統語処理と意味解釈を同時に行うシムテムについて

白井英俊　横尾英俊　（東京大学工学部）

§1. はじめに

パージングは、自然言語処理の最も基礎的で重要な部分であり、これまでもいろいろな手法やシステムが提案されている。その中で、電総研の田中らによる拡張LINGOL[1]は、高速で柔軟であると共に、広汎な応用可能性を持ったすぐれたパーザであり、我々も1978年の末に東京大学大型計算機センター，HITAC 8700/8800のHLISP上に移植して以来，いくつかの実験や拡張を試みている[2][3][4]。特に、入力文を格文法的な意味表現に変換する意味抽出実験を行い，その過程で、拡張LINGOLの有効性を確認し、同時に、いくつかの問題を指摘する事ができた。次の二項目は、問題点のひとつの側面を表わしたものである。

(i) 「構文解析⟶意味処理などの後処理」という一方向的な処理が仮定されているために、構文解析に意味情報などの後処理の結果を反映させにくい。

(ii) 同音異義語や多義語、そして入力文のあいまいさなど、複数の可能性に対処する能力に欠ける。特に、構文解析の結果（構文樹）が、原則として一意に決まるものとされているので、「もっともらしさ」という尺度を伴った複数個の結果を出力することが困難である。

(i)に関しては、田中[5]も指摘しているように、意味の問題が未解決である現在、統語情報と意味情報の相互作用を安易に利用することはできない。しかし、我々が意味を格文法的表現として規定しているように、確定された意味抽出の枠組の範囲で、その情報を構文解析に反映させる事は、決して無益な事ではないと考えられる。特に、例文(1)や(2)のように、意味的に明らかに異なる統語構造を持ちながら、単純な構文規則では区別できない文に対しては、有効な手段になり得るであろう。

(1) 花子を酔払った太郎がなぐった。
(2) 花子を愛した太郎が死んだ。

(1)と(2)に対しては、動詞の取り得る格要素数(価数)を明示する事により、構文解析レベルで区別可能のように思われるが、より複雑な文では不可能で、どのようなものが格要素になるかという意味的な制限がより本質的であると考えられる。もちろん、拡張LINGOLでも、各種の制御機能(<cog>部，<advice>部等)を用いて意味情報を構文解析に反映させる事は可能であるが、構文解析段階での意味抽出と構文解析後の意味抽出という二重の意味処理を行う事は，必ずしも自然な扱いではない。構文解析に意味情報を反映させるのであれば、構文解析と意味抽出が同時に終了するのが、人間の言語理解のモデルとしても自然である。しかしながら、拡張LINGOLでは、構文解析の途中では、部分的な意味情報しか使えず、その時点で誤った判断を下す事になる。これは、拡張LINGOLに(ii)の性質、すなわち、構文樹生成の過程で、ある条件が満たされた場合に、一個の可能性のみが優先的に採用されるという性質があるからに他ならず、優先度の付加された複数個の可能性を残す方式にすれば

解決できる問題である。

以上のような考察に、拡張LINGOLのプログラムの検討を加え、我々はMELING (Modified Extended LINGol)とよばれるシステムを作製し、その上で、統語処理と意味解釈を同時に行うシステム、EXAMを試作している。そこでは、拡張された手続き付加を用いることで、意味解釈の「もっともらしさ」を表わす定量的な優先度を構文解析にフィードバックさせている。

§2. MELING について

拡張LINGOLは、基本的にCFパーザであり、その辞書及び文法の記述形式は図1.のようになっている。

(<left> <right> (<advice> <cog>) <sem>)

CF 文法規則　　補強部

図1. 拡張LINGOLの文法／辞書記述形式

拡張LINGOLでは、入力文に対し、ユーザの与えた辞書及び文法を用いてパーズするが、一般にその構文樹は複数個できる。しかし、構文樹を、入力文の各要素間の係り受けの表示と考えれば、人間にとっては入力文に対する構文樹は普通一意的であり、自然言語を機械処理する観点からも、人間の直感に合う構文樹が生成できる事が望ましい。その為、拡張LINGOLでは、<advice>部及び<cog>部に書かれたプログラムで、構文樹の生成を制御する様に意図されている。但し、<advice>部と<cog>部では、その評価時期も、評価結果が及ぼす影響も異なっている。<advice>部には、各文法規則が適用されると予測された時、及びその適用を実際に検討する時に評価され、その文法規則の適用・非適用を決定するようなプログラムが書かれる。ここには通常動詞の語幹と語尾の承接条件検査のためのプログラムや助詞，助動詞間の承接条件検査のためのプログラムが書かれる事からわかる様に、文法規則の適用を局所的な情報だけで、かつdeterministicに決定できるような種類のプログラムが意図されている。また、拡張LINGOLでは、<cog>部は「あいまいな註」部分構文樹が生成された時、及び入力文全体に対する構文樹が完成した時に評価され、その結果により、どの部分構文樹を優先するかが決定される。この優先度は数値でなければならず、「あいまいな」部分構文樹のうちで優先度が最大のものだけが構文樹生成の材料として残され、他のものは棄却さ

註) ここで言う「あいまいである」とは、図2.のように同一のルート・ノード(「体言」)及び同一の末端語彙の列(「黒い髪の少女」)を有する部分構文樹が存在する時の事を言う。

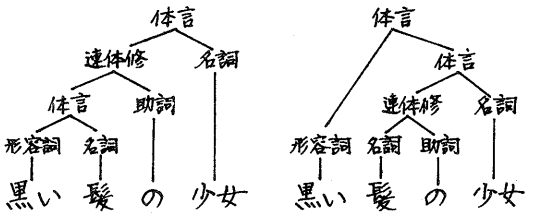


図2.「あいまいな」部分構文樹の例

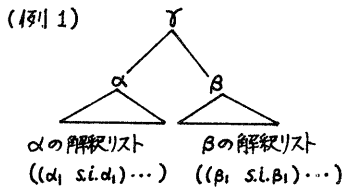
れる。また、<sem>部にはできあがった構文樹をプログラム木とみなして評価されるようなプログラムを書く事ができる。

MELINGにおける主要な拡張は次の三点である。

1) 意味処理のパージングへの反映
2) 本質的な「あいまいさ」があった時の処理
3) 音便処理を含む形態素処理の強化

この1)、2)を効率的、かつ効果的に行う為に、MELINGでは、<advice>部では統語的な処理を、<cog>部では、意味的な処理を行なうよう意図されている。そして、拡張LINGOLでは、<cog>部は単にあいまいさが生じた場合の部分構文樹の優先度を決定する為、優先度(数値)を値として返すよう意図されていたが、MELINGでは、「その部分構文樹で得られる『意味解釈』と、その『もっともらしさ』(優先度)とのリスト——これを解釈リストと呼ぶ——のリストを返す」ようなプログラムと規定されている。

(例1)



(γ (α β) (<advice>
(#MM 'Ωα+β=γ (ΩLC) (ΩRC))) 0)

<cog>部

ΩLCを評価するとαの解釈リストがとられ、ΩRCを評価するとβの解釈リストがとられる。但し、α_1, β_1は数値で、s.i.α_1, s.i.β_1は、α, βのそれぞれの意味解釈とする。

この規則の<cog>部の評価により、これらの解釈リストを引数とする意味処理関数 Ωα+β=γ が評価されて

((γ1 s.i.γ1) ····)

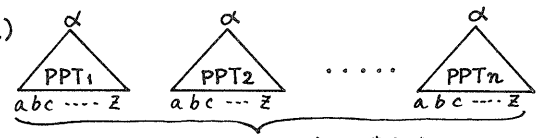
というγの解釈リストが作られる。γの「もっともらしさ」を表わすγ_1という数値は一般に $\alpha_1+\beta_1$ に等しくない。それは、そこでの意味解釈結果に依存して決まるものである。#MMについては後述。

<cog>部が評価されるのは、普通、ある程度大きな部分構文樹に対してであるから、文法規則が適用される度に意味処理を行なう方式(例えば<advice>部で意味処理を行なう方式)に比べ、通常統語処理に比べて時間のかかる意味処理をそう頻繁に行なわなくてもよいという利点に加え、より大きな立場から解釈を行えるという利点がある。

しかし、§1.で述べたように、<cog>部で得られた意味解釈を拡張LINGOL式に一個しか上位のノードに伝達しないのでは、部分的な意味情報しか用いていない事からくる誤りの可能性がある事と、また、同程度に「もっともらしい」部分構文樹が存在する場合は、恣意的にならざるを得ない、という欠点がある。だが、得られた解釈をすべて上位のノードに伝達するのでは、場合数の組み合わせ的増大が予想され、能率的でない。従って、MELINGでは、ユーザがグローバル変数#N#に値を設定する事によって、上位のノードに伝達される解釈リストの個数を制御するようにした。この#N#の値が0以下ならば、拡張LINGOL方式と同じで

上位に、ただ一個しか伝達しないが、#N# が1以上の時は次の例のように、特殊な制御方式になる。

(例2)



上図のように複数個のあいまいな部分構文樹(PPT)が存在し、各々の解釈リストを集めてソートしたものを、仮に、

((15 s.i.1) (10 s.i.2) (10 s.i.3) (0 s.i.4) ····)

とする。ここでMELINGは、優先度順に相異なる解釈リストを #N# 個選ぶ。そして、それ以外のもののうち、優先度が選ばれたものと同じであるものをも上位ノードに送る。従って、#N# = 2とすると、上位ノードに送られるのは、ここでは、次のリストである。

((15 s.i.1) (10 s.i.2) (10 s.i.3))

この方式において、#N# のいろいろな値についてのパーズ時間、及び解釈結果は次節で述べるが、この方式の思想は、「同程度にもっともらしいものは、同等に扱われるべきだ。」という事である。

また、このような拡張に伴い、<cog>部に書かれる関数は当然、複数個の解釈リストの各々に対して意味処理を行なう関数となる。一般にこのような関数は書きにくいので、ユーザには、ただ一個の解釈リストを引数として処理する関数を書いてもらうだけでよいように、C下文法規則が、α→β型のものには #M1、α→β γ型のものには #MM というmap型関数を用意した。(例1の②α+β=γは一個ずつの解釈リストを引数として処理する関数である。)

以上のような拡張によって、次のような利点が得られた。

① <sem>部で意味処理を行なう方式では、望ましくない構文樹が得られた場合を考えなければならないので、意味処理のプログラムが複雑になるが、MELINGでは、統語構造をそのまま反映したプログラムを<cog>部に書けばよい。

② 本質的に breadth-first であるから、無用な back-track をしなくてすむ。

③ 同音異義語に対しても、統語的なあいまいさと同様に処理できる。——同音異義語に対しては、辞書の<cog>部に解釈リストを複数個書いておけば、統語的なあいまいさと同様に処理してくれる。

④ ユーザは自分が書いた意味処理プログラムではどのような解釈を「もっともらしい」ものとしてしまうか知る事ができる。例えば、次節で述べる応用プログラム EXAM では、例文(3)に対して、「太郎が一所懸命」と「次郎が一所懸命」という二つの解釈が同程度にもっともらしいという事を出力する。

(3) 太郎は次郎に一所懸命理科を勉強させた。

§3. 意味解釈システム EXAM について

EXAMは、MELINGをパーザとする試験的な意味解釈システムであり、次の三

つの特徴を有する。

1) 意味解釈がパージングに反映される。
2) フレーム的な意味記述言語を用いている。
3) 係り受けの非交叉条件を破った文、未定義語を含む文、あいまいさを伴った文等を柔軟に処理する能力を有する。

EXAMでは、統語規則がそこで行なわれる意味処理と密接に結びついている。例えば、

```
(KU (NP KU) (nil (#MM '@NP+KU=KU (@LC) (@RC))) 0)
```

という文法規則は、NPという名詞がKUという述語句の何らかの格(case)である、という事を表わし、そのような解釈を行なう意味処理プログラムが<cog>部に書かれている。このように、意味処理は統語規則に依存し、パージング過程を方向付けるのは、この意味処理から得られた「優先度」である。

§3.1 意味記述言語GIRL(仮称)

Bobrow & Winograd[6]は、知識表現言語KRLを提案し、その中で、フレームによる意味記述、手続き付加、multi-description等の有効性を示している。我々はパージングに必要な意味を記述するための言語GIRLを作り、これらを導入する事にした。GIRLではKRLにならって、意味記述のフレームをユニット, ユニット間のリンクをスロットと呼んでいる。

GIRLの形式は図3.のとおりである。

```
unit ::= [ unit-name  category  self-slot  slot_1 .... slot_n ]
category ::= PROTO|INSTANT
self-slot ::= (SELF  perspective_1 .... perspective_n)|(SELF)
perspective ::= (A  unit-name  WITH  semantic-triplet_1 .... semantic-triplet_n)
slot ::= syntactic-triplet|semantic-triplet|prototype-triplet
syntactic-triplet ::= (marker  facet  filler)
semantic-triplet ::= (role  facet  filler)
prototype-triplet ::= (role  Check-To-Fill  When-Filled)
```

図3. GIRLの意味記述形式

EXAMでは、パージングとはプロトタイプとなるフレームをinstantiateする事によって、その文や句の意味を表わすフレーム(定形フレーム)に変形する操作と考えている。フレームのカテゴリーをPROTOとINSTANTに分類し、又、スロットをいろいろな種類に分類しているのもこの考えの現われである。

あるフレームをinstantiateするとは、その中のprototype-tripletをsemantic-tripletに変換することや、いろいろなsemantic-tripletを付加、又は除去することと考えられる。prototype-tripletからsemantic-tripletへの変換では、まず、そのフィラーの候補となるものが、prototype-tripletのCheck-To-Fill(CTF)で検査される。ここで返される値は数値かNILであり、NILの場合は、その変換は行なわれない。数値

の場合は変換が行なわれ、この数値がそこでの「もっともらしさ」を表わすものとされる。さらに、When-Filledが実行されて、結局この過程での、意味解釈と「もっともらしさ」の決定がなされたことになる。

§3.2 「もっともらしさ」の判定

初めに断わっておきたいのは、この「もっともらしさ」という概念は、MELINGを用いたシステムのみならず、自然言語理解システムを作る上で重要な概念である、という事である。我々はこの概念をもっぱらパージングを方向付ける「優先度」として数値化したが、同音異義語の処理や文のあいまいさの処理に重要な役割を果たしている。

EXAMでは、「もっともらしさ」の数値化が次のようにして行われている。

ある意味処理関数で処理される対象の「もっともらしさ」は、$\alpha \rightarrow \beta\ \gamma$型の構文規則では、$\beta$の解釈の「もっともらしさ」と$\gamma$の「もっともらしさ」の和を、$\alpha \rightarrow \beta$型では、$\beta$の解釈の「もっともらしさ」をその基礎点とする。そして、この基礎点に、その意味処理関数で得られた「もっともらしさ」を加点して、この結果の「もっともらしさ」とする。この時、加点される「もっともらしさ」の主なものは次のとおりである。

◎格を述語句に埋める場合

CTFは二つの要素からなり立っている。一つは評価結果がTかNILかで、そのフィラーの棄却が決定されるものであり、もう一つの要素はNILならば、「もっともらしさ」を低くするものである。普通、二番目の要素には意味素性検査のための関数が書かれる。

- CTFの二つの検査に合格した場合は、その格要素とされる。
  主語、直接目的語、間接目的語、その他という順に段階付けられた点数。
  （例　10点，8点，6点，5点）
- CTFの第一の検査のみに合格した場合。
  −50点とする。
- 一般格要素の場合
  述語句のフレームのself-slotにより、その上位のフレームを調べることでその格が一般格要素か否かが決定される。5点から、上にたどる度に2点ずつひいていく。
- どの格要素にもならなかった場合。
  棄却はせず、UNFILLEDをroleとするsemantic-tripletを作り、−50点とする。

◎副詞句を述語句に埋める場合

副詞によって、その述語句の意味素性から「もっともらしさ」が決まる。それにより、−100点から10点までの値を与える。また、表層における副詞と述語との距離によりこの点は多少変動する。

◎関係節等を名詞句に埋める場合

格を述語句に埋める場合に準じる。

ここで注意しておきたいのは、EXAMの「もっともらしさ」とは、その文の「自然さ」を表わすためのものではない、という事である。即ち、この数値は絶対的な価値を表わすものではなく、あくまで「あいまいな」部分構文樹が生成された時の優先度を決定するための相対的なものであるということである。

§3.3 EXAMの実行例

図4に、例文(1)、(2)から得られた構文樹を、図5に例文(3)から得られた意味解釈結果をあげる。また図6.は、例文(4)の処理に要した時間をいろいろな#N#の値でグラフにしたものである。

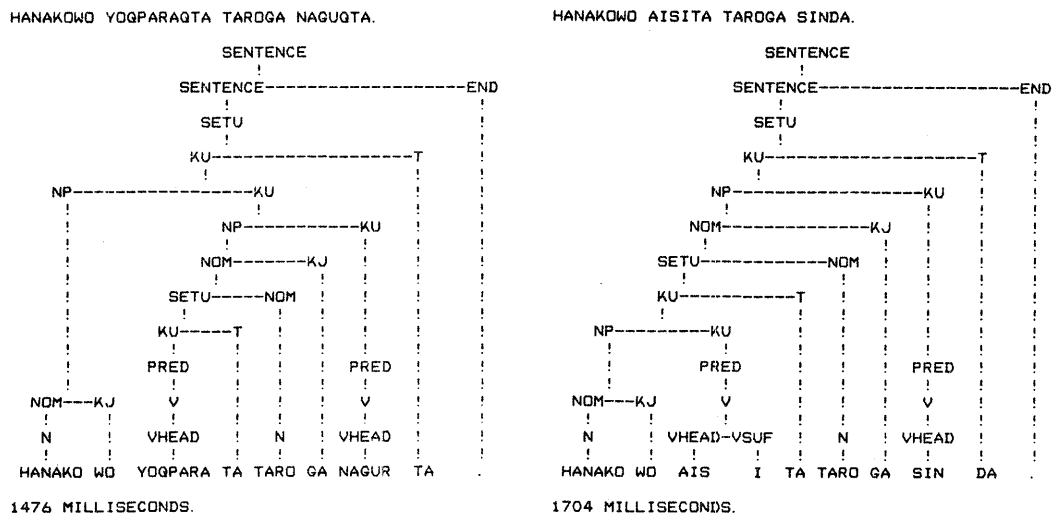


図4. 例文(1),(2)に対する構文樹。時間は意味解釈とパージングの和。

```
TAROHA ZIRONI IQSYOUKENMEI RIKAWO BENKYOUSASETA.  ← 入力文
LIKELIHOOD=38                    :もっともらしさ
< SASERU10  INSTANT              :「させる」
    ( SELF ( A SASERU ) )
    ( AGENT = TARO )             :主体者は「太郎」
    ( PATIENT = ZIRO )           :受動者は「次郎」
    ( GOAL = BENKYOU6 )          :させる事
    ( ASPECT = ( KANRYO ) ) >    :時制は「過去」
< BENKYOU6  INSTANT              :「勉強する」
    ( SELF ( A BENKYOU ) )
    ( MOD = IQSYOUKENMEI )       :「一所懸命」
    ( OBJECT = RIKA )            :対象は「理科」
    ( AGENT COREF ( ^ PATIENT ) ) > :主体者は、
                                  このフレームを指している
                                  フレームのPATIENT slot
                                  の値と同じ、即ち、「次郎」
< RIKA  PROTO
    ( SELF ( A KAMOKU ) ) >
< ZIRO  PROTO
    ( SELF ( A HITO ) ) >
< TARO  PROTO
    ( SELF ( A HITO ) ) >
1939 MILLISECONDS.

LIKELIHOOD=38
< SASERU8  INSTANT
    ( SELF ( A SASERU ) )
    ( AGENT = TARO )
    ( PATIENT = ZIRO )
    ( MOD = IQSYOUKENMEI )
    ( GOAL = BENKYOU5 )
    ( ASPECT = ( KANRYO ) ) >
< BENKYOU5  INSTANT
    ( SELF ( A BENKYOU ) )
    ( OBJECT = RIKA )
    ( AGENT COREF ( ^ PATIENT ) ) >
< RIKA  PROTO
    ( SELF ( A KAMOKU ) ) >
< ZIRO  PROTO
    ( SELF ( A HITO ) ) >
< TARO  PROTO
    ( SELF ( A HITO ) ) >
```

図5. 例文(3)の二つの意味解釈

(4) 2、3日たつと、その花はしぼんで
だんだん黒っぽい色に変わって
いきます。

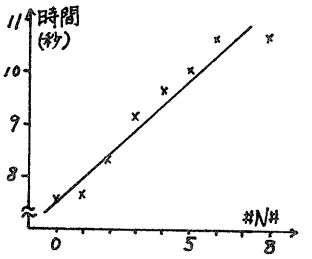


図6. 例文(4)の解釈のポーズ時間

§4. むすび

EXAM上でいろいろな例文の処理を行い、望ましい処理結果を容易に得られる事などの有効性が確かめられた。又、#N#は、高々2か3で十分であり、この事による処理時間の増加はほとんど問題にならず、従来の拡張LINGOL方式で得られなかった利点を考えあわせれば、むしろ効率は向上したと見るべきである。

MELINGは、拡張LINGOLを含むシステムであり、全く同じ使い方をすることも可能である。そのために、<sem>部は従来のままで残してある。しかし、以上述べたように、MELING独自の使い方によって得られる利点は多く、さらに、本稿ではふれなかった未定義語の処理能力等も強力であり、今後は、この上のシステム、EXAMの充実を次のような点から考えて行きたい。

<1> 「もっともらしさ」の与え方をより妥当なものにする。
<2> 文脈処理をも含むシステムとする。

特に、<2>については、いくつかの成果を得ており、システムの語彙数の増加と共に、現在作業を進めている段階である。

謝辞

日頃御指導いただく伊理正夫教授に感謝する。拡張LINGOLや貴重な文献の提供をはじめ、多大の御支援を下さる電総研の淵一博推論機構研究室長、田中穂積技官に感謝する。また、伊理研究室の溝口博君には、システムのインプリメンテーション等で多くの協力を得た。ここに記して謝意を表する。

文献

[1] 電子技術総合研究所推論機構研究室：「拡張LINGOL」, 1978.
[2] 白井英俊：「日本語文の意味解析のための生成文法的モデル」, 東京大学情報工学修士論文, 1979.
[3] 横尾英俊：「文脈的推論を含む日本語理解システム」, 東京大学情報工学修士論文, 1980.
[4] 溝口 博：「日本語文の構文解析と意味抽出の手法の研究」, 東京大学計数工学科卒業論文, 1980.
[5] 田中穂積：「計算機による自然言語の意味処理に関する研究」, 電子技術総合研究所研究報告, 第797号, 1979.
[6] Bobrow, D. G. & Winograd, T. : "An Overview of KRL, A Knowledge Representation Language," Cognitive Science, Vol. 1 (1977), No. 1, pp. 3-46.