LR表への複数の接続制約の組み込みによる一般化LR法の拡張

橋本泰一　植木正裕　白井清昭　徳永健伸　田中穂積
東京工業大学大学院 情報理工学研究科 計算工学専攻

1 はじめに

文脈自由文法 (CFG) を用いた構文解析アルゴリズムの一つに一般化 LR 法 (GLR 法) がある [6]. GLR 法は, CFG を扱えるように LR 法を拡張したものである [1,3]. GLR 法では, 与えられた CFG から LR 表と呼ぶ構文解析用の動作表を作成し, その表に従って解析を行う. 解析動作は, 先読み記号と状態番号によって定義される. GLR 法は, 無駄のない解析を行うため経験的に最も効率の良いアルゴリズムとされている.

CFG の問題点として, 隣接する形態素間の接続制約を記述すると規則数が多くなり繁雑になることが挙げられる. 田中らは, この接続制約を CFG とは別の形式 (接続表) で記述し, それを LR 表に組み込む手法を提案している [4,5]. このように接続制約と文法規則を個々に記述することで接続制約を考慮しながら文法を作成する必要がなくなるため作成コストが削減できる.

言語には音素, 文字, 単語, 形態素などのように複数の記号のレベルが存在する. 各レベルの記号間にはそれぞれ接続制約が存在する. そこで, 複数のレベルの接続制約をまとめて LR 表に組み込む手法が提案されている [7,8]. この手法で作成した LR 表には GOTO 部が存在しない. 従来の GLR 法のアルゴリズムを修正する必要があり, 田中らは拡張 GLR 法と呼ぶ新たな解析アルゴリズムを提案している. しかし, このアルゴリズムに工夫を施こさなければ冗長な解析を行う可能性があり, 効率が悪い.

本論文では, 拡張 GLR 法について概説し, 従来の GLR 法の効率化の手法 (パックとマージ) の適用について述べる. さらに, その効率化の手法では回避できない冗長な解析が存在することを指摘する.

2 拡張GLR法の概説

複数の接続制約を LR 表に組み込むためには, 与えられた CFG が層を持つ必要がある. n 層を持つ CFG G_n は, その部分集合として, n - 1 層を持つ CFG G_{n-1} を含む.

この複数の層を持つ CFG から各層毎に終端記号の接続制約を組み込んだ LR 表を作成することが可能である. 接続制約を組み込むことで, LR 表上のアクションが削減され, 接続制約に違反する解析を未然に防ぐことができる. しかし, 複数の接続制約を組み込むために通常の LR 表とは異なる LR 表を作成しなければならない. 2 層の CFG と各層の接続表とその文法と接続表から作成した LR 表を図 1 に示す. ただし, 文法は, 後述する解析例の説明用の人工的な文法である. 通常の LR 表との変更点は, 以下通りである.

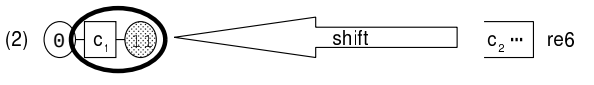
- GOTO 部がなく, アクション部のみ存在
- 非終端記号も先読み記号となる

LR 表の変更に伴い, 解析アルゴリズムも変更する必要がある. 田中らは, GLR 法の解析アルゴリズムを基に, 文脈依存性を扱い, しかも LR 法をベースにした決定的な解析を行うことが可能な CS(k) パーザ [2] の考え方を取り入れ, 2 本のスタックを用いて解析を行う手法を提案した. 2 本のスタックは, それぞれ解析スタック, 入力スタックと呼ぶ.[1] これらを用いる解析動作を次のように変更した.

- reduce 動作は, 解析スタックをポップしてから, 対応する規則の左辺記号を入力スタックにプッシュする. 非終端記号が先読み記号となる.
- shift 動作は, 入力スタックの先頭をポップし, 解析スタックにプッシュする.

このように GLR 法の解析アルゴリズムを拡張することで, 複数の接続制約を組み込んだ LR 表を用いて解析を行うことができる. 図 1 の文法と LR 表を用いて "c_1 c_2 c_3 c_4 c_5 c_6 \$" という文を解析する.

shift 動作で入力スタックのトップのシンブルを解析スタックにプッシュする.



[1] 従来の GLR 法では, 解析スタックはグラフ構造化スタック, 入力スタックは入力文であり, 先読み記号は入力スタックの先頭から得られる. 解析が進むにつれて, 入力スタックは短くなる一方で, 伸びることはない.

1. S → PP S
2. S → PP v
3. PP → NP p
4. NP → NP n
5. NP → n

(G_2)

6. n → c_1
7. p → c_2
8. n → c_3
9. n → c_4
10. n → c_3 c_4
11. p → c_5
12. v → c_6

(G_1)

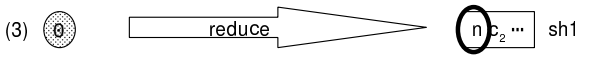
	c_1	c_2	c_3	c_4	c_5	c_6	$
c_1	0	1	0	0	0	0	0
c_2	0	0	1	0	0	0	0
c_3	0	0	0	1	0	0	0
c_4	0	0	0	0	1	0	0
c_5	0	0	0	0	0	1	0
c_6	0	0	0	0	0	0	1

	n	p	v	$
n	1	1	0	0
p	1	0	1	1
v	0	0	0	1

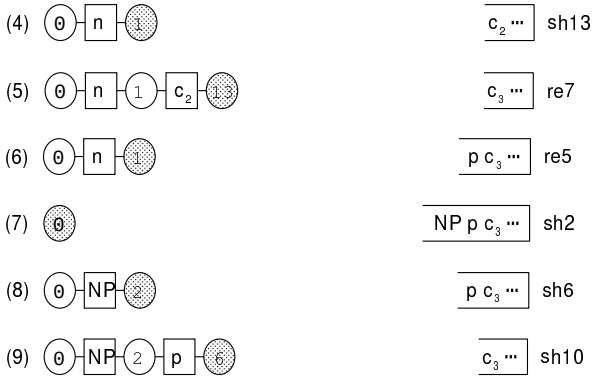
	c_1	c_2	c_3	c_4	c_5	c_6	$	n	p	v	NP	PP	S
0	sh11		sh10	sh9				sh1			sh2	sh3	sh4
1		sh13		sh9	sh12			re5	re5				
2								sh5	sh6				
3								sh1		sh7	sh2	sh3	sh8
4							acc						
5					sh12			re4	re4				
6			sh10			sh14		re3		re3			
7							re2						
8							re1						
9					re9								
10				re8/sh15									
11		re6											
12						re11							
13			re7										
14							re12						
15					re10								

図 1: サンプル文法と LR 表と接続表

reduce 動作で解析スタックをポップしてから, 対応する規則の左辺記号を入力スタックにプッシュする.



以後, 解析は次のように進む.



3 拡張GLR法への効率化手法の適用

前節で述べた拡張 GLR 法では, 解析に曖昧性が生じた場合について何も述べていない. 最も簡単な解決法として, 解析に曖昧性が生じたとき解析スタックを複製する方法が挙げられる. しかし, この手法では同じ解析を何度も行う可能性があるために効率が悪い. 一方, 従来の GLR 法では, グラフ構造化スタック, 圧縮共有構文森と呼ぶ 2 つのデータ構造とパック, マージと呼ばれる 2 つの手法で解析を効率化する手法が提案されている. これらの GLR 法の効率化の手法は拡張 GLR 法にもちろん組み込める. この場合, 解析動作の同期は, 入力スタックのステージを利用する[2].

3.1 先読み木

GLR 法では, 先読み記号が前終端記号であったが, 拡張 GLR 法では, 非終端記号も先読み記号となり得るため, 同じ記号であっても構文情報が異なる可能性がある. この構文的な違いを区別するために先読みは, 記

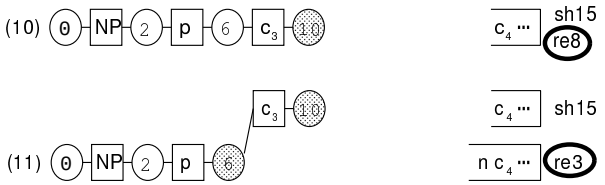
[2]ステージについては [5] を参照

号ではなく構文木で表現する．これを先読み木と呼ぶ．前終端記号の場合は, 子供を持たないノードと考える．構文的な情報を保持する理由については後述する．また, 先読み木の開始ステージとは, その先読み木が包含する入力文の範囲が始まるステージのことである．例えば, 例文の “c_3” から “c_5” まで包含する先読み木であれば, 開始ステージは 2 ステージとなる．

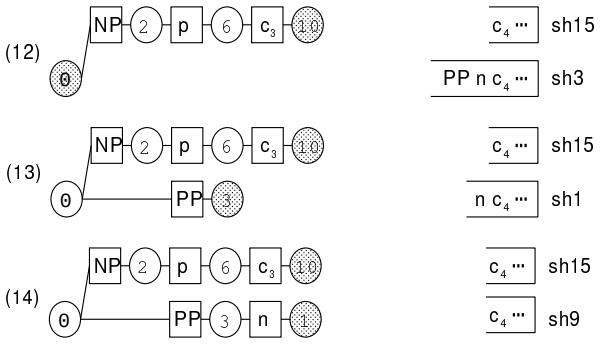
3.2 解析例

先程の解析をさらに進めると解析動作に曖昧性が生じる．

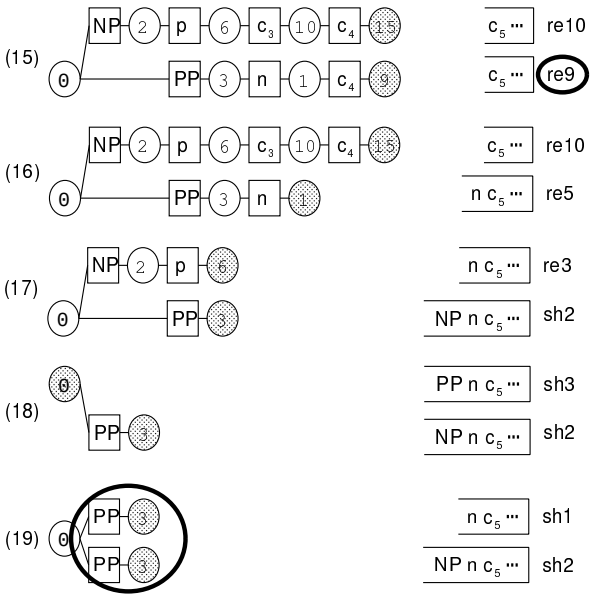
実行すべき複数の解析動作について, 先読み木の開始ステージが同じ場合, shift 動作よりも reduce 動作を優先する．(10) の場合, sh15 と re8 の二つの実行すべき解析動作がある．両者の先読み木 (c_4) は同じであるので, 先読み木の開始ステージも同じである．先の条件より, re8 を優先して行う．



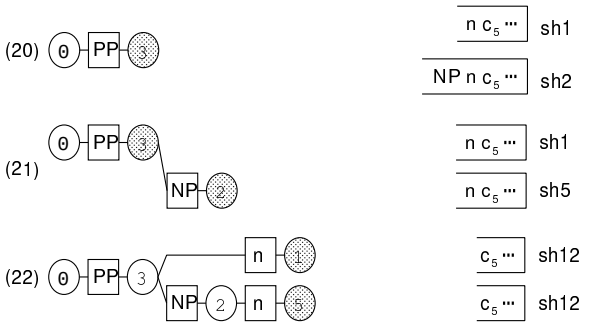
実行すべき複数の解析動作について, 各動作の先読み木の開始ステージが異なる場合, 先読み木の開始ステージが最も小さい解析動作を行う．(11) の場合, sh15 の先読み木 (c_4) の開始ステージは, 3 ステージであり, re3 の先読み木 (“n”) の開始ステージは, 2 ステージである．先の条件より, re3 を優先して行う．



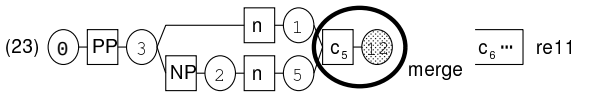
先読み木の開始ステージが同じ reduce 動作が複数ある場合は, ポップするステージ数が少ない reduce 動作を優先する．(15) の場合, re10 と re9 の先読み木は同じであるので, 開始ステージは一致する．re10 を行うと 2 ステージ (c_3) まで解析スタックをポップするが, re9 を行うと 3 ステージ (c_4) までしかポップしない．よって, ポップするステージ数の少ない re9 を優先する．

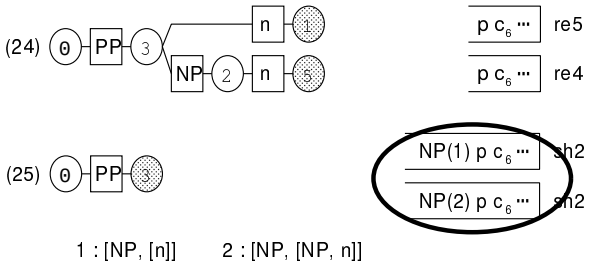


(19) の二本の解析スタックの “PP” は同じ構文木である．“PP” の直前で分岐している二本のスタックは同一なので一本にすることができる．このような状態になる原因については第 4 節で述べる．

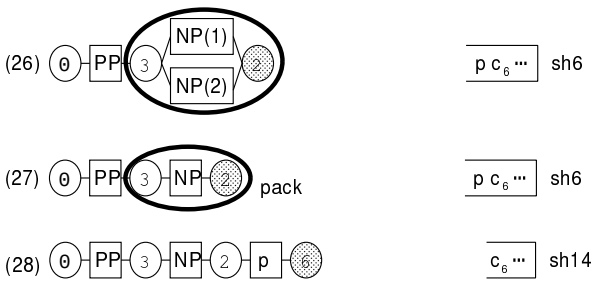


複数の解析スタックのトップの状態とステージ数が一致し, 各入力スタックが一致する場合にのみ解析スタックをマージできる．間違った解析パスを生成するのを防ぐために入力スタック上のすべての記号が構文的に一致するかどうか調べなければならない．例えば, (25) の二本の入力スタックは一致するとは言えない．





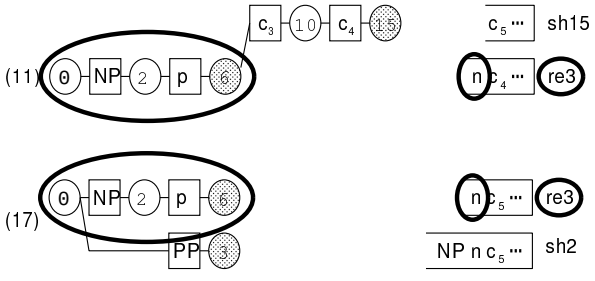
パックは, 従来の GLR 法と同様の条件で行う. 例えば, (26) の場合, “NP” がパック可能である. パック後の解析スタックは, (27) のようになる.



以降, 解析は進み, 入力文は受理される.

4 考察

前節で拡張 GLR 法にも従来の GLR 法の効率化の手法が適用可能であることを示した. しかし, 従来の効率化の手法では解決できない拡張 GLR 法独自の問題点がある. 解析例の (11) と (17) に注目する. 両者共に re3 が実行されるが, この二つの解析動作は同じ解析スタックのトップに対して行われる. つまり, 同じ解析を 2 度行っているのである.



このような冗長な解析の原因は, 先読み木 (n) が生成されるタイミングがずれているためである. 解析動作を行う順序を制御して, 先読み木 (n) が生成されるタイミングを合わせることができれば, 冗長であった解析動作 (re3) を同時に行うことができる.

本論文では, 従来の GLR 法に従い入力スタック (先読み木) のステージ数により解析動作の同期をとった. 解析動作の同期のとり方を変えて, 先読み木の記号のレベルを使って解析動作の同期をとれば, 先読み木の生成のタイミングを合わせることが可能である. しかし, 特定の解析スタックの解析が先行し, 入力スタックのステージ数がそろえられず, 入力を逐次的に処理する場合に向いていないという問題点もある.

5 おわりに

本論文では, 複数の接続制約を組む込んだ LR 表を用いた拡張 GLR 法について概説し, 従来の GLR 法の効率化の手法が拡張 GLR 法への適用可能であることについて述べた. また, 従来の手法では回避できない冗長な解析が存在することを指摘した. 今後の課題として, このアルゴリズムを実装し, 実際の効率化の効果とともに, 従来の手法では回避不可能な冗長な解析の影響について評価実験を行うことが上げられる.

参考文献

[1] A.V. Aho, S. Ravi, and J.D. Ullman. *Compilers, Principle, Techniques, and Tools*. Addision Wesely, 1986.

[2] Walter D.A. Deterministic context-sensitive languages: Part 1. *Information and Control*, 1980.

[3] D.E. Knuth. On the translation of languages left to right. *Information and Control*, Vol. 8, No. 6, pp. 607–639, 1965.

[4] H. Li. Integrating connection constraints into a GLR parser and its applications in a continuous speech recognition system. Technical Report TR96-0003, Department of Computer Science, Tokyo Institute of Technolog, 1996.

[5] Hozumi Tanaka, Takenobu Tokunaga, and Michio Aizawa. Integration of morphological and syntactic analysis based on LR parsing algorithm. 自然言語処理, Vol. 2, No. 2, pp. 59–74, 4 1995.

[6] M. Tomita. An efficient augmented-content-free parsing algorithm. *Computational Linguistics*, Vol. 13, No. 1-2, 1987.

[7] 綾部寿樹. 複数の接続表の制約の LR 表への組み込みと実装化. Master’s thesis, 東京工業大学大学院情報理工学研究科, 1998.

[8] 田中穂積, 今井宏樹, 白井清昭. 複数の接続表の制約を組み込んだ LR 表の生成と GLR 法の拡張. 言語処理学会第 5 回年次大会ワークショップ論文集, 1999.