TECHNICAL SURVEY

自然言語の高速な構文解析法
——一般化LR構文解析アルゴリズム——

解 説

田中 穂積

田中穂積：正員 東京工業大学情報理工学研究科

A Method of Efficient Natural Language Processing : Generalized LR Parsing Algorithm. By Hozumi TANAKA, Member (Graduate School of Information Science and Engineering, Tokyo Institute of Technology, Tokyo, 152 Japan).

ABSTRACT

プログラム言語のコンパイラで，LR法とよばれる構文解析法がよく使われている．これは，(1) LR法は決定的に解析を進めることができるので高速な構文解析が可能であること，(2) プログラム言語の文法を設計段階でLR文法に限ることができることによる．しかし，自然言語の文法としてLR文法の記述力は十分であるとはいえず，少なくとも文脈自由文法 (CFG) の記述力が必要である．一般のCFGを扱うことができるようLR法を拡張したものに一般化LR (GLR) 法があり，最近自然言語の高速な構文解析法として注目されている．まず1．では，GLR法の基本的な考え方を説明する．2．ではGLR法の原理を具体例を用いて説明する．3．では，解析結果の妥当性を確率的に計算する確率CFGとGLR法との関連を簡単に説明する．最後の4．では，GLR法に関するいくつかの問題を取り上げて将来の研究課題を展望する．

キーワード：自然言語処理，構文解析，一般化LR法，音声認識，確率CFG

1. はじめに

コンピュータで，日本語や英語などの自然言語を処理することは，コンピュータと人間との間の使用言語の相違に基づくギャップを埋めるための重要な技術として注目されている．自然言語処理の応用として，かな漢字変換システム，自然言語理解システム，機械翻訳システム，音声理解システムなどがあげられる．そのための基本技術の一つに自然言語解析技術がある．

自然言語解析技術の中で最も研究が進んでいるものに構文解析技術がある（構文解析と似た用語に統語解析がある．若干ニュアンスが異なるが以下では両者を特に区別しない）．構文解析の目的を述べておく．

1. 文法と辞書を用いて，文が文法にかなっているかどうかを判定する．
2. 文の構文構造 (syntactic structure) を抽出する．
   (1) 文の型（平叙文，命令文，疑問文，感嘆文など）を判定する，
   (2) 文の主部，述部を抽出する，
   (3) 修飾／被修飾関係を抽出する．

構文解析では，文を構成する単語の品詞の並びを問題にする．日本語であれば，「名詞・助詞」の並びは文法にかなっており，これは更に「連用修飾句」などのより大きな要素としてまとめられる．一方，「助詞・名詞」の並びは，より大きな要素としてまとめることができない．このような事実は，日本語文法規則の一部をなしている．文法規則を用いて，文が文法にかなっているかどうかを判定し，文の構文構造を抽出することが構文解析の主たる目的であるが，この構文構造は，意味解析や文脈解析をガイドする有用な情報となる．

構文解析を行うためのアルゴリズムとしてさ

まざまなものが考案されている．最近良く使われるアルゴリズムに，チャート法[11,12]と一般化LR（GLR）法がある．いずれも部分的な解析結果をボトムアップに組み上げ無駄なく最終的な解析結果を得るアルゴリズムを基本としている．そして一般の文脈自由文法（CFG）を用いるが，最近では，音声認識にも応用されている．GLR法は，先読み語のもつ品詞（前終端記号；preterminal）に関する情報を利用しながら解析を進めるもので，経験的に最も効率の良いアルゴリズムであるとされている．以下ではGLR法による構文解析について説明する．

GLR法の基本的な考え方は，Knuthによる[8]．彼は，構文解析の過程で数語先読みし，そこから得られる情報を用いて無駄なく構文解析を行う方法を考案した．

例えば文頭に，名詞にも動詞にもなりうる多品詞語が現れる英語の文を考えよう．文頭の語を走査した段階では，この文が平叙文なのか命令文なのか決められない．従って，平叙文を解析するための文法規則と命令文を解析するための文法規則を適用することになる．次に後続語の解析に移り，後続語が動詞であることがわかれば，命令文を解析するための文法規則の適用をこの段階で棄却する．しかし，文頭の語を走査した段階で，こっそり1語先読みしておけば，平叙文を解析する文法規則だけを当初から適用することができ，不必要な文法規則の適用を早い時期に棄却することができる．これがKnuthの基本的な考え方である．解析過程で，次に適用すべき文法規則が常に一意に決定可能な場合，その解析法を決定的（deterministic）であるとよぶ．

Knuthは，決定的に解析を進めることのできる文法（先読み語数をkとし，LR(k)文法とよぶ）の性質と，それを用いたLR(k)法とよぶ構文解析法を考案した．実際には先読み語が1語の場合が実用に供されており，単にLR文法，LR法といえば，$k=1$であると考えて良い．LR文法を用いて自然言語の性質の多くが記述できれば問題はない．残念なことに，自然言語を解析するための文法としてLR文法は十分でなく，少なくともCFGのレベルが必要になる．LR文法はCFGのサブセットである．

LR文法をCFGに拡張し，LR法を一般化した構文解析法をGLR法とよぶが，GLR法ではCFGを扱うため，もはや決定的な解析は保証されない．この非決定性の問題を解決するために，とりあえず一つの可能性を選び，それが不適切であることが判明した時点で後戻り（バックトラック；backtracking）し，次の可能性を選択する手法が考えられる．これは縦型（深さ優先；depth-first）探索とよばれるが，後戻りするとき，一度計算した結果を捨て去ることが普通であり，同じ計算を何度も繰り返さねばならないことがあり効率が悪い．この問題に決着をつけたのは富田である[1]．富田は縦型探索ではなく横型（幅優先；breadth-first）探索を採用した．横型探索では複数のCFG規則適用の可能性があるとき，それらをすべて考慮しつつ解析を進める．この場合後戻りがないため，再計算を防ぐことができる．富田は更に使用メモリ量の切詰めにも工夫を凝らした方法を開発した（2.3の圧縮構文森の導入など）．富田によるGLR法は，経験的に最も効率の良い構文解析法である．GLR法といえば以下では富田の開発したものを指す．但し先読み語数は1とする．

2. GLR法

2.1 最右導出・ボトムアップ解析・シフト／レデュース操作

図1の文脈自由文法（CFG）を考える．[1]から[4]は日本語の構文構造を表す規則，[5]から[10]は辞書項目の記述である．いずれの規則も，矢印の左辺に現れる一つの記号を，無条件に（どの脈絡で現れるか依存せずに）右辺の記号列に書き換え（導出）可能なことを示している．そのため文脈自由（context-free）とよばれる．

最右導出では，最も右側に現れる非終端記号を，CFG規則の適用により次々に書き換えて

[1] S → PP S　[5] v → きた　[9] p → から
[2] S → v　[6] v → 伝わる　[10] p → が
[3] PP → n p　[7] n → きた
[4] PP → S p　[8] n → 文化

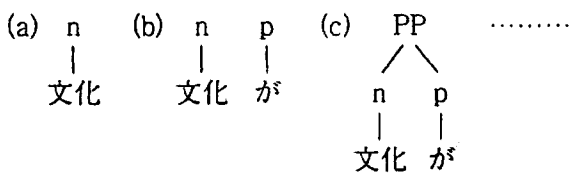
Sは開始記号で文を表す．例えば[3]は，連用修飾句(PP)は名詞(n)に助詞(p)が付いたものから構成されるなどと読む．

図1　簡単な日本語 CFG

いく．例えば，

S=>PP　S=>PP　v=>PP 伝わる
=>n　p伝わる =>n が　伝わる
=>文化　が　伝わる

は最右導出の例である．この最右導出を逆にたどると，次の解釈ができる．(a) [8] により「文化」をnに書き換える．(b) [10] により「が」をpに書き換える．(c) [3] により "n　p" の並びをPPに書き換える．(d) [6] により「伝わる」をvに書き換える．(e) [2] によりvをSに書き換える．(f) [1] により "PP　S" の並びをSに書き換える．この (f) から (a) の過程はボトムアップの構文解析過程に等しい．例えば次のようである．



一方スタックを用いてこれを解釈すると次のようになる．

	スタック	入力文	アクション
(0)	|-	文化 が 伝わる $	シフト
(1)	|- n	が 伝わる $	シフト
(2)	|- n p	伝わる $	レデュース
(3)	|- PP	伝わる $	シフト
(4)	|- PP v	$	レデュース
(5)	|- PP S	$	レデュース
(6)	|- S	$	受　理

(0) で行うアクションはシフトである．これは入力文の先頭の語「文化」をスタックにプッシュする操作を表す．このとき辞書引きにより実際には「文化」の品詞nをプッシュする．結果は状態 (1) になる．(1) では更に「が」の品詞pをプッシュし状態 (2) になる．(2) では，スタックに対して規則 [3] によるレデュースを施す．まず規則 [3] の右辺に二つの非終端記号があるので，スタックトップからnとpの二つを取り出す(ポップする)．そして，規則 [3] の左辺の記号 PP をプッシュする (状態 (3))．CFG 規則では，左辺の記号の数は必ず1，右辺の記号の数は必ず1以上だから，レデュース操作により，スタックの長さが増えることはない．右辺の記号の数が2以上なら必ずスタック長は減少する．レデュースには還元という意味と共にこうした意味がある．以下同様に解析が進み，スタックにSのみが残り，入力文がすべて消費されていれば解析は成功し，入力文を受理する．言い換えると，入力文は最終的にSに還元され，このとき先読み語が文末記号「$] になっている．

以上の説明から，最右導出，ボトムアップ解析，スタックとシフト・レデュースによる解析とは，密接に関連し合っていることが理解できたのではないかと思う．GLR 法は，これより複雑なスタック (グラフ構造化スタック) に対するシフト・レデュースを繰り返すことで解析が進む．

2.2　LR 表の作成[5]

2.1 では極めて重要なことが説明されていない．それは，各状態でスタックに施すべきシフト・レデュースアクションをどのようにして決めるか，という問題である．Knuth によれば，それは，スタックの状態と先読み語の品詞とから決めることができる．そのための表 (LR 表) は，CFG 規則の集合 (図1の場合，[1]～[4]) から，あらかじめ作成しておくことができる．LR 表の作り方にはいくつかある．以下では，図1の [1] から [4] の規則に，"[0] SS →S" を付加したものを用いて，正準 LR 表 (Canonical LR table) の作り方を説明する．

但し，2.2 の説明はやや込み入っている．理解に困難を覚える読者は，ここをスキップして 2.3 に進まれても一向に差し支えない．正準

LR表よりメモリ効率の良いLALR表を作る優れたソフトウェアパッケージが利用可能だからである.

統語解析が成功するためには,文が最終的にSに還元されなければならない.そこでアイテム [SS→·S;$] を作成することから始める.$は文末を表し,Sの解析が終了したときの先読み語になっている.アイテム中のドット記号「·」は,その右のSに関する解析をこれから始めることを表している.Sに関する解析は規則[1]と[2]を用いて更にブレイクダウンすることができるので,アイテム [S→·PP S;$], [S→·v;$] を作る.これらのうち前者からは,規則[3]と[4]を用いて,ドット記号の直後のPPを更にブレイクダウンしたアイテムを派生する.このとき派生したアイテム中にも,PPの解析が終了したときの先読み語を記録しておく.先読み語を求めるアルゴリズムは文献(5)に詳しいが大略次のようにする.

規則[1]は,PPの解析後にSを解析することを述べている.従って,PPの解析が終了したときの先読み語の品詞は,Sからの導出で最も左に現れる品詞になる.これは,規則[2]を適用すればvであるし,規則[1]を適用すれば(2.1で述べた導出からもわかるように)nかvである.いずれにしても規則[3]と[4]から作られるアイテムは,[PP→·n p;n/v], [PP→·S p;n/v] となる.更に,これらのうち後者からは [S→·PP S;p], [S→·v;p] を派生する.以上を新しいアイテムが作れなくなるまで続け,作られたアイテムを集合I_0としてまとめると図2のようになる.但し [S→·v;p] と先に作った [S→·v;$] をマージして [S→·v;$/p] としてある.

I_0でnの解析が終ると,(04)のドット記号を一つ右にずらした[PP→n·p;n/v]を作る(図2の(11)).このアイテムは「→」とドット記号に挟まれた記号(列)が解析済みであることを表し,ドット記号の右のpの解析をこれから始めることを表している.I_0でPPが解析されると,(02)から(31)を作り,先に述べた手続きにより,(31)から(32)(33)(34)(35)を派生する.(31)から(35)をまとめてアイテム集合I_3とする.以下同様にして図2に示す各アイテム集合を作ることができる.

アイテム集合I_0で,PPの解析が終ると(02)から(31)が作られることを既に述べた.これはPPの解析が終ると,I_0からI_3に移行すると解釈できる.このようにして図2から図3に示す有限オートマトンが得られる.なお,アイテム集合の添字は2.1の終りに述べたスタックの各状態に対応させてある.

図3では,アイテム集合I_mに状態mを対応させることができる.GLR法ではやや複雑なスタックを用いるが,そこには解析済みの非終

I_0: (01) [SS →· S ; $]
(02) [S →· PP S ; $/p]
(03) [S →· v ; $/p]
(04) [PP →· n p ; n/v]
(05) [PP →· S p ; n/v]

I_1: (11) [PP → n · p ; n/v]

I_2: (21) [PP → n p · ; n/v]

I_3: (31) [S → PP · S ; $/p]
(32) [S →· PP S ; $/p]
(33) [S →· v ; $/p]
(34) [PP →· n p ; n/v]
(35) [PP →· S p ; n/v]

I_4: (41) [S → v · ; $/p]

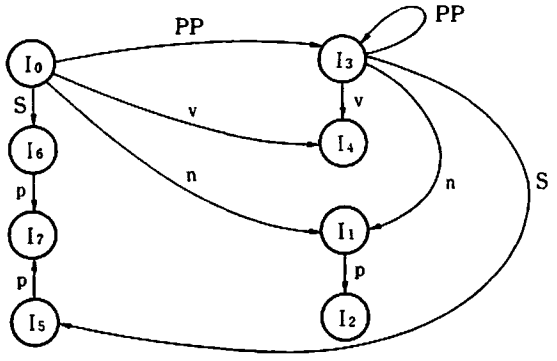
I_5: (51) [S → P P S · ; $/p]
(52) [PP → S · p ; n/v]

I_6: (61) [SS → S · ; $]
(62) [PP → S · p ; n/v]

I_7: (71) [PP → S p · ; n/v]

各アイテムは矢印とドット記号で挟まれた部分が解析済みを,ドット記号の右に現れる記号の解析をこれから行うことを表す.

図2 正準アイテムの集合



丸で囲まれたものが状態を表す．例えば，PP の解析が終了すれば状態 I_0 から I_3 へ推移することを矢印付きの弧は表している．

図3　有限オートマトンによるアイテム集合間の関係

端記号とこの状態を対にして記憶（プッシュ）しておく．そして，スタックトップの状態 s と先読み語の品詞 t からどのようなアクションを行うかを決める 2 次元配列の LR 表，LR[s, t] を用いて解析を進める．

図 2 と図 3 で正準 LR 表を作る準備ができた．これは概略以下のようにして作る．解析済みの非終端記号は状態と対にしてスタックにプッシュする．一方，アイテム中の「→」とドット記号で挟まれた部分の記号列は解析済みを表すので，これらはスタックに存在していることが保証される．例えばスタックトップが状態 2 なら，このスタックには n と p がこの順にプッシュされていることが保証される．

① シフトの書き込み

図 3 によると，スタックトップの状態が 0 で先読み語の品詞が n なら，状態 1 に推移する．このとき先読み語の品詞 n と状態 1 の対をスタックにプッシュ（シフト）する．慣例によりこのアクションを sh 1 と略記し，LR[0, n] に sh 1 と書き込む．こうして状態 1 のアイテム（11）で「→」とドット記号とで挟まれた解析済みの n がスタック上に存在することが保証される．同様に，スタックトップが状態 1 で先読み語の品詞が p なら，LR[1, p] に sh 2 を書き込む．このように，一般に図 3 で，先読み語の品詞 x のシフトにより状態 i から状態 j に推移する場合には，LR[i, x] に sh j と書き込む．このようにして LR 表にシフトアクションを書き込んでいく．

② レデュースの書き込み

図 2 の状態 2 を見るとアイテム（21）[PP→ n p·; n/v] がある．これは n と p が共に解析済みを表しているから，規則［3］を用いてスタック上の n と p をポップし，PP にレデュースすることができる．このアクションを慣例により re 3 と略記する．（21）には，n と p が解析終了後の先読み語は n か v である，と記されているので，LR[2, n] と LR[2, v] に re 3 と書き込む．

re 3 により，n と p をスタックからポップし PP にレデュースした（つまり PP が解析済みとなった）後で，スタックトップの状態をどう決めたらよいだろうか．状態 2 の re 3 で，n と p をスタックからポップすると状態 0 になることが図 3 からわかる．更に状態 0 で PP が解析済みなら，状態 3 に推移すべきことも図 3 からわかる．従って re 3 の実行後は，スタックトップに PP と状態 3 の対をプッシュする．そこで LR[0, PP] に goto 3（または単に 3）と書き込み，状態 3 への推移を指示する．以上のようにしてレデュースアクションを LR 表に書き込んでいく．

③ acc の書き込み

状態 6 に含まれるアイテム（61）（図 2 参照）は，解析が成功裏に終了したことを意味してい

状態	アクション部				GOTO 部	
	n	p	v	$	PP	S
0	sh1		sh4		3	6
1		sh2				
2	re3		re3			
3	sh1		sh4		3	5
4		re2		re2		
5		sh7/re1		re1		
6		sh7		acc		
7	re4		re4			

状態 0 で，先読み語の品詞が n なら，sh1 というアクションを行う．このとき，先読み語の品詞が p ならアクションが書き込まれていないので，エラーとなりこの解析はストップする．

図4　図 1 の CFG から抽出された正準 LR 表

るので，LR[6, $] にacc（受理）と書き込む．

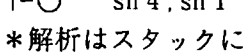
以上の手続きで得た正準LR表を図4に示す．正準LR表を2次元配列LR[s, NT] で表したとき，NTが品詞（前終端記号）の部分をアクション部，NTが品詞以外の非終端記号である部分をGOTO部とよぶ．

図4の正準LR表には，一つのエントリーに複数個のアクションが書き込まれているところがある（LR[5, p] のsh 7/re 1）．これは，図1のCFGが非決定的であるという事実を反映している．この場合コンフリクトがあるといい，GLR法では複数個のアクション（操作）を同時に考慮しながら横型探索により解析を進める．一方，アクションの書かれていないエントリーを参照しなければならなくなったら，その解析は失敗したことを表す．

2.3 GLR法による構文解析

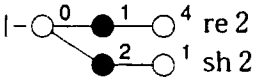
「きた から 伝わる $」という文を例にして，図4の正準LR表を用いてどのように解析が進むかを見てみよう．解析にはグラフ構造化スタックを用いるが，その意味は解析を追う過程で自然と明らかになるはずである．

（0）先読み語「きた」：sh 4 (v) ; sh 1 (n)



＊解析はスタックに状態0の節点を積むことから始まる（0）．先読み語は「きた」であり，その品詞はvとnである．そこでLR[0, v] とLR[0, n] に書かれたsh 4とsh 1を共に実行し，vとnをスタックに積むと共に，それぞれ状態4と状態1にシフトする．二つのアクションを実行したため，スタックが2本に分枝する（1）．

（1） 先読み語「から」：

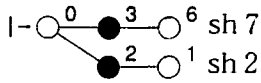


1. [v きた]
2. [n きた]

ここで黒丸と白丸の節点の肩に付けた数字は，それぞれ部分解析結果（スタックの右に示す）と状態を表す．両者は常に対になってシフトされたりレデュースされることに注意．

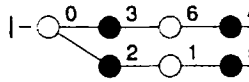
＊先読み語「から」の品詞はpであるから，LR[4, p] とLR[1, p] に書かれたre 2とsh 2を実行するが，このときレデュースがすべて終了してからシフトを行うことにする．re 2によりスタックからvの節点と状態4の節点をポップすると状態0がスタックトップに現れる．re 2は規則［2］によりSに還元するので，LR[0, S] から状態6に移行すべきことを知る（2）．さて状態6でLR[6, p] を調べるとsh 7とあり（レデュースがすべて終ったので）「から」をシフトし（3）になる．

（2） 先読み語「から」：



3. [S (1)]

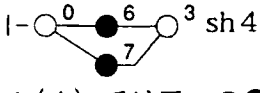
（3） 先読み語「伝わる」：



4. [p から]
5. [p から]

＊（3）のre 4とre 3の実行後はGOTO部から，両スタックトップとも状態が3になる．このような場合，両スタックとも，そこから先(右)は全く同じ動作を行うことになる．この無駄を回避するために，GLR法では状態3の二つの節点をマージする（4）．

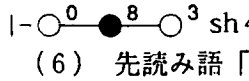
（4） 先読み語「伝わる」：



6. [PP (3 4)]
7. [PP (2 5)]

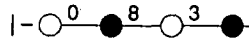
＊（4）では二つの●が状態0と3の節点で挟まれているから，部分解析木6と7を圧縮し圧縮構文森（packed forest）8を作る（5）．以下（6）～（8）と読者は容易に解析過程を追うことができるだろう．

（5） 先読み語「伝わる」：



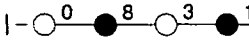
8. [PP (3 4)(2 5)]

（6） 先読み語「$」：



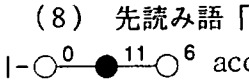
9. [v 伝わる]

（7） 先読み語「$」：



10. [S (9)]

（8） 先読み語「$」：



11. [S (8 10)]

GLR法による解析過程で注意すべきことが三つある．第1は，コンフリクトの扱いである．特にシフトとレデュースのコンフリクトが発生した場合には，レデュース操作を優先させる．

すべてのレデュースが終了してからシフトを行う．待ち合せを行い同期をとるのである．第2は，スタックトップに同じ状態の節点が複数個存在する場合にはマージして一つの状態節点にする．このようにして，以後の重複するスタック操作を回避し，無駄なく構文解析を進めることができる．第1と第2により，使用するスタックは絡み合った構造になるが，それをグラフ構造化スタックとよぶ．第3に，解析木の圧縮を行い，多数のあいまい性に対処する．圧縮構文森の導入がGLR法の高速な解析と使用メモリ量の切詰めに大きく寄与している[6]．まれに数億の構文木を得ることもあるが，それでもGLR法は通常のワークステーション上で十分な速度で動作する．

なお興味ある読者は，「きた」の品詞がnの場合の解析の様子を，2.1の最後で述べたスタックの様子と比べてみるとよい．この場合は「きた　から　伝わる」も「文化　が　伝わる」も，品詞の並びは "n p v" であり同じだから，2.2の冒頭で述べた問題がどう解決されているかがよくわかるだろう．

3. 確率 CFG と GLR 法

一般に，CFG を用いて構文解析した結果には，多数のあいまい性が含まれる．数十語からなる文の構文解析結果に，数百万のあいまい性が含まれることもまれではない．このとき，どの解析結果がもっともらしいかを決めるために，確率 CFG[7] を用いる方法が注目されている[8]．確率 CFG では，各 CFG 規則に確率が振られている．そして左辺が等しい各 CFG 規則に付与される確率の和は1であるとする．

Wright[9] らは確率 CFG で用いる確率と，LR 表の各アクションに振られる確率との関係を明らかにした．GLR 法はレデュース時に個々の CFG 規則を適用するので，確率 CFG では，レデュース時に確率を計算し，それらを掛け合わせて全体の確率を求めていることになる．一方 GLR 法ではシフトが何度か行われてからレデュースを行う．そこでレデュース時だけでなくシフト時にも確率の計算ができれば，より詳細な解析段階で解析結果の確率値による刈り込みができる．言い換えると，CFG 規則に振られた確率を，シフト操作にも分配し，確率値を用いたあいまい性の刈り込みを早期に行うことができる．この考え方は，実際音声認識システムに組み込まれている[9]．なお，チョムスキー標準形（後述）の CFG に確率を付与する方法については文献（10）を参照のこと．

確率を用いることの有効性が認識されるにつれて，確率のソースとなる大量の文データ（コーパス）の重要性がクローズアップされてきた．著作権の問題があるにしろ，我が国でこの問題に対する取組みが遅れているのが気になる．

ここで，確率 CFG やそれと等価な確率 LR を用いることの限界について述べておきたい．これは，同じ CFG 規則が異なる脈絡で使われても，全く同じ確率値を用いることによる．例えば，NP→pronoun という英語の規則は，目的語より主語の位置に現れやすい．この違いを考慮して Briscoe らは，GLR 法で，スタックをポップしたとき現れるスタックトップの状態を左文脈，先読み語を右文脈とすることにより，たとえ同じレデュース操作であっても，左文脈と右文脈の対が異なれば，それらのレデュース操作に異なる確率値を与えておく．このように脈絡に依存した確率値付きの LR 表を用いることで，よりよい構文解析木の確率値が計算可能なことを報告している[11]．

4. おわりに

これまで GLR 法の概略，その優れた性質について述べてきた．GLR に関連した確率 CFG の動向についても簡単に説明してきた．以下ではこれまで触れられなかった問題を述べる．

まず GLR 法による構文解析の計算の手間は，解析すべき文に含まれる単語の数を n としたとき，n^{m+1} のオーダになる．ここで m は CFG 規則の右辺に現れる非終端記号の最大数である．CFG 規則の右辺の記号数がすべて2以下なら（この形の CFG をチョムスキー標準形と

よぶ）計算の手間はn^3であり，チャート法などと変らない．しかし，一般のCFG規則はこの制限に従わないため計算の手間はn^3のオーダを越える．この問題はKippsが解決した[12]．本解説で説明したGLR法は横型探索であるので並列処理にも適している．これは文献（14）を参照されたい．

GLR法の枠組みの中で形態素解析と構文解析とを完全に統合化した新しいアルゴリズムも開発されている[15]．これは，形態素解析で用いる接続表とLR表の類似性に着目したものである．GLR法が音声認識の分野にも有効であることもわかってきている．これについては，文献（16）～（19）を参照されたい．

文法に合わない不適格文（Ill-formed Sentence）をGLR法で扱う試みもあるが，これは構文解析法の頑健性とも関連し，音声対話文の処理では特に重要になる．音声対話文には，「あー」とか「あのー」などの音声や，言い誤りや，それを修正したり，繰り返しなどが散見されるからである．GLR法は左から右に向けた処理の方向性があり不適格文の処理に問題があるとされていたが，筆者は今一度GLR法の枠組みでこの問題を考え直してみたいと思っている．

最後に，限られた紙幅のため駆け足になり説明がわかりにくい部分もあると思われる．その場合，文献を参照するなどして補ってほしい．文献（5）はLR法の基本原理を理解するのによい．GLR法については文献（4）が，また最近のGLR法の研究については文献（20）が参考になる．チャート法を含む構文解析アルゴリズム全般については文献（13）を参照されたい．

文　献

（1） Kay M.：“Algorithm Schemata and Data Structure in Syntactic Processing”, TR CSL80-12, Xerox PARC (1980).

（2） Winograd T.A.：“Language as a Cognitive Process- 1 Syntax”, Addison-Wesley (1983).

（3） Knuth D.E.：“On the Transition of Languages from Left to Right”, Inf. Control, 9, pp.607-639 (1965).

（4） Tomita M.：“Efficient Parsing for Natural Language”, Kluwer Academic Publishers (1986).

（5） Aho A.V., Sethi R. and Ullman J.D.：“Compilers Principles, Techniques and Tools”, Addison-Wesley (1986).

（6） Shann P.：“Experiments with GLR and Chart Parsing”, in Ref. (20), pp.17-14 (1991).

（7） Wetherell C.S.：“Probabilistic Language：A Review and Some Open Questions”, Comput. Surv., 12, 4, pp.361-379 (1980).

（8） Fujisaki T., Jelinek F., et al.：“A Probabilistic Method for Sentence Disambiguation”, in Proc. of 1st International Workshop on Parsing Technologies, CMU, Pittsburgh, pp.105-114 (1989).

（9） Wright J.H.：“LR Parsing of Probabilistic Grammars with Input Uncertainty for Speech Recognition”, Comput. Speech Lang., 4, pp.297-323 (1990).

(10) Lari K. and Young S.J.：“The Estimation of Stochastic Context-Free Grammars Using the Inside-Outside Algorithm”, Comput. Speech Lang., 4, pp.35-56 (1990).

(11) Briscoe T. and Carroll J.：“Generalized Probabilistic LR Parsing of Natural Language Corpora with Unification-Based Grammars”, Comput. Liguist., 19, 1, pp.25-59 (1993).

(12) Kipps J.N.：“GLR Parsing in Time O (n^3)”, in Ref. (20), pp.43-59 (1991).

(13) 田中穂積：“自然言語解析の基礎”，産業図書(1989).

(14) Nijholt A.：“Overview of Parallel Parsing Strategies”, in Tomita, T. (ed.)：“Current Issues in Parsing Technology”, Kluwer Academic Publishers, pp.207-229 (1991).

(15) Tanaka H., Tokunaga T. and Aizawa M.：“Integration of Morphological and Syntactic Analysis based on LR Parsing”, Proc. of Third International Workshop on Parsing Technologies, pp.101-109 (1993).

(16) 伊藤克亘，速水　悟，田中穂積：“音素文脈依存モデルと高速な探索手法を用いた連続音声認識”，信学論(D-Ⅱ), J75-D-Ⅱ, 6, pp.1023-1030 (1992-06).

(17) 北　研二，川端　豪，斉藤博昭：“HMM音韻認識と拡張LR構文解析法を用いた連続音声認識”，情処学論，31，3，pp.472-480 (1990-03).

(18) Lee K.F.：“Automatic Speech Recognition：The Development of the SPHINX System”, Kluwer Academic Publishers (1989).

(19) Nagai A., Sagayama S. and Kita K.：“Three Differnt LR Parsing Algorithms for Phonme-Context-Dependent HMM-Based Continuous Speech Recognition”, IEICE Trans. Inf. & Syst., E76-D, 1, pp.29-37 (Jan. 1993).

(20) Tomita M. (ed.)：“Generalized LR Parsing Technologies”, Kluwer Academic Publishers (1991).

田中　穂積（たなか　ほずみ）（正員）

昭39東工大・理工・制御工学卒，昭41同大学院修士課程了．同年電気試験所（現電子技術総合研究所）入所．昭58東工大工学部情報工学科助教授，昭61同大教授，工博．人工知能，自然言語処理の研究に従事．