自然言語処理 84－17
(1991. 7. 19)

日英機械翻訳システム JETS における日本語解析

荻野 紫穂
丸山 宏

日本アイ・ビー・エム（株） 東京基礎研究所

平成3年7月19日

Abstract

日英機械翻訳システム JETS の日本語解析は、(1) 形態素解析、(2) 係り受け解析、(3) 格解析の3ステップからなる。正規文法に基づく形態素解析は、選言を含む素性構造を出力し、係り受け解析に渡す。制約依存文法に基づく係り受け解析は、各選言の選択肢を絞り込んでいき、曖昧な係り受け関係を決定して格解析に渡す。格解析は、各ノード間の関係を決定し、必要に応じて変形操作を行ないながら、JCS (Japanese Cannonical Structure) を出力する。この JCS が、日本語解析部の出力となる。

本稿では、この三つの解析段階について、それぞれシステムの設計方針と文法体系の両面から、その概要と特徴を述べる。

Japanese Analysis Part of JETS, the Japanese-to-English Translation System

Shiho Ogino
Hiroshi Maruyama

Tokyo Research Laboratory, IBM Japan.
5-19, Sahban-cho, Chiyoda-ku, Tokyo 102, Japan

Abstract

The Japanese analysis part of JETS, the Japanese-to-English Translation System, consists of three substages; morphological analysis, dependency analysis, and case analysis. Morphological analysis is based on regular grammar. In this substage, the system divides an input sentence into words and produces a feature structure with choice points. In dependency analysis, one modifiee is selected from several candidates for each phrase. Dependency analysis rules are written based on Constraint Dependent Grammar. Case analysis is based on tranformation rules. This module determines grammatical relations between modifiers and modifiees and makes an abstract syntactic structure called JCS (Japanese Canonical Structure). In this paper, we describe the design principles of the system for each substage.

1 はじめに

機械による自動翻訳は実用化の時代に入ったと言われるが、依然多くの課題が残っている。我々は機械翻訳システム JETS の開発を通して、短期的にはユーザーの介入を許すことによって用途を限った実用化を狙い、また、長期的には個々の言語処理モジュールの熟成によって自然言語処理の基礎技術の蓄積を狙っている。

本論文では、JETS の解析部の設計について、システムと文法体系の両面から議論する。次節では、システムの設計について述べ、3 節で、文法の体系について議論する。4 節ではその実現について簡単に触れる。

2 システムの設計

JETS は構文トランスファーに基づく機械翻訳システムであり、その解析部の出力は JCS (Japanese Canonical Structure) と呼ばれる抽象的な構文構造である。JCS は表記や語順、態 (Voice) などの揺れを吸収した、標準的な構文構造であり、基本的には自立語とそれらの間の文法的関係からなるグラフである。文法的関係としては、ガ、ヲ、ニなどの格関係、場所、時などの周辺格、結果、原因、状態などのより意味に近い関係などが定義されている。

べた書きの日本語文から JCS を作るためには、語と語、文節と文節の境界を見つけ出し、それらの間の係り受けと、その文法関係を決定しなければならない。これらを単一の処理で行なうことには、文法記述体系が一つで済むなど、それなりのメリットがあるが、我々は以下に示すような理由から、図 1のような、形態素解析、係り受け解析、格解析の 3 つのモジュールからなるシステム構成とした。

- それぞれのモジュール毎に、記述が簡潔で、かつ、効率的な解析を可能にする文法記述体系を用いたい。
- 形態素解析は日本語処理においては比較的成功している領域であり、日本語構成支援、文献検索、OCR 後処理などのために単独で用いられることが多い。
- 文節間の係り受けを指示することは、一般の日本人ユーザーにとって比較的容易であるが、係り受けの文法的関係を決定することは困難であると思われるので、ユーザーの介入を許す係り受け解析と介入を許さない格解析は分離した方がよい。

以下、それぞれのモジュールの文法記述体系について述べる。

2.1 形態素解析

日本語の形態素解析の文法体系は、正規文法と等価なものが多い。[6] における文節文法は正規文法であることが示されているし、[2] や [8] 、[9] も隣接する二単語間の接続ルールだけに基づいて文節の適格性が定義されているので、これらも正規文法によるものと言ってよい。正規文法に基づく形態素解析は、単語の前後関係という非常に狭いスコープで文法を記述しなければならない反面、入力文の長さに比例する、高速な解析が可能である。

一方、[5] 、[1] などに見られるように、構文解析で用いられている単一化文法によって形態素解析を行なってしまうものもある。素性構造の単一化によって高い記述力をもつが、形態素解析結果が多くの曖昧さを持つ場合、それらを効率良く表現するのが困難である。

JETS における形態素解析の文法ルールは、正規文法のルールに素性構造とコストを付加したものである。素性構造は解析の失敗／成功には関与せず、解析が成功した段階で、解析結果を生成するのに用いられる。一方、コストは複数の解析結果のうちよりもっともらしいものを選び出すのに使われる。正規文法と等価な他の形態素解析文法の定式化 ([6, 9]) と比較して、

- 素性構造により、様々な文法的情報を結果として出力可能
- 空ストリングを単語として認めるため、文法記述が柔軟
- ルール毎にコストを付けられるので、細かいチューニングが容易

という特徴をもつ。また、文法の生成力としては正規文法の範疇を越えていないので、高速な解析が可能である。

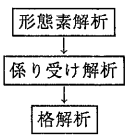

図 1: システム構成

```
「わたしは魚が食べたい。」
{[id=0,words={[id=0,syn={%[str='私',pos=代名詞],[str='渡し',pos=名詞]%},
                str='わたし']
              [id=1,syn=[str='は',pos=係助詞],str='は']},
  str="私は",modifiee={%1,2%}],
 [id=1,words={[id=0,syn=[str='魚',pos=名詞],str=魚],
              [id=1,syn=[str='が',pos=格助詞ガ],str=ガ]},
  str='魚が',modifiee=2],
 [id=2,words={[id=0,syn=[str='食べ',pos=動詞],str='食べ'],
              [id=1,syn=[str='た',pos=助動詞],str='た',fe={hope}],
              [id=2,syn=[str='い',pos=活用語尾],str='い'],
              [id=3,syn=[str='。',pos=句点],str='。']},
  str='食べたい。']}
```

図 2: 形態素解析の出力

```
define nocross(X,Y)
begin
     addc (X.pid < Y.pid & Y.pid < X?modifiee) =>
          Y?modifiee <= X?modifiee;
end;
```

図 3: Jaunt のルール

形態素解析の出力である素性構造を図 2に示す。形態素解析で決定できない係り先 (modifiee) や、語彙の曖昧さが、素性構造における選言（図では {%...%} で括られている部分）として表されている。

2.2 係り受け解析

形態素解析に比べ、日本語の係り受け解析には定まった方式がない。拡張文脈自由文法に基づくもの [11][1] などがあるが、明確な文法理論に則ったものは少ないといえる。

JETS の係り受け解析は、制約依存文法 [13] に基づく。制約依存文法は、文の解析を有限領域上の制約充足問題として捉える体系である。制約依存文法の文法ルールは、

$$\forall x, y : \text{文節}; P(x, y, mod(x), mod(y))$$

という形で与えられる。ここで、$mod(x)$ 、$mod(y)$ はそれぞれ文節 x 、yの係り先を表す。すなわち、任意の 2 つの文節の間の係り受けに関する制約を記述する。例えば、係り受けが交差しないという文法的制約は

$$\forall x, y : \text{文節}; pos(x) < pos(Y) \& pos(y) < pos(mod(x)) \Rightarrow pos(mod(y)) \leq pos(mod(x))$$

と書くことができる。また、このルールを緩めれば、「映画を新宿に見に行こう」のような係り受けが交差する文を解析することが出来る。一般に、このように係り受けが交差するような文の解析は、文脈自由文法に基づく文法体系では困難である。

JETS では、選言付き素性構造上の制約ソルバー Jaunt[7] を用い、制約依存文法のルールを、図 3のように記述する。ルールは図 2に見られるような入力素性構造の選言の値を決定する。

制約依存文法の特徴の一つは、解析の途中結果を（それがどんなに曖昧さを含んでいても）コンパクトな一つのデータ構造で表現できることである [12] 。また、解析の途中でユーザーからの係り受けの指示を一時的な制約として取り込むことも可能である [14] 。

2.3 格解析

格解析は、係り受けの決定した文節構造から JCS を作り出す。後に述べるように、格解析は構造の変形操作を含むので、一般的な定式化は難しい。現在この部分はパターンマッチングを含む書き換え規則の繰り返しの適用によって実現されている。

```
START -> 'な' [pos=keiyoushi]
         'さ' [pos=kantou_moji]
         'そう' [pos=jodoushi] 助動詞ソウ;
```

図 4: 複数要素からなる非終端記号

```
START -> '机' [pos=meishi] 名詞 cost=1000;
名詞 -> 'だろ' [pos=jodoushi] 未然形;
名詞 -> 'だっ' [pos=jodoushi] 連用形;
```

図 5: 形態素解析のルール例 1

3 文法体系

3.1 形態素解析

規則の形式

形態素解析の文法ルールは、具体的には図 6のように記述される。一般の形式は、

```
遷移名 -> {ストリング [素性構造]}* [遷移名] [コスト];
```

である。表記のストリングと素性構造のペアを連ねたもの（素性構造がなくてもよい）を終端記号と見なすため、例えば「なさそう」「よさそう」の「さ」のように、ごく例外的な接続について、遷移（あるいは接続条件）を増やさないで、形態素的複数語として扱うことが可能である（図 4)。遷移を増やしても処理は可能だが、多少の読みにくさはまぬがれない。

直観的に言って、一つの規則の左辺はそこに現れている終端記号の左連接属性、右辺の非終端記号は終端記号の右連接属性と見ることが出来る。つまり、最も原始的には、一つの表記エントリーがその前接語と後接語とを合わせた数だけ規則を持っていることになる。この原始的な規則のままでは、可読性が非常に悪いが、空ストリングを使うと、連接属性を分かりやすくまとめていくことが出来る。

例えば、図 5は原始的なルールの並びだが、'だろ''だっ'の間の関係がわかりづらい。これを空ストリングを使って図 6のように書き換えると、二つの間の左連接属性である'助動詞ダ'が明示できる。この際、空ストリングは助動詞ダの語幹と見ることが出来る。遷移の種類は増やしていけるので、連接属性を細分するのも容易であり、連接属性の数に拘らずに素性構造を作ることができる。(規則の表記は多少遠回りなものにはなる。) 空ストリングを使った規則を展開すれば、原始規則と同じ形の規則ができる。現在、活用自立語回り中心の規則が書かれているが、遷移の数は二百強であり、[16] などの連接属性数と比べて、さほど多い数ではない。

規則は必ずしも文節に頼らなくて良い。現在の JETS は、係り受け解析で文節を使用するため、形態素解析でも便宜的に文節を使っている。

素性構造

素性構造は、

```
[文節情報=[... 語情報=[... 構文情報=[... 意味情報=[...]]]]]
```

のような入れ子構造になっている。語の情報の中では、辞書など外部データベースから得た情報と、解析中に得た情報とは別々に保存されている。図 2では、words の要素がそれぞれの文節に含まれる形態素だが、その中の syn の値が外部データベースからの情報、その他の部分が解析中に得られる情報である。図 4の「さ」のように、辞書に存在しない語については、辞書情報が存在するべき箇所に、マークが付けられる。

```
START -> '机' [pos=meishi] 名詞 cost=1000;
名詞 -> '' [pos=jodoushi] 助動詞ダ;
助動詞ダ -> 'だろ' 未然形;
助動詞ダ -> 'だっ' 連用形;
```

図 6: 形態素解析のルール例 2

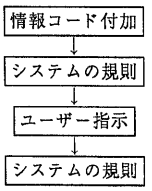

図 7: 係り受け解析の流れ

現在、形態素解析中に素性構造に付加される情報は、形態素の種類と性質情報が、主なものである。図 2では、feの値が性質情報に当たる。現在、活用語を中心に、三十強の形態素の種類をつけている。性質情報は、意味・構文双方の機能が一つの属性に入れられているため、可読性があまり良くない。これらは今後別の属性に分けて保存する予定である。これらの情報は、後段階で参照される。

遷移が連接属性を担っているので、素性構造は連接情報を反映している必要はない。現在は、読みやすさ、デバッグのしやすさなどを考え、大まかな連接属性を素性構造に付け加えているが、他の属性と同じく、素性構造中の連接属性は形態素解析自体の中では使われていない。

コスト

コストは、現在ではまだごく単純な差だけがついている。「付属語は軽く、自立語は重く」という大まかなものの他、活用語語幹と活用語のように、お互いが密接な接続関係を持つ規則はコストを軽くしている。品詞毎にコストを与える [8] などに比べて細かなチューニングが可能である一方、なぜ解析結果がこうなったのかという理由が分かりにくい、ad hoc なものになるおそれもあるので、全体的なコストのチューニングは、試行錯誤の段階にある。

3.2 係り受け解析

係り受け解析は、主に制約を使って、各文節の係り先の選択肢を絞り込んでいく働きを持つ。広い意味での構文解析段階用の解析文法は、適用範囲を広げようとすると解釈の曖昧さが増えやすく、曖昧さを減らそうとすると適用範囲が狭まりやすい。JETS はユーザーの介入を制約として取り込んで曖昧さを減らすことが出来るので、係り受け解析の文法は、適用範囲を広く持つことに重点を置く。このため、曖昧さを極端に減らすための不安定な選択制約などに依存しない。

ルールは [17] と同様に、「不可能な係り受けを発見する」という方向で書かれていて、一つの文節の係り先をローカルに決定はしない。但し、[17] と違い、一つの文節の係り先は唯一に定まるとする。他に意味的な関係がある場合は、格解析段階でその情報が構造に加えられる。

係り受け解析の流れを図 7に示す。最終的に係り受け関係は一意に決定されて、その依存構造が格解析に渡される。

解析の概要

係り受け解析は、ユーザーからの係り受け指示を容易にするために、文節を基本単位にしている。文節は、自立語は、名詞類（代名詞などを含む）、動詞、形容詞、形容動詞、副詞、接続詞類、連体詞のどれかを必ず一語含むものとする。原則的には、一文節の中には、他の文節の係り先になるものが一形態素しかないか、形態素の一つの連なりが他の文節の係り先になっていて、文節中の他の形態素はそれに関係していないかの、どちらかなのが望ましい。ただし、例えば名詞と助動詞のように、二つを離してしまうと単独の意味が分かりにくくなるものは、二つを連ねて一文節とする (図 8)。また、ある文節はほぼ直後の文節に係ることが分かっていて、その二つの文節を連文節で表しても他への影響が少ない場合にはその隣り合う文節を一文節とする。例えば、「走ること」という表現は、係り受けから考えると、二文節である方が処理しやすい。しかし、(もし「こと」がいわゆる形式名詞なら)「走る」は「こと」に係ることが推定できるので、この表現を一文節とする。これは、主に係り受け指示をするユーザーの煩わしさを軽減する目的によるもので、文法的な理由によるものではない。

各文節は図 2のように、解析に使われる情報を素性構造の形で持っている。係り先の候補も、選言として素性構造に含まれる。解析は、ユーザーが係り受け指示をシステムに与えやすくするため、緩い規則に制約を加えて、選択肢を徐々に絞っていく方法を採る。解析中に矛盾が生じた場合、システムはメッセージを表示すると共に、矛盾が生じる直前の素性構造を返す。

修飾文節が被修飾文節のどの要素を修飾しているかの判断は、係り受け解析で下すと便利だが、現在は格解析が必要に応じてその判断を下している。

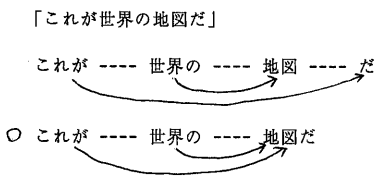

図 8: 名詞と助動詞の文節

規則の種類

係り受け解析の規則は、図 3のような論理式で定義される。規則は

- 係り先以外の素性構造を変更し間接的に係り受け関係を決定するもの
- 直接係り受けの是非を扱うもの

の二種にほぼ大別される。現在の係り受け解析規則の約八分の一が前者に当たる。

これらとは別に、最終規則が定義されている。最終規則は、最後まで残った曖昧な係り受け関係を、意味素性と格パターンとに基づいて決定する。

辞書情報

係り受け解析及び格解析用の辞書情報には、原則的に特定の分野に依存しない、格パターン、正表記（見出し)、品詞、意味素性などが含まれている。格パターンは活用自立語に付けられている。助動詞レルのように、表層で要求される格パターンを変化させるものは、変化のパターンのコードを持っている。格マーカーについては次節で述べる。

名詞は [11] の 9 の意味素性にワイルドカードの役目をする素性を加えた 10 の素性を持っている。。これは名詞が格要素になり得る時の、述語側のフレームとのマッチングを使った選好に主に使われる。この他、構文的情報として、程度副詞を受け得るかどうか、助詞ニをとって副詞的連用修飾となり得るかどうかなど、七種類のコードを持っている。

動詞は格フレームの他、七種類のアスペクト情報を持っている。また、「～という」の「いう」や「～による」の「よる」のように格助詞的に使われて、本来の動詞的な働きをしない動詞には、そのコードがつけられている。

形容詞は格フレームの他、[18] のような分類をつけている。

副詞は程度副詞になるかどうか、呼応があるかどうかなど、六種類のコードを持っている。これは主に係り受け解析で使われる。接続詞のコードもほぼ副詞と同じである。

連体詞は、連用修飾を受けるかどうかのコードを持っている。これは「小さな」「大きな」など形容詞と同じ働きをするものに付けられる。

解析用情報

文節は、文節内に含まれる各語の持つ情報の他に、文節自体の解析用情報コードを持つ。特に連用修飾機能について、名詞類と副詞は、程度副詞になり得るかどうかなど、十数種類の情報を持っている。これらと格関係の情報・語の種類の情報などを使ってパターンマッチをした結果が、文節自体の情報コードに記される。これらは、係り受けの細かい分類を示しており、一種の係り受けコードとも言える。現在、約七十種類の情報コードがある。これは確定的な種類ではなく、係り受け解析規則の開発に従って作っていったもので、コード全体の見通しが悪く、可読性も良くないため、整理再分類を予定している。

この情報コードは、大まかに言って、

- その文節内で判断できるコード
- 他の文節との関わりで判断するコード

の二種に分けられる。例を挙げれば、前者は体言を文節内に持つかどうか、後者は文節の位置などである。情報コードの約三分の二が、前者である。

今の JETS では、係り受け解析をする前段階に、情報コードをつけるモジュールが設けられている。特に後者の情報を付け直すのは、負担が重くなりやすく、係り受け解析に持ち込みづらい。しかし、そのままでは、情報コードを変えるような選択の結果を、素性構造に反映していくのが多少繁雑なので、徐々に、係り受け解析の中に情報コード付加の判断を移していく予定である。

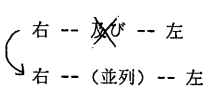

図 9: ノードの削除

3.3 格解析

格解析は、格関係その他、文節間の関係を決定するのが主な役割だが、ノード内の情報を変更し、構造を単純化するなどの機能もある。はっきりした方略はまだなく、ad hoc なプログラミングによってほとんどをまかなっている。

Japanese Canonical Structure

JCS(Japanese Canonical Structure) は、ノードとノード間の関係からなる構文構造である。入力された日本語文の全ての意味を反映した構造というよりは、むしろ、翻訳用に一つの代表的な構造に単純化した構造である。

係り受け解析では、連文節も一文節にしているので、格解析部のノードは係り受け解析から渡された文節とは別概念である。ノードはいわゆる自立語を含んでいなくとも良い。例えば、使役の助動詞「せる」など、格関係を自ら支配するものは、単独で一つのノードになることが出来る。また、並列名詞句は、各々をまとめた一つのノードがある方が、処理しやすい。このような場合は、概念的なダミーノードを認めている。

各ノードは、それぞれが持つ語の情報と、ノード自体の情報を持つ。但し、語の情報は、必ずしも日本語文入力時の語と 1 対 1 に対応しているとは限らない（次項参照）。

ノード間の関係は一種の標準表現であり、いくつかの異なった日本語表現を一つの関係にまとめる働きを持つ。例えば、「右と左」「右及び左」は双方が同じ関係に落とされ、翻訳の後段階のトランスファーは、この二つの差を考慮しない。現在は非常に単純な関係だけが定められている。大まかにいって、文と文、格要素と述語、連用修飾と被連用修飾、連体修飾と被連体修飾、名詞句と名詞句（並列）、その他、の六組のノードのペアについて、それぞれ関係が別体系で定義されている。

格マーカーは、ガ・ガ 2・ヲ・ニ・ヘ・ト・ヨリ・カラ・デの九種類を使用している。格はほぼ表層格に一致しており、[19] のような深層格へのマッピングは行なわない。これは以下の理由による。

- 各々の格要素の意味素性で区別のつく格フレームについては、翻訳においては、意味素性とマーカーとの組み合わせで訳語選択が可能である。
- 意味素性で区別がつかないフレーム同士は、そのままでは、深層格へのマッピングも困難である。よって、フレーム毎に訳語が決まっているならば、マッピングをせず、頻度などによって、訳語を決定する。

この構造は、原則的には順番の概念を持たない。しかし、語の順番が英語生成の質に関係することも多いので、情報の一つとして順番を残している。

処理の概要

ノード情報の設定部分は、独立して存在しているものではなく、他の解析と相互に実行される。時制の情報などはこの部分で設定される。

格関係解析部分では、まず、有形の格マーカーが仲介する格関係が決定される。その後、有形の格マーカーを持たない文節についても受け側との格関係が推定されるが、この精度は多分に辞書情報の（性質も含む）質に左右されやすい。

他の関係解析部分は、関係を決定していく際に、必要に応じてノードそのものやノードが含む語の情報をを削る機能を持つ。接続詞のように、それ自身が意味を持つよりは、他のノード同士の関係を表している語が一語でノードを作っている場合、その語の機能は他のノード間の関係に写すことによって残されるが、ノード自体が残っている必要はない。例えば、「右及び左」と言った場合、「及び」は接続詞として一ノードを形成するが、「右」と「左」とが並列の関係であるという情報が保存されれば、「及び」自体がノードとして残っている必要はない。このような場合にはこのノードを削ることによって、後の処理の重複を避けている (図 9)。また、翻訳向きの構造にするため、係り受け構造を変更したり付属語を追加したりする機能も含まれている。

この解析部はまだ非常に原始的なものであるため、今後、ノード間の関係の整理を中心に、拡張改良する予定である。

モジュール	言語	サイズ (LOC)
形態素解析	COB[15]	4,000
係り受け解析	Jaunt[7]	2,000
格解析	Lisp	4,000

図 10: 各モジュールのサイズ

4 実現

機械翻訳システム JETS はクライアント–サーバーモデルに基づく分散アプリケーションとして LAN 上に実現されている。すなわち、翻訳を行なうサーバーは Unix ワークステーション上に、また、ユーザーインターフェースとなるクライアントはパソコン上で稼働する。各モジュールのコードサイズは図 10の通りである。

5 まとめ

JETS の現在の日本語解析部の設計について、システムと文法体系の両面にわたって述べた。

日本語解析は形態素解析、係り受け解析、格解析の三段階に分かれているが、特に格解析部分は ad hoc なプログラミングによっている。今後は格解析部分を中心に、構造的な解析の改良をするとともに、後段階に役立つ情報をより早い段階で付与していく方法を開発していく予定である。

参考文献

[1] 加藤, 小暮, 1987, "素性構造の単一化手法の効率," 自然言語処理研究会資料 NL64.

[2] 坂本, 1983, "日本語形態素解析の基本設計," 自然言語処理研究会資料 38.

[3] 杉村, 赤坂, 久保, 1988 "論理型形態素解析 LAX," *Proc. of The Logic Programming Conference '88.*

[4] 田中, 1989, "自然言語解析の基礎," 産業図書.

[5] Tomita, M(ed.), Mitamura, T, Musha, H, Kee, M, 1988, "The Generalized LR Parser/Compiler Version 8.1: User's Guide," *CMU-CMT-88-MEMO.*

[6] 日高, 1989, "自然言語理解の基礎 – 形態論," 情報処理, Vol.30, No.10.

[7] 丸山, 1990, "選言付き素性構造を対象とした制約ソルバー,"

[8] 吉村, 日高, 吉田, 1983, "文節数最小法を用いたべた書き日本語文の形態素解析," 情報処理学会論文誌, Vol.23, No.6.

[9] 吉村, 武内, 津田, 首藤, 1989, "未登録語を含む日本語文の形態素解析," 情報処理学会論文誌, Vol.30 No.3.

[10] 丸山, 1989, "機械翻訳システム JETS における知識の蓄積管理," 文法的知識と意味的知識の蓄積管理シンポジウム論文集.

[11] Maruyama, N, Morohashi, M, Umeda, S, and Sumita, E. 1988, "A Japanese Sentence Analyzer," *IBM Journal of Research and Development,* Vol.32, No.2.

[12] Maruyama, H. 1990, "Structural Disambiguation with Constraint Propagation," *Proc. of Annual Meeting of ACL '90,* Pittsburgh, to appear.

[13] Maruyama, H. 1991, "Constraint Dependency Grammar and Its Weak Generative Capacity," *Advances in Software Science and Technology,* to appear.

[14] Maruyama, H., Watanabe, H., and Ogino, S. 1990, "An Interactive Japanese Parser for Machine Translation," *Proc. of COLING '90,* Helsinki, to appear.

[15] Onodera, T., Kuse, K, and Kamimura, T., 1990, "Increasing Safety and Modularity of C-Based Objects," *Proc. of 3rd International Conference TOOLS 3,* Sydney.

[16] 日本電子化辞書研究所, 1990, "日本語単語辞書" *EDR Technical Report,*TR-018.

[17] 杉村, 三吉, 向井, 1987, "コンストレイントに基づく日本語の係り受け解析," 情報処理学会第 35 回全国大会講演論文集.

[18] 西尾寅弥, 1983, "形容詞の意味、用法の記述的研究," 秀英出版.

[19] 野村, 村木, 1989, "機械翻訳システム PIVOT の日本語格フレームモデル" 情報処理学会第 38 回全国大会講演論文集.