自然言語の構文解析のための LR 解析表の圧縮法

秋葉 友良[†] 伊藤 克亘^{††}

LR 構文解析法で利用する LR 解析表のサイズを削減する新規の手法を提案する。提案法は、(1) 従来の LR 表縮小方法と同時に適用可能、(2) 提案法によって作成された LR 表は従来の LR 構文解析アルゴリズムでほぼそのまま利用可能、(3) 解析結果や解析効率に影響を与えない、といった特徴を持つ。提案法を実際の自然言語処理用法に適用したところ、元の文法のサイズによって、約 60% 程度から、25% 程度まで、LR 表が圧縮されることを確認した。

キーワード: 構文解析, 一般化 LR 法, LR 表圧縮

A Method of LR Table Compaction for Natural Language Processing

TOMOYOSI AKIBA[†] and KATUNOBU ITOU^{††}

This paper presents a method to reduce the size of the parsing table used in the LR parsing algorithm. The proposed method has the following significant characteristics; (1) that it can be applied along with any other methods for the parsing table reduction already known today, (2) that the parsing tables constructed by it can be used in the existing LR parser without modification, and (3) that it does not affect the parsing results and the parsing efficiency. We applied the method to construct the reduced LR table from some existing grammars used for NLP, and compared the produced LR tables with the tables constructed by the ordinary method. Our method showed that the produced tables had the sizes of between 60% and 25% of their original sizes according to the grammars.

KeyWords: *Syntactic Analysis, Generalized LR Parsing, Parsing Table Compaction*

1 はじめに

LR 構文解析法は、構文解析アルゴリズムとして最も効率の良い手法の一つである。LR 構文解析法の中でも、横型探索で非決定的解析を行うことにより文脈自由言語の扱いを可能にした方法は一般化 LR 法 (GLR 法) と呼ばれ、自然言語処理および、音声認識で利用されている。また、LR 法の構文解析過程に確率を割り当てることで、確率言語モデルを得ることができる。確率一般化 LR (PGLR) モデル (Inui, Tokunaga and Tanaka 1998)、およびその一般化である APGLR モデル (Akiba and Itou 2001) は、構文解析結果の構文木の曖昧性解消や、音声認識の

[†] 産業技術総合研究所情報処理研究部門, Information Technology Research Institute, National Institute of Advanced Industrial Science and Technology

^{††} 名古屋大学大学院情報科学研究科メディア科学専攻, Dept. of Media Science, Graduate School of Information Science, Nagoya University

確率言語モデル (永井, 鷹見, 嵯峨山, ハラルド 1994; 今井, 田中 1999; Akiba and Itou 2001) として利用されている。

LR 構文解析法では, 文法が与えられた時点であらかじめ計算できる解析過程を先に求め, LR 解析表 (以下, LR 表) で表しておき, 文解析時に利用する。LR 法は, 言わば, 空間効率を犠牲にする (LR 表を作成する) ことによって, 解析時間の効率化を実現する手法である。LR 法を実際の問題に適用する場合の問題点の一つは, 文法の規則数増加に伴う LR 表のサイズの増大である。計算機言語の解析 (Aho, Sethi and Ullman 1986), 自然言語の解析 (Luk, Meng and Weng 2000), 音声認識 (永井他 1994), それぞれの立場からこの問題点が指摘されている。

LR 表のサイズを抑えるひとつの方法は, 解析効率を犠牲にして空間効率をある程度に押える方法である。本来 LR 法が利用されていた計算機言語用の構文解析においては, LR 法は決定的解析器として利用されてきた。決定的解析としての LR 法が扱える文法は, 文脈自由文法のサブセットである。LR 表は, その作り方から幾つかの種類に分類されるが, それらは決定的解析で扱える言語に違いがある。単純 LR (Simple LR; SLR) 表は, 作り方が単純で表サイズを小さく抑えられるが扱える文法の範囲が狭い。正準 LR (Canonical LR; CLR) 表は, サイズは非常に大きくなるが扱える文法の範囲は最も広い。両者のバランスを取る LR 表として, サイズを小さく抑えつつ扱える文法の範囲をそこそこ広くとれる, LALR (Look Ahead LR) 表が提案されている。一方, 文脈自由文法を扱う自然言語処理で LR 表を利用する場合は, 非決定的解析として利用するのが普通である。決定的解析で扱える言語の大きさは, 非決定的解析での解析効率に相当する。すなわち, SLR, LALR, CLR の順に効率は良くなるが, それに伴い表のサイズは増大する。

また, 計算機言語に用いる LR 表のサイズ圧縮手法には, 2 次元配列としてのスパースな表をいかに効率よく圧縮するかという視点のものも多い。これらは, 作成後の表を表現するデータ構造に工夫を行ったもので, 表自体が運ぶ情報には違いがない。自然言語処理の分野でも, 解析表縮小の手法が提案されている。田中らは, 文脈自由文法と単語接続の制約を切り放して記述しておき, LR テーブル作成時に 2 つの制約を導入する手法 (MSLR 法) (Tanaka, Tokunaga and Aizawa 1995) を用いることで, 単独の文脈自由文法を記述するより解析表のサイズを小さくすることができたと報告している (田中, 竹澤, 衛藤 1997)。Luk らは, 文法を小さな部分に分割して, それぞれを扱うパーザを組み合わせることで, 解析表のサイズを抑える方法を提案している (Luk et al. 2000)。

以上の従来手法をまとめると, 次の 3 つの手法に分類できる。

- (1) 処理効率を犠牲にして空間効率を稼ぐ方法。
- (2) 表のデータ構造を工夫して記憶量を引き下げる方法。
- (3) 文法の記述方法を工夫してより小さな表を導出する方法。

本稿では, LR 表のサイズを圧縮する, 上記の 3 分類には当てはまらない新規の手法を提案

する。提案法は従来の手法と異なり、LR 表作成アルゴリズムの再検討を行い、解析に不要な情報を捨象することによって、表の圧縮を実現する。本手法は、次のような特徴を持つ。(1) 上記の従来の縮小手法とは手法の軸が異なるため、どの手法とも同時に適用可能である¹。(2) 入力文の構文木を得るという自然言語処理用途において、提案法は解析時の効率に影響をあたえることはない²。(3) 従来の表作成および解析アルゴリズムへの変更箇所は小さく、プログラムの軽微な修正で適用可能である。特に、提案法によって作成された圧縮 LR 表は、既存の LR 構文解析プログラムではそのまま利用可能である。

本稿の構成は以下の通りである。まず 2 節で、提案法の基本原理を説明する。また、提案法の性質を考察する。続く 3 節では、提案法の実装方法と、実際の文法に提案手法を適用した実験結果を示す。4 節では、提案手法の限界を克服するための拡張方法について述べ、実際の文法に適用した結果を報告する。5 節では、関連研究について述べる。

2 LR 解析表の圧縮

本稿で提案する LR 解析表の圧縮方法について、その原理と性質について述べる。

2.1 提案法の基本原理

LR 構文解析法は、LR 表と、スタック、先読み語を参照し、次の動作を決定する。LR 表は、文法が与えられた時点であらかじめ計算できる構文解析過程を表したものである。LR 表の作成(付録.A 参照)には、構文解析中のある状態を表すデータ構造として、LR 項 (LR Item) を用いる。LR 項とは、ある生成規則 (CFG 規則) の右辺の記号列中のある位置にドット ‘ $\cdot$ ’ を付けたデータ構造である。ドットは、記号列のどの部分まで解析が進んだかを表す。下記の LR(0) 項³は、生成規則 $A \rightarrow \alpha$ において、 α の先頭部分列 β まで解析が終了し、その後記号列 γ を解析する必要がある状態を表す。

$$[A \rightarrow \beta \cdot \gamma]$$

β や γ は空記号列の場合も含まれる。例えば、生成規則 $A \rightarrow B_1 B_2 B_3$ に対して、 $[A \rightarrow \cdot B_1 B_2 B_3]$, $[A \rightarrow B_1 \cdot B_2 B_3]$, $[A \rightarrow B_1 B_2 \cdot B_3]$, $[A \rightarrow B_1 B_2 B_3 \cdot]$ の 4 つの LR 項が考えられる。このように、従来の LR 表で用いられる LR 項は規則毎に作成され、LR 項だけで解析中の規則とその解析位置、解析された記号列 (β) およびこれから解析すべき記号列 (γ) を特定することができる。

¹ ただし、MSLR 法 (Tanaka et al. 1995) との同時適用には、表作成に若干の修正が必要である。MSLR 法では、提案法で解析に不要とする情報の一部を利用しているためである。MSLR 法への対応方法については、付録.B で述べる。

² 計算機言語の構文解析では、解析時に規則に付随するアクション (プログラム) を実行することが要求される。提案法による圧縮 LR 表では適用される CFG 規則は解析時に動的に求まるので、規則から付随するアクションを検索する処理の分オーバーヘッドが生じる。入力文から構文木を得ることを目的とする自然言語処理用途では、このオーバーヘッドは生じない。

³ 説明では、LR(0) 項を用いるが、提案法は、LR(1) 項にも適用可能である。すなわち、SLR, LALR, CLR などの LR 表の種類に依らず適用可能である。

一方、LR 構文解析法では、解析途中の状態を表現するデータ構造として、解析済の(終端および非終端)記号を記憶する解析スタックが併用される(文脈自由言語を受理するプッシュダウン・オートマトンのスタックに相当する)。上の例の LR 項が表す状態では、解析スタックには、記号列 β が、スタックトップから逆順に保持されているはずである。例えば、LR 項 $[A \rightarrow B_1 B_2 \cdot B_3]$ が表す状態では、スタックは $[\dots B_1, i, B_2, j]$ (ただし、 i, j は状態番号、スタックトップは右) となる。

したがって、すでに解析済の記号列は、LR 項と解析スタックに重複して記載されており、冗長である。そこで、LR 項から冗長な解析済記号列 β の記号情報を捨象することが可能となる。ここで、捨象可能なのは、記号情報だけで、記号数の情報は保持する必要があることに注意されたい。解析スタックには、解析済の記号がその記号が属する規則に関係なくフラットに保持される。スタック $[\dots, B_1, i, B_2, j, B_3, k]$ を見ただけでは、記号 B_3 が規則 $A \rightarrow B_3$, $A \rightarrow B_2 B_3$, $A \rightarrow B_1 B_2 B_3$ のいずれに属する B_3 なのか、区別することができない。そこで、LR 項が表す解析状態で、スタックトップからいくつの記号がこの規則で解析中かという情報を保持しなければならない。

以上のことから、従来の LR 項のドットの左側の記号列を抽象化して、その記号の個数で置き換え、新しい LR 項とする。すなわち、次のような LR 項を用いる。

$$[A \rightarrow |\beta| \cdot \gamma]$$

ここで、 $|\beta|$ は、記号列 β の記号数を表すものとする。この LR 項を左方抽象化 LR 項と呼ぶことにする。例えば、生成規則 $A \rightarrow B_1 B_2 B_3$ に対して、 $[A \rightarrow 0 \cdot B_1 B_2 B_3]$, $[A \rightarrow 1 \cdot B_2 B_3]$, $[A \rightarrow 2 \cdot B_3]$, $[A \rightarrow 3 \cdot]$ の 4 つの左方抽象化 LR 項が考えられる。

LR 表は、生成規則集合が与えられた時点で、そこから求められる構文解析時のあらゆる解析途中の状態を抽出し、また各状態間の遷移関係を求めて、状態遷移図(プッシュダウンオートマトン)として表現したものである。ここで解析途中の状態は、LR 項の集合(クロージャ)に対応し、規則適用の部分的解析結果(LR 項に相当)の複数の可能性(集合)を表している。この時、解析状態の同一性は、クロージャの同一性で判断される。例えば、異なる解析パスから、同一の LR 項集合が得られる場合、それらは同じ状態とみなすことができる。

クロージャの要素として、従来の LR 項の代わりに、不必要な情報が捨象された左方抽象化 LR 項を用いると、より多くの解析状態(クロージャ)が同一の状態と見なされる。したがって、結果として得られる LR 表の状態数が減少し、表のサイズが縮小される。これが提案手法の原理である。

2.2 圧縮 LR 表作成アルゴリズム

LR 表作成アルゴリズムは, 従来の LR 項の代わりに, 上記の左方抽象化 LR 項を用いても, 新たな処理を加えること無く若干の修正だけで適用できる. 以下では, LR 表作成手順の [クロージャ] [GOTO 手続き] [LR 項集合の集合] [LR 表の作成] の各手続き (付録.A 参照), それぞれについて, 修正手続きを示す. (この手法を, **提案法 1** と呼ぶ. また提案法 1 によって作成された LR 表を**圧縮 LR 表**と呼ぶ.)

[クロージャ], [GOTO 手続き], [LR 項集合の集合] では, LR 項のドットの左側を参照する手続きが存在しないので, 左方抽象化 LR 項を用いて, ほぼそのまま, 「LR 項集合の集合」を作成できる. 唯一, ドットの左側の生成・修正の手続きに若干の変更を加える. 具体的には, 以下の 2 点を変更する.

[クロージャ] 作成手続きのステップ 2 を次のように変更する.

- (2) $\text{closure}(I)$ に左方抽象化 LR 項 $[B \rightarrow |\beta| \cdot A\gamma]$ があれば, 左辺が A の全ての生成規則 $A \rightarrow \alpha$ について, LR 項 $[A \rightarrow 0 \cdot \alpha]$ を加える. この手続きを新たな LR 項が加えられなくなるまで繰り返す.

また, [GOTO 手続き] $\text{GOTO}(I, A)$ を次のように変更する.

- I 中の, ドットのすぐ右が B である全ての LR 項 $[A \rightarrow |\beta| \cdot B\gamma]$ に対し, LR 項 $[A \rightarrow |\beta| + 1 \cdot \gamma]$ を求め, そのクロージャを返す.

得られた LR 項集合の集合から [LR 表の作成] において, action 表への shift や accept の書き込み, および goto 表の作成にも変更点はない. 一方, reduce 動作の引数には, 従来の生成規則の代わりに, 生成記号の左辺記号と右辺記号列の記号数のペアを記述する. すなわち, [LR 表の作成] の reduce 動作の書き込みを次のように変更する.

- I_i に LR 項 $[A \rightarrow |\alpha| \cdot]$ が存在するならば, $a \in \text{follow}(A)$ について, action(i, a) に reduce $|\alpha|, A$ を加える.

このように, 修正手続きで作成される LR 表では, reduce 動作の引数には生成規則のうち, 左辺の記号列の記号情報を捨象した一部の情報しか記述されない. しかし, この情報 $|\alpha|$ と A さえあれば, どの生成規則が適用されたかは, 構文解析時に特定可能である点に注意されたい. 左辺記号 A と, 解析スタックから $|\alpha|$ 個の記号をポップすることで得られる記号列 α から, $A \rightarrow \alpha$ と復元可能である.

また, 提案手法での修正箇所は, 既存の表作成アルゴリズム中のある手続きを同等の手続きで置き換えただけであり, 新たな手続きの呼び出しは行っていない. したがって, 従来の表作成アルゴリズムと同じ計算オーダで作成可能である.

2.3 提案法の適用例

日本語において、動詞の格を表す句は、語順が自由であり、任意に省略可能であることが多い。次のような日本語を解析する文法 (図 1) を考える。

太郎が 花子に りんごを 与える
 花子に 太郎が りんごを 与える
 りんごを 太郎が 花子に 与える
 太郎が りんごを 与える
 りんごを 与える
 与える

この文法から、従来の表作成アルゴリズムと、提案法による表作成アルゴリズムによってつくられた GOTO グラフの一部 (記号 'S' を左辺に持つ規則の集合に相当する部分) を、それぞれ図 2, 図 3 に示す。従来法では、規則右辺に現れる記号列の文脈によって異なる状態が作成されるため、木の形に分岐したグラフが作成される。一方提案法では、記号列の文脈によらずに後方部分がマージされたグラフが作成され、状態数が 32 から 12 へ大幅に減少することが分かる。

2.4 提案法の効果と解析効率

提案法の効果と効率について、その性質を考察する。

圧縮の効果

LR 項においてドットより左方の記号列が記号数へと抽象化されることにより、従来異なる LR 項として認識された以下のような 2 つの LR 項が、同一の LR 項として認識されることになる。

$$[A \rightarrow \beta_1 \cdot \gamma] \quad , \quad [A \rightarrow \beta_2 \cdot \gamma]$$

$$\text{where } |\beta_1| = |\beta_2|$$

これらから生成されるクロージャも同一のクロージャとなる。すなわち、「LR 項集合の集合」作成時に、従来異なる状態 (LR 項集合) となっていたものが、1 つの状態にマージされることになる。よって、最終的な LR 表の状態数は減少する。

ここで、マージされる LR 項が存在するための、文法 G の必要条件を考察する。上記の同一視される 2 つの LR 項について、 β_1 と β_2 以外は等しくなければならない。また、 $|\beta_1| = |\beta_2|$ 、すなわち $|\beta_1 \gamma| = |\beta_2 \gamma|$ である必要がある。以上をまとめると、文法 G に、以下の条件を満たす規則のペア $A_1 \rightarrow \alpha_1$, $A_2 \rightarrow \alpha_2$ が少なくとも一組以上存在する必要がある。

- 左辺記号 A_1 , A_2 が等しい。

$S \rightarrow PP_HUMAN_ga \ PP_HUMAN_ni \ PP_OBJECT_wo \ VP$
 $S \rightarrow PP_HUMAN_ga \ PP_OBJECT_wo \ PP_HUMAN_ni \ VP$
 $S \rightarrow PP_HUMAN_ni \ PP_HUMAN_ga \ PP_OBJECT_wo \ VP$
 $S \rightarrow PP_HUMAN_ni \ PP_OBJECT_wo \ PP_HUMAN_ga \ VP$
 $S \rightarrow PP_OBJECT_wo \ PP_HUMAN_ga \ PP_HUMAN_ni \ VP$
 $S \rightarrow PP_OBJECT_wo \ PP_HUMAN_ni \ PP_HUMAN_ga \ VP$
 $S \rightarrow PP_HUMAN_ni \ PP_OBJECT_wo \ VP$
 $S \rightarrow PP_OBJECT_wo \ PP_HUMAN_ni \ VP$
 $S \rightarrow PP_HUMAN_ga \ PP_OBJECT_wo \ VP$
 $S \rightarrow PP_OBJECT_wo \ PP_HUMAN_ga \ VP$
 $S \rightarrow PP_HUMAN_ga \ PP_HUMAN_ni \ VP$
 $S \rightarrow PP_HUMAN_ni \ PP_HUMAN_ga \ VP$
 $S \rightarrow PP_OBJECT_wo \ VP$
 $S \rightarrow PP_HUMAN_ni \ VP$
 $S \rightarrow PP_HUMAN_ga \ VP$
 $S \rightarrow VP$
 $PP_HUMAN_ga \rightarrow NP_HUMAN \text{ が}$
 $PP_HUMAN_ni \rightarrow NP_HUMAN \text{ に}$
 $PP_OBJECT_wo \rightarrow NP_OBJECT \text{ を}$
 $NP_HUMAN \rightarrow \text{太郎}$
 $NP_HUMAN \rightarrow \text{花子}$
 $NP_OBJECT \rightarrow \text{りんご}$
 $NP_OBJECT \rightarrow \text{本}$
 $VP \rightarrow \text{与える}$
 $VP \rightarrow \text{渡す}$

図 1 文法例

- α_1 と α_2 の接尾記号列が一致する. すなわち, $\alpha_1 = \beta_1\gamma_1$, $\alpha_2 = \beta_2\gamma_2$ と書けるととき, $\gamma_1 = \gamma_2$ となる γ_1, γ_2 が存在する.
- $|\alpha_1| = |\alpha_2|$

このような規則のペアが, 文法 G 中に多く存在するほど, 提案法による状態数の削減の効果は大きい.

図 2 従来法による GOTO グラフ

解析の効率

圧縮 LR 表を用いて構文解析を行った場合の効率について考える。提案法は、すでに解析済の情報 (ドットの左側の記号情報) だけを捨象する。これから解析する部分 (ドットの右側の記号情報) には手を加えない。提案法によって状態の統合が行われた場合、統合後の状態 (クロージャ) は以前の状態と同数の LR 項を持ち、それぞれの LR 項のこれから解析する部分 (ドットの右側の記号情報) も等しい。したがって、統合後もその後の解析の処理量は等しく、構文解析の効率は悪くならない。

図 3 提案法による GOTO グラフ

3 実装と実験

3.1 実装

提案法を実装するには、LR 表作成プログラム、構文解析プログラムの修正が必要となる。しかし以下に示す通り、既存の処理系のわずかな個所の修正で実装可能である。

LR 表

圧縮 LR 表では、従来の LR 表と比べ、唯一 reduce 動作の引数の意味が変更になる。従来の LR 表では、reduce 動作の引数には生成規則を指定する。実際には、規則へのポインタ (規則番号) が記述される。一方、圧縮 LR 表では、生成規則から右辺の記号情報を捨象した情報、すなわち「左辺記号と右辺の記号数」(へのポインタ) である。あらかじめ存在する生成規則のリストとは異なり、「左辺記号と右辺の記号数」のリストは表作成のために新規に導入する概念である。厳密に実装するならば、LR 表作成プログラムでこのリストを新規に作成し、構文解析プログラムとの間でこのリストを共有しなければならない。しかし、「左辺記号と右辺の記号数」は生成規則の一部であることを利用して、reduce の引数に「左辺記号と右辺の記号数」の条件を

満たす任意の規則番号を記述し代用することで、リストの受け渡しを避けることが可能である。

LR 表作成プログラム

提案法で用いる左方抽象化 LR 項は、次のような方法によって、既存の LR 表作成プログラムに比較的容易に導入することが可能である。左方抽象化 LR 項は、文法に現れない記号 X を $|\beta|$ 個ドットの左に書くことでも表現できる。例えば、 $|\beta| = 2$ とすると、次のように書ける。

$$[A \rightarrow XX \cdot \gamma]$$

このような LR 項表現は、従来の LR 項と容易に交換可能である。GOTO 手続きにおいて、LR 項のドットを右へ一つ移動する時に、飛び越えた記号を X で置き換えるように変更するだけで、従来法に組み込むことができる。

構文解析プログラム

「左辺記号と右辺の記号数」リストの参照を除けば、圧縮 LR 表は、従来の LR 構文解析アルゴリズムでそのまま利用可能である。また多くの実装系では、読み込んだ規則集合のうち、実際の解析に利用する「左辺記号と右辺の記号数」だけを保持するものが多いため、提案法の実装は極めて容易である。実際、MSLR パーザ(東京工業大学・田中・徳永研究室)では、LR 表読み込み部分の若干の修正で動作可能となった。また、LR 法を利用した音声認識システム niNja(伊藤, 速水, 田中 1992)では、全く修正の必要はなくそのまま動作可能であった。

3.2 実験

提案法の効果を調べるため、3 種類の文法から従来法と提案法 1 で LR 表を作成し、表のサイズを比較した。比較には、LR 表(および GOTO グラフの)状態数、表中の空欄でないセルの数を表すエントリ数、を用いた。

文法「道案内 1」「道案内 2」は、道案内対話(Itou, Akiba, Hasegawa, Hayamizu and Tanaka 1999)に現れるユーザの発話をモデル化した文法で、音声認識用に設計された(Akiba and Itou 2001)。語彙サイズはどちらも約 380、規則数はそれぞれ 616, 1302 である。「道案内 2」は、「道案内 1」に比べて、意味的に整合性のある文だけを受理するように、より強い制約を加えた文法である。文法「旅行会話」は、ATR 研究用自然発話音声データベース(Morimoto 1994)旅行会話タスクの発話を受理するように記述した文法である(今井, 田中 1999)。語彙サイズは 2839、規則数 3971 と、文法 1,2 に比べて大規模な文法である。また、自然言語処理用途に開発されており、入力文に対して構文的に可能な数多くの構文木を割り当てる。

提案法 1 で LR 表を作成し、その性質を調べた。結果を表 1 に示す。すべての文法について、LR 表圧縮の効果が得られていることがわかる。LR 表のサイズは、「道案内 1」「道案内 2」に関しては約 60%前後に、「旅行会話」に関しては約 1/4 まで圧縮することができた。文法の規則数

表 1 LR 表圧縮の効果

	文法	道案内 1		道案内 2		旅行会話
	LR 表タイプ	SLR	CLR	SLR	CLR	SLR
	規則数	616		1,302		3,971
従来法	状態数	703	876	2,079	3,530	4,017
	エントリ数	13,689	15,164	21,092	29,698	10,054,708
提案法 1 (左方抽象化 LR 項)	状態数	444	568	757	1,815	935
	(圧縮率)	(63.2 %)	(64.8 %)	(36.4 %)	(51.4 %)	(23.3 %)
	エントリ数	8,031	9,155	12,282	18,669	2,250,317
	(圧縮率)	(58.7 %)	(60.4 %)	(58.2 %)	(62.9 %)	(22.4 %)
提案法 2 (可変長 LR 項)	状態数	404	518	669	1,600	842
	(圧縮率)	(57.5 %)	(59.1 %)	(32.1 %)	(45.3 %)	(21.0 %)
	エントリ数	7,111	8,132	10,435	16,345	1,978,521
	(圧縮率)	(51.9 %)	(53.6 %)	(49.5 %)	(55.0 %)	(19.7 %)

が大きいほど圧縮率が大きくなる傾向が見られるが、これは規則数が増えることで、2.4節で述べた性質を満たす規則対の候補が増えることに起因すると考えられる。また、作成した LR 表を用いてテキスト解析の実験を行ったが、従来の LR 表を使った場合と全く同じ結果が得られ、解析時間にも差は認められなかった。

4 更なる圧縮のための改良手法

2.4節で述べたように、(1) 規則左辺の記号が同じ、(2) 右辺の接尾部分が共通、(3) 右辺の記号数が同じ、の 3 条件を満たす規則の組が文法中に多く現れるほど、提案法の効果は大きい。このうち、(3) 右辺記号数の条件は、改善の余地がある。本節では、この条件を克服するための拡張方法について述べる。

4.1 可変長 LR 項

ドットの左側に記号数が必要なのは、reduce 動作時にスタックからポップする要素数を記録するためである。この要素数は、LR 表に reduce 動作の引数として、静的に記述される。しかし、このポップ要素数は解析時に動的に求めることもできる。そこで、図 4 のように、この情報をスタックに保持することを考える。すなわち、reduce 動作の際、スタックに保持された区切り位置までポップするような構文解析アルゴリズムを考える。

このような区切りをスタックに入れるような LR 表はどのように生成すればよいだろうか。区切りは、規則右辺の最初の記号をスタックに積む時に挿入すればよい。すなわち、LR 項

$$[A \rightarrow \cdot B\alpha]$$

から GOTO 手続きによって、LR 項

図 4 区切り付きスタック

$[A \rightarrow B \cdot \alpha]$

を生成する際に、スタックに挿入すればよい。ドットが最左にある場合の、次の状態への遷移(すなわち、shift 動作や goto)の場合に、スタックに区切りを挿入する。注意すべきなのは、ある状態(クロージャ)に、ドットのすぐ右の記号が同じで、ドットが最左のものと規則途中にあるものの、2つ以上の LR 項が含まれている場合があることである。すなわち、次のような2つの LR 項が、同じクロージャ I に含まれている可能性がある。

$[A \rightarrow \cdot B\alpha]$, $[A \rightarrow \beta \cdot B\gamma]$

この場合、前者(ドットが最左のもの)と後者(ドットが途中のもの)からの、記号 B による遷移($GOTO(I, B)$)を別に扱うことを考える。例えば、前者を記号 B_{start} による遷移、後者を記号 B_{cont} による遷移とし、 $GOTO(I, B_{\text{start}})$ と $GOTO(I, B_{\text{cont}})$ を別々に計算する。

このように変更した GOTO グラフでは、もはや LR 項にドット左の記号数は必要ない。ただし、ドットが最左であるか、途中であるかの区別は必要となる。ドットが途中にある場合の LR 項を、次のようなドット左方可変長の LR 項で表すことにする。

$[A \rightarrow * \cdot \gamma]$

記号 $*$ は、長さ 1 以上の記号列があることを表す。このような LR 項を用いることで、ドット左方の記号列に関する情報がさらに抽象化され、GOTO グラフ作成時の状態数がさらに減少することが期待できる⁴。LR 表生成アルゴリズムに必要な変更点は、以下の通りである。(この変更を行った手法を提案法 2 とする。)

[GOTO 手続き]

- I 中の、ドットのすぐ右が B である LR 項 $[A \rightarrow \cdot B\gamma]$ すべてに対し、LR 項 $[A \rightarrow * \cdot \gamma]$ を求め、そのクロージャを $GOTO(I, B_{\text{start}})$ の返回值とする。また、LR 項 $[A \rightarrow * \cdot B\gamma]$ に対し、LR 項 $[A \rightarrow * \cdot \gamma]$ を求め、そのクロージャを $GOTO(I, B_{\text{cont}})$ の返回值とする。

⁴ ただし、GOTO 手続きの記号を場合分けしたことによって、状態数が増加するため、全体としてかならずしも状態数減少になるとは限らない。

[LR 項集合の集合]

- (2) C の各 LR 項集合 I , G' のある記号 A , $s \in \{\text{start}, \text{cont}\}$ について, $\text{GOTO}(I, A_s)$ を計算し, LR 項集合 I' を求め, C に加える. この手続きを, C に新たな LR 項集合が加えられなくなるまで繰り返す.

[LR 表の作成]

- 終端記号 a と, $s \in \{\text{start}, \text{cont}\}$ について, $\text{GOTO}(I_i, a_s) = I_j$ ならば, $\text{action}(i, a)$ に $\text{shift } s, j$ を加える.
- 非終端記号 A , $s \in \{\text{start}, \text{cont}\}$ について, $\text{GOTO}(I_i, A_s) = I_j$ ならば, $\text{goto}(i, A)$ に s, j を加える.

構文解析アルゴリズムに必要な変更は, 以下の 3 点である.

- shift 動作と goto 時に, LR 表の記述に従い, スタックに区切り記号を挿入する.
- shift 動作と goto 時に, $\text{shift}/\text{shift}$ コンフリクト, goto/goto コンフリクトを扱えるようにする. 実装は, GLR 法による $\text{shift}/\text{reduce}$ コンフリクト, $\text{shift}/\text{reduce}$ コンフリクトの扱いと同様に, スタックを分岐させればよい.
- reduce 動作の際, 従来の固定数ポップ動作の代わりに, スタックの最初の区切り記号までポップする.

4.2 実験

3.2節で用いた 3 種類の文法から, 提案法 2 を用いて LR 表を作成した. 実験結果を表 1 に示す. LR 表のサイズをさらに 10% 程度縮小できることを確認した. また, 音声認識システム niNja(伊藤他 1992) に提案法 2 の解析アルゴリズムを実装し, 従来法と同じ解析結果が得られることを確認した.

5 関連研究**5.1 共通記号列のまとめ処理**

生成規則 ($A \rightarrow \alpha$) に対し右辺記号列 (α) 中のある位置にドット ‘ $\cdot$ ’ を付けたデータ構造 ($A \rightarrow \beta \cdot \gamma$, ただし $\alpha = \beta\gamma$) は, 項 (Item) と呼ばれ, 構文解析中に規則のどこまで解析が進んだかを表すために, 本稿で述べた LR 項の他, Earley 法 (Earley 1970) やチャート法 (Key 1980) など, 種々の構文解析アルゴリズムで共通に利用されている. 本稿で示した手法は, LR 法において Item 以降の解析がドットの左側 (右辺記号列の prefix, β) には依存しないことを利用し, ドットの右側の記号列 (右辺記号列の suffix, γ) が共通なものをまとめあげることによって, LR 表の圧縮を実現したと考えることができる. 同様に, Item のドット左右の記号列について複数の規則の間で共通する記号列をまとめて処理することによる, 解析の効率化手法が知られ

ている。

本稿の提案法のように、ドットの右側の記号列 (γ) が共通な Item をまとめて扱う手法が提案されている。文献 (Leermakers 1992) では、Earley 法において Item 以降の解析がドットの右側の記号列 (γ) のみに依存し、ドットの左側 (β) や生成規則左辺の記号 (A) には依存しないことを利用して、これらを重複処理しないことによる効率化手法が示されている。文献 (Moore 2000) では、同様の手法をチャート法に適用している。

逆に、ドットの左側の記号列 (β) が共通な Item をまとめて処理する手法としては、LR 法が挙げられる。LR 法では、共通な prefix を持つ複数の Item をまとめて解析の一状態とするように LR 表を作成することで、解析の効率化を実現している。文献 (Nederhof 1994) では、この考え方を進めて、共通の prefix をもつ規則をすべてまとめて処理する手法が示されている。また、共通した prefix を持つ 2 つ以上の規則を持たないように文法を変形することによって効率化を行なう手法も提案されている (Moore 2000)。

5.2 可変な規則長の扱い

本稿の提案法 2 では、ドット左側の記号数情報を捨象した可変長 LR 項の導入のため、reduce 動作時にスタックからポップする記号数を動的に求める必要があった。そのために、GOTO 手続きを規則の解析開始か途中かによって別々に計算する手法を示した。同様の考え方は、規則の右辺に記号の正規表現を許した拡張 CFG (正規右辺文法) を扱う LR 構文解析法として提案されている (Nederhof 1994; Purdom and Brown 1981)。正規右辺文法では、規則の右辺に合致する記号数を予め知ることができないので、解析時に動的に求める必要があるためである。

6 結論

解析に使用する LR 表の大きさが問題であった LR 構文解析法について、表作成に用いられる基本データ構造 (LR 項) の見直しを行うことにより、LR 表の状態数を減少させ、サイズを圧縮する手法について述べた。提案法を実際の文法に適用したところ、規則数 500~1500 程度の文法に対しては元のサイズの 60% 程度、規則数 4000 の文法に対しては 25% 程度に圧縮できることを確認した。提案法は、従来の LR 表作成アルゴリズム、解析アルゴリズムに大きく手を加えることなく実装可能であるとともに、解析効率に影響を与えることもない。また、提案法を拡張し圧縮率を改善する手法を検討した。アルゴリズムへの変更箇所は増加するが、実験結果ではさらに 10% 程度サイズを圧縮できることを確認した。本研究により、これまで解析表のサイズの問題で LR 法の適用が困難であった分野、例えば大規模な文法を用いた自然言語処理や音声認識、また計算資源 (記憶容量) に制限がある環境 (例えば、モバイル用途) での使用などにおいて、効率の良い LR 法を適用する機会が増えると考えられる。

謝辞

本稿の実験には, 東工大田中・徳永研究室で開発された MSLR パーザ (東京工業大学・田中・徳永研究室) および文法を使用した。また, 本手法に関して御討論いただいた田中・徳永研究室の皆様にご感謝する。

参考文献

- Aho, A. V., Sethi, R., and Ullman, J. D. (1986). *Compilers: Principles, Techniques, and Tools*. Addison Wesley.
- Akiba, T. and Itou, K. (2001). “A Structured Statistical Language Model conditioned by Arbitrarily Abstracted Grammatical Categories based on GLR Parsing.” In *Proceedings of 7th European Conference on Speech Communication and Technology (Eurospeech)*, pp. 705–708.
- Earley, J. (1970). “An Efficient Context-free Parsing Algorithm.” *Communications of the ACM*, **13** (2), pp. 94–102.
- 今井宏樹, 田中穂積 (1999). “音声認識を目指した確率 GLR 法を用いた言語モデルの構築.” 情報処理学会論文誌, **40** (4), pp. 1404–1412.
- Inui, K., Tokunaga, T., and Tanaka, H. (1998). “Probabilistic GLR Parsing: A New Formalization and its impact on parsing performance.” 自然言語処理, **5** (3), pp. 33–52.
- 伊藤克亘, 速水悟, 田中穂積 (1992). “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.” 電子情報通信学会論文誌, **J75-D-II** (6), pp. 1023–1030.
- Itou, K., Akiba, T., Hasegawa, O., Hayamizu, S., and Tanaka, K. (1999). “A Japanese Spontaneous Speech Corpus Collected using Automatic Inference Wizard of Oz System.” *The Journal of the Acoustical Society of Japan (E)*, **20** (3), pp. 207–214.
- Key, M. (1980). “Algorithm Schemata and Data Structures in Syntactic Processing.” Tech. rep., Xerox PARC, Palo Alto California.
- Leermakers, R. (1992). “A recursive ascent Earley parser.” *Information Processing Letters*, **41**, pp. 87–91.
- Luk, P. C., Meng, H., and Weng, F. (2000). “Grammar Partitioning and Parser Composition for Natural Language Understanding.” In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, Vol. 3, pp. 486–489.
- Moore, R. C. (2000). “Improved Left-Corner Chart Parsing for Large Context-Free Grammars.” In *Proceedings of the Sixth International Workshop on Parsing Technologies*, pp. 171–182.

- Morimoto, T. (1994). "A Speech and Language Database for Speech Translation Research." In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pp. 1791-1794.
- 永井明人, 鷹見淳一, 嵯峨山茂樹, ハラルドシンガー (1994). "隠れマルコフ網と一般化 LR 構文解析を統合した連続音声認識." 電子情報通信学会論文誌, **J77-D-II** (1), pp. 9-19.
- Nederhof, M.-J. (1994). "An Optimal Tabular Parsing Algorithm." In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 117-124.
- Purdom, P. W. and Brown, C. A. (1981). "Parsing Extended LR(k) Grammars." *Acta Informatica*, **15**, pp. 115-127.
- 田中穂積 (1989). 自然言語解析の基礎. 産業図書.
- Tanaka, H., Tokunaga, T., and Aizawa, M. (1995). "Integration of Morphological and Syntactic Analysis Based on LR Parsing Algorithm." 自然言語処理, **2** (2), pp. 59-74.
- 田中穂積, 竹澤寿幸, 衛藤純司 (1997). "MSLR 法を考慮した音声認識用日本語文法 - LR 表工学 (3) -." 97-SLP-15, 情報処理学会., pp. 145-150.
- 東京工業大学・田中・徳永研究室. "Morphological and Syntactic LR Parser - MSLR parser -." <http://www.cl.cs.titech.ac.jp/>.

付録

A LR 表作成手順

LR 表は, 文法が与えられた時点であらかじめ計算できる構文解析過程を表したものである. LR 構文解析法におけるパーザは, この LR 表と, スタック, 先読み語, を参照し, 次の動作を決定する. LR 表には, SLR (Simple LR), LALR (Look-Ahead LR), CLR (Canonical LR), などの幾つかのバリエーションがあり, これらは決定性解析で扱える言語の範囲に違いがある. 以下, 説明のため LR 表の作成法を簡単に述べる. 詳細は文献 (Aho et al. 1986; 田中 1989) 等を参照されたい.

LR 項は, LR 表作成に使用するデータ構造であり, 構文解析中のある状態を表すものである. SLR 表作成には LR(0) 項を, CLR 表作成には LR(1) 項を用いる⁵. LR(0) 項とは, ある生成規則の右辺の任意の位置にドットを付けたデータ構造である. ドットは, 規則中のどの部分まで解析が進んだかを表す. 下記の LR(0) 項は, 生成規則 $A \rightarrow \alpha$ において, α が $\beta\gamma$ と書けるとき, β まで解析が終わった状態を表している.

$$[A \rightarrow \beta \cdot \gamma]$$

LR(1) 項とは, ドット付き生成規則と, 1 つの先読み語 (終端記号) からなるデータ構造であ

⁵ LALR 表作成には, LR(0) 項, LR(1) 項それぞれから作成する手法が知られている.

る。下記の LR(1) 項は、同様に β まで解析が終わった状態を表し、さらに 1 つの先読み語 a を持つ。 a は、直感的には、この生成規則適用の直後に現れる先読み語を表す。

$$[A \rightarrow \beta \cdot \gamma, a]$$

以降の説明では、LR(0) 項を用いて SLR 表を作成する場合について考えるが、LR(1) 項から CLR 表を作成する場合も、先読み語の計算が付け加わるだけで、ほぼ同様に求めることができる。

[クローージャ]

クローージャ(closure)とは、LR 項の集合であり、ある LR 項集合 I が与えられると、以下の手続きでクローージャ $\text{closure}(I)$ を求めることができる。

- (1) I の全要素を $\text{closure}(I)$ に加える。
- (2) $\text{closure}(I)$ に LR 項 $[B \rightarrow \beta \cdot A\gamma]$ があれば、左辺が A の全ての生成規則 $A \rightarrow \alpha$ について、LR 項 $[A \rightarrow \cdot \alpha]$ を加える。この手続きを新たな LR 項が加えられなくなるまで繰り返す。

クローージャとは、直感的には、ある LR 項の表す解析状態と同時に現れ得る全ての LR 項を、あらかじめトップダウンに展開して求めたものである。

[GOTO 手続き]

GOTO 手続き $\text{GOTO}(I, B)$ は、アイテム集合 I と記号 B から、新しいアイテム集合を次のように求める。

- I 中の、ドットのすぐ右が B である全ての LR 項 $[A \rightarrow \beta \cdot B\gamma]$ に対し、ドットを一つ右に移動した LR 項 $[A \rightarrow \beta B \cdot \gamma]$ を求め、そのクローージャを返す。

直感的には、構文解析のある状態 I で、 B が得られたときの、次の状態を求めている。

[LR 項集合の集合]

上記クローージャ、GOTO 手続きを用いて、ある拡大文法⁶ G' から、LR 項集合の集合 C を次の手順で求める。

- (1) 初期の LR 項集合を $\text{closure}([SS \rightarrow \cdot S])$ とし、 C に加える。
- (2) C の各 LR 項集合 I 、 G' のある記号 A に対して、 $\text{GOTO}(I, A)$ を計算し、LR 項集合 I' を求め、 C に加える。この手続きを、 C に新たな LR 項集合が加えられなくなるまで繰り返す。

[LR 表の作成]

「LR 項集合の集合」 C の各状態 (すなわち LR 項集合) は、LR 表の 1 状態に対応する。LR 表は、状態 i と終端記号 a からパーザの動作を決める表 $\text{action}(i, a)$ と、 i と非終端記号 A から

⁶ 開始記号を S とする文法 G に対して、新しい開始記号 SS と生成規則 $SS \rightarrow S$ を追加して得られる文法を、 G の拡大文法という。

状態 j を決める表 $goto(i, A)$ から成る. C の各 LR 項集合 I_i について, 次の手順で LR 表を作成することができる.

- I_i に LR 項 $[A \rightarrow \alpha \cdot]$ が存在するならば, 終端記号 $a \in \text{follow}(A)$ について, $\text{action}(i, a)$ に $\text{reduce } A \rightarrow \alpha$ を加える. $\text{follow}(A)$ は, ある記号の次に現れ得る終端記号を計算する手続き.
- I_i に LR 項 $[SS \rightarrow S \cdot]$ が存在するならば, $\text{action}(i, \$)$ に accept を加える.
- 終端記号 a について, $\text{GOTO}(I_i, a) = I_j$ ならば, $\text{action}(i, a)$ に $\text{shift } j$ を加える.
- 非終端記号 A について, $\text{GOTO}(I_i, A) = I_j$ ならば, $\text{goto}(i, A)$ に j を加える.

LR 構文解析法

LR 構文解析法は, 解析スタックのトップに積まれている LR 表状態, 次の入力語 (先読み語) から, action 表を参照して, 次のように解析を進める.

- $\text{shift } i$: 先読み語と状態 i をスタックにプッシュする. 先読み語を一つ読み進める.
- $\text{reduce } A \rightarrow \alpha$: スタックから $2|\alpha|$ 個ポップし, その時のスタックトップの状態 j を用い, スタックに非終端記号 A と $\text{goto}(j, A)$ をプッシュする.
- accept : 入力を受理

B MSLR 法への適用

MSLR 法 (Tanaka et al. 1995) は, 文脈自由文法から生成した LR 表に, 形態素の接続可能性を表した接続表の制約を組み込む手法である. 終端記号 a と b の接続可能性を使って, LR 表から次のような動作を削除することで, 接続表を組み込むことができる.

動作 Act の直前に実行する動作が shift で, その shift の先読み語が a , Act の先読み語が b であるとき, a と b が接続不可なら Act を削除する.

この手法を可能にするためには, shift 直後の状態が, その先読み語 (a) 毎に分節されていなければならない. しかし, 提案法の LR 表では, shift 直後の状態では, 直前の先読み語 (a) の情報は抽象化されてしまい, その結果, 先読み語 (a) 毎に分節された状態にはならないため, MSLR 法をそのまま導入することはできない.

この問題を解決するためには, 提案手法においてドットのすぐ左の終端記号を抽象化しないようにすればよい. すなわち, 次のような LR 項を導入する.

$[A \rightarrow |\beta|a \cdot \gamma]$ (a が終端記号の場合)

$[A \rightarrow |\beta A| \cdot \gamma]$ (A が非終端記号の場合)

このような LR 項から LR 表を作成すると, 終端記号の直後の shift 動作で遷移する状態については, 終端記号毎に分節された状態となり, MSLR 法の適用が可能となる.

略歴

秋葉 友良: 1990 年東京工業大学理学部情報科学科卒業。1995 年同大学院総合理工研究科システム科学専攻博士課程了。同年，電子技術総合研究所入所。2001 年，産業技術総合研究所。現在，情報処理研究部門主任研究員。博士（工学）。自然言語処理，音声認識，音声対話の研究に従事。

伊藤 克亘: 博士（工学）。1993 年，電子技術総合研究所入所。2003 年，名古屋大学大学院情報科学研究科助教授。現在に至る。音声を主とした自然言語全般に興味を持つ。

(2002 年 11 月 29 日 受付)

(2003 年 3 月 18 日 再受付)

(2003 年 8 月 12 日 採録)