メタ文法を用いた文字列照合パターンの作成と編集†

南川忠利†† 会田一夫††

形式的文字列照合機能は基本ソフトばかりではなく，CAIにおける解答照合や，パターン認識の構造記述などにも広く利用される．しかし応用によっては始めから目的の照合パターンを決めるのが困難な場合があり，ユーザが試行錯誤により照合パターンを作成編集するための支援機能が必要である．本論文は照合パターン自身を照合するメタレベルの照合機能を用いて，目的の文字列照合パターンを編集改良する方法を示す．まず，文脈自由文法に基づいた汎用の文字列照合変換プログラムを開発し，その照合文法形式自身を照合するメタレベルの照合文法を自己記述する．このメタ照合文法を土台にして，任意の文字列照合文法を照合・変形するための，メタレベルの変換文法を作る．メタ変換文法として以下の例を示す．(1)文字列の揺動的な変動に対処するための，段階的照合機能拡張．(2)照合文法の簡単化．応用例として英会話のCAIを取り上げ，生徒のさまざまな解答に対処する照合パターンを，実際の解答例から段階的に生成編集することを試みた．予備的な実験の結果，次の効果が確認された．(1)ユーザが直接文法形式を入力せずに照合パターンの生成編集ができる．(2)目的の文法と，メタレベルの文法に同一の照合機能を用いるので，一元的な管理ができる．

1. はじめに

文字列照合技術の基本的性質は形式言語理論と計算可能性の理論によりかなり明らかにされている[1)~3)]．これらの研究成果は，コンピュータ言語の仕様記述とシンタクス解析，エディタやコマンド，データベースのキーワード検索はもちろん，最近ではパターン認識における図形や文字の構造記述，CAIなど，広範囲の分野で利用されている[4)~9)]．

ところでパターンの構造記述やCAIの解答分析などの応用分野ではコンピュータ言語と異なり，目的の文法を最初から正しく記述することは困難である．このため多数のテストデータや観測データを用いた試行錯誤を行いながら文法を改良し，パターンの照合能力を高める方法がとられる．この試行錯誤を自動化する研究として，例題から文法を推定するGold, Angluinらの研究があるが[10)~12)]，主として文法の同定可能性や計算量の理論構築を目指したものであり，まだ現実的なシステムへの応用は少ない．

これらの研究に対し，例題の提示だけではなく人間の知識を利用して文法を取り扱う対話的なシステムを作成することが実用上有効と考えられる．目的の文法に到達するまでに，延々と例題だけを与え続けるよりも，不完全でもよいからユーザが知っている知識や文法を，ヒントとして利用するほうが，ユーザにとっては使いやすい．このためには文法をさらに1段上のレベルで照合したり変形する基本手段の提供が必要と考える．

本論文では文字列照合機能にパターン変換機能を追加し，照合パターンの文法そのものを照合・変換可能にしたプログラムMMについてまず述べる．ついでその応用例として，MM自身を用いた文法の拡張，簡単化などの方法を示す．特に文法をそれ自身で一元的に取り扱えるMMの特徴は処理の体系化やユーザの知識の集積にとって有利である．

以下の説明では

(1) 通常の文字列（被照合文字列）

(2) (1)の文字列を照合する文法を表す文字列（照合パターン）

(3) (1)と(2)の説明をするための記号

の3レベルの文字列が現れ，混乱が予想される．

(2)については次の章で説明する．

(3)のために用いる記号は以下のように定める．

照合の対象になる通常の文や単語は α, β, γ などで表す．照合のための文法は ϕ, ψ などで表す．しかし通常の文字列そのものもその文字列だけからなる文法とみなせるので，両者を区別しない場合もある．紛らわしい場合には，文字列 α，文法 ϕ のように表記する．また文字列としての等号を≡で表す．変換を目的とする文法は Φ, Ψ などの大文字で表す．Φ によって，α が β に変換されることを $\beta \equiv \Phi\{\alpha\}$ あるいは Φ を省略して，$\alpha \rightarrow \beta$ と書く．$\Phi\{\alpha\}$ がまた文法で，これを γ に適用する場合，$\Phi\{\alpha\}\{\gamma\}$ と書く．

2. 汎用文字列照合・変換プログラム MM

一般に汎用的な文字列照合機能 f は，目的とする

† Generation and Editing String Matching Pattern Using Metagrammar by TADATOSHI MINAMIKAWA and KAZUO AIDA (Toshiba Information and Communication Systems Laboratory).

†† (株)東芝情報通信システム技術研究所

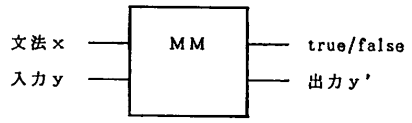

図 1 文字列照合・変換プログラム MM
Fig. 1 MM: A string matcher and transformer.

パターンを表す文法 x と入力文字列 y に対して，$f(x, y)$=TRUE または FALSE を返す．実用システムでは，x のカテゴリとして，正規文法を採用するものが多い．筆者らは x のカテゴリとして，より強力な文脈自由句構造文法にパターン変数を導入したものとし，さらに，照合部分を他の文字列に変換できる照合・変換プログラム（以下 MM と呼ぶ）を開発した(**図1**)．以下に3章以後に必要な MM の機能の概要を例を用いて説明する．

2.1 MM の文字列照合機能

(1) OR: 'THIS', 'THAT'
は THIS または THAT と照合する．文字列は ' で囲むが，混乱がなければ ' を省略してよい．

(2) 空白: BIG-NOSE
- は任意の長さの空白を表す．

(3) 省略: SHE-IS-<VERY->CUTE
< >内は省略可能である．

(4) 反復: SHE-IS-(VERY-)*CUTE
VERY がいくら続いてもよい．

(5) ワイルドカード: THE+IS WILD
+は任意の文字列とマッチする．

(6) 数字: #
は0～9までの数字1文字を表す．

(7) アルファベット: @
@はアルファベット1文字を表す．

(8) 文字列変数: $X1$
変数 $X1$ に代入されている文字列と照合する．例えば $X1$ に IS が代入されていれば THIS-$X1$-A-PEN は THIS-IS-A-PEN と同等である．変数は $X0$, $X1$, $X2$, … 等と表す．

(9) 代入: $X3$=(THIS, THAT)-IS-A-PEN
照合の結果が $X3$ に代入される．したがって $X3$ には THIS IS A PEN または THAT IS A PEN のいずれかが代入される．変数に照合結果を代入するタイミングは，照合をすべて終えた後である．

(10) 即時代入: ($X2$: (MARY, JANE))-LIKES-$X2$
=と違い，：はその場で変数に代入するので，これは MARY LIKES MARY にマッチする．

(11) 文法変数: &$X2$
$X2$ に代入されている文字列を文法として扱う．例えば $X2$ に THIS-IS-A-PEN という文字列が代入されていれば，直接 THIS-IS-A-PEN と書いたのと同等である．変数は再帰的に用いてもよい．これにより，文脈自由な句構造文法を表現できる．

(12) 単語: THIS-IS-A-@～
@～は任意長の単語とマッチし，空白やコンマなどの特殊記号に出会うまでアルファベットを食いつくす．同様に #～は任意長の数字とマッチする．

(13) AND: +MARY+ ; +JANE+
文の中に MARY と JANE が同時に含まれていなければならないことを示す．コンマが OR であるのに対し，；は AND を表す．同一文字列に対して異なる分析をするときに用いる．

2.2 MM の文字列変換機能

文法において変換したい部分を，(ϕ/ψ) あるいは，<ϕ/ψ>の形式で指定すると，照合条件 ϕ にマッチした部分が ψ に置換される．変換は照合成功後の変数代入 (Xi=…) 実行時に行われる．以下に変換の機能を例題によって説明する．

(1) 単純な置換: THIS-(IS/WAS)-A-PEN
THIS IS A PEN とマッチし，IS が WAS に置換されて，THIS WAS A PEN になる．

(2) 照合結果の置換: (MARY, JANE/SHE)-IS-CUTE
MARY IS CUTE または JANE IS CUTE とマッチし SHE IS CUTE になる．

(3) 変数 X の利用:
$X2$ に WAS が代入されているとき，THIS-(IS/$X2$)-A-PEN は(1)と同等である．

(4) 変数 &X の利用:
$X2$ に MARY, JANE が代入されているとき，&$X2$ はその文字列を文法として扱うことを表す．(&$X2$/SHE)-IS-CUTE は(2)

と同等である.

(5) 変数 &X による照合結果の参照:
X2 に MARY, JANE が代入されているとき, (&X2/THIS &X2)-IS-CUTE は例えば, JANE IS CUTE にマッチして, JANE は THIS JANE に置き換えられる. この結果, 全体は THIS JANE IS CUTE になる. すなわち, /の右側にある &X2 は, /の左側にある &X2 にマッチした文字列を表す.

(6) 変数 &&X による置換結果の参照:
X2 に, (MARY, JANE/ALICE) が代入されているとき, (&X2/THIS &&X2)-IS-CUTE に JANE IS CUTE をマッチさせると, 今度は THIS ALICE IS CUTE になる. /の右側にある &&X2 は, 左側にある &X2 とマッチした入力をそのまま用いずに, &X2 で指定された置換を行った結果を表す (&& に注意). 同じ X2 で, (&X2/THIS &X2) IS CUTE とした場合は (& が1個であることに注意), (5)と同じで, THIS JANE IS CUTE になる.

(7) 挿入: THIS-IS-(/A) PEN
THIS IS PEN とマッチし, THIS IS A PEN になる. すなわち, /の左側が空の場合は, 挿入を意味する.

(8) 削除: THIS-IS-(A-/) MARY
THIS IS A MARY にマッチし, THIS IS MARY になる. すなわち, /の右側が空の場合は削除を意味する.

以上では簡単な例を示したが, これらの変換命令をもっと複雑な文法の中に埋め込んでもよい. またφは複雑な文法であってもよい.

3. メタ文法を用いた文字列照合パターンの操作

本論文の目的は, 文字列照合文法 φ の対話的作成編集を支援することである. ここで述べた照合文法や数式のような形式構造の編集機能としては, 一般に次の三つの機能が考えられる.

まず第1に, 文法の基本的編集機能として, 通常のエディタと同様の挿入, 削除, 置換機能が必要である. これらについては2.2節で述べた置換, 挿入, 削除機能が利用できることは明らかであるが, さらに言えば, 構造化エディタに見られるような, シンタクスに基づく編集機能が望ましい.

第2の機能として, ユーザがよく用いる一連の編集手順の自動化がある. これは応用に依存し, アドホックであるが, ユーザの知識を具体化した強力な機能である. ここでは文法の照合能力を評価データから逐次に改善することを目的としているので, 照合文法φの機能を自動的に拡張する機能を考える. 例えばφを新たな文xを照合できるように拡張したい. 拡張の方法は種々考えられるが, ここではφの能力を少しずつ増加させ, ついにはxを照合可能にする方法を取り上げる.

第3の機能として, 上記の編集を行った後のφの形式の整理, 簡単化があげられる. 特に拡張操作を行った後は, 不要の括弧や無駄な照合部分が生じるためこの機能は重要である.

以上の3種類の機能を実現するためには, 編集機能がφの文法を陽に「知っている」必要があり, またユーザが編集機能を自由に拡大できることが望ましい. これを実現する方法として, 図2に示すように, MM の照合機能自身を用いて φ を分析編集することを考え, その基本機能を述べる.

3.1 MM 文法の自己記述

メタ文法による操作の第一歩として φ の形式を定めている MM 文法を MM 自身で記述したメタ文法 Φ_{mm} を図3に示す (ただし説明のため多少省略してある). X1 は MM 全体の文法を表し, MM (X1,

図2 メタ文法による文法の解析・変換
Fig. 2 Analysis and transformation of a grammar using meta-grammars.

X1≡& X2<(',' & X2)*>	OR
X2≡& X3<(';' & X3)*>	AND
X3≡<'X'#～('=', ':')>& X4	代入
X4≡& X5*	単項列
X5≡& X6<'*', '～'>	繰返し
X6≡@～,	単語
'¥' '+' ¥",	'φ'
'('& X7')',	(φ)
'<'& X7'>',	<φ>
'+', '−', '#', '@',	wild card
<'&'<'&'>>'X' #～	変数
X7≡<& X1><'/'<& X1>>	φ/ψ

図3 MM の文法の MM 自身での記述
'φ' は文字列φを表す. ¥はエスケープ記号で, ¥' は ' を表す.
Fig. 3 Self-description of the MM grammar.

ϕ)=true である．$X2, \cdots, X7$ はその部分文法を表している．Xi が照合できる ϕ の集合を Li で表すと Li (i=1〜7) は各々以下の部分パターンを表している．

$L1$ は ϕ 全体の集合でありコンマで区切られた OR のパターン，$L2$ は ；で区切られた AND のパターンである．$L3$ は＝または，：による代入，$L4$ は単項の列 a (a, b) b など，$L5$ は単項に＊または〜がついたもの，$L6$ は abc や $X1$ や (a, b) などの単項，$L7$ は（ ）や< >の中身で，ϕ かまたは ϕ/ϕ の形式である．

3.2 文法の拡張

ここではユーザが与えた文法 ϕ の照合範囲を徐々に拡大する操作を考える．

例えば $\phi 1$≡This is a pen の場合（これはこの1文しかマッチしない単純な文法とみなせる），$\phi 2$≡This-is-a-pen は単語間に任意長のスペースを許すことになり，$\phi 1$ の拡張になっている．このような拡張は

Φ_{space}≡(@〜<-/'-'>)＊

を $\phi 1$ と照合させることによって得られる．

@〜が $\phi 1$ の単語部分，- が $\phi 1$ の空白部分にマッチし，空白を-で置き換える．＊はその繰返しを表す．

より複雑な拡張の一つとして以下に述べる「1次の拡張操作」Φ_{ext} がある．これは単語の部分的挿入削除が生じても照合できるように文法の逐次拡張を目的とする変換操作である．以後説明を簡単にするために，α と β と γ の3単語の列からなる文，$\alpha\beta\gamma$ を考える．

3.2.1 1次の拡張操作 Φ_{ext}

α を単語，ϕ を単語列からなる文とするとき，1次の拡張操作 Φ_{ext} を次のように定義する．

$\Phi_{ext}\{\alpha\}$≡<α<ω>>

$\Phi_{ext}\{\alpha\phi\}$≡($\alpha\Phi_{ext}\{\phi\}$, $\Phi_{ext}\{\alpha\}\phi$)

ここで ω は-@〜を表し，任意の単語とマッチする．これを $\alpha\beta\gamma$ に適用すると，

$\Phi_{ext}\{\alpha\beta\gamma\}$≡($\alpha$($\beta$<$\gamma$<$\omega$>>,
<β<ω>>γ), <α<ω>>$\beta\gamma$)

となる．括弧をはずして展開すると，

$\Phi_{ext}\{\alpha\beta\gamma\}$≡$\alpha\beta$<$\gamma$<$\omega$>>,
α<β<ω>>γ, <α<ω>>$\beta\gamma$

になる．$\Phi_{ext}\{\alpha\beta\gamma\}$ は

（1） $\alpha\beta\gamma$

（2） α か β か γ のいずれか1単語を削除した文

（3） α か β か γ のいずれか1単語の後に任意の1単語を挿入した文

```
X1≡&X2<(','&X2)*>
X2≡&X3<(';'&X3)*>
X3≡<'X' ₩〜('=', ':')>&X4
X4≡&X5,
      (&X5&X4/
      '('&X5&&X4', '&&X5&X4')')
X5≡(/'<')&X6(/'<−@〜>>')
                        <'*', '〜'>
X6≡@〜,
      '¥' '+' ¥",
      ' ('&X7') ',
      '<'&X7'>',
      '+', '−', '₩', '@',
      <'&'<'&'>>'X' ₩〜
X7≡<&X1><'/'<&X1>>
```

図 4 1次の拡張操作の MM による記述
Fig. 4 Extension grammar of first order described in MM.

とマッチする．

この拡張操作 Φ_{ext} を，図3の Φ_{mm} に埋め込んだものを図4に示す*．$X4$ と $X5$ だけが Φ_{mm} と異なっている．$X4$ は $\Phi_{ext}\{\alpha\phi\}$，$X5$ は $\Phi_{ext}\{\alpha\}$ の変換に対応する．また陽には現れていないが，$X1$ では，$\Phi_{ext}\{\phi 1, \phi 2\}$≡$\Phi_{ext}\{\phi 1\}$，$\Phi_{ext}\{\phi 2\}$ $X5$ では，$\Phi_{ext}\{(\phi)\}$≡($\Phi_{ext}\{\phi\}$) のように変換が再帰的に適用されていることに注意を要する．

3.2.2 n 次の拡張操作

$\Phi_{ext}\{\phi\}$ も MM 文法であるので，Φ_{ext} を2回以上適用することができる．

$\Phi^n{}_{ext}$ により n 回の操作を表すと，$\Phi^2{}_{ext}\{\alpha\beta\gamma\}$ は

（1） $\Phi_{ext}\{\alpha\beta\gamma\}$ がマッチする文．

（2） α か β か γ のいずれか2単語を削除した文．

（3） α か β か γ のうちいずれか2単語の直後に各々任意の1単語ずつを挿入した文．ただし，その2単語は同一でもよい．

（4） α か β か γ のうちいずれか1単語を削除し，さらに α か β か γ のうちいずれか1単語の直後に任意の1単語を挿入した文．両者の位置が同じ場合には置換した文．

になる．以下同様に，n 次の拡張操作は n 次までの削除と挿入を表現することになる．

3.3 文法の部分的拡張

前節では与えられた文法 ϕ からその中の任意の単語に対して，挿入削除による変動を許容する文法 ϕ を作り出す文法を示した．ここでは1単語だけについて変動した入力文 y が与えられたときに，その単語部分だ

* 実際には $X5$ で表された単項のうち，Φ_{ext} を適用するものとしないものを分ける必要がある．例えば，- には適用しない．

けを拡張した文法を作るメタ文法を考える．例えば，$\phi \equiv \alpha\text{-}\beta\text{-}\gamma$ に対し，入力が $y \equiv \alpha\gamma$ のときに，

$\phi \equiv \alpha\text{-}<\beta>\text{-}\gamma$

を作り出したい．これは次のような2段階の操作で行える．

$\phi \equiv \Phi_{ext1}\{\phi\}\{y\}$

まず Φ_{ext1} は前節の Φ_{ext} と似た操作であるが，Φ_{ext} が ϕ の各単語 α を $<\alpha<\text{-}@\sim>>$ に変換したのに対して，Φ_{ext1} は α をつぎの機能を持つ「変換文法」に変換する．

(1) 空 → $<\alpha>$
(2) $\alpha \rightarrow \alpha$
(3) $\alpha\xi \rightarrow \alpha<\xi>$
(4) $\xi \rightarrow (\alpha, \xi)$

すなわち，α に変動があればそれを許容し，なければ何もしない．(4)は α が ξ に置換された場合である．これを MM の文法で表現すると，

$\Phi_{ext1}\{\alpha\} \equiv (/'<'\alpha'>'),$
$\alpha<(/'<')@\sim(\text{-}/'\text{->}')>,$
$(/'(\alpha, ')@\sim(/')')$

$\Phi_{ext1}\{\alpha\phi\} \equiv (\alpha\Phi_{ext1}\{\phi\}, \Phi_{ext1}\{\alpha\}\phi)$

Φ_{ext1} を実現するには図4の $X5$ で上記の変換が生成されるようにすればよい．

3.4 文法の簡単化

文法に挿入・削除などの操作を行った結果，冗長な文法になる場合がある．例えば，$\Phi^2_{ext}\{\alpha\beta\}$ をそのまま展開してみると，

$\Phi_{ext}{}^2\{\alpha\beta\} \equiv \alpha\Phi^2_{ext}\{\beta\}, \ \Phi_{ext}\{\alpha\}\Phi_{ext}\{\beta\},$
$\Phi_{ext}\{\alpha\}\Phi_{ext}\{\beta\}, \ \Phi^2_{ext}\{\alpha\}\beta$

となり，$\Phi_{ext}\{\alpha\}\Phi_{ext}\{\beta\}$ は冗長である．また，ユーザが文法の記述中にうっかり冗長な記述をすることも考えられる．冗長性を除去し，できるかぎり簡単な文法に変換することは計算時間を省くためにも重要である．

MM 文法の簡単化は数式処理における多項式の簡単化に類似性を求めることができるが，MM では変数代入や，置換の副作用があるため，交換法則が単純には成立しない．このために，数式処理で行われている項の並べ換えによる式の整理が，MM 文法では行えない．ここではヒューリスティックな変換を繰り返し適用する方法をとる．

簡単化の変換パターンを Φ_{simp} とするとき，入力 ϕ の簡単化は Do while $\phi = \Phi_{simp}\{\phi\}$ is true. を実行すればよい．Φ_{simp} のうち MM に特有な簡単化の例を示す．ここで，α と β と γ は $L4$，また，ϕ と ψ と η は $L1$ のパターンとする．

$\alpha\beta, \alpha\gamma \rightarrow \alpha(\beta, \gamma)$
$\beta\alpha, \gamma\alpha \rightarrow (\beta, \gamma)\alpha$
$\phi, \alpha, \psi, \alpha, \eta \rightarrow \phi, \alpha, \psi, \eta$
$\alpha\gamma, \alpha\beta\gamma \rightarrow \alpha<\beta>\gamma$
$<\alpha>, \alpha \rightarrow <\alpha>$
$<' ' *> \rightarrow -$
$++ \rightarrow +$
$-- \rightarrow -$
$+, \phi \rightarrow +$
$@\sim@\sim \rightarrow @\sim$
$@\sim, \alpha \rightarrow @\sim$ α は単語

3.5 括弧の除去

多項式のように括弧をはずす処理も必要である．単語 α と β と γ からなる列 $\alpha\beta\gamma$ のような単語の列を ϕ で表し ($\phi \in L4$)，$\phi 1, \phi 2, \cdots, \phi n$ の形式を ϕ で表すと ($\psi \in L1$)，変換知識は以下のようになる．

$\phi 1(\phi 2, \phi)\phi 3 \rightarrow \phi 1 \ \phi 2 \ \phi 3, \phi 1(\phi)\phi 3$
$\phi 1, (\phi, \phi 2), \phi 3 \rightarrow \phi 1, \phi, \phi 2, \phi 3$
$(\phi) \rightarrow \phi$
$(\ ) \rightarrow$ 空

4. CAI の解答分析への応用[13]

ここでは，3章で述べた MM の基本照合・変換パターンの利害得失を調べるため，実動している語学 CAI システム[14]における解答分析への応用例を述べる．

CAI システムでは教育者の意図や経験を，手作りでプログラム化する方法がとられる．なかでも解答の分析とそれに対する適切なアドバイスを与えることは，個人学習を身上とする CAI の重要な役目である．筆者らが経験した英語会話の文型を教える CAI システムの開発では，フィールドテストによる解答分析と，教材の継続的改良が，不可欠であると同時に労力を要する作業であった．このような教材編集作業を効率化するためにメタレベルの文字列照合・変換機能を用いた実験例を述べ，本方式の効果と問題点を考察する．

実際の教材は 1,000 問以上，数万の MM 照合パターンを含む．ここでは1問分の教材しか取り上げないが，どの照合パターンもここで取り上げる例に類似しているので，本方式が適用可能と考えられる．

4.1 問題の設定

CAI プログラムを簡単化して考えると，1個のフ

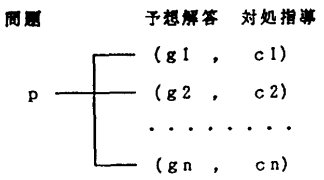

図 5 語学 CAI のフレームのモデル
Fig. 5 A simple model of a CAI frame for English conversation training.

	(文法 g)	(分類 c)
A.	I-hope-⟨that-⟩it-will-snow-tomorrow	正解
B.	I-hope-it-rain-tomorrow	rain—誤
C.	I-hope-it-((rain, snowed), snow)-tomorrow	will—欠
D.	I-hope-it-is-(snow, snowing)-tomorrow	is—誤
E.	I-hope-it-will-be-((snow, rain), snow)-tomorrow	be—誤
F.	I-hope-it-will-be-rain-tomorrow	rain—誤
G.	I-hope-it-will-be-sonw-tomorrow	sonw—誤

図 6 メタ変換文法を用いた CAI 解答照合文法の生成
Fig. 6 Application of meta-grammars to generation of CAI answer matching patterns.

レームは，問題 p に対して予測される生徒の解答パターンを表す照合文法 g と，それへの対処指導方法 c の対 (g, c) の集まりによって表せる (**図 5**)．g は誤答ばかりでなく，正答もある．c としては種々の処理が含まれるが，ここでは簡単のために，g がどのような誤り（または正解）であるかを示すカテゴリとしておく．今このフレームを多数の学習者が学習したときに入力された実際の解答を，$y1, y2, \cdots, ym$ とする．この中には予測しない（どの g でも照合できない）入力 y がいくつか含まれている．問題は，少なくとも $y1, \cdots, ym$ に対しては適切にカバーできるように $(g1, c1) \cdots (gn, cn)$ を効率よく改良することである．

4.2 実験システム

y に対するカテゴリは人間が定めるとして，以下の作業を支援するシステムを考える．どの g でも照合できなかった入力を y とするとき，

（1） y に近い g を探す．もしも g が存在し対応するカテゴリが適切ならば，g を拡張改良する．

（2） y に近い g がなければ，あるいはカテゴリが不適切ならば，y から新しい文法 g を作り出す．

y に近い文法 g の探索には，1次の拡張機能 Φ_{ext} を利用する．すなわち，$\Phi_{ext}\{g\}\{y\}=$true になるような g を探し出す．g が存在した場合の g の改良には，Φ_{ext1} を利用する．これにより，g に対する摂動的な改良が可能になる．$\Phi^{n}_{ext}(n=1,2,\cdots)$ を適用することも考えられる．

y に近い文法 g が見つからない場合の新規文法の生成には，Φ_{space} を利用する．

図 6 は実際に CAI で使われている教材の1フレームに対して上記の処理を行った結果の一部を示す．問題 p は「"明日雪が降るといいなあ" を英語に直しなさい」である．模範解答は I hope it will snow tomorrow である．これに対して 48 種の解答が実際に収集されている．実験では，g の初期集合を空とし，一つずつ y を選んで(1), (2)の処理を行った．ただし拡張は1次までとし，挿入削除のほか，置換も扱えるように修正した．カテゴリ c の判断は人間が行う．

主な誤りは be 動詞を入れてしまう，will を使わない，sonw などのスペルミス，雪と雨を取り違える，等である．一つの c を一つの誤りに対応させたので，例えば二つの誤りを持つ I hope it rain tomorrow は，図でBとCの両方に反映されている．EとGの sonw，EとFの will be rain も同様である．また，BとFは統合可能であるが，両者の近さを認識するには2次の拡張が必要である．

4.3 考　　察

この実験システムは，次の点で省力効果を期待できる．

（1） 蓄積された多数の文法パターンから改良候補を自動選択してくれる．

（2） 新規文法の生成と改良操作を自動的に行うので，ユーザは複雑な形式文法を入力する必要がない．

この効果は主として文法の拡張操作からくる．しかしながら，ここで提案した拡張操作は，形式的な取扱いが容易な反面，以下の問題点も明らかになった．

第1に高次の形式的拡張をやみくもに行うと，変換結果の文法パターンが爆発的に大きくなる．実際に図6の例では2次の拡張までが限度であった．この問題はデータ構造の工夫により改善可能と考える．

第2に，形式的な拡張の結果，照合機能の無意味な汎化を生じる．実際に，実験システムで y に近い g を探す際に，無意味な候補も取り込むきらいがあり，適切な g を選択するには人間が行うカテゴリ判断に依存せざるをえなかった．この問題はどのような拡張をどの程度すべきかという本質的な問題に関係するが，応用に強く依存しており，今後の研究課題である．

また，本実験では試みなかったが，スペルミス程度

の判断は自動化可能と考えられる.

5. 他の研究との関連

ここで取り上げた問題は，帰納推論の一部である例題からの文法推論と関係が深い．文法推論を理論的に取り扱ったパイオニアは Angluin と Gold である[10)~12)]．文例を与えていくと有限回の試行でその文例にあった正しい文法を作り出せるときに，この言語の文法は例題から推論可能であるという．彼らは

（1） ある言語の族が与えられたとき，これらの言語を洗いざらい数え上げることができれば（帰納的に枚挙可能ならば），その族の言語 L は文例の完全提示（L に属する文例も，属さない文例も，両方とも提示すること）で推論可能であることと

（2） 正提示だけでは推論できない言語族があることを示した.

（1）は一見明るい結論と見えるが，言語を総当たり的に調べ上げれば理論的には同定可能であることを証明しただけで，実用的な同定方法を示したわけではない．逆に（2）は悲観的結論に見えるが，Angluin が示した言語族の例は特異な例であり，正提示からの推論を全面的に否定するものではない．極端な例として有限言語ならば，正提示だけでも推論可能である.

実際に工学で取り扱う言語の多くは長さに制限をつけてもさしつかえない場合が多く，有限と考えてもよい．したがって実用的観点からは「推論可能かどうか」よりも，どれだけ効率的に目的にかなった言語を見つけ出せるかが重要である．篠原は Angluin が正提示だけから推論可能な例として考案したパターン言語の一種で，長さの多項式時間以内で推論可能な無限文法を見いだし，文献検索に応用している[12)]．

言語の推論と関係の深い帰納推論の研究に Shapiro のモデル推論がある[15)]．これは例題の完全提示から PROLOG のプログラムを自動作成する研究であるが，PROLOG を用いて文字列照合プログラムあるいはパーサを作ることは容易であるので，言語の帰納推論に直接応用できる．実用サイズのプログラムを作れるまでには至っていないが，本論文の拡張操作に当たる精密化操作は参考になる.

これらの研究を実用に結び付けるためにはヒューリスティクスの採用とのトレードオフが必要である．本論文で示した方法は応用分野に依存した知識を用いて文法空間の探索を助ける方向を目指している点で，これらの研究と異なるアプローチである.

応用分野に依存した文法推論の例としては，Fu らによるパターン認識への応用研究がある[8)]．文法の終端記号に，図形の基本要素を対応させれば，確率的な生成文法により図形のゆらぎやノイズが表現できる，そこで文法のルールの確率を調整することで，特定の図形を認識する文法を作り出す．したがって目的とする図形の文法をゼロから生成するものではない．CAI システムにおいては，解答照合に関する文献は少なく，解答照合パターンの改良支援に関する研究は筆者の知る限りでは見あたらない.

6. おわりに

文字列照合・変換プログラム MM を開発し，MM の文法そのものを照合・変換するメタ変換文法をベースとして，文法の拡張や簡単化等の知識を一元的に表現する方法を提案した．一例として語学 CAI の解答分析への応用を示した.

現状ではまだ本格的な応用を云々できるレベルにはないが，MM のパターン照合・変換機能を用いると，通常のプログラミングでは結構面倒な記号処理の問題が，非常に簡単に解決できることが実感できた．また，パターン照合に関する知識の追加が容易であることも分かった.

MM の実装は C 言語で行い，パソコン J 3100 と VAX 8800 で実験を行った．Φ_{ext} の処理（メタレベルの照合）には，J 3100 で 1～10 秒，オブジェクトレベルの照合では 0.01～0.1 秒を要する.

今後の大きな研究課題として，

（1） より広い応用分野での実用化研究

（2） メタレベルの変換知識の集積

が挙げられる.

謝辞 本研究の機会を与えていただくとともに，ご指導いただいた，神田外語学院の佐野学院長と中村教務課長，東芝情報通信システム技術研究所，渡辺所長に謝意を表する.

参考文献

1) Hopcroft, J. E. and Ullman, J. D.: *Formal Languages and Their Relation to Automata*, Addison-Wesley, Reading, Mass. (1969).
2) Aho, A. V. and Corasick, M. J.: Efficient String Matching: An Aid to Bibliographic Search, *CACM*, Vol. 18, pp. 333-340 (1975).
3) Sedgewick, R.: *Algorithms*, Addison-Wesley, Reading, Mass. (1983).
4) Lesk, M. E.: Lex—A Lexical Analyzer Gen-

erator, Computing Science Technical Report, No. 39, Bell Laboratories, Murray Hill (1975).
5) Johnson, S. C.: Yacc: Yet Another Compiler Compiler, Computing Science Technical Report, No. 32, Bell Laboratories, Murray Hill (1975), also in The SUN Reference Manuals, Programming Utilities, Sun Microsystems, Mountain View, California (1986).
6) Peterson, J. L.: Computer Programs for Detecting and Correcting Spelling Errors, *CACM*, Vol. 23, No. 12, pp. 676-687 (1980).
7) Sherwood, B. A.: *The Tutor Language*, Computer-based Education Research Laboratory & Department of Physics, Univ. of Illinois, Urbana (1975).
8) Fu, K. S. and Lee, H. C.: Stochastic Linguistics for Picture Recognition, Purdue Univ., School of Electrical Engineering, TR-EE 72-17 (1972).
9) 長尾 真, 松山隆司: 構造的パターン認識〔1〕, 〔2〕, 計測と制御, Vol. 20, No. 4, pp. 431-440; No. 6, pp. 604-613 (1981).
10) Gold, E. M.: Language Identification in the Limit, *Inf. Control*, Vol. 10, pp. 447-474 (1967).
11) Angluin, D. and Smith, C. H.: Inductive Inference: Theory and Methods, *Comput. Surv.*, Vol. 15, No. 3, pp. 237-269 (1983).
12) 篠原 武: 言語の帰納推論, 知識の学習メカニズム (淵 一博監修), 共立出版 (1986).
13) 南川忠利, 会田一夫: 自己変換可能な文字列照合を用いた解答分析, 第36回情報処理学会全国大会論文集, pp. 2415-2416 (1988).
14) Tanabe, Y. et al.: CAI in Language Education, *Computers in Education* (Lecarme, O. and Lewis, R. eds.), p. 675, North-Holland (1975).
15) Shapiro, E.: Algorithmic Program Debugging, Ph. D. dissertation, Computer Science Department, Yale Univ. (1982).

(平成元年 5 月 12 日受付)
(平成元年 10 月 11 日採録)

南川 忠利 (正会員)

昭和 17 年生. 昭和 41 年東京大学計数工学科卒業. 昭和 43 年同大学院修士課程修了. 同年(株)東芝総合研究所入社. 昭和 62 年より(株)東芝情報通信技術研究所に勤務, 現在に至る. CAD, CAI, エキスパートシステムなどの研究開発に従事. その間昭和 49 年より 1 年半, 米国 MIT 訪問研究員として Project MAC の MACSYMA の研究開発に参加. 著書,「電子回路の CAD」「第 4 世代言語」, 訳書「AI ビジネス」等 (いずれも共著共訳). 電子情報通信学会, ME 学会, ACM 各会員.

会田 一夫 (正会員)

1947 年生. 1972 年東京都立大学理学部数学科卒業. 同年(株)東芝に入社. 語学用 CAI, 医学用 CAI, エディタ等のマンマシーンシステムの研究開発に従事. 現在, 同社情報通信システム技術研究所に勤務. CAI 学会会員.