自然言語処理 65－4
（1988. 3. 25）

分かち・構文・意味の平行処理をおこなう日本語パーサ

寺下 陽一　二口 邦夫
金沢工業大学

形態素解析，構文解析，意味解析を同時平行的におこなう日本語解析用パーサを作成し，若干の実験をおこなった。パージングはＡＴＮ方式に基き，それを拡張したものであり，入力文は文末から文初へスキャンすることとし，従って文法規則も逆向きに定義される。これにより左回帰の困難を避け，さらに活用語尾の解析が効率的におこなえるようになった。単語分割を含む形態素解析は，ＡＴＮアクション“ＣＡＴＥＧＯＲＹ”を拡張することにより実現した。意味解析をおこなうに際しては，種々の意味構造作成規則を通常の構文規則と混合して記述し，パーサ自体に意味処理のための特別な機能は持たせていない。意味構造としては格文法形式を基本とし，動詞（形容詞，形容動詞を含む）およびそれに付随する種々の役割（ｒｏｌｅ），格助詞，名詞の意味素性の関連から意味の整合性をチェックするようになっている。この方式のパーサが効率などの点で実用に耐えうるかをみるには，さらに広範な実験が必要であるが，現在のところネガティブな結果は得られていない。また，パーサの構造が簡単であり，文法記述も平易におこなえるので，少なくとも種々の日本語解析の実験をおこなうのに適していると思われる。

A JAPANESE-LANGUAGE PARSER CAPABLE OF PARALLEL HANDLING OF MORPHOLOGICAL, SYNTACTIC AND SEMANTIC ANALYSES

Yoichi TERASHITA and Kunio FUTAKUCHI
Kanazawa Institute of Technology
Nonoichi, Ishikawa 921, Japan

A Japanese-language parser, capable of parallel handling of morphological, syntactic and semantic analyses, was developed. The parser is based on the ATN scheme, with some extensions by means of which to achieve the above goal. The input sentence, written without word delimiters, is scanned in reverse direction, and the grammar is defined accordingly. This arrangement makes it possible to avoid the difficulty of left-recursion (a feature of Japanese sentences), and at the same time simplifies the handling of word inflection. Morphological analysis, including word extraction, is imbedded in the parser through an extended CATEGORY action. Semantic rules can be intermixed with the usual syntactic rules in such a way that no special features are needed within the parser itself. Semantic structures are patterned after the case-grammatical formalism. By consulting the dictionary, semantic relations between individual verbs, nouns and particles are examined to determine whether or not a given triplet is acceptable. It is still early to conclude if a parser of this type can effectively be used in a practical environment, but the results so far obtained are encouraging. It can at least be said that, owing to its simplicity, the parser is suitable for various experiments involving Japanese-language analyses.

1．はじめに

自然言語の解析をおこなう場合，レベルの異なる幾種類かの処理をどの時点でおこなうべきかが問題になる。日本語の場合には，分かち処理を含む形態素解析，構文解析，意味解析，という３種の異なった処理を，段階的におこなうべきか，同時平行的におこなうべきか，あるいはその中間的な方式によるべきかが議論の分かれるところである。これら３種の処理は互いに依存しているため，同時平行的におこなえば解析の途中あるいは最終的に生成される解析結果の個数がしぼりこめ，無意味な探索操作を除去することが出来るので，この方式が望ましいということになるわけであるが，コスト，システム構成上の問題から，同時平行方式は日本語処理ではあまり用いられていないようである。

この報告では，以下のような方法を用いることにより，同時平行処理型パーサが割合簡単に実現でき，また，これが効率的にも他の方式に比べて不利でなく，さらに構造が簡単なため，種々の実験に適していることを示す。

ａ）パージングにはＡＴＮ方式を用いる。
ｂ）入力文は文末から文初へとスキャンし，従って文法も逆向きに定義する。
ｃ）分かちを含む形態素解析はＡＴＮアクション“ＣＡＴＥＧＯＲＹ”を拡張することによりおこなう。
ｄ）意味解析をおこなうために，構文解析木のノード（現在は文と名詞句のみ）に意味構造（格フレーム）を持たせ，これを操作するための諸関数を構文規則と混合して文法に含める。

ただし，ここでいう意味構造は現在はかなり表層的なものであり，辞書情報のみにより決定されるものである。従って，パーサの出力は必ずしも充分な意味構造を備えたものではない。当面の我々の目的は，文の厳密な意味を抽出することではなく，意味的制約を用いることにより，形態素解析，構文解析のあいまいさを除去することに重点を置いた。

2．諸定義

2．1　ＡＴＮアクション

以下に，我々のシステムで用いられている種々のＡＴＮアクションの記述形式を略述する。これらは Charniak & McDermott(1985)[1] によるものである。

(CAT x)：次に読み込まれた単語のカテゴリーが x かどうかを調べる。
(PARSE x)：x なるネットワークに移る。
(SEQ x_1 x_2・・・)：x_1，x_2・・・なるアクションをこの順序でおこなう。
(EITH x_1 x_2 ・・)：x_1，x_2・・・のいずれかをおこなう。
(OPT* x)：x を０回または任意回数おこなう。
(OPT x)：x を０回または１回おこなう。

最初の CAT(Category)アクションは，形態素解析を可能とするために拡張されており，この定義は必ずしも正しくない。詳しくは後に述べる。図１はこれらのアクションにより記述された単純な日本語文法の例であり，図２に簡単な例文の構文解析木を示す。この例では，形態素解析，意味解析は含まれていない。従って，解析木は複数個できるわけであるが，ここでは正しいもののみを示した。なお，１章で述べたように入力文は文末から文初へとスキャンされ，従って，文法も逆向きに定義されている。

```
S：(SEQ (CAT V)
        (OPT* (PARSE RP)))

RP：(SEQ (CAT P) (PARSE NP))

NP：(SEQ (CAT N)
         (OPT (PARSE S)))
```

図１．簡単な日本語文法

文法中に現れるフレーズマーカの意味は

Ｖ：動詞，Ｐ：格助詞，Ｎ：名詞，ＮＰ：名詞句
ＲＰ：役割句（Role Phrase）　Ｓ：文

である。なお，パーサはこのような文法および辞

書情報を参照しながらトップダウン的に解析を進め，構文解析木（および後に述べる意味構造）を作成する。

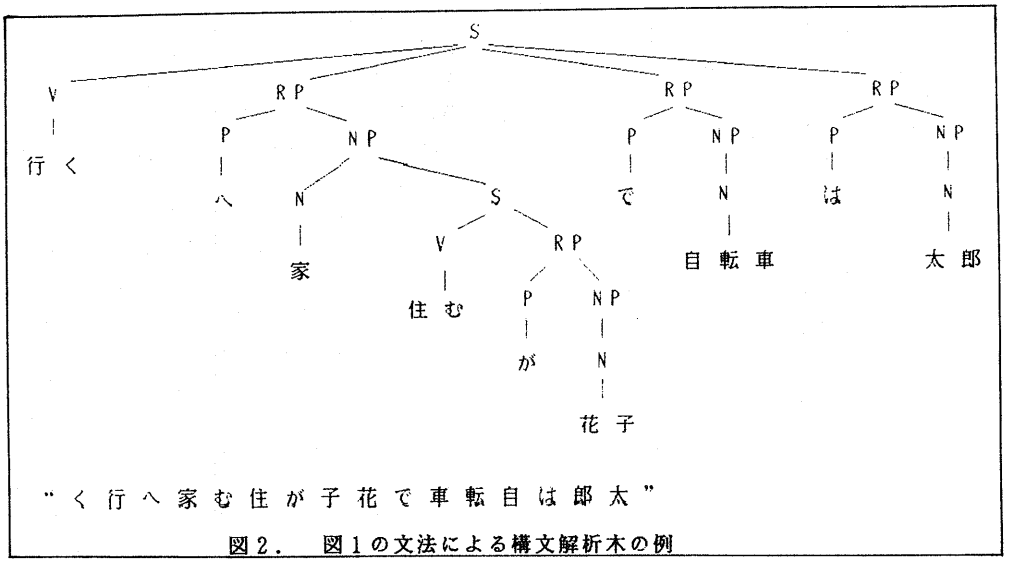


図２．　図１の文法による構文解析木の例

２．２　構文解析木およびその操作

文解析中に生成される構文解析木の各ノードには種々の属性（およびその値）を付加するようになっている。それらの属性のうちで，意味解析に用いられるものがＲＥＦ（Referent）属性であり，これはそのノードに対応する意味構造を指すポインタである。現在のところ，このＲＥＦ属性はＳノードとＮＰノードに対して定義されている。

構文解析木を操作する変数および関数のうち主要なものを以下に示す。

$LAST：直前に完了した（POP された）ノード。

($$ nd)：現在作業中のレベルより上にある nd 型のノードで，最も近いものを戻す。例えば，($$ NP),($$ S)など。

(THE attr OF nd)：指定されたノード(nd)の，指定された属性(attr)の値を戻す。例えば，(THE 'REF OF ($$ NP))。

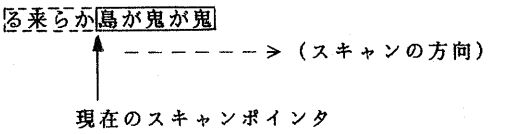
(GET-WORD tnd)：指定された終端ノード(tnd)の単語を戻す。

(THE-FIRST c OF nd)：ノード(nd)の子でｃなるカテゴリを持ち，一番左にあるもの。

３．　形態素解析

３．１　単語分割

このパーサはベタ書きの入力文を文末からスキャンしていくが，その際，次に来るべき単語は各時点での入力の先頭部分を調べ，ある条件を満たす文字列が単語として抽出される。この操作とＡＴＮパーサの親和性を図るため，ＣＡＴアクションを以下のように拡張した。まず，例で示す。



図に示すような時点で，（ＣＡＴ　Ｎ）（名詞を要求）なるアクションに出くわした場合，このアクションは

(EITH (SCAN-WORD N　島が鬼)(SCAN-WORD N 島))

なる制御構造で置き換えられる。すなわち，入力文から名詞として認定される文字列をある制限内

（通常は辞書単語中の最大長）で全て拾い出し，ＣＡＴアクションの対象の候補として，選択の余地を残して列挙する。

SCAN-WORDは新しく定義されたＡＴＮアクションであり，入力文から指定された文字列を取りはずし解析を一歩先へ進めるものである。このアクションは fail することはない。

一般に（ＣＡＴ　ｃ）が実行されると，パーサは入力列からカテゴリーが　ｃ　であるような単語 $w_1, w_2, \cdots, w_n$ を拾い出し，

```
(EITH (SCAN-WORD c w1)
      (SCAN-WORD c w2)
            .
            .
      (SCAN-WORD c wn))
```

を作る。ただし，ｎ＝１の場合は単に(SCAN-WORD c w_1)を戻し，ｎ＝０の場合は（ＣＡＴ　ｃ）は fail する。

通常のＣＡＴアクションは入力列から単語を１個取り去るが，この方式では入力列に変更を加えることはなく，与えられた遷移ネットワークの一部に動的かつ一時的な拡張をおこなう役割を持つ。上の例の場合は，図３で示される。

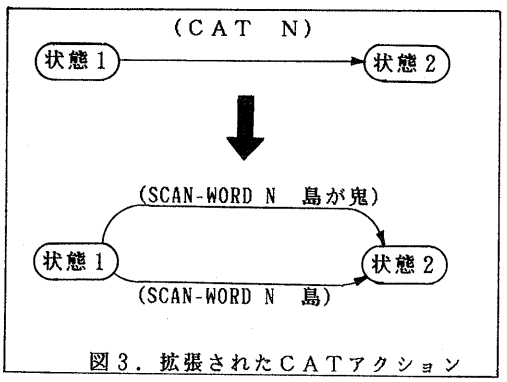


図３．拡張されたＣＡＴアクション

これがなされると，その後は通常のパーサ機能によって解析が進められる。いくつかの候補の中から正しいものを選択する作業もパーサによって自動的におこなわれる。ただし，パーサは構文および意味の解析も同時におこなっているので，制約条件が格段に厳しくなり，不適切な選択枝にとらわれる可能性はずっと小さくなる。この点が，段階的文解析に比べて大きな利点と言える。

３．２　語尾変化および語の接続

前節では単語の形状が変化しない場合を想定し，従って候補単語（$w_1, w_2, \cdots$）を認識するのに辞書の参照だけでよかった。しかし，実際には多くの単語は語形が変化し，また語間の接続条件なども考慮しなければならないので，単語の認識に際しては辞書の参照に加え，何らかの手続きが必要になる。なかでも特に重要なのは用言，助動詞の活用および隣接語との接続関係であるが，以下にその扱い方を略述する。

用言の活用には（場合によっては複数個の）助動詞が後続し，それらの助動詞も活用する。また，最後に終助詞を伴うこともしばしばである。さらに，接続する語の間には，それらの品詞および活用形に関する規則が存在する。我々は，入力列スキャンの過程においてこの接続条件を順次受け渡していく方式をとった。すなわち，ある単語を認識することに成功すると，その単語の次（ここでは文を逆方向にスキャンしているので，実際は“その前”）に来ることが可能な品詞及びおよび活用形は何であるかを決め，その情報を基に次の単語の抽出をおこなう。例を挙げると，

－－－－行かせます
　　③　②　①

なる場合，まず　①“ます”が終止形の助動詞であることが認定されると，辞書より　①⟶②の接続条件が，「動詞または動詞型助動詞の連用形」なることが判り，続けてスキャンし，　②“せ”が（辞書より）助動詞“せる”の未然形であることが判り，これを受理する。次に，やはり辞書より，　②⟶③の接続条件が，「動詞の未然形」なることが判り，スキャンを続けて“行か”が動詞“行く”の未然形であることを見出す。語尾変化について，先ず助動詞に関しては終止形および活用形のそれぞれを辞書エントリとして登録しておく（例えば，終止形“せる”，未然形／連用形“せ”，仮定形“せれ”，命令形“せよ”）。接続条件は終止形のみに与えておく。現在の辞書中の（活用形も含めた）助動詞のエントリ数は約８０であり，これで日常的な文については，大部分カバーできるものと思われる。

動詞の語尾処理については，辞書に頼らず専用の関数を用い，最初に出会う文字を手掛かりにして，可能な終止形を生成する。辞書には終止形の

みが登録されている。上の例ならば，“か”を未然形の最終文字とするような動詞の終止形を，辞書を参照しながらスキャンを続けることにより，“行く”が生成される。

動詞と助動詞の接続は文法において，

(SEQ (OPT* (CAT AUX))(CAT V))　(AUX:助動詞)

のように陽に示すようにしている。従って，前章で述べたＣＡＴアクションの機能により，上記の語認定において複数の可能性が発生しても，それらを全て保存出来る。しかし，助動詞についてはこれを厳密におこなうことはあまりにも負荷が大きすぎるようなので，最長一致のもののみを残すようにしている。これで，実用的には問題ないように思われる。動詞については，全ての選択枝を保存するようにしている。

4．意味処理

我々の当面の目的は厳密な意味解析をおこなうことではなく，むしろ意味的制約を課して構文解析および形態素解析のあいまいさを除去しようとするものである。従って，意味処理については，当面以下のような割合単純なやり方を試みた。しかし，パーサ自体の構造は意味処理機能と独立しているので，より厳密な意味処理機構を組み込むことは充分可能である。

4．1　意味構造

このパーサは，文法中に意味規則が混入されていても他の構文規則（通常のＡＴＮアクション）と区別されることとなく解析が進められる。従って，適当な意味解析機能を定義することにより，構文解析（および形態素解析）と平行して意味処理を進めることが可能である。我々のとった方式は以下のようである：

a）構文解析木の生成と平行して，必要に応じフレーム型の意味構造を発生させる。

b）やはり構文解析に平行して，それらフレームのスロットを辞書情報をもとに埋めていく。これがうまくいけば先へ進めるが，失敗すると全体としてバックトラックを起こす。

意味構造としては格文法形式を基範とした。すなわち，構文解析中にＳノードが発生すると，それに対して Eventフレームと呼ぶものを生成し，動詞（形容詞，形容動詞）およびそれに付随する名詞句に対応する識別子をスロット値として挿入する。また，名詞句に対しても意味構造 Object フレームを発生させるが，これは現在のところ解析に重要な働きを持っていない。例を図４に示す。

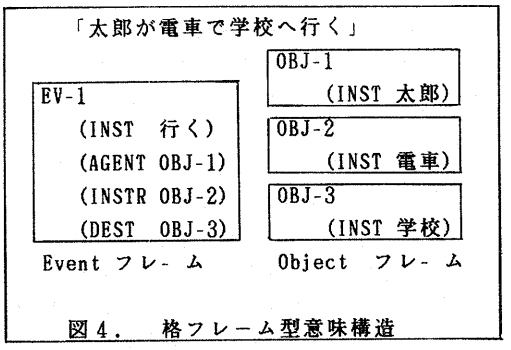


図４．　格フレーム型意味構造

Event フレームへのスロット値の充塡は，動詞および名詞の辞書情報を参照しておこなわれる。動詞項目の意味部には，

「格助詞　スロット名　名詞意味素性」

なる形式の３つ組の可能なものを列挙しておく。また，名詞には意味素性を付加しておく。ただし，動詞において指定されている意味素性と名詞の意味素性の比較は，名詞の意味階層データを用いておこなう。辞書記述の例を図５に示す。

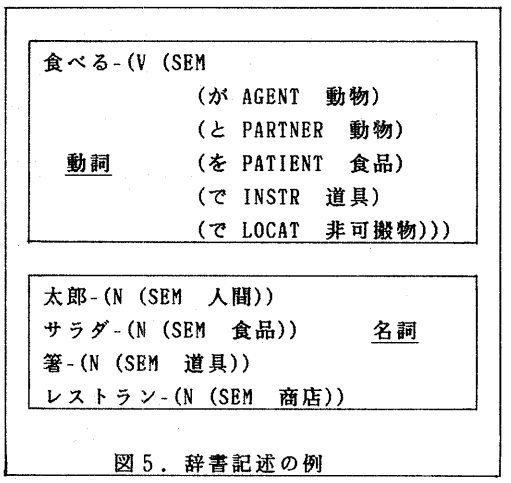


図５．辞書記述の例

4．2　意味処理を含む文法記述

上記のような意味構造を操作するための関数の主要なものを以下に示す。

(DEF-FRAME x)：x　で示される（構文木の）ノードに対してフレームを定義する。

(INIT-FRAME x y)：xで示されるフレームに初期値としてy で示される単語を(INST y)のスロットとしてセットする。

(FILL-SLOT x y):x で示されるEvent フレームに，スロットとスロット値を挿入する。スロット名（役割）とスロット値は，y で示される役割句（名詞句＋格助詞）およびフレームに初期値として入っている動詞より決められる。すなわち，辞書より，動詞の意味情報，名詞の意味素性，格助詞の関係を調べ，役割スロット名を決める。決定不能の場合，あるいは決定されていてもそのスロットが既に使用されている場合にはfail値を戻す。

(FILL-SLOT1 x y)：これはFILL-SLOT と似ているが，役割句を扱うものではなく，埋込み文により修飾されている名詞の（その埋込み文における）役割を決め，そのスロットを挿入する。この場合は格助詞がないので，名詞の意味素性のみを手掛りにして役割を決定する。決定不能の場合あるいは決定できてもそのスロットが既に使用されている場合にはfail値を戻す。

以上のような意味処理関数を用い，図１の文法を拡張すると図６のようになる。なお，ここでは動詞と助動詞の列を扱うためにVP（動詞句）カテゴリをも導入している。

図６の説明：

a）S およびNPに対してフレームを定義している。

b）　①：現在作成中のS ノードのフレームに対して，直前に作成されたVPノードの最初の動詞をINSTとしてセットする。

c）　②：任意個数受理される役割句（RP）のそれぞれ（$LAST）　に対する〔役割スロット　スロット値〕の組を，現在作成中の S　ノードのフレームに挿入する。

d）　③：現在作成中のＮＰのフレームに対して直前に受理された名詞をINSTとしてセットする。

e）　④：現在作成中のＮＰのフレームの名詞について，直前に受理されたＳノードのフレームに〔役割スロット　スロット値〕の組として挿入する。

f）ＶＰは０個または任意個の助動詞に続いて動詞が来る。

```
S: (SEQ (DEF-FRAME ($$ S))
        (PARSE VP)
        (INIT-FRAME (THE 'REF OF ($$ S))
                    (GET-WORD (THE-FIRST 'V OF $LAST)))
        (OPT* (SEQ (PARSE RP) (FILL-SLOT (THE 'REF OF ($$ S)) $LAST)))

VP: (SEQ (OPT* (CAT AUX)) (CAT V))

RP: (SEQ (CAT P)(PARSE NP))

NP: (SEQ (DEF-FRAME ($$ NP))
         (CAT N)
         (INIT-FRAME (THE 'REF OF ($$ NP)) (GET-WORD $LAST))
         (OPT (SEQ (PARSE S)
                   (FILL-SLOT1 (THE  'REF OF $LAST)) (THE 'REF OF ($$ NP)))))
```

①　②　③　④

図６．意味処理を含む文法例

5．　文解析実験

パーサは，与えられた文法および辞書を参照しながら，入力文を文末からスキャンし，構文木と意味構造をトップダウン的に作成していく。図6の文法の場合の解析過程を図7に示す。

図は名詞“花子”をスキャンし終わった時点でのスナップショットである。（実線は完成した枝，点線は未完成の枝を示す。）この時点までにS_1ノードが生成され，EV-1 フレームに動詞句 VP_2 の動詞“行く”が挿入されている。名詞“家”はスキャンされたが，これには埋込み文の修飾があるので名詞句 NP_8 はまだ完成していない。パーサは埋込み文 S_{10}を作成中であり，EV-2フレームへのスロット充填をおこなっている。“花子が”はうまく入った(FILL-SLOT) が，次に“自転車で”を EV-2 に挿入しようとして失敗する（動詞“住む”に対しては，助詞“で”に対応する役割スロットがない）。そこで，“花子”までで S_{10}の作成を完了し，次に“家”を EV-2 に挿入する(FILL-SLOT1)ことを試み，これに成功し，EV-2も完了する。パーサは S_1 および EV-1 の作成に戻り，“・・家へ”，“自転車で”，“太郎は”，の役割句を次々に処理し，全て成功する。

上の文法はごく単純なものであるが，実験ではもう少し実用的な文法を試作し，さまざまなタイプの文を試している。以下はこの実験用文法で解析できる文の例である。

①「太郎は日本人でしょうか」
　＊〔名詞句＋は＋名詞句＋だ〕型の文。

②「太郎は勉強した」，「太郎は勉強をした」
　＊サ変動詞の処理例。

③「鬼が島から来なかった」
　＊意味的にあいまい。構文木は2個できる。

④「太郎は京都に住む花子が書いた手紙を読んだ」
　＊3重の埋込み文。

⑤「今年の冬はとても寒そうだ」
　＊形容詞に対しては，動詞と同じ形式の Event フレームを作る。

⑥「太郎は花子に料亭で食べた美味しい刺身の話をした」
　＊形容詞の名詞修飾は埋込み文として扱われる。従ってこの場合，「刺身」は2個の埋込み文により並列に修飾されている。

⑦「太郎は花子の台所のやかんで湯を沸かした」
　＊助詞「の」の扱いは不完全であるが，形式的にある程度処理できる。

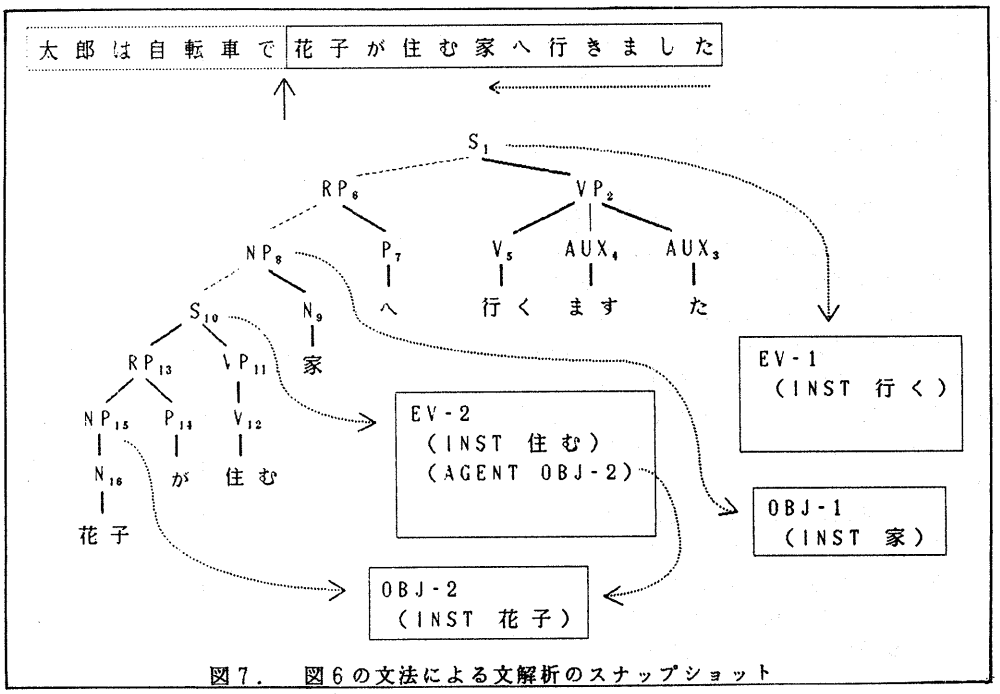


図7．　図6の文法による文解析のスナップショット

⑧「太郎は朝刊と夕刊を読んだ」
＊並列助詞「と」の例。
⑨「太郎は花子と秋子に会いに行きました」
＊接続助詞「に」の例。しかし，「と」に関して意味的あいまいさがあり，解析木は2個できる。
⑩「太郎は朝ご飯を食べて学校へ行きます」
＊「朝ご飯」で分かちのあいまいさが残り，解析木が2個できる。

この範囲の文型では，当初心配されたような探索スペースの爆発的な膨張は起こっておらず，バックトラックの回数も許容できる程度のものにおさえられている。ちなみに，これらの実験は通常の16ビットパソコン上のＬＩＳＰでおこなったものである。辞書格納の問題は残るが，その程度の計算機設備でも，問題の範囲を限定すればある程度の実用化が出来るものと予想される。

6．まとめ

ＡＴＮ方式を一部拡張することにより，形態素，構文，意味を同時解析する日本語用パーサを割合簡単に実現できることがわかった。また，入力文を逆走査することにより，左回帰の困難が回避でき，バックトラックによる時間浪費を大巾に減少できることもわかった。上で示したように，3種の解析過程は同時平行的におこなわれるが，プログラム構造の面から見ると，これらはほぼ完全に分離されているので，日本文解析システムの構築に際して種々の実験が比較的容易におこなうことができる。

現在までの実験結果から，形態素解析，構文解析については，ほぼこのままでよいと思われる。一方，意味処理規則については，その記述方式が多分に煩雑であり，種々の複雑な文型を扱うためには，種々の意味処理関数を個々に作成する必要が起こる。このようなアドホック性を回避するため，統一的な意味規則記述方式を設定する必要があると思われ，これが差し当たっての課題となっている。

パージングの効率については，現在のところ他方式との比較をおこなっていないので，断定できないが，少なくとも特に劣ることはないように思われる。また，ここでは述べなかったが，EITH，OPT などの分岐制御については1語先読み機構も用意されているので，これを使用して効率を上げることも可能となっている。

最後にあたり，この研究に関して貴重な助言を頂いた京都大学電気第2教室の長尾真教授ならびに辻井潤一助教授に謝意を表します。

参考文献

〔1〕Charniak,E. and McDermott,D.,"Introduction to Artificial Intelligence",Addison-Wesley(1985).