5C−2

双方向並行構文解析と熟語処理への応用

山上勝義、松本裕治、長尾真
京都大学

あらまし

本稿では、並行構文解析システムSAX [松本 & 杉村 86]の双方向解析への拡張について、そして双方向構文解析の応用として主に熟語処理について述べる。さらに、熟語処理での説明をもとに下位範疇化、カテゴリー文法への応用についても言及する。

1 はじめに

自然言語の構文解析において、どのような言語においても熟語の処理は重要な位置を占めている。熟語処理の方法には様々な立場がある。熟語を文法規則として書くというのも一つの方法である。だが、その場合文法の数が膨大なものになる。すると解析の途中結果も膨大な量となり、いたずらに構文解析の効率の低下を招くだけで、現実的な方法ではない。

一方、熟語に関する情報を辞書の中の単語項目に記述し、その単語が文中に現れた時のみその熟語情報を解析に用いる方法がある。我々は後者の考え方を並行構文解析システム SAX 上に実現した。

このアプローチによると文中に現れる単語を含む熟語のみが考慮されるので、熟語処理のプロセスを通常の構文解析プロセスに組み込んでも効率の低下は必要最小限に抑えられる。

SAX は左から右への一方向の構文解析プログラムを生成するが、熟語中のどの単語で熟語プロセスを起動するかによっては逆方向への構文解析の必要も生じる。(例えば、熟語 "fall in love with" において "love" の辞書記述に熟語情報がある場合、この熟語を解析するプロセスは "love" で起動され左方向へ "fall" と "in" を探しに行くことになる。)そこで、双方向構文解析への拡張が必要となるのである。

SAX では DCG で記述された文法を Prolog のボトムアップ構文解析プログラムに変換する。解析の方法はチャートパージング [Kay 80] と同等な処理である。DCG の終端記号(単語に相当)および非終端記号は基本的にそれと同じ名前を持つ Prolog の述語に変換される。個々の述語はプログラム中では並行に動作するプロセスとして動作する。それぞれのプロセスは隣の述語プロセスが出力した解析結果を受けとり、次の述語プロセスに解析結果を渡すことで解析が進んでいく。SAX の解析法は解析の中間結果にその時点でのあらゆる可能性が含まれている横型探索なので、バックトラックをする必要がない。よって Prolog のような逐次処理系でも高速な構文解析が可能である。

2 双方向パージングと熟語処理

"take care of" を例にとって辞書記述について説明する。この熟語は普通の辞書においては見出し語 "care" のところにある。それにならって図 1 のような辞書項目の述語を定義する。ただし、この定義は便宜的なものであり、現実には他の情報も含まれる。

```
dict(care,[cat(n),
   local_g([
     [ r([cat(prep),lex(of)]),
       l([cat(vt),lex(take)]) ] => vt
   ] )] ).
```

図 1: "take care of" の辞書記述

熟語情報は "local_g(...)" の部分に書かれている。これを局所文法 (Local Grammar) と呼ぶことにする。その意味は、「 "care" の右隣に 前置詞 の "of" があり、左隣に他動詞の "take" があれば、"take_care_of" という他動詞があるとみなす。」ということである。

ひきつづき、実際に熟語を処理する方法について述べる。"take care of" という熟語を認識するプロセスは文中に "care" を見つけた時から開始される。熟語情報によると、まず右隣に前置詞の "of" を探さなければならない。そこで前置詞だけが受けとる識別子をつけた情報を隣の述語に渡す。この方向は通常の SAX のデータの流れる方向と同じなので、特別な操作を必要としない。その情報には "of" を探していること、この時点 ("care" が受けとった) での解析の途中結果、その次に行なうべき動作の記述が含まれている。もし隣に前置詞がなければこの情報は読み飛ばされ、その時点で熟語処理は終了する。

いま "of" が見つかったとする。再び熟語プロセスが呼び出され次の動作を実行する。今度は左方向に他動詞の "take" を探すことになる。我々はオリジナルの SAX に手を加えて、それまでに完成した部分の解析結果が、そのカテゴリー名を識別子に持つようにしている。(例えば、名詞句ができたら "np" を識別子にする。)

よって、この場合は保持していた途中結果の中で "vt" という識別子を持ちかつそれが "take" であるものを探すことになる。もし、"take" がなければ熟語プロセスは何もせず熟語はなかったことになる。そして "take" が見つかった場合は、"take_care_of" という他動詞が文中に現れたとみなして、それに対応する新しいプロセスが生成される。そのプロセスは

Bidirectional Concurrent Parsing and its Application to Idiom Handling

Katsuyoshi YAMAGAMI, Yuji MATSUMOTO, Makoto NAGAO
Kyoto University

"take" が保持していた解析結果を受けとり自分自身を局所的な解析結果として "of" の隣の述語へと渡す。

このように、熟語を SAX の通常のプロセスとして出力することによって容易に熟語を認識している。

これらの処理は SAX と同じ解析メカニズムによって行なわれ、辞書項目に記述された熟語のための局所的な文法があたかももとの文法規則に含まれていたかのように動作する点に注目して欲しい。

また "take care of" の変形として "take much care of" の場合は "much care" という名詞[1]に "care" の辞書情報を継承させ、"much care" という名詞が上で述べたのと同じ動作をすることにより解析が可能である。

続いて、他のパターンの熟語について考える。先に例として挙げたようなパターンの熟語は、膠着型[2]の熟語である。しかし "not only ... but also ..." のように熟語のキーワードが分離していて、 ...の部分にさまざまな文法カテゴリーを取り得るものがある。この場合は左右に具体的な単語を指定することができない。この熟語は単語 "only" の辞書項目に記述するとして、図 2 のように書かれる。

```
dict(only,[cat(adv),
   local_g([
    [ l([cat(adv),lex(not)]),
      r([cat(Cat)]),
      r([cat(conj),lex(but)]),
      r([cat(adv),lex(also)]),
      r([cat(Cat)]) ] => Cat
    ]) ] ).
```

図 2: "not only ... but also ..." の辞書記述

図 1 の場合と違う点は、"only" と "also" の右側の文法カテゴリーそして出力する文法カテゴリーと同じ名前の変数になっていることである。これはこれらのカテゴリーが任意ではあるが同一でなければならないことを意味する。したがって、"only" と "also" の右側に来る文法カテゴリーが熟語全体の文法カテゴリーを決定することになる。

3 下位範疇化とカテゴリー文法

前節で述べてきた方法を拡張すれば、下位範疇化の考え方を同様の方法で実現できる。

目的語の数、それが生物か無生物か、などの制限は述語動詞に依存する場合が多い。個々の動詞に関する下位範疇化の情報を辞書に記述し、熟語処理に用いた手法を応用してその動詞が要求する目的語などのフィーチャーをチェックすることで、様々な形態の動詞を扱うことができる。この場合、基本的な少数の文法があればよいことになる。

もちろん、"cat(...)" や "lex(...)" の他にも必要な制限を書くことになる。図 3 に "give" の目的語に関する辞書記述例を示す。これは "give" の構文として、"give *IO DO*" と "give *DO* to *IO*" があり、*IO*,*DO*の文法カテゴリーは名詞句で *IO* は生物 *DO* は無生物であることを意味している。もちろん、各名詞句のフィーチャーはその中の名詞のフィーチャーを継承していることが前提である。

```
dict(give,[cat(v),
   local_g([
    [ r([cat(np),animate]),
      r([cat(np),physobj]) ] => vp,
    [ r([cat(np),physobj]),
      r([cat(p),lex(to)]),
      r([cat(np),animate]) ] => vp
    ]) ] ).
```

図 3: "give" の辞書記述例

双方向構文解析で注意しなければならないのは、解析の重複である [Satta & Stock 89]。我々が現在許している熟語情報の記述では、熟語の成立条件は評価の方向がそれぞれ指定されている。したがって解析が重複することはないのだが、多少柔軟性に欠けるところでもある。

カテゴリー文法にこの手法を応用することは容易である。今までは特定の単語の辞書記述に書いてある局所文法に基づいて双方向プロセスが構文解析していた。つまり、双方向プロセスを起動するのは特定の単語である。プロセスの起動を単語からだけではなく非終端記号（複数の単語からなり特定の文法カテゴリーを持つ）からも行なうことでカテゴリー文法に基づく双方向構文解析が可能になる。

4 おわりに

主に熟語処理を具体例に挙げて SAX における双方向構文解析について述べてきた。熟語や、下位範疇化のように多様で局所的に出現する現象を扱うのに、有効である。

特に強調しておきたい点は、一方向ボトムアップパーザの枠組の中に自然な形で双方向のパージング機構を組み込むことができるということである。

参考文献

[松本 & 杉村 86] 松本裕治, 杉村領一, 「論理型言語に基づく構文解析システム SAX」, コンピュータソフトウェア, Vol.3, No.4 pp.4-11, 1986.

[Kay 80] Kay, A. "*Algorithm Schemata and Data Structure in Syntactic Processing*", Technical Report CSL-80-12, XEROX PARC, 1980.

[Stock 89] Olivero Stock, "*Parsing with Flexibility, Dynamc Startegies, and Idioms in Mind*", Computational Linguistics, Vol.15, No.1, pp.1-18, 1989.

[Satta & Stock 89] Giorgio Satta, Olivero Stock, "*Formal Properties and Implememtaion of Biderectional Charts*", IJCAI, pp.1480-1485, 1989.

[1] noun → adj, noun という文法があるとする。

[2] 熟語を構成する単語が連続して現れるという意味で用いた。