自然言語処理 47－9
（1985. 1. 25）

Muプロジェクトにおける翻訳実験支援環境

片桐恭弘（武蔵野通研）、中村順一（京大・工）
辻井潤一（京大・工）、長尾真（京大・工）

1. はじめに

高品質の機械翻訳を実現するためには、辞書・文法規則等大量の情報を必要とする。また細かな言語現象に対応し、良質の訳文を得るためにはシステムを最初から固定して考えることができず、実験を繰返しながら拡張・変更を頻繁に行い、開発を進めることになる。Muプロジェクト［1］では，科学技術文献抄録を対象とし，トランスファ方式に基づいて，日英・英日機械翻訳システムを開発している。実験対象文の増加とともに段階的にシステムは拡張されてきた。このような状況では、ソフトウェアや文法規則だけでなく、

1．辞書システムの管理
　a．全体の管理
　b．個々の内容の管理
2．翻訳の中間結果の管理

を支援するソフトウェアが必要になる。その際特に、管理すべきデータの種類が豊富でかつ量が膨大であること、頻繁な拡張・変更にすばやく対応できなければならないこと、プログラミングに習熟していない人でも作業を行えること、の三点に留意しなければならない。本稿ではMuプロジェクトにおける辞書システムの管理支援ソフトウェアを中心に述べる。

2. 辞書操作の管理

辞書システムを運用する場合、辞書データに対する各種の操作、たとえば、辞書内容の検索・編集、データ形式の検査・変換等を行う必要がある。これらの操作は相互に関係しあうので、翻訳システムが大規模なものになるにつれ、人間だけでは扱いきれなくなり、ソフトウェアによる支援が必要になる。ここでは、辞書データに対する各種の操作を支援するためのツールについて述べる。

2．1． 辞書の運用方式

Muプロジェクトにおける辞書システムは、［2］で述べたように、辞書データベースを特定のシステムとは独立なものとして設計、作成し、その辞書データベースから特定のシステムで使用する辞書を生成する方法を用いている。この方法により、元となる辞書の作成が容易になるとともに、同じ辞書を文解析、文生成の両システムで使用できるといったように辞書データベースそのものの汎用性を高めている。

辞書は、まず、辞書記述者がフォーマット用紙に辞書作成作業手順書［3］に従って個々の単語の情報を記入することにより作成される。フォーマット用紙の記述は、MTフォーマット辞書という汎用の辞書データベースに整理され、これをもとにして日英・英日の翻訳システムで実際に使用される辞書が作られる［4］。最終的な処理用の辞書には、文法規則により解釈を受ける受動的な木構造形式の辞書（木構造辞書と呼ぶ）と辞書そのものが単語固有の文法規則（辞書規則）として動作する能動的なルール形式の辞書（ルール辞書と呼ぶ、これは、文法記述用言語 GRADE ［5］で表現されている）がある。この形式変換の流れを図1に示す。

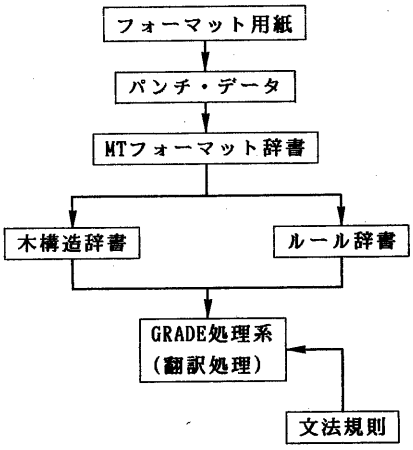

図1　辞書の形式変換の流れ

2．2． 辞書操作の管理用ツール

Muプロジェクトで使用している辞書には、言語別、品詞別、処理別、データセット（ファイル）の形式別、の観点から見て、以下の4種類に区分できる。

１．日本語辞書、日英変換辞書、英日変換辞書、英語辞書。
２．名詞辞書、動詞辞書、形容詞辞書、等。
３．解析処理用辞書、変換処理用辞書、生成処理用辞書。
４．エディット用辞書（順アクセス・ファイル）、辞書引用辞書（ダイレクト・アクセス・ファイル）

この４つを組合わせた辞書、たとえば、エディット用日本語形容詞辞書や辞書引用英語生成処理用動詞辞書等が存在する。このため、辞書の種類が多くなっている。また、先に述べたように３の処理用辞書は、１のMTフォーマット辞書から形式変換を行なって生成されるので、互に独立ではない。このため、辞書に関する情報を一ヵ所に集中して記述しておき、各種の操作を行うプログラム群がこの情報を参照するようにすれば、辞書の管理が容易になる。以上の点から、具体的なデータセット名や形式変換の方法等辞書システム全体を管理し、各種の操作を行うためのソフトウェア・ツールを使用している。

辞書管理用ソフトウェア・ツールは、

１．辞書データセットの管理

２．辞書操作用プログラムの管理

を行う機能を備えており、各種の辞書の定義とその定義を解釈し辞書形式の変換等、辞書を操作するプログラムから構成されている。

それぞれの辞書の定義は、辞書の種類を表す名前、及び、その辞書に関する情報（データセット名等）を記述する属性名と属性値の組の形で表現される。辞書の上位下位の関係、たとえば、英語辞書と英語動詞辞書の関係を自然に表現し、更に、辞書の定義の管理を容易にするため、定義には階層性を持たせることができる。また、図２に示すように、同一のMTフォーマット辞書から、解析と生成の２つの処理用辞書を生成する必要があるので、個々の処理用辞書の定義には、MTフォーマット辞書と解析または生成用辞書との両方の情報が必要になる。そこで、オブジェクト指向型プログラミングにおける多重継承の考え方を用いた定義を行なっている。すなわち、各種の辞書の上位下位の関係をネットワークで表現し、必要な情報は、このネットワークを探索することにより得られるようにしてある。

英語の動詞の辞書の定義の階層関係を図３に、その定義の一部を図４に示す。辞書の定義は、

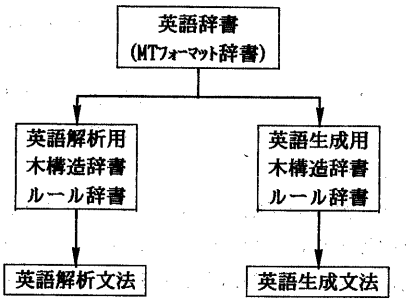

図２　処理用辞書の生成

(DEFINE-DICTIONARY <辞書名> [(EDITTABLE)]
　　<属性名> = <属性値>
　　...)

の形式で行なう。属性名の内で SUPER は、辞書の上位下位の関係を指定する特別な属性名である。図３[e]の例で、

　　SUPER = (E-GENERATION-DICTIONARY E-V)

は、この辞書の上位の辞書名が E-GENERATION-DICTIONARY と E-V であることを指定している。その他の属性名は、すべて、辞書の情報を記述するためのものであり、形式変換実行プログラムや辞書エディタ等が参照している。なお、定義中の (EDITTABLE) は、その辞書が辞書エディタによる編集対象になることを示している。

翻訳システムで使用している辞書の定義は、汎用の辞書であるMTフォーマット辞書の定義と形式の変換の方法を記述する形式変換辞書（木構造辞書とルール辞書）の定義との２つに分離して行なわれている。図３の例では、 DICTIONARY [a] や ENGLISH-DICTIONARY [b]、 E-V [c] がMTフォーマット辞書の定義であり、 E-GENERATION-DICTIONARY [d] と E-V-GEN [e] が形式変換辞書の定義である。

たとえば、英語の動詞のMTフォーマット辞書（E-V、English-Verb）は、英語のMTフォーマット辞書（ENGLISH-DICTIONARY）の下位の辞書として定義されている。そこで、英語のMTフォーマット辞書全体が保存されているVSAMデータセット（ダイレクト・アクセス・ファイル）名は、 ENGLISH-DICTIONARY の定義の部分に記述されており、英語の動詞辞書固有の情報である、品詞名（CAT-VALUE）や、MTフォーマットの形式のパターンの定義（PATTERN、３節参照）、辞書内容を編集した場合にその変更情報を保存するデータセットの名前（HISTORY-FILE）

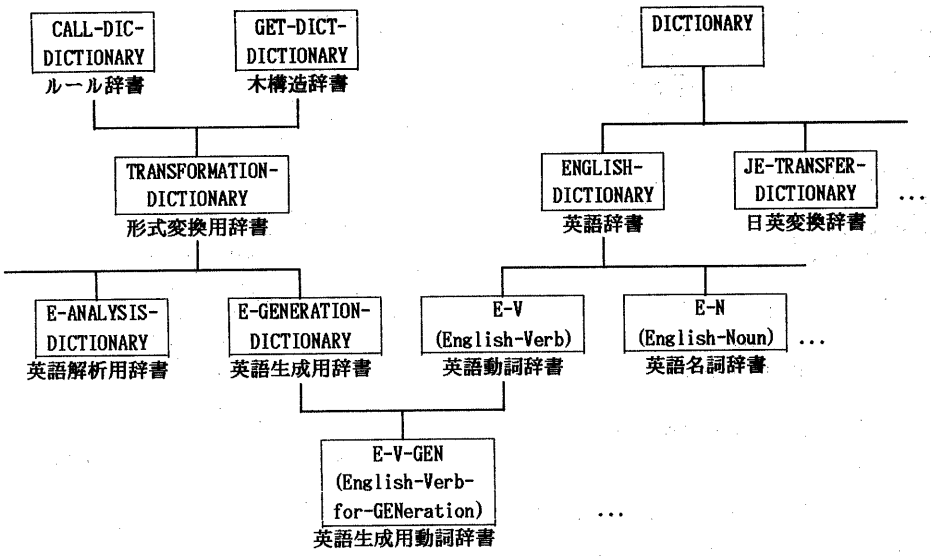

図3　辞書定義の階層関係

```
(DEFINE-DICTIONARY DICTIONARY
   SUB-DICTIONARIES     =
   (ANALYSIS JE-TRANSFER-DICTIONARY
    E-GENERATION-DICTIONARY E-ANALYSIS-DICTIONARY)
...)                                    [a] MTフォーマット辞書

(DEFINE-DICTIONARY ENGLISH-DICTIONARY
   SUPER                = DICTIONARY
   VSAM                 = "'ABF2801.GEN.MTF.VSAM'"
   KEY-PATH             = (E_LEX)
   CAT-PATH             = (E_CAT)
...)                                    [b] 英語MTフォーマット辞書

(DEFINE-DICTIONARY E-V (EDITTABLE)
   SUPER                = ENGLISH-DICTIONARY
   VSAM-AREA-NUMBER     = 153
   CAT-VALUE            = V
   HISTORY-FILE         = "'ABF2801.GEN.V.MTFORM'"
   PATTERN = "'APH1667.DICTSYS.LISP(DEFMTF)'"
...)                                    [c] 英語動詞MTフォーマット辞書
```

```
(DEFINE-DICTIONARY E-GENERATION-DICTIONARY
   SUPER                = TRANSFORMATION-DICTIONARY
   SUB-DICTIONARIES     =
      (E-N-GEN E-V-GEN E-ADJ-GEN E-ADV-GEN
       E-DET-GEN E-PRON-GEN E-CONJ-GEN E-CARD-GEN
       E-ORD-GEN E-ATR-GEN E-AUX-GEN)
   GET-DICT-VSAM        = "'ABF2801.TGDIC.VSAM'"
   CALL-DIC-VSAM        = "'ABF2801.TGDIC.VSAM'"
   CALL-DIC-MACRO-DEF-PS-FILE-NAME =
                        "'ADH3759.TRAGEN.MACRO'"
...)                                    [d] 英語生成辞書

(DEFINE-DICTIONARY E-V-GEN
   SUPER                = (E-GENERATION-DICTIONARY
                           E-N)
   CALL-DIC-MACRO-SOURCE-FUNCTION =
                        WRITE-GENV-MACRO
...)                                    [e] 英語動詞生成辞書
```

図4　辞書の定義（一部）

は、 E-V の定義の部分に記述されている。

これに対して、英語の生成処理用の辞書である、木構造辞書（GET-DICT辞書）とルール辞書（CALL-DIC辞書）とに関する情報は、英語の生成辞書（E-GENERATION-DICTIONARY）とその下位の辞書定義である英語動詞生成辞書（E-V-GEN、English-Verb-for-GENeration）との部分に定義されている。たとえば、木構造辞書とルール辞書のVSAMデータセット名（GET-DICT-VSAM、CALL-DIC-VSAM）と、マクロ定義（CALL-DIC-MACRO-DEF-PS-FILE-NAME、すべての品詞の定義が同じデータセットに書かれている）は、 E-GENERATION-DICTIONARY の定義の部分に記述されており、英語の動詞辞書のマクロ表層生成プログラム名（CALL-DIC-MACRO-SOURCE-FUNCTIUON）は E-V-GEN の部分に定義されている。

図3のネットワークを下から上へ、深さ優先で探索することにより、辞書データセット名等、辞書形式の変換に必要な情報を取り出すことができる。たとえば、英語の動詞のMTフォーマット辞書から、英語生成用ルール辞書を作成する場合は、図5に示すように、入力のMTフォーマット辞書VSAMデータセット名、ルール辞書生成用関数名、ルール辞書

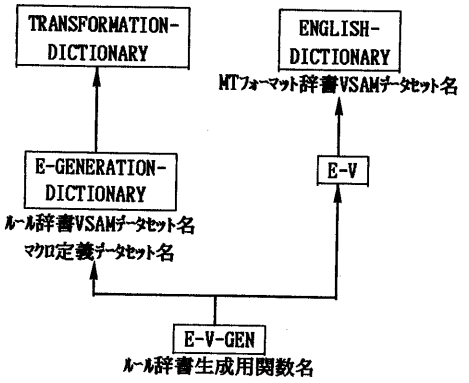

図５　必要な情報の取り出し方

展開用マクロ定義が保存されているデータセットの名前、ルール辞書データセット名をそれぞれの辞書定義から取り出すことができる。

メニュー形式でこのような形式変換を行うためのツールも用意されている。このツールは、辞書定義のネットワークから取り出した情報をディフォールト値として利用者に示し、形式変換を実行する。

３．　MTフォーマット辞書管理システム

３．１　MTフォーマット辞書管理システムの概要

修正・追加等、辞書内容の管理の対象となるのは単語の言語学的性質を記述したMTフォーマット形式のデータである。MTフォーマット辞書管理のためにMTフォーマットデータの形式整合性のチェック、検索、変換を行うためのツールが用意されている。MTフォーマットは図６に示すように一項目ごとに一つずつLispのＳ式となっており、基本的に属性・属性値の組み合わせから成っている。MTフォーマットに関する情報は辞書定義(前節参照)に一括して記述されている。これには図７に示すようなMTフォーマットの形式を記述したパターンが含まれる。情報を一箇所に集中することによって見易さが向上し、変更も容易になる。また、MTフォーマットの形式をパターンとして記述しておくことによって、辞書データの形式と内容とが分離され、形式上の誤りの検出や形式の変更等、辞書の形式の側面を扱う部分の自動化が可能となるため、辞書管理者は辞書の内容に専念できる。

MTフォーマット辞書管理システムの概要を図８に示す[6]。個々のMTフォーマットデータとその

```
((SEQ 1530)
 (J_LEX 考慮)
 (J_CAT 名詞)
 (USAGE
   ((E_LEX consideration)
    (E_UID 1)
    (E_CAT N)
    (CORRESPONDENCE
      ((J_LEX 入れる)
       (E_LEX
         take into consideration)
       (E_CAT V)
       (E_UID 1)
       (CASE_FRAME
          ((J_SURFACE_CASE が)
           (J_DEEP_CASE 主体)
           (E_SURFACE_CASE SUBJ)
           (E_DEEP_CASE AGT))
          ((J_SURFACE_CASE を)
           (J_DEEP_CASE 対象)
           (E_SURFACE_CASE OBJ1)
           (E_DEEP_CASE OBJ))
          ((J_SURFACE_CASE に)
           (J_DEEP_CASE 場所一終点)
           (J_POS_FLAG DEL))))))))
```

図６．MTフォーマットデータの例

```
((SEQ ?_FIXED-NUM:SEQ)
 (J_LEX ?_JEF-ATOM:J_LEX)
 (J_CAT 名詞)
 (USAGE
  (!ORDER-FREE
   (E_LEX ?_JEF-ATOM:E_LEX)
   (E_CAT ?_CAT:E_CAT)
   (E_UID ?_FIXED-NUM:E_UID)
     ...
   !OPTIONAL
   (CORRESPONDENCE
     !REPEAT <CORRESPONDENCE>
     ((J_LEX ?_JEF-ATOM:CORRESPONDENCE_J_LEX)
      (E_LEX ?_JEF-ATOM:CORRESPONDENCE_E_LEX)
      (E_CAT ?_CAT:CORRESPONDENCE_E_CAT)
      (E_UID ?_FIXED-NUM:CORRESPONDENCE_E_UID)
      (CASE_FRAME
        !REPEAT <CASE_FRAME>
        (!OPTIONAL
         (J_SURFACE_CASE ???_JEF-ATOM:J_SURFACE_CASE)
         !OPTIONAL
         (J_DEEP_CASE ?_JDCASE:J_DEEP_CASE)
         !OPTIONAL
         (J_POS_FLAG ?_EBCDIC-ATOM:J_POS_FLAG)
         !OPTIONAL
         (E_SURFACE_CASE ?_ESCEBC:E_SURFACE_CASE)
         !OPTIONAL
         (E_DEEP_CASE ?_EDCEBC:E_DEEP_CASE)) ))) ))
```

図７．MTフォーマット定義パターンの例

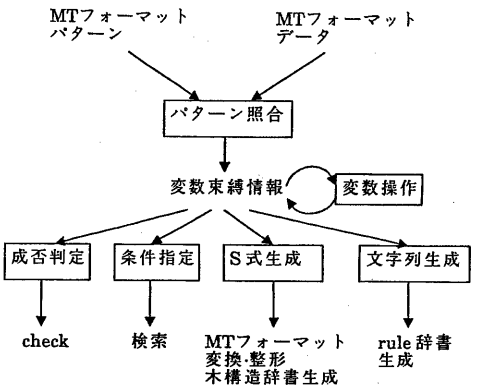

図８．MTフォーマット辞書管理システムの概要

型のMTフォーマットのパターンとの照合が基本部分である。パターン照合によって得られる変数束縛情報が個々の辞書項目の内容を示すものであり、それをもとにMTフォーマットのチェック、検索、変換が行われる。

３．２　パターンの記述とパターン照合

MTフォーマットデータの形式を表すパターンは図７に示したように基本的にはMTフォーマットのＳ式の構造をなぞって作られる。Ｓ式の構造上の自由度を表すために三種類のキーワードが用意されており、また情報内容の自由度を表すためにタイプ付き変数を用いることができる。三種のキーワードの機能を以下に記す。

!optional ： 次の要素は有っても無くてもよい。従って (X !optional Y Z) は (X Y Z), (X Z) のいずれとも照合する。

!order-free ： それ以降の要素は順序自由で出現してよい。従って (X !order-free Y Z W) は (X Y Z W), (X Y W Z), (X Z Y W), (X Z W Y) ... と照合する。

!repeat ： 次の要素が任意回(0回を含む)繰返される。従って (X Y !repeat Z) は (X Y), (X Y Z), (X Y Z Z) ... と照合する。但し、繰返し要素は必ずリストの最後でなければならない。また、繰返し構造を後で参照するためには繰返し構造に名前を付ける必要があり、その場合には (X Y !repeat <名前> Z) のように記述する。

いずれの場合でも X,Y,Z 等はそれ自身パターンであり当然中にキーワードを含む場合も有りうる。

変数には一個の要素と対応するものと、任意個の要素と対応するものとの二種類あり、それぞれ次の形に書く。

?_変数の型:変数名

???_変数の型:変数名

変数の型としては、英数字アトム、英数字文字列、日本語アトム、日本語文字列、数の５種類が予め用意されているが、格名称や意味マーカのように予め値の範囲の定まっているものはそれらを列挙して新たな変数型を作ることも可能である。これは入力誤りの検出に有効である。

パターン照合プログラムはパターンと MT フォーマットデータとを対応付け、照合に成功した時には変数の束縛情報を返す。照合に失敗した場合には失敗箇所に関する情報が出力されるが、照合の途中で失敗が分った場合でも、そこで終わらずに最後まで照合を行うため、すべての失敗箇所を一度に得ることができる。また、整備されていないデータを扱うために、失敗が起こった場合でも可能な範囲で変数束縛情報を返すモード、変数の型のチェックを行わないモード等が用意されている。後者はコード変換を行う場合等に用いられる。

３．３　辞書データの形式のチェック

辞書データは現在のところ手作業で作られる。従って入力誤りや作業者による記入のばらつきが含まれている。また、段階的に辞書を拡張していくため、各段階で仕様の細部が変更されることもある。このように辞書データが大量になるにつれて、それらの正しさ・整合性を保持するのが次第に困難になってくる。

前述のパターン照合を用いれば、辞書データの形式上の正しさをチェックすることが可能である。MTフォーマット辞書定義に記述されているパターンとMTフォーマットデータとの照合が成功すればそのデータは形式上は正しいこととなる。チェックを有効に行うためにはパターンをなるべく細部まで厳密に記述しておき、変数の型も細かく定義しておいた方が良い。

MTフォーマットデータのチェックには一括チェックと修正後チェックとの二通りがある。一括チェックではMTフォーマットの種類を指定し、その種類のデータすべてに対して一度でチェックを行う。修正後チェックでは辞書データを修正するごとにそれらを辞書に登録する前にチェックを行い、正しいもののみを登録する。前者は単独のツールとなっており、端末から必要な情報を入力しながら実行する。後者は、辞書修正用エディタに組み込まれている。図９にチェックの結果の一例を示す。構造上の誤り、データの欠落、綴り誤り、型の誤り等を検出することができる。

３．４　辞書データの検索

MTフォーマットデータに修正・変更を加える場合、すべてのデータを一度に変更するよりも、特定の条件を満たすデータだけに変更を加えたい場合

```
言及
  *OK*

限界
  ((CASE_FRAME ((J_SURFACE_CASE を) ...) -- UNKNOWN

差異
  (E_UID ?_FIXED-NUM:E_UID) -- MISSING
  ((UID 1)) -- UNKNOWN
専門家
  名 -- MUST BE 名詞
  範囲 -- MUST BE OF TYPE JDCASE
```

図９．チェック結果の例

の方が多い。従ってデータの量が多くなるとある条件を満たすデータのみを抽出する検索ツールが必要となる。一旦MTフォーマットデータとパターンとの照合に成功して変数束縛の情報が得られれば、変数の値に関する条件によってMTフォーマットデータの検索が可能となる。

変数の値はパターン照合プログラムの結果として得られる変数束縛情報に含まれているが、それを参照するために、prefix @を用いることにした。@変数名によって変数名に対応する値を参照することができる。図１０は検索条件の例である。図に示すようにパターン中の変数を Lispの変数とほぼ同様に見なして検索条件を記述することが可能である。図１０(c)は繰返し構造中の変数を参照する記法の一つで、繰返し構造中の変数@E_NUMBER_TYPEのうちR1と等しい(EQ)ものがどこかに一つでもあれば良いことを示している。

３．５　辞書データの変換

辞書データの変換は、変換の目的という見地から見ると、MTフォーマットデータの整合性の保持や、仕様変更への対処のために行うMTフォーマットデータ内の変換と、MTフォーマットデータからGRADEシステムで用いる形式への変換との二通りに区分できる。前者には、コード変換、値の修正、カッコ

```
(a)  (MEMQ @E_CAT
           '(ADVERB ADJECTIVE CONJUNCTION))

(b)  (AND @PLURAL
          (EQ 'REGULAR @INFLECTION))

(c)  (CONDITION SOME
                ((X @E_NUMBER_TYPE))
                (EQ 'R1 X))
```

図１０．検索条件の記述

のレベルの変更等が含まれる。また、辞書情報を拡張するために、予めMTフォーマットを新しい属性とdummyの属性値とを追加した形に変換しておき、後の作業の負担を減らす等の利用も考えられる。後者は木構造辞書への変換、および辞書規則への変換の二つである。辞書データの変換はまた、変換によって得られるデータの形式という見地からLispのＳ式への変換と文字列への変換との二種類に区分することもできる。前述のうちでは辞書規則への変換のみが文字列への変換であり、その他はすべてＳ式内部での変換である。

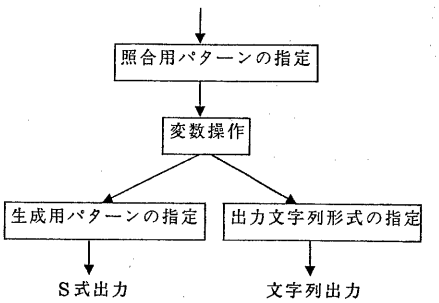

図１１．MTフォーマット変換操作の手順

図１１に変換手順を示す。まず照合用パターンを指定し、パターン照合によって変数の束縛情報を得る。照合用パターンはMTフォーマット辞書定義に記述されているものがデフォルトとなり、必要に応じてそれを修正して用いる。次に変数の値に対する操作を行う。これはLispの一般化代入文を用いて

```
(:=  @変数名  新しい値)
```

のように記述する[７]。但し、繰返し構造内の変数を操作するには繰返し構造の名前を指定して

```
(WITH-REPEAT  <名前>
        (:=  @変数名1  新しい値1)
        (:=  @変数名2  新しい値2)
        . . . )
```

のように記述するものとした。コード変換や、値の修正はこの変数の値の操作によって実現される。

次にＳ式を変換出力とする場合には、生成用パターンを指定する。これは照合用パターンの指定の時とほぼ同様に行なわれる。変更された変数束縛情報と生成用パターンとからシステムは新たに変換されたＳ式を作り出す。照合用パターンと生成用パターンとの差によって構造上の変更が行なわれる。

図１２はMTフォーマット変換手続きの一例である。データに応じたコード変換及び通し番号振りを行っている。

文字列を変換出力とする場合には、MT-LINE及びMT-PHRASEという文字列出力用の関数を用いて出力文字列の形を指定する。MT-LINEは

(MT-LINE <文字列パターン> <変数> ...)

という形で、一行分の文字列を文字列パターンに変数の値を埋めこんで作り出す。MT-PHRASEはMT-LINEとほぼ同様だが、改行がされない。図１３に文字列への変換の一例を示す。入力となるMTフォーマットは図６に示すものである。図１３(a)は出力文字列形式の指定、(b)は出力される文字列である。これは、辞書規則生成用マクロであり、マクロ展開によってさらに辞書規則に展開される[2]。

```
(DEFUN MTF-CONVERT ()
  (:= @E_CAT (ATOM_CONV @E_CAT))
  (WITH-REPEAT '<USAGE>
     (:= @SEQ (:= *SEQ* (1+ *SEQ*)))
     (:= @E_SUBCAT (E_SUBCAT_CONV @E_SUBCAT))
     (:= @E_UID (FIXNUM_CONV @E_UID))
     ... ))
```

図１２．MTフォーマット変換プログラムの一例

４．　その他の実験支援プログラム

翻訳システムの質を向上するためには、多人数で、多量の文の翻訳結果を詳細に検討しながら、システムを改良していかなければならない。この場合、各種の実験支援プログラムが必要になる。そこで、処理状態のトレーサなどのディバッガ［８，９］以外にも、以下に述べる実験支援プログラムを作成し、使用している。

１．翻訳対象文の管理、検索

同一の文であっても、解析、変換、生成のそれぞれの立場で、色々な面から文を詳細に検討しなければならない。そこで、翻訳実験を行いたい文を簡単に選択し実験対象とすることができるようになっている。また、一群の文章の中から特定の言語現象が現われている文、たとえば、特定の接続詞を含む文、だけを文字列レベルの検索により取り出し、文法記述者が検討することも可能である。

```
(DEFUN MACRO-OUT ()
  (MT-LINE "TRANSN(/C,D,,,,,,,,,," @J_LEX)
  (MT-PHRASE "RRINTN(/C" @J_LEX)
  (LET ((CNT 0))
    (WITH-REPEAT '<CORRESPONDENCE>
      (MT-PHRASE ",/C" (:= CNT (1+ CNT)))))
  (MT-LINE ")")
  (LET ((CNT 0))
    (WITH-REPEAT '<CORRESPONDENCE>
      (MT-LINE ",")
      (MT-LINE "ITEMD(/C,/C," @CORRESPONDENCE_J_LEX
                              (:= CNT (1+ CNT)) )
      (MT-PHRASE "JCONDN(/C/C" @CORRESPONDENCE_E_CAT
                               @CORRESPONDENCE_E_UID)
      (WITH-REPEAT '<CASE_FRAME>
        (MT-PHRASE ",/C,/C" @J_DEEP_CASE @J_POS_FLAG))
      (MT-LINE "),")
           ...

    (MT-PHRASE "))") ))
  (MT-LINE ")") )
```

(a) 出力文字列形式の指定

```
TRANSN(考慮,D,,,,,,,,,,
RRINTN(考慮,1),
ITEMD(入れる,1,
JCONDN(V1,主体,,対象,,場所一終点,DEL),
ECONDN(take into consideration,
       V,1,SUBJ,AGT,,,OBJ1,OBJ,,,,,,)))
```

(b)　出力される文字列

図１３．文字列生成の一例

２．中間結果の管理

変換、生成の処理は、それぞれ前の処理結果（解析結果、変換結果）に影響されることが多い。このため、解析、変換、生成の各文法、辞書を独立に改良すると、その改良が互いに影響しあうので、それぞれの部分の検討が困難になる。そこで、各処理の中間結果を保存し、それぞれの文法記述者は、独立に必要な入力データを選択し、実験に使用することができるようになっている。また、それぞれの中間結果には処理をおこなった日時が付加されているので文法を改良した日時と比較することもできる。図１４に処理日時の表示例を示す。

３．評価結果の検索

Muプロジェクトでは、翻訳システムの訳文の質を人手により評価している［１０］。その評価結果を検索し、特定の評価が与えられている文、たとえば、忠実度の低い文、の対訳などを出力し、文法記述者が検討できるようになっている。

KEY	:AOUT	TROUT	GOUT
E82060001_2_1	:11/29/84 13:56:43	11/24/84 13:16:39	11/24/84 13:22:07
E82060001_3_1	:12/03/84 14:26:52	11/14/84 19:25:00	11/14/84 19:25:06
E82060001_4_1	:11/24/84 11:31:43	11/24/84 13:16:50	11/24/84 13:22:32
E82060001_5_1	:11/24/84 11:34:19	11/24/84 13:16:59	11/24/84 13:23:03
E82060001_6_1	:11/24/84 11:41:26	11/24/84 13:17:19	11/08/84 10:32:49
E82060002_2_1	:11/24/84 12:25:04	10/18/84 17:09:44	10/18/84 17:09:58
E82060002_3_1	:11/24/84 12:26:07	10/03/84 17:07:20	10/09/84 09:24:58
E82060002_4_1	:11/24/84 12:28:43	10/03/84 17:07:43	10/09/84 09:26:00
E82060002_5_1	:11/24/84 12:31:56	10/03/84 17:08:17	10/09/84 09:27:04
E82060002_6_1	:11/24/84 12:33:52	11/24/84 12:45:02	11/24/84 12:45:47
E82060003_2_1	:11/24/84 12:56:57	10/03/84 17:08:44	10/09/84 09:27:32
E82060003_3_1	:	10/08/84 17:28:21	10/09/84 09:28:08
E82060003_4_1	:09/20/84 12:19:24	10/03/84 17:10:58	10/09/84 09:30:11
SUCCESS	: 12/ 13 (92%)	13/ 13 (100%)	13/ 13 (100%)
ESCAPED	: 0/ 13 (0%)	0/ 13 (0%)	0/ 13 (0%)

図１４　中間結果の処理日時の表示例
(E8206001_3_1 等は文の番号で、AOUT、TROUT、GOUT はそれぞれ解析、変換、生成結果である。)

５．　おわりに

Muプロジェクトにおける翻訳実験支援ツールのうち、辞書システム管理支援ソフトウェアを中心に述べた。辞書情報が大量になるにつれ、情報の集中管理が重要となる。そのためにオブジェクト指向の考えを取り入れて階層的な辞書システムの組織化を行った。また、パターン記述を用いて辞書情報の形式面と内容面との分離を計り、MTフォーマット辞書管理ツールの統一化を進めた。今後は

１．文法規則管理も含めて支援環境統合化を進め、
２．オブジェクト指向の考え方を辞書の管理だけではなく、文法規則により辞書データを参照する場合にも利用できないか検討する予定である。

最後に、この研究は、Muプロジェクトの処理システムおよび辞書システム作業グループのメンバの協力により開発されたものであることを記して、感謝の意を表する。

なお、本研究は国の科学技術振興調整費による「日英科学技術文献の速報システムに関する研究」の一部として行ったものである。

参考文献

［１］長尾真、科技庁機械翻訳プロジェクトの概要、情報処理学会自然言語研究会資料38-2、1983。

［２］中村順一他、Muプロジェクトにおける辞書の運用方式－日英変換辞書と英語生成辞書－、情報処理学会自然言語シンポジウム予稿集、1984。

［３］日本科学技術情報センター他、日英科学技術文献の速報システムに関する研究：日－英科学技術用語辞書データベースの開発に関する報告書、1984。

［４］電子技術総合研究所他、日英科学技術文献の速報システムに関する研究：言語処理システムの開発に関する報告書、1984。

［５］中村順一、文法記述用ソフトウェアGRADE、情報処理学会自然言語研究会資料38-4、1983。

［６］片桐恭弘、MTM & MTUTY 利用の手引き、京都大学工学部電気工学第二教室長尾研究室内部資料、1984

［７］Charniak, E., C. K. Riesbeck, and D.V. McDermott, Artificial Intelligence Programming, Lawrence Erlbaum Associates, 1980.

［８］小木正三他、機械翻訳のためのソフトウェアGRADEの文法開発環境、情報処理学会第28回全国大会、p1271、1984。

［９］中村順一他、マルチウィンドを利用した機械翻訳のための文法開発ツール、情報処理学会第29回全国大会、p1225、1984。

［１０］長尾真他、Muプロジェクトにおける日英翻訳結果の評価、情報処理学会自然言語研究会資料47-11、1985。