自然言語解析のためのプログラミングシステム COMPLAN について†

西田豊明†† 堂下修司††

自然言語解析システムは，与えられた規則に基づいて入力文を解析する専門家システムとして捉えることができる．専門家システムの能力は規則の内容に強く依存するので規則作成は重要な過程である．本論文ではこの点を重視し，比較的単純な処理機構を中心に，解析規則の記述の容易性，解析過程の制御に関するヒューリスティックスの組込み，豊富な診断機能の組込みなどに重点をおいて設計した，自然言語解析のためのプログラミングシステム COMPLAN について報告する．COMPLAN では対象言語の記述のための規則と解析のための規則を分離し，それぞれの目的に都合のよい形式を用いる．すなわち，対象言語の記述には文脈自由文法規則を拡張した形式を用い，文解析には，拡張遷移網型の手続き的な解析規則を用いる．対象言語の記述を手続き的な解析規則に変換する過程を明確にして両者を対応づけることにより，文脈自由文法規則を中心とする記述のわかりやすさ・モジュラリティと，手続き的な解析規則による柔軟な制御構造を結びつけた．また，対話的診断機構を組み込んで，解析規則に組み込まれた知識が不十分でも，人間の補助によりマンマシンシステムとして稼動できるようにした．さらに自然言語解析システム開発の段階では，辞書などの仕様の変更などがかなり起こりうることが予想されたので，辞書や解析規則の容易かつ一貫した変更・修正を補助するためのユーティリティを作成した．

1. まえがき

近年，自然言語処理の研究成果を機械翻訳等の実際的処理に応用しようとする試みが活発化している[1),7)]．計算機によって実際のテキスト*に現れる文を解析するためには，ある程度の範囲の言語現象を取り扱いうる機械処理用文法規則や辞書を用意しなければならない．そのとき重要な役割を果たすのは係り受けなどの制約条件に関する知識であると考えられる．それが欠けていると，計算機は与えられた入力文に対して正しい解析結果のほかに多数の意味的ないしは語用論的に不適切な解析結果を出力することになる．このような知識は，概念一般に関するもの，対象領域固有のもの，慣習的なものなどであるが，それらを全て含む知識ベースはまだ利用できるようになっておらず，また近い将来実現されたとしても，部分的・断片的な性格をもつものであると考えられる．すると，現在有効かつ実現可能な自然言語解析システムは，(1)各解析規則において種々の知識を容易に利用できることを前提とするが，(2)知識が不完全であっても，それを人間が（最小限の）対話で補えば，広い範囲の文に対して正しい解析結果を得ることができ，さらに(3)解析規則の作成・追加・修正に有効な診断情報を人間にフィードバックできるようなものであると考えられる．

本論文では以上の立場から設計・作成した自然言語解析のためのプログラミングシステム COMPLAN (a COMprehensible Parser for natural Language ANalysis) について報告する．

2. COMPLAN の設計方針

2.1 設計目標

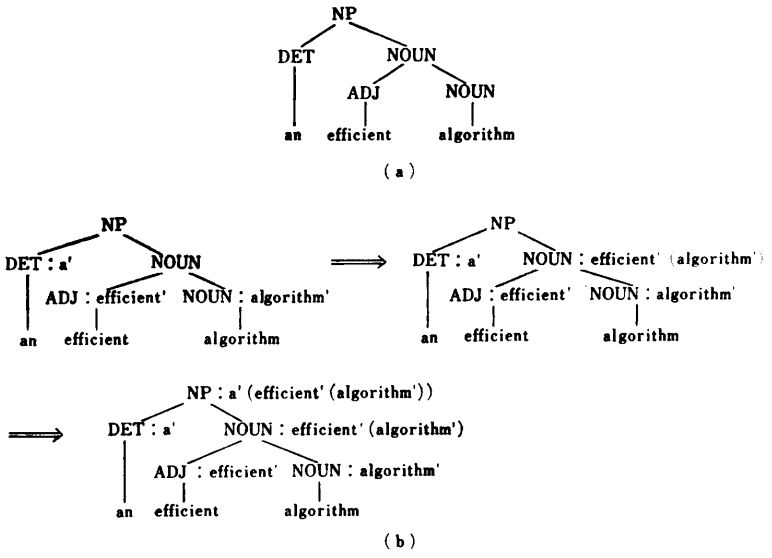
COMPLAN の具体的な設計目標は次のとおりである．

（目標1） 解析規則の記述の容易性
（目標2） 解析規則のモジュラリティ
（目標3） 有効な診断情報をフィードバックできること
（目標4） 解析に要する記憶量・時間量が小さいこと
（目標5） 解析規則で種々の知識を参照できること
（目標6） 人間の補助により正しい解析結果が効率的に得られること
（目標7） 解析規則や辞書管理のためのユーティリティ機能

2.2 既存のモデルについて

これまで人工知能研究分野で提案されてきた種々のパーザは形式言語理論で考えると次のように大別される（対象とする言語をLとする）．

† COMPLAN: A Programming System for Natural Language Analysis by TOYO-AKI NISHIDA and SHUJI DOSHITA (Department of Information Science, Faculty of Engineering, Kyoto University).
†† 京都大学工学部情報工学教室
* ここでは計算機マニュアルなどの制限された科学技術文献を対象とする．



ここで，非終端記号 NT に，意味表現 α が対応することを，NT: α と表している．
(a) 名詞句 an efficient algorithm に対する句構造，(b) 名詞句 an efficient algorithm の意味表現の計算過程．

図 1 関数型意味論に基づく言語記述の一例

Fig. 1 An example of language description based on functional semantics.

（タイプ1） G(L)+パーザ

（タイプ2） A(L)+規則インタプリタ

（タイプ1）のモデルでは，Lの文を生成する文法 G(L) を句構造文法で与えておき，入力文が与えられるとそれがどのように初期記号に還元されるかをパーザが解析する．LINGOL[12]，DCGS[11] はこのタイプであると考えられる．

一方，（タイプ2）のモデルでは，Lの文を受理するオートマトン A(L) の記述とそれを解釈実行する規則インタプリタによって入力文を解析する．ATN[16]，Word Expert Parser[13]，PARSIFAL[5] などはこのタイプであると考えられる．

この分類に基づいて前節の各設計目標を検討してみよう．一般に，設計目標1～3はタイプ1のモデルのように句構造規則という，言語の宣言的な記述方法に基づいたモデルのほうが望ましいと考えられる．とくに拡張 LINGOL[15] のような記述法を用いると，対象言語を関数的に記述することができる．これをモンテギュー文法[3] のような関数型意味論に基づく文法理論と組み合わせるとすぐれた性質が得られる．たとえば，an efficient algorithm という名詞句に対する句構造は 図1(a) のようになるが，この名詞句に対する意味表現は 図1(b) のように逐次得ることができる．このように，各部分構造が必ず対応する意味表現をもち，それが子の句構造のもつ意味表現だけから決定されるという性質は，大規模文法を作成し，デバッグするとき非常に有効な性質であると考えられる．

ところが，入力文の解析処理の面（設計目標4と5）からみれば，タイプ1のモデルでは，一様なパージングアルゴリズムを中心に解析を行わざるをえないので，解析過程を柔軟に制御することはできない．たとえば，イディオムの処理，動詞句パターンの処理，形態素解析，関係節の処理，などに関しては制御構造の柔軟なタイプ2のモデルのほうが有効であると考えられる．

解析システムの処理方式に関しては，入力文に対する可能な解析を全て並行的に求めるやり方と，後戻り処理をしながら一つずつ順に求めてゆくやり方とが考えられる．これについては，係り受けの制約条件に関する知識が不足していると長い入力文には組合せ的に多数の解析が可能になること，途中結果を保持するための記憶量，個々の可能性の計算に要する時間量，人間の診断の有効性などを考えると，後者の，順に解を求めてゆく方式のほうが適していると考えられる．

また，自然言語解析システム開発の段階では，辞書などの仕様の変更などがかなり起こりうると考えられるので，辞書の編集や変更を容易かつ一貫して行えるユーティリティが必要である．

2.3 設計方針

以上のような考察に基づき，以下のような方針で解

析システムを設計することにした．すなわち，対象言語を宣言的に記述するための文法規則と，対象言語の文を解析するときに使用する解析規則の2レベルの記述の階層を用いる．解析規則の作成者は，まず対象言語を文脈自由文法を拡張した形式（Augmented Context Free 文法，AUGCF 文法とよぶ）で記述する．AUGCF 文法は拡張 LINGOL の記述形式をさらに拡張したものである．次に，各 AUGCF 文法規則に対して，それが解析過程のどの局面でどのように用いられるかという情報を与えてタイプ分けし，手続き的な解析規則の中に埋め込む．この解析規則を解釈実行する規則インタプリタは後戻り制御を中心に規則適用の管理を行うものであり，トップダウンパーザやボトムアップパーザをサブルーチンとして呼び出すようになっている．この意味で，ここでの規則インタプリタはそれ自身は自然言語解析という問題解決過程を制御するだけの，専門家システム[4]でいう推論エンジンにあたる．

このような2段階の規則記述方式によって2.2節で検討した設計目標1～5を満足することができると考えた．

（設計目標6について）解析システムは入力文に対し，与えられた解析規則によって可能な解析を一つずつ出力する．これらの解析結果では，もとの入力文に多くの情報が付加されたものになっているので，人間はどの部分の解析が誤っているかを容易に判断できるものと考えられる．そこで，人間が解析結果の各部を対話的に診断することにより誤りを指摘すれば，それ以後は誤りを指摘された部分構造を含まない解析だけを行うようにして人間の対話的補助による解析処理の効率化を図る．また，規則作成者は行われた診断を分析してその結果を新たな知識として解析規則に組み込むことにより，解析規則自体の改良を行う．

（設計目標7について）解析規則や辞書の修正や変更は，実験的にいくつかの文を解析してみるための試行錯誤的な修正と，仕様変更などに伴う辞書項目などの一括変更に分けられる．このために，まず解析規則のファイルや辞書を文の解析中に直接参照し変更できるようにする．そして，一時的変更のためには辞書エディタを作成し，一括変更に対しては，辞書項目や解析規則をパターンとみなし，パターン変換規則によって適合するパターンを含む記述を一括変更するようなユーティリティを作成する．

3. 対象言語の記述

3.1 拡張文脈自由文法規則による統語と意味の記述

対象とする自然言語の統語と意味の記述は文脈自由文法規則を拡張した形式（AUGCF 規則）と辞書によって行う．はじめに AUGCF 規則について述べる．AUGCF 規則 R_n は次のような形式をしている．

$$R_n: A \xrightarrow{f_{syn},\ f_{score},\ f_{sem}} \beta_1 \cdots \beta_n \qquad (1)$$

ここでAは非終端記号であり，NP（名詞句），S（文）のような対象言語の文法カテゴリを表す記号を用いる．$\beta_i (i=1, \cdots, n)$ には非終端記号 B_i，または表現 $(B_i\text{-}C_i)$（これを削除項とよぶ）が許される．削除項 $(B_i\text{-}C_i)$ は，文法カテゴリ B_i の句構造から文法カテゴリ C_i の句構造がちょうど1個削除された構造を表す．たとえば英語の関係節は NP→NP. which.(S-NP) と記述することができる．C_i に対応する句構造が B_i のなかから削除されてどこへ移動したかは適宜に（本論文では破線・⇢で）示されるものとする．削除項の導入は文脈自由文法の拡張となっている．関係節のような構造は通常の文脈自由文法規則を用いると，(S-NP) を展開したすべてのパターンを記述せねばならず，規則の見通しと効率の両面が悪化してしまう．

各文法カテゴリに対応する句構造には，syn 値，score 値，sem 値とよばれる値が定義される．syn 値は，与えられた句構造について人称，数，性などに関する付加的情報を属性-属性値対のリストの形で表す．score 値はその句構造の評価値を，sem 値は対応する意味表現を表す．AUGCF 規則の f_{syn}, f_{score}, f_{sem} は，それぞれ親の句構造の syn, score, sem 値を子の句構造の syn, score, sem 値から計算する関数である．f_{syn}, f_{score}, f_{sem} が引数の組合せにより特殊な値（⊥：bottom）をとると規則の適用は中止される．この機構は，人称，数，性の一致や，ある種のヒューリスティックスや，係り受けに関する制約条件を記述するために用いることができ，これにより文脈自由文法規則の記述力が拡張できる．

3.2 辞書記述

辞書は次の形式で記述する：

〈単語〉：(〈文法カテゴリ〉, 〈syn〉, 〈score〉, 〈sem〉) (2)

ここで，〈syn〉, 〈score〉, 〈sem〉 部は，AUGCF 規則によって句構造の syn, score, sem 値を再帰的に構成するための初期値を与える．多品詞語や同音異義語に

は，同じ見出語をもつ複数個の項目が対応する．

3.3 拡張文脈自由文法による簡単な言語記述例

以上に述べた記述形式はモンテギュー文法のような関数型意味論に基づく言語記述に適していると考えられる．そこで，英語の小さな断片に対する文法記述の例（これをサンプル文法とよぶ）を**図2**に示す．サンプル文法において，たとえば AUGCF 規則 R1 は，文 (S) が主語の名詞句 (NP) と述語の動詞句 (VP) を派生することを表す．関数 subjvp は NP と VP の syn 値を調べて主語と述語動詞の一致をチェックする．subjvp は NP の syn 値と VP の syn 値の許される組合せを表にしてもつ．たとえば，NP(I) VP(△) have… のような入力文に対して，NP の syn 値は，NBR=SGL(単数)，PSN=1(一人称)，CASE=ϕ(格変化なし)で，VP の syn 値は，FORM=ORG (原形) となり，この組合せは許された組合せに含まれているが，NP(I) VP(△) has… のような入力文に対しては VP の syn 値が，FORM=+S となり，この組合せは許されない．また，R1 の f_{sem} にあたる部分 (*sem1(*sem2)) は，S の意味表現が NP の意味表現 α を VP の意味表現 β に適用した形 $\alpha(\beta)$ であることを示している．

このサンプル文法によって簡単な英文 This command needs no operand を記述した例を**図3**に示す．

4. 手続き的な解析規則とそのインタプリタ

AUGCF 規則は対象言語記述のためのものであり，それがただちに入力文解析に適しているというわけで

(a) サンプル文法の文法カテゴリ
初期記号：S (文)

ADJ	形容詞	NP	名詞句	VP	動詞句
DET	限定詞	PP	前置詞句	VT	他動詞
NOUN	普通名詞	PREP	前置詞	WHICH	関係代名詞

(b) サンプル文法の AUGCF 規則

R1: S $\xrightarrow{\text{subjvp}, +10, *\text{sem}_1\ (*\text{sem}_2)}$ NP. VP

R2: NP $\xrightarrow{\text{pm}_2, +10, *\text{sem}_1\ (*\text{sem}_2)}$ DET. NOUN

R3: NOUN $\xrightarrow{\text{pm}_2, +10, *\text{sem}_1\ (*\text{sem}_2)}$ ADJ. NOUN

R4: NP $\xrightarrow{\text{pm}_1, +0, \text{make-np-pp}}$ NP. PP

R5: PP $\xrightarrow{\text{ok}, +10, \text{make-prep-np}}$ PREP. NP

R6: NP $\xrightarrow{\text{pm}_1, +0, \text{make-np-rel}}$ NP. WHICH, (S−NP)

R7: VP $\xrightarrow{\text{pm}_1, +10, \lambda x[*\text{sem}_2(\lambda y[*\text{sem}_1(x, y)])]}$ VT. NP

ここで f_{syn}, f_{score}, f_{sem} として用いられた関数の定義は次の通りである：

f_{syn}: ok: 統語上の検査は行なわない(無条件に成功させる)
pm$_i$: i 番目の子の syn 値を返す．
subjvp: NP の syn 値と VP の syn 値を検査し，主語と述語動詞の一致を調べ，一致していなければ規則の適用を中止する．

f_{score}: $+\alpha$: 子の句構造の score 値をすべて加え，さらにそれに α を加える．

f_{sem}: *sem$_i$: i 番目の子の句構造の意味表現 (sem 値) をあらわす．make−np−pp, make−prep−np, make−np−rel はそれぞれ適切な意味表現を返すものとする．

(c) サンプル文法の辞書項目の例

This: (DET, ( ), 10, this')
command: (NOUN, (NBR=SGL, PSN=3), 10, command')
needs: (VT, (TENSE=PRES, FORM=+S), 10, need')
no: (DET, ( ), 10, no')
operand: (NOUN, (NBR=SGL, PSN=3), 10, operand')

図2 英語の小さな断片を記述するサンプル文法
Fig. 2 A sample grammar for a small fragment of English.

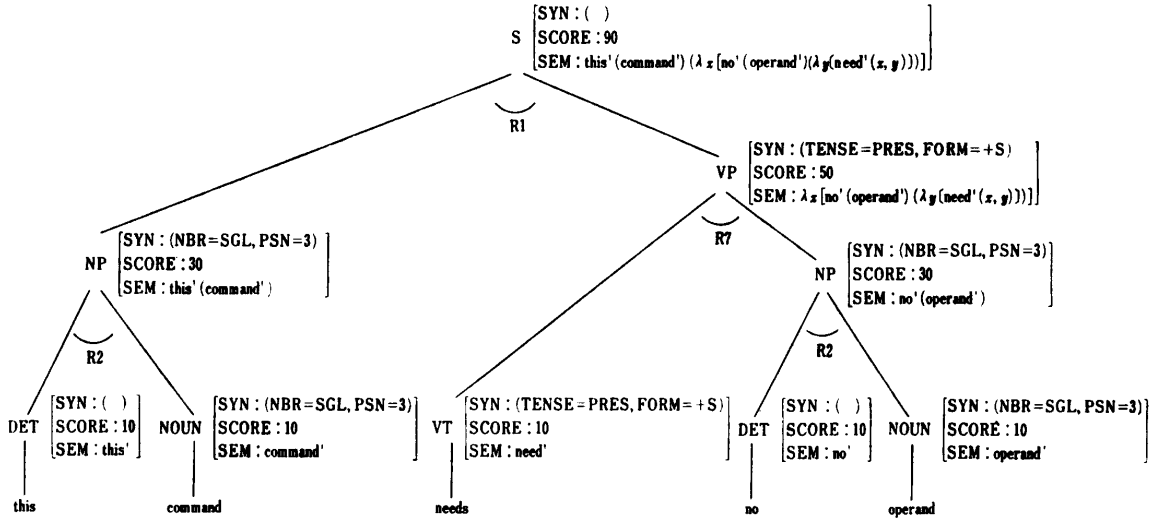
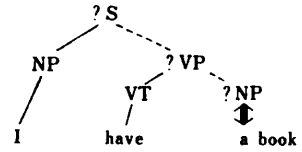


図3 サンプル文法による英文の解析例
Fig. 3 The analysis of a sentence in terms of the sample grammar.

パケット:		
	A	現在のゴールまたは次に実行すべき手続き
	B	ゴールまたは手続きのスタック
	C	スキャナの位置
	D	左部分木のスタック
	E	フラグ-COMP ノード対のスタック

(a) パケットの構造



(b) パージング過程の一状況

A	NP
B	((pop_2_elements_and_construct_a_P_Vnode) (pop_2_elements_and_construct_an_S_node) (SUCCEED))
C	3
D	($tree_3$ $tree_1$)
E	ϕ

$tree_1$: NP[…]
I
$tree_2$: VT[…]
have

(c) (b)の状況を記述したパケット

図 4 パケットの構造

Fig. 4 The structure of a packet.

はない．各 AUGCF 規則がどのように適用されるかという制御に関する記述を与えたほうがより効率的である．そこで各 AUGCF 規則を参照のされ方によってタイプ分けし，手続き的な規則の中に組み込む．たとえば，S→NP・VP という AUGCF 規則は，「S がゴールとして予測されたとき，それをサブゴール NP と VP に分解せよ」という手続き的な規則に変換し，NP_1→NP_2・PP という AUGCF 規則は，「句構造 NP_2 が構成されると，つづいてゴールとして PP を予測し，それが成功すれば句構造 NP_2 とあわせて句構造 NP_1 を構成せよ」という規則に変換する．以下では，まず，手続き的な解析規則のインタプリタについて述べ，次に解析規則について述べる．

4.1 手続き的な解析規則のインタプリタ

規則の適用の流れは，パケットとよぶデータ構造を用いて制御する．パケットは**図 4** (a) のように 5 個のスロットから成る．たとえば 図 4 (b) のような解析の局面を表すパケットは 図 4 (c) のようになっている．解析規則の適用は非決定的に行われ，それぞれの枝分かれに対応するパケットが存在する．それらのうちの 1 個（アクティブなパケットとよぶ）について処理が行われ，他のものは sstack とよばれるスタックに入れられる．

入力文解析は，sstack からパケットを一つずつポップアップし，それを変更したり新しくパケットを生成したりして再び sstack にプッシュダウンするという基本サイクルをくり返すことにより行われる．入力文が与えられると，まず初期記号 S がゴールとして設定される．ゴールは適当なサブゴールに分解されてゆき，辞書びきがされ，辞書記述をもとに解析規則によってしだいに入力文に対する統語・意味構造が構成されてゆく．

解析規則はこの過程のどの局面でどのように参照されるかによって，E-rule, U-rule, B-rule L-rule と分類される．E-rule は対応する文法カテゴリがゴールとして予測されたとき起動される．U-rule は対応する文法カテゴリの句構造が構成されたとき起動される．B-rule はボトムアップパーザから参照され，これには AUGCF 規則をそのまま用いる．L-rule は辞書中に埋め込まれたものであり，特定のゴールが予測されると起動される．B-rule を除く各解析規則は与えられたパケットに対する操作としてインプリメントされる．

4.2 手続き的な解析規則

4.2.1 E-rule

E-rule は次のような形式で記述する．

goal＝〈ゴール〉⇒…；〈条件$_i$〉→〈手続き$_i$〉；….

〈ゴール〉で示された文法カテゴリが予測されたとき，〈条件$_i$〉に記述された関数を評価し，それが真になれば〈手続き$_i$〉を起動する．二つ以上の条件が真になれば非決定的処理を行う．〈条件〉部には任意の LISP プログラムを書くことができる．同様に，〈手続き〉部にも任意の LISP プログラムを書くことができる．〈条件〉部，〈手続き〉部のためのシステム組込み関数を**表 1** に示す．

左回帰的でない生成規則をもつ AUGCF 規則は E-rule として記述できる．たとえばサンプル文法の R 1 は，

```
goal＝S⇒T→expand [(NP VP); subjvp;
        +10; ＊sem1(＊sem2)].
```

と書くことができる．また，辞書びきによって句構造が構成される文法カテゴリ PREP に対しては，

```
goal＝PREP⇒T→getfromdict [ ].
```

という E-rule を用いる．単純名詞句をボトムアップ解析するためには，

```
goal＝NP⇒T→bottomup [ ].
```

という E-rule を用いる．

表 1 E-rule のための組込み手続き
Table 1 Built-in Functions for E-rules.

＜条件＞部に使用する関数	?cat [category]	現在スキャンしている語が category で示された文法カテゴリの辞書項目をもてば真，さもなければ偽
	?lex [word$_1$; …; word$_n$]	現在スキャンしている語が word$_1$, …, word$_n$ のどれかであれば真，さもなければ偽
	?[feature; value]	現在スキャンしている語の属性 feature の値が value に等しければ真，さもなければ偽
	ブール関数 ∧, ∨, ¬	(LISP 組込み)
＜手続き＞部に使用する関数	expand [subgoals; f_{syn}; f_{score}; f_{sem}]	現在のゴールを subgoals に展開する．それらすべてが成功し，それぞれに対する句構造が構成されれば，その syn 値，score 値，sem 値にそれぞれ関数 f_{syn}, f_{score}, f_{sem} を適用し，ゴールに対応する句構造を構成する．
	bottomup [ ]	現在スキャンしている位置から，与えられたゴールを目標とするボトムアップ解析を行う．この解析では B-rule を参照する．
	getfromdict [ ]	現在スキャンしている語が与えられたゴールの文法カテゴリの辞書項目をもつかどうかを検査し，もっていればその語に対する句構造を構成する（もっていなければ処理は失敗する）．

4.2.2 U-rule

U-rule は，

constructed＝〈文法カテゴリ〉⇒…；〈条件$_i$〉
→〈手続き$_i$〉；….

の形成で記述する．この規則は，〈文法カテゴリ〉に記述された文法カテゴリの句構造が構成されたとき起動され，〈条件$_i$〉に記述された関数を評価して真になれば，〈手続き$_i$〉を起動する．〈条件〉部には E-rule の場合と同じ関数が使用できる．〈手続き〉部のための組込み関数には，次の consif を用意した：

consif [subgoals; new-category; f_{syn}; f_{score}; f_{sem}]:
新しく subgoals に与えられたゴールを設定する．もしそれらが全て成功すれば，new-category の文法カテゴリの句構造を，関数引数 f_{syn}, f_{score}, f_{sem} を subgoals に対応して構成された句構造の syn, score, sem 値に，それぞれ適用することにより，構成する．

左回帰的な生成規則をもつ AUGCF 規則は U-rule に書き換える．たとえば，サンプル文法の R4 は次のように書き換えられる．

constructed＝NP⇒
?cat [PREP]→consif [(PP); NP; pm$_1$; +0; make-np-pp]; …(R6に対応する規則)….

削除項 (B_i-C_i) の書き換え

削除項は解析過程を制御する関数 del*[category; comp] に書き換える．この関数は文法カテゴリ category をゴールとして予測する．そのとき comp で宣言された句構造（これを COMP ノードとよぶ）をパケットの E-スロットに保持しておく．以後の過程で COMP ノードと同じ文法カテゴリが予測されたとき，COMP ノードをとり出し，予測にあてはめてみてうまくいけば，削除項に対する解析は成功する．COMP ノードのとり出しは非決定的に行う．

COMP ノードの宣言は関数，

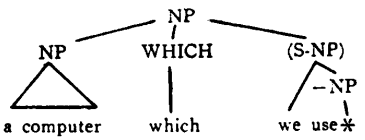
dclcomp [category; pos-of-antecedent; embedded-procedure]

によって行う．この関数は，文法カテゴリ category をもち引数 pos-of-antecedent の位置の先行詞と同じ syn, score, sem 値をもつ COMP ノードを初期化する．そしてこれを手続き embedded-procedure が修飾する．

サンプル文法の R6 は次のような U-rule に変換される．

constructed＝NP⇒
…(R4 に対応する規則)…;
?lex [WHICH]→
consif[
list [NP; del* [S; dclcomp [NP; ＊−2; make-antecedent]]]; NP; pm$_1$; +0; make-np-rel]. (3)

たとえば名詞句 a computer which we use は，



と分析される．このとき名詞句 a computer に対応する句構造は，

$$NP\begin{bmatrix}SYN: (NBR\ SGL,\ PSN=3)\\ SCORE: \cdots\\ SEM: a'\ (computer')\end{bmatrix}$$

となっているが，これをもとに (3) の dclcomp でつくられる COMP ノードは

$$NP\begin{bmatrix}SYN:(NBR=SGL,\ PSN=3)\\ SCORE:\cdots\\ SEM:\ it_n\end{bmatrix}$$

となる．ただし，it_n は新しく生成された変数である．これにより，(S-NP) の部分には，文 we use it_n に対応する意味表現が割り当てられ，全体の名詞句に対しては大体 a computer$_n$ [such that we use it_n] のような形の意味表現が割り当てられる．

4.2.3 B-rule

B-rule としては AUGCF 規則をそのまま用いる．B-rule は単純名詞句のように入力文の局所的な情報だけからデータ駆動型解析を行うのが適している場合に用いる．たとえばサンプル文法では，R2 と R3 を B-rule にする．

4.2.4 L-rule

L-rule は辞書中の各語の中に分散されていて，その語が入力文中に出現したとき対応する文法カテゴリが予測されていると起動される．L-rule はおもにイディオムなどの解析に用いると有効である．L-rule の形式は，

(L-RULE (〈ゴール$_1$〉(条件$_{11}$)→〈手続き$_{11}$〉)…)…)

となっており，イディオムの先頭の語の辞書項目中に記述する．たとえば，either A or B というイディオムに対し，either の辞書項目中に，

```
(L-rule (NP (T→expand[(=either NP =or NP);
    ck-either-or; +10; mk-either-or])))
```

という記述を与えておく．すると，NP がゴールとして予測されたとき，either がスキャンされていると無条件（T）で expand […] が実行される．NP というゴールは =either, NP, =or, NP というサブゴールの列に展開され，それらが全て成功するとそこから NP の句構造が構成される．

L-rule により，解析規則を辞書の中に分散し，E-rule や U-rule が不用意に増加することを防いでいる．

4.3 AUGCF 規則から手続き的な解析規則への変換

AUGCF 規則から手続き的な解析規則への変換は規則作成者（人間）によって行われるが，これは規則的に行われ，与えられた AUGCF 規則をどのタイプの解析規則に変換するかさえ決めればよい．得られた解析規則は起動条件をチェックするプログラムを強化したり，等価な働きをするプログラムにおきかえて効率化してゆくことができる．

5. 対話的診断機能について

手続き的な解析規則のインタプリタは入力文に対して可能な解析結果を一つずつ順に出力する．人間はコマンドを用いて解析結果の各部をスキャンし，syn, score, sem 値や統語構造を表示させ，適切な解析結果が得られているかどうかをチェックする．そして，適切な解析結果が得られた部分とそうでない部分とをそ

表 2 診断用コマンド
Table 2 List of diagnostic commands.

SCAN α	現在の句構造の子孫で，パターンαにマッチする句構造を順にトラバースする．
F(orward)	SCAN α の文脈下において，次にαにマッチする句構造に移動する．
B(ackward)	SCAN α の文脈下において，前にトラバースした句構造にもどる．
UP	現在の句構造の親に移動する．
DOWN	UP による移動をもとにもどす．
D(isplay) n	現在の句構造をnレベルの深さまで分解して表示する．
SYN	現在の句構造の syn 値を表示する．
SCORE	現在の句構造の score 値を表示する．
SEM	現在の句構造の sem 値を表示 (pretty print) する．
TREE	現在の句構造の支配する句構造を木の形に表示する．
GOOD α	現在の句構造の子孫でαにマッチする句構造が適切なものであるという指示を与える．αに複数個の句構造がマッチすれば対話的に決定する．
NOGOOD α	現在の句構造の子孫で，αにマッチする句構造が不適切なものであるという指示を与える．αに複数個の句構造がマッチすれば対話的に決定する．
ACCEPT	現在の解析結果を受理し，入力文解析を終了する．
REJECT	現在の解析結果を棄却する．ひきつづき，GOOD, NOGOOD で指摘された制約内で解析を続行する．
EXHAUST	全探索モードに切換え

〈手続き〉::=(FOREACH / FORFIRST / FORUNIQUE) (PATTERN 〈パターン〉) (IN 〈範囲〉) (DO 〈手続き〉) [(CHECK・YES)])|
(IF 〈条件式〉 THEN 〈手続き〉 [ELSEIF 〈条件式〉 THEN 〈手続き〉]*[ELSE 〈手続き〉])|
(PUT 〈パターン〉 IN 〈範囲〉)|
(REMV 〈パターン〉 FROM 〈範囲〉)|
(PRINT 〈パターン〉)|…
〈範囲〉::=DICT|〈変数〉
〈条件式〉::=(〈変数〉 HAS 〈パターン〉)|…
〈パターン〉::=〈定数〉|〈変数〉|(〈パターン〉[〈パターン〉]*)|(〈変数〉 WHERE 〈条件式〉)

ただし，α, α_i をメタ式とするとき，記法 $\left\{\begin{matrix}\alpha_1\\ \vdots\\ \alpha_n\end{matrix}\right\}$, $[\alpha]$, $[\alpha]^*$ はそれぞれ $\alpha_1,\cdots\alpha_n$ からの選択，α のオプショナルな選択，α の0回以上のくり返しを意味する．

図 5 辞書管理のためのコマンド言語の構文（概要）

Fig. 5 Syntax of a command language for dictionary management.

れぞれ規則インタプリタに指示することができる．規則インタプリタは，適切であると指示された部分に対しては，他の可能性を消去し，解析を固定する．不適切であると指示された部分に対しては，得られた解析結果を消去し，他の可能性を試みる．以後の解析過程では，不適切であると指示された部分構造を含まない範囲内で解の探索が行われ，人間の診断結果が解析処理に反映される．この機能により，うまく診断を行うと，解析規則に組み込まれた，係り受けなどの制約条件に関する知識が不足していても適切な解析結果にすばやく到達できることが実験的に判明した（7章参照）．対話的診断に用いるコマンドを**表2**に示す．

規則インタプリタは，解析の各時点において，入力文の各部分に対してそれまでに行われた解析結果と構成された部分構造を記憶するようになっており，同一の部分に対して同一の解析が2度以上行われないようになっている．対話的診断が行われると，不適切であると指示された部分構造，およびそれらを子孫にもつ全ての句構造が消去される．

6. 解析規則と辞書管理のためのユーティリティ

入力文の解析実験において必要なときただちに解析規則や辞書の変更や登録を行えるようにするため，これらをオンラインファイル化し，簡単なリスト構造エディタによって編集が行えるようにした．とくに，未登録語に遭遇したとき類似の語の辞書項目をコピーし，修正を加えてただちに登録できるようにしている．

一方，仕様変更などに伴う辞書項目の一括修正は簡単なコマンド言語とインタプリタによって行う．たとえば，「文法カテゴリ NOUN の全ての辞書項目について，〈sem〉部において，(SEM・α) というパターンをもつが，(ISA・β) というパターンをもたないものには (ISA・α) というパターンを追加せよ」というコマンドは，

(FOREACH (PATTERN (x NOUN y z u v))
(IN・DICT)
(DO (IF ((v HAS (SEM・α)) & (NOT (v HAS (ISA・β)))) THEN (PUT (ISA・α) IN v))))

と記述される．ここで，$x, y, z, u, v, \alpha, \beta$ はパターン変数である．たとえば，辞書項目，

(COMPUTER NOUN…(<u>(SEM・COMPUTER)…</u>))
〈sem〉部

は*

(COMPUTER NOUN
…(<u>(SEM・COMPUTER) (ISA・COMPUTER)…</u>))
〈sem〉部

と修正される．コマンド言語で (CHECK・YES) と指定すると，修正をチェックすることができる．辞書管理のためのコマンド言語の構文を**図5**に示す．

7. COMPLAN による英文解析の実験

以上述べたシステムを研究室の LISP 1.7[2] 上に作成した．パージングの基本アルゴリズムに相当する部分は約400行であったが，ユーティリティを含めたシステム全体のプログラム行数は2,000行以上になる．

このシステム上で，実際の計算機マニュアルやテクニカルレポートに現れる文を解析するための辞書，解析規則，形態素解析プログラムを試作した．この英文解析システムは，英語の入力文をモンテギュー文法に基づく内包論理式[8),9)]に変換するものであり，現在主として英文和訳システム[10)]において用いられている．試作した解析規則は約150の AUGCF 規則に対応し

* 辞書項目〈単語〉: (〈文法カテゴリ〉, 〈syn〉, 〈score〉, 〈sem〉 に対するパターン記述は (〈単語〉〈文法カテゴリ〉〈syn〉〈score〉〈sem〉) とする．

(1) The assembly language provides a means for writing a program without having to be concerned with actual memory addresses or machine instruction formats. (more than 41, 41番目以後, 40秒, 3)
(2) Labels (symbols) can be assigned to a particular instruction step in a source program to identify that step as an entry point for use in subsequent instructions. (more than 30, 30番目以後, 1分, 3)
(3) The source program is processed by the assembler to obtain a machine language program (an object program) that can be executed directly by the Z 80. (more than 11, 1番目, 3分58秒, 1)
(4) Ethernet is a branching broadcast communication system for carrying digital data packets among locally distributed computing stations. (6, 2番目, 40秒, 2)
(5) The packet transport mechanism provided by Ethernet has been used to build systems which can be viewed as either local computer networks or loosely coupled multiprocessors. (42, 1番目, 45秒, 1)
(6) An Ethernet's shared communication facility, its Ether, is a passive broadcast medium with no central control. (12, 8番目, 30秒, 3)
(7) Coordination of access to the Ether for packet broadcasts is distributed among the contending transmitting stations using controlled statistical arbitration. (24, 6番目, 45秒, 2)
(8) Switching of packets to their destinations on the Ether is distributed among the receiving stations using packet address recognition. (8, 6番目, 35秒, 2)
(9) VERSAbus incorporates state-of-the-art features to support system architecture built around multiple microprocessors. (3, 2番目, 22秒, 2)
(10) The asynchronous operation of VERSAbus allows very slow peripherals to be matched to high speed processors without significant reduction to the overall system speed. (more than 25, 13番目, 48秒, 3)

各文の後の（ ）内の数字：(可能な解析の数，適切な解のある位置，最初のパーズ結果が得られるまでの時間，対話的診断機能を用いたとき適切な解が得られるまでの診断回数)

例文の引用

文 1～3: Zilog: "Z 80—Assembly Language programming Manual", 1977.
文 4～8: Metcalfe, R. M. and Boggs, D. R.: "Ethernet: Distributed Packet Switching for Local Computer Networks", Xerox Palo Alto Research Center, CSL-75-7, reprinted 1980.
文 9～10: Motorola: "VERSAbus Preliminary Specification", 1979.

図 6 COMPLAN 上に作成した英文解析システムによって解析された文の例

Fig. 6 Examples of sentences analyzed by an English analyzer built on COMPLAN (together with quantitative results).

ている．**図6**にこのシステムによって解析された例文のいくつかを解析に要した時間・診断回数などとともに示す．たとえば，例文（7）に対しては試作した解析規則によって可能な解析は 24 通りあり，適切な解は6番目にある．システムが最初の解析結果を出力するまで 46 秒を要した．これは適切なものではないので，診断情報を与えて棄却すると次に適切な解析結果が出力された．したがってこの診断によって4個の不適切な解析結果がスキップされ，解析結果が適切であることを判断するための診断とあわせて計2回の診断で適切な解に到達したことになる．

試作した解析規則には係り受けなどの制約条件に関する知識がまだ十分に組み込まれていないため，各入力文に対して多数の不適切な解析結果が存在するが，対話的診断機能により比較的少ない診断回数で目的とする解析結果が得られた．また，種々のユーティリティ機能により解析規則や辞書作成を効率よく行うことができた．パージングアルゴリズム自体は特殊なものを用いなかったが，E-rule や U-rule の条件部に簡単なチェックプログラムを書くことでむだな探索を極力省くことができ，また中間結果を記憶する後戻り処理によって，実験的使用に耐える程度の応答速度が得られた．解析の作業用領域は 6k～10k セル程度の自由セル領域でガーベージコレクションを行えば足りた．以上の実験結果は当初設定した設計目標1～7をほぼ満足するものであった．

8. 検　討

本論文で述べた手法は，文脈自由文法規則を中心とする記述の明確さと，手続き的な解析規則による効率的な解析の利点を組み合わせたものである．このような方向では，パターン記述の導入により ATNG の記述力を改善しようとした PLATON[6] や，文脈自由文法に手続き的な拡張を加えた拡張 LINGOL[15], DIAGRAM[14] などがある．これらに対し，本論文の手法は言語の記述のレベルと解析のレベルの2層を別々に設定しておいてそれぞれの役割に適合する記述法を用い，しかる後に両者を対応づけたところに特色がある．本論文では，自然言語解析システム開発において実験を通して解析規則や辞書を作成・修正してゆく過程を重視しており，そのために規則に組み込まれた知識の不完全性を補う対話機構や，辞書・解析規則管理のためのユーティリティを開発した．この解析システムは，対話的診断機能により，解析規則に組み込まれた知識が不十分でも人間の補助によりマンマシンシステムとして稼動し，また十分な知識が与えられると人間の補

助が少なくなるという柔軟性をもつ.

今後の課題としては,

(1) 人間の診断による解析規則の学習的作成・修正機能の組込み

(2) 各句構造の syn, score, sem 値とその上の関数をより問題指向で抽象度の高いものにおきかえること

があげられる.

本研究の一部は文部省科学研究費による.

参考文献

1) Charniak, E.: Six Topics in Search of a Parser : An Overview of AI Language Research, Proc. IJCAI-81, pp. 1079-1087 (1981).
2) 堂下, 平松, 角井: 直接アクセス方式のバルクメモリを用いた LISP システムの作成, 信学論 (D), Vol. J-61 D, No. 5, pp. 360-361 (1978).
3) Dowty, D.: *Introduction to Montague Semantics*, Reidel (1981).
4) Feigenbaum, E.: The Art of Artificial Intelligence : I. Themes and Case Studies of Knowledge Engineering Proc. IJCAI-77, pp. 1014-1029 (1977).
5) Marcus, M.P.: *A Theory of Syntactic Recognition for Natural Language*, The MIT Press (1980).
6) 長尾, 辻井: 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, pp. 654-661 (1974).
7) 長尾, 辻井: 自然言語処理プログラム, 情報処理, Vol. 18, No. 1, pp. 63-75 (1977).
8) Nishida, T. and Doshita, S.: Hierarchical Meaning Representation and Analysis of Natural Language Documents, Proc. COLING-80, pp. 85-92 (1980).
9) Nishida, T. and Doshita, S.: A Knowledge-based Literature Guide System, Proc. IFIP Congress 80, pp. 699-704 (1980).
10) 西田, 清野, 堂下: モンテギュー文法に基づく英文和訳システムの試作, 情報処理学会論文誌, Vol. 23. No. 2, pp. 107-115 (1982).
11) Pereira, F.C.N. and Warren, D.H.D.: Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augmented Transition Networks, AI 13, pp. 231-278 (1980).
12) Pratt, V.R.: A Linguistic Oriented Programming Language, Proc. IJCAI-73, pp. 372-381 (1973).
13) Rieger, C. and Small, S.: Word Expert Parsing, Proc. IJCAI-79, pp. 723-728 (1979).
14) Robinson, J.J.: DIAGRAM : A Grammar for Dialogues, *SRI Technical Note* 205 (1980).
15) 田中, 佐藤, 元吉: 自然言語処理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, Vol. 60, D 12 (1977).
16) Woods, W.A.: Transition Network Grammars for Natural Language Analysis, *Comm. ACM*, Vol. 13, No. 10, pp. 591-606 (1970).

(昭和 56 年 10 月 15 日受付)
(昭和 57 年 1 月 20 日採録)