自然言語処理 34－1
（1982. 12. 7）

# 機械翻訳システム

## －－ トランスファ及び訳文生成 －－

天野真家
東芝電気総合研究所

平川秀樹
ＩＣＯＴ



### １． はじめに

ここに報告するトランスファ方式は、ヨーロッパなどで本来意図されたトランスファ方式とはその目的において一部異なった所がある。

ヨーロッパにおけるトランスファ方式の創出は第一に能率の良い多国語間翻訳を目的としている。多国語間翻訳を各国語のペアで直接プログラムすることは経済性がよくない。そこでペアで書く所はsource languageの内部構造からtarget languageの内部構造への変換部分だけとし、かつその部分を極力小さくしようとするのが、その意図である。

彼等の方法では普通、分析結果はなんらかのtreeとして必ず出力される。また、構文分析における曖昧性は複数の分析結果をもたらすのではなく、曖昧なまま結果のtreeに含まれる。トランスファ方式はこの様な分析結果をtarget languageの内部構造にそのまま変換するのに向いているのである。souce languageにおける構文的曖昧性はそのままtarget languageにひきつがれ、しかもその曖昧性は、target languageにおいても意味的に解消されてしまうのである。この意味でトランスファ方式は本来同族言語の翻訳のためのものなのである。

筆者等がそれにもかかわらずトランスファ方式を採用した理由はこの方式を用いて高品質翻訳を行なう意図にある。トランスファの機能 －－ source languageの内部構造をtarget languageの内部構造に変換する －－ を最大限に利用してそれを行なう。従ってトランスファは最少限に止める訳にはいかず、むしろ極めて大きなトランスファを行なう。この点がヨーロッパの行き方と異なる所である。

生成はトランスファにより得られた内部構造をsource languageの生成規則に則って解釈して得られる。この方式は多国語間翻訳を意図している。

この生成方式は又、柔軟な訳文の生成ができることを意味する。内部構造は訳文の意味を表すが、直接、表層構造を表しているわけではない。内部構造から表層構造への“翻訳”は生成規則に掛っている。生成規則を如何にするかにより、同じ内部構造から異なる表層構造が得られるのである。

### ２． トランスファ

#### ２．１ トランスファの種類

トランスファは、原言語の中間構造を目的言語の中間構造に変換する過程であり、次に示す処理を行う。

1. 原言語に対応する目的言語の単語を付加する。（語彙トランスファ）

2. 原言語の構文を目的言語の構文に変換する。（構造トランスファ）

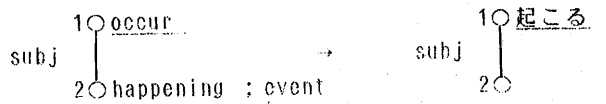
トランスファは、語彙トランスファ、構造トランスファの順で実行される。語彙、構造トランスファ共に 1つの規則は同じ形式で書かれる。これは、語彙トランスファであっても、語彙の選択によって構造が決定する場合があるからである。 1つのトランスファ規則は、中間表現に対する条件の記述部分と、それに対する変換操作を記述する部分から成っている。基本的な違いは、語彙トランスファが、言語の単語、熟語に結びついた規則であるのに対し、構造トランスファは、単語、熟語にかかわりなく適用される規則であることである。

語彙トランスファには次の２種がある。

１） 単純語彙トランスファ
２） 構造的語彙トランスファ

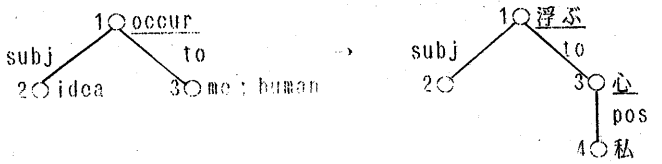
単純語彙トランスファは、通常の語対語の翻訳を意味する。構造的語彙トランスファはこれより複雑で訳語が文脈依存する場合に用いる。従って文脈というパタンで変換されることになる。例えばoccur の訳語は次の様になる。

１）




1 ( subj__ 2) = 1 ( subj__ 2) / [ 2 : 2.sem = “event ” ]
[ setf ( 1 ; ( entry ; 起こる) ) ]

２）




1 ( subj__2 to__ 3) = 1 ( subj__2 to__ 3 ( pos __ 4) ) / [ 3 : 3.sem = “human ” ] [ setf ( 1 ; ( entry ; 浮ぶ) ) , setf ( 3 ; ( entry ; 心) ) , setf ( 4 ; ( entry ; 私) ) ]

１）は、subj（主語）アークの繋がるノードの意味特性がevent であり、更にtoとなずけられたアークがなければ、その訳語は“起こる”になることを意味する構造的語彙トランスファである。

２）は、to－アークがありそれがhumank特性を持つノードに繋がれば訳語は

“浮ぶ”になり、かつ構造変化を起こし“心”が挿入され、更にmeはpos（所有格）－アークで“私”に繋がることを意味する。

## ２．２　トランスファ規則のシンタクス

トランスファ規則のシンタクスを次に示す。

マッチングパタン＝ターゲットパタン／［コンディション］［アクション］

### ２．２．１　マッチングパタン

マッチングパタンは、中間構造の構造的な条件を記述する部分であり、次の特徴を持つ。

1. マッチングパタンは、リスト形式で記述されるが、基本的には非順序リストである。
2. アーク名を明示し、意味的な記述向きになっている。
3. 非存在（！）および任意存在（？）を記述することができる。
4. マッチングの対象となるノードの 1つ上位ノードに関する参照が可能である。

マッチングパタン中のノード変数は通常任意のリテラルが許されるが、
‘０’及び‘１’は特殊な意味を表わす予約変数である。

0：マッチングの対象となる語の親ノードを示す。
1：マッチングの対象となる語

### ２．２．２　コンディション

パターン・マッチングが成功しても直ちにパターン変換が起こる訳ではない。ノード（語）が持つプロパティに関する条件が成立した場合だけパターン変換が実行される。条件はノードの各プロパティにつき論理式で書き表わすことができる。

例：

１．sem ＝“human ”

これはノード１の意味指標がhuman の時、真を返し、そうでなければ偽を返す論理式である。条件の詳細は参考文献（４）を参照。

コンディションは次の 2つの部分より構成されている。

1. どのノードに対する条件であるかを指定する部分
2. 条件本体

例

[ 2: 2. number= 1.number ]

## 2．3．4　アクション

アクション部分では、パタンのマッチング、及びターゲットパタンへの変換を通じて、束縛されたノードに対して種々の操作を行なう。対象となるノードは解析時に生成され、既に多くのフィーチャーを持つもの、パタン変換時に新たに生成され、フィーチャーを持たないものの 2種類がある。この部分は以下の機能を有する。

1. フィーチャーのセット（存在しないフィーチャーの追加、及び存在する場合はその置き換え）
2. フィーチャーの追加
3. フィーチャーの削除、一部削除

## 3．　訳文生成

2．までで得られた結果は目的言語の内部構造であり、そのままでは訳文にはならない。これを構文分析とは逆の過程により再解釈すると、訳文の原型（形態素処理を行なっていないもの）が得られる。解釈はその方法如何により複数個存在する。これはひとつの原文から複数個の訳文を出すことができる事を意味する。良い訳文を出せるかどうかはどの様な解釈を施すかにかかっている。即ち、解釈部分も他の分析、トランスファ部分と同様手続き部と規則部とが完全に分離されている。

## 3．1　生成規則のシンタックス

生成規則（GRULE）は次のシンタックスで定義される。

```
<grule> : = <left> <right>
<left> : = <S> | <arc_name> | <variable>
<right> : = <right1>*
<right1> : = <arc_name> | <variable> | <node> | <literal> | <
             feature> | <combination>
<fearture> : = (arc_node)*.property
<combination> : = <(feature | literal)>
<arc_name> : = <upper case>*
<node> : = <lower case>*
<variable> : = <$><alphaneumeric>*
```

## 3．2　生成規則の意味

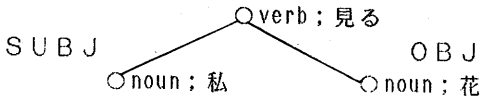
小さな生成規則の集合を例にしてその意味を説明する。

1　S＝ＳＵＢＪ，“は”，ＯＢＪ，“を”，verb；
2　ＳＵＢＪ＝noun；
3　ＯＢＪ＝noun；

第１行は文は

主語、目的語、動詞

という語順であり、かつ主語の後に格助詞の“は”が入り、目的語の後に格助詞の“を”が入ることを意味する規則である。
第２、第３行は、それぞれ主語は名詞、目的語は名詞であることを意味する。
この規則で次の内部構造を解釈する。




上記内部構造をたどるポインタは最初最上位ノードであるverbの位置にある。Ｓ＝ＳＵＢＪ　によりこのポインタはＳＵＢＪアークをたどりnounノードへ至る。同時に規則はＳＵＢＪ＝nounに状態遷移する。内部構造はノードnounを指しており、かつ規則もnounを指しているのでこの規則は生成に成功しノードnounの値“私”を持って呼ばれた規則Ｓ＝ＳＵＢＪ…に帰り、これを訳文領域にセットする。この時、内部構造のポインタもverbノードに帰る。次にＳ＝ＳＵＢＪ，“は”…　の“は”が取られ私の後に置かれる。これはリテラルなので内部構造ポインタは動かない。次に同様にしてＯＢＪアークがたどられ、“花”が訳文領域にセットされる。その後“を”がセットされ、最後にverbがセットされる。この様にして

私は花を見る

と言う訳文が生成される。

## ３．２．１　フィーチャ及びコンビネーション

上の例では格助詞の生成にリテラルのみを用いた。しかし、問題はこれ程簡単でなく実際にはもっと複雑である。動詞によっては目的格格助詞として、“を”を取らないものがある。例えば“好き”がそうである。この動詞は目的格格助詞として“が”を取る。この様な場合に対応するのがフィーチャである。好きが目的格格助詞として“が”を取ることはトランスファ辞書に書いておき語彙トランスファによって“好き”のpropertyに（tcase ；“が”）をセットしておく。又、生成規則にも若干修正を加えて次の様にする。

S＝…、ＯＢＪ、＊．tcase 、…

＊．tcase は既にＡＴＮＧパーサ（文献　４）やトランスファ規則のaction部で述べた記号法である。＊はカレント・ノード（今の場合ＯＢＪから戻った時点であるverbノード）を示し、．tcase はそのpropertyであるtcase を取り出すことを意味する。こうすることによりＯＢＪたる名詞の後には、それにふさわしい格助詞が付くことになる。しかし、全ての動詞にtcase を辞書でセットするのは能率的でない、通常、目的格格助詞は“を”なのだから、tcase をもたない場合のdefault 格助詞の機構が欲しい。これがコンビネーションである。
（＊．tcase ！“を”）　により＊．tcase が空なら“を”を値とすることができる。

３．２．２　変数

変数は表記の節約のために用いられる。例えばＳＵＢＪもＯＢＪも形式的には同じ名詞句からなる。この時、それぞれに名詞句の長い定義を書くことは表記の無駄である。従って次の様に一段媒介変数を置くことが許される。

ＳＵＢＪ＝＄ＮＰ；
ＯＢＪ＝＄ＮＰ；
＄ＮＰ＝名詞句の定義；

３．２．３　規則適用の方法

規則は初期規則のＳから始まり順次その右辺を内部構造をたどりながら展開してゆくことによって実行される。もし規則が内部構造に矛盾する様なことが起これば生成の失敗となる。しかし、これは分析の時程厳しくなく、本質的に除去できるものである。

４．　終りに

本システムの設計は、高品質な訳文を生成するためにトランスファ及び訳文生成の過程で柔軟な処理が施しえることに重点を置いた。

本研究は1982年 4月までに東芝総合研究所でなされたものである。
面倒な辞書作りに終始協力して下さった守山裕子嬢に感謝する。

参考文献

1. 石原好宏，田町常夫；　D－treeモデルとそれに基づく英日機械翻訳のための言語分析について，電子通信学会論文誌 '74／ 7／Vol. 57－D，no.7
2. 長尾真、他；　日英機械翻訳システムの作成とその問題点、電子通信学会技術資料　ＡＬ　79－74，1979
3. 天野真家、平川秀樹；　機械翻訳用辞書作成の問題点，情報処理学会第23回全国大会，pp1015 ～1016
4. 天野真家，平川秀樹；　英日機械翻訳用パーサについて，情報処理学会自然言語処理研究会技術資料　32，　1982