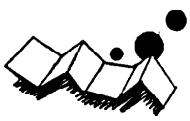
解　説



日本語の構文解析†

佐藤泰介†† 田中穂積††

1. はじめに

会話の文も，書いた文も，いずれも大きな物理的制約が課せられている．線条化という制約である．文は一定の意味を聞き手（もしくは読み手）に伝達する．ところが意味の世界には，そうした制約はなく，立体的な構造をしているものと推察される．そこで，聞き手が，線条的な記号の系列から，立体的な意味構造を再構成しやすいように，線条的な記号の系列には，一定の秩序がある．日本語には日本語の秩序がある．文のもつこの秩序のことを構文とよび，与えられた文の構文を明らかにすることを構文解析という．

構文解析の分野は，意味処理等の分野に比べて比較的明確な枠組みを備えている．しかし，処理の深さや背景となる言語理論によってさまざまな定義が考えられ定義がはっきりしない．第1に，定義された規則により，適格文の範囲が決められている人工言語とは異なり，自然言語では，適格文の定義が明確ではない．第2に，自然言語処理システムの目的により，入力文から取り出す最終構造の形式に，句構造や意味ネットワーク，論理式などさまざまある．第3に，自然言語処理の場合には，実際には形態素解析と構文解析，構文解析と意味解析との間の相互作用が考えられ，そのため構文解析の定義は，一層複雑化する．

しかしながら，機械処理の立場から見た構文解析の役割りは，少なくとも次の二つに要約される．

i）非適格文の排除

ii）文の構文構造を取り出し，より深い意味処理への手がかりとする．

このような目的に沿ってさまざまな構文解析手法やアルゴリズムが開発されている．本稿では，それらの概観を与えることを目的とする．主題は，「日本語の構文解析」であるが，構文解析には，言語の種類に依存する側面と，依存しない普遍的な側面があると思われるので，両者を意識した立場から解説してみたいと思う．

2. 構文解析用のアルゴリズムとシステム

形式言語は自然言語の対極にあるものとして捉えられがちであるが，その構文解析手法は，自然言語の構文解析にとって大いに参考になるものと思われる．そこでまず形式言語の構文解析について，その基本事項を述べ，次に形式言語の構文解析手法を強化応用したシステムについて説明する．

2.1 形式言語の構文解析

形式言語はオートマトン理論の分野で良く研究されている[1),2)]．一つの形式言語は使用する記号と書き換え規則により定められる．形式言語をクラス分けすると，3型言語（正則，regular），2型言語（文脈自由，context free），1型言語（文脈依存，context sensitive），0型言語の順に広い言語族を形成し，それぞれの受理機械として，Transition Net (TN), Push Down Automata (PDA), Linear Bounded Automata (LBA), Turing Machine (TM) が対応する．0型を除いて，各タイプに属する言語は帰納的集合を成すが，0型言語は任意の帰納的に可算な集合を生成できる．

CFG（文脈自由型文法，2型言語）については良く研究され，また具体的システムへの応用も多いので，以下 CFG に焦点を絞る．

CFG の構文解析の結果は構文解析木 (parsing tree) を使うことにより解り易く表示される．**図-1** に例が示されているように，構文解析木は，木の形で文の内部構造を表現したものである．構文解析アルゴリズムは木の作り方により，top-down/bottom-up, serial/parallel の区別がある．top-down 法は木を上から作り，bottom-up 法は木を下から作る（図-1 参照）．一度の構文解析により，serial 法は一つの木を作ろうとし，parallel 法はすべての可能な木を作ろうとする．

† Syntactic Analysis of Japanese Sentence by Taisuke SATO and Hozumi TANAKA (Electrotechnical Laboratory).

†† 電子技術総合研究所

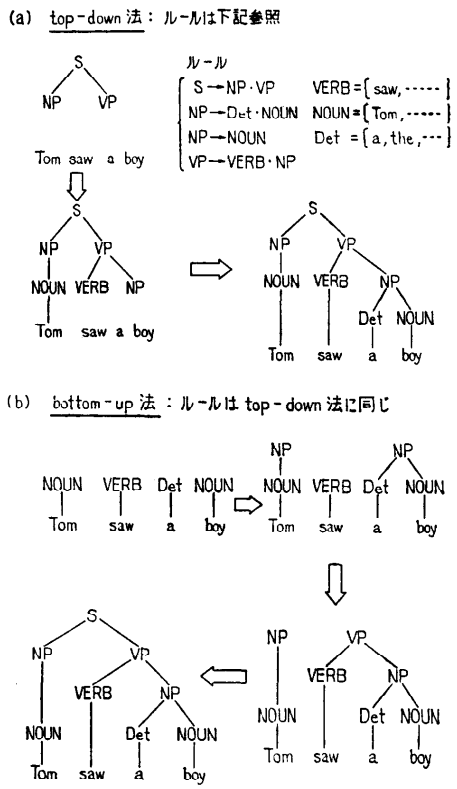


図-1 top-down 法と bottom-up 法構文解析

top-down & serial または bottom-up & parallel の組み合わせでパーザ (parser) が作られることが多い．

図-1 の (a) に示した top-down (& serial) 法を使う時は，解析が失敗した時のためのバックトラックメカニズムが必要である．また S→S・NP 等の再帰的ルールや，A→B，B→A等のサイクルを成すルール群の取り扱いに注意を払わないと，無限に木を成長させてしまう恐れがある．

Top-down 法は，通常入力文の長さ n に対し指数関数的に処理時間が伸びるが，WFST[11] (Well Formed Substring Table) を利用することにより $O(n^3)$ に押さえることができる．WFST は構文解析の途中で作られた PPT (部分構文解析木) を貯めておくテーブルで，バックトラックが起きても，それまでの部分的に正しい結果を残しておき，後で活用するためのものである．これにより構文解析の過程で同じ PPT を2度作らなくなり，処理速度の改善につながる．

図-1 の (b) に示した bottom-up なやり方では，再帰的ルールについては心配ないが，サイクルを成すルール群に関しては無限に木を成長させてしまう恐れがあるので取り扱いに注意が要る．bottom-up な解析法で有名なのは Young-Cocke-Kazami の方法で，長さ n の入力に対し $O(n^3)$ の時間で解析を終える[1]．

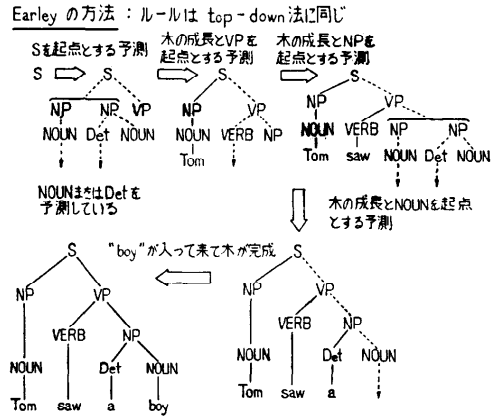


図-2 top-down 法と bottom-up 法の融合

bottom-up 法の欠点は，木を下から上に作って行く時，果してその木が最終的な構文解析木の一部に成りえるのか，全く気にしないため，無駄な木を生成しがちなことである．この点を改良したのが Earley のアルゴリズムである (図-2 参照)．このアルゴリズムは top-down 過程と bottom-up 過程を持ち，前者は成長すべき PPT のトップ・ノードのカテゴリを予測し，後者はその予測に従って実際に PPT を成長させる．長さ n の入力に対し，処理時間は，文法に曖昧さがある時 $O(n^3)$，ない時 $O(n^2)$ である．

Earley の方法では，top-down (予測) 過程は，ルールの集合と，起点となるカテゴリが決まれば，入力文に関係なく決ってしまう部分である．この点に着目して，ルールが与えられた時点で top-down 過程を "precomrpile" してビットテーブルの形を覚えておき，実際に入力文が与えられた時そのテーブルを参照する事により，同じ top-down 過程を何度も繰り返さないようにすると，Pratt のアルゴリズムになる．このアルゴリズムは言語処理システム LINGOL[10] の基礎になっている．

Young-Cocke-Kazami, Earley, Pratt の方法は一般的アルゴリズムであるが，もし文法に曖昧さがない場合には更に速く解析できる．もし文法が $LR(k)$ なら k 個の先読みにより $O(n)$ で解析できる．$LR(k)$ は Pratt の方法の中で，上に PPT を成長させて行く際あらかじめ計算された先読みストリングセットの中

に，実際の先読みストリングが入っている時のみ PPT を成長させるもので，“文法の precompile” を更に徹底したものと考えることができる．

以上紹介したアルゴリズムはいずれも純粋な構文解析法であり，自然言語の構文解析にそのまま使って適切かどうかは判断の分れるところであるが，WFST, Earley, Pratt, $LR(k)$等の手法に見られる，“同じ木を2度作らない”，“あらかじめ無駄とわかる木は作らない”，“文法が与えられた時，そこから最大限の情報を引き出してコンパイルしておく”等の考え方自体は，自然言語の構文解析システムを作る際に参考になると思われる．

2.2 構文解析システム

ここでは，形式言語の構文解析手法を背景とする自然言語構文解析システムを取り上げる．前節のアルゴリズミックな解析法とは異なり，自然言語の多様な現象を取り扱うため，色々な工夫や強化がはかられている．

言語学的観点に立ったシステムは Harvard predictive analyzer [Kuno 1962] や MITRE [Zwicky et al. 1965] 等1960年代に試みられたが[3),4)]，1970年代に入ると，AI（人工知能）の立場に立ったシステムが作られるようになった．想定されている言語は通例 CFG+α である．

積木の世界に関する Q.A. システム SCHRDLU [Winograd 1971] の構文解析部分は PROGRAMMAR と呼ばれ[9)]，Lisp プログラムの集まりである(図-3(b)参照)．Halliday のシステミック文法に基づいた定式化になっている．構文解析は積木の世界の知識を使い top-down 方式で，曖昧さのある場合には，良さそうな木を一つ作ろうとする．構文解析のすべてをプログラム的に行うことにより，柔軟性を得ている反面，システムの修正・変更・理解が困難になるという欠点が明らかになった．

このような，プログラムで何でもやろうという PROGRAMMAR の立場とは異なり，構文解析のコントロール部分を Transition Net (TN) の形に定式化したものが ATN (Augmented Transition Net) [Woods 1970] である[5)]．

ATN は TN を augment（強化）したもので，状態の遷移は，使用者によって書かれたプログラムをチェックする事により行われ，ついでにレジスタに情報をセットできる．状態間の情報交換はレジスタを介して行われる(図-3(c)参照)．ATN は基本的には top-down, depth-first であるが，逆に bottom-up, breadth-first に処理を進めるのが Heidorn の APSG である[7)]．各ルールに対し適用条件，適用時の動作がプログラムで書け，処理結果は“レコード”と呼ばれるデータ構造にまとめられる．

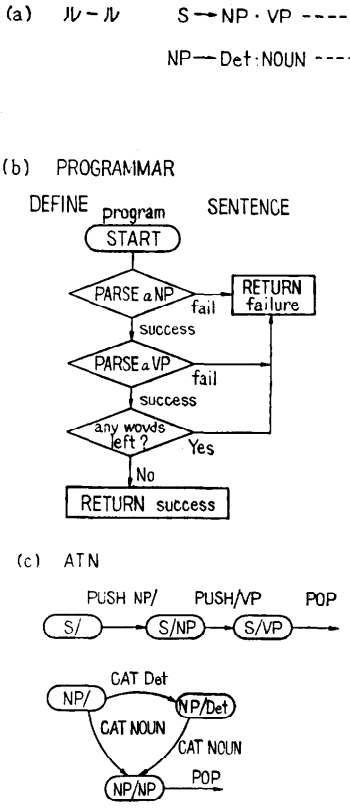


図-3 さまざまなコントロール方式

ATN-フォーマリズムはさまざまなシステムに採用されているが，日本語の語順が比較的自由であるという特性を考慮して，パターンマッチングの機能を付け加え，また巧妙なバックトラックメカニズムも加えたものが PLATON [長尾・辻井 1974] である[24)]．パターンマッチングにより語順の自由度をうまく吸収して，ルールを適切に起動する事ができる．

Marcus の W & S (Wait and See) パーザはパターンによって起動されるモジュールと呼ばれるデーモンから成り立っている[8)]．各モジュールは自分の利用できる部分構造がバッファに蓄積されるまで充分待ってから起動され，より上位の構造を作り出す．これにより，情報不足のまま作った構造を，後でバックトラ

ックにより捨ててしまうという事態を防いでいる.

今まで述べた手続き的システムに対し，よりアルゴリズミックなシステムにLINGOL [Pratt 1973]である[10]. これは前節に述べたPrattのアルゴリズムに基づき，各ルールにCOG部，SEM部の二つの使用者に開放された部分を付け加えたものである. LINGOLを改良して，下から構文解析木を成長させるところで，使用者が成長させるべき木を選択できるようにし，更に形態素解析に伴うバックトラック・メカニズムを付け加えたものが拡張LINGOL [田中ほか 1977]である[34]. 拡張LINGOLは形態素解析と構文解析が一体化しており，左から右へと文を解析し，その過程で意味情報を利用する事も可能である.

以上紹介したシステムの他にもMIND[40]やグルノーブル大学のシステム(本特集号の長尾氏の解説参照)等々紹介すべきシステムは多いが，紙面の都合上省略する. 述語論理を応用した構文解析については，文献41)，42)を参照されたい.

3. 日本語の構文解析

3.1 日本語の特徴と構文解析

日本語の持つ特徴がその構文解析に与える影響について幾つか取り上げてみたい.

日本語は漢字入力法に関する問題があるが，(本特集号の高橋氏の解説参照) それはともかく構文解析で問題になるのは，日本語には分かち書きの習慣がないという事である. したがって構文解析のため，文から語を切り出す必要があるが，“ニワ／トリ／ガイル”のように，切り方により文の意味が異なる場合もあれば，“ニワトリニイク……2羽採りに行く”を“ニワトリ”と“ニイク”と間違って切り，失敗することもある. 通常は字種の切れ目情報や最長一致 (longest match) を用いるが，上の例で解かるように常にうまく行くとは限らない.

構文解析を援用して語を切り出す方法もあり，この場合辞書に，ニワトリ，ニワ，トリが入っているとすれば，“ニワトリガイル”は二つの構文解析木を，“ニワトリニイク”は唯一つの構文解析木を生じるであろう. 切り方が複数ある時，正しい切り方を選択するためには意味，文脈情報の利用が不可欠である.

第2に日本語は語順が自由であると言われている. したがって2.1節で述べたような純アルゴリズム的方法は適さないとも考えられる. しかしながら良く考えてみると，日本語単文を，名詞句の並び一述語一助動詞・補助用言等一終助詞として捉えた場合，自由度があると言われているのは，主として格を成す名詞句の並びであり，名詞句内部や助動詞等を含む述語文節内部に自由度があるとは言えないであろう. 2.1節で述べたアルゴリズムとの関連で言えば，述語から終助詞までは正則文法でほぼ解析できると言われており[19]，格を成す名詞句の並びの自由度もルールの作り方により吸収できる.

副詞に関しても述語文節内部に割り込ませることはできず，特に終助詞の後には如何なる語も接続しない. 総じて，文末に近づくほど語順が厳しくなる (この辺の事情については13)参照). 更に基本的な格の並びには順序があると言われており，例えば存在文では場所格が先頭に来る例が多い[18]. また談話の観点からしても，旧い情報をになう語が文の先頭の方に，新しい情報をになう語が文末の方に来ると言われ[17]，談話レベルの情報構造が名詞句の並びを統制していることも考えられる. したがって日本語に於いては，係る文節が受けの文節より左にある限り基本的語順は確かに自由であるが，他の原因から決っている場合もあり，2.1節で述べたような方法が適用できる部分もあるとするのが妥当であろう.

第3に同音異義語が多い. 英語には“Time flies like an arrow”の例が示すように多品詞語が多いが，(time, flies: 動詞，名詞，like: 動詞，前置詞) 日本語には“キタ”(北，着た，来た) のように同音異義語が多い. そのため入力文がカナ表記・ローマ字表記の場合曖昧さを生じる事が多く，構文解析上問題となる[20].

同音異義語による曖昧さを消すために，文の先頭に動詞は来ないから“キタへ行ク”の“キタ”は“北”であるというように統語情報を使う事もあるし，“薬”は意味素性が-animateなので“薬ガキイタ”の“キイタ”は“聞イタ”ではなく“効いた”であるというように意味が使える事もある. 勿論文脈や常識を使って初めて解決できる曖昧さも多い.

第4に省略が多い. 日本語では文脈から明らかな語は省略されるのが普通である. 省略語を補うためには広く談話レベルの情報まで必要とされるので，非常に困難な問題である.

第5に日本語はSOV型言語であり，次節の係り受け解析の項で述べるように，修飾関係は原則として文の前方から後方へ向き，重要語は後ろに来ると言われている. このことは文全体の意味関係を捉えるため，

まず重要語を文の後方から先頭へと探索して見つけ出し，この語を鍵として意味解釈を進めるという方法を示唆している．実際このやり方を採っているシステムもある[25),35)]．

その他，日本語は左枝分れ型言語であるとか，敬語表現が発達している等の特徴があるが紙面の都合で省略する．

3.2 日本語の係り受け解析

日本語文を文節の有限列として見た場合，文節* 間に成り立つ広義の修飾関係を係り受けと言い，大まかに連体修飾関係（"美しい花"），連用修飾関係（"美しく咲く"），格関係（"花が咲く"）に分類できる．修飾する方を係りの文節，される方を受けの文節と言うことにすると次の原則が成立する．

（i）係り受け非交差……係りの文節から受けの文節へ，矢印を書いて係り受け関係を示した場合，矢印は交差しない．A B C D のようなことはない．

（ii）文末にない文節はその右側のいずれかの文節に係る．

この原則をもとに係り受けを決めて行くわけであるが，アルゴリズム的に見れば i）の非交差条件からして，botom-up に構文解析木を作っている事に等しく，係りと受けを括弧でくくればこの間の事情は明白である．しかしながら，係り受け本来の意味からして，係り受け解析は意味的なものであり，意味情報主導型の構文解析であると言える．

具体化に当っては（i），（ii）の条件を生かしスタックを使って，文の先頭から処理する（係る方を捉え，後で受ける方を探す）か，文の後方から処理する（受けとなるものを先に捉え，後でそれに係るものを探す）方法が考えられる．

なお日本語の係り受け解析に相通ずるものとして，Hays らの依存関係理論がある[39)]．これは文の構成要素の依存関係を重視する立場から，依存関係規則により句構造規則では表示されない支配項（governor，受ける方）と被支配項（dependent，係る方）の依存関係を明示する文法である．

3.3 日本の構文解析システム

日本における構文解析システムは係り受け解析を基礎としているものが多く，書き換え規則を使った統語規則（Syntactic Rule）主導型の解析システムは少ない．一般に日本語構文解析システムを設計する際，係り受け解析等の意味主導型にするか，統語規則主導型にするか，二つの選択肢があるが，それぞれ異なった特徴を持つ．前者は積極的に意味・文脈情報を使い，統語・意味解析を同時に行う一方，そのため，システムの理解のし易さ，モデュラリティが失われやすい危険性がある．後者は文法規則が明示され，アルゴリズムも良く研究されているが，解析過程のコントロールが統語規則に依存しがちで，柔軟性が失われる危険性がある．しかし両者は決して排他的なものではなく，係り受け解析によっても構文解析木が作られ，また統語規則は意味情報によって強化・運用される必要があるのは当然である．統語解析と意味解析を同時にする事については Woods の意見等も参考になるだろう[6)]．

以下では日本の構文解析システムを幾つか取り上げて概説する．

言語学者の作ったシステムとして，IRIS 0 A がある[27)]．石綿の作ったこのシステムは依存関係文法を基礎として，変形および変形を探知する規則も含み，扱う言語と理論を特定しない，述語を中心とする解析システムである．辞書と文法はプログラム本体とは別である．処理は①辞書引き，②主要述語の発見，③文型をパターンマッチにより確定した後，被支配項を求める．これを繰り返して文にあるすべての支配項を求める，④変形の逆探知などして解釈のつかない被支配項の支配項を求める，⑤結果の出力，の順に行われる．同音異義語に対してはすべての可能な解釈の組み合わせを作ってから処理する．日本語を扱う時は，入力はカタカナ分ち書きで入れている．

石綿と同じく依存関係文法に基づいたシステムに，石原等の D-tree を使った英日機械翻訳システムがある[28),29)]．D-tree は文中の"語群"の依存関係をセマンティクネットとして表現した木構造で，翻訳のための中間言語として使われる．英文入力は単語・熟語関係の辞書により単語識別が行われ，次に語群辞書により各単語がまとめられて語群系列（WGC）に変換される．例えば "the boy"，"have sent"，"look for" はそれぞれ一つの語群にまとめられる．次に WGC は規則によりブロックに分割され，中の述語に機能パタ

* ここでは，学校文法で用いられている意味での文節を考えている．

ーン（一種の文型）を割り当ててから，文脈に依存しない二つの語群の依存関係を示した第1種の依存規則を使ってブロックごとに部分 D-tree を作る．次により大域的な依存関係を示した第2種の依存規則を使ってこの部分 D-tree を二つずつまとめ（等位接続，関係節等の処理），入力文に対するさらに大きな D-tree を得る．翻訳は，語群の訳出順序を決定した後，訳語辞書により語群ごとに対応する訳語を求め，訳語等の選択をして完成する．依存関係等の曖昧さに関してはすべての解を求めている．

日本語の係り受け解析については吉田等のグループが実証的研究を進めている[21),22)]．彼等は2文節間の係り受けを7のタイプに整理する．そして各タイプの係り受けについて，与えられた係り受けを標準的な係り受けの形に還元するための手法を提案している．辞書には標準的係り受けの“意味”が記述されているので，これを利用することにより非標準的な係り受けの意味を，上述の手法を使って理解することができる．更にこれを基礎として“文の標準型分解”の概念・方法が示され，意味処理への展望を開いている．

また日本語文の述部に関しても検討しており[23)]，活用，付属語の意味，アスペクト等を考慮した述部処理システムが作られている．ローマ字書きの入力を与えると最長一致により語・付属語を left-to-right にバックトラックをともないつつ切り出し，それらの接続条件のチェックを行い，たとえば推量，断定などの意味表示を伴った分ち書きが得られる．これから英語述語へ翻訳が行われる．

京大の長尾等は ATN を改良した PLATON[24)] を用いて，中学校理科一化学の教科書を対象とした文理解システムを作っている[25),26)]．統語処理・意味処理・文脈処理を同時に行う方針をとり，出力結果はセマンティクネットの形である．単文の処理は文中から動詞を取り出し（受けの方），その前にある名詞句（係りの方）を格助詞を手懸りとして取り出す所から始まる．次に名詞のカテゴリ，意味的役割り，上位一下位関係等が記述されている名詞辞書その他により，必要ならバックトラックを行いつつ名詞句内部の係り受け関係を確定し，次に格文法の考えに従って整理された動詞辞書を使って，動詞の格構造をもとに動詞と解析済みの名詞句の係り受け関係を確定する．もし埋まらない必須格*があればM-スタックとKM-スタック（それまでの解析から得た，または存在が推論できる名詞が置いてある）を探して埋めようとし，それでも埋まらない時には“トラップ”を設けて，後続文の処理により埋めてくれる名詞句ができるのを待ちかまえる．重文・複文は単文処理を再帰的に行う事により処理している．M/KM-スタックを使ってアナホーラ（指示・照応）の問題も扱っており，その後の我が国の言語解析の研究に大きな影響を与えた．

電電公社の島津等は算術と地理に関する Q. A. システム MSSS78 を作製している[30),31)]．そこでの構文解析手法を見てみよう．

入力文はカタカナ分かち書き自由で与える．処理は語い辞書（詞と辞，詞と辞の接続関係を transition net の形で記述）を使って文から文節を切り出す所から始まり，次に左から右へ文節間の係り受けを調べつつより大きな係り受け関係にまとめる．そのための知識・手続きは“詞”の case patterns に埋め込まれており，処理対象となる文節と照合することにより起動される．case patterns は各“詞”に対して与えられ，その詞が受ける文節の統語的・意味的条件，チェックのための手続き，バックトラックのための手続き，Q. A. のための手続き等々が書かれている．解析はスタックを使って行われ（shift-reduce-parsing に似ている**．スタックトップは受ける文節・句，係る文節・句は未処理入力文節またはスタックの底の方にある），また Rlist と呼ばれるリストによりバックトラックのための状態保存，可能な他のパスの保持が成される．曖昧さに関しては全部の可能な解を出すようにしている．

電総研の池田は格構造の考えを一般化し[32),33)]，述語のみならず名詞等品詞一般も格構造（受けのあり方）を持つとし，この考えに基づいた構文・意味解析システムを作製している．

このシステムは，すべての文法的知識を辞書の中に持ち，入力スタックに置かれた文節以下に分かち書きされた入力文を，先頭から shift-reduce-parsing 的に処理する．結果は句（一つの詞を中心とする意味的まとまり）から成る木構造として出力スタックに残る．

辞書には，品詞，活用，辞の接続情報および格構造，

---

* 格文法の用語[35)]．格文法では，たとえば「行く」という動詞には行為者格と目標格（これらを深層格とよぶ）をもつ格フレームがあると考える．格フレームを構成する深層格中，省略不可能なものを必須格とよぶ．

** スタックを使ったパージングで，例を挙げて説明する．今スタックの内容が $\boxed{\ }\boxed{C}\boxed{B}\boxed{A}$，入力文が $W_1 \cdot W_2 \cdots$ とする．まず入力文の先頭の語 W_1 をスタックに移す（shift）と $\boxed{W_1}\boxed{C}\boxed{B}\boxed{A}$ の形になる．ここでもしルール $D \rightarrow W_1 \cdot C$ があればスタック上部の W_1 とCを取り除きDに置き換える（reduce）．結局，スタックは $\boxed{D}\boxed{B}\boxed{A}$，入力文は $W_2 \cdots$ の形になる．これを繰り返し，入力文が尽きた時にスタックの底にシンボル“S”が残っていればパージングは成功である．詳細は文献1）参照．

係りの構造（語が他の語に係るあり方）が記述されている．補文構造を持つ使役・受身等の助動詞，～たい，～てもらう等の辞については，辞書に書かれた語の格構造を変形したものを用いて処理している．

連体修飾構造は処理ルーチンを再帰的に使い処理し，また曖昧さに関してはすべての可能な解を求める．しかし，係り受けの良さ（例えばより多くの依存関係を持つ句のあり方が良い）を求める事により，句の解釈結果に優先度を設け，またそれにより不要な句の解釈をすてることもある．指示語，省略語の同定も試みられている．

同じ電総研の田中等は今までの係り受け解析型の構文解析とは異なり，拡張 LINGOL を使いアルゴリズミックな構文解析を行っている[34]．

プログラムにより強化された CFG ルールと辞書を与えると文の構文解析が可能になる．入力はローマ字分かち書き自由で，語を最長一致で構文情報も利用しつつ切り出し，left-to-right に高速に処理する．曖昧さがある時にはすべての構文解析木が得られる．さまざまなユーティリティプログラムが用意されており，会話的に構文解析システムを作り上げることができる．現在拡張 LINGOL を使って日本語文の意味抽出システム EXPLUS が作製されている[35]．EXPLUS の意味抽出過程は，まずパージングによって各ノードに意味抽出のためのプログラムが付属した構文解析木を作る．次にこの木を手懸りに各プログラムを上から評価する事により，係り受け等の意味抽出処理を行うという2段階戦略を採っている．

パージングは left-to-right に行うが，意味抽出は，日本語文の重要語は文末側にあるという特性をふまえて right-to-left に行っている．意味構造の表現にはフレーム的形式を用い，上位一下位関係，全体一部分関係，意味素性等が体系的に使われている．

最後に紹介するヤチマタは日本語によるデータベース照会システムであり，実用的色彩が濃い[36],[37]．

データモデルは関係形式に似た名詞表と動詞表から成り，それぞれ名詞句と述語に対応する．システムに対して〈名詞句〉は？　という形の文を問いかけると，〈名詞句〉に対応した名詞表が得られる．

入力文はカタカナ分かち書き自由であり，REL[12]等で用いられた Kay のチャート式アルゴリズムにより，bottom-up に構文解析木を作る[12]．木の終端の名詞にはデータベースの名詞表が対応する．ヤチマタの意味処理は，名詞句と名詞句の関係を，基本的データベース検索演算とその引数の関係に変換する（意味の木を作る）事であるが，この作業は構文解析と同時に行われる．文法は制限されており，例えば動詞句は名詞句を修飾する形でのみ許される．意味処理を終えても曖昧さが残る場合（複数の検索コマンドが出来ている）には，使用者の指示を受けるようになっている．

以上紹介したシステム以外にも取り上げるべきものは多々あるが，残念ながら紙面の都合上省略せざるを得なかった．

5. おわりに

構文解析は自然言語処理にとって不可欠であり，また重要な位置を占めるものである．したがって構文解析システムを設計する場合充分慎重でなければならない．ここに設計方針を決めるに当り考慮すべき項目を列挙してみよう．

- 背景となる言語学的枠組み――国文法，変形文法，Montague 文法，……或いは自分で作る
- アルゴリズムの枠組み――係り受け解析，または CFG+α（α の中味），その他
- 形態素解析・構文解析・意味解析の相互作用――密接にする，または分離する．
- コントロール構造――スタックを使う，ATN 式にする，LINGOL 式にする，その他
- 入力の形態――分かち書き，字種の問題
- 曖昧さ――可能な解をすべて出す，一つ出す．また解消するための手立て．
- 指示語，省略語，熟語，同音異義・多品詞語，未知語，並列表現，「の」による名詞連続，関係節，補文構造の取り扱い．
- 文法的知識の表現――どこに置くか，フレーム的表現，ネットワーク的表現，或いはその他．上位一下位・部分一全体関係，意味素性，常識の利用方法

実際の文解析システムは，上に挙げた項目に対しさまざまな態度をとり，また独自の工夫を凝らしている．構文解析の手法・実働化が一つの方向へ収束しつつあるとは言い難いが，明確な言語的枠組み，柔軟なコントロール，効率の良いアルゴリズムを備えたシステムの発展を望みたい．

参考文献

1) Aho et al.: The theory of parsing, transla-

tion and compiling Vol. 1, Prentice-Hall, Inc. (1972).
2) Hopcroft et al.: Formal language and their relation to automata, Addison-Wesley (1969).
3) Kuno: The predictive analyzer and a path elimination technique CACM, 8 (1965).
4) Zwicky et al: The MITRE syntactic analysis procedure for transformational grammars, AFIPS, FJCC, Vol. 27 (1965).
5) Woods: An experimental parsing system for transition network grammar, Natural language processing, in Rustin (ed.), Algorithmic Press (1970).
6) Woods: Semantics and quantification in natural language question answering, Advances in computers, Vol. 17, Academic press (1978).
7) Heidorn: Augmented phrase structure grammar, Theoretical issues in natural languege processing, in Schank and Webber (eds.), Cambridge, Mass (1978).
8) Marcus: Diagnosis as a notion of a grammar, Theoretical issues in natural language processing, in Schank and Webber (ed.), Cambridge, Mass (1978).
9) Winograd: Procedures as a representation for data in computer programfor understanding natural language, Academic Press (1972).
10) Pratt: LINGOL-A progress report 4IJCAI (1975).
11) Grishman: A survey of syntactic analysis procedures for natural language, AJCL, microfiche 47 (1976).
12) Thompson et al.: Practical natural language processing: The REL system as a prototype, Advances in computers, Vol. 13, Academic Press (1975).
13) 久野 暲: 日本文法研究, 大修館 (1973).
14) 奥野敬一郎: 生成日本文法論, 大修館 (1974)
15) 井上和子: 変形文法と日本語, 大修館 (1976)
16) 柴谷方良: 日本語の分析, 大修館 (1978).
17) 安井 稔: 新しい聞き手の文法, 大修館(1978)
18) 北保保雄: 助動詞の相互承接についての構文論的考察, 国語学, 83 (1970).
19) 西村他: 日本語基本文法単文篇, 電総研い報 No. 783 (1978).
20) 吉田他: 日本語の機械処理 日本語文の曖昧さと関係表現について, 電気四学会連合大会 (1978).
21) 吉田 将: 二文節間の係り受けを基礎とした日本語の構文分析, 信学論D, Vol. 55-D, No. 4 (1972).
22) 吉田他: 日本語の機械処理——構文分析から意味分析へ——, 信学論D, Vol. 55-D, No. 6 (1972).
23) 首藤他: 日英機械翻訳のための述部処理システム, 信学論D, Vol. 60-D, No. 10 (1977).
24) 長尾他: 自然言語処理のためのプログラミング言語 PLATON, 情報処理, Vol. 15, No. 9 (1974).
25) 長尾他: 意味および文脈処理を用いた日本語文の解析——名詞句・単文の処理——, 情報処理, Vol. 17, No. 1 (1976).
26) 長尾他: 意味および文脈処理を用いた日本語文の解析——文脈を考慮した処理——, 情報処理, Vol. 17, No. 1 (1976).
27) 石綿敏雄: 変形とその逆探知を含む構文分析, 電子計算機による国語研究Ⅲ, 国研報告59(1976).
28) 石原他: D-tree モデルとそれに基づく英日機械翻訳の為の言語分析について, 信学論 D, Vol. 57-D, No. 7 (1974).
29) 石原他: D-tree モデルに基づく一つの英日機械翻訳システムおよび実験, 信学論D, Vol. 57-D, No. 7 (1974).
30) 雨宮他: 図形世界を話題とした質問解答システム, 情報処理, Vol. 18, No. 8 (1977).
31) 島津 明: 日本語の構文・意味解析——質問応答システム MSSS 78 での試み——, 信学会, AL 研究会, AL 78-94 (1978).
32) 池田尚志: 一般化された格構造による意味表現を用いた構文解析法について, 信学論D, Vol. 60-D, No. 10 (1977).
33) 池田尚志: 語の構造・意味記述と文の構造・意味分析, 信学会, AL 研究会, AL 78-32 (1978).
34) 田中他: 自然言語のためのプログラミングシステム——拡張 LINGOL について——, 信学論 D, Vol. 60-D, No. 12 (1977).
35) 田中穂積: 日本語の意味構造を抽出するシステム EXPLUS について, 信学論D, Vol. 61-D, No. 8 (1978).
36) 諸橋他: ヤチマタにおける擬似日本語, 情報処理, CL 研究会, CL-8 (1976).
37) 藤崎他: データベース照会システム「ヤチマタ」と名詞句データ模型, 情報処理, Vol. 20, No. 1 (1979).
38) Fillmore 著, 田中春美他訳: 格文法の原理——言語の意味と構造——, 三省堂 (1975).
39) Hays: Dependency theory: a formalism and observations, Language, 40, 4 (1964).
40) Kay: The MIND system, Natural Language Processing, in Rustin (ed.) Algorithmic Press (1971).
41) Colmerauer: Metamorphosis Grammers, in Natural Language Communication with Computer, Lectures Notes in Computer Science, Springer Verlag (1978).
42) 渕 一博: 定理証明としてみた Earley/Pratt のパージング・アルゴリズム, 情報処理学会 17 回全国大会 (1976). (昭和 54 年 9 月 5 日受付)