自然言語処理 35－6
（1983. 1. 25）

語法規則の翻訳にもとづく日本語からの村械翻訳

池田尚志（電総研）

1. はじめに

良質の村械翻訳をめざすためには、原言語の表現を目標言語の表現にさまざまに対応させる規則を記述でき、かつ見通しよくしかも効率的に処理できるしくみを工夫することが必要である。そのためには原言語の文を充全に分析すること、目標言語の表現への対応を明示的に規則化することが当然前提となる。

目標とするところはあくまでも村械翻訳であり、行間の推察や微妙なニュアンスまで含めて翻訳し、手直しの必要もないすぐれた文体に訳出することではない。そのような翻訳は、共感するという感情と創造性を持った人間にしてはじめて有り得る業である。村械翻訳の目標は、大量の自然言語情報や速報性の必要のある自然言語情報あるいは定型的な自然言語情報などに対して、意味の骨格を間違いなく伝達し、できるだけ自然に読める文体に訳出するという、いわば血も涙もないまさに村械的な翻訳である。そのような翻訳に対する需要は増大しつつあり、それを実現するための計算村環境は整いつつあるようにみえる。

筆者は、語句と語句の結合関係の記述を基本においた係り受け解析方式の日本語文分析システム(JAS)を開発してきた。[1,2,3] これは、個々の語の用法や仂らき(語法)の記述があって、文の解釈はその評価・相互作用で進行するというイメージに基って、語彙と文法を統一して記述するという立場を指向したものである。

JASは村械翻訳システムへの応用を特に意図したものではないが、今回、翻訳部内を作成しJASの分析結果からの翻訳を試みた。この翻訳システムは日本語の語法規則に対応して翻訳規則を記述しておいて、分析された各部分構造に対応する翻訳規則を見出して評価していくという方式であり、構造変換方式といえよう。ただし、変換された構造は、目標言語の文を分析したとき得られるであろう構造とは異なり、目標言語を生成するための、翻訳に独自の構造である。目標言語としては、現在英語を設定しているが、翻訳規則と若干の翻訳プログラムを書き直せば他の任意の言語を設定することが可能である。翻訳部内のプログラムは、現在、Lispで数百行という程度である。(なお、JASを含めて全てのプログラム及びデータはLisp系言語Petit[7]で記述されている。)

2節でJASの概略について述べ、3節で翻訳システムについて述べる。

2. 構文・意味分析システム(JAS)

JASは、語の格構造記述を基本においた係り受け解析によって日本語文の構文・意味分析を行っている。村能語によって基本の文型(格構造)が変形されたり(助動詞)、係りと受けの関係が規定されたり(助詞)するという日本語文の基本的な枠組は分析プログラムの中に組み込まれており、個々の語句の用法(格構造等の結合構造)や仂らきは辞書に記述されている。つまり文法の基本の枠組はプログラムに固定されていて個々の語法がデータベース化されている。

分析プログラムは、次の3つのフェーズから成っている。

① 文節部内

入力記号列から文節、自立語＋(付属語)* を切り出し、概念語＋(材能語)* の形に整える。つぎにその概念語の格構造を辞書からコピーし、必要な変形を施こす。（この単位を句という。）

②文部内

出力スタック上の句と、文節部内からの句との間に係り受け関係があれば結合して（なければそのまま）出力スタックに積む。このようにして文の構文・意味構造をボトムアップに分析する。

③文脈部内

単位文間の関係や前文との関係などから省略語や指示語の同定、材能語のスコープなどの意味分析を行なう。この部内は未開拓の部分が大きい。(b)

図1-bに見るように 句はすなわち解析木のノードである。

JASでは、同音語や多義語など複数の解釈に関しては構造探索を行なうが句と句の結合の際算出される結合の強さ(理解のレベル, gness)がある閾値より低い句は途中で捨てられる。またこの閾値を制御することによって、ある種の'おかしな'文も分析できるようになっている。

JASでは、原則として文はバックトラックせずに文頭から逐次に理解されていくという仮定に基いた分析を行っているが、連体うめこみ文や「ノ」による名詞連続については総当探索を行っている。

JASのデータベースは表1に示すようなものである。これらは全てキーインデクスファイルになっており、データベース管理プログラムを通して作成され修正される。これらのデータのう

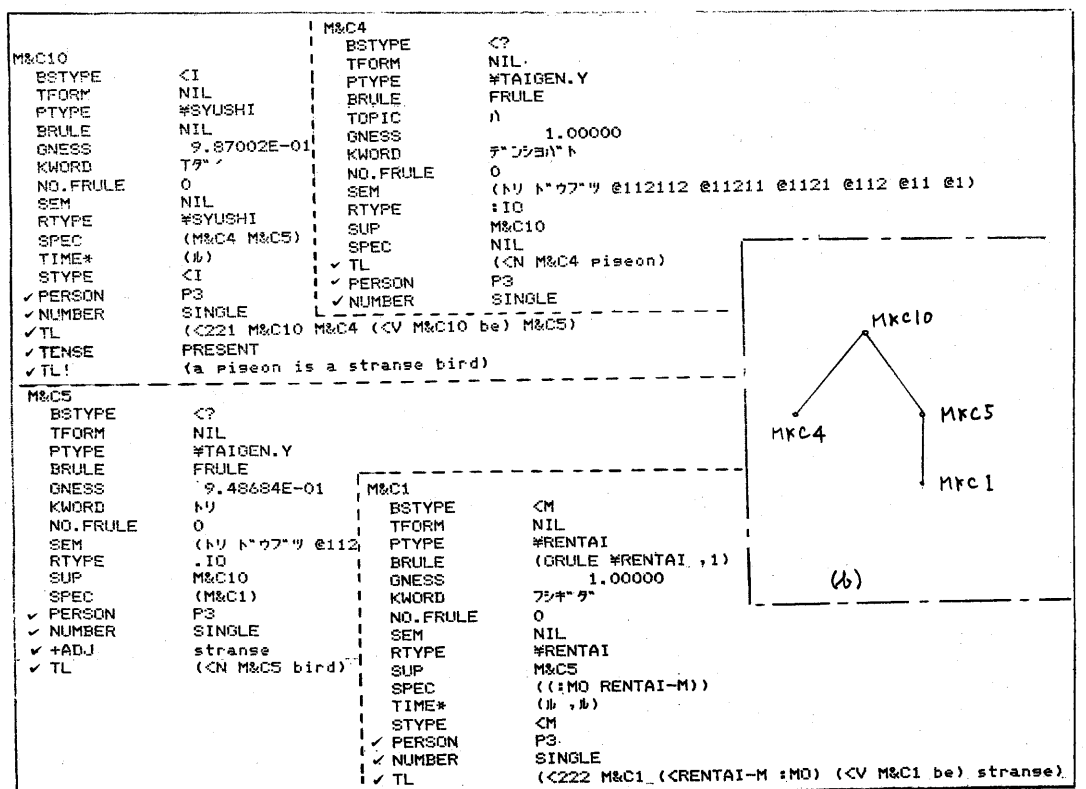

図1.「ﾃﾞﾝｼｮﾊﾊﾞﾄﾊ ﾌｼｷﾞﾅ ﾄﾘﾀﾞ」に対する分析・翻訳結果
（✓印の項は翻訳システムの処理結果）

ち、語基とそれに対応する内部コードの対はあらかじめ読み込まれるが、他のデータは、必要になった時点でシステムが自動的に読み込む。

JASでは句と句の依存結合関係を調べることが分析の中核となっているが結合規則には表2に示す7つの種類がある。①～⑤の規則が分析システムの中でこの順に評価される。結合は直接には⑥,⑦の規則を通じてあるいは直接①～⑤の中で為される。すなわち狭義の結合規則は3種に分かたれる(表3)。結合規則がこのように階層化されているので例外的な規則の扱いが容易である。実際には大部分の結合規則が格構造として記述され⑤で起動される。

	手続的規則(i.e.プログラム)	宣言的規則
受けの規則	①概念語の属性としての結合規則 ②句のカテゴリ -----	⑥概念語… (≒格構造)
係りの規則	③概念語 ---- ④付能語 ---- ⑤句のカテゴリ ---	⑦概念語…

表2. 結合規則の種類

FRULE	表2の⑥による結合
MRULE	表2の⑦による結合
GRULE	①～⑤の中での直接の結合

表3. 狭義の結合規則

ファイル名(モジュール名)	見出し	内容
/DICT	データファイル名	データファイル中の属性リスト
/BUNKEI	文型名	文型
/SEM	意味素性名	上位下位の意味素性
/RULE(/GLOBAL)	定項名	システム中で使う定項の値
(/KRULE)	活用型名	活用表
(/CRULE)	活用形名	句のカテゴリ名
(/PRULE)	句のカテゴリ名	結合規則等，翻訳規則
(/TRULE)	変形規則名	付能語の属性となる変形規則
(/ARULE)	格役割名	格を支える付能語のリスト
(/RRULE)	格条件項名	格構造中の条件記述
/ANNEX	付属語	内部コード，語幹，活用型，接続規則，整形規則，変形規則，結合規則，翻訳規則
/IDIOM	熟語	内部コード，語幹，熟語記述，活用型，翻訳規則
/LEX	自立語	内部コード，語幹，活用型，意味特徴，結合規則，翻訳規則
/TL (/VRULE)	(英語)動詞	活用表
(/NRULE)	(英語)名詞	複数形，格変化表

表1. JASと翻訳システムのデータベース

3. 翻訳システム

3.1 翻訳木構造の生成と翻訳文の生成

翻訳システムは、①入力文の木構造の各部分木を翻訳して翻訳文の木構造を得る過程と、②その翻訳木構造から翻訳文を生成する過程の2つのフェーズから成る。翻訳木構造は、実際には、

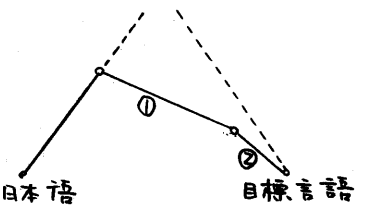

図2. 翻訳システム

入力文の木構造の各ノードに翻訳に関する諸情報を付加したものである(図1)。

翻訳木構造の生成は、入力文の木構造を形成している結合規則に対応する翻訳規則をとり出して評価し、それにしかるべき変形操作を施こしていくという処理を基本としており、入力木構造に対してトップダウンに帰納的に行なわれる。

翻訳木構造は一般にプログラムの木であり、それを順次に評価してつなぎあわせることによって翻訳文が生成される。

図3に翻訳木構造生成の手順を示す。手順②と④は用言句以外の場合は当然スキップされる。

①その/ードの中核語の格構造に対応する翻訳規則を評価し翻訳句をとり出す。
②法属性 mod.A, mod.M, mod.D に応じて翻訳句を変形する。
③翻訳句に現われる下位のノードの翻訳木構造を展開する。
④否定文、質問文、命令文などに応じて翻訳句を変形し、時・数・人称などに関連する処置を施こす。
⑤格構造以外の結合規則で結合されている残りの従属句(副詞句、連用句、仮定句、その他)に対応する翻訳規則を評価する。

図3. 翻訳木構造の生成手順

3.2 結合規則と翻訳規則

JASでは、句と句は表3に示した3種の(狭義の)結合規則のいづれかで結合されるが、いづれの場合でもどの結合規則が成立して為された結合であるのかということは解析木の各ノードに属性の1つとして印されている。データベースには各結合規則に対応して翻訳規則が記述されているが、翻訳規則は全てプログラムである。翻訳システムは対応する翻訳規則をとり出して評価する。すなわち対応する翻訳は全て2段階の選択を経て得られることになる。

FRULE ⟷ TL-FRULE
MRULE ⟷ TL-MRULE
GRULE ⟷ TL-GRULE

図4. 結合規則と翻訳規則

図5にTL-FRULEの例を示す。TL-FRULEによる翻訳は図3①でとり出される。評価した結果(翻訳句)は目標言語での対応する表現とその表現のパターン名との対である。このパターン名は図3の②,④での処理に際して場合を選択するキーとして機能することもできるし、また関数として定義しておいて翻訳文生成の際に必要な処理を施こすこともできる。どのような表現パターンを区別しておく必要があるか未だ定かでないが用言については当面Hornbyによる文型パターンを参照していく予定である。また体言については少くとも主名詞の位置による区別は必要である。{ex. 大文字→ big letter, 義母→ mother in law}

```
ハトゴヤ
  ¥TL-FRULE   '(<N pigeon box)
フシギダ
  ¥TL-FRULE   '(<222 :MO be stranse)
              '(<000 :MM feel .MO stranse)
  ¥FRULE      (<M (:MO >P1 @1))
              (<M (:MM >P1 @112112) (.MO >P1 @1))
フル
  ¥TL-FRULE
(PROG
 (¥C)
 (SETQ ¥C (ASSR '¥C SPEC))
 (RETURN
   (COND ('¥C
          (CASEQ (GET ¥C 'KWORD)
                 ((ツレテ)
                  '(<18? :TT take .LO (<FROM .TS) (<TO .TE)))
                 ((モドル)
                  '(<18? :TT (come back) (<FROM .TS) (<TO .TE))))
          (T
           '(<18? :TT come (<FROM .TS) (<TO .TE)))))))
  ¥FRULE
(<T (:TT >P1 @112112) (.TS >P1 @15 @123) (.TE >P1 @15 @123))
```

図5. TL-FRULEの例

図6にTL-MRULEとTL-GRULEの例を示す。これらの規則は図3⑤でと

り出され評価される。図6中の変数MKCは現在処理しようとしている従属句そのものを指し、TLMAINは図3の④までで得られている翻訳句つまりMKCが係っていく句の翻訳を指す。また関数TL-PCRは翻訳木構造を生成するプログラムである。これらの規則の翻訳結果は、その従属句の翻訳を含んだ新たな翻訳である。

```
ﾁｬﾝﾄ
  ¥TL-MRULE  (LIST TLMAIN 'correctly)
  ¥MRULE     (_A >P1 @2)
ﾂﾙﾃ
  ¥TL-GRULE
 (LIST TLMAIN '(accompanied by) (TL-PCR M&C))
  ¥DEP-M
 (GRULE
   (COND ((MEMQ %C-F &YOUGEN)
          1.0000000E+00))
   ¥C. ﾂﾙﾃ 1)
ﾄﾉ
  ¥TL-GRULE  (LIST TLMAIN 'of (TL-PCR M&C))
  ¥DEP-M     (GRULE
               (COND ((MEMQ %C-F &TAIGEN)
                      2.9999923E-01))
               '? ﾄﾞﾚ 1)
ﾄﾔ
  ¥TL-GRULE  (LIST (TL-PCR M&C) 'and TLMAIN)
  ¥DEP-M     (COND (%H-F
                    (GRULE (HEIRITSU?)))
                   ((NULL ISTACK)
                    (GRULE 1.0000000E+00 .& ﾄﾔ 1)))
¥RENTAI.
  ¥TL-GRULE
 (PROG
  (M&C.SUP RROLE TLSUB)
  (SETQ TLSUB (TL-PCR M&C)
        RROLE (CADR (ASSQ 'RENTAI-M TLSUB))
        M&C.SUP (GET M&C 'SUP))
  (CASEQ (CAR TLSUB)
      ((<222)
       (PUT M&C.SUP (NTH 4 TLSUB) '+ADJ)
       (RETURN TLMAIN))
      (T
       (RETURN (LIST TLMAIN (<ANT M&C.SUP RROLE) TLSUB)))))
  ¥DEP-M
 (COND ((MEMQ %C-F &TAIGEN)
        (OR
          (FRULE)
          (GRULE (RENTAI-M?)))))
¥RENYOU
  ¥TL-GRULE  (LIST (TL-PCR M&C) '(, and) TLMAIN)
  ¥DEP-M     (COND ((NOT (MEMQ %C-F &YOUGEN))
¥KATEI
  ¥TL-GRULE  (LIST '(if) (TL-PCR M&C) '(,) TLMAIN)
  ¥DEP-M     (GRULE
               (COND ((EQ %C-F '¥RENYOU)
```

図6. TL-MRULEとTL-GRULEの例

```
¥YOUGEN.S
  ¥TL-GRULE  (TL-MOD.C M&C TLMAIN (TL-PCR M&C))
  ¥DEP-M     (COND ((FRULE))
                   ((MRULE))
                   ((GRULE
                      (COND ((EQ %C-F '¥RENYOU)
Yﾉﾃﾞ                  ------
  ¥TL-MOD    (LIST TLMAIN 'because TLSUB)
  ¥MOD       (MOD.C 'Yﾉﾃﾞ)
ﾃﾓ
  ¥TL-MOD    (LIST '(even if) TLSUB '(,) TLMAIN)
  ¥MOD       (MOD.C 'ﾃﾓ)
```

図7. mod.Cに関連する翻訳規則

3.3 法属性と翻訳

日本語文で用言に後接する枝能語はJASでは6種類の法属性として分析される[5](表4)。

	枝能語の例	働き
mod.A	タイ, ガル, ハジメル, オワル	付加的叙述
mod.N	ナイ	否定
mod.M	ネバナラナイ, サエスレバヨイ	義務・意図など
mod.T	タ	時
mod.Θ	ラシイ, ソウダ, ヨウダ	話者の判断
mod.C	ノデ, ノニ, カラ, ナラ, ガ	文間の関係

表4. 法属性の種類

このような働きをもつ枝能語の翻訳は、観念語の場合のように単に何らかの翻訳語句が対応するということではなく、目標言語の構文・形態を規定しあるいは変更する働きとして翻訳されることになる。

mod.Nとmod.Tは、英語の場合、述語部の形態を規定する情報であり、図3の④で処理される。mod.Nについては文型パターンに応じて翻訳句を否定構文に変更し、mod.Tについてはテンス属性をそのノードに付加する。テンスに応じた具体的な動詞の形態は、翻訳文生成のフェーズで英語の辞書を参照して生成される。

mod.Cは、図3の⑤で処理される翻訳規則の中で、関数TL-MOD.Cを介して処理される。TL-MOD.Cは、法属性mod.Cに関与している枝能語に付随している翻訳規則(プログラム)を評価する。図7にその例を示す。

mod.A, mod.M, mod.Θは、図8に例示するように英語の構文(図3の①でとり出された翻訳句)を変形する働きをもつ。この変形が図3の②で処理される。これはTL-MOD.Cの場合と同様にその枝能語に付随している翻訳規則(プログラム)を評価することによって為される。図9にこの翻訳規則の例を

示す。図9の中の変数TLRULEは変形の対象となっている翻訳句である。mod.Aに関与する材能語の中には、日本語文でも格構造の変形をひきおこすものがあるが(セル, レル, ガル, ---)，この場合には、関連する翻訳句の変更(従属句の識別名の変更)を翻訳規則の評価に先だって行なう。

この変形処理においては、変形の順序は　日本語文での材能語の出現順に沿って行なえばよいという仮定にたっている。

4. おわりに

分析の際の語句の結合規則に翻訳規則を対応づけるという方式で　日本語から他言語(英語)への翻訳を試みた。この方式は、個々の語の使われかたに関する知識を個々の語の属性としてデータベース上に記述し、それを解釈実行することによって文を分析し、翻訳するという方式であり、いろいろな言語事象に柔軟に対処できるように思われる。

プログラムとデータベース上の規則との区分けは必らずしも明解でなく、また規則のプログラミングで、分析プログラム・翻訳プログラム中のどの変数を参照・変更できるかなどについても未だ確然と定めてはいないが、これらは、どのような言語事象があってどのように規則化できるか、どの程度にまで把握し規則化すべきかということに関連して定めるべきものである。

英語の表現パタンの分類や代名詞化など文脈的扱かいその他残っている問題は多いが、今後、分析例・翻訳例を拡げつつシステムの整備をはかっていく予定である。

```
ﾀﾛｳﾊ ﾆﾎﾝﾆ ｲｯﾀ。                          (Tarou went to Japan)
ﾀﾛｳﾊ ﾆﾎﾝﾆ ｲｶﾈﾊﾞﾅﾗﾅｲ。                    (Tarou has to go to Japan)
ﾀﾛｳﾊ ﾆﾎﾝﾆ ｲｷﾀｶｯﾀ。                        (Tarou hoped to go to Japan)
ﾀﾛｳﾊ ﾆﾎﾝﾆ ｲｸｿｳﾀﾞ。                        (They say that Tarou goes to Japan)
ｼﾞﾛｳﾊ ﾀﾛｳｦ ﾆﾎﾝﾆ ｲｶｾﾀｶｯﾀｿｳﾀﾞ。
                    (They say that Jiro hoped to make Tarou go to Japan)
```

図8. mod.A, mod.M, mod.Bに関連する翻訳実験例

```
Yﾈﾊﾞﾅﾗｽ
  ¥TL-MOD     (CONS. '<OOO (NTH 1 TLRULE) 'have (LIST '<TOINF (NTH 2 TLRULE))
                    (NTHCDR 3 TLRULE))
  ¥MOD        (MOD.M 'Yﾈﾊﾞﾅﾗｽ)
ｶﾞﾙ
  ¥TL-MOD     (CASEQ STYPE
                  ((<F)
                   TLRULE)
                  ((<M <N)
                   (APPEND '(<OOO :WO be easer to) (NTHCDR 2 TLRULE))))
  ¥MOD        (MOD.A 'ｶﾞﾙ)

ﾀｲ
  ¥TL-MOD     (CONS. '<OOO ':FO 'hope 'to (NTHCDR 2 TLRULE))
  ¥MOD        (MOD.A 'ﾀｲ)
```

図9. mod.A, mod.M mod.Bに関する翻訳規則の例

```
ﾃﾞﾝｼｮﾊﾞﾄﾊ ﾌｼｷﾞﾅ ﾄﾘﾀﾞ。 ﾃﾞﾝｼｮﾊﾞﾄﾗ 1000ｷﾛﾓ ﾄｵｸﾆ ﾂﾙﾃｲｯﾃ ﾊﾅｼﾃﾓ
ﾏｲｺﾞﾆﾅﾗﾅｲ。 ﾁｬﾝﾄ ｼﾞﾌﾞﾝﾉ ﾊﾄｺﾞﾔﾆ ﾓﾄﾞﾘﾂﾃｸﾙ。
(a pigeon is a stranse bird)
(even if (?? ,:TT) take a pigeon far away , and (?? ,:LO) set
 (?? ,,LO) free , (?? ,:PO) do not lose (<POSSESIVE M&C22 (?? ,:PO)) way)
((?? ,:TT) come back to a pigeon box of oneself correctly)

ﾀﾛｳｶﾞ ﾆﾎﾝﾆ ｲｹﾊﾞ ｼﾞﾛｳﾓ ﾆﾎﾝﾆ ｲｸ。
                (if Tarou goes to Japan , Jiro goes to Japan)
ﾀﾛｳｶﾞ ﾆﾎﾝﾆ ｲｯﾀﾉﾃﾞ ｼﾞﾛｳﾓ ﾆﾎﾝﾆ ｲｯﾀ。
                (Jiro went to Japan because Tarou went to Japan)
ﾆﾎﾝﾆ ｲｷﾀｶｯﾀ ﾀﾛｳ。                    (Tarou who hoped to go to Japan)
ﾀﾛｳﾊ ﾌｼｷﾞﾀﾞ。                          (Tarou is stranse)
ｼﾞﾛｳﾆﾊ ﾀﾛｳｶﾞ ﾌｼｷﾞﾀﾞｯﾀ。                (Jiro felt Tarou stranse)
ｱﾒ ｱｶﾃ ｼﾞ ｶﾗﾏﾙ。                        (after rain comes fair weather)
```

図10. 翻訳実験例

文　献

1) T. IKEDA, "J-Analyser: Analysis of Japanese Sentences ---", JIP, Vol 3, No 4, 1981

2) 池田, "語の構造意味記述と文の構造意味分析" 通信学会 AL-78, 32

3) 池田, "一般化された格構造による意味表現を用いた日本語文の構文解析法" 信学論 60-D, 10

4) 池田, "日本語文の文析と構造翻訳" 通信学会部内研究会 1979.10

5) 池田, "日本語文における材能語の働き" 第21回情報処理学会全国大会, 1980

6) 池田, "日本語文における文脈処理の枠組" 第23回情報処理学会全国大会, 1982

7) 塚本, "Petlシステム説明書" 電総研, 1982