1995年度 人工知能学会全国大会（第9回）

語順の自由性を考慮した文法のLR表作成アルゴリズムに関する研究
Generating an LR table from Context-Free Grammar with flexible word order

○亀谷 由隆*, 田中穂積
Yoshitaka KAMEYA, Hozumi TANAKA
東京工業大学情報理工学研究科 計算工学専攻
Department of Computer Science, Tokyo Institute of Technology

平成7年 6月 14日

In this paper, we propose a new grammatical form (named 'scrambled CFG') synthesized from Context Free Grammar(CFG) and ID/LP grammar which provides word order flexibility. While CFG is incompatible with natural language with flexible word order, Scrambled CFG is more natural and compact. Moreover, we propose a method of GLR parsing specialized for scrambled CFG. Since GLR parsing has been considered efficient empirically, this method is more efficient than ID/LP parsing. We can show an advantage that an LR table for scrambled CFG has much less number of states than the one for the equivalent CFG.

1 はじめに

自然言語は多種でそれぞれの言語の性質も一様ではない．例えば語順なども言語によって自由度が大きく異なる．

文脈自由文法 (Context Free Grammar, 以下ではCFG) 形式は，右辺の記号列の並び (すなわち語順) に関して厳密であるので，日本語のように語順の自由性を容認する言語とCFG形式は相性が悪い．

この問題を解決するために語順の自由性を吸収するID/LP文法が提案されているが，Shieberの方法などでは統語解析時の効率に大きな問題が残る [2]．

我々はShieberのアルゴリズムの中に見られるような記述形式をベースとし，語順の自由性を吸収できるようにCFGの形式を若干拡張した．これはID/LP文法とCFGの中間のような文法規則の形式である．この文法形式を「かき混ぜCFG(Scrambled Context Free Grammar)」と呼ぶ．

この文法形式に経験的に効率がよいとされる富田らのGLR(Generalized LR, 一般化LR)法 [5] を用いてパーズする新しい方法を提案する．そのためにはかき混ぜCFGに適応したLR表を作成することが必要である．かき混ぜCFGのLR表作成アルゴリズムはCFGのLR表作成アルゴリズムを拡張することによって得られる．

本研究ではかき混ぜCFGによる文法記述形式とそれをGLR法で統語解析する手法の利点と問題点を明らかにすることを目的とする．そして本論文で提案した文法形式と解析手法が他の手法に比べて優れていることを示す．

2 語順の自由性を考慮した文法の記述

2.1 CFGでの表現

CFGにおいて語順の自由性を表現する場合に，はじめに考えられるのは (1) のように可能な要素の並びを全て列挙することである．

(1) $S \rightarrow abc$ $S \rightarrow bca$
$S \rightarrow acb$ $S \rightarrow cab$
$S \rightarrow bac$ $S \rightarrow cba$

この方法では規則数が非常に大きくなり，文法の規模が大きくなると解析に支障をきたす恐

*〒 227 東京都目黒区大岡山 2-12-1 TEL.(03)5734-2186 FAX(03)5734-2186 email:kame@cs.titech.ac.jp

れがある．他にも方法は考えられるが，どれも語順の自由な文法の記述には不十分である．

2.2 ID/LP 文法での表現

ID/LP 文法は Gazdar らの GPSG システムの一部であり，そこでは言語の文法は ID 規則，LP 規則と呼ばれる 2 つの形式によって別々に記述する [3]．ID/LP 文法 (2) は CFG(3) と等しい．

(2i) $A \rightarrow_{\mathrm{ID}} B, C, D$
(2ii) $C \prec D$

(3) $A \rightarrow BCD$
$A \rightarrow CBD$
$A \rightarrow CDB$

LP 規則は全ての ID 規則の右辺において有効な大域的 (global) 制約になっていることに注意する．

3 新しい文法記述形式

3.1 記述形式の詳細

記述例として単純な日本語のかき混ぜ CFG(4) を挙げてみる．1 つめの規則の PP_1, PP_2, PP_3 は任意であるので { } で囲む．以下ではこの文法形式を「かき混ぜ CFG(Scrambled CFG)」，{ } で囲まれた記号集合の箇所を「かき混ぜ部 (scrambled part)」と呼ぶ．

(4) $S \rightarrow \{PP_1, PP_2, PP_3\}\ v$
$PP_1 \rightarrow n\ p_1$
$PP_2 \rightarrow n\ p_2$
$PP_3 \rightarrow n\ p_3$
$p_1 \rightarrow$ が
$p_2 \rightarrow$ を
$p_3 \rightarrow$ に
$n \rightarrow$ 私
$n \rightarrow$ 彼
$n \rightarrow$ 学校
$v \rightarrow$ 連れていく

かき混ぜ部内の要素数は有限であり，各要素の出現は一度だけであることに注意しなければならない．複数回出現させたいときには $S \rightarrow \{PP_1, PP_2, PP_2, PP_3\}$ のようにかき混ぜ部に重ねて記述する．

3.2 CFG との比較

等価な変換としてかき混ぜ CFG の CFG への展開が考えられる．以下ではかき混ぜ CFG を展開した CFG を「展開 CFG(expanded-CFG)」と呼ぶ．

一般にサイズ n のかき混ぜ部を 1 つ持つかき混ぜ CFG 規則を展開すると $n!$ の CFG 規則が得られる．かき混ぜ部のサイズ (要素数) が大きい場合には爆発的な規則の増加が起こる．

3.3 ID/LP 文法との比較

ID 規則のみからなる ID/LP 文法が上のかき混ぜ CFG の形式に変換できるのは明らかであるが，LP 規則を含んだ ID/LP 文法に関しても後に述べる語順制限規則の導入によって，等価な変換が可能である．

しかし，ID/LP 文法の ID 規則の右辺記号の並びは完全に自由であり，語順が定まっている場合では LP 規則を用いなければならないのに対して，かき混ぜ CFG ではその必要がない．例えば (4) の $PP_1 \rightarrow n\ p_1$ のように CFG の形式で書けばよい．

3.4 語順制限の導入

上で見てきたような形式では，かき混ぜ部内の語順は完全に自由である．しかし必要ならば，かき混ぜ部内にも語順の並びについて制限をかける記述があった方がより自然に表現できる．

かき混ぜ CFG での LP 規則は大域的にも局所にも宣言可能であるとしている．例えば (5)-(5i),(5ii) のように記述する．(5ii) が新たに導入された語順の制限に関する記述である．

"global" は ID/LP 文法の LP 規則のように全ての規則に関する制約で，"local(1,1)" というのは規則番号 1 の規則 $S \rightarrow \{A, B, C, D\}\{A\}$ の 1 番目の記号集合 {A,B,C,D} に関する語順制限規則である，という意味である．(5ii) では global と local が矛盾しているが，その場合には local の制約を優先する．

(5i) $S \to \{A, B, C, D\}A$
$A \to a$
$B \to \{e, f\}$
$C \to c$
$D \to d$

(5ii) global: $A \prec B \prec C$
local(1,1): $C \prec B$

4 かき混ぜ CFG の LR 表作成アルゴリズム

かき混ぜ CFG の LR 統語解析アルゴリズムは CFG の LR 統語解析アルゴリズムを拡張することによって得られる.

4.1 CFG の LR アルゴリズム

Aho らのアルゴリズム [1] では次のような体系がとられていた.

- LR 表作成アルゴリズム
  - LR(1) 集合作成 ⇒ 関数 *goto*, *closure*
  - LR 表エントリーの決定
- parging アルゴリズム

4.2 拡張の概要

かき混ぜ CFG への拡張において, 注意すべき点として次のようなものがあげられる.

- parsing アルゴリズムおよび LR 表エントリー決定手続きを拡張/変更する必要はない.
- LR(1) 集合作成アルゴリズムを構成する関数 *goto*, *closure*, および FIRST 関数の拡張を行なう

以下では例としてかき混ぜ CFG の拡大文法 (6) を掲げ, その LR 表の作成手順において, 逐次拡張点を指摘する.

(6) $S' \to S$
$S \to \{A, B, C\}$
$A \to a$
$B \to b_1 \mid b_2$
$C \to c$

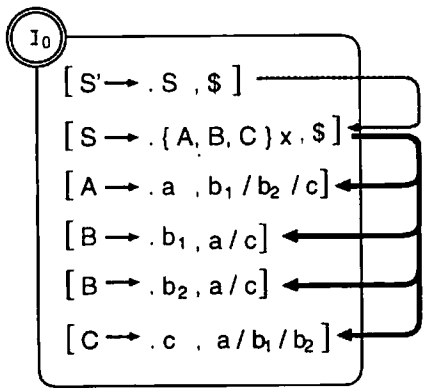

図 1: 拡張した関数 closure の動作

4.2.1 LR(1) 集合作成手続き

$[S \to \cdot\{A_1, A_2, \cdots, A_n\}, \$]$ という項があったときに, $\{A_1, A_2, \cdots, A_n\}$ のいずれも次に解析可能である, というのがかき混ぜ CFG の LR(1) 集合を作る上での基本的考えである.

まず *closure*($\{[S' \to S, \$]\}$) によって初期状態 I_0 を求める (図 1). 関数 closure の拡張により, かき混ぜ部内の全ての非終端記号 ((6) では $\{A,B,C\}$) に関して項が追加されている (太線). また FIRST 関数の拡張によって, $[A \to \cdot a, b_1/b_2/c]$ の先読みは $\{A, B, C\}$ から A を除いた $\{B, C\}$ の各要素 B と C の FIRST (それぞれ b_1, b_2 と c) である $b_1/b_2/c$ となる.

次に I_0 から拡張した関数 *goto* を用いて状態を遷移させる (図 2). このような注意を払いながら, 新たな状態が生まれなくなるまで繰り返すと, (6) の LR(1) 集合が得られるが, ここで注意すべき点は *goto*(B, I_1) と *goto*(A, I_2) は等しく I_3 であることである (図 3).

5 LR 表の特性

5.1 状態数の減少

先で拡張された LR 表作成アルゴリズムによって作成されたかき混ぜ CFG の LR 表の状態数はそれと等価な展開 CFG の LR 表のものに比べて (文法によって程度の差はあるが) かなり減少する.

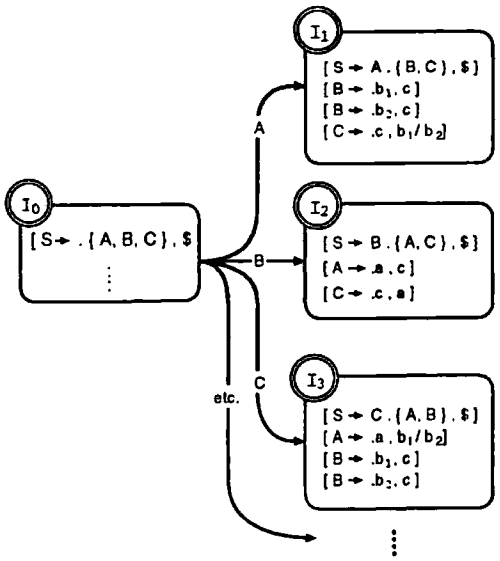

図 2: 拡張した関数 goto の動作

状態数減少は図 3に見られたような状態のマージに起因する．図 3を更に大きなエリアで見ると，状態図の形状がかき混ぜ CFG はグラフ状，展開 CFG では木状となることは容易に想像がつく．

5.2 状態の減少数

ある 1 つのかき混ぜ CFG 規則と，その規則を展開した複数の展開 CFG 規則の LR(1) 項を含んでいる状態の減少数を計算することができる (しかし文法間の状態の減少数は，かき混ぜ CFG 規則の形状に影響されるので，単純に計算することはできない)．

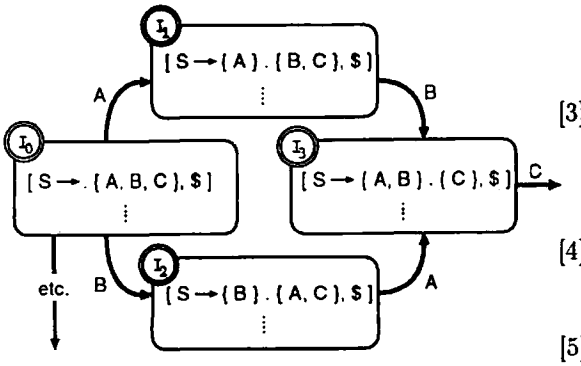

図 3: かき混ぜ CFG の拡大文法 (10) の状態遷移図 (一部)

ごく大雑把に言うと，この状態数の差は順列 (P) の和と組合せ (C) の和の違いによると考えられる．一般にある規則 r の右辺のかき混ぜ部のサイズを $[N_1, N_2, \cdots, N_n]$ とすると次のような式で状態数の差 *diff* を求めることができる．

$$diff = \sum_{k=1}^{n}\left\{\left(\prod_{m=1}^{k-1} N_m!\right)\left(\sum_{i=1}^{N_k} N_k P_i\right) - \sum_{i=1}^{N_k} N_k C_i\right\}$$

5.3 ε-規則を用いたかき混ぜ CFG における問題

かき混ぜ CFG によって種々の自然言語を記述する際に，ε-規則を用いるとより簡潔に表現できる．

残念なことに，ε-規則を用いたかき混ぜ CFG を LR 構文解析しようとするときに問題が残る．それは LR 表に shift と ε-reduce[1] のマルチエントリーが頻出することである．

しかしこの現象は展開 CFG でも現れ，かき混ぜ CFG 特有の問題ではない．

参考文献

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

[2] G. Edward Barton and Jr. On the complexity of id/lp parsing. *Computational Linguistics*, Vol. 11, No. 4, pp. 205–218, 10 1985.

[3] G. Gazdar, E. Klein, G.K. Pullum, and Sag I.A. *Generalized Phrase Structure Grammar.* Basil Blackwell, 1985.

[4] 田中穂積. 自然言語解析の基礎. 産業図書, 1989.

[5] M Tomita. *An Efficient Parsing for Natural Languages.* Kluwer, Boston, Mass, 1986.

[1] ε-規則での reduce のこと．