自然言語処理 35－5
（1983. 1. 25）

機械翻訳システムLUTEにおける意味構造変換処理

片桐恭弘　野村浩郷
（日本電信電話公社　武蔵野電気通信研究所）

1. はじめに

機械翻訳における構造変換処理は，原言語に対する内部表現と目的言語に対する内部表現との間の違いを吸収する機能を持つ必要がある。内部表現の形式は，対象とする言語ごとに異なり，また言語解析・生成の手法とも密接に関連して決定される。一般にヨーロッパ語相互のように近い言語間の翻訳では内部表現の構造も類似しており，小規模な構造の置換が主となるが，日本語・英語間のように離れた言語間では大規模な構造の置換が必要となる。さらに構造変換では両言語の言語構造の違いだけでなく，各言語に対する内部表現の形式的な違いをも吸収しなければならない。また実用的な翻訳システムを目指す場合，解析の中間段階での構造変換も必要となってくる。従って構造変換自体は特定の内部表現の特徴に依存せず，解析・生成処理と独立して任意の構造相互の変換を許容する広い枠組を提供するものでなければならない。多言語間相互翻訳を構造変換方式で行おうとする場合には特にこの点が重要となる。また構造変換処理では最初からすべての仕様が決定できない場合が多いため，処理手順変更の容易さも要求される。これらの要求を満たすためには，構造を取り扱うための規則記述のシステムを構築し，その上で構造変換処理を行うのが最適である。規則は記述が容易であり，かつ強力でなければならない。

また，機械翻訳においても他の自然言語処理と同様，高品質の処理を目的とした場合，各種の言語的・非言語的知識の利用が必須となる[1]。このためには知識表現あるいは問題解決の手法と構造変換処理との枠組とを融合させる必要がある。

このような観点から，我々は知識表現のためのフレーム表現言語を土台としてその上に構造変換規則を用いた構造変換処理システムを構成した。これは，日英，英日翻訳を行うLUTE実験システムにサブシステムとして組み込まれている。ここでは解析・生成には触れず，構造変換処理に限定して議論する。

2. LUTE構造変換処理システムの特徴

(1) 知識表現のためのフレーム表現言語を基礎として用いる。
一般的知識のみならず，言語的知識もフレームによって統一的に表現される。例えば，各言語カテゴリーはクラスフレームとして表現され，文の意味表現はそれらのインスタンスフレームの作るネットワークとして表わされる。

(2) message passing の概念を取り入れる。
各フレームに手続きを付し，フレームにメッセージを送ることによってそれを起動する。構造変換は入力文解析結果のインスタンスフレームに対して変換用メッセージを送ることによって行われる。全体の処理の制御は Top-Down 型になる。

(3) 構造変換規則は直観的に理解しやすい if_then_else 型の syntax に従って記述する。構造を扱うために規則の中にフレームネットワークのパターンを記述する。構造変換規則は各言語カテゴリーに変換用メッセージに対する手続きとして付与される。

3. フレーム表現言語

フレームは良く知られているようにMinsky[2]によって知識を表現するための枠組として提案された概念であり，知識をまとまり(chunk)として捉えようとする思想に基づいている。物・状態・行為・出来事等あらゆる物事がフレームと呼ばれるノードによって表現される。各フレームはスロットと呼ばれる属性名－属性値対を任意個有しており，それによって他のフレームとの関係が示される。我々はフレーム表現言語自体を研究の中心課題としたわけではないので，既存のフレーム表現言語FRL[3]の簡略版としてWinstonの紹介しているものに若干変更を加えて用いている[4]。従って各フレームはフレーム名を示すアトムに対して，slot，facet，valueの三重のA-listを属性リストとして付した構造をしている。ベース言語としてはMacLispを用いた。

階層構造と属性遺伝　フレームの特徴の一つとして，フレーム間の階層構造とそれを利用した属性の遺伝がある。階層構造を作るための関係は様々に存在するが，我々はまず，クラスとインスタンスとの関係を示すinstance-of関係と，クラス間の包含関係を示すAKO(A Kind Of)関係とを区別し，図1のようなフレーム間階層構造を用いることにした。属性遺伝は図中の矢印に従ってなされるものとした。従って太郎というフレームの特定のスロット値をアクセスしようとした時，もし太郎のフレーム自身にそのスロット値が見つからない場合には人間，動物，生物の順にスロット値の探索がなされる。また，すべてのクラスからなるクラスを表わすフレーム*CLASS*を最上位に置いた。これはクラスに対して一般的に成立する属性を保持している。

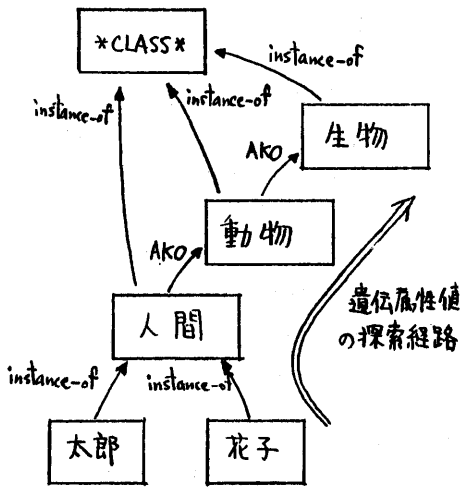

図1. フレームの階層構造.

手続き付与とmessage passing　もう一つのフレームの特徴としてスロットあるいはフレームへの手続きの付与がある。これにはservant/demon，trap/triggerの区別[5]やIF-ADDED，IF-NEEDED等機能別の分類があるが，我々は各フレームをオブジェクト指向言語[6]におけるオブジェクトと同等に見なして各フレームにメソッド手続きを付与し，メッセージを送ることによってそれを起動するようにした。ただし，図2に示すようにメソッド手続きはクラスフレームに対して付与し，インスタンスフレームがメッセージを受け取った時にはそれの属するクラスから対応するメソッド手続きを探して起動する。メソッド手続きの探索はAKOリンクを伝わって行うため，一般的な手続きは階層の上位に位置するフレームに付けておくことができる。最上位フレーム*CLASS*には新しくクラスのインスタンスを作るためのメッセージnew:，スロット操作のためのメッセージget:，put:，remove:に対応するメソッド手続き等が用意されている。

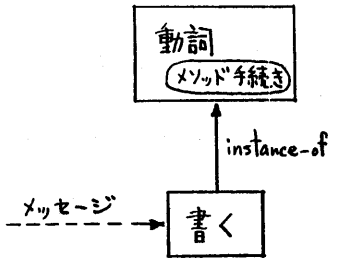

図2. メッセージによる手続きの起動.

4. 変換規則記述言語

構造変換処理で行われる基本操作は原言語文解析結果から適当な部分構造を抽出し，それに対応する目的言語内部表現の部分構造を作り出すことである。従って構造変換処理に必要とされるのは，(1)パターンマッチング処理，(2)新らたな構造の生成の二点である。この他に，これらの基本操作を積み重ねて行うために処理の制御方式を決定する必要がある。

上記の二つの基本操作を組み合わせるとcondition-action対を構成する。我々はif_then_else型の規則記述syntaxを用いてパターンマッチングと新らたな構造の生成とを表わすこととした。一般的な規則のフォーマットは次のように表わされる。

```
(defrule <type> (if <matching pattern> then <action>))      または
(defrule <type> (if <matching pattern> then <action> else <action>))
```

<type>には変換規則を付与すべき言語カテゴリーを示す。

パターンマッチング処理　<matching pattern>にはこの規則の適用されるフレームサブネットワークの構造を各フレームごとに次のような形式で記述する。

```
(<frame variable> = (<category name>
                     (<slot name>  <slot value specification>)
                     (<slot name>  <slot value specification>) ... ))
```

パターンには任意個のフレームを記述することが出来る。パターンに記述されたすべてのフレームについて，カテゴリーおよびスロット構造がマッチした時に限りパターンマッチングは成功し，規則の<action>部が実行される。

<slot value specification>には定数，変数の他に，有っても無くても良い要素の指定や，任意個の要素とマッチする変数を用いることが出来る。また，論理演算子AND・OR・NOTや，クラス帰属性を示すA(AN)を用いることによって複雑な条件を指定することも可能である。<slot value specification>中の変数は対応するスロット値と束縛され，それ以降のパターンマッチングおよび<action>部の実行の際に用いられる。パターンの代わりあるいはパターンと同時に任意のLisp式を用いて規則の適用判定を行うことも可能である。

新らたな構造の生成　<action>部には新らたにインスタンスフレームを生成するのに必要な情報を次のようなフォーマットで記述する。

```
(exec <instructions> where <new frame pattern>)
```

<instructions>部では局所変数の定義，部分構造の変換，Lispプログラムの実行，最終的に作るべきインスタンスフレームの指定等を行い，<new frame pattern>部に新らたに作られるフレームネットワークの構造を記述する。フレームネットワーク構造の記述は<matching pattern>と同様のフォーマットによって行うが，<slot value specification>部では定数，変数の他に(! <category name> <variable>)という形式を用いることによって新らたなインスタンスを作り出すことを示す。パターンマッチングの場合と同様，<new frame pattern>部には任意個のフレームを記述することが出来る。

規則の中の<action>部あるいは<instruction>部にif-規則を埋め込むことによって単純なcondition-action対だけでなく，任意に複雑な規則を作ることも可能である。図3に例として日英翻訳における単文の変換規則の一つを示す。selfはパターンマッチングの始点となるフレームを示している。

```
; An example transfer rule for Japanese simple sentences

(defrule J:SSENT
    (if (self = (J:SSENT                ;matching pattern determines
                 (*predicate            ;whether this rule can be
                  (AND (A J:PREDICATE)  ;applied to the current
                       (var pred)))     ;instance or not.
                 (*case (rest cases))))
        then
        (exec
         (local (rest E-clauses))       ;local variable declaration
         (pred -> E-pred)               ;recursively invoking transfer
         (((for-all)cases) -> E-cases)  ;processes for substructures
         (LISP
          (:= E-clauses                                 ;Lisp code to pick
              (for (X in E-cases)                       ;up cases that were
                   (when (isa* X 'E:MODIFIER-CLAUSE))   ;transferred into
                   (save X)))                           ;English clauses
          (for (X in E-clauses)
               (do (delq X E-cases))))
         (return (! E:SENTENCE E-sent)) ;specify resulting instance frames
         where                          ;and their slot structures
         (E-sent = (E:SENTENCE
                    (*main (! E:PREDICATE E-pred))
                    (*modifier E-clauses)))
         (E-pred = (E:PREDICATE
                    (*instance (! E:INSTANCE E-inst))))
         (E-inst = (E:INSTANCE
                    (*case E-cases)
                    (*self-instance E-sent))))))
```

図3. 構造変換規則の例.

5. 語彙変換辞書

構造変換処理ではシステム全体が統一的にフレーム表現を用いているので，個々の単語もそれぞれが属する品詞のクラスのインスタンスフレームとして表現されている。従って単語レベルの置き換えも構造変換規則を用いて統一的に記述することが可能であるが，単語は数が膨大となるため，語彙変換辞書を別個に設け，構造変換と語彙変換とを区別することとした。変換規則からは辞書検索のためのLisp関数を呼ぶことによって訳語の選択を行う。

語彙変換辞書エントリーは訳語対ごとに作られ，図4に示すようにエントリー名と日英両単語の辞書エントリー名，品詞名を記述する。これは処理の時点では図5に示すように原言語辞書項目のフレームと目的言語辞書項目のフレームとを媒介フレームを介してポインタで結合した構造になっている。媒介フレームには訳語選択に関する各種情報を付すことができる。例えば動詞の翻訳の場合，格構造の変換パターンや，格要素の意味素性に関する制限条件等を記述する。辞書検索関数はこれらの情報を参照して適当な訳語を選択する。

```
(defentry tegami-letter
          (tegami MEISHI)
          (letter NOUN))

(defentry kaku-write
          (kaku-write DOUSHI)
          (write VERB))
```

図4. 語彙変換辞書エントリーの例.

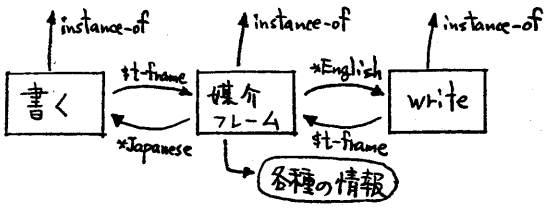

図5. 語彙変換辞書の構造.

6. 構造変換処理

構造変換処理はLispの中で行われるが，あらかじめ両言語で用いられている言語カテゴリーをクラスフレームとして定義しておく必要がある。それらに4節に示した構文に従って定義した構造変換規則を変換用メッセージtransfer:に対応するメソッド手続きとして付与する。変換規則はif_then_else型からLisp関数に変換した後，必要ならばコンパイルし，クラスフレームに付与する。さらに語彙変換辞書を読み込み，辞書エントリーのフレームを作ることによって構造変換処理の環境が出来上がる。原言語入力文解析結果の最上位インスタンスフレームに対してtransfer:メッセージを送ることによって構造変換処理が開始される。

transfer:メッセージを受け取ったインスタンスフレームは自分の属する言語カテゴリーフレームに対応する変換規則が有るかどうか調べる。一般に各言語カテゴリーは複数個の変換規則を有している。それらは定義の順に従ってメソッド手続きとして付されており，その順に従って適用可能かどうか判定される。パターンマッチングに成功し，適用可能と判定された変換規則が見つかった場合にはその規則の<action>部が実行される。<action>部では通常，部分構造フレームに対して再帰的にtransfer:メッセージを送ってそれらを変換し，得られた結果をまとめて新らたな目的言語部分構造を作り出す。変換規則の探索はAKOリンクを辿って上位のカテゴリーフレームに対しても成される。従って一般的な規則は階層の上位に，特殊な規則は下位に付ければ良い。また同一カテゴリー内でも，より特殊な場合を扱う規則は初めの方に定義しておくことによって最初にチェックされるようにする。適用可能な変換規則が見つからない場合には構造変換は失敗する。

今，例文として「太郎が寝ていた時，次郎は花子に手紙を書いていた。」を考えてみよう。LUTEシステムの日本語解析部では日本語入力文を格構造を基本とした内部表現に解析するため，上述の例文に対しては日本語解析の結果図6に示す構造が得られる。最上位フレームは単文であるため，それにtransfer:メッセージを送ると日本語単文クラスフレームから対応するメソッド手続きの探索が始まる。この場合，図3に示した変換規則が適用可能と判定される。パターンマッチングが成功し，変数predは述語1に，変数casesはリスト（格要素1...格要素4）に束縛される。規則の<action>部では，まず部分構造predすなわち述語1の変換が行われる。これは実際には述語1に対してtransfer:メッセージを送ることに対応する。この結果は変数E-predに束縛される。次に格要素が変換されるが，これはやはり格要素1...格要素4に対してtransfer:メッセージを送ることによってなされる。格要素1以外は一単語のみからなるため，辞書検索を行い，対応する英語の格要素を作り出す規則によって処理される。「太郎が寝ていた時」に対応する格要素1に対しては別の変換規則が適用され，これは英語の修飾節構造に変換される。格要素変換の結果は変数E-casesに束縛されるが，修飾節構造は別扱いするために，一旦変数E-clausesへ移しておく。英語の複文を

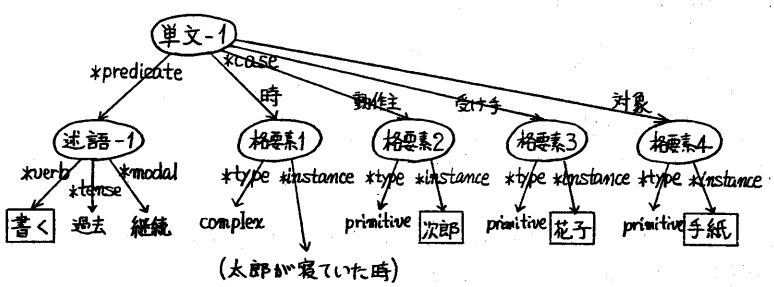

図6．例文「太郎が寝ていた時，次郎は花子に手紙を書いていた。」の解析結果。

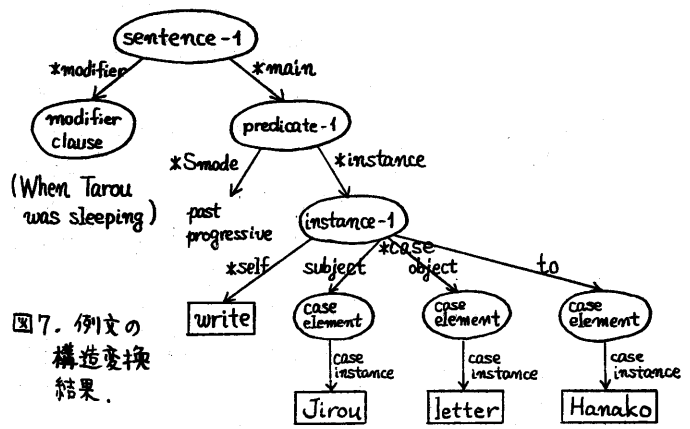

図7. 例文の構造変換結果.

示すSentenceのインスタンスフレームを一つ作り，E-clausesに保持されていた英語修飾節構造を*modifierスロットに入れる。残りの格要素と述語E-predから主文の構造を作り，*mainスロットに入れる。最終的に得られる構造を図7に示す。この内部構造から英語生成によって，訳文"When Tarou was sleeping, Jirou was writing a letter to Hanako."が得られる。

7. おわりに

本稿では，フレーム表現による言語の意味表現を基礎として，if_then_else型の構造変換規則を用いる構造変換システムについて述べた。一般的なネットワーク構造をパターンとして扱うif_then_else型規則は直観的にも理解し易く，記述も容易である。またフレーム表現は言語の意味表現のみならず知識表現一般に用いられる一般的な枠組である。以下に現状での問題点を列挙する。

(1) 不定個のフレームとマッチするようなパターンが記述できない。
(2) 部分構造の変換に上位の構造の特徴が影響する場合処理が煩雑になる。
(3) 熟語の翻訳等構造変換と語彙変換の境界の取扱いが分かりにくい。

(1)についてはそのようなパターンを用いる場合は余りないため必要性は少ない。(2)(3)はTop-Down制御に起因する問題点であるが，いずれも上位構造に対する変換規則を用いれば一応解決できる。しかし，変換の情報を付与するには直観的に分り易い場所があり，また処理効率を考えるとすべてを上位構造に対する規則とするには無理がある。いずれにせよ構造変換では，必要となる情報を，その情報自体として，用いられる場所とは独立に最も適切な場所に置き，システムのモジュラリティを損わずにそれを必要となる場所から自由にアクセスできるメカニズムを用意することが重要である。最後に，日頃御指導戴く畔柳基礎研究部長，橋本統括調査役，山下第一研究室長に謝意を表する。

参考文献

1. J. G. Carbonell, R. E. Cullingford, and A. V. Gershman, "Steps toward knowledge-based machine translation," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-3, pp. 376-392, 1981.
2. M. Minsky, "A framework for representing knowledge," in P. H. Winston (ed.), The psychology of computer vision, New York, McGraw-Hill, 1975.
3. R. B. Roberts and I.P. Goldstein, "The FRL primer," MIT AI Memo 408, 1977.
4. P. H. Winston and B. K. P. Horn, LISP Chap. 22, Addison-Wesley, 1981.
5. D. G. Bobrow and T. Winograd, "An overview of KRL, a knowledge representation language," Cognitive Science, vol. 1, pp. 3-46, 1977.
6. Xerox Learning Research Group, "The Smalltalk-80 system," BYTE, vol. 6, no. 8, pp. 36-48, 1981.
7. 島津明，内藤昭三，野村浩郷，"機械翻訳システムLUTEの日本語意味解析," 情報処理学会自然言語処理研究会資料，NL33-6，1982.