1N-8

# LangLABの構文処理アルゴリズムの高速化

上脇正　田中穂積　沼崎浩明

(東京工業大学工学部)

## 1．はじめに

本稿で説明する自然言語処理システム LangLAB は、BUP_XG[今野 86]をベースにした自然言語処理のためのソフトウエアシステムである。LangLAB システムには BUP_XG節をおよそ6・5倍高速化し、使用記憶容量を節約した構文処理アルゴリズムが組み込まれている。これは LangLAB トランスレータによって DCG_XG から最適化したBUP_XG節を生成することで実現される。本稿では、この高速化の方法を実験結果とともに示す。 筆者は LangLABシステムによる構文処理については、十分満足できる速度が得られたと考えている。

## 2．Link節の除去による高速化

BUP_XGでは、構文処理が進み、構文木が下から上(bottom-up)に成長した時、ゴールへの到達可能性をBUP-XGトランスレータが予め計算しておいたlink節によって調べている[田中86]。

使用文法規則の数が多くなるにつれて、BUP_XGトランスレータにより生成されるlink節の数は著しく増大する。ちなみに筆者等が使用している英語の文法規則の数はおよそ400であるが、BUP_XG トランスレータにより生成されるlink節の数は約600に達している。したがって、link節の検索時間の短縮が望まれる。

BUP_XG節のボディに現れる述語 linkを調べてみると、必ずその第一引数はアトムであり、第二引数は変数になっている。link(a,G) の形をしているのである。LangLAB トランスレータはDCG_XGを変換する時、BUP_XG節のボディに現れる link(a,G)を a(G) に変えたものを出力する。これを以下ではリンク対とよぶ。

次の文法規則：

```
np --> det, noun, srel../np.        (1.1)
```

は、大略次のBUP_XG節に変換される。 ここで、np(G) がリンク対であることに注意。

```
det(G,...) --> {np(G)},
               goal(noun,...),
               goal(srel,...),
               np(G,...).           (1.2)
```

このような最適化によって、非終端記号 np からゴール Gへの到達可能性を調べるための時間が大幅に短縮される。述語名 np をキーにしたハッシュ検索がなされるためである。

一方 goal節のボディ([田中86]図3参照) に現われる辞書引き後の link(C,G) は、

```
Link =.. [C,G],
call(Link)
```

に変更する。

さてリンク対の実際の呼び出し時の様子を観察してみると、第一引数 G は常にアトムになっているから決定的な動作をする。したがって、BUP_XGトランスレータが計算し出力するlink節のボディには、カット記号を入れることができる。そこで、LangLAB トランスレータは、BUP_XGトランスレータが出力する link(f,g) を次のものに変えて出力する。

```
link(f,g). ===>  f(g) :- !.
```

以上の様にして link節を別の形にして除去することにより、構文処理速度をおよそ6倍向上させることができる。筆者等はこの改善により、LangLAB システムの構文処理速度は十分満足できるレベルに達したと考えている（5章の実験結果参照）。

## 3．差分リストのインデックス化表現による記憶の節約

BUP_XGでは、文の処理対象部分が差分リストとして表現される。構文処理過程で生じる中間的な成功結果と失敗結果とは、それぞれwf_goalとfail_goalとしてアサートされる。この時、述語wf_goalの最後の2引数は、処理対象部分文の差分リストになっている。文法の規模が大きくなり、しかも処理すべき文の長さが長くなるにつれて、アサートされるwf_goal節とfail_goal節の数が著しく増大し、そのために大きな記憶容量を必要とする。この記憶容量は、差分リストをインデックス化することで節約できる。また計算結果を再利用する場合にも、それを検索するためのユニフィケーション時間が短縮されるため、 構文処理速度が幾分改善される[今野 84]（5章の実験結果参照）。

インデックス化のために、たとえば "you walk." という文が入力された段階で、

```
text(0,[]).
text(1,[walk]).
text(2,[you,walk]).                 (2.1)
```

をアサートしておく。そして辞書引きのとき与えられる

---

The Improvement of LangLAB's Syntax Analyzing Algorithm

Tadashi KAMIWAKI,Hozumi TANAKA,Hiroaki NUMAZAKI

Tokyo Institute of Technology

インデックスの対から、述語text を呼出して差分リストを復元し、実際の辞書引きを行う。したがって、goal 節のボディに現れる （辞書引きのための） 述語 dict （[田中86]図３参照）は、LangLABシステム では dictionaryという述語に変えられている。

述語 dictionary は、インデックスの対から（処理対象部分文の）差分リストを復元して辞書引きするだけでなく、辞書引きに失敗した場合には形態素処理を行う。また、辞書引きか形態素処理のいずれかに成功した場合には、その成功結果を wf_dict としてアサートしておき、再計算に備える。

## ４．直接辞書引きによる高速化

(1.1)のような文法規則は、(1.2)のようなBUP_XG節に変換されることは既に述べた。

ここで noun は単語の品詞であるとしよう。単語の品詞は非終端記号に属すが、他の非終端記号と異なり、書き換え規則により別の非終端記号に展開されることはない。言い換えると、単語の品詞からは書き換え規則により単語（終端記号）にしか展開されない。したがって BUP_XG 節の右辺から、noun の様な非終端記号に対する述語 goal を呼出す場合には、述語 goal の代りに述語 dictionary （２章参照）を用いて、直接辞書引きしても良い。(1.2)のBUP_XG 節を次のように変えてもよいのである。

```
det(G,...) --> {np(G)},
               dictionary(noun,...),
               goal(srel,...),
               np(G,...).          (3.1)
```

LangLAB トランスレータは、与えられた文法規則をみて、直接辞書引き可能な非終端記号を抽出し、上に示すBUP_XG節を出力する。直接辞書引きを行う BUP_XG 節を用いることにより、アサートされる wf_goal,fail_goal の数が減り、構文処理速度が１割弱改善される（５章の実験結果参照）。

## ５。ＢＵＰ　ＸＧ節の最適化に関する実験結果

以下に実験結果をしめす。コンパイルすると、構文処理時間は更に（Ｄ）の少なくとも１／５程度に短縮できよう。なお使用文法規則数は、DCG_XG の形式でおよそ４００ほどである。

（Ａ）使用計算機：SUN3/160ワークステーション

（Ｂ）使用Prolog：Ｃ－Prologインタープリタ

（Ｃ）使用例文：

1 There are three on the table now.

2 The books that were on the table are difficult to read.

3 If you have time, read these books.

4 This paper presents an explanatory overview of a large and complex grammar that is used in a computer system for interpreting English dialogues.

（Ｄ）実験結果と評価

実験は上記した各例文について、全ての構文処理結果が得られるまでの時間（秒)を以下の４つの場合に分けて測定した。

（ａ）最適化する前のBUP_XG 節を使用、

（ｂ）Link節の削除（１章）、

（ｃ）差分リストのインデックス化（２章）、

（ｄ）直接辞書引き（３章）。

測定時間には、形態素処理の時間も含まれている。また、例文番号に括弧付きで添えられた数字は、得られた構文処理結果の数を表す。

| 例文番号 | (a) | (b) | (c) | (d) |
|---|---|---|---|---|
| 1(2) | 13.11 | 1.85 | 1.83 | 1.73 |
| 2(1) | 52.56 | 8.55 | 8.31 | 8.10 |
| 3(1) | 57.48 | 9.23 | 9.06 | 8.76 |
| 4(8) | 186.36 | 47.48 | 46.55 | 39.93 |

結果を比較すると、次のことが分かる。

（他の例文の結果も合わして計算）

（１）　（ａ）／（ｂ）＝５．９９

（２）　（ａ）／（ｃ）＝６．１０

（３）　（ａ）／（ｄ）＝６．５２

３・１節、３・２節、３・３節に述べた最適化により、構文処理速度が次第に向上し、最終的には６・５２倍の高速化が実現されていることが分かる。特に link 節の除去による高速化が顕著である。Prologインタープリタではなくコンパイラが使用できれば、構文処理時間は更に１／５に短縮されようから、LangLAB システムは十分な構文処理速度に達した考えている。

## ６．おわりに

LangLAB システムの構文処理速度に関して、筆者らは十分満足できるレベルに達したと考えている。再計算を避けるアルゴリズムを導入することによって、構文処理速度がおよそ１桁改善可能なことが、[松本 83] では、報告されているから、再計算を避けるアルゴリズム導入以前の、最も初期のBUPシステムに比べてLangLAB システムは、構文処理に関しておよそ６０倍以上の高速化がはかられていることになる。また、コンパイルによりさらに高速化が可能である。

## 参考文献

[今野 84] 今野聡、奥村学、田中穂積：ボトムアップ構文解析システム BUP の高速化、 日本ソフトウエア科学会第１回大会論文集、3A-2(1984).

[今野 86] 今野聡、田中穂積：左外置を考慮したボトムアップ構文解析、日本ソフトウエア科学会編、コンピュータソフトウエア、3,2,115-125 (1986) .

[田中 86] 田中穂積：自然言語処理システム「LangLAB」、本大会予稿集(1986)

[松本 83] Matsumoto, Y. et.al.: BUP--A Bottom-up Parser Embedded in Prolog, New Generation Computing, 1, 2, 145-158(1983).