自然言語処理　62－7
（１９８７・７・２４）

継承階層 PROLOGによる自然言語処理

赤間　清

（北海道大学　文学部　行動科学科）

ＴＡＬＫは，継承階層 prolog を基礎とした実験用の自然言語処理システムである．それは，漢字かな交じりの日本語文を入力として，意味表現を作り出す．ＴＡＬＫにおける意味解析の過程は，各部分の意味表現を統合してより大きな意味表現を作って行く過程である．その意味表現は，クラス束縛変数やインスタンスや関係名などから構成されるＳ式である．意味表現として，そのような継承階層 prolog におけるＳ式を用いていることにより，ＴＡＬＫの方法は，知識表現の記述が宣言的で容易であること，意味情報の統合過程が明快になること，組込のユニフィケーションにより推論の高速性が達成されること，などの特徴を備えている．本論文では，７つの例文を用いてＴＡＬＫにおける意味解析過程の基礎を説明する．

Natural Language Processing by the Inheritance Hierarchy Prolog : PAL

Kiyoshi Akama
(Hokkaido University, Sapporo-shi, 060, Japan)

TALK is an experimental system of natural language processing based on the inheritance hierarchy prolog : PAL. It receives Japanese sentences as inputs and makes semantic representations of the sentences. The semantic representations used in TALK can explicitly represent the each state of recognition in the process of understanding the whole sentence. Thus it makes clearer the process of semantic analysis of sentences, starting from the meanings of words in the sentence, combining them to get the meaning of the whole sentence.

1．　まえがき

本論文では，継承階層 prolog：ＰＡＬ[4]を自然言語処理に応用する可能性を探る．継承階層 prolog は，集合束縛変数 [5] という新しい変数が追加されて拡張された prolog である．集合束縛変数とは，集合データ型のデータに束縛された変数である．集合束縛変数には，いくつかの種類があり，それらはいずれも自然言語処理に有用であるが，そのうちでとくに欠くことのできないのは，クラスと呼ばれる集合データ型に束縛されたクラス束縛変数である．ここではクラス束縛変数を用いた例を示しながら，ＰＡＬによる自然言語の意味処理について議論する．

われわれが自然言語処理の研究を開始した理由の１つは，帰納的学習システムの理論を構築するために，より優れた意味表現が必要なためである．ある言語によって会話する学習システムはいったいどんな知識を獲得すべきなのだろうか．人間の学習にとって何を学習することが本当に何が大切だろうか．文法か意味かと問われれば，意味が重要であるのは明らかである．文法はどうでも意味さえわかれば，言語は立派に生きて行く役に立つからである．しかし人間の行う意味処理のなかで，何が重要なのだろうか．意味処理の大筋の構造をどうとらえるかは，学習するシステムの研究を進める上で決定的な重要性を持っている．

しかし残念ながら現在の自然言語の研究は，的確な答えを示してはくれない．多くの研究論文を見るとき，統語的な処理過程が比較的明快なのに対し，意味処理過程の実体はなかなか分かりにくい．その過程は自然言語処理プログラムの手続きの中に隠されている場合が非常に多い．また，各単語に与える意味表現のアド・ホック性の中に隠されている場合も多く，はたしてそれがどの程度の一般性ある処理によって得られるものか，たいていの場合不明瞭である．

この原因として，多くの研究者の支持できるほど強力な意味表現がまだ存在しないこと，意味処理過程を明確化する努力が足りないこと，が考えられる．もちろんそのような意味表現が本当に存在しうるかについては問題が残るが，少なくても，多くの研究者が有力と認めるいくつかの知識表現が設定され，いろいろな研究者の意味解析過程がそのもとでより具体的に記述され，もっと明確に比較できるようにならなければ，互いの研究者がそれぞれの研究成果を効果的に持ち寄って意味処理の研究を発展させていくことは難しいであろう．現在の自然言語処理の研究報告は，意味処理の分野においては，大半が粗いデッサンであり，それぞれの研究者が同じような知見を再発見せざるをえないという問題を抱えているように思われる．

まさにそのような状況認識をうけて，我々の研究は進行している．われわれは，本研究の試みが自然言語処理の研究のそのような困難を打開する１つの糸口に発展することを期待している．

2．　継承階層 Prolog による意味処理

ＴＡＬＫは，継承階層 prolog を基礎とした実験用の自然言語処理システムである．ＴＡＬＫでは，形態素解析，構文解析，意味解析の各処理を独立した処理とはせずに，文の各部分ごとにそれらを並行して行う．そしてその結果ＴＡＬＫは，漢字かな交じりの日本語文から（中間の構文解析木などを作ることなく）一気に意味表現を作り出す．構文解析の部分は，松本によって考案されたＳＡＸ[6]を拡張して利用している．ＴＡＬＫでは，それが自然に形態素解析や意味解析と融合している．そして文が与えられると，backtrack によってすべての可能な意味表現を出力することができる．本論文の主要な目的は，ＴＡＬＫで実現された例を用いながら，継承階層 prolog のＳ式に基づいてわれわれが与える知識表現と情報統合過程が，意味解析にとって非常に自然かつ有効なものであることを示すことである．ＳＡＸとの関係については別の論文で改めて議論する予定であり，構文解

析については触れない．ここでの意味表現や意味解析は，ＴＡＬＫにおける構文解析方法や融合方式などの選択とは無関係に利用可能な一般性を持っている．

継承階層 prolog：ＰＡＬとＴＡＬＫにおける自然言語処理には次のような特徴がある．

(1) ＰＡＬでは，知識表現において徹底した宣言的記述を追及している．それはシステムの動作と知識の表現における明瞭性，拡張性をもたらす．また，知識コンパイルによるさらなる高速化にも好都合である．

(2) ＰＡＬは標準 prolog の機能を含んでいるので，prolog による自然言語処理研究の知見をすべて容易に利用できる．

(3) 名詞句に対応する概念や，文法カテゴリーの文に対応する意味表現を，Ｓ式で明快に表現している．

(4) 文の意味解析過程を，基本的に，情報の統合過程としてとらえている．ＰＡＬに基づく意味表現は，情報の変化過程を記述するのに向いており，そのような観点から自然言語処理を明快に議論する手段を提供している．

(5) ＰＡＬの継承階層は単継承である．ＰＡＬは単継承のメリットを十分に引出すことができる言語環境を提供している．

(6)ＰＡＬによる表現は単継承によって不都合を生じない．従来多重継承で書いたような知識を別の方法で効果的に書くことができるからである．

(7) 安易な多重継承の利用を抑制することによって，処理の高速化が得られるだけではなく，より明快な理論の構築が期待できる．たとえば「ＡがＢを表現できる」という形の知識が継承階層から分離できる．それらは意味の研究への新たな糸口を与える可能性がある．

３．　自然言語処理のための基礎的な述語

３．１　組込述語

継承階層 prolog ：ＰＡＬの基本的な組込述語のうち，自然言語処理に関連の深いものを挙げる．

(asc *cc *pc) ：*cc がクラスであり，その親クラスが *pc であることを宣言する．

(asi *ci *pc) ：*ci がインスタンスであり，その親クラスが *pc であることを宣言する．

(subclass_of *dc *ac)：*dc はクラス *ac の子孫クラスである．

(instance_of *di *ac)：*di はクラス *ac の子孫インスタンスである．

(binding_class *v *c) ：クラス束縛変数*v の束縛クラスを *cに与える．

(class_bind *v *c)：通常変数，あるいは，クラス束縛変数 *v を クラス *c まで束縛する．*c に限定できる場合にのみ成功．

３．２　ＴＡＬＫのための述語

ＴＡＬＫのために定義した基本的な述語の概要を列挙する．ここでの説明には，次の約束をする．インスタンスや変数やクラス束縛変数などを分子と呼ぶ．分子と関係から構成されるＳ式で，ある実体に対応する表現を概念と呼ぶ．文の意味を表現するＳ式を文と呼ぶ．

bc述語：　(bc *o *c)
*o に分子または概念が与えられて，*cにクラス名を返す．*o がインスタンスの場合はそれが属するクラス，クラス束縛変数の場合には，束縛クラスを返す．概念の場合には，その第１要素の分子に対するクラスを返す．

cb述語：　(cb *o *c)
分子または概念 *o，クラス *c が与えられて，*o をクラス *c までに制限する。*o がインスタンスのとき，*oが*cに属する時だけ成功し，*oは変化しない．*o がクラス束縛変数のとき，*o のクラスが *c と交わると

きだけ成功し，*o のクラスは，それらのクラスの共通部分となる．*o が概念のとき，その第１要素の分子だけを同様に変化させて新たな概念を得る．たとえば，(a) *o = pot i，*c = animalのときは成功し，*o は変化しない．(b) *o が *x^animal，*c が dog であるとき，*oが*x^dog に変化して成功する．(c) *o が *x^dog，*c がanimal であるとき，*o が *x^dog のままで成功する．(d) *o が *x^dog，*cがcat であるとき，呼出しは失敗する．(e) *o が

(*x^animal (==>color black))，

*c が dog であるとき，*o が

(*x^dog (==>color black))

に変化して成功する．

point述語：　(point *o *c *oo)

*o は 概念，*c はクラスである．*o がクラス *c のものを表わしうるときに，結果として想定される概念を *oo に得る．

marge述語：　(marge *s1 (*k . *x) *s2)

*s1と*s2は文である．*kは格助詞または係助詞またはｚ，*x は概念である．始めの２つの引数が与えられて，それらが統合できるか検査し，統合可能なときにはその結果を *s2 に返す．これは，

文 → 後置詞句　文

なる文法に対応する意味処理で，文に後置詞句（あるいはそれに相当する情報）を追加してより大きな文をつくる場合に起動される．

m_no_m述語：　(m_no_m *mx *my *mz)

２つの名詞句Ｘ，Ｙの意味表現 *mx,*my が与えられて，名詞句「ＸのＹ」の意味表現を求め *mzに返す．

m_to_m述語：　(m_to_m *mx *my *mz)

２つの名詞句Ｘ，Ｙの意味表現 *mx,*my が与えられて，名詞句「ＸとＹ」の意味表現を求め *mzに返す．

４．　継承階層 prolog による意味処理

４．１　例文［男が女を見た］

２つの名詞は，ＴＡＬＫの辞書では，

男 -----> c_meisi man

女 -----> c_meisi woman

と記述してある（c_meisi の c はclass の c である）．これよりＴＡＬＫは，それらに対して，それぞれ

男 = (*x2^man)

女 = (*x3^woman)

という意味表現をつくる．いっぽう，ＴＡＬＫでは，動詞「見る」には次のような知識を付随させている．

```
agent  animal  ga
object object  wo
object event   wo
time   time    ni
place  place   de
```

「見た」は，まず最も小さい文と認定され，

```
( *X1^SENTENCE
    (VERB SEE)
    (MOD PAST))
```

なる意味表現が与えられる．次にmarge述語がこれに

「女を」　=　(wo . (*x3^woman))

の情報を付加えて，

```
( *X1^SENTENCE
    (VERB SEE)
    (MOD PAST)
    (object *x3^woman))
```

をつくる．このとき，動詞「見る」の定義の中の

object object wo

が使われ，

(1)　格助詞が「を」であること

(wo = wo)

(2)　「女」が「物」であること

(woman is_a object)

がチェックされたことは言うまでもない．同様に

「男が」　=　(ga . (*x2^man))

を付加えて，全体の文の意味表現：

```
( *X1^SENTENCE
    (VERB SEE)
    (MOD PAST)
    (AGENT *X2^MAN)
    (OBJECT *X3^WOMAN))
```

が完成される．

4．2　例文［太郎が花子を見た］

自然に考えつく（よくある）やり方は，「太郎」や「花子」を man あるいは woman であるとし，

「太郎」--isa--> man --isa--> human --isa--> animal

「花子」--isa--> woman --isa--> human --isa--> animal --isa--> object

の関係を用いて，「見る」と結びつけるものである．そうすれば処理の方法は前述の例文の場合と同様になる．しかしわれわれの意味表現では，「太郎」や「花子」は「人間」とは排反なクラスである「名前」にしてあるので，別の方法を用いている．それは，

「太郎」という「名前」は人間（男性）を表わしうる

「花子」という「名前」は人間（女性）を表わしうる

という知識を用いて，「太郎」や「花子」を man や womanに結びつける方法である．これを図式的にかけば次のようになる．ただし，--point--> は point述語による結合である．

「太郎」--isa--> man_name --point--> man --isa--> ...--isa--> animal

「花子」--isa--> woman_name --point--> woman --isa--> ...--isa--> object

最終的に得られる全体の意味表現は，次のようになる．

```
( *X1^SENTENCE
    (VERB SEE)
    (MOD PAST)
    (AGENT *X2^MAN
           (==>NAME TARO))
    (OBJECT *X3^WOMAN
           (==>NAME HANAKO)))
```

これを見ればわかるように，「太郎が花子を見た」は，ここでは，「太郎という名前の男性が花子という名前の女性を見た」の省略であるとみなされている．

このような方法を選ぶにあたって２点を指摘したい．１つは，多重継承に関する問題である．次の２つの文を考えてみよう．

「太郎は大学を捨てて画家の道を選んだ」

「太郎は画家と口論を始めた」

多くの研究では，上記の「表わしうる」のような工夫をしていない．そしてこの２つの文を理解するには，「画家」は「職業名」であり，同時に「人間」である必要があると結論しがちである．その背後には，

(1)　多くの研究者が多重継承のできるプログラム言語環境で自然言語処理システムを考えている，

(2)　多重継承を利用することは，単継承の場合に享受できるいろいろなメリットを捨てるという意味において，非常に大きいコストを払っていることを見逃している，

(3)　そのため，多重継承できる言語はそれができない言語より絶対的に優れていると結論していることがよくある，

(4)　単継承のメリットを実際にうまく引出せる言語環境をもっていない，

などの事情がある．われわれは，このような状況を改善し，単継承による表現の可能性を十分に追及することが，より明快な理論に至るために必要であることを指摘したい．

もう１つの注意すべき点は，「ＡはＢを表わしうる」という知識は，自然言語の現象を説明する上で非常に重要だということである．したがってそれらを「ＡはＢである」という知識（継承階層の知識）と安易に混合せずに，それ自体の詳しい分析に進むべきである．

「ＡはＢを表わしうる」という知識には，たとえば，

「犬の名前」は「犬」を表わしうる

「職業名」は「人間」を表わしうる

などがある．これらと，

「ポチ」は「犬の名前」である
「医者」は「職業名」である
という（継承階層の）知識を組合せると，意味表現は，それぞれ，

```
「ポチ」 = (*X1^DOG (==>NAME POTI))
「医者」 = (*X2^HUMAN
                (==>JOB *X3^DOCTOR))
```

となる．これによってＴＡＬＫでは，「ポチ」や「医者」を，それぞれ，DOG や HUMAN として扱うことが可能になるので，「走る」や「診る」などの動作者を動物や人間などに限定しているにもかかわらず，

「ポチが走る」
「医者が太郎を診た」

などの解釈が自然におこなわれる．

４．３　例文［太郎は花子が泳ぐのを見た］

例文中の「の」は「できごと」を表わす名詞として扱われており，「花子が泳ぐ」という文がそれを修飾している．この例文の解釈には，上記の「見る」の意味のうち，

```
object  event  wo
```

にあたる部分が用いられる．意味表現は次のように得られる．

```
( *X1^SENTENCE
    (VERB SEE)
    (MOD PAST)
    (THEME *X2^MAN)
    (AGENT *X2^MAN
           (==>NAME TARO))
    (OBJECT *X3^EVENT (==>CONTENT
  *X4^SENTENCE
    (VERB SWIM)
    (MOD NOW)
    (AGENT *X5^WOMAN
           (==>NAME HANAKO)))))
```

このうち（THEME ....）は，係助詞「は」による主題化を示すために付加えられている．この意味表現は（一見そうではないように見えるが）再帰的な構造をしている．とくに，*X1^SENTENCE の率いる全体のＳ式と *X4^SENTENCE の率いる部分Ｓ式が同じような構造をしていることに注意せよ．それは，全体のＳ式を，

```
( ? (VERB . ?) (MOD . ?)
    (THEME . ?) (AGENT . ?)
    (OBJECT ? (==>CONTENT . *SENT)))
```

とマッチさせてみればわかる（==>CONTENT の次の「.」が 重要）．そのとき *SENT は，

```
( *X4^SENTENCE
    (VERB SWIM)
    (MOD NOW)
    (AGENT *X5^WOMAN
           (==>NAME HANAKO)))
```

となり，全体のＳ式と相似である．

４．４　例文［太郎の母は見た］

ここでは「母」の扱いを問題にする．「母」は明らかに「女性」のような一般のクラスとは異なる．それは単なる「女性」ではなく，常に「その子供」の存在を感じさせるからである．そのような一群の名詞を，以下では平井ら[7]にしたがって，関数名詞と呼ぼう．継承階層 prolog による意味表現は，関数名詞を扱うのにも適している．たとえば「母」の意味は，

```
(*1^WOMAN (<==MOTHER *2^HUMAN))
```

と表現される．すなわち「母」の定義は「ある人間の母たる女性」である．この扱いを，これまでの名詞が，

```
「太郎」 = (TARO)
「女性」 = (*^WOMAN)
```

という単純な表現を与えられたのと比較しなさい．ＴＡＬＫではこのように，文の意味だけでなく基本単語の意味も，インスタンスやクラス束縛変数などではなく，より複雑なＳ式で与えられることがある．

「太郎の母」を解釈するのに必要な意味処理について述べる．その処理は m_no_m という述語でなされている．それは名詞句と名詞句が「の」で結ばれて「ＸのＹ」の形で構成

する名詞句の意味表現を与える述語である．Xの意味表現を *mx，Yの意味表現を*my としよう．もし，*my が

(*s (*f . *ys) . *r)

の形をしており，*f が関数名詞などの定義に使われる「関数」のリストに属しているならば，述語 m_no_m はまず *ys の束縛クラスを調べ（bc述語 を使う），もし *mxが*ysと同じものであるという解釈が成立しうるときには，その解釈を利用して *ys を *mx で表現し直す（point 述語を用いる）．この例の場合，

```
*mx = (TARO)
*my = (*X1^WOMAN
          (<==MOTHER *X2^HUMAN))
```

である．<==MOTHER は上記の「関数」リストにのっている．また，TARO は MAN_NAMEであり，MAN を表わし得る．MAN は HUMAN であるから，point 述語は，

```
(*X2^HUMAN)
```

を

```
(*X2^MAN (==>NAME TARO))
```

で置き換える．かくして，「太郎の母」の意味として，

```
(*X1^WOMAN
   (<==MOTHER *X2^MAN (==>NAME TARO)))
```

が得られる．

4．5　例文［象の鼻は長い］

「象の鼻」の意味表現をつくるために，「XのY」を処理する述語 m_no_m が起動される．述語 m_no_m は，この場合 述語 xnoy_y の知識を呼出す．xnoy_y は，

```
(xnoy_y animal_body <==part_of animal)
(xnoy_y plant_body <==part_of plant)
```

などが成立つように定義されている．この意味を前者の節によって説明する．その節のおおよその役割は，述語 m_no_m に知識を供給して，「XのY」の背後に「動物」の「体」という構造を探し，それが発見できれば，「XのY」全体の意味として，「その動物の一部分である体」というような意味表現を作ることを可能にすることである．Xは必ずしも「動物」である必要はなく，point 述語で動物と関係がつけばよい．このことはYと「体」についても同様である．これによってたとえば，「医者の手」などの表現も扱うことができる．すでにのべたようにＴＡＬＫでは「医者」は「動物」ではなく，「職業名」である．ＴＡＬＫによって得られる意味表現は次のようになる．

```
(*3^sentence
   (verb long)
   (mod now)
   (theme *2^nose)
   (<==length
          *2^nose
            (<==part_of *1^elephant))
   (comp *0^long))
```

4．6　例文［象は鼻が長い］

これは，「象の鼻が長い」とほぼ同じ意味になる．しかし「は」の扱いは「の」の場合よりも難しい．その理由は，たとえば「象はむかし鼻が長くなかった」のように「は」が（いくつかの）単語を飛越して２つの対象（ここでは象と鼻）を結びつけるからである．

```
(*3^sentence
   (verb long)
   (mod now)
   (theme *1^elephant)
   (<==length
        *2^nose
          (<==part_of *1^elephant))
   (comp *0^long))
```

4．7　例文［鼻が長い象を見た］

これも同じで「鼻が長い」という文と「象」の関係は，「象は鼻が長い」の場合と同じ処理が必要である．このように文が名詞句にかかる場合，それらをつなぐ関係名は便宜上 z

を用いている．
また，修飾される対象（名詞句）が修飾する文においてどのような役割を果しているかを明確にするために，その対象（ここでは *1^elephant）を修飾文（ここでは *3^sentence が率いる文）のなかに登場させている．

```
(*4^sentence
   (verb see)
   (mod past)
   (object
      *1^elephant
        (z *3^sentence
             (verb long)
             (mod now)
             (<==length
                  *2^nose
                    (<==part_of
                        *1^elephant))
             (comp *0^long))))
```

4．8　例文［太郎は花子が青いりんごを食べたのを知らない］

「青い」という形容詞には，普通の動詞などと同じく１つの文としてのコストをかけている．これは，「青いりんご」のところが「昨日畑で青かったりんご」などと変化しうることを考えればその必要性が分かる．

```
(*7^sentence
   (verb know)
   (mod now not)
   (theme *6^man)
   (agent *6^man (==>name taro))
   (object
      *5^event
        (==>content
 *4^sentence
   (verb eat)
   (mod past)
   (agent *3^woman
          (==>name hanako))
   (object *1^apple_food
           (z
  *2^sentence
     (verb blue)
     (mod now)
     (<==color *1^apple_food)
     (comp *0^blue))))))
```

5．　むすび

継承階層 prolog：ＰＡＬに基づく意味表現は，これまでのいくつかの優秀な研究の成果を統合し発展させる上で，重要な役割を果し得ると考えられる．

文　献

［１］　赤間清：継承階層コンパイラ，情報処理学会，知識工学と人工知能研究会資料，48-4,pp.25-32 (1986)

［２］　赤間清：継承階層 prolog と多重継承，日本ソフトウエア科学会第３回大会，B-5-2, pp.189-192 (1986)

［３］　赤間清：概念階層クローズ・インデキシング，情報処理学会，知識工学と人工知能研究会資料，51-2, pp.9-16 (1987)

［４］　赤間清：ＰＡＬ：継承階層を扱う拡張PROLOG，情報処理学会論文誌　Vol.28 No.4 pp.27-34 (1987)

［５］　赤間清：集合束縛変数に基づく意味表現とユニフィケーション，情報処理学会，自然言語処理研究会資料，（本号）(1987)

［６］　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・ソフトウエア，Vol.3, No.4, pp.4-11 (1986)

［７］　平井誠，北橋忠宏：日本語文における「の」と連体修飾の分類と解析，情報処理学会自然言語処理研究会，58-1 (1986)