英文における並列関係の検出

武田 紀子†

文章を解析する上で，文中に現れる並列関係を正確に見つけることは，構文の曖昧性を解消するためにも大切なことである．英文中に現れる等位接続詞は，その前後の句を並列につなげるが，句の構造が入れ子になっていたり，多品詞語の品詞の確定ができない等の理由により並列関係の範囲を決めるのは，困難な場合が多く，特に，長い複雑な文においては，より難しい．そこで，語順により各語の構文的役割が決まりやすいという英文の特徴を生かし，まず，英文レベルで，各句のキーとなる語が同じ語か，同様な語かという同値性を比較することにより，並列関係の範囲を決めるという効率よい方法で処理するサブシステムを英日機械翻訳システムの中に実現した．実現にあたって，語の同値性を簡単に比較できるよう文，文法的範疇が入る2つのスタックを用意した．また，範囲決定のための制限（同じ語，同様な語，同じ文法的形態か，等）規則をその優先順位と共に記述するための簡単な記述法を定義し，文法記述者が，システムに最適な制限規則を定義できるようにした．この制限規則は，英日機械翻訳システムとは，独立して記述され，翻訳システムが実現されている言語 Common Lifp に翻訳され，システムの中に組み込まれる．実現されたシステムで，6語から60語まで，平均23語の約300例文の解析を行った結果，約90%の正解率を得ることができた．

Detection of Coordinate Relation in English Sentence

NORIKO TAKEDA†

To eliminate the ambiguity of a sentence, it is important to detect the coordinate relations in a sentence. However it is difficult to detect the coordinate relations in the case when a sentence with coordinate conjunctions has nested phrases or the part of speech of a word is not definite, and depends upon the context. After performing a few close observations of sentences, it is obvious that those phrases with same or paired-word or having same syntactic structure tend to coordinate easily. That means it is possible to detect those phrases with same or paired-word by comparing key words of phrases before and after conjunction using two stacks: grammatical category stack and sentence stack. Therefore, this idea was implemented into the Coordinate Analysis subsystem. In this subsystem, a simple expression for describing restriction rules to analyze coordination is defined. By using this expression, a grammar writer can easily define the rules for restrictions. These rules are then translated to a Lisp program and implemented into an English-Japanese machine translation system. More than 300 sample sentences were tested, and the results showed that about 90 percent of these sentences could be analyzed accurately.

1. はじめに

英日機械翻訳を行う上で，英文解析における曖昧性除去の困難さは，多品詞語の品詞決定，係り受けの曖昧さによるところが多い．英文は，その語順により，語，句の役割が決まることが多いので，句の構造が決定されれば，その後の解析は，文法に沿って，比較的簡単になされる．しかし，文を解析するにあたって，表面上は，同じ形態をしていても，係り受けの異なることがあり，句構造の決定を一意的にすることができない場合が多く，英文解析における課題となっている．

特に，並列関係の解析においては，等位接続詞の前後の句が正しく解析され，並列関係を形成する範囲が正しく決められないと，構文全体に，影響を及ぼす誤りを犯すことになる．等位接続詞は，文法的，意味的，形態的に同じようなもの同士の並列関係を形作るので，接続詞の前後の文を見て，同じようなもの同士を見つければよいが，等位接続詞の解析の段階で，英文解析における係り受け等の曖昧性が除去されず，句の文法的構造の決定がなされていない場合も多く，同じようなもの同士を見つけるのも困難なことが多い．また，“同じようなもの”の定義も曖昧である．

英文に対する並列関係の処理は，ATN における SYSCONJ[1),2)]で，提案された．しかし，このシステ

† 成蹊大学工学部経営工学科
Department of Information Science, Faculty of Engineering, Seikei University

ムは，可能なすべての組み合わせに関して解析を行った上処理するので，効率が悪く，また，組み込まれた複雑な構造が扱えない等の問題点がある．これに対し，meta-restriction grammar[3]における処理は，解析途中に制限を加え処理の爆発を抑え，効率のよいシステムをめざしている．しかし，このシステムも複雑に入れ子になっている句を含む，長い文における並列関係の処理を行うには，不十分である．

一方，日本語における長い文に対する並列関係の解析については，黒橋と長尾[4]により，接続詞の前後の文節の類似関係を調べ，そこから文節列の類似度を計算して並列関係の範囲を決めるという方法が提案されている．この方法を英文の解析に応用することも考えられるが，英語のように，

・解析中の文の第一語目，あるいは，最初の数語によって，句の構造が決まりやすい．(例えば，1語目の品詞が前置詞なら，前置詞句を生成する)
・語順によって，各語，各句の文法的役割が決まりやすい．
・同じ形をした，多品詞語が多い．

というような特徴をもつ言語では，すべての語(文節)に対して類似度を調べるのではなく，各句のキーとなる語の類似度を調べればよいと考えられる．

通常，複雑な並列関係を見つける時，等位接続詞の前後の文から，同じ語，同様な語，同じような形が存在するかを見て，見つかれば，それを基本にして，並列関係を発見するということが多い．

そこで，並列関係解析において，

s1. 英文の等位接続詞の前後の句のキーとなる語から，同じ語，同様な語を見つけることにより，並列関係を決定する．

s2. 見つからなかった場合は，より似た形態のもの同士に並列関係をもたせる．

という制限を与えることにより，長い文における複雑に組み込まれた並列関係を，効率よく見つけだすことができるようなシステムを考え，実現した．

また，これらの制限規則を定義する記述法も開発し，文法記述者が，並列構造に関する処理を，独立して書くことができるようにした．

2. 並列関係解析システムの概要

並列関係解析システムは，英日機械翻訳システムの中で実現されている．この英日機械翻訳システムは，曖昧性除去のための構文的，意味的制限を加えたリスト構造で書かれた文法(文法例1)を Common Lisp プログラムに翻訳することによって実現されている．

[文法例1] 名詞句の定義

```
(np ((det ap verb_ing_pp noun) (sp_test))
    ((det ap noun)             (sp_test))
    (det verb_ing_pp ap noun)  (sp_test))
        .....
```

文法規則の最後の要素 (sp_test) が制限．この制限では，限定詞と，名詞の単複数を比較して，矛盾がないときのみ，前に定義されている文法に沿った解析結果が受理される．

並列関係の処理は，次のような手順でなされる．

s1. ある句の処理中，等位接続詞に出会うと，これまでの処理は中断され，並列関係処理システムに入る．

s2. 接続詞の後の文を見て，処理中の句と同じ句を生成するかどうかを調べる．

s2.1. 生成する場合は，等位接続詞の前後の句をつなぎ合わせ1つの句として処理し，中断されていた元の句の処理に戻る．

s2.2. 生成しない場合は，中断されていた句の処理に戻り，この句の処理を終了させ，入れ子の外側の句の処理に戻る．

ここで，再び，等位接続詞に出会うことになり，s1，s2 の処理を繰り返す．

[例1]

The field engineer replaced the board and adjusted the disk driver.

この文の処理において，“the board” からなる名詞句の処理過程で，等位接続詞に出会う．並列関係処理システムで，接続詞の後の文は，名詞句を生成するかを調べると，名詞句とはならない．そこで，中断されていた名詞句の処理に戻り，名詞句 “the board” の処理は終わり，入れ子の外側，動詞句の処理に戻る．ここで，再び，等位接続詞と出会い，並列関係処理システムに入る．すると，今度は，接続詞の後の文は，動詞句となるので，接続詞の前後の動詞句は，接続詞によってつながれた1つの動詞句として解析され，元の処理に戻る．

[文の解析結果]

```
(文 (主語       the field engineer)
    (動詞句
      (動詞句   replaced the board)
      (接続詞   and)
```

(動詞句 adjusted the disk driver)))

しかし，これだけでは，接続詞をはさんだ最も近い同じ句同士が並列となってしまうので，並列関係解析過程に，より同じようなもの同士の並列を優先させるという制限を加えることにより，入れ子となった句を含む複雑な並列関係も正しく解析できるようにした．

[例2]

It can be asked for one line description of commands specified by name, or for all commands whose description contains any of a set of DS.

「同じ前置詞で始まる前置詞句同士の並列関係は，優先させる」という制限を与えると，上の文の処理は，次のようになる．前置詞句 "by name" の処理過程で接続詞に出会い，並列関係処理システムに入り，接続詞の後の文と並列関係を形成するかどうかを調べる．すると，接続詞の前後の文は，共に前置詞句を生成するが，前の文の範囲を広げると，後の文と同じ for で始まる前置詞句がある．従って，制限により，接続詞は，下線で示されるように，for で始まる2つの前置詞句の並列関係を形成するとみなされる．ここで，前置詞句の文法規則は，前置詞＋名詞句としているので，先に述べた制限により，前置詞句の類似性を判断するには，前置詞句の第一語目のみを比較すればよい．

3. 並列関係決定のための制限規則

3.1 制限規則定義のための記述法

並列関係の範囲を決める制限規則を定義する記述法を定義した．この記述法では，並列関係を形成すると思われる句の形態を優先度をつけて定義する．並列関係解析における曖昧性は，この優先順位によって解消させる．次に，記述法の詳細について述べる．

[制限規則のための記述法の一般形]

(優先度 範疇名 規則)

(優先度 範疇名 (範疇名 規則))

これは，入れ子になった構造のための制限規則

ただし，

優先度は，値が大きいほど，高いとする．

範疇名は，通常の文法規則に書かれる文法的範疇名（句の名前，例えば，declm（文←名詞句＋動詞句），np（名詞句），prepp（前置詞句）etc.）であるが，すべての範疇に対して，共通な規則に対する範疇名は，all とする．

規則は，次のキーワードにより，定義される．

[キーワード]

same-word 同じ語

ただし，a, the は，除く

similar-word 同様な語

同様な語の定義は，品詞によって異なる．前置詞の場合は，単語の数が限られているので，in と out, to と from のように対になって使われる語をあらかじめ登録しておき，これらの語を互いに同様な語という．名詞句の場合は，それを生成する名詞，動詞句の場合は，その主動詞の意味カテゴリが同じ語をいう．従って，この場合は，意味カテゴリの分類，定義の仕方により，結果の精度が異なることがある．

same-hinshi 同じ品詞の語

same-tens 同じ時制

same-syntax 同じ文法的形態

check n check 関数呼出し

例外事項処理のために，check 関数を用意した．個々の例外事項の区別は，nで指定される識別番号によってなされる．

first n 最初からn語

last n 最後からn語

ただし，first, last で，n が，1 のときは，省略できる

nil 制限なし

接続詞の前後が範疇名で指定された同じ範疇を生成するなら，並列関係を成立させる

[制限規則の例と，意味]

(10 declm nil)

等位接続詞の前後の文が，declm（文）を生成しているならば，優先度 10 で，無条件で，これらの並列とする．

(5 prepp same-word)

等位接続詞の前後の文が，同じ語で始まる前置詞句の場合は，優先度5で，これらの並列とする．

(5 np same-word first 3)

等位接続詞の前後の文が，名詞句で，名詞句の最初の3語に，共通の語が含まれる場合は，優先度5で，これらの並列とする．

(4 np-prepp (prepp same-word))

等位接続詞の前後の文が，[名詞句＋前置詞句]で，この前置詞句が，共に同じ前置詞で始まる場合は，優先度4で，前置詞句は，名詞句に係り，これ

らの名詞句の並列とする．

(11 declm check 1)

識別番号1の check 関数で指定されている制限と一致した場合は，解析中の等位接続詞は，文 (declm) の並列を形成しない．この制限は，優先度11で，適用される．

3.2 制限規則の定義

上に，述べた記述法を使用し，並列関係に関する制限規則が書けるが，現時点では以下のような方針で，[制限規則1] を定義した．

[方針]

並列関係を見つけるための基本的な優先度は，次のとおりとする．

1. 文，節，不定詞句，分子句，動詞句の並列が成り立つときは，それらの並列とする．
2. 接続詞の前後の単語の品詞が同じときは，単語の並列とする．
3. 接続詞の前後の句 (e.g. 名詞句，形容詞句) が同じときは，句の並列とする．

これに，入れ子になっている句の解析が正しくなされるよう，より同じ形をした句の並列を優先させる制限を加えた．更に，[例3] で示すような例外的な文の処理ができるよう check 関数を加えた．

[例3]

They heard John and his friends leave the house.

この文は，and の後の文で，文 (declm) が生成されるので，文の並列と解析することもできる．しかし，この場合，leave 以下の動詞句は，heard を修飾すると解析されなければならない．これは，辞書の hear の属性に，[動詞＋名詞句＋原形不定詞句] となるような，動詞句を持ち得ると登録されているので，構文的には矛盾がない．このように，文の主動詞の属性に，[動詞＋名詞句＋原形不定詞句]，[動詞＋名詞句＋過去分子句] となる動詞句を持ち得ると登録されている場合は，“文の並列のよりもこの構文の解析を優先させる”，という制限を check 関数1で定義する．これにより，[例3] の and は，文の並列ではなく，名詞句の並列を形成すると解析される．

[制限規則1]

```
r 1.  (11 declm check 1)
r 2.  (10 declm nil)
r 3.  (10 to_inf nil)
r 4.  (10 clause nil)
r 5.  (10 gerund nil)
r 6.  (9  vp      (verb same-word))
r 7.  (8  vp      (verb same-tens))
r 8.  (7  vp      nil)
r 9.  (6  all     same-hinshi)
r 10. (5  prepp   same-word)
r 11. (5  np      same-word first 3)
r 12. (4  prepp   similar-word)
r 13. (4  np-prepp (prepp same-word))
r 14. (3  np-prepp (prepp similar-word))
r 15. (3  np-rel_cl   nil)
r 16. (3  np-gerund   nil)
r 17. (3  np-pastp_cl nil)
r 18. (2  np      similar-word first 3)
r 19. (1  np-prepp  nil)
r 20. (0  all     nil)
```

4. 処理の概要

4.1 処理の実現のための方針

並列関係のための制限規則を適用するにあたり，解析木のスタックを利用することもできるが，より少ない情報で，効率のよい処理をすることを考え，次のような2つのスタックを用意した．

[範疇のスタック]

ある範疇の処理に入る時，その範疇名をスタックに入れる．ただし，この対象となる範疇は，並列関係に関する制限規則に現れる範疇名のみでよい．そして，その範疇に関する処理が終わると，名前は，スタックから除かれる．

また，各範疇の処理の過程で，係り受けの入れ子関係が成立するかどうかの判断は，いくつかの制限規則により，係り受けの曖昧性が除去されるまで延ばすことにする．

[文のスタック]

範疇の処理に入る時，範疇名のスタックへの登録と同時に，解析対象となる文をスタックに入れる．これも，範疇に対する処理が終わると，除かれる．

並列関係の処理は，次のような手順でなされる．

s 0. ある範疇（範疇1）に関する処理を行っているとき，等位接続詞に出会うと，並列関係処理サブシステムに入る．

s 1. [制限規則1]に，範疇1に関する制限規則より優先度の高い規則が存在し，その規則の範疇名が，範疇スタックに含まれるとき，その規則による並列

関係が成立するかをみる．

s1.1. 成立する場合，s2.1 へ

s1.2. 成立しない場合，範疇1に関する規則が成立するかを優先度の高い順に調べる．

s1.2.1. 成立する場合，2つのスタックをみて，この並列関係がより広い範囲で形成されるものかをみる．

s1.2.1.1. 広い範囲での並列関係形成の場合，s2.1 へ

s1.2.1.2. 処理中の句との並列関係形成の場合，s2.2 へ

s1.2.2. 成立しない場合 s2.1 へ

s2.1. 処理中の句の処理を終了させ，スタックを更新し，この句は，更新後のスタックのトップに登録されている句の一部とみなされ，並列関係の処理は一応終わる．文は，接続詞のところから再び解析され，s0 からの手順が繰り返される．

s2.2. 処理中の句と，接続詞の後の文から成る句との並列とみなされ，これらは，1つの句として処理される．ここで，並列関係の処理は終わり，残りの文の処理を継続する．

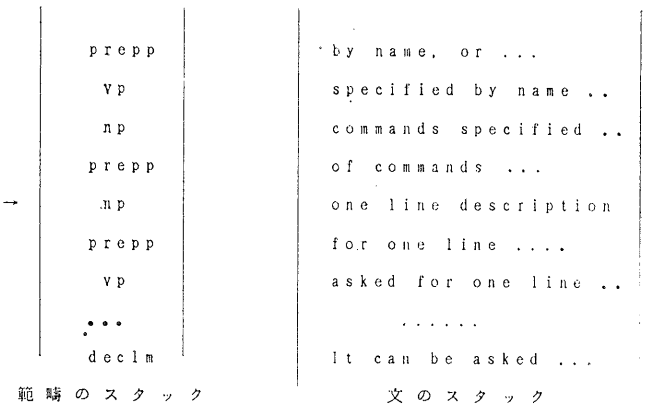

図1 スタックの状態
Fig. 1 Status of stacks.

4.2 具体例をともなった処理過程

次に，例文を通して，処理の手順の中に，制限規則がどのように適用され処理が行われるかを見る．

[例4]

It can be asked for one line description of commands specified by name, or for all commands whose description contains any of a set of DS.

前置詞句 by name の処理をしているとき or がくる．(s0) そこで，s1. に入り，or に続く句が，どの句と，並列関係を形成するかを [制限規則1] によりスタックを使用し調べる．そのときのスタックの状態は，**図1**のようである．

[制限規則1]より，現在処理中の前置詞句に関する規則より高い優先度を持つ規則は，r1〜r9 で，これらの規則の範疇名の中でスタックに登録されているのは，declm と，vp である．or 以下の文は，declm, vp を生成しないので，r1〜r8 は，成立しない．また，r9 も，or の前後の語の品詞が，同じでないことから成立しない．(s1) 次に，or 以下の文が，or の前と同じ前置詞句を生成するかを調べる．or 以下は，前置詞句を生成するので，前置詞句に関する規則の中で，最も優先度の高い規則 r10 から成立するかどうかを調べる．(s1.2) r10 より，範疇のスタックの各 prepp の位置と対応する文スタックに登録されている文の第一語目が，or に続く語と同じ for であるかを比べると，三度目の比較で，for で始まる前置詞句が登録されていることが分かる．従って，and 以下の for で始まる前置詞句は，スタックに登録されている for one line から始まり，and の直前の by name で終わる前置詞句と並列関係を形成するとみなされ (s1.2.1, s1.2.1.1)，前置詞句，by name の処理は，接続詞の前で終わり (s2.1)，スタックから除かれる．図1の矢印の上のスタックに登録されているすべての句の解析において，同様のことがいえ，これらの結果は，入れ子になって for one line で始まる前置詞句を生成する名詞句に係ると解析される．そして，or は，for で始まる2つの前置詞句の並列関係を形成する等位接続詞と解析される．

[例5]

We also learn how to store the value of a computation for subsequent use, and how to recover from an error.

and は，名詞句の処理過程で現れ，そのときのスタックの状態は，**図2**のとおりである．

まず，[制限規則1]で，名詞句に関する規則より高い優先度を持つ規則，r1〜r10 に関しては，並列関係が成立するかどうかを調べる．(s1) and 以下の文で，declm, to_inf（不定詞句）は，生成されないので，次の規則 r4 が，成立するかを調べる．すると，範疇のスタックに，clause（節）が登録されており，かつ，

and 以下の文は，節を生成するので，この and は，how to で，始まる2つの節の並列関係を形成すると解析され，(s1.2.1) subsequent use からなる名詞句は，how to store...から成る節の一部とみなされる．

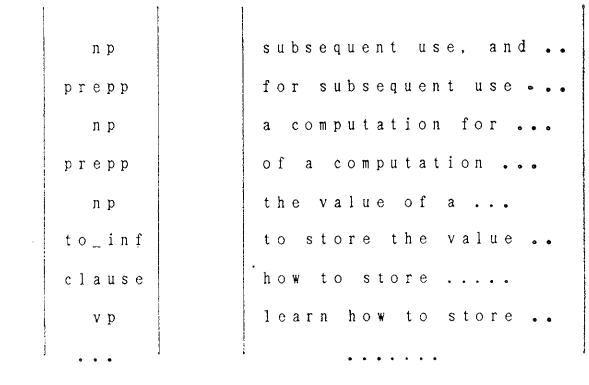

図 2 スタックの状態
Fig. 2 Status of stacks.

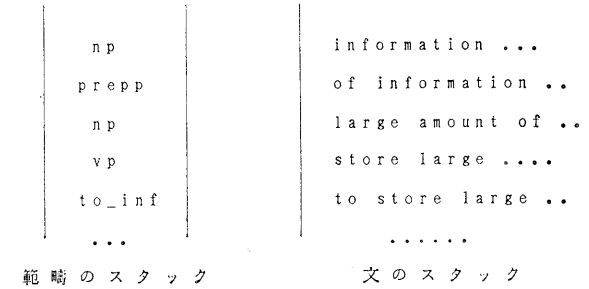

図 3 スタックの状態
Fig. 3 Status of stacks.

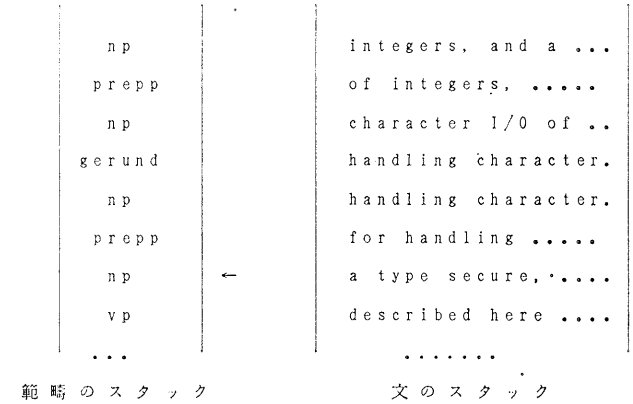

図 4 スタックの状態
Fig. 4 Status of stacks.

[例 6]

It was the ability of these new electronic machines to store large amounts of information and process it at very high speed that gave researchers the vision of building systems which could emulate some human abilities.

and は，名詞の処理過程で現れ，そのときのスタックの状態は，図3のとおりである．

ここで，and の後の語 process は，動詞，名詞の両方の品詞を持ち，この2つの品詞の使用度は，変わらないとする．まず，process を名詞とすると，and は，名詞 information との並列関係を形成するとみなされ，動詞とすると，動詞句の並列，つまり，store large... から始まる動詞句との並列関係を形成するとみなされる．しかし，(s1,s1.1,s2.1) 制限規則によると，同じ時制をもつ動詞句の並列の優先度は 8 (r 7) で，同じ品詞を持つ語の並列の優先度 6 (r 9) より高い．よって，process は，動詞とみなされ，and は，動詞句の並列関係を形成すると解析される．

[例 7]

The standard stream I/O library described here provides a type secure, flexible and efficient method for handling character I/O of integers, and a simple model for extending it to handle user-defined type.

2番目の等位接続詞 and は，名詞句の処理過程で現れ，その時のスタックの状態は，図4のようである．

ここでは，(s1.1) 名詞句に関する最も優先度の高い規則，r 11 より高い規則は成立しないので，名詞句に関する並列関係が，成立すると仮定される．(s1.2)

and 以下の文は名詞句－前置詞句
(np a simple model)
(prepp for extending it...))

となり，また，r 11は，成立しない．図4をみると，範疇スタックの矢印のところに np, prepp と，登録されており，prepp に対応する文スタックの第1語目は，for である．従って，r 13 が成立し，for 以下の前置詞句は，名詞句に係り，この接続詞は，下線で示すような，名詞句＋前置詞句で構成される2つの名

詞句の並列関係を形成する．

これらの処理過程をみることによって，並列関係の解析に付随して，係り受け，多品詞語による曖昧性が，解消されることが分かる．

4.3 制限規則の実現

4.3.1 制限規則をシステムに組み込む方針

並列関係処理における制限規則をいかに英日機械翻訳システムの中に組み込み，実現するかを述べる．

1. 文法にそって書かれた制限規則を Common Lisp の関数に，変換する関数を用意する．

また，文スタック，範疇スタックは，リストで実現する．

2. 並列関係を表すコンマの処理は，接続詞の処理と同様に行う．また，並列関係が入れ子になる可能性のある場合は，コンマの有無により，下の例のように入れ子関係を決め，それに基づいて処理を行う．

[例8]

A, and B and C

Aと，B・Cとの並列

A, B and C, and D

A と，B・C と，D との並列

3. same-word, similar-word の処理は，文スタック，範疇スタックを見て，同じ語，同様な語で始まる同じ範疇の句が，等位接続詞の前に存在するかどうかの判断によりなされる．同じ語，前置詞句における同様な語の場合は，文スタックの先頭の指定された数の語と，等位接続詞の後の文の語とを比べればよい．また．名詞句における同様な語では，意味カテゴリを比較する．

4. 等位接続詞の前の句が入れ子になっているために起こる接続詞の後の文の解析の実質的な繰り返しは，さけるようにした．

5. check 関数の中味は，例外事項を表す一般的記述法を定義するのが困難なため，直接 Lisp プログラムで書くこととした．

4.3.2 制限規則のプログラムへの変換

制限規則のための関数は，次のように Lisp プログラムに変換され，実現される．

(1) 並列関係に関する規則を優先度の高い順に並べ替える．

(2) 制限規則変換ルーチンにより，規則を Common Lisp のプログラムに変換する．

[例9]

(10 to_inf nil)

は，次のような lisp プログラムに変換される．

```
(cond ((and
         (or
           (to_inf の処理中に接続詞に出会った)
           (この規則が優先度の高い規則である))
         (to_inf は，check 関数により除かれない)
         (接続詞の前に，to_inf がある))
       (cond ((setq tree (to_inf 接続詞の後の文))
              (cond ((to_inf の処理中)
                     (to_inf の並列形成)
                     (t return nil))))))
```

各制限規則に対するプログラムは，優先度の高い順に書かれるため，優先度の高い制限ほど先に判断され，適用される．

5. 結果と考察

以上の方針により，システムを作成し，科学技術系の本，マニュアルを中心とした文書から，等位接続詞を含む文をとりだし解析した結果を考察する．

並列関係の範囲決定は，制限規則を優先度の高い順に適用することによってなされる．従って，優先度の高い規則による並列関係が成立する場合は，全体が長い文，等位接続詞の前後が長い文においても，優先度が低い規則によって並列関係が決まる場合より，次に示すように，少ない比較回数で，並列関係の範囲を決定することができる．

[例5] を範疇スタックを参照しながら，[制限規則1] を適用して，解析してみる．and 以下の文は declm, 不定詞句を生成しない（これは，and の次の語，how をみただけで判断できる）ので，r1, r2, r3 は，成立しない．そして，次の規則 r4 により，等位接続詞は，2つの節の並列を形成すると解析される．このように，長い文における並列関係の検出においても，3回の解析，比較で並列関係を求めることができる．

これに対し，優先度の低い規則により並列関係の範囲が決まる場合は，優先度の高い規則から成立の可否を判断しなければならないためより多くの解析，比較が必要とされる．しかし，例えば，[例4] をみると，[制限規則1] の r1〜r9 の不成立は，and 以下の文の1語目をみただけで判断でき，前置詞の入れ子関係の判断をも含んだ並列関係を形成する句の範囲の決定は，範疇スタックに登録されている3つの前置詞句と対応した文スタックの文の第1語目の比較で決定され

る．このように，優先度の高い規則の不成立が，短時間の解析で判断される場合は，システムの効率に大きな影響を及ぼさない．

並列関係が，正しく解析された50文（1文の平均語数，約23語，等位接続詞の前にある語の平均語数，約10.5語，後にある語の平均語数，約11.5語）をみると，比較判断された制限規則の数は，平均約3.2個であった．また，並列関係を形成する語の平均は，前後各約7語であった．

次に，実際の科学技術系の文書の中から，等位接続詞を含む329文（1文の最高語数，60語，最低語数，6語，平均語数．23語）を取り出し調べてみたところ，解析結果が明らかにおかしいと思われるものは，32文であった．前後の状況により，どちらともとれるという場合も一応正解とすると，約90%の正解率を得ることができた．

解析できなかったいくつかの例をあげてみる．

・入れ子となった句の処理に関する誤り

[例10]

An escape mechanism is specified to allow control data such as Xon/Xoff to be transmitted transparently over the link, and to remove spurious control data which may be.....

ここで，接続詞 and は，r3 により，to be から，and までの文からなる不定詞句と，to remove ...からなる不定詞句の並列になるとみなされる．しかし，スタックをみると，allow から始まる不定詞句も登録されており，この場合は，remove と be より，remove と allow の方が，意味，用法等が似ているので，allow 以下の不定詞句と，並列関係をなすと解析されるべきである．

・制限規則を適用したために起こる誤りによるもの

[例11]

Our primary reason was a load and search times, which were.....

ここで，search は，名詞と動詞を持つ多品詞語である．ここでは，[制限規則1]により，r7が適用され，and は，下線のように動詞句の並列と誤って解析された．

上の2つの例にみられるような解析の誤りは，等位接続詞の前後の文をより詳細に比較するため，文を生成するすべての語，句の類似性を比べなくとも，キーとなる語（この場合は，動詞句を生成する主動詞，名詞句を生成する名詞）の類似性が正しく求められれば解決される．現システムでは，動詞の意味分類は，10，名詞の意味分類は，19のカテゴリとしているため，多くの動詞，名詞が，類似したものとなってしまう．動詞，名詞の意味分類をより細かくする等，類似性の概念を確立した上で，[制限規則1]に，

(11 to_inf (vp (verb similar-word)))

(8 noun similar-word)

を追加すれば，上の例のような誤りは，減少するであろう．

このほか，複雑なギャップを含む文等，制限規則に対する例外により起こる誤りはある．例外事項の処理は，適切な check 関数の導入等によりある程度解決することはできるが，当然，処理効率は悪くなり，文のキーとなるところの比較により並列関係を見つけ，システムの効率を上げるという本研究の意図に反する．しかし，文法記述者が，簡単に制限規則を追加することができるので，対象となる分野，作者等の特徴をふまえた対象別の制限規則，check 関数を定義すれば，処理効率をそれほど落とすことなく，より正確な解析結果を得ることができるであろう．また，並列関係処理システムの独立性から，意味，概念の分類等，翻訳システムの改良に伴った，制限規則の変更にも容易に対応できると考えられる．

謝辞 日頃，ご指導ならびに的確なご助言を頂いている成蹊大学名誉教授，現日本アルゴリズム会長和田 弘氏に，深く感謝いたします．

参考文献

1) Woods, W. A.: An Experimental Parsing System for Transition Network Grammars, *Natural Language Processing*, Rustin, R. (ed.), pp. 111-154, Algorithmics Press (1973).
2) Grishman, R.: *Computational Linguistics*, Cambridge Press (1986).
3) Hirschman, R.: Conjunction in Meta-Restriction Grammar, *J. Logic Programming*, Vol. 3, No. 4, pp. 299-328 (1986).
4) 黒橋禎夫，長尾 真：長い日本語文における並列構造の推定，情報処理学会論文誌，Vol. 33, No. 8, pp. 1022-1031 (1992).
5) 武田紀子：英日機械翻訳システムにおける並列関係の検出，情報処理学会自然言語処理研究会，Vol. 92, No. 74, pp. 9-16 (1992).
6) Dahl, V. and McCord, M. C.: Treating Coordination in Logic Grammars, *Am. J. Computational Linguistics*, Vol. 19, No. 2, pp. 69-91 (1983).
7) Sedogbo, C.: A Meta-Grammar for Han-

dling Coordination in Logic Grammars, *Natural Language Understanding and Logic Programming*, Dahl, V. and Saint-Dizier, P. (eds.), pp. 65-78, North Holland (1985).
8) Marcus, M. P.: *A Theory of Syntactic Recognition for Natural Language*, The MIT Press (1980).
9) 長尾 真, 辻井潤一, 田中伸佳, 石川雅彦：科学技術論文における並列句とその解析, 情報処理学会自然言語処理研究会, 36-4 (1983).
10) 渡辺藤一：前置詞・接続詞・間投詞, 現代英文法講座 第5巻, 研究社 (1958).

(平成5年7月23日受付)
(平成6年7月14日採録)

武田 紀子（正会員）

1970年東京女子大学文理学部数理学科卒業. 同年成蹊大学工学部経営工学科助手として勤務, 現在に至る. 自然言語処理, 機械翻訳の研究に興味を持つ.