ＬＲ(1)構文解析表の動的構成法

2R－1

美馬　秀樹，　安藤　一秋，　青江　順一
徳島大学工学部知能情報工学科

1．まえがき

本論文では，LR(1)クラスの文法に対してパージングテーブルをインクリメンタルに構成する手法について述べる．

機械翻訳システムやコンパイラ等の言語処理システムを構築する場合，事前にターゲットとなる言語の文法を設計する必要がある．一般に，LR構文解析のようなテーブル駆動型の言語処理システムでは，文法の設計／修正の段階で，テーブルの生成，パージングテスト，文法の修正のサイクルを繰り返す必要があり，また，１つの文法の修正に対してもテーブル全体を構成し直さなければならないため，非常な時間的，空間的労力を要する．

このような問題に対し，文脈自由文法に基づく文法を効率的に構築するために，LRパーサーをインクリメンタルに実行する手法が種々提案されている[2][3]．しかし，従来の手法のほとんどは状態遷移表内のアイテムをすべて求め直す必要があるため，効率的な手法とは言えない．

本論文では，LR(1)文法に対し，パージングテーブルをインクリメンタルに構成する手法を提案する．本手法により，少ない資源で効率的かつ迅速に文法を構築することが可能となる．また，本手法は，高速にパージングテーブルを再構成するため，文法の自動学習の一手法としても有効である．

本稿では，まず，２．でLRパーサージェネレーションの概要を述べ，３．でLRパージングテーブルのインクリメンタルな構成手法を提案する．４．では種々の数の文法に対し，全体を構成し直す場合との速度の比較により本手法の有効性を示し，５．でまとめと今後の課題を述べる．

2．LRパーサージェネレーション

2．1　LRパージングの概要

図1に一般的なテーブルジェネレーションのフローを示す．LRパージングでは，図1により，予め定義した文法よりパージングテーブルを生成しておく必要がある．LRパージングテーブルの生成とは，概念的には，定義した文法に対してその状態遷移表をトップダウンに構成することである．したがって，ここでは状態遷移表よりパージングテーブルを作成する手順の詳細は省略する．

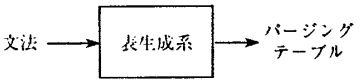

図１．パージングテーブルの生成

2．2　LR(1)状態遷移表の構成

文脈自由文法をG=(V_N, V_T, P, S)と表す．ここで，V_N, V_Tはそれぞれ非終端語，終端語の有限集合である．PはA::=αなる生成規則の有限集合である．S はV_Nの要素で開始記号を表す．まず，α* を文法記号の任意の列とすると，終端語の集合FIRST(α*) を次のように定義する．

FIRST(α*) = {a | a∈V_T, $\alpha \overset{*}{\Rightarrow}$ a･･･}

ただし$\alpha \overset{*}{\Rightarrow} \varepsilon$ならFIRST($\alpha$*)に$\varepsilon$を追加する．尚，$\overset{*}{\Rightarrow}$は生成規則による導出における反射推移閉包を示している．

文法Gに対し，LR(1)による状態遷移表は次により生成され，パージングテーブルが出力される．尚，LR(1)項中の$は文末を表す記号である．

入力: 文法G;
出力: LR(1)によるパージングテーブルT;

procedure PTG(G):

```
begin
G := G∪{S' ::= S}
C := CLOSURE({[S' ::=・S,$]})
  repeat
    for GOTO(I,X)が空でなくしかもCに含まれていないような,Cの各項の集合Iおよび各文法記号X do
        GOTO(I,X) を C に加える.
  until Cに加えることができる項がもはやない
CよりパージングテーブルTを作成する.
end.
procedure GOTO(I,X):
begin
  [A ::= α・Xβ,a]がIに含まれるような，項
  [A ::= αX・β,a] の集合をJとする.
  return CLOSURE(J)
end.
procedure CLOSURE(I):
begin
  repeat
    for I の各項[A::=α・Bβ,a]，各生成規則 B::= γ
及びFIRST(βa)の各終端記号b. ただし，[B ::=・γ,b]が
Iに含まれないもの do
        [B ::=・γ,b]をIに加える.
  until I に加えることができる項がもはやない.
  return I
end.
```

例えば，次の文法G1に対して，LR(1)による状態遷移表は図2のようになる．

G1:={ S ::= C C, C ::= c C, C ::= d }

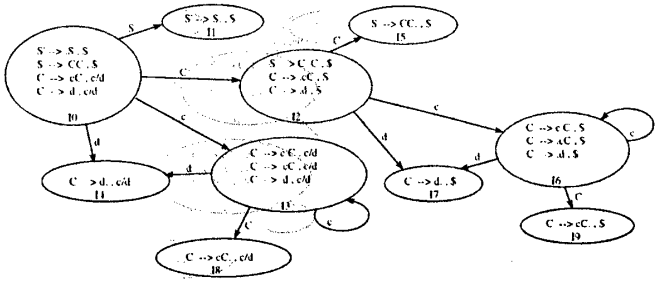

図２．G1に対するLR(1)状態遷移表

3．インクリメンタルテーブルジェネレーション

通常のLR(1)テーブルジェネレーションでは，１つの文法の修正に対しても上記のアルゴリズムにより状態遷移表全体を構成し直さなければならないため，非常な時間的，空間的労力を要する．そのような問題に対し，本章では，LR(1)パージングテーブルをインクリメンタルに構成する手法を提案する．本手法により，文法の追加に対して，高速にパージングテーブルを再構成することが可能となる．

図1に示した通常のテーブルジェネレーションに対し，図3に本手法によるインクリメンタルジェネレーションのフローを示す．図3に示すように，インクリメンタルジェネレーションでは，追加後の文法集合に対し，以前に構成したパージングテーブルとカーネル項のみの情報を用いて新たな状態遷移表を再構成し，パージングテーブルを作成する．ここで，カーネル項の情報を使用するのは，再構成により変更された状態の併合可能性の判定のために，それらの状態を完全に再現する必要があるためである．よって，

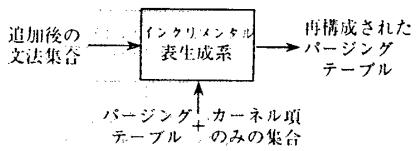

図３．インクリメンタルテーブルジェネレーション

Incremental Generation of LR(1) Parse Tables
Hideki MIMA, Kazuaki ANDO, Jun-ichi AOE
Faculty of Engineering, The University of Tokushima

インクリメンタル処理を継続するならば再構成後のカーネル項の集合も出力とする．

再構成処理のほとんどは，新たに追加される文法に関するCLOSUREと先読み情報の再計算となる．したがって，本手法では，1)文法の追加により変更される状態のみをCLOSUREの対象とする，2)先読み変更箇所の特定と，変更された先読みを持つ項のみを状態遷移表内に生成する，の概念を基本に再構成を行う．

以下にLR(1)状態遷移表をインクリメンタルに構成し新たなパージングテーブルT'を出力するアルゴリズムを示す．

入力: 以前に作成されたパージングテーブルT; カーネル項の集合C'; 追加する文法{A::=α};
出力:再構成後のパージングテーブルT';

```
procedure IPTG( T, C', {A ::= α} ):
begin
G := G∪{A ::= α}
la := FIRST( A )
for 右辺にAを含み，左辺がAでないすべての文法p∈G do
  for pでreduceを行うすべての状態I'  do
    ε∈ la ならばI'でのreduce(p)を行う終端語よりpの先読みla2を復活する．ここで計算したI'内の項を〔B ::= βXAγ・,la2〕とする．
    repeat
      I'より〔B ::= βX・Aγ,la2〕の項を持つべき状態I''までAγの遷移を逆に辿り，I''に〔B ::= βX・Aγ,la2〕の項を追加する．
      C'にCLOSURE(I'')を追加する．
      if X∈VN then
        I''より〔B ::= β・XAγ, la2〕の項を持つべき状態I'''までXの遷移を逆に辿り，I'''に〔B ::= β・XAγ, la2〕の項を追加する．
        if ε∈la then la3 := la ∪ la2 ∪ FIRST(γ)
        else la3 := la
        endif
        C'にS-CLOSURE(I''', X, la3)を追加．
      endif
    until βにもはやAが含まれない，またはドット位置が最も左．
    repeat
      for GOTO(I,X)が空でなくしかもC'に追加またはC'に含まれるカーネル項を変更するまたは追加された項のドット位置より右にAを含む，C'の各項の集合Iおよび各文法記号X do
        GOTO(I,X) を C' に加える．
    until C'に追加または変更する項がもはやない
  C'よりパージングテーブルT'を抽出する．
  T' := T' ∪ T
end.
procedure S-CLOSURE(I, X, la):
begin
  repeat
    for Iに対し，生成規則X::=γ及びla．ただし，〔X ::=・γ,la〕がIに含まれないもの do
      〔X ::=・γ,la〕を I に加える．
  until I に加えることができる項がもはやない．
  return I
end.
```

図4に文法G1に{C::=a}を追加した状態遷移表を示す．図4での網掛け部分が追加された状態であり，またボールドで示した項が追加または先読みの変更されたものである．

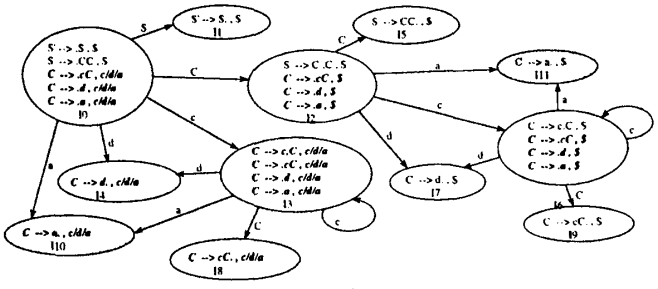

図4．再構成後の状態遷移表

4．評価

本手法の評価として，文法数 2000～6000程度の文法に対して１個のルールを追加する実験を行った．表1と図5に追加速度に対する結果を示す．稼動マシンはSUN4である．尚，文法には自然言語の解析を目的とするものを用い，また，速度の平均値は10種類の追加文法の平均である．結果として，すべての状態遷移表を作りなおす場合と比べ最高で100倍程度高速であるという結果が得られた．また，表1より，本手法では文法数が増えるほどその差が大きく現れるため，特に自然言語解析のような文法数の多くなる大規模システムの構築には非常に効果的であるといえる．

表１．文法追加に対する時間的評価

文法数		2000	3000	4000	5000	6000
PTG	(sec)	34	50	66	82	109
IPTG	最小	1	1	1	1	1
	最大	10	13	18	25	39
	平均 (sec)	1.9	3	4.7	6.6	12

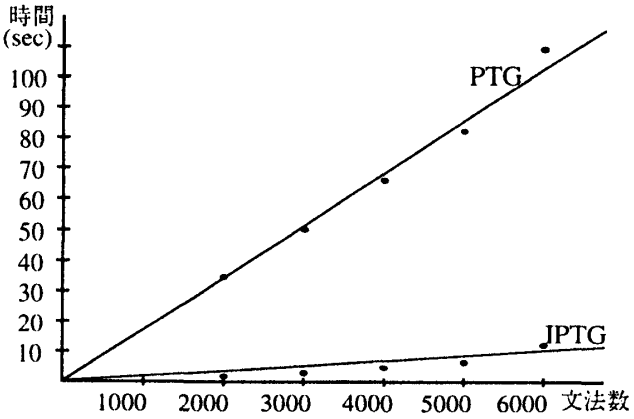

図５．追加速度の時間的評価に対するグラフ

5．まとめと今後の課題

LR(1)パージングテーブルのインクリメンタルな構成法を提案した．本手法では，パージングテーブルの再構成の際に状態遷移表の変更箇所のみを高速に書き換えるため，より少ない資源で効率的かつ迅速に文法を構築することが可能である．文法数2000～6000程度の文法に対して1個のルールを追加する実験により，すべての状態遷移表を作りなおす場合と比べ，最高で100倍程度高速であるという結果が得られた．

今後の課題として，文法の削除に対するテーブルの再構成法と，それら手法の応用による文法の自動学習法の確立等があげられる．また，解析時の未知語の推定と自動学習を組み合わせることによるロバストな構文解析器の実現等についても興味深い課題である．

【参考文献】

1)Aho. A. V, and Ullman : Compiler, 倍風館
2)R. Nigel Horspool : Incremental Generation of LR Parsers, Comput. Lang. Vol. 15, No. 4, pp. 205-223, 1990
3)Jan Heering, Paul Klint, and Jan Rekers: Incremental Generation of Parsers, IEEE Transcations on Software Engineering, Vol. 16, no. 12, Dec. 1990