

## 先祖表を用いた一般化LRパースアルゴリズムの性能評価

山田 耕一、K.G.Suresh、帖左 正裕、沼崎 浩明、田中 穂積  
東京工業大学工学部

最近Kippsは、グラフ構造化スタックの上の同一パスを繰り返したることを避けるために、先祖節点を記憶する先祖表を用い、一般のCFGに対する認識時間が文の長さ $n$ に対して $n^3$ となる一般化LR認識アルゴリズムを開発している[Kipps 91]。Kippsのアルゴリズムは、与えられた文の文法的妥当性を知るための認識アルゴリズムに過ぎず、統語解析木(構文構造)を得ることが出来ないので、実用上問題があるとされている[Schabes 91]。本論文では、Kippsのアルゴリズムをベースにして、スタックトップにある節点の先祖表から統語解析木を作成する情報を抽出するパーズングアルゴリズム[Tanaka 92]をインプリメントし、それを用いた評価実験結果を示す。富田法と比べて、我々のアルゴリズムは高速なパーズングが可能であることを示す。

Implementation and Evaluation of Generalized LR Parsing  
Algorithms Using Ancestor TablesKouichi YAMADA, K. G. Suresh, Masahiro CHOSA,  
Hiroaki NUMAZAKI and Hozimi TANAKADepartment of Computer Science  
Tokyo Institute of Technology  
2-12-1 Oookayama Meguro-ku  
Tokyo 152, Japan

We have implemented a new generalized LR parsing algorithm which makes use of a set of ancestor tables that is introduced by Kipps [Kipps 91]. As Kipps's recognition algorithm does not give us a way to extract any parsing result, his algorithm is not considered as a practical parsing algorithm [Schabes 91]. In this paper we will show it is possible to extract every parsing trees from a set of ancestor tables. While the time complexity of Tomota's parsing algorithm can exceed  $O(n^3)$  for some CFG's, the time and space complexity of our parsing algorithm using ancestor tables is in order of  $n^3$  and  $n^2$  for any CFG, respectively, since our algorithm is based on the Kipps's recognition algorithm. The experiment suggest our parsing algorithm is more efficient than Tomita's algorithm and seems to be very promising.

## 1 はじめに

最近 Kipps は、富田法のアルゴリズムに若干の改良を加えて、一般の CFG に対する認識時間が文の長さ  $n$  に対して  $O(n^3)$  となる一般化 LR 認識アルゴリズムを開発している [Kipps 91]。しかし、このアルゴリズムは、与えられた文の文法的妥当性を知るための認識アルゴリズムに過ぎず、実用上問題があるとされてきた [Schabes 91]。本論文では Kipps のアルゴリズムをベースにしたパーズアルゴリズム [Tanaka 92] のインプリメントおよび実験結果について述べる。我々の提案するアルゴリズムは、Kipps による認識アルゴリズムをベースにしているため、Kipps のアルゴリズム同様に、パーズング時間のオーダーは  $O(n^3)$  となる。

2 章では、本アルゴリズムの基本的なアイデアを説明する。3 章では、実験結果を示し、本アルゴリズムの高速性を実証し、その考察をする。4 章では、まとめと今後の研究課題について述べる。

## 2 AGLR アルゴリズムの基本的アイデア

富田法の認識アルゴリズムが、一般の CFG に対して  $n^3$  を越える認識時間を要する原因は、葉から  $q$  だけ離れた高々  $i$  個の先祖節点を求めるために、最悪の場合  $i^q$  のオーダーの計算時間が必要になることによる。これは、先祖節点を求める過程で、グラフ構造化スタック中の同一のパスを何回もたどることによる。

図.1 に示すグラフ構造化スタックでは、レデュース操作により例えば葉  $v_6$  から 3 だけはなれた (ただ一つの) 先祖節点  $v_1$  を取り出すために、このスタックを 4 回たどる必要がある。

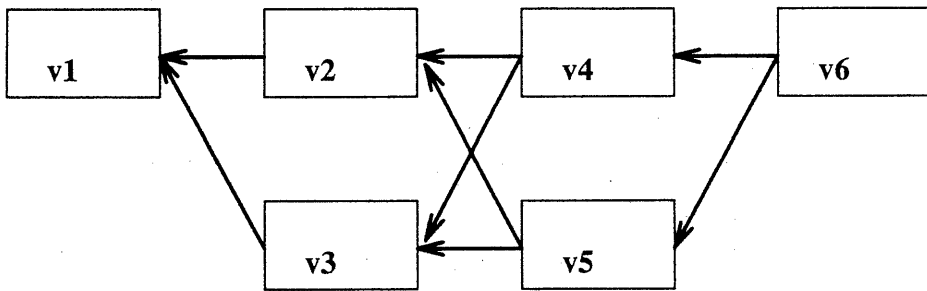


図.1

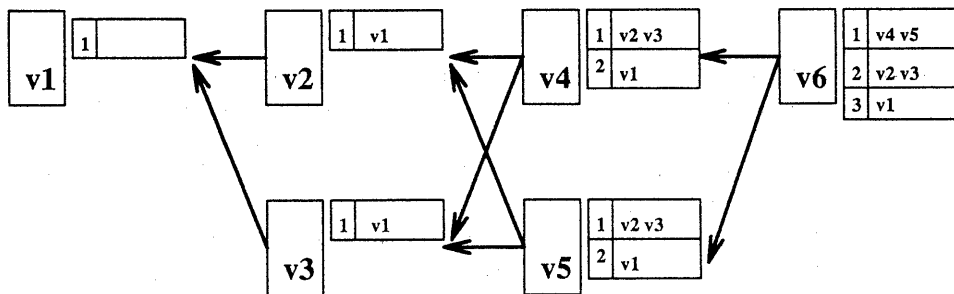


図.2

同一パスを何度もたどることを防ぐことが出来れば、先祖節点を求める時間を短縮することが出来る。そのため Kipps は、節点のデータ構造を三つ組  $\langle i, s, A \rangle$  とした。ここで、 $i$  はシフトされた語の位置番号、 $s$  は状態、 $A$  は先祖表 (ancestors table) である。先祖表  $A$  は、図.2 に示すように、その節点からの距離と、その距離だけ離れた所に位置する節点のリストの対応表である。

このような先祖表を各節点が持てば、シフト操作とレデュース操作時に新たな葉(節点)を作るとき、その葉の先祖表は、葉の親節点の持つ先祖表を利用して漸進的に埋めることが出来る。こうしてレデュース操作時に先祖節点の集合を求めることは、先祖表の検索に置き換えられ、検索に要する時間は一定時間になる。Kippsによれば、先祖表を埋めるのに要する時間は $i^2$ のオーダーである。以上のことから、長さ $n$ の文に対して、 $i=1, 2, \dots, n$ までの総和をとって、Kippsの認識アルゴリズムの計算時間のオーダーが $n^3$ となることがわかる。

AGLR(Ancestor Generalized LR)パーザは、このスタックトップの節点の先祖表から構文解析木を構成するのに必要な情報をとり出すことによって、パーズングを進めていくアルゴリズムである。AGLRは、大略スタックトップの節点の先祖表にレデュース規則を付加したものを記憶していく。AGLRパーザのアルゴリズムの詳細については、[Tanaka 92]を参照されたい。

### 3 実験結果

AGLRパーザの評価を行なうために、いくつかの実験を行ない、富田法との比較を行なった。その結果について述べる。

#### 3.1 実験環境

アルゴリズムのインプリメントは、AGLR、富田法共にC言語を用い、NEWS3860上で、実験を行なった。NEWS3860は約20MIPSの性能を持つ。

実行毎に時間にばらつきが出るので、10回測定してその平均をとった。しかし、このばらつきはかなり大きく、また、測定できる時間の最小単位が10msecであったので、短い時間の部分の信頼性は低い。

文法は3つ使用した。一つは、非常に多くの曖昧性のあるものを採用した。

文法 A

```
s → s, s, s, s.  
s → s, s.  
s → a.
```

この文法は、 $O(n^5)$ の構文的曖昧性を持っている。もうひとつは、実際の文法に対しても、有効であることを示すため、[Tomita 86]の用いた文法のうち、文法 III と文法 IV を使用した。文法 III は 224 のルールが、文法 IV は 394 のルールからなる。後者は、東工大の高倉が開発した文法である [Takakura 84]。

入力文は、上記した文法 A に対して、 $a a a a \dots$  を、文法 III と IV に対しては、同じく [Tomita 86]にある、SENTENCE SET I と II をパーズングの対象文として使用した。SENTENCE SET II は pp-アタッチメント文である。

SENTENCE SET I の具体的な例を以下に幾つかあげる。

- the assembly language provides a means for writing a program without having to be concerned with actual memory addresses
- it allows the use of symbolic codes to represent the instructions
- labels can be assigned to a particular instruction step in a source program to identify that step as an entry point for use in subsequent instructions

SENTENCE SET II は、次の形をした文である。

noun verb det noun (prep det noun) $n-1$

実験では、この構造を持つ次の文章を入力に使用した。本論文においては、 $1 \leq n \leq 16$  の範囲で実験を行なった。

I saw a man in the park with a telescope ...

### 3.2 実行速度の比較

この節では、富田法とパーズングの実行速度の比較を行ない、AGLR の高速性を実証する。

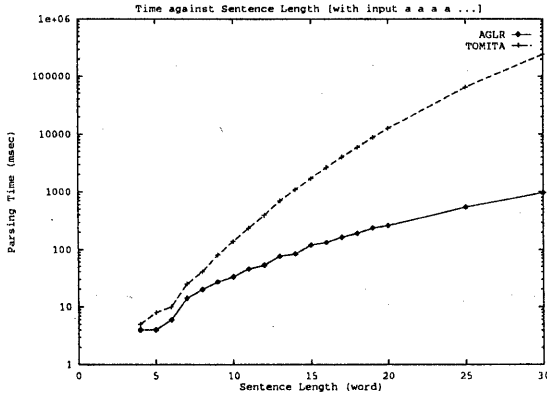


図.3

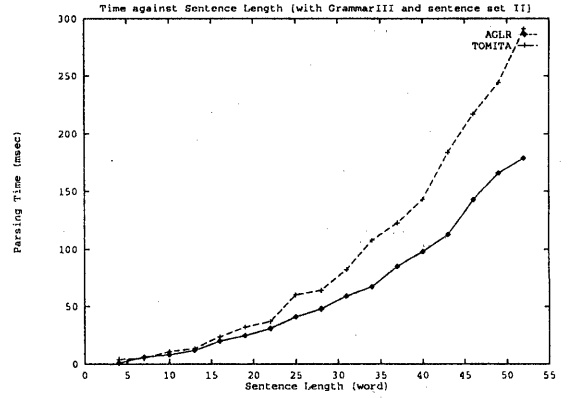


図.4

図.3 は文法 A に、a a a a ...を入力した結果である。横軸は a の個数である。非常に曖昧性の多い場合、(図.3 入力文の長さ 20-30)50 倍から 250 倍ほど、AGLR は富田法と比べて高速であることが分かる。図.4 は、文法 III に、SENTENCE SET II を入力したものである。横軸は入力文の長さである。

文の長さ とパーズング時間の関係をより詳しく見るため、富田法と AGLR のパーズング実行時間の比をとったものを図.5 と図.6 に示す。

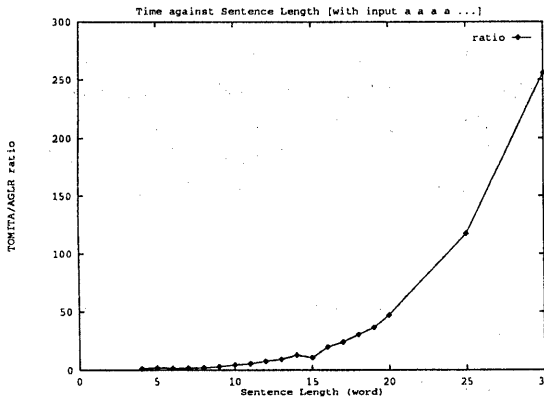


図.5

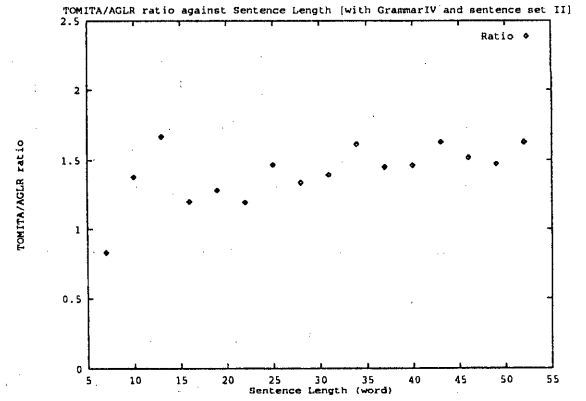


図.6

図.5 では顕著に、図.6 ではややグラフが右上がりであることが図から読みとれるように、文章の曖昧性が多くなればなるほど AGLR は富田法に比べて高速なパーズングが可能である。

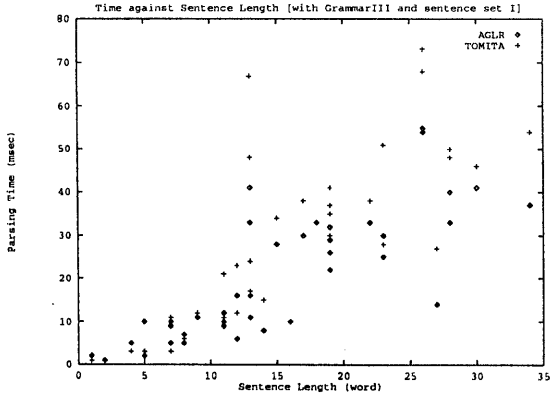


図.7

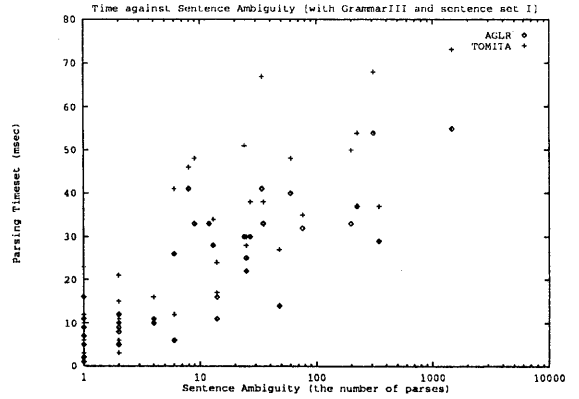


図.8

図.7と図.8は、文法IIIに対し、SENTENCE SET Iをパーズングした結果である。図.7は文の長さとのパーズング時間の関係、図.8は、パーズング結果の曖昧性の数に対するパーズング時間を示す。富田法とAGLRのパーズング速度の比較をするために、図.9、図.10に両者の比をとったものを示す。AGLRの方が富田法よりも高速なパーズングが可能であることが分かる。

次にそれぞれ比をとって調べてみる。

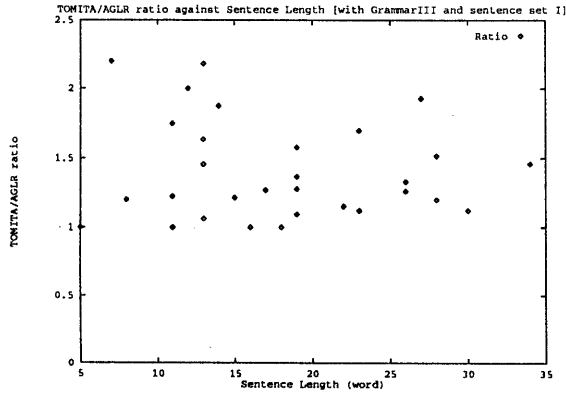


図.9

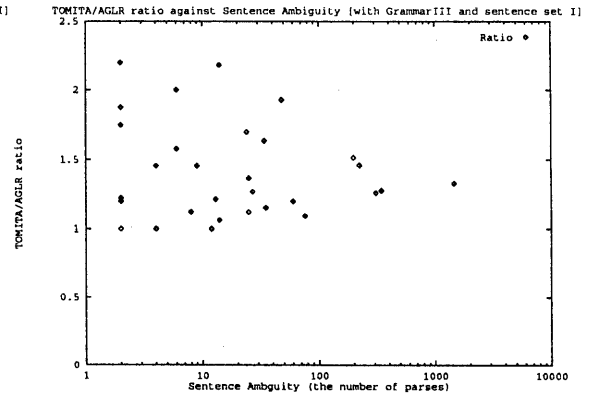


図.10

平均するとAGLRは富田法と比べて、おおよそ1.5倍ほど高速なパーズングが可能である。文法によるパーズング時間の違いを見るため、文法IVを使用した場合の実験結果を図.11、図.12に示す。

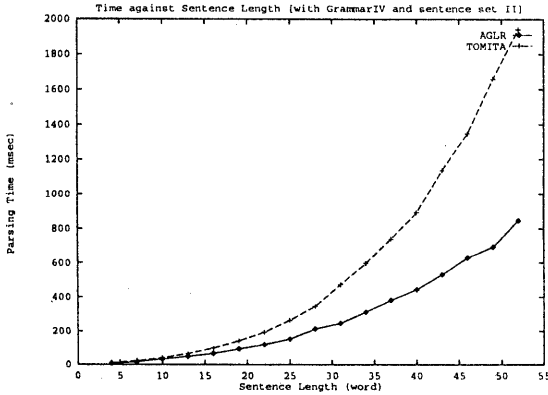


図.11

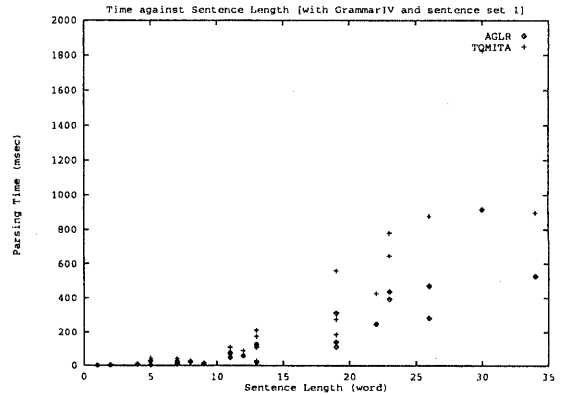


図.12

図.11は、SENTENCE SET IIを、図.12は、SENTENCE SET Iをそれぞれ入力した時の、文章の長さに対するパーズング時間のグラフである。図.11は図.4に、図.12は図.7に対応している。文法 IV は、文法 III よりも多くの曖昧性を生ずるため、文法 III よりもパーズングに時間がかかっている。

文法 IV に対して、AGLR と富田法のパーズング時間の比をとった結果を、次の図.13, 図.14 に示す。

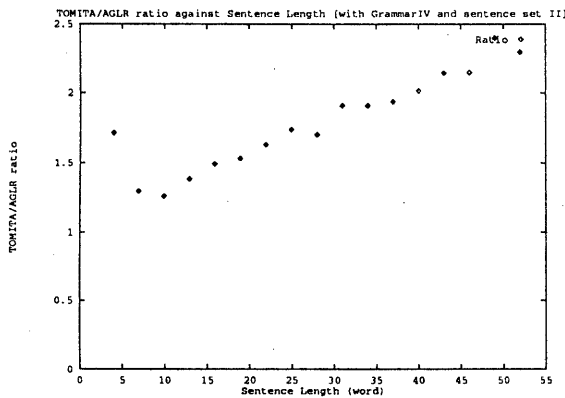


図.13

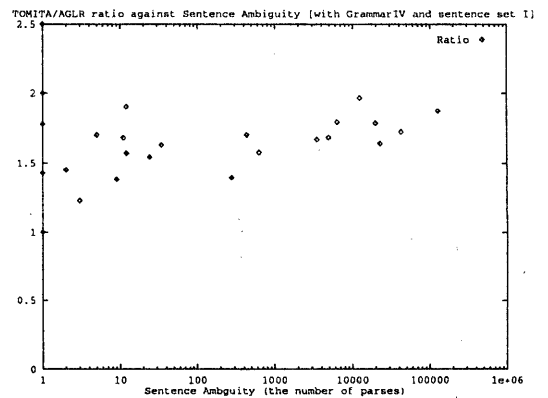


図.14

SENTENCE SET I, II 共に、文法 III より文法 IV の方が TOMITA/AGLR の比が大きくなっている。これは、文法 IV の生成規則の右辺が文法 III よりも長いことに起因していると思われる。

曖昧性が多くなればなるほど、TOMITA/AGLR の比が大きくなる傾向にあることが図.13、図.14 から読みとれる。図.13(SENTENCE SET II) のグラフの傾きは、図.6 のグラフよりも大きい。同じ入力文を使用した、図.9、図.10 と比較すれば、文法による速度の違いが良くわかるであろう。

最後に実際の実験データの一部を示す。

SENTENCE SET I								
文の長さ	文法 III				文法 IV			
	曖昧性 の数	富田法 実行時間 (msec)	AGLR 実行時間 (msec)	比	曖昧性 の数	富田法 実行時間 (msec)	AGLR 実行時間 (msec)	比
13	34	67	41	1.56	3475	212	127	1.67
19	346	37	29	1.26	6304	559	312	1.79
19	76	35	32	1.09	12321	277	141	1.96
22	35	38	33	1.15	42159	427	248	1.72
23	25	28	25	1.12	19773	780	437	1.78
26	1464	73	55	1.33	4923	476	283	1.68
26	309	68	54	1.26	127338	878	469	1.87

### 3.3 文法の生成規則の右辺の長ささと計算量

文法 III で文章をパーズングした場合の富田法と AGLR のパーズング時間の差は平均 1.5 倍程である。文法 IV を用いてパーズングを行なった場合は、文法 III の時に比べて富田法と AGLR の差が広がっている。文法 A を用いた場合の差は、歴然としている。この、文法による富田法/AGLR のパーズング時間の比の差の原因は何であろうか。

ここで、使用する文法がチョムスキー標準形なら、富田法のパーズング時間のオーダーも  $O(n^3)$  となることに注意したい。文法 III と文法 IV のルール中の、チョムスキー標準形の占める割合をはかった。文法 III においては、224 のルールのうち 169 が文法規則の右辺の長さが 2 以下の生成規則であり、その割合は、75.4% である。一方文法 IV では、394 のルールのうち文法規則の右辺の長さが 2 以下のルール数は 181 であり、全体に占める割合は、45.9% である。また、ルールの右辺の平均の長さは文法 III が 2.03、文法 IV が 2.75 であった。これからも明らかのように、文法 IV の方が長い生成規則を使用している。実験より、文法 IV の方が富田法/AGLR の速度比が大きいため、AGLR は、パーズング時に使用される生成規則が長いほど高速なパーズングが可能であることが分かる。

### 3.4 使用メモリの比較

この節では、AGLR パーザのメモリ効率について富田法と比較する。以下のグラフに現れる使用メモリ数には、LR テーブルが使用するメモリは含まれていない。即ち、グラフ構造化スタック (AGLR の先祖表を含む)、バックドフォレスト (富田法の場合) またはアイテム (AGLR の場合) などで使用するメモリの総量である。

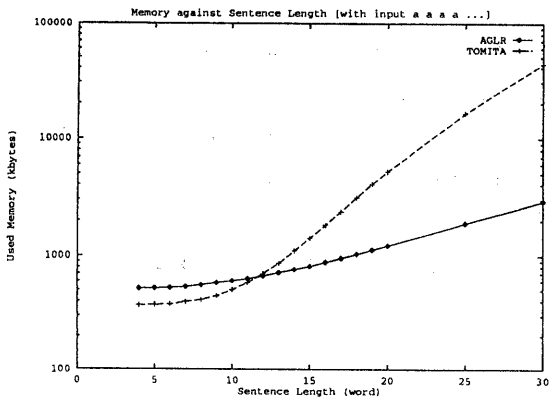


図.15

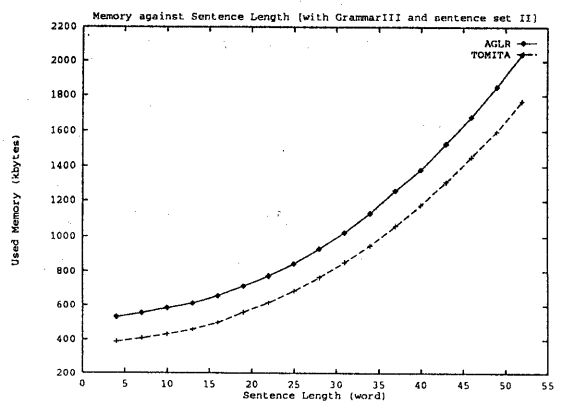


図.16

図.15 は、文法 A で a a a a … を入力した時、図.16 は文法 III を使用して、SENTENCE SET II(pp-アタッチメント) をパースした時の使用メモリである。

曖昧性が少ない領域では、AGLR の方が先祖表を使うため、メモリの使用量が多くなっている。しかし、図.15 に見るように、曖昧性が多くなると富田法では、使用するメモリの量が急激に増加している。これは、富田法のバックドフォレストよりも、AGLR のアイテムの方がメモリ効率が良いことを示唆している。図.16 は、先祖表の分だけ、AGLR の方がメモリ効率が悪く、それが改善されない。その理由については、現在検討中である。

## 4 おわりに

本論文では、先祖表を利用した一般化 LR パーザの基本的なアイデアと、インプリメントと実験結果を示した。このアルゴリズムは、Kipps の認識アルゴリズムに若干の修正を加えただけのものであり、パーズング時間のオーダが  $O(n^3)$  の極めて高速なパーズングが可能である。

今後の研究課題として、以下のものを挙げる事が出来る。

- 並列パーズングアルゴリズムの研究。
- 統語パーズング結果を抽出するための、並列アルゴリズムの研究。
- AGLR パーザをベースとした自然言語解析用のツールの開発。

## 参考文献

- [Aho 72] Aho, A. V. and Ulman, J. D.: *The Theory of Parsing, Translation and Compiling*, Prentice-Hall, New Jersey(1972).
- [Early 70] Earley, J.: *An Efficient Augmented-Context-Free Parsing Algorithm*, Comm. of ACM, 13, 1-2, pp.95-102(1970)
- [Kipps 91] Kipps, J. N.: *GLR Parsing in Time  $O(n^3)$* , in Tomita, M ed.:*Generalized LR Parsing*, Kluwer Academic Publishers, pp.43-59(1991).
- [Schabes 91] Schabes, Y.: *Polynomial Time and Space Shift-Reduce Parsing of Arbitrary Context-free Grammars*, Proc. of 29th ACL, pp.106-107(1991).
- [Johnson 91] Johnson, M.: *The Computational Complexity of GLR Parsing*, in Tomita, M ed.:*Generalized LR Parsing*, Kluwer Academic Publishers, pp. 35-41(1991).
- [Tomita 86] Tomita, M.: *Efficient Parsing for Natural Language*, Kluwer, Boston, Mass(1986).
- [Suresh 91] Suresh, K.G. and Tanaka, H.: *Implementation and Evaluation of Yet Another Generalized LR Parsing Algorithm*, Proc. of the Indian Computer Congress, Tata McGraw-Hill, pp.506-515(1991).
- [Tanaka 92] Tanaka, H, Suresh, K, G, and Yamada, K: 先祖表を利用した一般化 LR パーズングアルゴリズムのファミリー, 自然言語処理 90-5, pp. 33-40(1992).
- [Takakura 84] Takakura, K: *Prolog* による英語のボトムアップ構文解析, 東京工業大学卒業論文.