Vol. 23 No. 2 情報処理学会論文誌 Mar. 1982

モンテギュー文法に基づく英文和訳システムの試作†

西田豊明†† 清野正樹†† 堂下修司††

モンテギュー文法の枠組に基づいて英文和訳を行うシステムを試作したので報告する．このシステムは，一種の内包論理式（λ-深層構造）を中間表現として用い，英文解析，トランスファ，日本語生成という3段階で翻訳処理を行う．英文解析の段階では入力文を解析してλ-深層構造を得る．このλ-深層構造は英語よりのものであるので，これに対し，トランスファの段階で単語の置き換えと簡単な構造変形を行うことにより，これを日本語よりのλ-深層構造に変換する．最後に，日本語生成段階で，日本語よりのλ-深層構造から日本語の統語構造を生成し，形態素の処理を行って最終的な出力文を得る．

機械翻訳にモンテギュー文法の考え方を導入することによる主な長所としては，内包論理に基づく深いレベルの中間表現の使用が英語と日本語のように語族の異なる言語間の翻訳処理に適していること，モンテギュー文法における関数的意味論あるいは準同型写像の考え方に基づいて翻訳過程を関数的プログラムによって実現できるのでシステムが単純化し，見通しがよくなること，等が考えられる．

試作システムは研究室のミニコン上のLISPシステム（LISPI 1.7）上に作成した．このシステムによって，まず基本的な文を対象に翻訳実験を行い，次に，実際のテキスト（計算機マニュアルの一部）の翻訳を行って本方式の有効性を確かめた．

1. まえがき

計算機による異なる自然言語間の自動翻訳（機械翻訳）の研究は長い歴史をもつが[4),8)]，英語と日本語のように異なった語族に属していて異なった性格をもつ言語間の翻訳には深い処理が必要となり，その結果，翻訳システムも複雑化し，見通しも悪くなる傾向にある．そこで，翻訳という情報処理過程を基礎づける強力で系統だった言語理論に基づいて翻訳システムを構成することが必要となる．自然言語の数理的枠組として提案され，その後研究を続けられてきたモンテギュー文法[3),6),7)]はこのような基礎理論として有効な枠組であると考えられる．

モンテギュー文法では，あいまい性を含みうる言語の表現は統語規則によって（複数個の）あいまい性を含まない言語（disambiguated language）の表現に対応づけられ，そこから翻訳規則によって内包論理式に翻訳され，意味解釈規則によってモデル理論的解釈を受ける．統語規則，翻訳規則，意味解釈規則は準同型写像の制約のもとに構成されており，意味解釈は統語構造を下から上に逐次たどることにより行われる．モンテギュー文法における準同型性の原理は関数的プログラミングによるシステムの単純化と見通しのよさにつながり，一方，意味解釈まで含んだ枠組は英語と日本語のように言語間の距離が大きい場合の翻訳に有効であることが期待できる．

我々は，モンテギュー文法に基づく機械翻訳の第一歩として内包論理式のレベルを中間表現とした英文和訳システムの構成を試みた．英語から日本語への翻訳は，英語の解析，トランスファ，日本語の生成という3段階で行う．ただし，モンテギュー文法で用いられている内包論理では内包と外延の区別を常に意識せねばならず，そのため処理が複雑になるので，より単純にすべての表現を内包的に取り扱うCresswellのλ-深層構造[1)]を英語，日本語間の中間表現として用いた．中間表現としては，英語の語彙をもつものと日本語の語彙をもつものの2種類を準備し，トランスファ過程で語彙や表現法の変換を行う．

本稿の以下の部分では，このシステムの翻訳処理機構について述べる．次に，実験結果を示し，このシステムの有効性，改良すべき点などについて検討したい．

2. 英語と日本語の中間表現

英語と日本語の中間表現となるλ-深層構造はλ-範疇言語で記述される．英語のためのλ-深層構造と日本語のためのものとは語彙が異なるため，それぞれを区別し，EFR (English-oriented Formal Representation), JFR (Japanese-oriented Formal Representation) と

† The Development of an English-Japanese Machine Translation System based on Montague Grammar by TOYO-AKI NISHIDA, MASAKI KIYONO and SYUJI DOSHITA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学教室

よぶ.

2.1 λ-範疇言語

λ-範疇言語では，各表現に統語範疇（以下，タイプとよぶ）が唯一に割り当てられる．基本タイプとして0（文）と1（名辞）のみが与えられる．これはそれぞれのタイプの表現が指示値 (denotation) として文および個体をもつことを示す．複合タイプ $\langle\alpha, \beta_1, \cdots, \beta_n\rangle$ は，このタイプの表現の指示値が β_1 から β_n までのタイプの表現の指示値の直積からタイプ α の表現の指示値への関数であることを示す．たとえば，タイプ $\langle 0,1\rangle$ は1引数述語のクラスを，タイプ $\langle 0,\langle 0,1\rangle\rangle$ はそれをさらに文のクラスに写像する高階の述語のクラスを示す．

タイプ α の表現の集合 E_α は次のようにして得られる：まず，各タイプには互いに素な基本表現の集合が対応づけられる．次に，複合表現は次の二つの規則によって帰納的に定義される：

（i） 関数適用

$\delta\in E_{\langle\tau,\sigma_1,\cdots,\sigma_n\rangle}$, $\alpha_i\in E_{\sigma_i}(1\leqq i\leqq n)$ のとき，

$\delta(\alpha_1,\cdots,\alpha_n)\in E_\tau$.

（例） $\text{the}\in E_{\langle\langle 0,\langle 0,1\rangle\rangle,\langle 0,1\rangle\rangle}$, $\text{system}\in E_{\langle 0,1\rangle}$ とすると，$\text{the (system)}\in E_{\langle 0,\langle 0,1\rangle\rangle}$.

（ii） λ-抽象化

β がタイプ σ の変数，$\alpha\in E_\tau$ のとき，

$\lambda\beta[\alpha]\in E_{\langle\tau,\sigma\rangle}$.

（例） $\text{analyze}\in E_{\langle 0,1,1\rangle}$，$x$ と y をタイプ1の変数とすると，$\lambda x[\text{analyze}\,(x,y)]\in E_{\langle 0,1\rangle}$.

このように定義されたλ-範疇言語の表現は，一意的にタイプが定まり，無あいまいである．

2.2 英語の中間表現（EFR）

英語の各文法範疇に λ-深層構造のタイプを文献1）に従って対応づける．これを**表1**に示す．英語の各文法範疇に属する単語に，λ-範疇言語の対応するタイプの要素(語)を割り当てる．英語の多品詞語には，それぞれの品詞に対応するタイプの語を割り当てる．たとえば，"design" という英単語には，$\text{design}_{\text{vt}}\in E_{\langle 0,1,1\rangle}$, $\text{design}_{\text{noun}}\in E_{\langle 0,1\rangle}$ という二つの語を割り当てる．

EFR には英語の機能語に属する語がいくつか含まれる．これを**表2**に要約する．

英語の統語規則を句構造規則によって表わす．句構造 α が句構造規則，

$$A\rightarrow B_1\cdots B_n \qquad (1)$$

によって句構造 $\beta_1\cdots\beta_n$ から構成されたもの，すなわち

表1 英語の文法範疇に対する EFR のタイプの対応づけ
Table 1 Type assignments for English syntax.

英語の文法範疇	タイプ
S: 文	0
なし（名辞）	1
VP: 動詞句	⟨0,1⟩
VI: 自動詞句	⟨0,1⟩
NP: 名詞句，NC: 名詞節	⟨0,⟨0,1⟩⟩
NOUN: 名詞	⟨0,1⟩
DET: 限定詞	⟨⟨0,⟨0,1⟩⟩,⟨0,1⟩⟩
ADJ: 形容詞（句，節）	⟨⟨0,1⟩,⟨0,1⟩⟩
ADV-S: 文副詞	⟨0,0⟩
ADV-VP: 動詞句副詞	⟨⟨0,1⟩,⟨0,1⟩⟩
PREP: 前置詞	⟨⟨0,0⟩,1⟩
PREPP-ADJ: 形容詞的前置詞句	⟨⟨0,1⟩,⟨0,1⟩⟩
PREPP-ADV: 副詞的前置詞句	⟨0,0⟩
AUXV: 助動詞	⟨⟨0,1⟩,⟨0,1⟩⟩
C-CONJ: 等位接続詞	⟨0,0,0⟩
S-CONJ: 従属接続詞	⟨⟨0,0⟩,0⟩

表2 機能語の働きをする EFR の語
Table 2 EFR atoms for functional words.

語	タイプ	機能
a*	⟨⟨0,⟨0,1⟩⟩,⟨0,1⟩⟩	表層に違しない限定詞
comp	⟨0,1⟩	比較属性によって特定される性質
which	⟨⟨⟨0,1⟩,⟨0,1⟩⟩,0⟩	関係節形成
*poss	⟨⟨⟨0,⟨0,1⟩⟩,⟨0,1⟩⟩,1⟩	所有格形成
*en	⟨⟨0,1⟩,⟨0,1,1⟩⟩	受動態形成
# ante	⟨0,⟨0,1⟩⟩	削除された先行詞のトレース
that	⟨⟨0,⟨0,1⟩⟩,0⟩	that 節形成
inf	⟨⟨0,⟨0,1⟩⟩,⟨0,1⟩⟩	名詞的不定詞形成
parti	⟨⟨⟨0,1⟩,⟨0,1⟩⟩,⟨0,1⟩⟩	現在分詞形成
whether	⟨⟨0,⟨0,1⟩⟩,0⟩	名詞化された Yes-No 疑問文形成
nom	⟨⟨0,⟨0,1⟩⟩,⟨0,1⟩⟩	動名詞形成
*ap	⟨⟨⟨⟨0,1⟩,⟨0,1⟩⟩,1⟩,⟨⟨0,0⟩,1⟩⟩	前置詞を形容詞的前置詞に変換するオペレータ
*pl	⟨⟨0,1⟩,⟨0,1⟩⟩	名詞の複数化
*and/*or	⟨⟨0,⟨0,1⟩⟩,⟨0,⟨0,1⟩⟩,⟨0,⟨0,1⟩⟩⟩	名詞句を結ぶ and と or

$$\begin{array}{ccc} A:\alpha & & \\ \swarrow & \searrow & \\ B_1:\beta_1 & \cdots & B_n:\beta_n \end{array}$$

であるとすると，α の EFR, $s(\alpha)$ は，

$$s(\alpha)=f(s(\beta_1),\cdots,s(\beta_n)) \qquad (2)$$

によって得られる．ただし，f は句構造規則(1)に対応する関数であり，$s(\alpha), s(\beta_1),\cdots,s(\beta_n)$ はそれぞれ，$\alpha,\beta_1,\cdots,\beta_n$ の EFR である．このような方針で英語の句構造に EFR を対応づけた例を**図1**に示す．

2.3 日本語の中間表現（JFR）

日本語に対しても英語と同様の方針で λ-深層構造 (JFR) を割り当てる．日本語の各文法範疇に対する JFR のタイプの対応づけを**表3**に要約する．英語の場合とのちがいは助詞の扱いである．一部の助詞（必須格を表わす格助詞，「が」，「に」，「を」等と格助詞

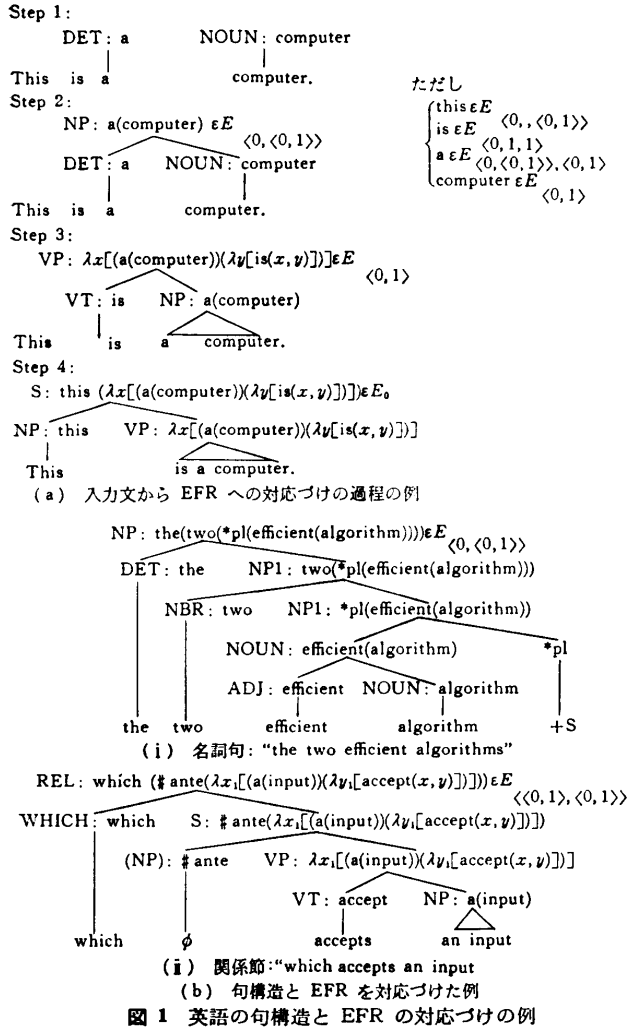

図 1 英語の句構造と EFR の対応づけの例
Fig. 1 Examples of associations between English phrase structures and EFR's.

「の」) 以外は英語の前置詞と同様の取り扱いをする.「の」は名詞句と結びついて形容詞の役割を果たすようにし, 必須格を表わす助詞に対応する要素は JFR に含めないで特殊な取り扱いをする. **図 2** に日本語の統語構造と JFR の対応づけの例を示す.

3. 英語の解析

英語解析の過程では, 入力文をパーズして EFR を抽出する.

3.1 英語の文法記述

第 2 章で示したような英語の句構造と EFR との対応づけの規則を次のような一連の拡張文脈自由文法規則 (以下, AUGCF 規則とよぶ) の形で記述する:

$$\mathrm{R}: \mathrm{A} \xrightarrow{\langle \mathrm{syn} \rangle, \langle \mathrm{score} \rangle, \langle \mathrm{sem} \rangle} \beta_1 \cdots \beta_n \qquad (3)$$

ここで A は非終端記号, B_i は非終端記号 B_i または表現 B_i−C_i. B_i−C_i は B_i というラベルをもつ句構造から C_i というラベルをもつ句構造がちょうど 1 個削除されていることを示す.

(3) において, $A \rightarrow \beta_1 \cdots \beta_n$ の部分は句構造生成規則である. 〈syn〉部は, 人称, 数, 性, 語形変化などに関する情報を統語素性として伝達し検査する関数である. 〈score〉部は規則の適用順位を決めるスコアを計算する関数である.

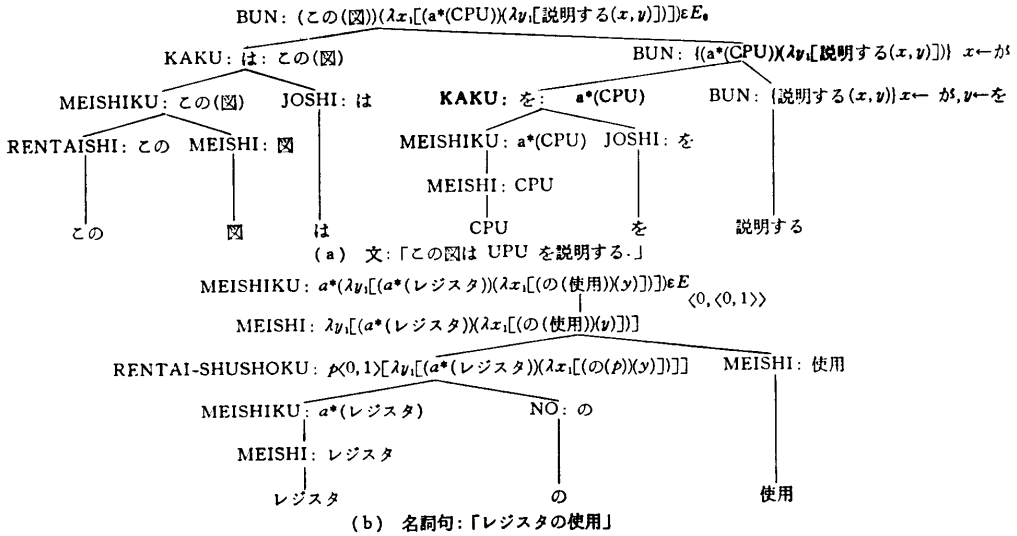

図 2 日本語の統語構造と JFR の対応づけの例
Fig. 2 Examples of associations between Japanese and JFR.

表 3 日本語の文法範疇に対する JFR のタイプの対応づけ
Table 3 Type assignments for Japanese syntax.

日本語の文法範疇	タイプ
BUN: 文	0
なし (名辞)	1
DOUSHI: (各種の) 動詞	⟨0, 1, ⋯, 1⟩
MEISHI: 名詞	⟨0, 1⟩
MEISHIKU: 名詞句	⟨0, ⟨0, 1⟩⟩
KEIYOUSHI: 形容詞	⟨⟨0, 1⟩, ⟨0, 1⟩⟩
KEIYOUDOUSHI: 形容動詞	⟨⟨0, 1⟩, ⟨0, 1⟩⟩
RENTAI-SHUSHOKU: 連体修飾	⟨⟨0, 1⟩, ⟨0, 1⟩⟩
FUKUSHI: 副詞 文修飾	⟨0, 0⟩
用言修飾	⟨⟨0, 1⟩, ⟨0, 1⟩⟩
RENYOU-SHUSHOKU: 連用修飾	⟨⟨0, 1⟩, ⟨0, 1⟩⟩
RENTAI-SHI: 連体詞	⟨⟨0, ⟨0, 1⟩⟩, ⟨0, 1⟩⟩
JOSHI: 助詞	
KAKU-JOSHI-1: 必須格を表わす格助詞	特別な扱い
KAKU-JOSHI-2: 自由格を表わす格助詞	⟨⟨0, 0⟩, 1⟩
NO: 格助詞「の」	⟨⟨⟨0, 1⟩, ⟨0, 1⟩⟩, 1⟩
SETSUZOKU-JOSHI: 接続助詞	
SETSUZOKU-JOSHI-1: 文連結の接続助詞	⟨⟨0, 0⟩, 0⟩
SETSUZOKU-JOSHI-2: 名詞句連結の接続助詞	⟨⟨0, ⟨0, 1⟩⟩, ⟨0, ⟨0, 1⟩⟩, ⟨0, ⟨0, 1⟩⟩⟩
JODOUSHI: 助動詞	⟨⟨0, 1⟩, ⟨0, 1⟩⟩
ASPECT: アスペクト形式素	⟨⟨0, 1⟩, ⟨0, 1⟩⟩
MODAL: モーダル形式素	⟨⟨0, 1⟩, ⟨0, 1⟩⟩

⟨sem⟩ 部はAの λ-深層構造を $\beta_1 \cdots \beta_n$ の λ-深層構造から計算したり，意味素性を伝達し検査する関数である．⟨syn⟩ 部，⟨sem⟩ 部の関数が扱うデータ構造は属性—属性値対のリストの形をしている．

一方，辞書は次のような形式で記述する：

⟨単語⟩:-(⟨文法範疇⟩; ⟨syn⟩; ⟨score⟩; ⟨sem⟩) (4)

辞書項目のうち，⟨syn⟩ 部，⟨sem⟩ 部はそれぞれ，その単語の統語素性および意味素性（λ-深層構造を含む）が記述される．⟨score⟩ 部はその辞書項目のスコアである．いわゆる同音異義語には（4）の形式の辞書項目が複数個対応することになる．

（例1） 主語—動詞句の規則は，

$$\text{R 10}: \text{S} \xrightarrow{\text{syn}_{10},\ \text{score}_{10},\ \text{sem}_{10}} \text{NP}\cdot\text{VP} \quad (5)$$

と表わされる．ここで，関数 syn_{10} は主語の名詞句と述語動詞が数・性・時制において一致しているかどうかを検査し，一致していなければこの規則の適用を禁止する．一方，関数 sem_{10} は NP に対応する EFR を VP に対応する EFR に適用 (apply) した形の EFR をつくる関数である．

（例2） 関係節は，

$$\text{R 11}: \text{NP}_1 \xrightarrow{\text{syn}_{11},\ \text{score}_{11},\ \text{sem}_{11}} \text{NP}_2\cdot(\text{S-NP}_1) \quad (6)$$

```
NP --> DET.NP1
NP --> NP1
NP --> PRON
NP --> *POSS.NP1
NP --> NP.PREPP
NP --> NP.VPP
NP --> NP.VING
NP --> NP.WHICH.S-NP
NP --> NP.THAT.S-NP
NP --> NP.WHO.S-NP
NP --> NP.WHOM.S-NP
NP --> NP.PREP.WHICH.NP.VP-PREPP
NP --> NP.WHOSE.NP1.S-NP
NP --> NP.NP.VP-NP
NP --> WHAT.S-NP
NP --> WHETHER.S
NP --> THAT.S
NP --> INF
NP --> NOM
NP --> NP.CONJ.NP
NP --> UNKNOWN.COMMA.NP
NP --> NP.UNKNOWN

NP1 --> NOUN
NP1 --> NBR.NP1

NOUN --> ADJP.NOUN
NOUN --> NOUN.NOUN
NOUN --> VPP.NOUN
NOUN --> VING.NOUN

*POSS --> DET.NP1
```

図 3 AUGCF 規則の一部（名詞句および名詞節に関する AUGCF 規則について句構造生成規則部を示した．）
Fig. 3 Part of AUGCF rules for English. (The production rule sections of AUGCF-rules for noun phrases and noun clauses are shown.)

と表わされる．R 11 は，1個の文（S）からちょうど1個の名詞句 (NP) が削除されたものが，ある名詞句のあとに続くと，それら全体で1個の名詞句を構成することを示している．

このような方式で，英語の基本文型と考えられる範囲について約 140 個の AUGCF 規則，および規則中に現われる関数を実現するプログラムを作成した．図3に名詞句および名詞節に関する規則を示す．動詞型については，AS ホーンビーの分類* に従った．これは，動詞に関する統語パターンの詳細な分類がなされており，それが実際の辞書において実践されているためである．また，性や数の一致などをチェックするために，6種類の統語素性および名詞のタイプや従属する前置詞句の種類などをチェックするために 13 種類の意味素性を用いた．

3.2 英語のパージングについて

AUGCF 規則の記述だけによって入力文をパーズすることも可能であるが，より効率的にパーズするためには制御に関する知識を用いることができるようにす

* たとえば AS ホーンビー著／伊藤健三訳注：英語の型と語法，東京オックスフォード大学出版局 (1977) を参照．

る方が有利である．そこで，AUGCF 規則を拡張遷移網 (ATN)[10] の中に埋め込む作業を行った．

(例1)

$$R\,10: S \xrightarrow{syn_{10},\ score_{10},\ sem_{10}} NP \cdot VP$$

という AUGCF 規則は，

(S(T(EXPAND (NP VP)(syn10 score10 sem10))))

というアーク表現に書き換える．このアークは次のことを意味する：「Sが予測されたとき，もし条件部が成立すれば（今の場合，条件部は T(真) であり，常に成立する)，Sというゴールを展開(expand)して，新たにサブゴールとして NP-VP を設定せよ．」

(例2)

$$R\,12: NP \xrightarrow{syn_{12},\ score_{12},\ sem_{12}} NP \cdot PREPP$$

という AUGCF 規則は，

(NP((?CAT PREP) (CONSIF (PREPP) NP (syn12 score12 sem12))))

というアークに書き換える．このアークは次のことを意味する：「NP 節点が構成されたとき，もし現在スキャンしている語の品詞が PREP（前置詞）なら，PREPP (前置詞句) を新たなゴールとして解析し，それが成功すれば新しい NP 節点を構成せよ．」

パーザの制御プログラムは ATNG インタプリタと同様に後戻り制御を行いつつパージングを実行する．

4. トランスファ

トランスファの過程では英語の入力文を解析して得られた EFR を JFR に変換する．主な処理は語彙の置き換えと英語日本語間の表現の「ずれ」を補正するための構造変換であり，統語構造の変換操作は不要である．これらの処理は λ-範疇言語内で行われるため次のような制約条件が課される：

(制約条件1) EFR の任意の表現 α は同じタイプの JFR の表現に変換されなければならない．

また，JFR を EFR から独立に設定する意味で，

(制約条件2) JFR のすべての記号は高々一つの単語または形態素にしか対応しない，

という制約条件を課す．

4.1 トランスファ過程における構造変換規則

本システムでは，トランスファ過程は単語の置き換えが中心であり，構造変換は必要最小限度にとどめている．制約条件2によって細かな構造変換規則がいくつか必要となるが，それを**表4**に示す．

より本質的に構造変換が必要となる場合は，英語で

表4 トランスファ過程における細かな構造変換規則
Table 4 Minor transformational rules in the transfer step.

1. 形容詞⇒＊ADJ＋名詞 (ただし，*ADJ は形容詞形成作用素) (例) logical (formula)∈EFR⇒(*ADJ(論理))(式)∈JFR "logical formula" 「論理式」
2. 形容詞⇒名詞句＋助詞 (例) particular (instruction)∈EFR "particular formula" $\lambda x[(a^*(\text{特定})(\lambda y[((\text{の}(y))(\text{命令}))(x)])]\in$JFR 「特定の命令」
3. 副詞⇒形容動詞＋連用形形成作用素 (♯KRENYO) (例) efficiently∈EFR⇒♯KRENYO(効率的だ)∈JFR "efficiently" 「効率的に」

no, few, little 等を用いたときの否定構文である．たとえば，"no" は通常，「…であるような～はない」，「～は…しない」，「どの～も…しない」と翻訳されるが，"no" 自身に対応する句構造を日本語において見い出すことはできない．このような否定構文の翻訳は次のような λ-入深層構造の変換規則によって記述できる：

$$no \Rightarrow \lambda q[\lambda p[(a^*((\text{のような}(p))(q)))(\text{ない})]] \quad (7)$$

$$few,\ little \Rightarrow \lambda q[\lambda p[(a^*((\text{のような}(p))(q)))(\text{ほとんど(ない)})]] \quad (8)$$

$$nothing \Rightarrow \lambda p[(a^*((\text{のような}(p))(\text{もの})))(\text{ない})] \quad (9)$$

この規則を用いると，たとえば，

(no (operand)) (*en(need)), "no operand is needed."

は，

$$((\lambda q[\lambda p[(a^*((\text{のような}(p))(q)))(\text{ない})]])(\text{オペランド}))(\text{必要とされる})$$
$$=(a^*((\text{のような}(\text{必要とされる}))(\text{オペランド})))(\text{ない}),$$

「必要とされる(ような)オペランドはない」

とトランスファされ，また，

(the(command))
$(\lambda x[(\text{no(operand)})(\lambda y[\text{need}(x, y)])])$,
"the command needs no operand."

は，

$$(\text{その(コマンド)})(\lambda x[((\lambda q[\lambda p[(a^*((\text{のような}(p))(q)))(\text{ない})]])(\text{オペランド}))(\lambda y[\text{必要とする}(x, y)])])$$
$$=(\text{その(コマンド)})(\lambda x[(a^*((\text{のような}(\lambda y[\text{必要とする}(x, y)]))(\text{オペランド})))(\text{ない})]),$$

「そのコマンドが必要とする(ような)オペランドはない．」

とトランスファされる．

4.2 トランスファの変換手続き

トランスファ過程の変換手続きは，与えられたEFRを外側から逐次評価してゆく．EFRが原子記号（アトム）であればトランスファ用の辞書をひいて，JFRの表現に置き換える．EFRがアトムでなければ，それを関数部のEFRと引数部に分解し，関数部のEFRのタイプに対応して用意された変換手続きを呼び出す．関数部のEFRがアトムであればトランスファ辞書をひいて対応するJFRの表現に変換する．関数部がアトムでなければ，上の処理を関数部がJFRにトランスファされるまで行う．関数部がJFRに変換されたら次に，各引数についてトランスファ処理を行い，その結果を関数部の結果と合成する．

トランスファ用の辞書はEFRのアトムを見出しとし，そのアトムのタイプとJFRへの変換規則が書かれる．

（例） 英語の形容詞に対応するEFRのアトム“automatic”のトランスファ辞書記述は，

```
((: SCAT・((0 1)(0 1)))
 (: TRANJ (KEIYO (LEX. JIDOUTEKI))))
```

であり，これはautomatic∈EFRがjidoutekiとJFRに置き換えられることを示す．一方，particular∈EFRのトランスファ辞書記述は，

```
((: SCAT. ((0 1)(0 1)))
 (: TRANJ (MEISHI (LEX. TOKUTEI))
          (JO (LEX. NO))))
```

となり，トランスファ手続きはparticular∈EFRをこの辞書によって，「特定の」に対応するタイプ〈〈0, 1〉, 〈0, 1〉〉のJFR表現，

$\lambda p_{\langle 0,1\rangle}[\lambda x_1[(a^*(\text{特定}))(\lambda y_1[((\text{の}(y))(p))(x)])]]$

に変換する．

5. 日本語の生成

日本語文の生成過程では，JFRから日本語の統語構造を生成し，次にこの終端記号列をとり出して形態素の合成（たとえば，S+Aから「さ」を合成する処理）を行って最終的な漢字かな混り文を得る．

5.1 日本語の統語構造の生成

トランスファ過程と同様に，与えられたJFRを逐次分解しながら統語構造を生成してゆく．生成手続きは，各文法範疇ごとの生成を受け持つ個別手続きと，個別手続きの呼び出しや統語構造の構成など，生成過程の制御を行うモニタ手続きから成る．これらの処理には，助詞・助動詞の選択，活用語尾の決定，付属語の選択などに関する情報を含んだメッセージを用いる．

モニタ手続きは引数として，生成すべき統語構造の文法範疇，JFRの表現，メッセージ，変数束縛リストが与えられると，第1引数に対応する個別生成手続きを呼び出し，他の三つの引数をその個別生成手続きの引数としてひきわたす．個別生成手続きは与えられたJFRの表現を吟味し，必要な情報を与えられたメッセージあるいは変数束縛リストから取り出し，必要であればモニタ手続きを再帰的に呼び出して，目的の統語構造を得る．たとえば，文の範疇の生成を受け持つ手続き，SENTENCEは与えられたJFR表現 $\alpha(\beta_1, \cdots, \beta_n)$ をおよそ次のように処理する：

```
if type(α)=⟨0, ⟨0, 1⟩⟩ (名詞句)
    then call SENTENCE (β(x))x: call NP(α)
else if then(α)=⟨0, 0⟩ (副詞句)
    then call SENTENCE (β)ADV: call ADV(α)
else if type(α)=⟨0, 1^n⟩ (動詞)
    then call VERB(α)
    ...
```

また，**図4**に日本語統語構造生成のサンプル例を示す．

この過程で用いられる生成用辞書はJFRのアトムを見出しとして構成され，そのアトムのλ-範疇言語におけるタイプ，格構造パターン，統語構造パターン，語尾の音韻情報，さらにはJIS漢字コードが記述されている．

個別の生成手続きから下位の統語構造が返されると，モニタ手続きは現在の統語構造をもう一段成長させ，その結果を次のような形式で返す：

```
(文法範疇 (DEC・⟨下位の統語構造⟩)
          (MSG・⟨下位からのメッセージ⟩)).
```

5.2 漢字かな混り文の出力

上の過程で生成された統語構造の終端記号をとり出して一次元に並べる．これは，JIS漢字コードとローマ字の混在した系列であり，隣り合うローマ字をカナ文字に対応するJISコードに変換することにより，漢字かな混り文（完全なJISコード列）を得る．

6. 実験と検討

以上述べてきたシステムを研究室のミニコン上のLISPシステム（LISP 1.7）[2]上に作成した．約35Kセル（約5,000行）がプログラム領域に必要とされた．辞書はすべてLISP 1.7の直接アクセスファイル

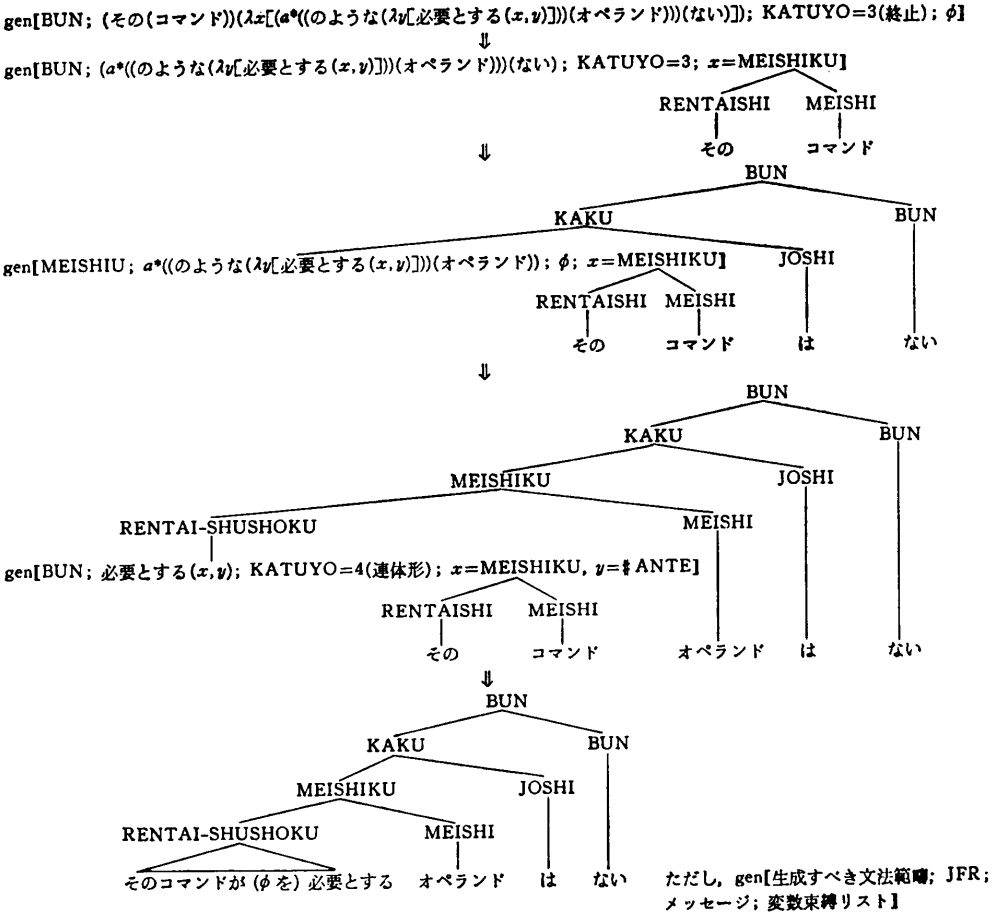

図 4 日本語統語構造生成の一例

Fig. 4 A sample generation of Japanese syntactic structure from a JFR.

上に置かれ，必要な単語のみ LISP のデータ領域にもってこられる．

実験の方針としては，まず文法書に基づいて，英語の基本的な文型を翻訳するための規則を，英文解析，トランスファ，日本語生成の各ステップについて作成した．それによって基本文型の翻訳実験を行った．(そのうちの約 100 個が文献 5) に示されている．) その結果，まとまった範囲の言語現象について翻訳システムの能力が確認できたので，現在，第 2 段階として現実のテキスト（計算機マニュアル）を対象に翻訳実験を行い，規則の修正や問題点の整理などを行っている．**図 5** にマイコンマニュアル (Zilog Z-80 Technical Manual) の一部の翻訳結果を示す．ただし，現在のシステムは基本文型を対象としたものであるため，原文を一部修正して実験を行うことを余儀なくされた．なお，このために約 500 の辞書項目を登録した．この実験において，人間による補助は英文解析時のあいまい性の解消だけであった．すなわち，システムが入力文に対して可能な論理式を次々に人間（専門家）に提示し，人間がそのうちの一つを選択する．または，実際にそれらの候補に対して訳文を生成して，人間がそのうちの一つを選択することも可能である．

一般に，機械翻訳では種々のあいまい性が問題になる．困難な問題は，ある語の品詞が決定されてもなお，意味のあいまい性が残り，訳語が一意に決定できないことがありうることである (語の多義性の問題[9])．我々のシステムでは，ある語が複数の語義をもつとき，それぞれ別の語であるとみなして英文解析の段階で処理しているので，たとえば同一品詞で 2 通りの語義をもつ語が 3 か所で出現すると，結果としてそれについて $2^3=8$ 通りの異なる論理式 (EFR) が出力されることになってしまう．(ただし，このうちのいくつかは意味的チェックで消滅するかもしれない．) モンテギュー文法の枠組でもこのようなやり方がとられてい

The assembly language provides a means for writing a program without being concerned with actual memory addresses or machine instruction formats. It allows the use of symbolic addresses to identify memory locations and mnemonic codes to represent the instructions. Labels 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. Operands following each instruction represent storage locations, registers, or constant values. The assembly language includes assembler directives that supplement the machine instruction. A pseudo-op is a statement which is not translated into a machine instruction. A pseudo-op is a statement which is interpreted as a directive that controls the assembly process.

(a) 入力テキスト

アセンブリ言語は実際のメモリアドレスないしは機械命令形式を知ることなしにプログラムを書くための方法を与える．それはメモリロケーションを識別するための記号アドレスとその命令を表現するためのニモニックコードの利用を許す．ラベルはそのステップを後続の命令のなかの利用のためのエントリポイントとして識別するためにソースプログラムのなかの特定の命令ステップに代入されることができる．各命令に続いているオペランドはストレッジロケーションないしはレジスタないしは定数を表現する．アセンブリ言語はその機械命令を補うアセンブラ命令を含む．擬似命令は機械命令に翻訳されないステートメントである．擬似命令はアセンブリ過程を制御する命令と理解されるステートメントである．

(b) 生成された日本語の訳文

図 5 試作システムによる翻訳例

Fig. 5 An example of translation of a real text: in Z80-Assembly Language Programming Manual, Zilog, 1977 (with some modifications).

るが，実際に広い範囲のテキストの翻訳を行うときは効率的に問題である．英文解析の段階で文脈処理まで行っても語の多義性の問題を完全に解決できるとは期待できないので，いくつかの語義をひとまとめにして，あいまい性を含んだ一つの語彙として処理し，解析の途中でできる限りあいまい性を解消し，残りについてはトランスファ段階で単語おきかえを行うとき，適切な人間の補助を受けるようにするのが実際的であると思われる．この問題は今後残された主な課題である．

7. むすび

本論文ではモンテギュー文法に基づいて英文和訳を行うシステムについて報告した．内包論理のレベルの，深い中間表現を用いたので，翻訳能力はかなり強力で，トランスファ過程がほとんど単語の置き換えだけでよく，実際の計算機マニュアル程度のテキストであれば，原文に少しの修正を加えるだけで後処理なしでも人間に十分読める程度の訳文が得られることがわかった．また，モンテギュー文法の枠組はシステム構成の単純化に有効であった．

最後に，討論していただいた研究室の諸氏に感謝いたします．また日頃有益なご助言をいただく本学電気工学第二教室の長尾真教授，辻井潤一助教授に感謝いたします．本研究の一部は文部省科学研究費による．

参考文献

1) Cresswell, M. J.: Logics and Languages, Methuen & Co. Ltd. (1973), (石本，池谷（訳）：言語と論理，紀伊國屋書店 (1978)).
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., Wall, R. and Peters, JR.: Introduction to Montague Semantics, Reidel (1981).
4) Hutchins, W.: Progress in Documentation, Machine Translation and Machine aided Translation, J. Documentation, Vol. 34, No. 2, pp. 119-159 (1978).
5) 清野正樹：英日機械翻訳システムの作成，京都大学修士論文 (1981).
6) Montague, R.: Universal Grammar, in Thomason (ed.): Formal Philosophy, Yale University, pp. 222-246 (1974).
7) Montague, R.: Proper Treatment of Quantification in Ordinary English, in Thomason (ed.): Formal Philosophy, Yale University, pp. 247-270 (1974).
8) 長尾　真：機械翻訳，情報処理，Vol. 20, No. 10, pp. 896-902 (1979).
9) Wilks, Y.: An Artificial Intelligence Approach to Machine Translation, in Schank and Colby (eds.): Computer Models of Thought and Language, Freeman and Company, pp. 114-151 (1973).
10) Wood, W.: Transition Network Grammers for Natural Language Analysis, Comm. ACM, Vol. 13 (1970).

(昭和56年4月13日受付)
(昭和56年9月7日採録)