自然言語処理 45－1
（1984. 9. 28）

辞書編集用フレームエディタ

小暮 潔　　横尾 昭男　　島津 明　　野村 浩郷
(日本電信電話公社 武蔵野電気通信研究所)

1. はじめに

計算機による自然言語理解、その応用としての機械翻訳などを行うためには、多様な知識が必要であり、それらを蓄積するための知識ベースとして、辞書が不可欠である。また、辞書に蓄えられた知識の質と量が言語理解の奥行き、機械翻訳などの結果の質を決定すると言っても過言ではない。そのため、種々の情報の入った大規模辞書の構築法が必要となる。

我々は、自然言語理解の具体的な題材として機械翻訳を取り上げ、機械翻訳実験システムLUTE (Language Understander, Translator & Editor) を作成している。その中で、以下のような立場から辞書の構成について検討した。

(1) 辞書は、言語モデルに基づき、言語理解に必要な知識を蓄える知識ベースの重要な部分である。

(2) 辞書に蓄積すべき知識は変更され、追加される可能性を持っている。したがって、辞書は、人間とのインタラクションなどから知識を吸収し、成長する。

特に、辞書の蓄積すべき内容、辞書の表現方法、及び、辞書を構築するための道具について検討した。まず、格構造モデルに法情報、接続情報を加えた拡張格構造モデル[1]に基づき、辞書の内容についての検討を行った。この内容を表現するために、FRL[2],[3]を元にしたフレーム型知識表現言語を用いた。また、このフレーム表現による辞書に快適なマン・マシーン・インタフェースを与えるなどの目的から、言語処理支援環境の中核となるマルチウィンドのフレームエディタを作成した。このエディタは、Lisp Machine 上で実現されている。LUTE においては、解析結果の表示などにもこれを使用する。本報告では、これらについて述べる。

2. 辞書の位置付け

辞書は、言語理解に必要な知識ベースである。したがって、言語モデル、言語処理によって、その役割は異なる。LUTEシステムは、翻訳の過程を「解析」、「変換」、「生成」の三段階に分ける意味構造トランスファ方式をとるが、辞書はこれらの処理に必要な知識を提供しなければならない。各過程におけるあいまい性の解消などにおいては、common sense reasoning が重要な役割を果たし、そのための表現と意味の間の関係を示す言語的知識や専門知識、一般常識が不可欠である。

また、辞書を知識を蓄積する器と考えた場合、先ず、言語モデルに基づき、必要な知識を記述できる記述能力が要求される。それと同時に、将来の言語モデルの拡張などに合わせて、記述を拡張できる発展性も要求される。

辞書に必要な知識は、本格的な言語理解システムの構築を考えると、膨大な量となる。また、これらの知識は現在、まとめられた形で存在するわけではない。したがって、辞書を一度に構築することは非常に困難である。そこで、システム作成者やユーザが段階的に構築することになる。そのためには、辞書の内容を容易に検索、編集できることが重要である。さらには、編集された知識の無矛盾性などを管理する機能も必要となろう。

以上のような要求を踏まえて、辞書は構築されねばならない。

3. 辞書の内容

3.1 解析・生成用辞書

解析・生成用辞書に入る情報を以下の3種類に分類する。

(1) 形態素・構文に関する情報
(2) 意味に関する情報
(3) 文脈に関する情報

3.1.1 形態素・構文に関する情報

主に、文章を構成する文字列から語を抽出し、構文を抽出するために使う情報である。形態素・構文に関する情報として、日本語・英語の辞書は、それぞれ以下のような情報を持っている。

(1) 日本語解析・生成用辞書
　　品詞、活用、連接、
　　表記(漢字・ひらがな・ローマ字)

(2) 英語解析･生成用辞書

品詞、活用(原形･過去･過去分詞･比較級など)

3.1.2 意味に関する情報

主に、表層の表現から意味構造表現を得るために使う情報である。意味に関する情報として、辞書は、各語の各語義について、概念カテゴリ、対象関係、格関係、事象関係、事象-対象関係などの情報を持っている。

(1) 概念カテゴリ

日本語の辞書においては、各語義について、概念カテゴリの情報がある。名詞などの場合には、上位概念が入る。また、動詞などの述語の場合、その意味素性が入る。

英語の辞書においては、名詞の各語義に対して、Longman社のLEXICON辞書のカテゴリを拡張したものが入る。例えば、"infant"に対しては、"C008"("kinds of child")が入っている。

(2) 対象関係

複合名詞を構成する名詞と名詞との関係や、日本語の助詞「の」で結ばれた名詞の間の関係を解析するためなどに必要な情報である。例えば、「急行電車の窓」の場合、名詞「窓」に対して、先行する名詞の概念カテゴリが「乗り物」で、間を結ぶマーカが助詞「の」の場合、全体-部分の関係になることを示すためなどに用いる。これは、次のようなパターンによって表現される。

```
[ 関係カテゴリ  = 全体-部分
  マーカ        = 格助詞「の」
  前接カテゴリ  = 建築物、乗り物 ]
```

また、「心の窓」のような比喩的な表現の場合、「心」というインスタンスが「窓」を修飾することによって、「心」の概念が抽象化される。これは、手続きを埋め込むことによって示せる。

```
[ 関係カテゴリ    = 比喩
  マーカ          = 格助詞「の」
  前接インスタンス = 「心」
  手続き          = (カテゴリ←抽象) ]
```

(3) 格関係

動詞などの述語の各語義に対して、格パターンを示すための情報である。各々の格に対して、格名、マーカ、カテゴリが入る。動詞「書く」の一つの語義は、次のようなパターンで示せる。

```
[ [ 格名       = 動作主
    格マーカ   = 格助詞「が」
    カテゴリ   = 人間 ]
  [ 格名       = 対象
    格マーカ   = 格助詞「を」
    カテゴリ   = 文字、文、書籍 ]
  [ 格名       = 場所
    格マーカ   = 格助詞「に」
    カテゴリ   = 紙、黒板 ] ]
```

(4) 事象関係

複文などにおける事象間の関係、例えば、時間関係などを解析するために用いられる情報で、述語の語義に付与される。

(5) 事象-対象関係

埋め込み文などにおいて、明確に格として捕らえにくい表現(例えば、「魚の焼けるにおい」)や同格表現などを解析するために用いる情報である。

3.1.3 文脈に関する情報

省略の補完、代名詞の照応、相対的な時間関係の解析などの文脈処理に必要な情報である。その他、専門知識や一般常識もある。

3.2 変換用辞書

LUTEシステムにおける変換過程では、原言語依存意味構造表現と目的言語依存意味構造表現の間で、構造変換と語彙変換を行う。構造変換は、IF-THEN-ELSE型の規則を適用することにより行う。すなわち、変換規則と原言語依存意味構造表現の部分とのパターン･マッチを行い、対応する目的言語依存意味構造表現を作り出す過程を再帰的に行う。語彙変換は、訳語の対を参照しながら行う。したがって、変換用辞書には、構造変換規則と訳語の対が入ることになる。

4. 辞書の表現方法

4.1 辞書に要求される表現能力

辞書は、前の章で述べた情報を表現できなければならない。そのためには、次のような要求条件があげられる。

(1) 辞書エントリの持つべき情報は、各品詞ごと、あるいは、各エントリごとに大きく異なっている。また、語義や、意味に関する情報は、一つの項目に複数の情報が入る可能性がある。したがって、定形的なレコード形式では表現しきれない。より柔軟な表現が要求される。

(2) 辞書エントリの持つべき情報は、階層性を持っている。また、意味に関する情報などは、多様なパターンで表されている。すなわち、構造を表現する能力が要求される。

(3) 意味情報の中の対象関係の例で示したように、意味情報の中には手続き的知識が入る。したがって、宣言的知識と手続き的知識の双方が表現できなければならない。

(4) 用例の収集や言語処理の実験結果などにより、辞書の内容は更新、拡充されていくことになる。また、将来の言語モデルの拡張によって、必要な情報が大きく拡張される可能性もある。したがって、表現の拡張性が要求される。

また、辞書が持つことが望まれる条件として、以下のようなものがある。

(5) 辞書の中では、派生語とそのもとの語のように、一部が全く同じ情報を持つ辞書エントリがある。このような場合、情報を共有し、一ヵ所で持っていれば、変更時に、一ヵ所だけ変更すればよく、辞書の更新が容易になる。

(6) 意味に関する情報の中には、上位概念の情報が下位概念に対して、そのまま、適用できるものがある。したがって、情報の上位概念から下位概念への遺伝(inheritance)ができれば、表現の重複を避けることができる。また、意味的な階層関係以外にも、形態素情報、例えば、品詞などからの情報の遺伝が考えられる。したがって、複数の要素から情報を遺伝させる(multiple inheritance)の機構が必要と考えられる。さらには、情報の種類によって、遺伝を制御できる機構があれば、便利である。

(7) 言語理解の実験などにおいては、知識がない場合の挙動を知りたいことがある。このような場合、辞書の内容を変化させるのではなく、辞書の各情報に錠前を付け、辞書をアクセスするときの鍵によって、情報の授受を制御する機能があれば、便利である。

(8) 辞書は言語理解に使うための知識を蓄えるための知識ベースであることから、言語の意味構造表現と親和性を持っていることが重要である。

そこで、これらの条件を満足している方法としてフレーム型知識表現言語を用いている。

4.2 フレーム型知識表現言語

フレームは、基本的にFRLに準拠した。フレームは、最高5レベルまで埋め込まれた関係リストから構成されている。各サブストラクチュアは、それぞれ、スロット、ファセット、データ、ラベル、メッセージとよばれる。全体としては、

```
(frame
   (slot   (facet (datum    (label message...)
                            (label message ...) ...)
                  (datum ...) ...)
           (facet (datum ...) ...) ...)
```

のような形になる。

CONDUCT-VERB		
*TYPE	$VALUE	PROTOTYPE
*POS	$VALUE	VERB
*ROOT-FORM	$VALUE	CONDUCT
*MEANING	$VALUE	CONDUCT-VERB-1

CONDUCT-VERB-1		
*TYPE	$VALUE	PROTOTYPE
*CASE-FRAME	$VALUE	CONDUCT-VERB-CASE-FRAME-1

CONDUCT-VERB-CASE-FRAME-1		
*TYPE	$VALUE	PROTOTYPE
*VCAT	$VALUE	TV
*CASES	$VALUE	CONDUCT-VERB-CASE-1-1
		CONDUCT-VERB-CASE-1-2

CONDUCT-VERB-CASE-1-1		
*TYPE	$VALUE	PROTOTYPE
*CATEGORY	$VALUE	I136
*MARKER-CATEGORY	$VALUE	SUBJECT
*NAME	$VALUE	AGENT

CONDUCT-VERB-CASE-1-2		
*TYPE	$VALUE	PROTOTYPE
*CATEGORY	$VALUE	*Z012
*MARKER-CATEGORY	$VALUE	OBJECT1
*NAME	$VALUE	RANGE

Fig-1 フレームによる辞書記述の例
(英語の動詞"CONDUCT")

各スロットには、ファセットを使って、様々な情報を付与することができる。ファセットには、実際のデータを保持する$VALUEのほかに、デフォルト値のための$DEFAULT、データの追加、削除などによって起動される手続きのための$IF-ADDED、$IF-REMOVEDなどがある。

メッセージでは、辞書のアップデートの記録を残すHISTORY、データの錠前となるRESTRICTを新しく、追加した。このRESTRICTを使い、inheritanceを制御することができる。また、inheritanceのために上位概念を指定するスロットを変更する機能を追加した。

フレームを操作する基本関数として、データを参照·追加·削除するFGET、FPUT、FREMOVEなどを用意し、これを基に関数は定義される。

5 辞書編集用フレーム·エディタ

フレーム型知識表現言語は、計算機上で知識を表現するという点では、非常に便利な道具である。しかし、我々、人間の立場で考えた場合、フレームで表現された知識を素早く理解し、編集することが難しいなどの問題点がある。特に、自然言語理解システムなどによって文章を解析して作られた言語の意味構造表現のフレームを理解することは難しい。これには、以下のような理由が考えられる。

(1) 一般に、一つのまとまった知識は、複数のフレームによって表現されている。例えば、LUTEシステムにおいて、一文の意味構造を表現するのに数十個~数百個のフレームが使われている。

(2) このようなフレームは、単純な木状に関係しているのではなく、双方向のリンクによって相互に複雑に関係しあったネットワークになっている。

(3) このようなフレームの集まりの全体像、概観などを表示、編集するための有効な道具がない。

このようなフレーム型知識表現言語の短所を克服し、辞書の構築を容易にするための道具として、マルチウィンドの辞書編集用フレームエディタを作成した。

5.1 フレームエディタの設計構想

フレームエディタの目標は、フレーム型知識表現言語によって構築された知識ベースの編集を容易にする環境を提供することである。ここでは、フレームエディタを使用する状況として以下のような状況を想定してみる。

(状況1) 言語解析用の辞書を更新した後、解析プログラムを起動し、その効果を確認する。

(状況2) 原言語の意味構造表現と対応する目的言語のそれを比較しながら、意味構造変換の規則を作成し、テストを行う。

このような使用形態から以下の点を考慮した。

5.1.1 表示

言語の意味構造表現は、多くのフレームから構成されている。したがって、多くのフレームを同時に表示できることが重要である。そのためには、高精細で大きなスクリーンを持つ必要がある。また、これらのフレームは、集まりとなって、意味を持つものである。したがって、フレームの集まり全体の構造が把握できることが重要である。

実際に、特定のフレームの内容を編集しようとするとき、情報を全て表示することが望まれる場合もあれば、特定の情報だけ表示したい場合もある。すべてのフレームの必要な情報をすべて表示しようとすると、表示する情報が多過ぎ、同時に表示できない。したがって、複数のフレームを適切に表示する方法が要求される。

また、(状況2)の場合、複数のフレームの集まりを表示する画面と、規則などのそれ以外の情報を表示する画面を同時に表示することが重要である。

さらに、フレームエディタでは編集対象によって表示したい画面は様々に変化する。したがって、ユーザがある程度、自由に画面を構成できる機能が望まれる。

5.1.2 編集

本システムで考慮する編集要素は以下である。

(1) 編集コマンドセット

フレームの場合、明確な構造を持っている。また、フレームを操作する関数が用意されている。そこで、本システムでは基本コマンドとして、PUT、REMOVEを用意する。それ以外のコマンドは、簡単にユーザが追加できるように設計されている。すなわち、自己拡張システムになっている。(状況1)、(状況2)では、フレームの集まりに対して、コマンド「解析」「変換」なども実行できるようになる。

(2) コマンドの指定方法

コマンドを指定する方法としては、

- コマンドの指定後、編集対象を指定する。
- 編集対象の指定後、コマンドを指定する。

の二通り考えられる。コマンドの拡張によって、編集対象によって適用できるコマンドの種類や定義が変わることを考え、後者を採用する。ユーザは、マウスにより対象を指定し、表示されたメニュの中からコマンドを選択し、編集する。

(3) 他の処理との関連

他の処理との関連として、

- 他の処理表示部として簡単に起動できる。
- 他の処理を容易にコマンドとして取り込める。

という2点も重視した。

5.2 フレームエディタの構成と特徴

フレームエディタは、Fig-2 に示すような構造からなる。エディタ全体は、Lispインタプリタの上で動作しているため、Lispプログラムと親和性がよく、拡張性が高い。この上にフレーム型知識表現言語がのり、フレーム自体を編集するコマンドは、この言語を媒介して、実行される。

フレームエディタの中心的な役割を果たすのはフレームを表示するための3種類のウィンドである。各ウィンドは自分自身のプロセスとして、コマンドループを持っていて、コマンドを解釈し、知識表現言語に受け渡す。フレームエディタは、独立したウィンドの集合体である。そのため、自分の編集対象に合わせてウィンドを組み合わせることが可能で、好みの編集スタイルがとれる。

5.3 フレームの省略表示

フレームを編集する場合、ユーザが必要な情報は、フレームの持っている情報の一面に過ぎない。他の情報は、ユーザが必要な情報に注目するのを妨げることになる。フレームを表示する際、必要な情報だけを取り出し、不必要な情報を省略することは、ユーザにとって、重要である。また、これは限られた画面を有効利用するためにも必要である。ただし、この情報の重要性は編集目的によって変化する。そこで、必要で適切な情報だけを表示できるようにした。これは、フレームを表示するウィンドに対する指定、ユーザ·デフォルトの指定とシステム·デフォルトの指定の3種類によって行われる。指定は、この順に優先される。

フレームの省略は、以下のパラメータの指定によって行う。

(1) スロットに対して

(a) (:MEMBER <スロット名>...)

指定されたスロットだけを表示する。

(b) (:EXCEPT <スロット名>...)

指定されたスロット以外を表示する。

(c) (:RESTRICT <述語>...)

全ての述語を満足するスロットを表示する。

ファセット、データに対しても同様に指定できる。

(2) オプション

以下のものから複数、指定できる。

(a) :DATA-ONLY

データ以下の構造であるラベル、メッセージを省略する。

(b) :FRAME-ONLY

フレームを指すデータだけを表示する。

5.4 ウィンドによるフレームの表示

個々のフレームの情報を既に述べた省略表示に従って表示するウィンドとして、Lispのプリティ·プリントのイメージで表示するプリティ·プリント·ウィンドと表形式で表示するテーブル·ウィンドの2種類を用意した。前者は、Lispのプリティ·プリトのイメージであるため、フレーム内に埋め込まれたLispのS式の構造が明確に理解できる。後者は、表形式で表せるので、スロット、ファセット、データの構造が明確に理解できる。また、後者のウィンドでは、フレームのサブストラクチュアがマウスで選択できるオブジェクトになっていて、マウスを使って、編集できる。

5.5 フレームの木表示

上で述べたウィンドは、個々のフレームの持つ情報を表示するには適している。しかし、一つ一つに広い面積を必要とするため、多くのフレームを表示するには不適である。そこで、Fig-3 (c) のように、フレームを木状に表示するウィンドを用意した。ここで、直方体で囲まれたノードはフレームを示し、直線上の名前はリンクの種類を示すスロット名である。

この木状に表示するウィンドでは、フレーム表示でのウィンドの省略指定と同様に、リンクの種類を指定できる。また、木は、縦にも横にも表示することができる。さらに、マウスを用いてノードを指定し、そのノードの子孫のノードを表示したり、消去することができる。このウィンドで、個々のフレームの情報を詳細に知りたいときは、上で述べ

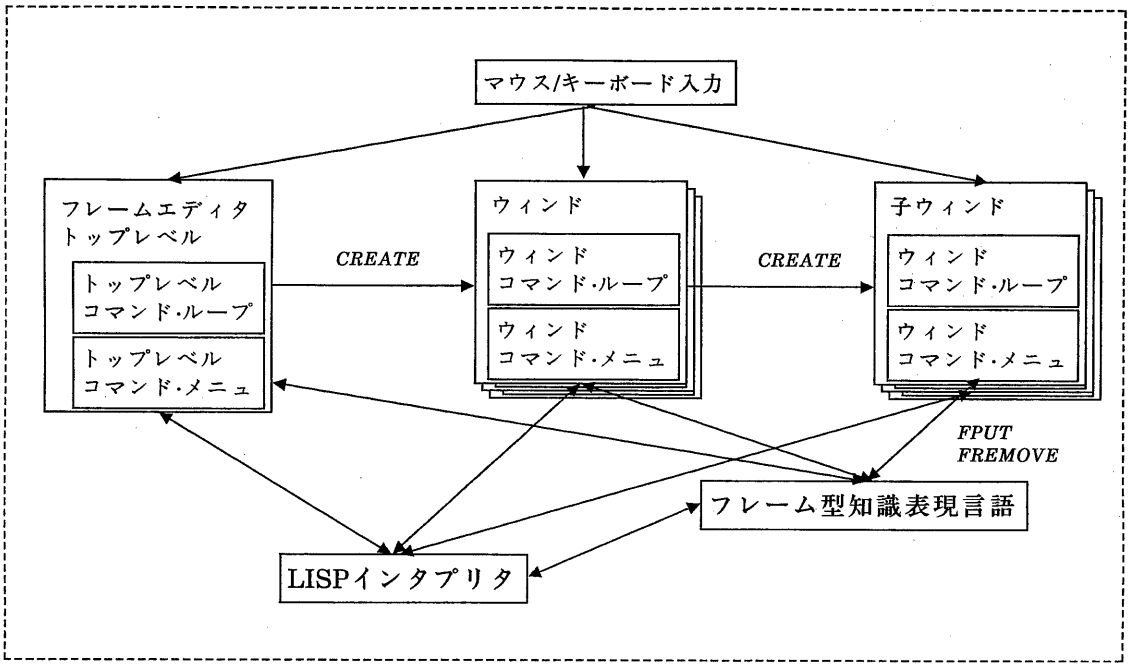

Fig-2 フレームエディタの構成

たウィンドをノードの位置に表示することができる。

5.4、5.5で述べたウィンドをもとにマルチウィンド機能を使って、複数のフレームを同時に表示することができる。

5.6 編集機能

フレームエディタの編集機能は大別して2種類ある。一つは、フレームエディタ全体に及ぶ機能で、フレームエディタを呼び出したときに、コマンド･メニュの中に含まれているトップレベルのコマンドが持っている機能である。他方は、トップレベルのコマンドによって作られたフレームを表示するウィンドに付与された機能である。

フレームエディタ全体に及ぶ機能は、全体のトップレベルであるコマンド･メニュの中からマウスで選択することによって起動される。現在、ファイルからのフレームのロード、ファイルへのセーブ、ウィンドの生成などができる。

フレームを表示するウィンド、フレーム間の関係を木状に表示するウィンドは、個々のウィンドがそれぞれ自分自身のプロセスを持っている。各プロセスのトップレベルは、Fig-4 のようなループになっている。

編集対象であるフレームのサブストラクチュアは、それぞれ自分自身に適用できるコマンド･セットを持っている。マウスで選択されると、自分に適用できるコマンドをメニュにして表示する。メニュの中からコマンドが選択されると、それを解釈し、実行した後、再表示を行う。すなわち、オブジェクト指向になっている。

このウィンドの機能は、メニュとして選択できるコマンド･セットによって決まる。フレーム操作は、基本的にはフレームのサブストラクチュアを追加するFPUTと削除するFREMOVEである。そこで、編集用の基本コマンドとして、各レベルのサブストラクチュアに対してPUTとREMOVEを用意した。これらは呼び出されると、自分自身を指すポインタとユーザから与えられた引数で、FPUT、FREMOVEを呼び出す。したがって、$IF-ADDED、$IF-REMOVEDなどの手続きを起動することができる。コマンドは、簡単に追加できるように設計されている。

最後に、フレームエディタを使って、辞書を編集している画面を示す。Fig-5 は、日本語と英語の意味構造表現を比較しながら、変換規則を作成している画面である。

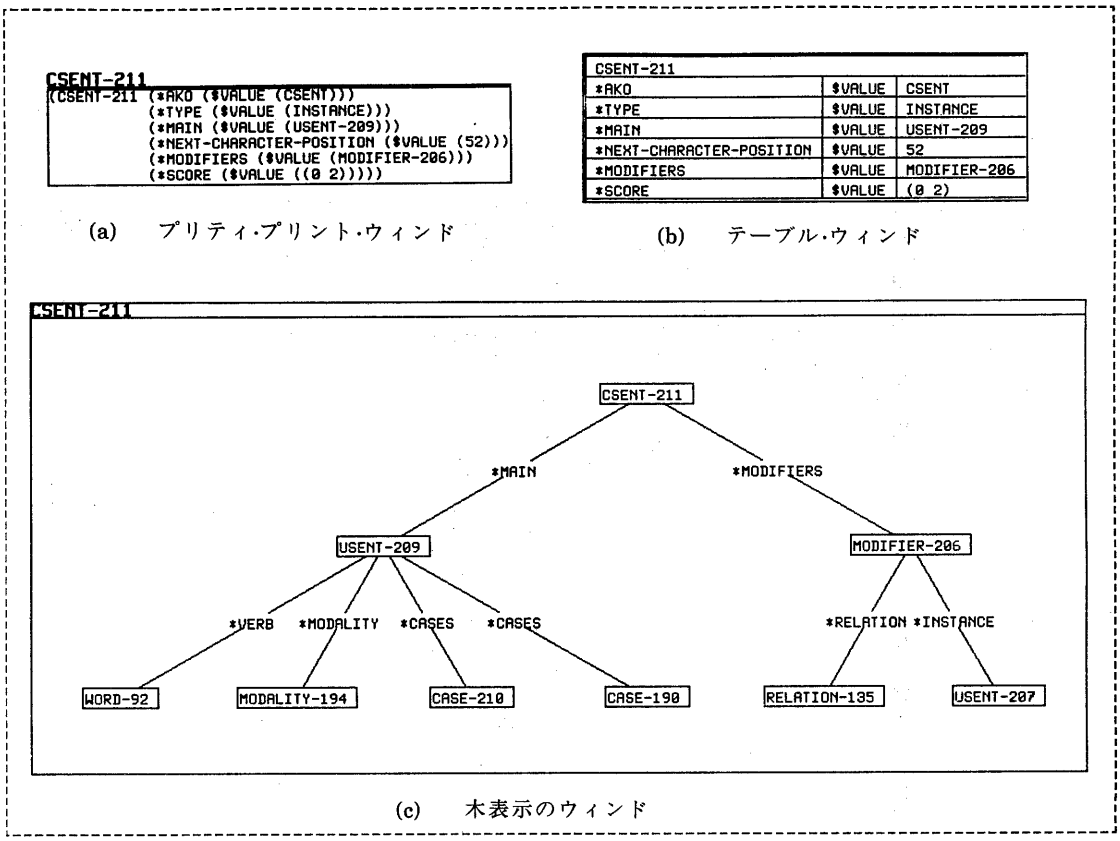

Fig-3 フレームエディタにおけるウィンドの種類

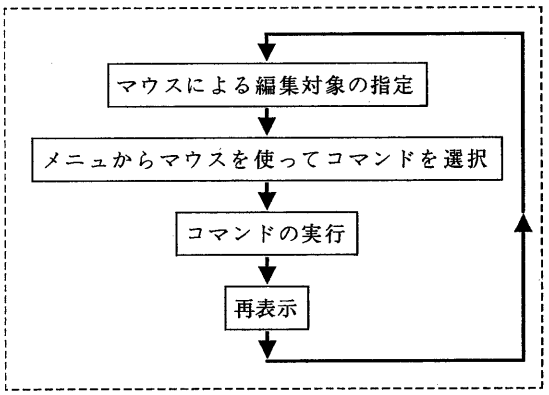

Fig-4 ウィンドのコマンド・ループ

6 おわりに

本報告では、自然言語理解システムにおける辞書の位置付け、辞書の含むべき内容、辞書の表現方法、および、辞書構築のための道具であるフレームエディタについて述べた。辞書は、拡張格構造モデルに基づき、意味に関する情報を中心としている。この意味に関する情報などを表現するために、フレーム型知識表現を用いた。フレームを用いることによって、拡張性、柔軟性のある辞書が構成された。辞書編集用のフレームエディタは、自然言語理解などの研究環境を支援するシステムとして、拡張性の高い道具である。

最後に、本研究を進めるにあたり、協力していただいた言語処理研究グループの諸氏に感謝する。

文献

[1] A. Shimazu, S. Naito and H. Nomura "Japanese Language Semantic Analyzer based on an Extended Case Frame Model" Proceedigs of the Eighth International Joint Conference on Artificial Intelligence pp.717-720,August 1983

[2] R. Bruce Roberts and Ira P. Goldstein "The FRL primer" AI Memo 408 MIT,October 1977

[3] R. Bruce Roberts and Ira P. Goldstein "The FRL manual" AI Memo 409 MIT,September 1977

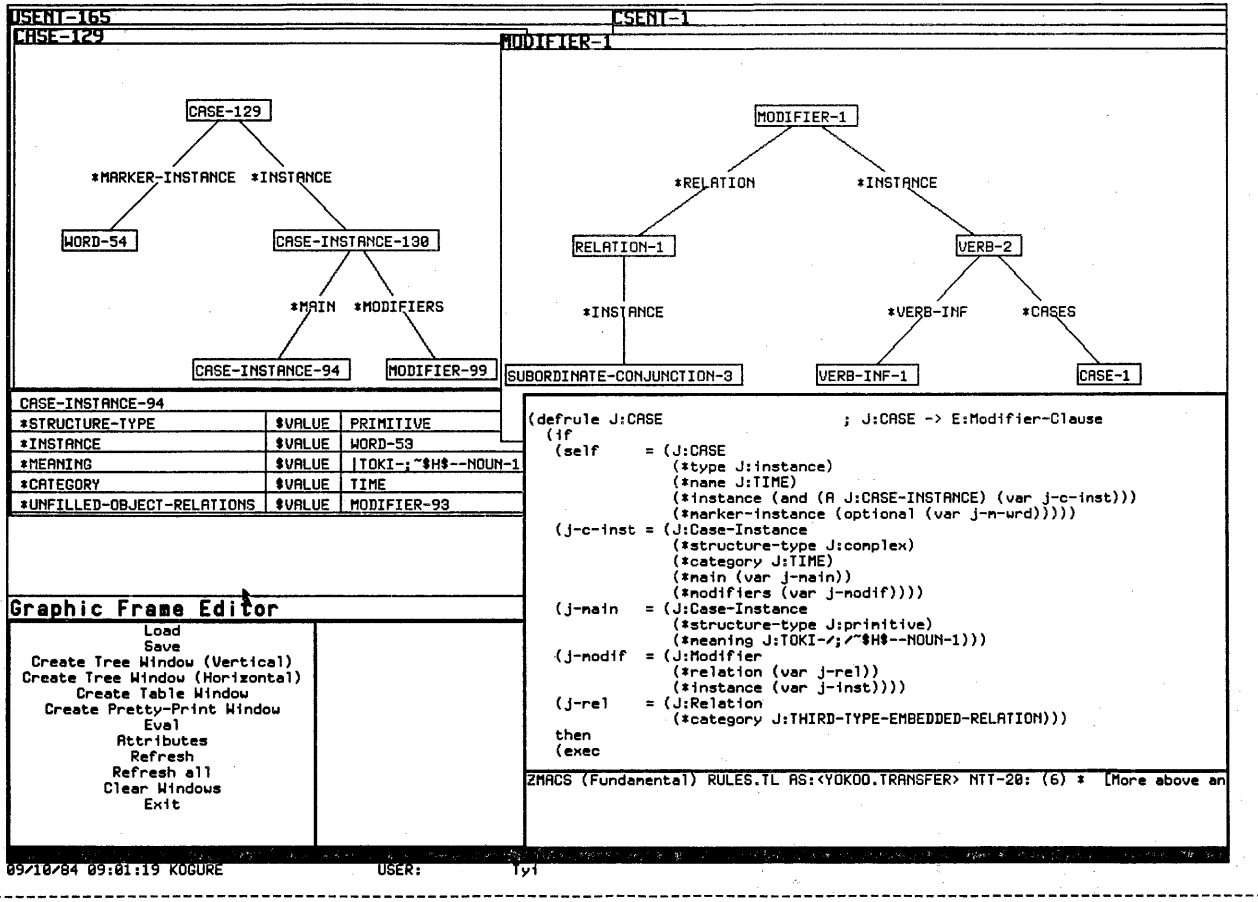

Fig-5 フレームエディタの画面の例（構造変換の規則を編集中の画面）

左上のウィンドは、解析された日本語の例文の意味構造表現の一部を木で表示している。右上のウィンドは、対応する英文を解析し、対応する部分を切り出し、表示したものである。下のウィンドでは、この二つの構造を比較しながら、構造変換の規則を編集しているところである。左下のウィンドは、エディタのトップレベルのメニュである。

（例文） 日本語 私は幼いとき、夢を持っていた。 （下線の部分が表示されている。）

英語 When I was young, I had a dream.