(ab\&X) \rightarrow s(\&X)$

というルールを得ることができる．

この変換過程で用いたメタルールを以下に示す．

p1 → p2 m1: $s(*X)$ の展開
p2 → p3 m2: $\forall$ の除去
p3 → p4 m4: 負制約に書き換え
p4 → p5 m3: $\exists$ の除去
p5 → p6 m5: $\Rightarrow$ の否定の除去
p6 → p7 等式制約を解消（$\#Y/ab\#W$）
p7 → p8 等式制約を解消
p8 → p9 m4: 負制約に書き換え
p9 → p10 m3: $\exists$ の除去
p10 → p11 等式制約を解消
p11 → p12 単位節を取り出す
p12 → p13 共通述語の除去
p13 → p14 m11: $s(*X)$ の fold 変換

A.6 プログラム生成過程の詳細

4.7 節で定義したルール生成アルゴリズムで，出力するルール集合 GR，およびアトムパターンの集合 Pat が，本論文で取り上げた問題に適用した際にどのように変化するかを説明する．

（while ループ前）

$GR_0 = \emptyset$,

$Pat_0 = \{s(\&X)\}$

（while ループ 1 回目終了後）

$GR_1 = \emptyset$,

$Pat_1 = \{s(\epsilon), s(a\&X), s(b\&X), s(c\&X)\}$

（while ループ 2 回目終了後）

$GR_2 = \{\ s(\epsilon) \rightarrow \langle true\epsilon\rangle, s(b\&X) \rightarrow s(\&X),$
$s(c\&X) \rightarrow s(\&X)\}$,

$Pat_2 = \{s(a), s(aa\&X), s(ab\&X), s(ac\&X)\}$

（while ループ 3 回目終了後）

$GR_3 = GR_2 \cup$
$\{\ s(a) \rightarrow \langle true\epsilon\rangle, s(ab\&X) \rightarrow s(\&X),$
$s(ac\&X) \rightarrow s(\&X)\}$,

$Pat_3 = \{s(aa), s(aaa\&X),$
$s(aab\&X), s(aac\&X)\}$

（while ループ 4 回目終了後）

$GR_4 = GR_3 \cup$
$\{\ s(aa) \rightarrow \langle false\epsilon\rangle,$
$s(aaa\&X) \rightarrow \langle false\epsilon\rangle,$
$s(aac\&X) \rightarrow \langle false\epsilon\rangle\}$,

$Pat_4 = \{s(aab), s(aaba\&X),$
$s(aabb\&X), s(aabc\&X)\}$

（while ループ 5 回目終了後）

$GR_5 = GR_4 \cup$
$\{\ s(aab) \rightarrow \langle false\epsilon\rangle,$
$s(aabb\&X) \rightarrow \langle false\epsilon\rangle,$
$s(aabc\&X) \rightarrow \langle false\epsilon\rangle,$
$s(aaba\&X) \rightarrow s(a\&X)\}$,

$Pat_5 = \emptyset$

(平成 11 年 6 月 18 日受付)
(平成 11 年 9 月 27 日再受付)
(平成 11 年 10 月 20 日採録)

吉田 忠行

昭和 50 年生．平成 10 年北海道大学工学部情報工学科卒業．同年同大学大学院工学研究科システム情報工学専攻修士課程．人工知能，知識処理，等価変換に基づく問題解決の研究に従事．電子情報通信学会会員．

赤間 清（正会員）

昭和 48 年東京工業大学工学部制御工学科卒業．昭和 50 年同大学大学院修士課程修了．昭和 54 年同大学院博士課程単位修得退学．同年同大学助手．昭和 56 年北海道大学文学部講師．平成 1 年同大学工学部助教授．平成 11 年より同大学情報メディア教育研究総合センター教授として現在に至る．人工知能，知識処理，等価変換に基づく問題解決の研究に従事，工学博士．人工知能学会，ソフトウェア科学会，認知科学会各会員．

宮本 衛市（正会員）

昭和 37 年北海道大学工学部電気工学科卒業．昭和 39 年同大学大学院修士課程修了．同年同大学講師．同助教授を経て，昭和 59 年より北海道大学工学部教授として現在に至る．分散システム，並列オブジェクト指向モデル，プログラミング環境等の研究に従事．工学博士．著書に「PASCAL — プログラミングと翻訳技法」等．ソフトウェア科学会，IEEE 各会員．