自 然 言 語 処 理 99－7
（1994．1．21）

ＬＲ表を用いたチャートパージングアルゴリズム

伊東　秀夫

（株）リコー・研究開発本部・情報通信研究所
〒222 神奈川県横浜市港北区新横浜3-2-3

概要

LR表を用いて、不活性弧のみからなるチャートと状態番号のリストを生成しながら、効率的にパージングを行なうアルゴリズムを開発した。このアルゴリズムは、任意の文法を対象として worst-caseの計算効率を抑えることよりも、自然言語の文法を対象として実際の計算効率を向上させることを主眼としたものであり、英語や日本語等に関して非常に高速かつコンパクトな全解生成型のパージングが可能である。

本稿ではアルゴリズムを解析例と共に説明し、本アルゴリズムと従来のチャート法および冨田法との比較を通して本アルゴリズムを特徴づける。また幾つかの自然言語の文法に関する冨田法との比較実験結果を示し、本アルゴリズムの高速性を示す。

Chart Parsing Algorithm Using LR Table

Hideo Ito
R&D Center, RICOH COMPANY,LTD
3-2-3, Shin'yokohama, Kohoku-ku, Yokohama, Kanagawa, 222, Japan.
hideo@ipe.rdc.ricoh.co.jp

ABSTRACT

We have developed a new chart parsing algorithm which makes use of LR table. Parsing is controlled by inactive arcs in the chart and lists of state numbers. The point of this algorithm lies on the real-world performance of the parser in connection with natural language grammars rather than on the theoretical worst-case time complexity of the parser in connection with arbitrary context free grammars. We present the experimental results which suggest our algorithm is more efficient than Tomita's algorithm and seems to be very promissing.

1. はじめに

LR表を用いて、不活性弧のみからなるチャートと状態番号のリストを生成しながら効率的にパージングを行なうアルゴリズムを開発した。本稿では、このアルゴリズムをLRチャート法と呼ぶ。

同様にLR表を用いたパージングアルゴリズムとして冨田法[冨田 86]がある。冨田法のworst-caseの計算時間のオーダがアーリ法のように$O(n^3)$に抑さえることができない(Johnson 89)ことから、幾つかの改良が提案されている(Kipps89, 田中, Suresh 91,Schabes 91)。しかし、任意のCFGを対象としてworst-caseの計算オーダを抑さえることが、自然言語解析の実際上の計算効率の向上に直接つながるわけではない。なぜなら自然言語の文法が上記のworst-caseを与える保証はないからである。実際の計算効率に大きく影響するのは、LR表やグラフ構造化スタック(GSS)の利用といった点であるが、これらは上記オーダを変えない。また、worst-caseの計算オーダを抑さえることは中間計算結果の記憶量を増大させる。このことは、実用規模の自然言語解析を行う際の問題となる。

以上の理由から、LRチャート法は冨田法やWFSパージング[中瀬 88]と同様、自然言語解析における実際上の計算効率を主眼に置いて開発された。

冨田法とLRチャート法との違いは、GSSの代わりにチャートと状態番号のリストを用いる点にある。これにより、GSSに比べ単純でオーバヘッドの少ないbook-keepingが可能になる。また、冨田法での重複した文法適用と不完全な構造sharingの問題(後述)が生じない。さらに、チャートは中間計算結果の記録と解析結果の記録に兼用されるので、GSSと圧縮統語森の総体に比べコンパクトな解析が可能になる。

一方、LRチャート法ではかなり処理が統制されるが、文法適用時に探索が生じる場合がある。しかし、後述の比較実験結果から、冨田法のように処理を完全統制する為にGSSによる複雑なbook-keepingを行うより、LRチャート法のように、ある程度の探索を許すことで単純なbook-keepingを行う方が高速な解析となることが示される。

以降、第二章でLRチャート法のアルゴリズムを説明する。第三章では、チャート法や冨田法などの従来法との比較を通してLRチャート法を特徴づける。第四章では、冨田法との解析時間に関する比較実験の結果を示し、LRチャート法の高速性を実証する。第五章でまとめと今後の展開について述べる。

2. LRチャート法

LRチャート法は、ボトムアップ・横型探索・逐次型のパージングアルゴリズムである。解析結果として、入力文に関する全ての文法的な統語構造が、圧縮統語森[冨田 86]と同様の形式で得られる。

本章では、LRチャート法で用いる幾つかのデータ構造を説明した後、アルゴリズムを解析例と共に説明する。

2.1.チャートと生成テーブル

チャート法[Kay 80]では、チャートに活性弧と不活性弧を加えてゆく。一方、LRチャート法では不活性弧のみをチャートに加える。以降では不活性弧のことを単に弧と呼ぶ。弧は3つ組 a(i,j,X) で表わせる。ここで i,j は弧の開始/終了位置であり、入力の終端記号間に割り当てられた位置番号(節点と呼ぶ)で表わす。X は弧が覆う範囲に対応する文法記号(終端記号又は非終端記号)である。

生成テーブルには、弧がどのような文法適用によって生成されたのか(生成履歴)を記録する。例えば文法 X→Y Z の右辺 Y,Z がチャート中の a(i,j,Y), a(j,k,Z) とマッチし a(i,k,X) が生成された場合 a, b, c を a(i,k,X), a(i,j,Y), a(j,k,Z) のラベル(識別子)とすると次の対応によって生成履歴を表わせる。

a - [b,c]

上記の右側を生成リストと呼ぶ。文法が曖昧性を含む場合、弧の生成リストは一意に定まらないので生成テーブルには

a - [[b,c], ..]

のように、弧と、生成リストのリストとの対応を記録する。

解析結果は、チャートと生成テーブルの2つによって記録される。チャート中の弧に生成履歴を反映させないことで構造の sharing が行われる。また生成テーブル中の弧に生成リストのリストを対応させることで曖昧性の packing が行われる。この解析結果の表現法は、LINGOL[Pratt 75]での表現法、圧縮統語森[冨田 86]、AND-OR木[中瀬 88]のものと等価である(以降ではチャートと生成テーブルの総体を圧縮統語森と呼ぶ場合がある)。特に、[中瀬 88]のWFSパージングはLRチャート法と関係が深い(LR表の情報で制御されたWFSパージングがLRチャート法であると言える)。

2.2. 状態リスト

LRチャート法では、チャートの各節点毎に、状態番号の一次元リストを作成してゆく。このリストを状態リストと呼ぶ。中間計算結果は、チャートと状態リストの２つによって記録される。つまりチャートは中間計算結果の記録と解析結果の記録の両方に兼用される。

2.3. LR表

LRチャート法では、LR表中の shiftとgoto は共に状態遷移を意味し、両者を区別する必要はない。ただし以降では説明の便宜上 shiftとgoto という用語を用いる。

2.4. アルゴリズム

入力の終端記号列を $T_1,T_2,..,T_n$ とする。入力終了を表わす終端記号$を入力列の最後に加えておく。

終端記号:	T_1	T_2	..	T_n	$T_{n+1}=\$$
節点:	0	1	2 ..	n-1	n

リストを[..]で表す。節点 i の状態リストをL_iで表わす。状態番号nと文法記号Xに対するLR表のエントリをLR[n,X]で表わす。開始状態番号を0とする。アルゴリズムを以下に示す。

```
parse()
・let  L0 := [0].
・for k from 1 to n
    add_arc( a(k-1, k, Tk), [] ).
```

```
add_arc(A,G)
・if Aがチャート中にある
    ・Gを生成テーブルに登録.
    ・return yes.
  else
    ・if check_arc(A) の返値が yes
        ・Aをチャートに登録.
        ・Gを生成テーブルに登録.
        ・return yes.
      else
        ・return no.
```

```
check_arc(a(i,j,X))
・Li 中の全ての状態番号nに関しLR[n,X]を
  調べ遷移先の状態番号を集合Sに集める.
・for all m in S
  ・if LR[m,Tj+1] が accept
      ・return yes.
    else if LR[m,Tj+1] が shift/goto     ..(a)
      ・Lj にmを加える.

・S中の全ての状態番号mに関しLR[m,Tj+1]
  を調べ適用可能な文法を集合Rに集める.
・for all M→α in R
  ・文法右辺の文法記号列αがチャート中の
    節点(k)から節点(j)の範囲でマッチし生成
    リストGが得られる度に
      ・add_arc(a(k,j,M),G).          ..(b)

・if (a)が一度も成立せず or (b)の返値が全てno
    ・return no.
  else
    ・return yes.
```

2.5. アルゴリズムの説明

パージングのトップレベルが parse である。節点0の状態リストに開始状態番号を設定し、入力の各終端記号に対して先頭から順に add_arc を行う。

add_arc の引数は弧Aと生成リストGである。(parse では終端記号の生成リストとして空リスト[]を与えている。) 引数の弧Aがチャート中に既に存在すれば生成リストGを記録しyesを返値としてadd_arc を終了する。さもなければ弧Aをチャートに登録すべきかどうかを check_arc によって判定する。

この判定は、弧の文法記号Xが、開始記号から節点iと節点jの間に導出される為の必要条件を用いて行う。この必要条件は、次の条件1)2)の少なくともどちらかが成立することとして規定できる。

条件1)　Xによる状態遷移の内、遷移先の状態からXの直後の終端記号により状態遷移が可能なものが少なくとも１つある。

条件2)　Xを最右導出する非終端記号Yに対応する弧の内、条件1)を満たすものが少なくとも１つある。

check_arcでは(a)により条件1)を調べ、かつ節点jの状態リストを作成し、(b)により条件2)を調べている。

(b)に先だって、文法左辺Mに対応する弧の開始節点kと生成リストGを得る為に、文法右辺αとチャートとのマッチングが行われる。集合SはXによる状態遷移先の状態番号の集合なのでLR[m,T_{j+1}]から得た適用可能な文法は必ずXを右辺の最右要素とする。よってチャートとのマッチングは、実際にはαから最右要素のXを除いた文法記号列に関して行えばよい。

各弧に関しcheck_arcは一度だけ行われ、かつ上で述べたように文法適用はその弧をキーにして行われるので、重複した文法適用を行うことはない。

なお前節のアルゴリズムは、各節点間の終端記号が一つの場合を示したが、次節の解析例でみるように二つ以上の場合を扱えるように容易に拡張できる。

2.6. 解析例

本節では、図2.1に示す文法を用いて文

『きたから伝わった』

を解析する場合のトレースを行う。図2.2にLR表を示す。この文と文法は文献(田中 89)のものである。語に対して与えられたN,V等の前終端記号をここでは便宜的に終端記号と呼ぶ。図2.4にチャートと状態リストの変化を、図2.3に生成テーブルの変化の様子を示している。ただし生成テーブル中で終端記号の生成リストは省略した。

まず節点0の状態リストとして開始状態番号0からなるリストを設定する(図2.4(1))。"きた"の終端記号Nに関してadd_arc(a(0,1,N),[])を行う。a(0,1,N)はチャートに登録されていないのでcheck_arc(a(0,1,N))を行う。節点0の状態リストには0のみが格納されている。状態0からNによる状態遷移があるかどうかをLR表で調べるとsh4、つまり状態4に遷移することがわかる。状態4から、Nの次の終端記号P("から"に対応)による状態遷移があるかどうかをLR表で調べるとsh7があるので、節点1の状態リストに4を加える。状態4で次の終端記号がPの場合に適用可能な文法があるかどうかをLR表で調べると無い。先ほど状態4に関して(a)が成立したのでcheck_arc(a(0,1,N))はyesを返しadd_arc(a(0,1,N),[])に戻る。check_arc(a(0,1,N))の返値がyesなのでa(0,1,N)をチャートに登録する。生成リスト[]を生成テーブルに登録しadd_arcを返値yesで終了する(図2.4(2))。

次の終端記号V("きた")に関しadd_arc(a(0, 1, V), [])を行う。check_arc(a(0,1,V))を呼び出し状態0とVから遷移先の状態番号3をLR表から得る。状態3とPでは状態遷移がないので状態番号3は節点1の状態リストには加えない。状態3とPからLR表でre2を得る。文法(2) S→Vの文法右辺から最右要素Vを除くと空列になるので、文法右辺とチャートとのマッチングは実際には行われずadd_arc(a(0,1,S),[c])が呼ばれる。ここでcはa(0,1,V)のラベルである。詳細には追わないがcheck_arc(a(0,1,S))で節点1の状態リストに1が加えられ返値yesで終了する。a(0,1,S)がチャートに加えられ生成テーブルにa(0,1,S)のラベル(b)と共に生成リスト[c]を要素とするリスト[[c]]が記録される。ここまでで、チャートと状態リストは図2.4(4)、生成テーブルは図2.3(イ)の内容になる。同様に解析は進んで最終的には図2.4(7)、図2.3(ハ)になる。

(1) S → PP S
(2) S → V
(3) PP → N P
(4) PP → S P

図2.1 文法

状態	ACTION				GOTO	
	N	P	V	$	PP	S
0	sh4		sh3		2	1
1		sh5		受理		
2	sh4		sh3		7	6
3		re2		re2		
4		sh7				
5	re4		re4			
6		sh5,re1		re1		
7	re3		re3			

図2.2 LR表

(イ)	(b)	-	[[c]]
(ロ)	(b)	-	[[c]]
	(d)	-	[[a, e], [b, e]]
(ハ)	(b)	-	[[c]]
	(d)	-	[[a, e], [b, e]]
	(f)	-	[[d, g]]
	(g)	-	[[h]]

図2.3 生成テーブルの変化

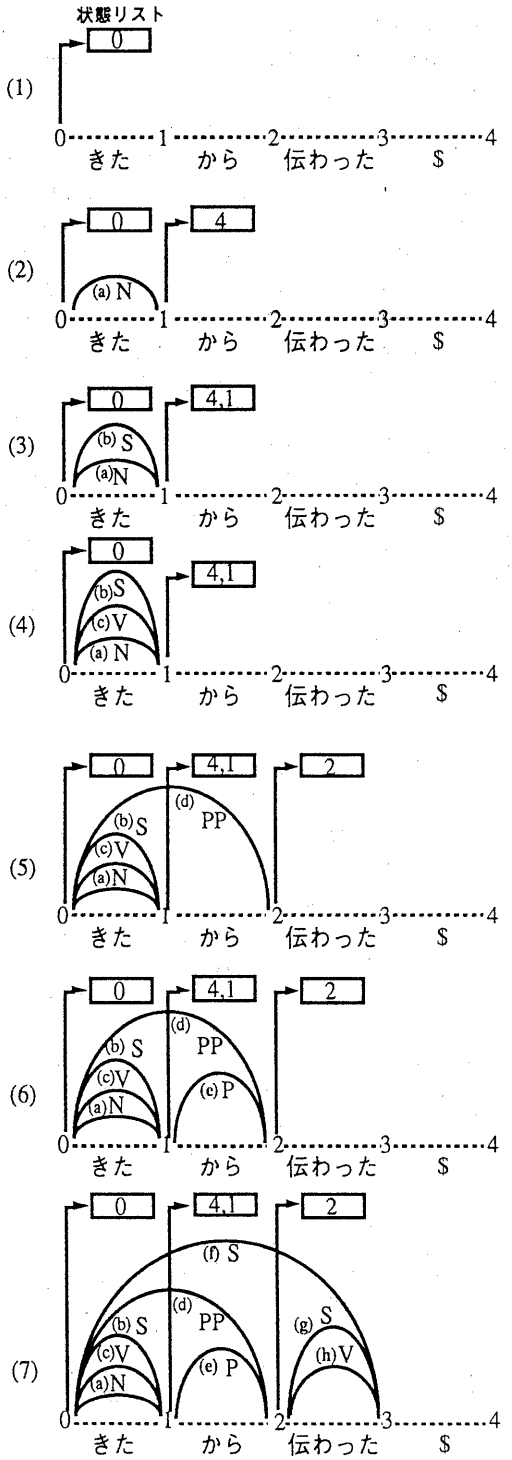


図2.4　チャートと状態リストの変化

3. LRチャート法の特徴

本章ではLRチャート法と従来法（チャート法と冨田法）を比較することによってLRチャート法を特徴づける。

3.1. チャート法との比較

LRチャート法はチャート法に比べ高速な自然言語解析が可能である。このことは、LR表の情報を用いることで、チャート法に比べ、統制されかつ無駄のない処理が行われることによる。

チャート法のworst-caseの解析時間のオーダは入力サイズをnとするとn^3である。一方、LRチャート法は冨田法と同様、文法右辺の長さが2より大きい場合、このオーダは保証されない。このことは活性弧を用いないことによる。しかし、LRチャート法と同じく活性弧を用いないWFSパージングでの実測オーダが[中瀬 88]によれば3000弱の英文法に関して$n^{1.9}$程度と報告されているように、自然言語解析において上記worst-caseが問題になるとは経験的に考えにくい。

チャート法では、中間計算結果は、活性弧と不活性弧によって記録される。一方、LRチャート法では不活性弧と状態リストによって記録する。状態リストの記憶量は、nのオーダーであり、実際上も問題にならない。チャートの記憶量のオーダはn^2であるが、活性弧を記録しないのでLRチャート法のほうが実際の記憶量は少ない。CFGの規模と入力サイズが大きくなるにしたがって両者の差は顕著になる（このことは、LRチャート法と同様、LR表を用いてチャートを作成してゆくアルゴリズム[Schabes 91]に対しても当てはまる。このアルゴリズムでは状態番号付きの弧(item)をチャートに記録するが、活性弧に相当する弧もチャートに記録してゆくからである）。またLR表による予測が有効に機能する場合は、無駄な不活性弧が生成されることが少ない分、チャート法より記憶量が少なくなる。

チャート法では、最終的な解析結果はチャートそのものとして記録されるが、LRチャート法では解析結果は圧縮統語森として記録される。一つの解析木を取り出す時間はチャートではn^2、圧縮統語森ではnの時間を要する。記憶量はチャートのn^2に対して圧縮統語森はn^3以上のオーダになるが、実測では上記の中瀬88によれば$n^{1.65}$程度である。圧縮統語森は曖昧性を明示的に記録しているので、パージングの後に曖昧性を解消する場合に見通しがよい。

3.2. 冨田法との比較

冨田法では重複した処理を避ける為に、GSSのマージを可能な限り行う。それにも関わらず重複した処理が生じてしまう場合がある。又、圧縮統語森での構造のsharingが適切になされない場合がある。このことを図4.2の文法を用いて次の終端記号列を解析した場合を例にして説明する。

n v n prep n

下図3.1は先頭からn,v,n,prepと処理が進み、最後の終端記号nをshiftした直後からGSSの変化を追ったものである。

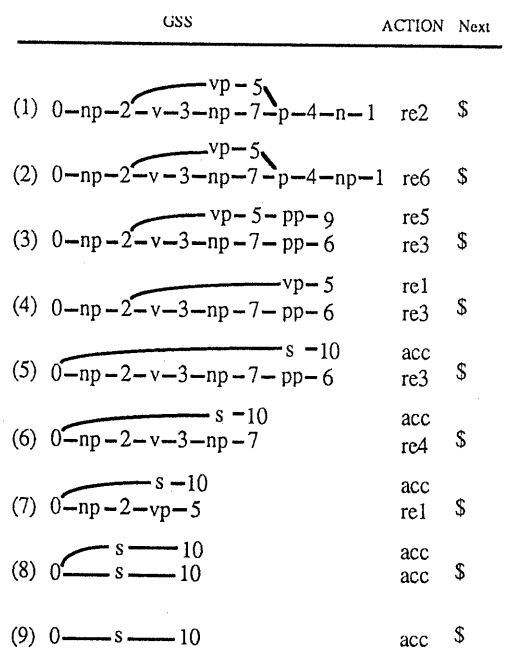


図3.1. 冨田法のGSSの変化

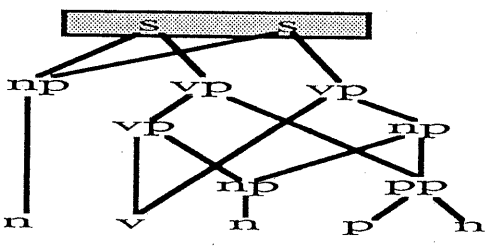


図3.2. 冨田法による圧縮統語森

図3.1の(3)では2つのアクティブなstackに対して異なるreduceが行われようとしているが、冨田法ではこのように複数のreduceが存在する場合は、どちらか一方を任意に選んで処理を進める。ここではreduce 5を先に行っている。(4)でvpが生成され、(5)でsが文法s->np vpにより生成される。reduce 3は(6)で行われるが、(8)で先ほどと重複した文法適用、即ちs->np vpによりsが生成される。mergeは(9)で行われるので上のように重複した文法適用が行われる。又、この結果、図3.2に示すようにうまくvpのsharingがなされていない圧縮統語森が生成されてしまう。この問題は(4)でreduce1の代わりにreduce3を行えばvp同士のmergeがなされるので生じないが、こうした最適なreduceの順番を選択するための情報はGSS上にはない。

一方、LRチャート法ではこの重複した文法適用は発生しない。なぜなら、(7)のvpに相当する弧Xが生成された時点では、すでに(4)のvpに相当する弧Yが既にチャート中に存在する為、Xを元に再び文法適用はなされないからである。そして、XとYの生成リストが生成テーブルに登録され図3.3に示す正しい圧縮統語森が得られる。

チャートは中間計算結果の記録と解析結果の記録の両方に兼用される。一方、GSSは計算途中で消滅する為、解析結果を圧縮統語森として別途記憶しておく必要がある。中間計算結果の記録の為だけに使用されるGSSと状態リストのサイズは、入力サイズnに対し各々$O(n^2)$とO(n)になることから、冨田法に比べコンパクトbook-keepingが可能である。

冨田法では、処理が完全統制されるが、GSSの操作を中心とした複雑なbook-keepingとなる。一方、LRチャート法では、文法右辺とチャートとのマッチングに探索が生じる場合があるが、GSSに比べ単純なbook-keepingとなる。そして、冨田法の複雑なbook-keepingに伴うオーバヘッドと、LRチャート法の探索に伴うオーバヘッドとの差が、両者の時間効率に差を生じさせる主な要因となる。

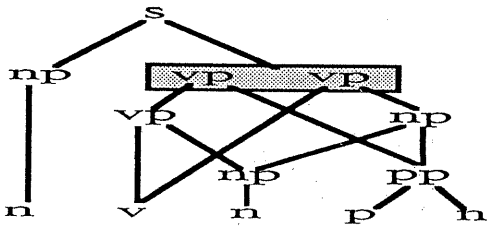


図3.3. LRチャート法の圧縮統語森

4. 実験結果

LRチャート法と富田法の解析時間に関する比較実験を通して、LRチャート法の高速性を実証する。

アルゴリズムの実装は、LRチャート法と富田法ともにC言語を用い、SPARC station IPX上で実験を行った。富田法のプログラムは東工大の田中研究室で作成されたものである。

解析時間は富田法とLRチャート法ともに、パージングを行い全解(圧縮統語森)を得るまでの時間である。ただし辞書引きの時間は含まない。時間測定は実行毎にばらつきが出るので各文毎に10回測定し平均を取った。

図4.1は、[富田 86]にある文法IIIでSENTENCE SET Iを解析した場合の結果である。文法IIIは224の英文法規則であり、SENTENCE SET Iは40の英文である。英文の各語には一つ以上の品詞が割り当てられている。図は両者の解析時間と文長、及び解析時間比と文長の関係を表わすグラフである。

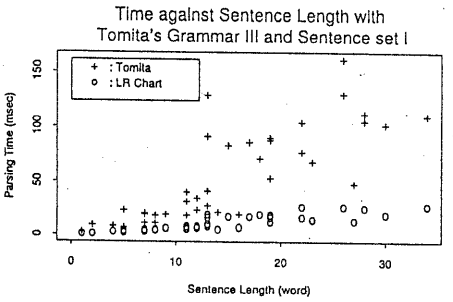
図4.2は、図4.3で示した文法で次の品詞列

n v n prep n prep n prep n ...

を解析した場合の結果である。この文法は英語のPP-attachmentに関するものである。

図4.1.では、両者の時間差は語長が長くなるにしたがって大きくなるが、時間比は平均値である4.5倍を中心に分布している。この結果から、富田法のように処理を完全統制する為にGSSによる複雑なbook-keepingを行うより、LRチャート法のように、ある程度の探索を許すことで単純なbook-keepingを行う方が高速な解析となることがわかる。

また、上記結果を調べた所、PP-attachmentを多く含む文において両者の差が大きくなる場合が多いことがわかった。そこで図4.2.のように、PP-attachmentに関する文法と文例によって両者を比較してみた。図4.2.では富田法との差がより顕著である。これは、LRチャート法では前節で述べた重複した文法適用がなされないこと、及び、この文法ではLR表による予測が有効に働かずGSSによるbook-keepingの為のオーバヘッドの影響が、より顕著に現われることに起因する。



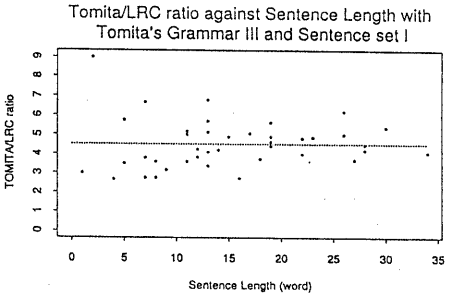


図4.1.

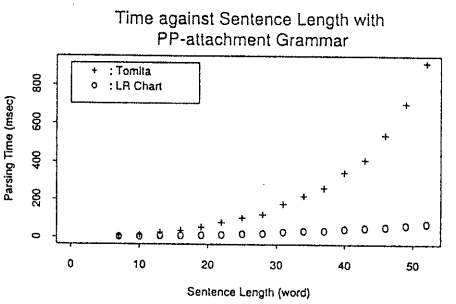


図4.2

(1) s → np vp
(2) np → n
(3) np → np pp
(4) vp → v np
(5) vp → vp pp
(6) pp → p np

	ACTION				GOTO			
	v	n	p	$	s	np	vp	pp
0		sh1			10	2		
1	re2	re2	re2	re2				
2	sh3		sh4				5	6
3		sh1				7		
4		sh1				8		
5			sh4	re1				9
6	re3	re3	re3	re3				
7			sh4 re4	re4				6
8	re6		sh4 re6	re6				6
9	re5	re5	re5	re5				
10				acc				

図4.3

5. おわりに

本稿では、LR表を用いて不活性弧のみからなるチャートと状態番号のリストを生成しながら、効率的にパージングを行なうアルゴリズムについて説明した。実験結果からもわかるように極めて高速な自然言語解析が可能である。今後の課題として以下のものを挙げることができる。

・ε-規則に対するアルゴリズムの拡張
・より本格的な自然言語文法を用いた評価

はじめに述べたようにworst-caseの理論的な計算オーダだけからは、自然言語解析における実際の能力は評価できない。この観点から具体的な自然言語文法を用いた評価実験が行われてきた(Slocum 81,Shann 91)。その際、本実験で使用した[冨田 86]のデータのように、様々な研究グループが共通して利用できる文法と辞書のサンプルを整備することは、実用的なアルゴリズムの発展に大きく寄与すると考えられる。

[謝辞]
いくつかの有益なコメントを頂いた東工大 田中穂積教授、並びに冨田法をインプリメントしたプログラムを提供していただいた田中研究室の皆様に感謝致します。

[参考文献]

[Aho 72] Aho,A.V.and Ullman,J.D. : The Theory of Parsing, Translation and Compiling, vol.1 - 2, Prentice-Hall 1972

[Johnson 91] Johnson, M. : Computational Complexity of GLR Parsing in Tomita,M ed. : Generalized LR Parsing. Kluwer , 1991

[Kay 80] Kay,M. : Algorithm Schemata and Data Structures in Syntactic Processing, TR CSL-80-12, Xerox PARC, 1980

[Kipps 91] Kipps,J,N. : GLR Parsing in Time O(n3), in Tomita,M ed.: Generalized LR Parsing. Kluwer Academic Publishers, 1991

[中瀬 88] 中瀬純夫 : 英日機械翻訳における解析手法について,情処 自然言語処理研究会, 69-7, 1988

[Pratt 75] Pratt,V.R. : Lingol-A Progress Report, IJCAI '75, 1975

[Schabes 91] Schabes, Y. : Polynominal Time and Space Shift-Reduce Parsing of Arbitarary Context-free Grammars, Proc. of 29th ACL, 1991

[Shann 91] Shann,P. : Experimnts with GLR and Chart Parsing, in Tomita,M ed.: Generalized LR Parsing Kluwer Academic Publishers, 1991

[Slocum 81] Slocum,J. : A practical comparison of parsing strategies, Proc. of 19th ACL, 1981

[田中 92] 田中穂積, Suresh,K.G, 山田耕一 : 先祖表を利用した一般化LRパージングアルゴリズムのファミリー, 情処 自然言語処理研究会, 90-5, 1992

[田中 89] 田中穂積 : 自然言語解析の基礎, 産業図書 1989

[冨田 86] 冨田勝: Efficient Parsing for Natural Language Kluwer Academic Publishers, 1986