自然言語処理 69－7
（1988. 12. 6）

英日機械翻訳システムにおける解析手法について

中瀬 純夫
（株）カテナ・リソース研究所

本稿では、商用の英日翻訳システム *STAR* に用いられている解析メカニズムに関し、その骨子を紹介する。

この解析方式の特徴の1つは、ほかの解析処理に先立って、文脈自由型の統語解析だけが独立して行われるところにある。ここではおもに、表層統語構造に対する解析手法について、WFS（*Well-Formed Substring*）の概念を用いた定式化と各種の評価を行う。

そのほか、*STAR* の翻訳過程について、その概略を示しておく。とくに、英文中間構造の作成、語彙規則や構造規則などの適用、莫大な解析アンビギュイティの解消手法等について報告する。

On Syntactic Analysis Technique in English-Japanese Machine Translation

Sumio Nakase

Catena-Resource Laboratories, Inc.
27, Ichiban-chou, Chiyoda-ku,
Tokyo, 102
Japan

This paper presents an outline of the analysis mechanism employed in *STAR*, an English-Japanese machine translation system.

What distinguishes this system from other similar systems is that it performs syntactic analysis with context-free rules separately from any other analysis process. Formalizations and evaluations of this analysis method are fully discussed using the concept of WFS（*Well-Formed Substring*）.

A brief explanation about the internal modules in the machine translation process is shown. Also reported here are some subjects about a deeper analysis technique, such as the creation of an intermediate expression for a given English sentence, application of the lexical and structural rules, and some strategies for disambiguation.

1．　はじめに

本稿では、英日翻訳システムＳＴＡＲ（［１］）の解析メカニズムに関する紹介を行う。ここでは、文脈自由型の統語解析について詳しく述べ、その延長上で、翻訳システムにまつわる諸々の要素が示唆される形をとった。

まず、２．でＳＴＡＲの翻訳部分の構成要素について最低限の紹介を行った上で（図１にその概略を示す）、３．において、表層の統語解析について、その定式化、評価、そのほか詳細に議論する。４．では、言語処理の内容に関わる解析を行う部分について説明する。

言語処理としての扱い方、色々のレベルでのシステム機能に関わる議論、ＳＴＡＲの開発、運用にともない累積している諸問題については、直接には今回は扱わない。

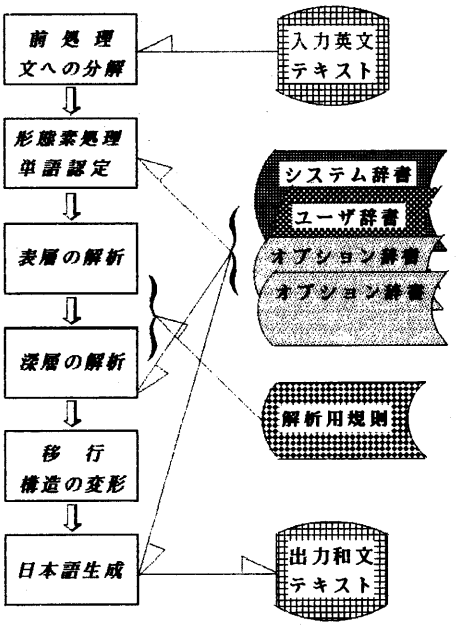

図１　ＳＴＡＲにおける翻訳プロセス

2　ＳＴＡＲにおける翻訳処理の概略

2．1　前処理

英文入力テキストを「文」の単位に分割する。それ以降は文単位に処理される。あわせて、最小限の書式情報、前編集記号などの解釈を行う。

2．2　形態素解析

入力文に含まれる「単語」の認定を行い、それらに辞書情報、形態情報を付与して、単語間の前接／後接関係に基づくＷＦＳ構造（ホモグラフ構造）として出力する。

辞書による認定のほかに、字句解析による慣用表現などの１語としての認定（ＬＲパーザによる）、未知語に対する語尾推定などを用いた辞書情報の推定などを行う。

2．3　表層の解析

ＳＴＡＲの解析処理では、表層の統語構造を文脈自由型の規則で捉え、そこで用いられた規則の随伴規則により、格文法的処理のしやすい依存構造を作成する。その構造の上で、辞書の記述や作成された構造そのものに基づいた種々の操作が施される。ＳＴＡＲでは、これが２つのフェーズに分けられている。第１のフェーズは文脈自由型規則に基づいた解析だけを行い、全ての可能な解を出力する。この内容は、３．で詳しく述べられる。

2．4　深層の解析

表層の解析結果を受けて、解の選択、語彙規則や構造規則などの実行、中間構造の作成などを行う。４．において、その概要が示される。

2．5　移行

解析の結果である英文中間構造を評価して可能な範囲での日本語構文構造への変形を行う。述語に対する連用修飾成分の語順の入れ替え、デフォルトとしての日本語送り、活用の指示、否定範囲の変更等々の操作である。

2．6　生成

変形された中間構造をトラバースしながら、各単語の日本語辞書記述を参照して日本語文を出して行く。例えば、図７や図８の構造を見ると、自然に日本語が出力できそうなことが想像できる。このとき、各単語に対し、自分自身や共起する語に対する送りや活用のコントロールがなされる。

2．7　ＳＴＡＲの辞書構成

ＳＴＡＲの翻訳処理に用いられる辞書は、翻訳基本辞

```
|=====================================================
|  - Dictionary Name : mydic
|  - Edited by       : nakase
|  - Date of Creation: Tue Nov  8 16:27:55 JST 1988
|  - Purpose         :
|-----------------------------------------------------
$meeting
-N0
 $ noun:::EVENT: (ミーティング)
 $ noun:::EVENT: (会議)
 $ noun:::EVENT: (大会)
 $ noun:::EVENT: (会議) [P(with)N](との)
$report
-N0
 $ noSa:::: (報告)
 $ noSa:::: (報告) [P(on,of)N](に関する)
-V0
 S0ii1:::X(報告する)
 S0ii1:::R(通信員である) [SH<HUMAN>] [P(for)N<ORGANIZATION>](の)
 S0ii1:::X(報告する) [P(on)N](について)
 S0it1:::X(報告する) [ON](を)
 S0it1::::(伝える) [ON<HUMAN>](を) [P(to)N](に) [P(for)N](のかどで)
 S0it1:::X(出頭する) [ON<ONESELF>]() [P(to)N](へ)
 S&it5:::X(報告する) [ON](を) [CV]S(と) ! 分詞
 $ it0:::X(報告する) [OC]S(と)          ! that 節
 $ it5:::X(報告する) [ON](が) [CV]S(と) ! to 一不定詞
 $ it9:::X(報告する) [OV]S(と)          ! 動名詞
```

図２　ユーザ辞書の記述例

書とオプション辞書とに大別される。前者はシステム辞書がベースとなるが、そのほかに、システム辞書への追加・修正内容を「ユーザ辞書」として与えることができる（図２）。ＳＴＡＲでは、ユーザ辞書を作成することにより、システム辞書の全ての内容が変更可能である。

オプション辞書は、図３の例にみるように、英日対訳をベースとした簡便な辞書形式であり、専門語辞書や私用辞書などにはこの形が多く用いられる。

１回の翻訳処理に際し、ユーザ辞書は高々１つ指定できるが、オプション辞書は０個以上任意個（実際はシステムの制約から１０数個程度まで）用いることができる。オプション辞書の記述は、骨格の翻訳基本辞書記述の中に、優先された訳語として挿入される。基本辞書にない語については、基本辞書相当の記述が作成される。

これらの辞書データは、翻訳処理に先立って、辞書コンパイラを用いて内部形式に変換するものとする。特に、動詞の活用、名詞の複数などの屈折形は、コンパイルの時点で生成される。

```
relay (n,*) : [DEVICE] 継電器
testing : [ACTION] テスト
individual (a) : 各々の
continuity : [PROPERTY] 導通
energized  (a) : 励磁
de-energized  (a) : 非励磁
ground : アース
```

図３　オプション辞書の記述例

３　ＷＦＳパージング

３．１　ＷＦＳについて

ＷＦＳ（Well-Formed Substring）は、書換え規則による統語解析の過程で生まれる構文成分を特徴付けるものである。筆者の知る限り、S. Kuno が［２］において、プシュダウン・オートマトン相当のパーザの処理時間の爆発を抑えるためにこの用語を導入している。ここでＷＦＳとは、句構造文法（書換え規則）に与えられたシンボルの一つと、対応する入力文の部分列（開始語位置と終了語位置）とだけにより特定されるものをいう。

たとえば、"I saw a man with a telescope." という文にたいする単純な文脈自由型の解析を例に取ってみよう。このとき、動詞句のＷＦＳとして、"saw a man" という部分列に対応するものと、"saw a man with a telescope" という部分列に対応するものの２つのＷＦＳが発生する。これにたいし、ボトム・アップ的に表現して、後者の「作られ方」は少なくとも２つある。"saw a man" という動詞句に "with a telescope" という前置詞句がついたものという解釈と、"a man with a telescope" という名詞句が動詞 "saw" についたものという解釈とである。しかし、ＷＦＳとしてはこの２つは区別されない。このことがＷＦＳの本質的な性格であり、このことによって、文脈自由型文法に対しては、高々入力語数のn^3の処理時間による解析が保証される。

もちろん、そこで用いる文法が非常に詳細なシンボルを持ち、例えば、「with により修飾済みの動詞句」といったカテゴリーを明示的に持っていれば、それらは区別されて、３つのＷＦＳを持つことになる。逆に、「作られ方」がそれ以降の書換え規則適用に反映するようなアルゴリズムでは、潜在的に莫大な数のシンボル集合を持った文法を用いているとも考えられる。本稿で「ＷＦＳ」を正面に打ち出しているのは、こうした戦略は避けるべきであるという立場に立っているとも言える。しかし、これは更に吟味すべき基本的な論点であると考える。

なお、ここで述べた例は、いわゆる「フローティング・ストラクチャ」の問題とは別問題である。ＳＴＡＲにおけるフローティング・ストラクチャの扱いとしては、例えば、動詞句と前置詞句がまとまるという規則を外し、文脈自由型規則の範囲内では常に目的語を前置詞句が修飾するという解析だけを行っておくといった方策に対応している。

３．２　ＷＦＳ構造

表層構造に対する我々の統語解析のアルゴリズムを定式化するために、「ＷＦＳ構造」という概念を定式化しよう。有限束、ネットワーク・フローなど、いろいろの定式化ができるが、ここでは、M. Kay のチャート・パージングの "inactive arc" のように、有向グラフ上のアークでＷＦＳを表現してみる。その時、ＷＦＳ構造は、ラベルのつけられたアークにより構成され、サイクル（方向を含めてのループ）を持たない有向グラフである。

ＷＦＳ構造のノードは、アークの間の接合点としての役割であり、抽象的なものでよいが、文脈自由型文法の場合は、「語位置」という呼び方で直感的な概念に一致するので、そちらを使うことも多い。

また、とくに、ソース・ノードとシンク・ノードという特別のノードを設定する。それらに対しては、「文頭」及び「文末」という呼び方もする。

ＷＦＳ構造は１つの書換え規則を前提にする。それを次に文法Ｇとして示しておこう。なお、実際のインプリメントでは、明示的に与えられた文法をアルゴリズム中で拡大することが多いが、ここでは暗黙的に拡大されたものまで含めてＧの中に記述されているものとする。

Ｇ＝（Ｓ，Ｒ，s_0）
Ｓ：シンボルの集合（ここでは、ターミナル、ノンターミナルの区別はしないでおく）。
Ｒ：書換え規則（一般的には、０型であってもよい）
s_0：開始シンボル（文記号）。

Ｇ上のＷＦＳ構造Ωは、次のようなものである。

Ω＝（Ｇ，Ｐ，p_t，p_e，Ｗ）
Ｇ：上述の文法。
Ｐ：ノード集合（語位置の集合）。
p_t：ソースノード（文頭位置）。
p_e：シンク・ノード（文末位置）。
Ｗ：ＷＦＳの集合。次のようなラベル付きの有向アークである。
$w \in W \Rightarrow w = (p_1, p_2, s)$
$: p_1, p_2 \in P, s \in S.$
ただし、Ｗはサイクルを生まない。すなわち、p_2からp_1へのパスはない。

本稿で述べるＷＦＳパージングは、初期ＷＦＳ構造から出発し、書換え規則の右辺のシンボル列に対応するＷＦＳ構造上の任意のパスに対し、同じ始点ノードと終点ノードを持ち、その書換え規則左辺のシンボル列に対応するようなパスを追加することによるボトムアップ・アルゴリズムである。ここで、ＷＦＳ（あるいは、ＷＦＳ列）の同定を行うことがＷＦＳパージングの本質である。

３．３　ＡＮＤ－ＯＲグラフによる解析履歴の構造

ＳＴＡＲの表層解析のフェーズは、ほとんどＷＦＳパージングだけを行うという、論理的にはきわめて単純なものになっている。その出力は、ＷＦＳパージングの解析処理の履歴になる。ＳＴＡＲでは、その結果の範囲内で再びパージング（探索）をすることにより、より深い解析やより深い構造の作成を行っている。これに対し、これらを表層の解析と同時に行うようなインプリメントをしたとしても、以下に述べるような解析履歴の構造を意識しておくことは重要であろう。

１つの入力単語列（ターミナル・シンボル列）に対する文脈自由型パージングの１つの解析結果（導出結果）は　通常、統語木として表現される。ＷＦＳ構造は、このシンボル列を拡張した横の構造ということができるであろう。これに対し、ＷＦＳパージングの経過と結果とは、統語木が重なりあった縦の構造であるということもできる。こうした縦の構造は、ＡＮＤ－ＯＲグラフの形式で自然に表現される。

図４に、簡単な文脈自由型文法による解析結果のＡＮＤ－ＯＲグラフを示す。但し、これは、解に貢献するＷＦＳだけについての部分構造であって、それ以外は刈り取られていることに注意していただきたい。

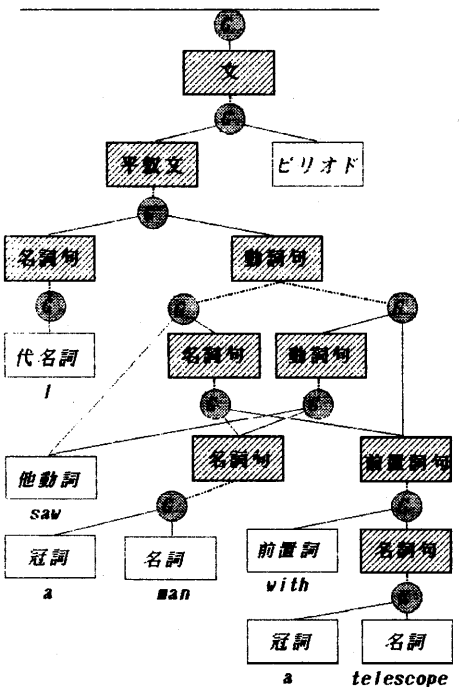

図４　解析結果のＡＮＤ－ＯＲグラフ

ここで、ＷＦＳ構造のアークはＯＲノードとして表現される。我々はこれをＷノードと称している。Ｗノードの子ノードは、そのＷノードに対応するＷＦＳシンボルを右辺として適用された規則の適用に対応するところのＡＮＤノードである。これをＧノードと呼ぶことにする。

各ノードの子ノードへのリンクは順序づけられている。Ｇノードの子ノードの並びは、ＷＦＳ構造での１つのパスに対応する。Ｗノードの子ノードは、後で述べる解析コストの小さい順に並べられている。

実際のデータ構造では、解のＡＮＤ－ＯＲグラフは、そこで用いられた規則と、子ノードに対応するＷＦＳの並びとで表現されたＧノード（生成情報）の集合として表現する。但し、ターミナル・シンボルについては、後続の処理で単語情報などを利用するために種々の情報をつけ加えておく。一般に、入力ＷＦＳに対してなんらかの「生成情報」を与えておくことにより、何段階かにわたりＷＦＳパージングを行うことができる。ターミナル・シンボルの辞書情報などは、このときの「生成情報」の特別な場合と考えることができる。

３．４　文脈自由型のＷＦＳパージング・アルゴリズム

ここで述べるアルゴリズムは、ＷＦＳ構造を順次入力して、解析履歴のＡＮＤ－ＯＲグラフを出力するためのものである。入力のＷＦＳ構造には前節で述べたような「生成情報」が付加されているものとする。入力のＷＦＳの条件は次の通りである。

（入力順序の条件）
２つのＷＦＳ、w_1とw_2に対し、w_1の終端ノードからw_2の開始ノードへのパスがあるなら、w_1はw_2より前に入力されなければならない。

ＷＦＳ構造に自然な半順序を導入すれば、「W_1の終了語位置がW_2の開始語位置より小さいか等しいならば、W_1はW_2より前に入力されなければならない」でもよい。

次に、アルゴリズム本体を示す。ここでは、入力されるかここで作成されるかするＷＦＳを順に取り出して、１つずつ着目して行く。そのＷＦＳを留めて、それより若いＷＦＳよりなる部分木構造と文法規則集合との照合を行う。ここでは、対応する規則を順に取り出して、その１つずつと照合している。このとき、ＷＦＳ側は対象ＷＦＳが、規則側は対象項が移り変わる。

一応、用いる規則を２型規則だけに限定しておく。０型、１型への拡張も容易であるが、ＡＮＤ－ＯＲグラフの拡張が必要であるなど、若干の手直しが要求される。

１°　ＷＦＳを１つ入力する。なければ、終了する。

２°　未着目のＷＦＳを１つ取り出し、着目ＷＦＳとし、右辺最終項（最右の項）のシンボルがそのＷＦＳのシンボルに一致する規則の集合を取り出す。
未着目のＷＦＳがなくなれば１°へ。

３°　規則集合から未着目の規則を１つ取り出し、その右辺最終項を対象項に、着目ＷＦＳを対象ＷＦＳにする。

未着目の規則がなくなれば２°へ。

４° 対象項が規則右辺第１項なら、規則適用は成功で、次のアクションをとったあと、６°へ移る：

対象ＷＦＳの開始語位置と着目ＷＦＳの終了語位置とを開始語位置と終了語位置とするＷＦＳで、規則の左辺シンボルをシンボルとするＷＦＳを作る。ただし、同じＷＦＳがあるなら作らない。
そのＷＦＳの「生成情報」（Ｇノード）に今回の対象ＷＦＳの列と規則番号とを追加する（出力する）。

５° 対象ＷＦＳの開始語位置を終了語位置とするＷＦＳで、そのシンボルが対象項の１つ前の項のシンボルに一致するものを１つとる。それがあれば、そのＷＦＳを対象ＷＦＳに、その規則項を対象項にして４°へ移る。

６° 対象ＷＦＳが着目ＷＦＳなら３°へ移る。
そうでないなら、対象ＷＦＳと対象項とがそれぞれ１つ前のものに戻り、５°へ移る。

結果として、文頭を開始語位置とし、文末を終了語位置として、開始記号をシンボルとするようなＷＦＳがあれば、この文は受理されたことになる。このＷＦＳの「生成」のＧノードをルートにするＡＮＤ－ＯＲグラフが解の構造になる。

これがないとき、ＳＴＡＲでは、入力文の「極大なＷＦＳ分割」を右辺とする仮想の規則を起動して、「救出」を行っている。

本節と同様のアルゴリズムはＡＬＩＣＥシステム［３］のためのパーザ（０型規則までサポート）などにインプリメントされたが、我々がＷＦＳの概念を全面的に採用したのはＳＴＡＲの文脈自由型パーザが初めてである。

３．５ 所要処理時間と記憶量の評価

まず、入力語数に対する処理時間と記憶量の形式的評価を見てみよう。ここでは、入力は線形のターミナル・シンボル列であると仮定する。こうしても、以下の評価では一般性を失わない。

入力語数をｎとしたとき、ＷＦＳの個数のオーダは高々n^2である。実際、シンボルの種類を無視すれば、ＷＦＳの総数は明らかにｎ（ｎ＋１）／２に抑えられる。シンボルの種類は高々これに対する乗数である。

文法が２型規則だけからなっており、その右辺項数が高々２であるならば、処理時間あるいはＡＮＤ－ＯＲグラフのサイズのオーダは高々n^3である。

これを見るために、１種類のシンボル「Ｘ」と、１つの規則「Ｘ－＞Ｘ・Ｘ」だけからなる文法を考える。この単純化は最大値評価のオーダを変えない。

このとき、ｎ－ｋ個ある長さｋのＷＦＳの「作り方」はｋ－１通りあるので、ＡＮＤ－ＯＲグラフのＧノードの個数はｎ（ｎ－１）（ｎ＋１）／６になる。これはまた、規則適用を試みる回数に一致する。

なお、このときの解の個数（アンビギュイティ）は、$_{n-1}C_{2(n-1)}/n$になる。これはリーフがｎである２分木の個数であり、ｎの指数関数のオーダになる。この形の数は、最終的なアンビギュイティには反映しなくとも、バックトラック・メカニズムだけによる単純な解析における計算回数を評価してみると、この形のコンプレクシティが頻繁に発生することが判るであろう。

ここでは、規則右辺の項数を高々２と仮定したが、一般の場合につき前記のアルゴリズムを適用するとn^3オーダの処理時間は保証されない。これを保証するには、規則を機械的に２項規則へ変換すればよい。それはまた、アルゴリズムにおいて吸収することもできる。それには、ＷＦＳのシンボルとして「完成途中」のシンボルを追加すればよい。これがチャート・パージングの "active arc" などの考え方に一致することは明らかである。

しかし、実際に記述される規則で、右辺項数の多い規則が処理時間の爆発に寄与するということは直感に反している。我々は、基本的に前節のアルゴリズムを踏襲し、理論的な上限をn^3で抑えるという方策は採らなかった。

一方、指数関数的な計算回数の爆発は、実際の規則記述によっても発生する。また、統語規則だけによる詳細な解析の結果として莫大なアンビギュイティが発生しうることも確かめられている。実際、後で示す実験でも、２億近いアンビギュイティを持つ解が受理されている。

したがって、表層の解析と並行して、どこまで可能性のない解釈の刈り取りが可能であるのかということが論点になる。本稿の立場は、（１）そこで十分な保証のある刈り取り手法は確立されていない、（２）そして、前節に述べたような「盲目的な」アルゴリズムにより全ての可能な解釈の探索を行っても十分にリーズナブルな範囲での処理時間と記憶容量で納まるということで、（３）詳細な語彙情報、意味情報、別の枠組みでの文構造表現などを用いた解析は、独立のモジュールにおいて体系化して差し支えないであろうというものである。

最後に、ＳＴＡＲのＷＦＳパーザを用いて、英字新聞記事やエネルギー関連のアブストラクトの５６３文について処理を行った結果を図５に示す。ここには、解析に失敗した文や、語数５未満の文は含まれていない。そのほかには、データの検定などの分析は経ていない結果であることに留意していただきたい。本格的な分析は次の機会に譲る。

なお、ここで用いられているのは、文脈自由型規則として３０００弱の規則である。ただし、その各規則は、次節で述べる「不定繰り返し」や「省略」の記述を含む。また、この実験はＮＥＷＳ８３０（ＳＯＮＹ）を用いているが、動作時の環境は最良のものではない。

この図を見ると、処理時間が入力語数の１．９乗程度と見積もれることが判る。しかし、中には４０語前後で４０秒以上を要している文も見受けられる。

ここでは、スペースの都合で処理時間の語数に対する関係だけを示したが、そのほかのいくつかの数値の語数との関係についても簡単に述べておこう。

入力のＷＦＳ（ホモグラフ）はほぼ入力語数に比例していたが（1.026乗）、これは直感的な予想に一致する。

解析中に作成されたＷＦＳの数はおおよそ入力語数の１．５乗強のオーダになっており、１０００前後以下に集中している。この数は、ＷＦＳパージングで最小限必要な記憶量に比例する。なお、ここで用いたＳＴＡＲのバージョンでは、処理可能なＷＦＳの数を８１９２以下に限っている。ちなみに、この時のＧノードを含めたパージング用の記憶量の合計は約３００ｋバイトである。

解析中に作成されたＧノードの個数は、語数の１．６５乗程度とかなり少ないものであった。実際に成功する規則適用は、上限値に比べればはるかに少ない。

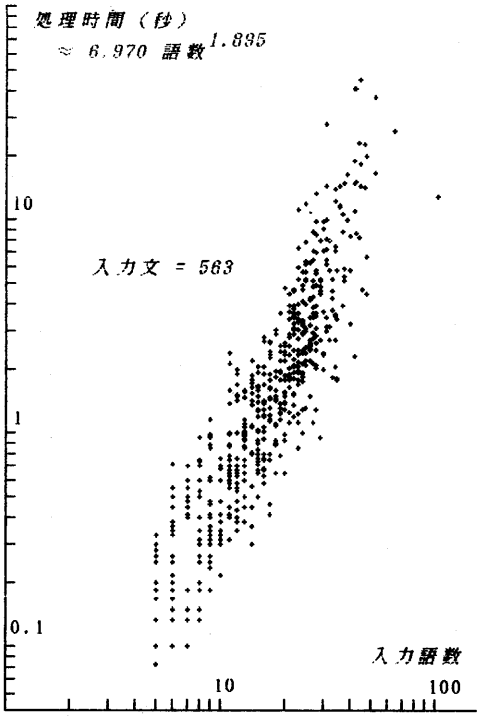

図５　ＷＦＳパージングの所要時間

解として取り出されたＷＦＳの個数はちょうど語数の１．５乗であった。ほとんどは１００前後以下に納まっているが、１０００を越えるものも１つあった。

成功した解析のアンビギュイティを評価すると、語数の３．５乗となったが、これまでのものに比べばらつきが非常に大きかった。６５語の文で 193,251,096 ものアンビギュイティを持つものも見受けられた。この文は人名の並列と同格表現がほとんどを占めている文である。これに対し、最大の語数１０３語の文のアンビギュイティは１万程度である。これらのことは、アンビギュイティが組合せ的な爆発の性格を持ち、規則の書き方や入力文の性格へ依存する度合が大きいことを示している。

なお、この６５語の文では、５９９のＷＦＳと８１７のＧノードを出力し、１０３語の文では、４３２のＷＦＳと５５６のＧノードを出している。

解の中での最小コストの値を評価してみると、入力語数の０．９５乗程度であった。これは、コスト関数の線形性から予想されることであるが、線形に密集した系列の左上方に「不自然さ」を持った解が分布している非常に興味深い分布であった。

３．６．　表層の解析におけるコスト関数（ウェイト）

アンビギュイティの解消のために、ＳＴＡＲでは、ウェイト・メカニズムと呼ぶ機構を用いている。これは、解析に関わる種々のコストを計算し、最終的にそのコスト関数の値の最小なもの（あるいは最小からいくつか）を解として出力するものである。表層の解析においては、その基本値を与える。それは、後続の解析フェーズにおいて、最適解探索のための推定コスト関数になる。

予め、辞書などに振り当てたウェイト値が計算の基礎となる。それらが、ターミナル・シンボルのＷＦＳのコスト値である。そこから、次のように、ＷノードとＧノードのコスト値が決定される。

Ｗノードｗのコスト：

$$\varphi(w) = \mathrm{min}_i \varphi(g_i)$$

ただし、g_iはｗの子ノード

Ｇノードｇのコスト：

$$\varphi(g) = \sum_i c_i \varphi(w_i) + c_0$$

ただし、w_iはｇのｉ番目の子ノードで、c_0は使われた規則の基本ウェイト値、c_iは使われた規則の右辺ｉ項目のウェイト係数の値。

ここで、ＳＴＡＲの文脈自由型規則の記述では、次の例のような記述が行われる。

```
:c VP --> vt1 ( 2:PP ) NP
:b NP --> NP (+a cma NP )* (cma) and NP
```

はじめの例は、他動詞ｖｔ１と名詞句ＮＰとから動詞句ＶＰが構成されるが、任意的に前置詞句がそのあいだに挿入されるという規則である。このとき、規則の基本ウェイト値はｃにより表現され、ウェイト係数は、ＰＰだけが２で、そのほかは１であることを示している。

次は、名詞句の並列の規則で、単に "and" や "or" を挟んで名詞句が並列するほかに、接続詞の直前にコンマがおかれることや、あいだにコンマに分かたれた名詞句が任意個挟まれて、いわゆるリスト構造を構成することもあることを示している。このときの基本ウェイト値は、ｂの表現する値に不定繰り返しの起こった回数だけａの値を足したものになる。

ここで、はじめの例のＰＰのウェイト係数は、間に挿入される前置詞句が長くなればなるほど不自然さが増すことを示し、２番目の例の括弧のウェイト値は、リスト構造が長くなるとコストが増すようなコントロールをしていることになる。

３．７．　そのほかの付加機構

これまでは、ＷＦＳパージングとして、下からの情報の伝播や環境状態の利用は「ストイックに」避けるべきことを述べてきた。しかし、ＳＴＡＲにおいても、限られた範囲ではその原則からはずれた部分がある。

ＳＴＡＲの規則記述では、終端記号は英字小文字３桁で、非終端記号は大文字２桁で記述することにしているが、それらの後に１桁の変数を指定することができる。この変数は、規則の左辺から右辺に合成・伝播し、その過程で規則の不受理を引き起こすことがある。現在のＳＴＡＲでは、このメカニズムを主語と述語の間の数の一致におもに用いているが、動詞の時制・態などや、形容詞・副詞の級の扱いなどへの拡張が想定される。次の例を参照のこと。

```
NPx --> noux          (※１)
VPz --> vt1z NP       (※２)
NPw --> artz NPx      (※３)
SGw --> NPx VPz       (※４)
```

```
SE --> SGc prd        (※５)
VPz --> vtvz SGy      (※６)
NPx --> NPx and NPx   (※７)
```

終端記号 nou の数は、左辺の NP へコピーされる（※１）。動詞類も同様である（※２）。※３に対し、NP の「数」が "p"であると、冠詞 "a(an)" の「数」は単数 "s" であるので、組み込み関数によって "NIL" が戻り、この規則は適用されない。これに対し、NP の数が "s" や単複同形の "c" であると、結果は "s" になり、それは W に代入される。※４の場合、主語が複数、述語が三単現なら、 w は NIL になるが、主語が単数、述語が複数といったケースの場合は、w は "y" という限られた「数」になり全面的な拒否はされない。しかし、通常は「数」"y" は受理されない（※５）。ただし、"order" などの特別な動詞の目的節としてはそうした「数」が受理される（※６）。そのほか、名詞並列で１つでも複数のものがあればそれらをまとめたものも複数になるといった特別の変換が行われる（※７）。

こうした変数値の伝播は、シンボル種類の拡大を意味する。すなわち、変数値まで含めてＷＦＳの同定が行われ、「数」の違うシンボルは別のＷＦＳと見なされる。

次に、ＳＴＡＲでは、文脈自由型の規則適用に反しているかのような動きをするものがある。例として、名詞の同格表現の次の規則を考えてみよう。

```
NP --> NP cma NP cma
```

実際の文章では、句切り記号（ピリオド、コンマなど）の直前にこの表現が現れたときには、規則右辺の最後のコンマは省略される。そこで、次のような文脈依存型の規則も必要になってくる。

```
NP prd --> NP cma NP prd
NP cma --> NP cma NP cma
```

実際には、これを文脈自由型の規則だけで表現することも可能であるが、それは莫大な規則数の増大を引き起こす。ＳＴＡＲでは、直後にピリオドやコンマのあるとき、先の規則の最終項を省略した規則が適用可能であるという特別手続きをアルゴリズムに追加している。

そのほかにも、ＷＦＳパージングの「ストイシズム」に従わなくとも対応できるものはいくつか残されている。従属節などでの「スラッシュ要素」（「ミッシング・ストラクチャ」）、「フローティング・ストラクチャ」などがその代表であり、我々の検討事項になっている。

最後に、プログラミング的な特別処理として、次の２点だけに触れておく。一つは、アルゴリズムの説明でも触れた、解析失敗の場合の極大ＷＦＳ分割をまとめる仮想規則の適応であり、もう一つは、Ｇノード・テーブルがあふれた場合の「ガベージ・コレクション」である。後者は、Ｇノードの選択肢の中で、最小コストのＷＦＳ以外を棄却することで、一時的にＧノード個数をＷＦＳ数にまで絞りこむ操作である。

我々はパージングの効率化を含む種々の実験を行ってきた。例えば、規則を予め探索して、環境による到達可能性を用いたＷＦＳ抑制や規則適用の抑制といったメカニズムである。しかし、ある程度コントロールされた規則記述を用いているからでもあろうが、我々の試行では、それらの効果は高々２、３割の処理時間の短縮にしかならず、逆に、解析に失敗する文に対しては、小さいＷＦＳしか作成されないことになる。その結果、解析失敗の時の翻訳結果の質が低下してしまう。ハードウェア性能の向上などを考慮するとき、この程度の効率化は余り有効でないと考える。

むしろ、我々の現在の関心は、より「深層」の解析との融合を「ストイックに」避けているＷＦＳパージングのメカニズムが本質的に失っている要因をいかにして補って行くかにある。とくに、語彙レベルの個別情報を表層の解析で活用することは重要な検討課題であろう。

４．　より「深い」解析

４．１　中間構造の作成

ＳＴＡＲの解析処理の中核で用いられるのは、文脈自由型文法に基づく統語構造ではなく、依存文法的な格支配構造である。また、日本語の生成に至るまで、原則として同じ形の構造が用いられる。この構造の原型は、ＷＦＳパージングに用いられた各々の規則の「随伴規則」により作られる。言い替えると、英語の解析規則は、表層の統語構造を把握するための条件部と、中間構造を作成する実行部との組になっているといってもよい。

図６に、簡単な文脈自由型規則と図で表現した随伴規則の組を示す。図７にこの規則の適用により作成された構造を示す。また、図８に埋め込み文のイントロデューサなどが下位の構造に移動する規則によった構造を示す。

ＳＴＡＲの中間構造は木構造をベースにしているので、共通の支配語を持つ要素の並列や形容詞節などでの明示的な支配構造表示はできない。

４．２　語彙規則の適用

ＳＴＡＲの辞書記述では、各単語は品詞細分類（ＳＳＣ：Syntactic Sub-Category）毎の記述の並びで構成されている。各ＳＳＣのもとで、訳語の違いや、用法の違いなど、なお複数の語彙記述がなされる。その記述の、日本語に関わる部分（訳や日本語送り・活用などの情報）の違いを無視したものを「英語文型パターン」と呼ぶ。解析処理では作成中の中間構造を評価しながら、英語文型パターンの決定を試みる。ここでは、その適合性を数値で表現し、最も好ましいものを採用することになる。また、その結果は文全体の解析のコスト値にも反映する。

その典型的な例は、前置詞や副詞小詞との共起記述である。ここでは、条件に合致しない英語文型には大きいペナルティが課される。これにより、ＷＦＳパージングでの解の適・不適が逆転する事も少なくない。

さらに、着目している語に共起する語の属性値によって文型を選択することもある。ＳＴＡＲでは、各単語の各個別記述に与えることのできる属性を内在素性と呼び、そこに与える記号をフィーチャ（またはフィーチャ・コード）と呼んでいる。一般的には、これは意味マーカなどと呼んでいるものに当たるが、現在のＳＴＡＲでは、フィーチャの値や体系は利用者に委ねられており、既存のものとの重複などによる混乱のない限り、どの様な内容のフィーチャ体系を定義してもよい。

例えば、次のような記述をすることができる。

```
"take" : {乗る} [SN<HUMAN>] [ON<VEHICLE>] {に}
"catch": {ひく} [ON<KAZE>]
"kill": {自殺する} [ON<ONESELF>] {}
```

```
"look": {する}
        [P(after)N<ANIMATE,HUMAN>]{の世話を}
"a": {一人の} [-N<HUMAN>]
"plane": :::VEHICLE: {飛行機}
"cold": :::KAZE: {風邪}
"himself": :::ONESELF: {}
```

フィーチャの制限を行うと、着目語の文型パターンが評価されるだけでなく、相手の語の文型パターンも限定されることになる。なお、ＳＴＡＲの辞書では、英語の表層格を記述し、深層格は陽には現われない。

そのほか、「受動態の時にのみ優先的に利用すべき記述」などの指定が評価される。また、前置詞句に対する共起指定は、「仮に」別の構文要素にかかっているフローティング・ストラクチャに対しても試み、これが採用されるとその係先が決定する。

語彙規則による共起要素への関係は、英文中間構造の上に新しいリンクを作成する。例えば、上の "take" の例では、"take" から主語のノードと目的語のノードへアークが張られる。これは、母体の木構造とは独立の構造であり、このアークに沿って、着目語から対象語への日本語の活用、送りなどのコントロールが行われる。

４．３　ディスアンビギュイティ

ＳＴＡＲにおける解析のアンビギュイティの解消は、ＷＦＳパージング結果のＡＮＤ－ＯＲグラフの上で最小コスト問題を解くことに帰着される。"analyser" と名付けられた、「より深い解析」を行うＳＴＡＲのモジュールは、ＷＦＳパージングの結果の推定コストが与えられたＡＮＤ－ＯＲグラフ上で、随伴規則、語彙規則、構造規則（解析上必要な範囲での構造の変形、情報の伝播、並列構造などの評価）を用いた中間構造の作成・評価を行いながら、最尤解析の探索を行うものである。このモジュールは、最終コスト値の最小のものから、指定された順位の解（英語中間構造）までを出力する。

現在のＳＴＡＲでは、この探索処理を原則としてトップ・ダウン・パージングの枠組みで、バックトラックを行いながら遂行する。このフェーズでだけは、探索回数の爆発を避けるためのヒューリスティックスを導入しているが、希に多大な処理時間が要求されることがある。

５．　おわりに

ここで得られた諸々の成果は、ＳＴＡＲの研究、開発、運用、実用化に色々の形で携わり、協力していただいた多くの方々に負うところが大きい。この場を借りて謝意を表したい。

［参考文献］

［１］（株）カテナ・リソース研究所：「英日翻訳システムＳＴＡＲシステム説明書」、1988

［２］Kuno,Susumu: The Predictive Analyser and a Path Elimination Technique、Communication of the ACM、Vol. 8 No. 7、1963

［３］Nakai,Hirosi et al.: "ALICE" A System for Analysing the Linguistic Structure of Japanese Sentences、COLING 80 資料、1980

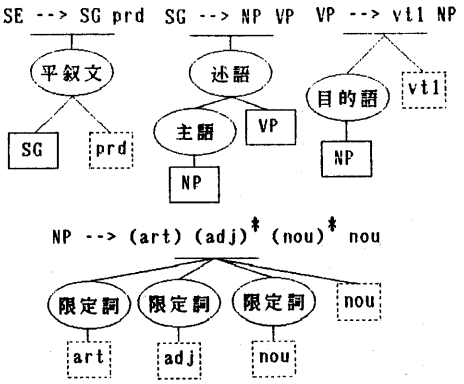

図６　構造作成規則

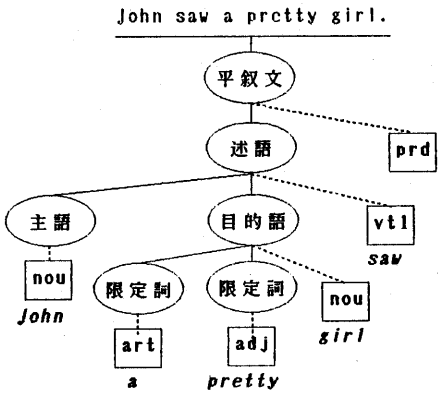

図７　随伴規則による中間構造

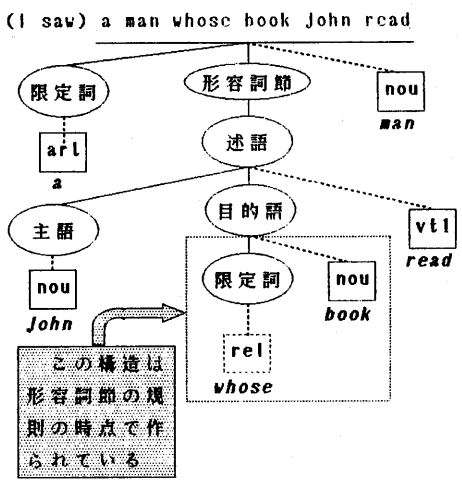

図８　繰り下げられた部分構造