準 LL(2) 文法に対する構文解析高速化のための解析表の構造†

吉田敬一†† 竹内淑子†††

筆者らは以前準 LL(k) 文法なるものを提案し，その k=2 の場合についての解析表作成アルゴリズム，ならびに解析アルゴリズムを示した．準 LL(k) 文法の文法クラスは LL(k)⊃準 LL(k)⊃強 LL(k) を満足するものであり，LL 文法がもつ「解析法が下降型であるため見通しがよい」，「解析表が小さい」などの特徴をもつ．また，準 LL(2) 文法は LL(1) 文法よりも大きい表現能力をもつ．Aho らは LL(k) 文法に対する解析表作成アルゴリズムと解析アルゴリズムを提案しているが，表作成のアルゴリズムには多くの手続きと計算を必要とする．さらに，いくつかの文法に対して作成された解析表の大きさは筆者らの 120～400 倍にもなる．ただし，構文解析の速さは筆者らよりも2倍程度高速である．本論文では吉田-竹内(1990) で提案した解析表の構造に工夫を加えることにより，構文解析の速さを従来の筆者らの方法の約2倍程度速くなり，Aho らの方法と同程度にすることができた．この変更の結果，解析表は大きくなったが，それでも Aho らのものに比べ，1/15～1/30 程度にとどまっている．

1. はじめに

LR 文法にくらべて，LL 文法は文法クラスが小さいという欠点をもつ一方で，(1)解析表の作成が容易である，(2)解析プログラムが小さくすむ，(3)下降型解析は見通しがよい，(4)属性文法との相性がよい，(5)プログラミング言語のかなりのものを LL(1) 文法で表すことができる，といった長所をもつことはよく知られている．筆者らは任意の k に対する準 LL(k) 文法の提案と，k=2 の場合の解析表作成法，ならびに解析アルゴリズムを示した[1),2)]．準 LL(1) 文法は LL(1) 文法と等しいので，準 LL(2) 文法は LL(1) 文法の拡張になっている．また準 LL(2) 文法は LL(2) 文法にわずかに制限を加えたものであり，準 LL(2)，LL(2)，強 LL(2) の3つの文法クラスの間には LL(2)⊃準 LL(2)⊃強 LL(2) なる包含関係が成り立つ[2)]．実用上から見るとき，LL(2) と準 LL(2) の両文法の差はごく小さなものと思われる．さらに，解析表の大きさも Aho らにくらべて筆者らのは 1/120～1/400 ですむ[1)]．しかるに論文1) で示されているように，両者の解析速度を比較するとき，筆者らの解析速度はAho らのそれの 1/2 程度となる．

本論文は，論文2) で示される解析法の解析速度を2倍程度速くするための解析表の新しい構造とそれを用いた解析アルゴリズムを提案するものである．

2. 記法，用語

本論文中で用いられる基本的な記号や用語について述べる．なお，ここで述べないで使用される記号や用語については，文献3) に従うものとする．

［定義1］ 文脈自由文法 G を

$G=(N, \Sigma, P, S)$

とする．ここに，N，Σ はそれぞれ文法 G の非終端記号の集合ならびに終端記号の集合，P は生成規則の集合であり，S は出発記号である．

［記法1］ N^*，Σ^* 等は，それぞれ N，Σ 上の空列を含むすべての記号列の集合を表す．

［記法2］ とくに述べないときは N の要素を A，B，C，Σ の要素を a，b，c，$N\cup\Sigma$ の要素を X，Y，Z，Σ^* の要素を s, t, u, $(N\cup\Sigma)^*$ の要素を α，β，γ で表す．また，ε，ϕ はそれぞれ空列，空集合を表す．これらの記号は添字をつけて用いることもある．

［定義2］ 集合 $\mathrm{FIRST}_k(\alpha)$ は以下で定義される．

$$\mathrm{FIRST}_k(\alpha)=\{u \mid (\alpha \overset{*}{\Rightarrow} u\beta, \|u\|=k) \text{ または } (\alpha \overset{*}{\Rightarrow} u, \|u\|<k)\}$$

ただし，$\|u\|$ は u の長さを表す．また，$\alpha \overset{*}{\Rightarrow} u\beta$ は生成規則を0回以上使用する最左導出を表す．なお，$\overset{k}{\Rightarrow}$ は生成規則を k 回使用する最左導出を表し，とくに k=1 のとき $\Rightarrow$ で表す．本論文で扱う導出は，とくに断りがない限り，すべて最左導出である．

［定義3］ 集合 END-FOLLOW (X) は以下で定義

† The Structures of Parsing-tables for Semi-LL(2) Grammars to Raise Parsing Speed by KEIICHI YOSHIDA (Department of Computer Science, College of Engineering, Shizuoka University) and YOSHIKO TAKEUCHI (Department of Computer Science, Hamamatsu Polytechnic College).

†† 静岡大学工業短期大学部情報工学科
††† 浜松職業訓練短期大学校情報処理科

される.

END-FOLLOW (X)

$= \{A \mid (A \overset{*}{\Rightarrow} \alpha X\beta, X\in N, \beta \overset{*}{\Rightarrow} \varepsilon),$ または

$(A \overset{*}{\Rightarrow} \alpha BX, B\in N, X\in\Sigma)\}$

[定義 4][3] L_1, L_2 を Σ^* の部分集合とするとき, 演算子 $\oplus_k$ は以下で定義される.

$L_1 \oplus_k L_2$

$= \{w \mid$ ある $x\in L_1$, $y\in L_2$ に対して,

$\|xy\| \leqq k$ ならば $w=xy, \|xy\|>k$

ならば $w=u$, ただし $xy=uv, \|u\|=k\}$

[定義 5] 文脈自由文法 $G=(N,\Sigma,P,S)$ において $S \overset{*}{\Rightarrow} u_i AX\xi_i$ のとき, PF (partial-FOLLOW) は以下で定義される.

$PF_k(A,X)=\cup_i \mathrm{FIRST}_k(X\xi_i)$

[定義 6] 文脈自由文法 $G=(N,\Sigma,P,S)$ において, $A\to\alpha$, $A\to\beta$ を相異なる生成規則とするとき

$S \overset{*}{\Rightarrow} uAX\nu$

なるすべての $uAX\nu$ に対して

$(\mathrm{FIRST}_k(\alpha)\oplus_k PF_k(A,X))$

$\cap(\mathrm{FIRST}_k(\beta)\oplus_k PF_k(A,X))=\phi$

が成り立つとき, G は準 LL(k) 文法であるという.

なお, 文献 1), 2) で示した準 LL(2) 文法に対する解析表作成アルゴリズムならびに解析アルゴリズムを Mold と呼び, 4 章で述べる新しいそれらを Mnew と呼ぶことにする.

以下では Mold, Mnew の話を展開するのに必要ないくつかの定義ならびに記法等について述べる.

[定義 7] 文脈自由文法 $G=(N,\Sigma,P,S)$ を用いて次の拡大文法 G' を定義する.

$G'=(N',\Sigma',P',S'),\quad N'=N\cup\{S'\},$

$\Sigma'=\Sigma\cup\{\$\},\quad P'=P\cup\{S'\to S\$\$\}$

ただし, \$ はスタックの底を示す特別な記号. 以下では, この拡大文法 G' を用いて議論を展開するが, 便宜上, N', Σ', P' をそれぞれ N, Σ, P と読みかえるものとする.

[記法 2] 生成規則ならびに導出

$A\underset{p}{\to}\alpha,\quad \alpha\underset{p}{\Rightarrow}\beta$

における p は, それぞれ生成規則 $A\to\alpha$ につけられた固有の番号, ならびに最左導出 $\alpha\Rightarrow\beta$ で用いられた生成規則の番号を示す.

[定義 8] 生成規則の番号 p, q で表すとき $[\ ]p$ (q) または $[X]p(q)$ を π 型生成規則番号という. ここで, p, q は以下の導出を満足するものとする.

$S' \overset{*}{\Rightarrow} u'Y\alpha' \underset{q}{\Rightarrow} u'\xi A\gamma\alpha' \overset{*}{\Rightarrow} uA\alpha \underset{p}{\Rightarrow} u\beta\alpha$

ただし, $\alpha=\gamma\alpha'$ で X は α の最左端一記号を表す. π 型生成規則番号は解析表作成のために, 途中の文脈の保存としてのみ用いられる. また, π 型生成規則番号 $[X]p(q)$ において, $X=\varepsilon$, $q=\varepsilon$ のとき, $[X]p(q)$ は $[\ ]p(\ )$ を表すものとする.

[定義 9] π 型生成規則番号 $[X]p(q)$ から (q) を取り除いた生成規則番号 $[X]p$ を, τ 型生成規則番号という. ここでも, $X=\varepsilon$ のとき $[X]p$ は $[\ ]p$ を表すものとする.

[定義 10] 解析表 T は行, 列がそれぞれ $(N-\{S'\})\cup\Sigma$, Σ の要素によって名付けられたマトリクスとする. 解析表 T の A 行, a 列の要素を $T(A,a)$, また a 行, b 列の要素を $T(a,b)$ で表し, これら各要素には τ 型生成規則番号の集合または nil (空) が記入される.

[定義 11] 集合 Q, R はそれぞれ以下で定義される.

$Q=\{(A,p)\mid A\underset{p}{\Rightarrow}\alpha\overset{*}{\Rightarrow}\varepsilon\}$

$R=\{(A,a,p)\mid A\underset{p}{\Rightarrow}\alpha\overset{*}{\Rightarrow}a\}$

3. 新しい解析表の構造と解析アルゴリズム

3.1 Mold の解析表の構造と解析アルゴリズム

Mnew と Mold の比較をする上で, 最小限必要と思われる Mold の解析表の構造とそれを利用した解析アルゴリズムの基本的な考え方について述べる. 詳しくは文献 2) を参照されたい. まず, Mold の解析表 T の構造を図 1 に示す. この解析表の行および列はそれぞれ, $(N\cup(\Sigma-\{\$\}))$, $\Sigma-\{\$\}$ で名付けられているのではなく, これらの集合を構成する要素で名付けられている. また, この解析表は準 LL(2) の定義より, 以下の性質をもつ[2].

[性質 1]

$S' \overset{*}{\Rightarrow} uA\alpha \underset{p}{\Rightarrow} u\beta\alpha \overset{*}{\Rightarrow} uab\gamma\alpha$

$\leftrightarrow$ $[\ ]p\in T(A,a)$ かつ $\lceil\ \rceil p\in T(a,b)$

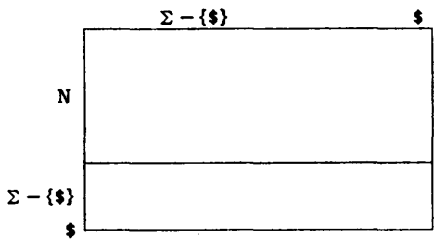


図 1 Mold の解析表 T の構造[1]

Fig. 1 Structure of parsing-table T constructed by Mold[1].

[性質 2]

$S' \fallingdotseq uAX\alpha \underset{p}{\Rightarrow} u\beta X\alpha \fallingdotseq uaX\alpha \fallingdotseq uab\delta$

$\leftrightarrow$ [ ]$p \in T(A,a)$ かつ [X]$p \in T(a,b)$

[性質 3]

$S' \fallingdotseq uAX\alpha \underset{p}{\Rightarrow} u\beta X\alpha \fallingdotseq uX\alpha \fallingdotseq uab\gamma$

$\leftrightarrow$ [X]$p \in T(A,a)$ かつ [X]$p \in T(a,b)$

[性質 4]

解析表の要素への記入は，前記3つの集合以外にはない．

次に，図1を用いた解析アルゴリズムの考え方を述べる．まず，解析アルゴリズムで用いる集合演算子$\sqcap$を以下に定義する．

[定義 12] 任意の $A \in N$, $a, b \in \Sigma$, 解析表要素 $T(X,Y)$ に対して，集合演算子$\sqcap$を以下のように定義する[2]．

$T(A,a) \sqcap T(a,b) = U$

とするとき，

① ([ ]$p \in T(A,a)$)$\wedge$([ ]$p \in T(a,b)$) なら，[ ]$p \in U$

② ([ ]$p \in T(A,a)$)$\wedge$([X]$p \in T(a,b)$) なら，[X]$p \in U$

③ ([X]$p \in T(A,a)$)$\wedge$([X]$p \in T(a,b)$) なら，[X]$p \in U$

とする．

④ ①，②，③の場合がなければ，Uは空集合とする．

MOLD の解析アルゴリズムの基本的な考え方を，$A \fallingdotseq ab\xi$ を例にとって示す．

① 解析表Tを用いて $U = T(A,a) \sqcap T(a,b)$ を求める．

② Uの値にもとづき，以下の処理を行う．

イ．$|U| = 0$ のとき．テキスト・エラー．

ロ．$|U| = 1$ のとき．$U = \{[\ ]p\}$ ならば生成規則番号pを選択する．$U = \{[X]p\}$ ならば解析スタックの上から2つ目の要素 NEXT とXを比較し，NEXT$=X$ ならば解析をつづける．NEXT$\neq X$ ならばテキスト・エラー．

ハ．$|U| \geqq 2$ のとき．[ ]$p \in U$ ならば生成規則番号pを選択する．[ ]$p \notin U$ ならば $X_i =$ NEXT を満足する p_i を $U = \{[X_1]p_1, [X_2]p_2, \cdots, [X_n]p_n\}$ $(n \geqq 2)$ の中から選択する．

こうした解析アルゴリズムが可能であるのは，以下に示す[定理 1][2]にもとづく．

[定理 1][2]

演算$\sqcap$により得られる集合Uが2つ以上の要素を含むとき，文法が準 LL(2) であるとすれば要素のいずれの2つをとり出しても以下の組合せのものは起こりえない．

(i) [ ]p, [ ]q, $p \neq q$

(ii) [ ]p, [X]q, $p \neq q$

(iii) [X]p, [X]q, $p \neq q$

3.2 MNEW の解析表の構造と解析アルゴリズム

3.1 節の基本的な考え方で述べたように MOLD による解析アルゴリズムでは，[定義 12] で定義されるUを求めるための集合演算が，Aho らの解析法にはない余分な計算である．しかも集合演算は一般に多くの時間を要するので，この部分の演算を解析時にやらなくてもよいようにするのが，解析高速化の1つの道であると考えられる．そこで，こうした計算を解析表の作成段階にすませてしまうことが考えられる．この結果，解析表の構造は次の T' のように変更される．

解析表 T' の構造

各行はNの要素で名付けられ，各列は長さが2以下のすべての終端記号列で名付けられたマトリクスである（**図2**）．図1の解析表Tと図2の解析表 T' の対応は以下のとおりである．

任意の $A \in N$, $a, b \in \Sigma$, $X \in N \cup \Sigma$ に対して，pを生成規則番号とするとき

[X]$p \in T(A,a) \sqcap T(a,b)$

$\leftrightarrow$ [X]$p \in T'(A, ab)$ (3.1)

解析表 T と T' の同等性

任意の非終端記号Aを含む導出

$S \fallingdotseq tAX_1\xi \underset{p_1}{\Rightarrow} t\alpha X_1\xi \fallingdotseq tab\zeta$

を考える．Aから ab の導出が解析表Tでは

$T(A,a) = \{[X_1]p_1, [X_2]p_2, \cdots, [X_i]p_i\}$

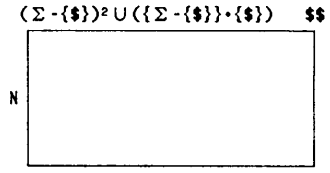


図 2 M_{NEW} の解析表 T' の構造
ただし，・は語の連接を示す演算子であり，$(\Sigma - \{\$\})^2$ は長さ2の終端記号列の集合を表す．

Fig. 2 Structure of parsing-table T' constructed by M_{NEW}.
Symbol • denotes an operator to concatenate strings, and $(\Sigma - \{\$\})^2$ denotes a set of terminal string with its length 2.

$T(a,b)=\{[X_1]p_1,[X_2]p_2,\cdots,[X_j]p_j\}$
ただし $i\leqq j$
とすると

$T(A,a)\sqcap T(a,b)=\{[X_1]p_1,\cdots,[X_i]p_i\}$ (3.2)

である．しかるに，MOLD の解析アルゴリズムから，$[X_1]p_1$ が選ばれるので，p_1 の番号をもつ生成規則が選択される．一方，MNEW の解析表 T' においては，(3.1)と(3.2)より

$T'(A,ab)=\{[X_1]p_1,\cdots,[X_i]p_i\}$

となる．後述の MNEW の解析アルゴリズムからやはり，$[X_1]p_1$ が選ばれるので，T の場合と同じ生成規則番号が選ばれる．このことはすべての導出について言えるから，解析表 T' において得られる解析列は解析表 T において得られる解析列と同じものであるといえる．また，入力テキストに誤りが含まれている場合，たとえば A から ab が導出できないときは

$T(A,a)\sqcap T(a,b)=\phi$

となるので，(3.1)より

$T'(A,ab)=\phi$

となり，解析表 T においてエラーとなるものは解析表 T' においてもエラーとなる．

解析時間の短縮

解析表 T による解析では，A から ab を導出する場合 $T(A,a)$ と $T(a,b)$ の2つの要素を探索し，かつ $\sqcap$ を演算しなければならない．それに対して，解析表 T' による解析では，$T(A,ab)$ の要素のみの探索ですむから

（1） 探索回数が一回ですむ，
（2） $\sqcap$ を演算する必要がない．

この2つの理由により，解析時間が大幅に減少することが想像される．

以下に新しい解析表にもとづく解析アルゴリズムを示すが，まず，解析アルゴリズムの説明に必要な2，3の事柄について述べる．

（1） 解析スタック R：解析の経過を記憶するためのスタックで，初期値は $S\$\$$ である．
（2） 入力テキスト記憶用領域 M：プログラム・テキストを記憶するための領域．プログラム・テキストの終りには $\$\$$ が付加される．
（3） $\mathrm{CURRENT}_1$, $\mathrm{CURRENT}_2$：入力テキスト M の目下解析の対象となっている要素，ならびにその右隣りの要素を格納するための領域．
（4） TOP, NEXT：スタック R の一番上の要素，ならびに2番目の要素を表す．
（5） I：入力テキストの I 番目の要素 $M(I)$ を表すための整数型変数．
（6） V：生成規則番号 p を保存するための整数型変数．

また，解析上必要とする固有の命令には以下のものがある．

（i） POP：解析スタック R より，TOP を1回に1個取り出す．したがって，NEXT が先頭になる．
（ii） PUSH (p)：解析スタック R に生成規則番号 p の右辺全体を，右辺の左端がスタック R の TOP になるように押し込む．

解析の手続きとその意味づけは以下のとおりである．

（1） TOP=$\mathrm{CURRENT}_1$ かつ NEXT=$\mathrm{CURRENT}_2$ ならば，POP を2度行い，M のポインタを右へ2つずらす．
（2） TOP=$\mathrm{CURRENT}_1$ かつ NEXT≠$\mathrm{CURRENT}_2$ ならば POP し，M のポインタを右へ1つずらす．
（3） TOP≠$\mathrm{CURRENT}_1$ かつ TOP が終端記号ならばテキスト・エラー．
（4） TOP が非終端記号ならば集合 U を求める．
$U=T(\mathrm{TOP},\mathrm{CURRENT}_1)$
$\sqcap T(\mathrm{CURRENT}_1,\mathrm{CURRENT}_2)$
（イ） $|U|=0$ のとき，つまり $U=\phi$ のときはテキスト・エラー．
（ロ） $|U|=1$ のとき，もし $U=\{[\ ]p\}$ であれば生成規則番号 p を選択し，POP し，PUSH (p) を行う．また，$U=\{[X]p\}$ であれば NEXT と X を比較し NEXT=X であれば POP し，PUSH (p) を行う．NEXT≠X であればテキスト・エラー．
（ハ） $|U|\geqq 2$ のとき
① $[\ ]p\in U$ のとき，生成規則番号 p を選択し，POP し，PUSH (p) を行う．
② $[\ ]p\notin U$ のとき，NEXT=X_i を満足する p_i を選択し，POP し，PUSH (p_i) を行う．
③ ①，②のいずれをも満足しないときはテキスト・エラー（[定理1]）．

さらに，[性質1]から[性質4]のことを配慮すると，上記の手続きを実現する解析アルゴリズムは以下のようになる．

次に，解析アルゴリズムを示す．陰のある部分がも

MNEWの解析アルゴリズム

```
begin
/* M(i)はテキストMのi番目の要素を示す */
  R←'S$$'; /* スタックの初期化 */
  M←text'$$'; /* Mの初期化 */
  i←1;
  CURRENT1←M(i);
  CURRENT2←M(i+1);
  repeat
    if TOP=CURRENT1 and NEXT=CURRENT2 then
      begin
        POP; POP;
        i←i+2;
        CURRENT1←M(i);
        CURRENT2←M(i+1);
      end
    else
      if TOP=CURRENT1 then
        begin
          POP;
          i←i+1;
          CURRENT1←CURRENT2;
          CURRENT2←M(i+1);
        end
      else
        if TOP∈Σ then  text error
        else
          begin
            find U such that U(TOP・CURRENT1・CURRENT2);/* ・は語の連接を示す*/
            case |U|=0 : text error;
                 |U|=1 : if there exists [ ]p in U then  select p and y←'p'
                          else
                            if there exists [NEXT]p in U then
                                                  select p and y←'p'
                            else  text error;
                 |U|≧2: if there exists [ ]p in U then  select p and y←'p'
                          else
                            if there exists [NEXT]pi in U then
                                                  select pi and y←'pi'
                            else  text error;
            end of case;
            POP; PUSH(y);
          end
  until TOP='$' and CURRENT1='$'
end.
                                                  [End of Algorithm]
```

との解析アルゴリズム[2]を変更した箇所である．

4. 適 用 例

[例1] 以下の準 LL(2) 文法 G_1 に対して，MOLD の解析表 T ならびに本論文で提案する MNEW の解析表 T' を求めると，それぞれ図3，図4が得られる．

1. $S \rightarrow aAaa$　　4. $A \rightarrow b$
2. $S \rightarrow bAba$　　5. $A \rightarrow \varepsilon$
3. $S \rightarrow Aa$

これらの2つの解析表による解析過程を以下に示す．

[例1 a] 正しいテキスト bba に対する解析

解析表 T による解析[2]

まず，入力テキストのはじめの2つ bb を考えると，図3の解析表 T より

$T(S, b) = \{[\ ]2, [\ ]3\}$

$T(b, b) = \{[\ ]2, [b]4\}$

$\therefore\ T(S, b) \cap T(a, b) = \{[\ ]2\}$

ゆえに，生成規則番号2を選択し PUSH (2) を行う．つまり，スタック R の S を生成規則番号2の右辺でおきかえる．この手順を前述の記法にならうと以下のように表される．

$(bba\,\$\$, S\,\$\$, \varepsilon)$

$\vdash (bba\,\$\$, bAba\,\$\$, 2)$

次に，入力テキストとスタックを調べると，

$\mathrm{TOP} = \mathrm{CURRENT}_1 = \{b\}$

であるから，POP を1度行うことにより b をスタック R より降ろし，M のポインタを右へ1つずらす．これによりスタック R の状態は以下のようになる．

$\vdash (ba\,\$\$, Aba\,\$\$, 2)$

次に，再び図3より

$T(A, b) = \{[\ ]4, [b]5\}$

$T(b, a) = \{[\ ]3, [a]4, [b]5\}$

$\therefore\ T(A, b) \cap T(b, a) = \{[a]4, [b]5\}$

となり，適用すべき生成規則が4と5の2つが対象となるが，NEXT=b なので $[b]5$ が適用される．ゆえに

$\vdash (ba\,\$\$, \varepsilon ba\,\$\$, 25)$

以下，同様の記法にならうと，次のようになる．

$\vdash (a\,\$\$, a\,\$\$, 25)$

$\vdash (\$\$, \$\$, 25)$

解析表 T' による解析

まず，S から bb の導出を考えると，図4より

$T'(S, bb) = \{[\ ]2\}$

であるから，

	a	b	$
S	[ ]1 [ ]3	[ ]2 [ ]3	
A	[a]5	[ ]4 [b]5	
a	[ ]1 [a]5	[ ]1	[$]3 [a]5
b	[ ]3 [a]4 [b]5	[ ]2 [b]4	
$			

図3 文法 G_1 に対する MOLD の解析表 T
Fig. 3 Parsing-table T MOLD constructed for grammar G_1.

	aa	ab	ba	bb	a$	b$	$$
S	[ ]1	[ ]1	[ ]3	[ ]2	[$]3		
A	[a]5		[a]4 [b]5	[b]4	[a]5		

図 4 解析速度を高速化するための M_{NEW} の解析表 T'
Fig. 4 Parsing-table T' M_{NEW} constructed for raising parsing speed.

$(bba\,\$\$, S\,\$\$, \varepsilon) \vdash (bba\,\$\$, bAba\,\$\$, 2)$

次に，TOP=CURRENT$_1$= $\{b\}$ であるから，

$\vdash (ba\,\$\$, Aba\,\$\$, 2)$

次に，再び図 4 より

$T'(A, ba) = \{[a]4, [b]5\}$

となるが，NEXT=b なので $[b]5$ が選択される．ゆえに

$\vdash (ba\,\$\$, \varepsilon ba\,\$\$, 25)$

以下，同様にして

$\vdash (a\,\$\$, a\,\$\$, 25)$

$\vdash (\$\$, \$\$, 25)$

解析表 T と解析表 T' のそれぞれによる解析から分かるように，前者では，たとえば $A \Rightarrow ab\xi$ に対して $T(A, a)$, $T(a, b)$ の 2 回の位置ぎめの時間を要するが，後者では $T'(A, ab)$ の 1 回ですむ．さらに，前者では Π の計算時間を要するが，後者ではそれを必要としない．これらの違いが解析時間の大幅な減少の理由となっている．

5. 性能評価

筆者らの提案する解析アルゴリズムの性能を評価するため，文献 1) の方法ならびに Aho らの解析法と，以下の条件の下で比較検討した．

- 使用機種：SUN-3 モデル 60M
- 使用 OS および言語：UNIX/PASCAL
- 対象言語：PASCAL−[4] (pascal minus)
  ISO PASCAL[5]
  JIS 基本 BASIC[6]

プログラムはすべて同一人が作成し，個人差による影響をできるだけ少なくした．

5.1 構文解析速度

3 つの解析アルゴリズム (M_{OLD}, M_{NEW}, Aho ら) の性能比較を以下の 3 つの文法に従って実測した．測定は SUN PASCAL がもっている CPU 使用時間測定用の命令を用い，計算機はスタンド・アロンの状態で使用した．

実験は M_{NEW} による方法，M_{OLD} による方法，Aho らによる方法の 3 つの方法を用いて，各々の構文解析の速さを比較した．

表 1 [実験 1] の結果．前出の簡単な文法を用いて，連続 200 回のくりかえし解析をしたときの合計所要時間（実測値）．（単位：秒）
Table 1 Result of experiment 1. Total time required for 200 parsings for an input string using a simple semi-LL(2) grammar described before. (time unit: second)

	方法 / データ	M_{OLD} の方法による解析時間	M_{NEW} の方法による解析時間	Aho らの方法による解析時間
解析時間	$(ab)^{16}cab^{\dagger}$	1.730	0.853	0.810
割合	$(ab)^{16}cab^{\dagger}$	1.65	1	0.95

†(注) $(ab)^{16}cab = abababababababababcab$

[実験 1] 小さな準 LL(2) 文法を用いた場合．

サンプル・プログラム

abababababababababcab

サンプル・プログラムの記述に用いた文法

1. $S \to AZ$　　7. $D \to \varepsilon$
2. $A \to BY$　　8. $E \to \varepsilon$
3. $B \to \varepsilon$　　9. $F \to c$
4. $B \to CXB$　　10. $X \to b$
5. $C \to DE$　　11. $Y \to FD$
6. $D \to a$　　12. $Z \to ab$

このサンプル・プログラムの 1 回当りの解析時間はすこぶる短いので，連続 200 回のくりかえし解析をしたときの合計所要時間で示した（**表 1**）．

[実験 2] PASCAL− 文法を用いた場合．

サンプル・プログラム

- 8 王妃 (eight-queen) 問題プログラム (46 ステップ)
- クイック・ソートプログラム (42 ステップ)

サンプル・プログラムの記述に用いた言語

- PASCAL−

2 つのサンプル・プログラムの 1 回の解析時間はいずれも短いので，測定誤差を小さくするために，連続 100 回のくりかえし解析をしたときの合計所要時間で示した（**表 2**）．

[実験 3] PASCAL−$^+$ 文法を用いた場合．

サンプル・プログラム

- M_{NEW} の解析プログラム (185 ステップ)
- M_{NEW} の解析表作成プログラム (1,007 ステップ)

サンプル・プログラムの記述に用いた言語

- PASCAL−$^+$

表 2 [実験2] の結果. PASCAL− 文法を用いて, 連続100回のくりかえし解析をしたときの合計所要時間 (実測値). (単位: 秒)
Table 2 Result of experiment 2. Total time required for 100 parsings for an input string using PASCAL−grammar. (time unit: second)

	方法 / データ	M_{OLD} の方法による解析時間	M_{NEW} の方法による解析時間	Aho らの方法による解析時間
解析時間	8 王 妃	12.64	6.50	6.45
	クイック・ソート	10.18	5.21	5.17
割合	8 王 妃	1.94	1	0.99
	クイック・ソート	1.95	1	0.99

表 3 [実験3] の結果. PASCAL−$^+$ 文法を用いて, 連続100回のくりかえし解析をしたときの合計所要時間 (実測値). (単位: 秒)
Table 3 Result of experiment 3. Total time required for 100 parsings for an input string PASCAL−$^+$ grammar. (time unit: second)

	方法 / データ	M_{OLD} の方法による解析時間	M_{NEW} の方法による解析時間	Aho らの方法による解析時間
解析時間	M_{NEW} の解析プログラム	54.04	27.23	27.43
	M_{NEW} の解析表作成プログラム	288.42	142.57	143.91
割合	M_{NEW} の解析プログラム	1.98	1	1.007
	M_{NEW} の解析表作成プログラム	2.03	1	1.009

上記のプログラムはいずれも, 本来, SUN PASCAL で記述されているが, これを PASCAL− 上で動かすため, PASCAL− で不足している機能 (ポインタ, for 文, repeat 文, 入出力など) を追加もしくは等価な機能への書きかえ (for 文, repeat 文→while 文) などを行った. この結果, 生成規則数は98本から126本に, 非終端記号数は54個から69個に, 終端記号数は46個から53個にそれぞれ増えた. 便宜上, この文法を PASCAL−$^+$ と呼ぶことにする.

この解析時間も1回当りがすこぶる短いため, 連続100回のくりかえし解析をしたときの合計所要時間で示した (**表3**).

さて, 3つの方法に対する解析表 T' の大きさを比較すると, **表4**をうる. 表4と[実験1], [実験2], [実験3]の結果から, 以下の結論が得られる.

表 4 3つの文法に対する M_{NEW} の方法による解析表 T' と生成規則表の大きさ (単位: バイト)
Table 4 Comparison of areas for parsing tables and productions for three different grammars using M_{NEW} method. (size unit: byte)

	[実験1]に用いた文法	[実験2]に用いた文法 PASCAL−	[実験3]に用いた文法 PASCAL−$^+$
解析表 T'	880	465,706	786,402
生成規則表	240	1,960	2,520
合 計	1,120	467,666	788,922
大きさの割合	0.002	1	1.7

- M_{NEW} による方法と Aho らによる方法との解析速度の比 Aho/M_{NEW} は, 文法が小さいうちは1よりやや小さいが, 文法が大きくなるとほぼ1になると考えられる. つまり, 両者の解析速度はほぼ同じになると考えられる.
- M_{OLD} による方法と M_{NEW} による方法の解析速度の比 M_{OLD}/M_{NEW} も, 文法が小さいうちは2よりやや小さいが, 文法が大きくなるとほぼ2になると考えられる. つまり, 解析表ならびに解析アルゴリズムの工夫により, M_{OLD} による方法の解析速度を2倍程度高速化できた.

5.2 記憶容量

解析表 T の構造 (図1) と解析表 T' の構造 (図2) の大きさについて, 比較してみる.

PASCAL− 文法, ISO PASCAL 文法を各々準 LL(2) 文法にかきかえた結果, 生成規則数などは**表5**のようになった. これらの各文法に対して, 解析表 T の構造, 解析表 T' の構造, Aho らの解析表の構造 (**図5**) の各々について, 解析表の大きさと生成規則表の大きさを求めた (**表6**).

解析表 T も解析表 T' も, 図3, 図4の例から分かるように解析表への要素の書き込みは, プログラム上

表 5 PASCAL−, ISO PASCAL 文法, JIS 基本 BASIC 文法を準 LL(2) で表したときの生成規則数, 非終端記号数, 終端記号数
Table 5 Numbers of productions, nonterminals, and terminals for the languages PASCAL−, ISO PASCAL, and JIS minimal BASIC by semi-LL(2) representation.

	PASCAL− 文法	ISO PASCAL 文法	JIS 基本 BASIC 文法
生成規則数	98	247	148
非終端記号数	54	151	83
終端記号数	46	61	68

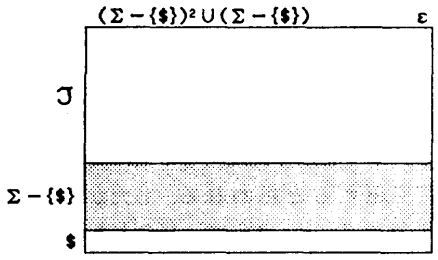


図 5 Aho らの解析表の構造
$\mathcal{T}$ は生成規則の書きかえにより発生した新しい生成規則の非終端記号の集合．記憶容量の関係で斜線部は生成しなかったが，この部分は表に含めず解析ルーチンで補うことができる（スタックからの降ろし，エラー発見，など）．なお，5.2 節記憶容量の比較のところでは，斜線部分の領域は除外してある．

Fig. 5 Structure of Aho-Ullman's parsing-table. $\mathcal{T}$ denotes a set of new non-terminal symbols generated by translating given production rules into Aho-Ullman's. The shadowed portion possible to be excluded from the figure because the parser can replace its function. Authors did not implement the portion due to the shortage of memory capacity. We did not count the portion as specified in the comparison of table sizes in 5.2.

ではすべて表の要素を根とするリストでつないで実現している．また Aho らの場合，生成規則の書きかえにより筆者らの生成規則数の8〜16倍に増えるので[1]，解析表の大きさを比較する場合，この生成規則表の大きさを無視することはできない．

さて解析速度を高速化するための解析表 T' の大きさを，解析表 T の大きさと比較してみると PASCAL− の場合で約 11 倍，ISO PASCAL の場合で約 17 倍程度前者が大きくなる（表7）．しかしそれでもなお，Aho らの解析表にくらべると 1/15〜1/30 程度の大きさですむことが分かる．

また，T と T' の大きさの比は以下の式で計算することもできる．

Σ および N の要素数をそれぞれ m，n とすると，図1，図2に対するそれぞれの欄の数 E_1', E_2' は

$$E_1'=(n+m)\cdot m$$
$$E_2'=n\{(m-1)^2+(m-1)+1\}$$
$$=n(m^2-m+1)$$

$m^2-m\gg 1$，$m\gg 1$ であるから

$$E_2'\fallingdotseq n(m^2-m)$$
$$\fallingdotseq m^2n$$

表の大きさを考えるとき，表の1つの欄に複数個の要素が入るので，そのことを考慮しなければならない．図1，図2において，表の1つの欄からポインタでつながれる平均的要素数を α_1, α_2 とし，ポインタでつながる要素1つを実現するのに要する領域の大きさを表の欄の場合の β 倍とすると，T および T' の大きさは

$$E_1=E_1'+\text{ポインタ部分}$$
$$=E_1'+E_1'\times\alpha_1\times\beta$$
$$=(n+m)m(1+\alpha_1\beta)$$
$$E_2=E_2'+\text{ポインタ部分}$$
$$=E_2'+E_2'\times\alpha_2\times\beta$$
$$=m^2n(1+\alpha_2\beta)$$

である．

$\gamma=m/n$ とすると

$$E_1=n^2(1+\gamma)\gamma(1+\alpha_1\beta)$$
$$E_2=\gamma^2n^3(1+\alpha_2\beta)$$

表 6 解析表ならびに生成規則表の大きさ（実測値）（単位：バイト）

Table 6 Sizes of experimental parsing-tables and production-tables. (memory size unit: byte)

			PASCAL−	ISO PASCAL
筆者ら	解析表 T	表部分	18,400	51,728
		ポインタ部分	20,830	76,290
		計	39,230	128,018
	生成規則表		1,960	4,940
	合計		41,190	132,958
筆者ら	解析表 T'	表部分	447,336	2,211,244
		ポインタ部分	18,370	68,930
		計	465,706	2,280,174
	生成規則表		1,960	4,940
	合計		467,666	2,285,114
Ahoら	解析表		7,091,104	69,412,560
	生成規則表		32,160	159,880
	合計		7,123,264	69,572,440

表 7 解析表等の大きさの比較（実測値）

Table 7 Experimental size-ratios of parsing-tables.

	M_{OLD} : M_{NEW}		M_{NEW} : Aho	
	PASCAL−	ISO PASCAL	PASCAL−	ISO PASCAL
解析表のみ	1 : 11.9	1 : 17.8	1 : 15	1 : 30
解析表＋生成規則表	1 : 11.4	1 : 17.2	1 : 15	1 : 30

表 8 α_1, α_2, β, γ のおおよその値（実測値）
Table 8 Approximate values of α_1, α_2, β and γ from experimental tables.

	PASCAL−		ISO PASCAL	
	解析表T	解析表T'	解析表T	解析表T'
表のエントリ数	4,600	111,834	12,932	552,811
ポインタ数	2,083	1,837	7,629	6,893
α	$\alpha_1=0.45$	$\alpha_2=0.02$	$\alpha_1=0.59$	$\alpha_2=0.01$
γ	0.85		0.4	
β	3			

$$E=\frac{E_2}{E_1}=\frac{n\gamma(1+\alpha_2\beta)}{(1+\gamma)(1+\alpha_1\beta)} \qquad (5.1)$$

表5よりγは

$\gamma=0.4$ (ISO PASCAL)

$\gamma=0.85$ (PASCAL−)

が得られる．また，α_1, α_2 についてはポインタ数を実測した結果，表8を得た．さらに，異種のプログラミング言語 JIS 基本 BASIC についても γ, α_1, α_2 を求めた結果

$\gamma=0.82$, $\alpha_1=0.62$, $\alpha_2=0.02$

となった．これらの値を比較してみると，γについては 0.4〜0.85 と，2倍以上の差になっている．これは言語の性質のちがいによるもので，多くの機能をもつ言語ほどγは小さくなると言えるだろう．α_1, α_2 は言語の性質によって多少値が異なってくるが，α_1, α_2 の定義から必ず $\alpha_1 \geqq \alpha_2$ である．

また，実測に使用した計算機システムでは欄1つの実現に4バイトを要し，ポインタでつながる要素1つの実現に対して 12 バイトを要しているので，βの値は3となる．α_1, α_2, γ のそれぞれの値は表8のものを用いた．表9はこうして求めた計算値と先の実測値の比較を示すものである．

次に，筆者らの新しい解析表と Aho らの解析表の大きさを図2，図5をもとに考察する．図5（Aho らの解析表）の行の大きさのうち，$\mathcal{I}$の個数は各非終端記号について，それに右接しうる文脈の個数の総和になる．したがって，1つの非終端記号について唯一の文脈しか存在しないときに限り $\|\mathcal{I}\|=\|N\|$ となるが，一般には1つの非終端記号について複数個の文脈が存在し，かつまた非終端記号の数は実用の文法では 100 個[1] を超えると思われる．したがって，$\|\mathcal{I}\| \gg \|N\| \gg 0$ である[1),2)]．さらに，列の大きさは両解析表とも同じであるから，新しい解析表は Aho らの解析表より十分に小さいことが期待される（表7）．ここで，Nは与えられた生成規則の非終端記号の集合であり，$\mathcal{I}$ は Aho らのアルゴリズムに従って生成規則を書きかえたときに新しく生成される非終端記号の集合である．

表 9 解析表の大きさの実測値と計算値の比較
Table 9 Experimental ratios and computed ratios of parsing-table sizes.

	実測値	計算値
	$T:T'$	$T:T'$
PASCAL−	1:11	1:11
ISO PASCAL	1:17	1:16

5.3 解析表作成時間

解析表 T' は，解析表 T から［定義 12］で示される U を計算することにより再構成される．再構成のアルゴリズムを以下に示す．

T から T' の再構成のアルゴリズム

```
begin
  for each A∈N and a,b∈Σ do
    if T(A,a)∩T(a,b)≠φ then
      T'(A,ab)←T'(A,ab)∪
              {[X]p|[X]p∈T(A,a)∩T(a,b)
                   かつ [X]p∉T'(A,ab)}
end
```

T から T' への再構成の時間は，Tを構成する時間にくらべてきわめて小さい．そこで，この再構成の時間の損失は無視できる程度のものと考えられる．このことを示すために，論文 1) で示された T の生成アルゴリズムの中で最も多くの時間を要すると思われる STEP 6 のアルゴリズムを取りあげ，それと再構成のためのアルゴリズムの時間を比較する．まず，STEP 6 のアルゴリズムを観察すると[1)]，そのループの構造から計算量のオーダは

$O(\text{step 6})=n(m+n)^2$

であることが容易に分かる．一方，再構成に要する計算量のオーダは

$O(T\rightarrow T')=m^2n$

である．ゆえに，Tを構成する時間に対して再構成が占める時間の割合τは

$$\tau=\frac{O(T\rightarrow T')}{O(\text{step 6})}=\frac{m^2n}{n^2m(n+m)^2}=\frac{\gamma}{(m+n)^2}$$

よりは小さくなる．一般に $\gamma \leqq 1$ であるから，τはほとんどゼロに近い値になる．つまり，解析表作成に対

表 10 新しい解析表構成の所要時間(実測値)(単位：秒)
Table 10 Experimental times required for constructing new parsing-tables. (time unit: second)

	PASCAL−	ISO PASCAL
解析表 T を構成する所要時間 A	11.87	192.16
解析表 T' にするための再構成の所要時間 B	0.41	2.0
B/A	0.03	0.01
計 $C=A+B$	12.28	194.16
Aho らの解析表を構成する所要時間 D	128.78	2,389.93
$C:D$	1:10.5	1:12.3

して占める書きかえ時間の割合はほとんど無視することができる.

表 10 は実測による解析表作成の所要時間と書きかえのための所要時間を示す. 表 10 の B/A は上記の推定の正しさを裏付けるものである. したがって, 書きかえ時間を含めてもなお Aho らの解析表作成所要時間のほぼ 1/10 程度におさまると言える.

6. むすび

論文 2) で提案されている準 LL(2) 文法に対する解析表作成法は Aho-Ullman の作成法にくらべて, 解析表が小さい (1/120〜1/400), 表作成の時間が短い (1/10 程度), などといった特徴をもつ一方で, 解析時間が Aho-Ullman の方法により 2 倍程度遅くなる短所があった.

本論文では, 解析速度を上げるために, 従来 (MOLD) の解析表 T にわずかに手を加えた新しい (MNEW) 解析表 T' の構造を提案し, さらに MOLD の解析アルゴリズムにやや手を加えた解析アルゴリズムを提案した. MOLD, MNEW, Aho ら, これら 3 つの方法の性能比較を 3 種類の実験を通して行った. その結果, MOLD による方法にくらべて MNEW による方法は文法が大きくなると MNEW の方法による解析速度は 2 倍程度速くなることが分かった. また, Aho らのものにくらべて, 文法が小さいうちは MNEW の方法よりも Aho らの方法のほうがやや解析速度は速いが, 文法が大きくなるとほぼ同程度になることが分かった.

さらに, これらの解析表の大きさについて PASCAL− および ISO PASCAL の 2 つの文法に対して実験的に検証した結果, 解析表の大きさは従来のものより 11〜17 倍程度大きくなったが, Aho-Ullman にくらべると 1/15〜1/30 程度ですむことが分かった.

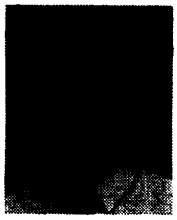
さらに, 解析表 T から解析表 T' を再構成するための所要時間は, 前者を構成する時間にくらべてほとんど無視できるほど小さいことを示した. したがって, Aho らとの所要時間の比はやはり解析表 T の作成の 1/10 程度である.

謝辞 本研究をすすめるにあたり, ご指導をいただいた東京理科大学・井上謙蔵教授に心から謝意を表します.

参考文献

1) 吉田, 竹内：準 LL(2) 文法に対する構文解析表の作成アルゴリズム, 情報処理学会論文誌, Vol. 31, No. 6, pp. 916-929 (1990).
2) 吉田, 竹内：準 LL(2) 文法に対する解析表の構造と解析アルゴリズム, 情報処理学会論文誌, Vol. 31, No. 9, pp. 1354-1365 (1990).
3) Aho, A. V. and Ullman, J. D.: *The Theory of Parsing, Translation, and Compiling*, Vol. I, pp. 348-356, Prentice-Hall (1972).
4) Hansen, P. B.: *Brinch Hansen on Pascal Compilers*, pp. 15-16, Prentice-Hall (1985).
5) Jensen, K. and Wirth, N.: PASCAL-User Manual and Report (3rd ed.), ISO PASCAL Standard, pp. 215-220, Springer-Verlag (1975).
6) 電子計算機プログラム言語 基本 BASIC, JIS C 6207, 日本規格協会 (1982).

(平成 3 年 1 月 28 日受付)
(平成 3 年 12 月 9 日採録)



吉田 敬一（正会員）

昭和13年生．昭和37年法政大学工学部電気工学科卒業．日本電気(株)を経て，現在，静岡大学工業短期大学部・教授（情報工学科）．工学博士（慶応義塾大学）．コンパイラの構文解析法，自然言語処理，情報処理教育に関心を持っている．著書「コンパイラの理論と応用」（共訳，学研），「コンピュータ・サイエンスのための言語理論入門」（共訳，共立出版）など．日本ソフトウェア科学会，ACM，ACS（オーストラリア）各会員．

竹内 淑子（正会員）

昭和29年生．昭和55年静岡大学工業短期大学部・情報工学科卒業．(株)東京システム技研（コンパイラ開発担当）を経て，現在，浜松職業訓練短期大学校情報処理科教官．静岡大学工業短期大学部非常勤講師．コンパイラの自動生成に関心を持っている．著書「BASIC プログラミングのすべて」（共訳，一橋出版）．日本ソフトウェア科学会，IEEE 各会員．