Vol. 29 No. 6 情報処理学会論文誌 June 1988

二次記憶上の大規模語彙を用いる自然言語処理システム†

横山晶一†† 元吉文男†† 井佐原均††

文法規則と辞書とを分離した自然言語処理システムにおいては，扱う文の数や範囲が増加するにつれて，文法規則と語彙双方とも増加する．このうち，文法規則は日本語の構文パターンに限りがあるので，ある程度のところで落ち着くが，語彙は無限に増加し，したがって辞書の大きさは拡張を続けることになる．従来の自然言語処理システムにおいては，文法規則と辞書を主記憶上に置いて，構文を処理する手法をとっているものがほとんどであった．しかしながら，上記のように，語彙の増加に伴って主記憶上の辞書だけでは容量的に対処できなくなってきている．本論文では，辞書を主記憶と二次記憶の両方に作成し，大規模な語彙を用いることができるようにした構文解析システムについて述べる．文の中に頻出する動詞，助詞，助動詞などは主記憶上におき，語彙の中で多数を占める名詞は二次記憶上に常駐させてある．二次記憶上の辞書を効率的に検索するために，辞書は順序ハッシュの形に展開されている．また，この辞書は，電子化された国語辞典の見出し語から自動的に抽出されたもので，約13万の語彙を有する．文法規則には，計算機処理を想定して書かれた水谷文法を用いた．この文法を用いると，従来文法と比較して，構文解析木の数が少なくなり，後の意味処理の負担が軽減されることも示す．

1. まえがき

構文や意味解析を行う自然言語処理システムは，これまで，どちらかといえば実験的なものが多く，その背景となる辞書については，システム開発者が独自に開発した辞書を用いることが多かった．しかしながらシステムの規模が大きくなると，そのような開発の仕方では不十分になり，辞書の形態をよく考え，語彙選択にも工夫をこらした辞書が必要になってくる．このような自然言語解析用辞書を独自に開発するには，人手と時間がかかる上に，計算機と言語双方をよく理解した専門家が必要とされる．そこで辞書の開発に，既存の国語辞典を援用すれば，開発に要する人手と時間はかなり軽減されるものと考えられる．

さて，自然言語解析用辞書の規模が大きくなると，これまでの実験規模のシステムとは異なる理念で取り扱う必要性が生まれてくる．まず最も問題になるのは大規模な辞書をどこに常駐させるかということである．通常の実験システムでは一般に辞書を主記憶上に持っている．すなわち，常に「インコア」の状態で辞書を取り扱っている．辞書の規模が大きくなった場合には，このような取り扱いをそのままにしておくことは難しくなる．そこで，当然のことながら二次記憶媒体上に辞書を置くことになるが，通常の順編成ファイルに対してアクセスを行うのでは効率が悪い．ここで述べるシステムでは順序ハッシュという手法を採用している．その詳しい内容は次節に述べるが，辞書が本来持っている順序性と，早くて効率のよい検索とを可能にしたファイル編成を行うことができる．

自然言語処理システムでもう一つ重要なのは文法情報をどのようにのせるかということである．ここでは水谷によって作成された文法[5]（以下水谷文法と呼ぶ）を用いる[16]．この文法は，従来の学校文法とは全く違った枠組の上に構築されたもので，従来扱えなかった構文が扱えたり，従来より木の深さが深いにもかかわらず，構文解析木の曖昧性が少ないなど数々の特徴を持っている．また，言語学者が機械処理を意識して書いた最初の日本語文法であることも注目に値する．水谷文法の特徴については3章で述べる．

以上のような辞書と文法を用いる基盤となるシステムには，構文解析用システムとして作られた拡張LINGOL[3),12)]を用いる．これはもともと文法や意味情報を容易に取り扱えるようにLISP上に構築されたシステムであり，活用語尾を処理するプログラム[11)]，構文解析を用いたカナ漢字変換[13)]，さらには国語辞典そのものの語釈文に対する構文解析[15)]などに応用されて発展してきた．ここではこの拡張LINGOLを漢字LISP[6)]の上にのせて漢字の入出力を取り扱えるようにした版を用いる（以下これを漢字LINGOLと呼ぶ）．従来の日本語構文解析システムでは端末からの入力を想定して，ローマ字入力を受け付けるものが多いが，ここでは，すでにデータベースなどに蓄えられた大量データを取り扱う場合を想定して，漢字のコー

† A Natural Language Understanding System with a Large Vocabulary in Secondary Storage by SHOICHI YOKOYAMA, FUMIO MOTOYOSHI and HITOSHI ISAHARA (Machine Inference Section, Information Sciences Division, Electrotechnical Laboratory).

†† 電子技術総合研究所パターン情報部推論システム研究室

ドを入力として扱うようなシステムを構築している．これによって形態素処理などが従来のシステムと異なってくるが，それについては3.2節で触れる．

2. 二次記憶上の辞書の構成

2.1 順序ハッシュ

大規模なファイルに対して効率よくアクセスしようとする場合には，ハッシングという技法をよく使用する．$x=a_1a_2\cdots a_n$ というストリングがあったとき，まずそれに対するハッシュ関数 $h(x)$ を次のように定義する．

$$h(x)=b_1f_1(x)+b_2f_2(x)+\cdots+b_nf_n(x) \qquad (1)$$

ただし $b_1, b_2, \cdots, b_n$ は，それぞれ各文字 $a_1, a_2, \cdots, a_n$ に対応した定数，$f_1, f_2, \cdots, f_n$ は適当な関数である．そして通常は得られた $h(x)$ をファイルのアドレス（レコード番号）として対応する場所に格納する．検索の場合には検索しようとするストリングに対して同じやり方でアドレス計算をしてそのアドレスに飛び，同じストリングなら探索を終了する．

このようなやり方で通常は高速に求めるストリングを見いだすことができるが，問題はハッシュ関数の値が同じになった場合，および計算したアドレスに所望とは異なるストリングが入っていた場合の処置である．通常は，アドレス j に求めるストリングがない場合には，アドレスを一つさかのぼって，$j-1$ とし，空白を検出するまで繰り返す（逐次追跡つきハッシュ法）．ファイルの先頭までさかのぼっても求めるストリングが発見できないときには先頭の一つ前をファイルの末尾と見なして同じことを繰り返す．このやり方は，空白が多い場合にはよいが，そうでない場合には，どこで停止するかが決定できず，ハッシュ法の持っている効率のよさを損なうことになる．

この欠点を改善するために開発されたのが順序ハッシュ法[1),4)]である．これは，上に述べた逐次追跡の部分を変更した次のようなアルゴリズムで構成されている．

ステップ 1：x に対するハッシュ関数 $h(x)$ を計算してアドレス $j=h(x)$ とする．

ステップ 2：j におけるストリングが辞書順で x より前にあったら探索をやめて停止する．

ステップ 3：ストリングが求めるものであれば成功して終了する．

ステップ 4：$j=j-1$ とする（アドレスをさかのぼる）．もし $j=0$ ならばファイルの末尾に行く．ステップ2に戻る．

このようにすると，辞書順で自分より前の語が出てきたときに，（空白の有無にかかわらず）必ず探索が停止する．

順序ハッシュのもう一つの特徴は，新しい単語を挿入する場合のアルゴリズムにある．このアルゴリズムは次のようになっている．

ステップ 1：x に対するハッシュ関数 $h(x)$ を計算してアドレス $j=h(x)$ とする．

ステップ 2：j におけるストリング y が辞書順で x より前にあったら y と x を入れ換える．つまり x をこのアドレスに格納し，もともとアドレス中にあったストリングを x とする．

ステップ 3：もしストリング x が空なら終了する．

ステップ 4：$j=j-1$ とする．もし $j=0$ すなわちファイルの先頭にきた場合にはファイルの末尾に移る．ステップ2に戻る．

このようにすると，ハッシュ関数が同じ値のものは，必ず辞書順に並び，それによってハッシュ値の異なる場所を侵したとしても，またそこが辞書順に並ぶので，最終的には（全くの同音語をどのように配列するかという点を除いて）一意的なファイルになる．したがって以下で述べるような構文解析システムの辞書の部分に用いるのにふさわしいと考えられる．なぜなら，構文解析の基本となる辞書のもとになるものは国語辞典であり，その上では見出し語が辞書順に並んでいるからである．もし異なる語でハッシュ関数の値が偶然同じになったとしても，修正された見出し語の配列は常に元の辞書の形を踏襲したものになる．次節では，国語辞典の見出し語から，ここに述べたような，順序ハッシュ化された辞書を構築する具体的な手法について述べる．

2.2 見出し語ファイルを用いた順序ハッシュ辞書の構成

自然言語処理システムの辞書として大規模な語彙を持つものを開発する場合には，既存の国語辞典などの情報を利用した方が開発にかかるコストなどが軽減されるばかりでなく，多くの情報を漏れなく扱うことができる．我々は，すでに新明解国語辞典を電子化して種々の用途に使用している[17),18)]ので，ここではそれを利用する．しかしながら，国語辞典の生データをそのままの形で利用するにはシステムの負担が大きすぎるので，ここではそれから抽出した見出し語ファイル[10)]を用いる．見出し語ファイルの形態を図1(A)に

表 1 見出し，重要語などの区分
Table 1 Classification of entry words, parts of speeches, and accents.

重要語区分	見出し区分	品詞	活用	語尾	アクセント
1 ＃(最重要語)	1 親見出し	1 無表記	1 カ行変格	1 -なる	0 アクセントなし
2 ＊(重要語)	2 子見出し	2 代名詞	2 上一段	2 -な	1 ⓪
3 @(非重要語)	3 派生語 (-さ，-み，…)	3* 自他動詞	3 上二段	3 -に	2 ①
	4 品詞派生語 (動寄集める)	4 自動詞	4 五段	4 -の	3 ②
	5 類音語 (㊀……)	5 他動詞	5 サ行変格	5 -たる	4 ③
		6 可能動詞	6 下一段	6 -と	5 ⋮
		7 形容詞	7 下二段	7 -する	⋮ ⋮
		8 連体詞	8 四段		N+1 Ⓝ
		9 副詞	9 ラ行変格		
		10 接続詞	10 ク活		
		11 感動詞	11 シク活		
		12 接頭語	12 特殊型		
		13 接尾語	13* ナ変		
		14 造語	14* 形動ダ型		
		15 助動詞	15* 形型		
		16 格助詞	16* 形ク型		
		17 副助詞	17* 形シク型		
		18 終助詞	18* -な，-に型		
		19 接続助詞	19* ます型		
		20* 漢語の造語成分	20* 補助ます型		

示す．この辞書の親見出し語の数は 58,409 である[14]が，子見出し，派生語も収録しているので，全部の語数は 72,747 となっている．この情報は，固定長のランダムファイルの形でディスクに収容されている．この図の「その他の情報」とは，重要語区分や品詞などで，その一部を示すと表 1 のようなものである．この表に示す番号が整数型で入力されている．品詞・活用部で＊がついているのは，元の辞書の凡例に記されていないが，辞書文中に記載のあるものを示す．

この見出し語ファイルを自動的に順序ハッシュファイルに変換する際の手順を例とともに図 1 に示す．まず(A)から見出し語の部分と，漢字仮名混じりの表記とを同列に並べてこれを JIS 漢字コード順に配列したファイル(B)を作る．JIS 漢字コードは，JIS 規格では，区点で表され，区，点各々が 7 ビットからなるが，ここでは各々の最上位に 0 を加えて 8 ビットとし，区，点の順にそれを並べた 16 ビットコードを用いる．以下ではこれを JIS コードと称する．ファイル(B)を作る理由は，入力された単語が仮名か漢字仮名混じりかによって処理を変えるような繁雑さを避けるためである，これにより，(A)に示す 32 words/record (1 word は，処理系により異なるが，32 ビットまたは 36 ビットで，そこに JIS コード 2 文字を入れる) のファイルは，(B)に示す 23 words/record のファイルへと形を変え，含まれる語数も，72,747 語から 131,968 語に増加する．増加したファイルの語数が元のファイルの 2 倍の語数よりも少ないのは，仮名表記のみの語があるのが主な原因である．

他の用途に使用する可能性もあるのでこのファイルは JIS コード順の形のままランダムファイルとして取って置き，ハッシュファイルは，このファイルに対するポインタを持つ別ファイル(C)とした．上記のような 23 words/record のファイルからこのようなポインタを作るときには，1 record の大きさは，見出し語の分として 9 words，さらに B のファイルでは同音語が並んでいるので，その語の最初のポインタと最後のポインタを付け加えることにより，計 11 words で済むことになる．このような同音語をまとめた場合の語数は，108,389 語となる．これを 199,999 records の空間にハッシュ値を用いて納める．ここで用いるハッシュ値は次のようなものである．ストリング x に対して関数 $h'(x)$ を次のように定める．

$$h'(x)=2^{14}\times a_1+2^{13}\times a_2+\cdots +2^2\times a_{13}+2\times a_{14}+\cdots \qquad (2)$$

ここで $a_1, a_2, \cdots$ は，このストリングに含まれる各文字の JIS コードである．文字数が 15 以上のものに対しては，それ以降の JIS コードを加算していく．この関数の値に対して，アドレス j を次のように定める．

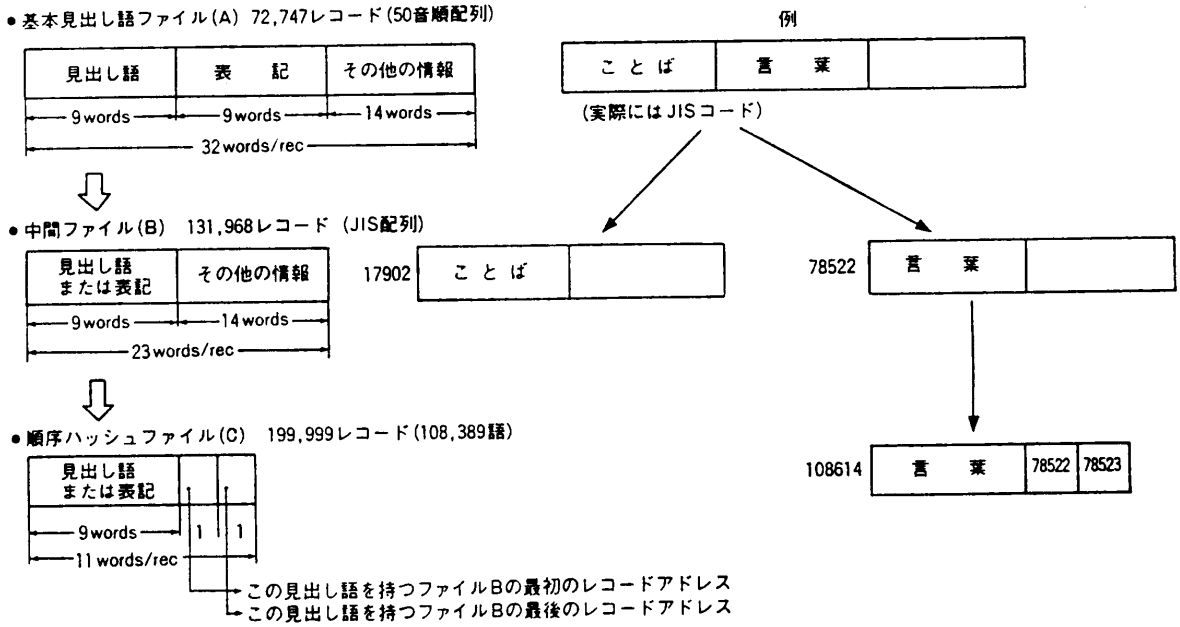


図 1 順序ハッシュファイルの作成
Fig. 1 Creation of an ordered hashing file.

```
(DE HASH (LST LGT)
    (SOSUREC (HASH1 40000 LST LGT)))
(DE HASH1 (SZ LST LGT)
    (PROG (KZ)
          (COND ((EQ SZ 0) (SETQ KZ 1))
                (T (SETQ KZ SZ)))
          (COND ((EQ LGT 1) (RETURN (TIMES KZ (CAR LST))))
                (T (RETURN
                     (PLUS2 (TIMES KZ (CAR LST))
                            (HASH1 (QUOTIENT SZ 2)
                                   (CDR LST) (SUB1 LGT))))))))
(DE SOSUREC (KZ)
    (COND ((LESSP KZ 200000.) KZ)
          (T (REMAINDER KZ 199999.)))))
```

図 2 順序ハッシュのアドレス計算関数
Fig. 2 Functions for address calculation according to ordered hashing.

$$j = h'(x) \bmod N \quad (N = 199,999) \qquad (3)$$

すなわち関数の値が全体のレコード数より小さい場合にはその数，そうでない場合には，関数の値を全体のレコード数で割った余りをアドレスと定める．このアドレスについて衝突が生じた場合には前節で述べた順序ハッシュの挿入アルゴリズムを用いて，衝突した部分を辞書順（ここでは JIS コード順）に並べ変える．

このようにして13万語強の語彙を持つハッシュファイルが完成する．検索の場合には，全く同じ式を用いてアドレスを計算することができる．(2)に示す関数はやや効率が悪いが，それでも二分木探索の検索回数 16.7 回の約半分の 8.9 回で探索可能である．**図 2** に LISP（ここでは standard LISP）で作ったアドレス計算関数のプログラム例を示す．

このプログラムは，上記(2)と(3)を計算するものである．例として図1に示した「言葉」(JIS コードで，「言」は 10 進で 14400，「葉」は 19797 である）を取り上げて説明する．まず図の第1行の LST には，「言葉」の JIS コードがリストの形で渡される．すなわち LST＝(14400 19797) となる．LGT はこのリストの長さで，ここでは2となる．2行目で，この引数は，関数 HASH 1 にそのまま渡される．HASH 1 の最初の引数 40000 は，8進で 2＊＊14 を表す．HASH 1 の中では，式(2)で示した計算を行う．この関数で，PROG 文で定義したローカル変数 KZ が，2の n 乗（$n \leqq 14$ の整数）を表す．初期値は 2＊＊14 にセットされる．結局式(2)の a_1 に 14400，a_2 に 19797 が代入され，$h'(x)$ の値（すなわち HASH 1 の最終的な return value) は，398106624 という大きな数になる．これが図の2行目の SOSUREC の引数となる．SOSUREC では，式(3)に書かれた計算を行う．すなわち，もし

引数が199,999以下であればそのまま値を返す（図で下から2行目）．そうでなければ（いまはこのケースである）199,999で割った余りを返す．図の一番下の行の REMAINDER という関数は，余りを返す組み込み関数の名前である．この計算によって，HASH の return value は108,614となり，「言葉」の入っているファイル（図1で示したC）のアドレスが分かる．ここで実際にファイルCにアクセスして，「言葉」の JIS コードとファイルの内容とを照合する．一致すれば所望の順序ハッシュファイルに到達したことになる．ここからさらに，ファイルBへのポインタをたどって品詞情報を取り出す．そして「言葉」と品詞情報とを組み合わせたものが辞書引きの結果として自然言語の構文解析システムに送られる．以下の操作については3章で述べる．「言葉」の辞書引きの場合には，ファイルアクセスはB，Cとも一回ずつですんでいる．これはハッシュ関数を用いた成果である．

3. システム上の辞書と文法規則の構成

3.1 システム上の辞書の構成

構文解析システムは，基本部分に LISP を用いたプログラミングツールである漢字 LINGOL を用いて構成している．漢字 LINGOL は，文脈自由文法に，文脈に依存した条件部や関数を書くことのできるメッセージ部を付加して，構文解析木の数を増大させずに解析を行うシステムで，文法規則と辞書とを組み合わせて，入力文から構文解析木を出力する．文法規則の部分と辞書の部分は全く独立に書くことができるが，辞書引きから最初の構文木（すなわち構文解析木の葉の部分）を作る場合の用語の統一，品詞の一致などの整合性は必要になる．

このシステムの辞書部を以下に示す．まず，2.2節で述べた約13万語から成る辞書がバックグラウンドとして二次記憶上にあるが，句読点，格助詞，動詞の語尾，サ変などの特殊な動詞は，入力文を解析するのに必ず用いられるものであるから一次記憶上に常駐させてある．また，ここで用いる水谷文法独自の用語を持つ特殊な品詞（従来文法のカテゴリと異なるもの）は一次記憶に置いてある．

図3には，これら一次記憶上に常駐している辞書の一部を示す．辞書は，（見出し語　品詞　（メッセージ　評価値）　意味部）という構成であるが，現在評価値と意味部は使用していない．図で，句読点，動詞，格助詞，助動詞，動詞語尾（カ未然などという品詞名を持つもの）は，従来文法の記述[15]と同じである．メッセージ部が NIL となっている部分は，構文解析に際して何もメッセージを送らないが，特動詞，動詞語尾，助動詞のところでは，各々記されたメッセージを送る．送られたメッセージは文法規則を適用する場合に考慮されるが，構文解析木の上に陽に現れることはない．現在このメッセージは，主として形態素解析に使用しており，図に示すように，HEAD, BODY, TAIL の3部分に分けてある．HEAD は前の語の TAIL との照合をとり，TAIL は後ろの語の HEAD との照合をとる．BODY は特殊な品詞のみを用いる文法規則部で使用する．構文解析を行う際に，これらの照合が失敗すれば木を形成しないので，木の数の増大を抑えることができる．

図で感応言，文結合子，情況語などが水谷文法独自の用語である．感応言，文結合子は，それぞれ従来文法の感動詞，接続詞にあたる．情況語は，情況化語と合わせて，従来文法の副詞的な句を作る．なお，二次記憶上の辞書は，表1の分類でも分かるように，従来文法にそった品詞名を用いているが，従来文法での品詞の概念とほぼ一致するものは，たとえば名詞→体言などの言い換えを行って用いている．

3.2 水谷文法の拡張 LINGOL 上への移植

文法規則の部分は，すでに述べたように，水谷文法

```
(。  終 (NIL 0) NIL)
(・中点 (NIL 0) 0)
(、読点 (NIL 0) 0)
(ああ  感応言 (NIL 0) 0)
(ええ  感応言 (NIL 0) 0)
(行  動カ語幹 (NIL 0) 0)
(せ  特動詞 (((BODY (サ変)) (TAIL (未然形))) 0) 0)
(し  特動詞 (((BODY (サ変)) (TAIL (連用形))) 0) 0)
(し  特動詞 (((BODY (サ変)) (TAIL (未然形))) 0) 0)
(か  カ未然 (((TAIL (未然形))) 0) 0)
(く  カ連体 (((TAIL (連体形))) 0) 0)
(ない  助動詞
       (((BODY (NAI))(HEAD (未然形)) (TAIL (連体形))) 0) 0)
(まし  助動詞 (((HEAD (連用形)) (TAIL (連用形)) 0) 0)
(ます  助動詞 (((HEAD (連用形)) (TAIL (連体形))) 0) 0)
(に  格助詞 (NIL 0) 0)
(を  格助詞 (NIL 0) 0)
(は  格助詞 (NIL 0) 0)
(しかし  文結合子 (NIL 0) 0)
(急  情況語 0 (NIL 0) 0)
(に  情況化語 ((TAIL (情況化))) 0) 0)
```

図3　一次記憶上の辞書の一部
Fig. 3　A part of the dictionary in main memory.

をベースとしている．水谷文法は，国語学者が機械処理を意識して書いた数少ない文法の一つであるとともに，日本語に特徴的な言語現象のいくつかを見通しよく扱う意図で書かれている．また，西村・水谷らが計算機による自然言語処理を試みた文法体系[8),9)]に比べて，文法規則の内容がかなり簡潔になり，規則の数も少ないという特徴がある．本節では，この水谷文法の特色を簡単に述べるとともに，漢字 LINGOL 上への移植と，その際の問題点について述べる．

まず，水谷文法の特徴としては次のような点が挙げられる．

1. トップダウンで，ほぼ文脈自由文法で書かれており，それに条件部が加わっている（規則によっては，有限状態文法で書けることを示唆している部分もある）．

2. 主として句より上のレベル，文より下のレベルを扱っていて，それ以下の形態素，語尾解析の部分，また文を越える部分についてはオープンになっている．ただし終端記号にまで言及したところもある．

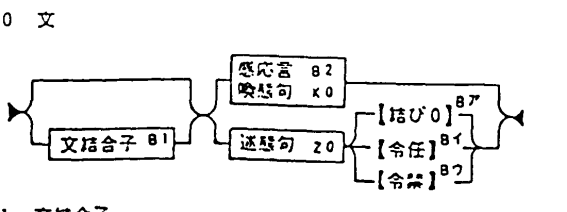
3. 山田・時枝文法の流れをくんでいるために，従来の学校文法とは用語，規則の提示の仕方が異なる．

4. 再帰的な規則はほぼ右下がりの木を生成するように書かれている．右下がりの木を生成すると，曖昧な木を生じにくいことが知られている[2)]．

5. 2のほかにもオープンのままにしている記述がある．また，この文法の範囲を越える文も明示している．

図4に水谷文法の規則の一部を示す．この図で，【　】でくくったのが条件部（たとえば規則Bア）で，［　］は任意記号（すなわち，この部分を適用してもしなくてもよい），＊は活用形を介したつながりを示す．

図4のような文法規則を，漢字 LINGOL 上にのせる．文法規則としてトップダウンに書かれた部分（たとえば図の規則B1）は，そのままの形で LINGOL 上にインプリメントできるが，条件部や形態素処理部は次のように扱う．



B1 文結合子
文をも接続し得る句接続詞[Z8]，及び文語「が」の類

B2 感応言
ーアア｜エエ｜オヤ〔マア〕｜サア｜オイ｜ネエ｜ウン｜ハイ｜

Bア 【結び0】
直前要素xで仕訳：
$x = \epsilon, \forall (x \in$ 助動詞[Z4] $\wedge x$形$\in$終止形) なら
x〔結び0〕$= x$［不定辞[B3]］［＊添加辞0[B4]］：

図4 文法規則の一部[5)]
Fig. 4 A part of rules in Mizutani grammar.

```
(文 ((文1 NIL)(終 NIL)) 0 0)
(文1 ((文1 NIL)(読点 NIL)(文1 NIL) 0 0)
(文1 (@ (文結合子 NIL)(感応言 NIL)) 0 0)
(文1 (@ (文結合子 NIL)
        (述態句 (COND ((MEMBER '(連体形) (MM 'TAIL)) NIL)
                      (T T)))
        @(不定辞 NIL)
        @(添加辞0 NIL)) 0 0)
(述態句 ((述態句0 (PMALL))) 0 0)
(述態句0 ((述業 (PMALL))
          (助動詞
            (COND ((EQUAL (FM 1 'TAIL) (MM 'HEAD))(PMALL))(T T)))) 0 0)
(述業 ((用連語 (PMALL))) 0 0)
(用連語 ((用連語0 (PMALL))) 0 0)
(用連語0 ((動詞 (PMALL))
          @(助動詞
            (COND ((EQUAL (MM 'HEAD)(FM 1 'TAIL))
                   (PMALL))(T T)))) 0 0)
(動詞 ((動カ語幹 NIL)(カ連用 (PMALL))) 0 0)
(動詞 ((動カ語幹 NIL)(カ連体 (PMALL))) 0 0)
```

図5 LINGOL 上の文法規則の一部
Fig. 5 A part of grammar rules implemented in LINGOL.

1. 条件部は漢字 LINGOL 上でのメッセージ機能の中に埋め込む．したがって，構文解析木の上に陽に現れることはない．

2. 形態素処理部は，従来の構文解析システム[15]上のものをこのシステムにのる形に変える．

3. 現在の LINGOL 上では終端記号をそのまま扱えないので，一段非終端記号を付加する．

図5に，図4に相当する部分の文法の一部およびその他のいくつかの規則を上のような条件に従って漢字 LINGOL にのるような形に書き直したものを示す．文法規則は（**A**（(B 条件部)（C 条件部）…）評価値 意味部）という形で A→B C …という規則を表す．B, C, … それぞれには条件部がついている．条件部には，LISP 関数の形でメッセージ関数を書く．この図で@は，図4の［ ］と同じで，任意記号を示す．すなわち，その規則に対する適用，または不適用を許す．LINGOL には，このほかにも，0回からの任意回の繰り返しを許す＊，1回からの任意回の繰り返しを許す＋という記号があり，いずれも水谷文法をインプリメントする際に用いられている．@は，文法が入力されると同時に展開される．たとえば図4で，文1を文結合子（接続詞），述態句（述語），不定辞（語尾の副助詞「か」など），添加辞0（終助詞）に展開する文法は，述態句を除く3つの要素がいずれも任意であるので，結局8通りの異なる文法規則に展開される．その他の繰り返し規則（＊，＋）は任意回の繰り返しを許す規則であるから，実際にその文法が適用されたときに展開して用いられる．

用語は図4にあるものとほぼ同じになるようにしてあるが，構文解析木の根の部分を「文」とするために，「文」→「文1」という二段構えの規則を設けてある．また，この文法には，句読点に関する記述が含まれていないので，それらの扱いも付け加えてある．図4と図5を比較すれば分かるように，条件部（たとえばBア）は，ほとんど原形を止めないで規則中に埋め込まれている．これにより，条件部が外に現れない解析木が生成される．また，図3に示した辞書と，図5とを比較すると分かるように，辞書からのメッセージが文法規則上のメッセージ関数（図で MM, FM, PMALL）で用いられて構文解析木に制限を加える．**図6**には，すでに示した簡単な文法と辞書を用いた，動詞の活用形のみからなる文の構文解析木を示す．上に述べたように，条件部は外に現れていない．もとの文法の終止形は，ここではすべて連体形として扱っている．述態句，述素は従来文法の述語に相当し，用連語はほぼ用言に当たる．図で分かるように，動詞の活用語尾の部分は，「カ未然」などの形式で用意されており，ローマ字で解析をする場合のように，“ik” と “a” という分かれ方に比べてやや形態素の規則が多くなる．なお，この形態素解析の部分は，水谷文法には明記されていないので，ほぼ従来文法と同じ規則を用いている．このようなやり方で，水谷文法がほぼそのままの形で漢字 LISP 上にインプリメントされる．

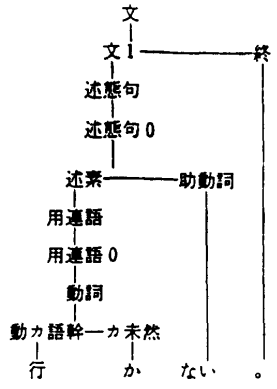


図6 構文解析木の例
Fig. 6 An example of parsing tree based on the grammar rules in Fig. 5.

4. 日本語に対する構文解析例

図7に，国語辞典の語釈文からとった例を示す．「ああ」という見出し語の語釈文「急に驚きなどした時に出す，言葉にならない言葉」の構文解析結果を，従来の文法[15]による解析木の一例（図の(a)）と，水谷文法による解析木の一例（図の(b)）とで示す．木の深さは後者の方が深いが，木の数は前者の方がはるかに多い．この図の文を解析すると，前者は 44，後者は4の構文解析木が得られる．前者の文法の方が，比較的少ない規則を再帰的に用いているのに対して，後者は木の深さは深いが，途中の枝分かれの数が少なく，また前述のように，右下がりの規則が多いので，木の曖昧性が少なくなっているものと考えられる．木の数が少ないと，以後の処理（意味処理や，機械翻訳に用いる場合の木変換など）が楽になるし，構文解析に要する時間も短くなる．また，解析木を収容する記憶量も少なくて済む．この構文解析システムは，もともと一般のテキストに現れる文を扱うために作られたものであるが，この図は，やや特殊な表現の多い国語辞典の語釈文の解析も可能であることを示している．

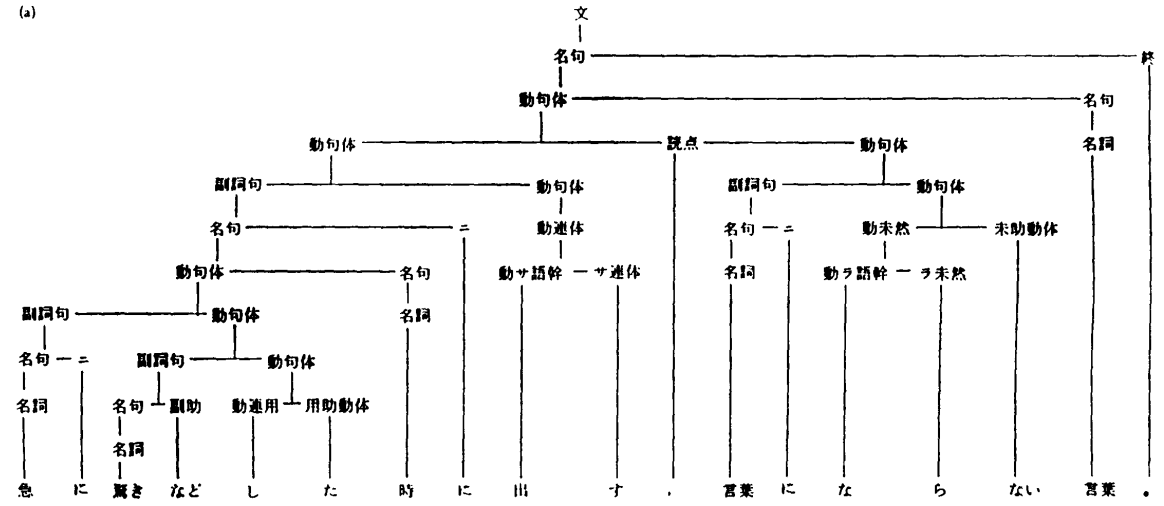


図 7(a)　辞書の語釈文の構文解析木例（従来文法）

Fig. 7(a)　An example of parsing tree of a definition sentence in the dictionary (using a traditional grammar).

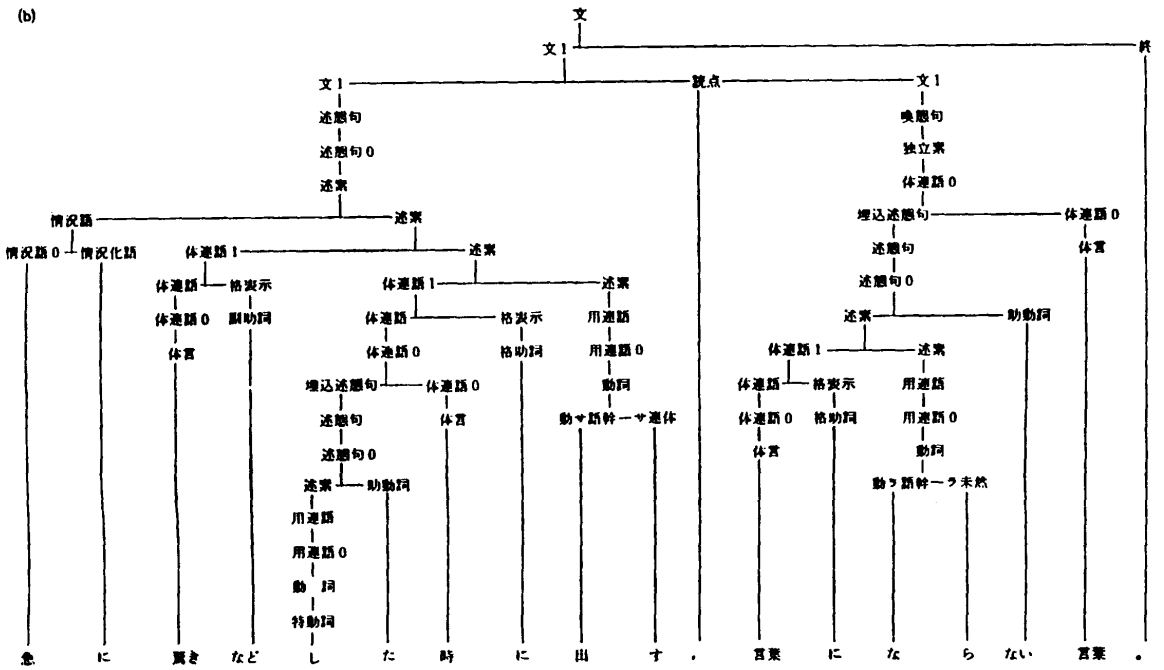


図 7(b)　辞書の語釈文の構文解析木例（水谷文法）

Fig. 7(b)　An example of parsing tree of a definition sentence in the dictionary (using Mizutani grammar).

この図で，辞書引きについて見てみると，どちらも「驚き」，「時」，「言葉」が二次記憶上に置かれたものである．「言葉」の辞書引きについては，すでに 2.2 節で説明した．他の語の辞書引きも，全く同じ手順で行われる．(a)では「急」も二次記憶からの辞書引きであるが，(b)の場合には，この言葉が特殊なカテゴリに入れられているので，辞書項目を一次記憶上に設けてある．このように，大規模な辞書では，特に名詞（体言）を二次記憶上におくことが有効であり，必要である．ここで二次記憶から高速で辞書引きが可能なの

は，すでに述べたようにハッシュ関数を有効に使用しているためである．ちなみに，この国語辞典データベースでは，全見出し語の8割以上が名詞であり，それを展開したこの辞書の名詞も同じ割合になっているものと考えられる．一方，助詞や助動詞は種類も少なく，しかも文の中に出てくる頻度が高いので一次記憶に常駐させておく方が辞書引きの効率がよいと思われる．このような「仕分け」を行うことによって解析の能率が向上し，しかも的確な解析を行うことができる．

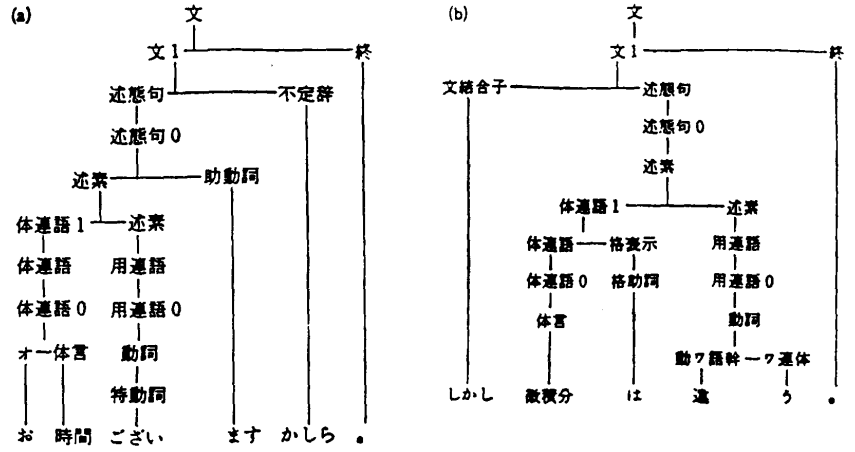


図8 構文解析の例
Fig. 8 Two specific examples of parsing trees based on Mizutani grammar.

図8には，従来文法と異なる解析木を示しておく．この図では，もとの文法規則を示すための例文[5]中からとった簡単な文を構文解析している．この図から分かるように，この文法では，文から終端記号までのパスが比較的長く，枝分かれの箇所も少なくなっている．図で(a)の文は，従来文法では扱わなかった文である．(b)も前の文とのつながりを文結合子（従来の文法では接続詞）でつないでいる．また，動詞の活用形の情報などは，途中からメッセージの形で上へ伝えられているために，木の上には見えない形になっている．終端記号まではかなり長いパスであるが，途中であまり枝分かれをせず，一本につながっているために，木の数が従来の文法に比べて特に多くなるという訳ではない．ただし，現在のところ，文法中で陽に示されたものを除いては，特にメッセージを用いて制限することをしていないので，従来の文法体系でいういわゆる埋め込み文（水谷文法では埋め込み文を認めない立場をとっている）のようなものが来た場合には，同じ規則が繰り返し適用されるので，従来の文法と同じように，係りの部分で多くの木が生じる．このあたりは文脈自由のみで書かれた文法に共通した問題点である．

5. まとめと考察

二次記憶上に約13万語の辞書を有する構文解析システムを漢字 LINGOL の上に構築した．その結果次のようなことが分かった．

①助詞，助動詞，動詞などのよく辞書引きされる語を一次記憶に置き，主として体言（名詞）を二次記憶から辞書引きしているが，構文解析の時間に影響を与えることなく，辞書引きを行うことができた．これは二次記憶上の辞書を順序ハッシュで配列しているために，二次記憶へのアクセス回数が少ないからである．

②体言を二次記憶に置いたために，全語彙の約8割を占める体言を一次記憶上の領域をとって確保する必要がなくなった．体言は，解析のときに必ず現れる語彙が動詞，助動詞などに比べて少ないので，一次記憶に常駐するものは少なくてもよい．

③文法規則によって構文解析木の数がかなり異なる．ここで用いた水谷文法は，木の深さは深くなるが，従来文法に比べて構文解析木の数が少なくなる傾向が見られた．解析木の数が少ない方がその後の意味解析が楽になる．

④このシステムは漢字仮名混じり文を直接扱えるので，電子化された大量の文書を解析することができる．また文法規則，辞書が漢字表記なので人間に見やすくなっている．

漢字 LINGOL 上に水谷文法をインプリメントする際の問題点としては次のようなものがある．

1. 文法規則中に現れる7種類の零記号（言い切り，中止などを表すのに用いる）のうち，LINGOL 上に表現しにくいものがある．たとえば，格助詞が欠如したことを表す零記号をこのシステムで表現するのは難しい．

2. 条件部のうち，システム上で表現することが難しいものは，オープンのままにしてあるが，こうするとループを生じる規則がある．

また，この構文解析システムの問題点として次のようなものが挙げられる．

1. 二次記憶上の辞書は，現在混ぜ書きを考慮していない．したがって，たとえば「よびかけ」と「呼び掛け」は入っているが，「呼びかけ」は入っていない．しかし一般には，このように混ぜ書きをすることが多い．

2. 一次記憶と二次記憶との間の語の配分は，解析時間に大きな影響を与える．現在二次記憶上には，主として体言（名詞）をのせてあり，方針としてはそれでよいと考えられるが，重要なものは一次記憶上に（頻度などを用いて）自動的に移動させるなどの措置が望ましい．

3. 辞書の展開の問題点もある．新明解国語辞典には，すでに述べたように，「名詞」というカテゴリが陽に示されていないので，品詞が付加されていない見出し語は，すべて名詞になってしまう．上に述べたような辞書引きのミスマッチも，このような語が「悪さ」をしている場合が多い．したがって不良な語を取り除くための簡単なツールも必要になる．

さらに，構文解析システムに共通する問題点として，次のことが挙げられる．

最初の辞書引きの時に最長一致法を用いているために，特にこのような大規模辞書を備えた文法解析システムでは，誤った辞書引きによる構文解析木が作られる可能性がある．この種のシステムでは，最長一致法で誤ると，それに対して適用可能な構文解析木が存在すれば，バックトラックによってもう一度辞書引きをやり直し，正しい解析木を得るということができなくなるという欠点がある．辞書引きを最長一致にしなければこの点は解決されるが，候補単語の増加が問題となるし，辞書と文法を統一して，各文字が集まって単語を形成するような文法も書くことは可能である[7]が，辞書とのマッチングのタイミングが難しくなって，特に二次記憶上に辞書を置くような場合には適切でない．名詞連続には，名詞ごとに分析を行って，連続に対する制限をつけることも考えられるが，そのためには意味まで含めた解析が必要になる．

意味の導入は構文解析システムに共通する問題点である．構文解析と意味解析を同時に行った場合には，構文解析の「明確さ」を損なう可能性があるし，構文解析木に対して意味解析をかぶせるやり方（LINGOLはこのようなやり方をとっている）であると，構文解析木の数が多くなった場合の取り扱い方が難しくなる．また，日本語の場合には語の意味を表現する手法である，知識表現法が確立したとは言い難い．これらの問題点を解決すべく，さらに研究を進めている．

謝辞 日頃御討論いただく，電子技術総合研究所パターン情報部推論システム研究室の皆様に感謝いたします．

参考文献

1) Amble, O. and Knuth, D. E.: Ordered Hash Tables, *The Computer Journal*, Vol. 17, No. 2, pp. 135-142 (1974).
2) 井佐原均，田中穂積：日本語文法作成に関する一考察，第22回情報処理学会全国大会論文集，3M-1 (1981).
3) 井佐原均，元吉文男，田中穂積：*N*進木拡張LINGOLのユーティリティ関数について，電総研彙報，Vol. 46, No. 12, pp. 740-766 (1982).
4) クヌース，D. E.（野崎昭弘（訳））：アルゴリズム，サイエンス，1977年6月号，pp. 46-59 (1977).
5) 水谷静夫：国文法素描，水谷静夫，石綿敏雄，荻野孝野，賀来直子，草薙 裕：文法と意味Ⅰ，朝倉日本語新講座3，pp. 1-80，朝倉書店，東京 (1983).
6) 元吉文男：漢字 LISP，第24回情報処理学会全国大会論文集，2L-1 (1982).
7) 元吉文男，井佐原均，石崎 俊：日本語用完全横型探索構文解析法，第32回情報処理学会全国大会論文集，4S-3 (1986).
8) 西村恕彦，水谷静夫，尾上圭介，田中幸子：日本語基本文法—単文篇，電総研研究報告，No. 783 (1978).
9) 西村恕彦，水谷静夫，尾上圭介，大野美江子：日本語基本文法—複文篇，電総研研究報告，No. 784 (1978).
10) 荻野孝野：国語辞典ファイル化作業，計量計画研究所研究報告 '81, pp. 31-40 (1982).
11) 田中穂積，横山晶一：活用語尾を処理するプログラムについて，第17回情報処理学会全国大会論文集，138 (1976).
12) 田中穂積：計算機による自然言語の意味処理に関する研究，電総研研究報告，No. 797 (1979).
13) 横山晶一，元吉文男，田中穂積：構文解析を用いたカナ漢字変換，第18回情報処理学会全国大会論文集，102 (1977).
14) Yokoyama, S.: Occurrence Frequency Data of a Japanese Dictionary, *Bul. Electrotechnical Laboratory*, Vol. 45, Nos. 9, 10, pp. 395-418 (1981).
15) 横山晶一，元吉文男，井佐原均：辞書の語釈文の構文解析について，第26回情報処理学会全国大会論文集，6H-1 (1983).
16) 横山晶一，元吉文男，井佐原均：水谷文法を用

いた日本語構文解析システム，第28回情報処理学会全国大会論文集，5M-3 (1984).
17) 横山晶一，荻野孝野：国語辞典磁気テープのドキュメント，電総研彙報，Vol. 48, No. 8, pp. 672-677 (1984).
18) 横山晶一，荻野孝野：国語辞典磁気テープのドキュメント—第三版に基づく磁気テープ—，電総研彙報，Vol. 50, No. 11, pp. 1110-1119 (1986).

(昭和62年12月7日受付)
(昭和63年4月19日採録)

横山 晶一（正会員）

昭和24年生．昭和47年東京大学工学部計数工学科卒業．同年電子技術総合研究所入所．現在パターン情報部推論システム研究室主任研究官．工学博士．この間，昭和59～60年西独シュトゥットガルト大学客員研究員．自然言語処理，電子辞書，知識表現，音声認識の研究に従事．計測自動制御学会，日本音響学会，計量国語学会，日本認知科学会，ACL 各会員．

元吉 文男（正会員）

昭和26年生．昭和49年東京大学理学部物理学科卒業．昭和51年同大学院理学系研究科物理学専門課程修士課程修了．同年電子技術総合研究所入所．現在，パターン情報部推論システム研究室主任研究官．自然言語処理，数式処理等の記号処理の研究に従事．著書「LISP で学ぶ認知心理学—言語理解」(共著) 東大出版会，「数式処理システム」(共著) 昭晃堂．日本ソフトウェア科学会会員．

井佐原 均（正会員）

昭和29年生．昭和53年京都大学工学部電気工学第2学科卒業．昭和55年同大学院工学研究科電気工学専攻修士課程修了．同年電子技術総合研究所入所．現在パターン情報部推論システム研究室主任研究官．自然言語理解，知識表現の研究に従事．日本認知科学会，ACL 各会員．