自 然 言 語 処 理 152－18
（２００２．１１．１３）

条件付到達可能性の拡張 TRIE 構造による表現について

笠 晃一　　横田 将生
福岡工業大学
〒 811-0295 福岡市東区和白東 3 丁目 30-1

あらまし　統語解析においては、最終結果に寄与しない部分解析木が大量に発生することが知られているが、このような部分解析木の発生を抑制するために従来より到達可能性が利用されてきた。しかしながら、到達可能性の効果はそれほど高くなく、場合によってはほとんど機能しないこともある。そこで、我々は到達可能性の自然な拡張である条件付到達可能性というものを提案した。これは、到達可能性を先読み情報によって補強するものである。ただし、条件部の記憶に多量のメモリが必要になるという欠点を持ち、これが不要な部分解析木の抑制による消費メモリの削減効果を阻害していた。今回、我々は条件部の記憶に TRIE 構造の拡張版を使用することにより、消費メモリ量を従来の 20 分の 1 以下にまで削減できたので報告する。

キーワード　統語解析、到達可能性、TRIE 構造

Representation of Conditional Reachability Using Extended TRIE Structure

Koichi RYU　Masao Yokota
Fukuoka Institute of Technology
3-30-1 Wajirohigashi, Higashi-ku, Fukuoka, 811-0295 Japan

Abstract　In parsing natural language sentences, useless partial analysis trees that do not contribute to the result are generated extensively when no heuristic information is available. Reachability has been used to control the generation of such useless trees. However, reachability does not have a strong restraint effect, and it sometimes hardly function. We have presented conditional reachability, which is a natural extension of reachability and reinforces reachability with look-ahead information. Conditional reachability nevertheless has a defect that it needs considerable memory for conditional parts. In this paper, we report that memory consumption can be reduced to 1/20 when an extended version of TRIE structure is used for conditional parts.

Keywords　Syntax analysis, Reachability, TRIE structure

１．はじめに

文脈自由文法を用いて自然言語文を統語解析する場合、最終結果に寄与しない不要な部分解析木の発生を抑制するために、到達可能性が使用されることが多い。これは、ある文法範疇から別の文法範疇への成長を予測するものである。しかしながら、我々の行なったいくつかの実験によれば、到達可能性を用いた場合、不要な部分解析木の抑制効果は必ずしも高くないようである。たとえば、100 個の日本語文を用いた実験の場合、到達可能性を使用しても、部分解析木の平均的な利用率は、何も使用しなかった場合に比べ５パーセント程度の上昇にとどまって

いた。ここに、利用率というのは、作成されたすべての部分解析木の個数に対する、最終的な解析結果に寄与する部分解析木の個数の割合のことである。我々は、到達可能性の予測の精度を上げるために先読み情報を用いることにした。すなわち、到達可能性によって、ある文法範疇 X から別の文法範疇 Y が予測されても、文法範疇 X の後方にキーとなる語彙範疇が出現しない限りその予測を無効とするのである。我々は、このような補強を施した到達可能性を条件付到達可能性 [1] と呼んでいる。上記の実験の場合、条件付到達可能性を使用すると、部分解析木の平均的な利用率は、何も使用しなかった場合に比べ 17.2 パーセントから 57.4 パーセントへと大きく上昇した。部分解析木の利用率が上昇するということは、より少ない作業メモリで解析が行なえるということを意味している。しかしながら、従来の条件付到達可能性の場合、条件部の記憶に多量のメモリが必要になるという欠点があり、これが作業メモリの削減効果を阻害していた。そこで、今回、条件部の記述を従来のリスト構造によるものから TRIE 構造の拡張版によるものへと変更してみた。その結果、条件部の記憶に必要なメモリを大幅に減少させることが可能になり、また、解析速度の向上も見られた。

2．条件付到達可能性

2．1　条件付到達可能性の定義

まず、文脈自由文法〈V, T, P, S〉に対する若干の制限を行なう。ただし、V は非終端記号の有限集合、T は終端記号の有限集合である。本稿では、非終端記号のことを文法範疇と呼ぶ場合もある。また、P は書換え規則の有限集合、S は開始記号である。この文脈自由文法の書換え規則は、次のいずれかの形をとるものとする。

［R1］語彙規則：$A \rightarrow a$　（$A \in V, a \in T$）
［R2］文法規則：$A \rightarrow \alpha$　（$A \in V, \alpha \in V^{+}$）

書換え規則に対するこのような制限は、文脈自由文法の能力になんら影響を与えないことが保証されている [2]。また、語彙規則の左辺に現れる非終端記号を特に語彙範疇と呼ぶことにし、すべての語彙範疇の集合を L で表すことにする。

そこで、文脈自由文法〈V, T, P, S〉に対する上昇型のアルゴリズムを用いて、下記のような単語列の統語解析を行なうことを考える。

$$w_1\, w_2 \ldots w_K \quad (w_i \in T) \tag{1}$$

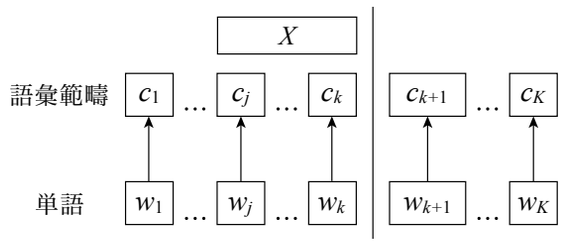


図 1　文法範疇 X を構成する語彙範疇

この単語列に対して語彙規則を適用し、次のような語彙範疇列が得られるものとする。

$$c_1\, c_2 \ldots c_K \quad (c_i \in L) \tag{2}$$

さらに、この語彙範疇の一部 $c_j \sim c_k$ を用いて、文法範疇 X が完成されていたとしよう。この様子を図 1 に示す。以下では、この図を用いて、条件付到達可能性を定義する。

文法範疇 X から文法範疇 Y に到達可能であることを $X \Rightarrow Y$ と書くことにする。すると、条件付到達可能性は、一般に次のように記述される。

$$X \Rightarrow Y \quad H_1 \,|\, H_2 \,|\, \ldots \,|\, H_N \tag{3}$$

ここに、$H_1 \sim H_N$ は語彙範疇のリストであり、文法規則から求められる範疇核というものを基にして作成される。記法 (3) の直観的な意味は、$H_1 \sim H_N$ のうちの少なくとも一つのリストの全要素が記された順に $c_{k+1} \sim c_K$ に出現するならば、文法範疇 X から文法範疇 Y に到達可能であるということである。なお、記法 (3) において、到達可能性に付加された部分のことを特に条件部と呼ぶことにする。

条件付到達可能性をより厳密に定義するために、部分リストというものを定義しよう。

定義　$A = [a_1, a_2, \ldots, a_s]$、$B = [b_1, b_2, \ldots, b_t]$ のとき、以下の条件を満たす $j_1, j_2, \ldots, j_t$ が存在するならば、B は A の部分リストである。

$$1 \leqq j_1 < j_2 < \ldots < j_t \leqq s \quad \text{かつ} \quad b_k = a_{j_k} \quad (k = 1, 2, \ldots, t)$$

さらに、図 1 において文法範疇 X の後方にある語彙範疇 $c_{k+1} \sim c_K$ を用いて次のようなリスト R を作成する。

$$R = [c_{k+1}, c_{k+2}, \ldots, c_K] \tag{4}$$

すると、記法 (3) のより正確な意味は、$H_1 \sim H_N$ の

ap → a	np → s np	d → " 実に "
ap → d ap	s → vp	m → " 細やかな "
mp → m	s → ppa s	p → " は "
mp → d mp	s → ap s	p → " を "
ppa → np p	vp → v b	v → " 持っ "
ppb → np no	vp → vp b	b → " ている "
np → n		a → " しっかり "
np → mp np	n → " 日本人 "	no → " の "
np → ppb np	n → " 神経 "	

図２　日本語書換え規則の例

a ⇒ ap	[]
a ⇒ s	[v,b]∣[b,v]
a ⇒ np	[v,b,n]∣[b,v,n]
np ⇒ ppa	[p]
np ⇒ ppb	[no]
np ⇒ s	[p,v,b]∣[p,b,v]
ppa ⇒ s	[v,b]∣[b,v]
ppa ⇒ np	[v,b,n]∣[b,v,n]
ppa ⇒ ppb	[v,b,n,no]∣[b,v,n,no]

図３　図 1 の書き換え規則に対する条件付到達可能性（部分）

うちの少なくとも一つがリスト R の部分リストであるならば、文法範疇 X から文法範疇 Y に到達可能であるということになる。

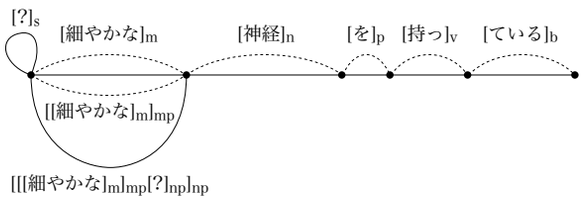
２．２　条件付到達可能性の例

条件付到達可能性は範疇核というものを基にして作成されるということを前節で述べたが、範疇核とは主辞駆動句構造文法（HPSG）[3] の主辞に類似した概念である。直観的に述べると、範疇核とはある範疇か必ず含んでいる語彙範疇のことであり、たとえば、通常は名詞句の範疇核は名詞であり、動詞句の範疇核は動詞である。範疇核は、文脈自由文法の書き換え規則から連立集合方程式を作成し、これを逐次近似法を用いて解くことによって得られるが、詳細は参考文献 [1] を参照されたい。

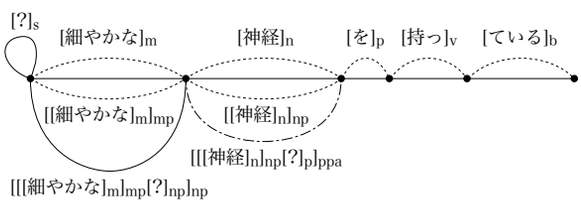
さて、ここで、図２に示すような日本語の書換え規則を考えてみよう。この書換え規則から、図３のような条件付到達可能性（部分）が得られる。一行目に空リストが出現しているが、これは無条件で範疇aから範疇apに到達可能であることを表している。図 4 の条件付到達可能性を使用して、次のような文を解析してみることにする。

細やかな神経を持っている　　(5)

ただし、解析には上昇型のチャート法 [4] を用いる



(a)「細やかな」に対する処理



(b)「神経」に対する処理

図 4　条件付到達可能性による解析例

ものとする。図 4(a) は「細やかな」に対する処理を行なっているところである。破線が不活性弧を、実線が活性弧を表している。到達可能性を使用するので、$[?]_s$ という活性弧がいちばん左の節点に付加されている点に注意して欲しい。図 4(a) では、まず不活性弧 [細やかな]$_m$ から不活性弧 [[細やかな]$_m$]$_{mp}$ が作成されている。このとき、条件付到達可能性の検査が行なわれるが、これについては省略する。次に、不活性弧 [[細やかな]$_m$]$_{mp}$ を基にして、活性弧 [[[細やかな]$_m$]$_{mp}$[?]$_{np}$]$_{np}$ の作成が試みられるが、このとき、範疇 np から範疇 s への条件付到達可能性の検査が行なわれる。図３より関連する部分を抜粋すると次のようになる。

np ⇒ s　　[p,v,b]∣[p,b,v]　　(6)

一方、不活性弧 [[細やかな]$_m$]$_{mp}$ の後に続く語彙範疇のリストを作成すると [n,p,v,b] となるが、(6) の条件部に出現する [p,v,b] は [n,p,v,b] の部分リストであるので、結局、範疇 np から範疇 s へ条件付到達可能であることが分かる。次に図 4(b) であるが、これは、「神経」に対する処理を行なっているところである。まず、不活性弧 [神経]$_n$ から不活性弧 [[神経]$_n$]$_{np}$ が作成されている。次に、不活性弧 [[神経]$_n$]$_{np}$ を基にして、活性弧 [[[神経]$_n$]$_{np}$[?]$_p$]$_{ppa}$ の作成が試みられるが、このとき、範疇 ppa から範疇 np への条件付到達可能性の検査が行なわれる。範疇 np は、作成が試みられている活性弧と接している活性弧 [[[細やかな]$_m$]$_{mp}$[?]$_{np}$]$_{np}$ に現れる最初の空所に対する範疇である。今の場合、図３の関連する部分は次のよ

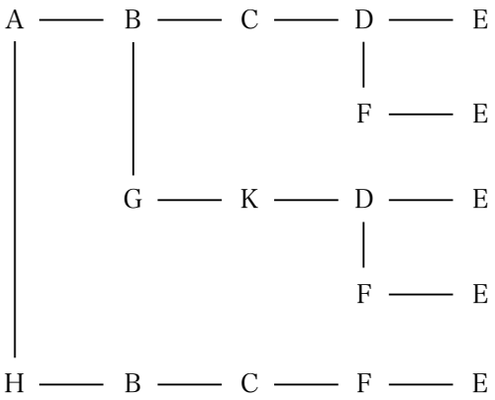


図 5　TRIE 構造による条件部の記述

うなものである。

$$\text{ppa} \Rightarrow \text{np} \quad [\text{v,b,n}]\,|\,|\,[\text{b,v,n}] \qquad (7)$$

また、不活性弧 [[神経]$_{\text{n}}$]$_{\text{np}}$ の後に続く語彙範疇のリストを作成すると [p,v,b] となるが、(7) の条件部に出現する [v,b,n] も [b,v,n] も [p,v,b] の部分リストにならないので、結局、範疇 ppa から範疇 np へは条件付到達可能でないことが分かる。このようにして、活性弧 [[[神経]$_{\text{n}}$]$_{\text{np}}$[?]$_{\text{p}}$]$_{\text{ppa}}$ の作成は抑制される。しかしながら、(7) に示されるように、ppa から np へは到達可能である。したがって、従来の到達可能性を使用した解析においては、活性弧 [[[神経]$_{\text{n}}$]$_{\text{np}}$[?]$_{\text{p}}$]$_{\text{ppa}}$ が作成され、さらに、不要な不活性弧 [[[神経]$_{\text{n}}$]$_{\text{np}}$[を]$_{\text{p}}$]$_{\text{ppa}}$ なども作成されることになる。

3．条件部の記述の圧縮

3．1　リスト構造から TRIE 構造へ

条件付到達可能性の条件部はリスト構造を選言で結合したものであるが、これらをよく観察してみるとリストはお互いに類似した部分を持っていることに気づく。たとえば、次に示すのは、データベース問合せ文用に作成された日本語書換え規則から得られる条件付到達可能性の条件部である。

$$[\text{A,B,C,D,E}]\,|\,|\,[\text{A,B,C,F,E}]\,|\,|\,[\text{A,G,K,D,E}]$$
$$|\,|\,[\text{A,G,K,F,E}]\,|\,|\,[\text{H,B,C,F,E}] \qquad (8)$$

ただし、文法範疇名は説明の都合上、分かりやすいものに変更してある。この条件部において、最初のリストと 2 番目のリストは、最初の方に [A,B,C] という共通の要素列を持っている。また、最初のリストと 3 番目のリストは、最後の方に [D,E] という共通の要素列を持っている。このような特徴を利用すれば、条件付到達可能性の条件部を圧縮することが可能となる。

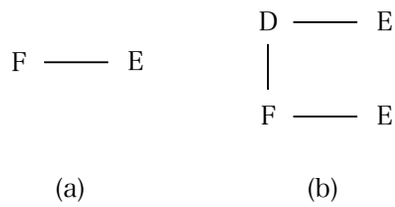


図 6　TRIE 構造における共通要素

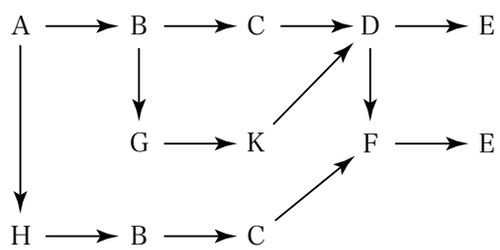


図 7　拡張された TRIE 構造による条件部の記述

まず、リストの最初の方に現れる共通の要素列であるが、これは TRIE 構造 [5] を用いれば、重複した記述を避けることができる。たとえば、条件 (8) を TRIE 構造を用いて記述すると図 5 のようになる。要素列 [A,B,C] と [A,G,K] が一回しか出現していない点に注意されたい。なお、条件部の記述に TRIE 構造を用いる効果は記憶量の減少だけではなく、条件部を検査するための処理の量も少なくてすむようになる。たとえば、条件 (8) のリスト構造の場合、処理中の不活性弧の後に続く語彙範疇のリストが語彙範疇 A を含むかどうかを 4 回調べなければならないが、図 5 の TRIE 構造だと一回で済む。

3．2　拡張された TRIE 構造による記述

リストの最後の方に現れる共通の要素列は、TRIE 構造でも圧縮できない。たとえば、図 5 においては、図 6 の (a) のような構造が 3 箇所、(b) のような構造が 2 箇所に現れていることが分かる。このような重複は、TRIE 構造を木構造から非サイクル的有向グラフ（directed acyclic graph）へと拡張することによって避けることができる。すなわち、共通部分が複数の親節点を持つことを許すのである。たとえば、図 5 の TRIE 構造を非サイクル的有向グラフを用いて書き直すと、図 7 のようになる。図 6 に示す構造が一回しか出現していないことが見て取れるだろう。

図 8　TRIE 構造のためのデータ構造

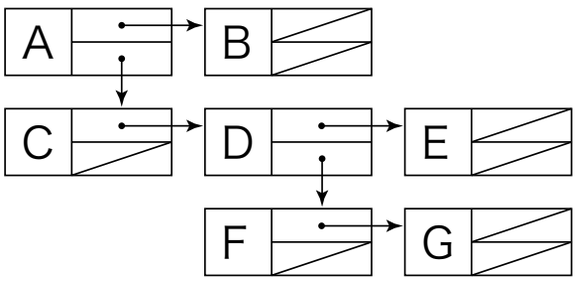


図 9　TRIE 構造の構成例

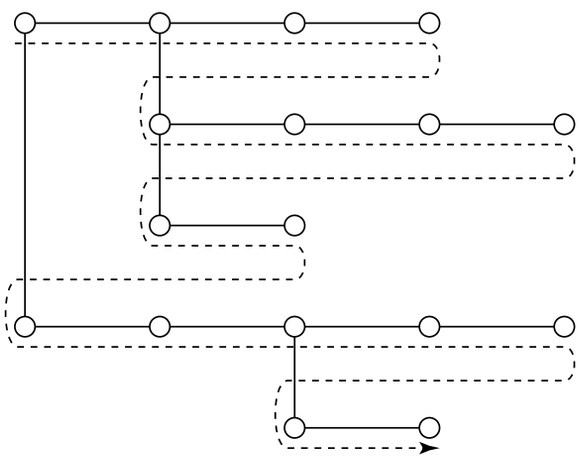


図 10　先行順による TRIE 構造の走査

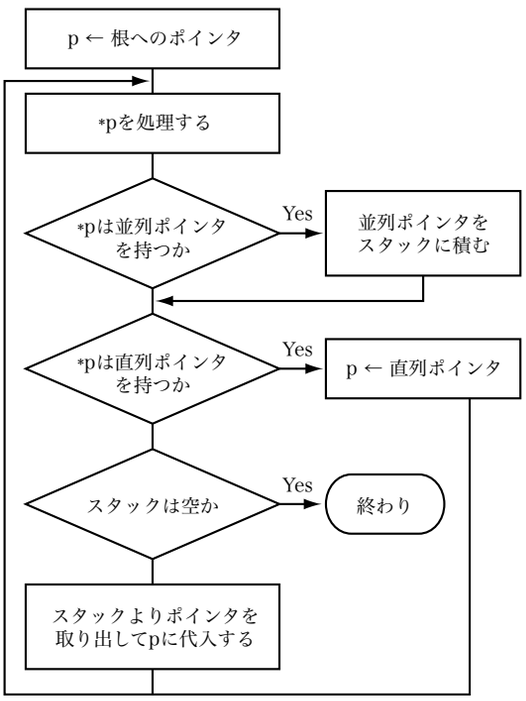


図 11　スタックを用いた先行順走査アルゴリズム

3．3　拡張された TRIE 構造の作成

ここでは、TRIE 構造がすでに完成しているものと仮定し、TRIE 構造から拡張された TRIE 構造を作成する方法について説明する。まず、TRIE 構造を図 8 のデータ構造によって構成することにする。このデータ構造による TRIE 構造の構成例を図 9 に示す。TRIE 構造から拡張された TRIE 構造を作成するには、共通構造を捜す必要があり、このためには、TRIE 全体を走査する必要がある。TRIE 構造は 2 分木の一種であるから、走査法として先行順、中間順、後行順の三つが考えられるし、また、アルゴリズムとして、再帰を用いる方法とスタックを用いる方法が考えられる。我々は、TRIE 構造の二つの部分を比較するのに便利なように、走査法として先行順、アルゴリズムとしてスタックを用いるものを採用した。TRIE 構造全体を走査する様子を図 10 に、スタックを用いた先行順走査アルゴリズムを図 11 に示す。ただし、図 11 において、*p はポインタ p が指すデータ構造を表している。

図 11 のアルゴリズムを用いると、スタックには未走査部分が積まれることになるので、*p と共通の構造をスタック上の部分 TRIE 構造内に捜すことにより、TRIE 構造全体の共通構造を求めることが可能になる。なお、共通構造が見つかったら、タグを付加することにする。すると、すでに共通構造として認識された部分は探索対象から外せるので、タグを発見したら、それ以降の構造は無視してよいことになる。以上の考察により、TRIE 構造内に共通構造を捜すためのアルゴリズムは図 12 のようになる。共通構造が求まったら、タグを基にしてポインタの付け替えを行なうことにより、非サイクル的有効グラフへと拡張された TRIE 構造が得られる。

4．実験結果

TRIE 構造および拡張された TRIE 構造の有効性を検証するために、条件部の記憶に必要なメモリ量と解析時間の両方について実験を行なった。書換

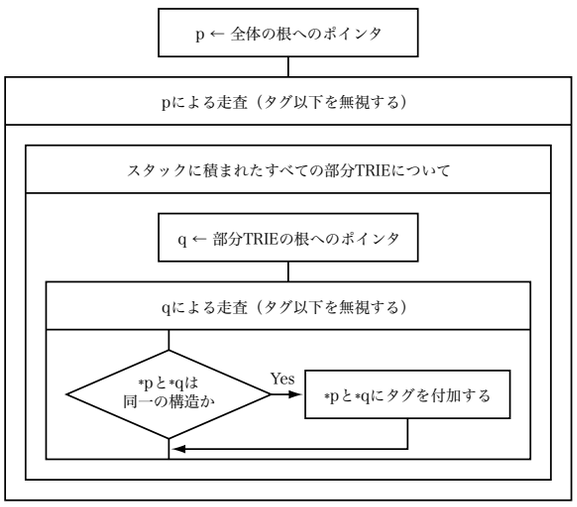


図 12　共通構造を求めるためのアルゴリズム

え規則としては、日本語によるデータベース問合せシステム [6] で使用した規則のサブセット約 100 個を用いた。実験に使用した例文の数も 100 個である。なお、実験に用いたコンピュータは Apple 社の Power Macintosh G4（クロック周波数 450MHz）で、640MB の主記憶を搭載している。

実験結果を表 1 と表 2 に示す。表 1 は条件部の記憶に必要なメモリ量を各構造に対して求めたものである。これによると、リスト構造から TRIE 構造にした場合のメモリ消費量の圧縮率は 18.6% である。また、TRIE 構造から拡張された TRIE 構造への圧縮率は 28.7% であったが、これは、リスト構造からの圧縮率で見ると 5.3% ということになり、約 1/20 に圧縮されていることになる。次に表 2 であるが、これは各構造に対し各例文の解析を 10000 回繰り返したときの時間を求めたものである。各例文において、解析時間の圧縮率を求め、これを平均すると 80% になっている。なお、拡張された TRIE 構造は、TRIE 構造のポインタを付け替えただけであるので、TRIE 構造と拡張された TRIE 構造とで解析時間は同一であった。

5．おわりに

条件付到達可能性の条件部の TRIE 構造および拡張された TRIE 構造による記述について述べた。拡張された TRIE 構造とは、木構造を非サイクル的有向グラフへと拡張したものである。実験の結果、拡張された TRIE 構造を用いれば、条件部の記憶量は解析木の記録に必要な記憶量に比べて、無視できるほど小さくできることが明らかになった。また、解析速度の改善も見られた。ところで、条件付到達可能性は不要な部分解析木を完全に除去できるわけではない。今後は、到達可能性以外の概念、たとえば、連接可能性などを併用したパーザの研究をやって行きたいと考えている。

表 1　各構造に対するメモリ消費量

構造	メモリ消費 (bytes)	圧縮率 (%)
リスト構造	408,892	100
TRIE 構造	75,876	18.6
拡張 TRIE 構造	21,780	5.3

表 2　各構造に対する解析時間（10000 回）

文番号	リスト構造	TRIE 構造・拡張 TRIE 構造	
	時間	時間	圧縮率 (%)
1	1.58	1.40	0.886
2	2.93	2.58	0.881
3	3.76	3.15	0.838
4	3.25	2.89	0.889
5	2.75	2.47	0.898
≈	≈	≈	≈
100	1260	806	0.640
平均			0.800

参考文献

[1] 笠 晃一 , 岡出高徳 , 弘中大介 , 横田将生 : 条件の付加による到達可能性の改良について , 情報処理学会論文誌 , Vol.41, No.4, pp.1028-1037.

[2] Aho, A.V. and Ullman, J.D.: *The Theory of Parsing, Translation and Compiling*, Vol.1, Parsing, P.541, Prentice-Hall, Englewood Cliffs.

[3] Pollard, C.J. and Sag, I.A.: *Head-Driven Phrase Structure Grammar*, University of Chicago Press.

[4] Kay, M.: *Algorithm Schemata and Data Structures in Syntactic Processing*, Technical Report CSL-80-12, Xerox PARC.

[5] Aho, A.V., Hopcroft, J.E. and Ullman, J.D.: *Data Structures and Algorithms*, Addison-Wesley.

[6] 笠 晃一 , 小林修二 , 白石正人 , 横田将生 : 自然言語問合せ文の意味表現方法とその応用 , 情報処理学会論文誌 , Vol.34, No.5, pp.925-933.