5M−6　論理型オブジェクト指向プログラミング言語CESPによる[1] UNIXコンサルテーションシステムの開発 ー実現方式と評価ー

川越 恭二[†]、渡部 潤[††]、宮下 恵子[††][2]
([†](株)AI言語研究所、[††]日本電気マイコンテクノロジー(株))[3]

1　はじめに

CESPは、論理型ワークステーションPSI上でしか動作しなかったESPを一般の計算機でも使用できるように拡張・汎用化したものである。また、CESPは論理型言語であるPrologにオブジェクト指向を融合したプログラミング言語である。

UNIXコンサルテーションシステムは、このCESPの機能を利用しながら開発を行い、その過程でCESPの評価を行った。

2　CESPによるUNIXコンサルテーションシステムの実現

2.1　クラス構成

本システムのクラス構成を以下に示す。

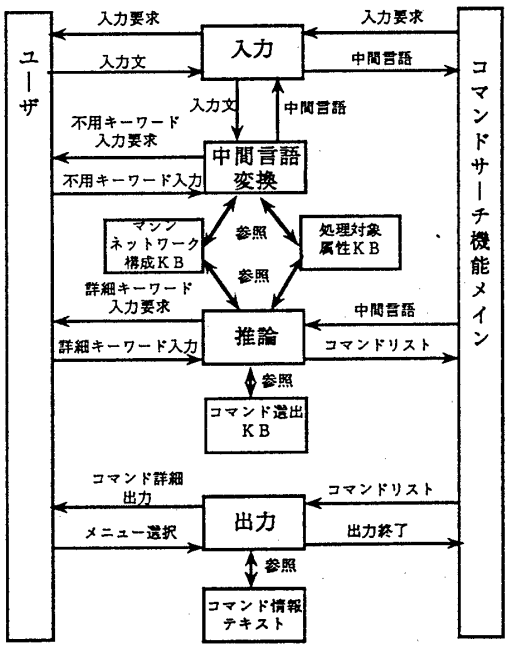

図1　システム構成図

それぞれの枠はオブジェクトを表している。入出力、および推論はさらにいくつかのオブジェクトで構成されている。

オブジェクト同士が、互いにメッセージを送り合うことでシステムが動作する。

2.2　対話式ユーザ・インタフェース

2.2.1　日本語による対話式入力

対話式入力は、次のような一連の入力操作である。まず最初に日本語により問い合わせ文を入力する。そして、それを中間言語に変換する。このとき問い合わせ文が不十分な場合は、不足分の入力が必要となる。一方、問い合わせ文が抽象的な場合は、より具体的な知識の選択入力必要となる。

CESPは、X-windowとのインタフェースを持っており、また日本語処理機能を有する。これらの機能を使うことにより、日本語による入力が可能となった。また、CESPはウィンドウをオブジェクトとして扱うため、上記の入力をそれぞれ個別のウィンドウとして機能させること容易に実現した。

問い合わせ文は入力後リストの形の中間言語に変換されるが、これはCESPのストリング処理機能を使うことで容易に実現した。

2.2.2　メニュー選択による対話式出力

問い合わせ文をもとに知識ベースを探索して推論し、適切なコマンドのリストとその簡単な解説をメニューとして出力する。ここで任意のコマンドを選択すると、'使い方'・'解説'・'オプション'・'環境設定'という項目のメニューが現れ、さらにそこで必要とする項目が選択されると、詳細なコマンド情報を提供する。

最後の詳細出力のための情報は、そのために特別に知識ベースを構築するのではなく、UNIXとのインターフェース機能を使ってテキストファイルの一部を読み込むようにした。

このとき出力を項目別にウィンドウに出力したり、画面のスクロールをマウスで行うことがウィンドウライブラリにより容易に実現した。

2.3　オブジェクト指向型知識ベース

本システムの知識ベースは以下のものがあり、オブジェクト指向の概念を用いて構成してある。

1. マシンのネットワーク構成を表す知識ベース
2. 処理対象の属性を表す知識ベース
3. コマンドを選出する知識ベース

これらの知識ベースは、図2のような関係をもっている。

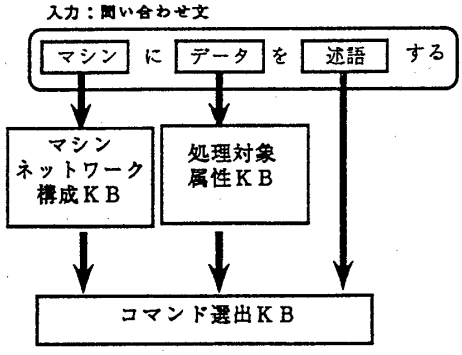

図2　知識ベース間の関係

[1] UNIX CONSULTATION SYSTEM by Object-Oriented Logic Programming Language CESP - Implementation and Evaluation -
[2] Kyouji Kawagoe[†], Jun Watanabe[††], Keiko Miyashita[††]
[3] [†]AI Language Reaearch Institute,Ltd.
[††]NEC Microcomputer Technology,Ltd.

2.3.1 階層型知識ベース

マシンのネットワーク構成を表す知識ベースと処理対象の属性を表す知識ベースは、階層を使って表現することにしたが、これらをオブジェクト指向のクラス階層を利用して表現する。これにより上位のクラスの概念を下位のクラスに継承できる。こうした階層型知識ベースは、CESP のクラス継承機能を用いることで容易に実現できた。

例えば、1. の知識ベースは、CESP では以下のように記述できる。

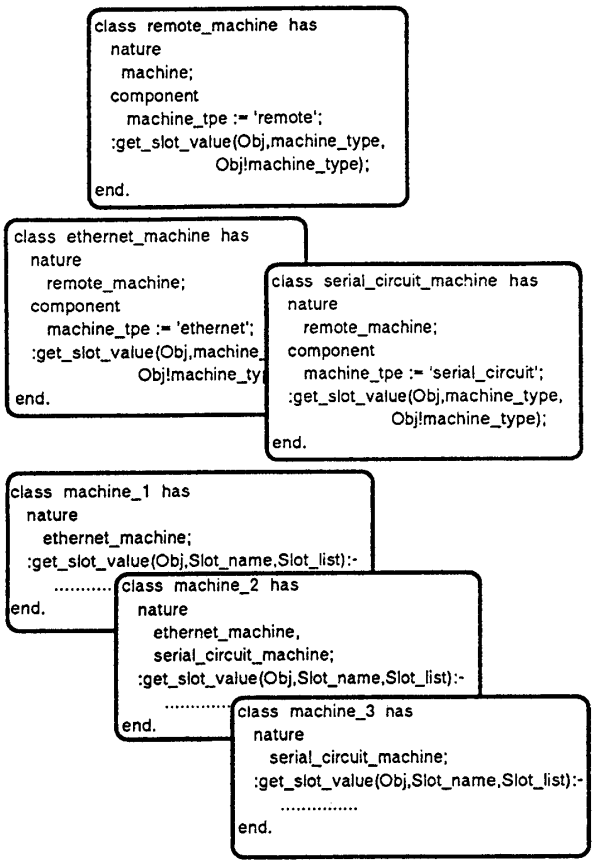

図3 CESPによるマシンのネットワーク構成ＫＢ

この知識ベースは、主にマシンの属性を参照するために使用するものであるが、CESP はスロットの記述ができ、ここにマシンの属性を定義し、それを参照することで属性を得ることができる。上位で定義したものは、書換えの場合をのぞき定義しなおす必要がない。

CESP のスロットには、他のクラスからも参照できる属性スロットと、定義したクラスのみで参照できる要素スロットの二つタイプがある。クラス継承することで上位のスロットを参照するためには、属性スロットで定義することで可能となるが、その探索の順序は深さ優先のため、多重継承してる場合に直上のクラス全てのスロットを求められない。そこで、要素スロットを使いスロットの参照をそのクラス内部で行うようにした。

処理対象の属性を表す知識ベースは、処理対象となる物の属性を参照するために使用するものであるが、マシンのネットワーク構成の知識ベースと同様に記述した。

2.3.2 マトリックス型知識ベース

コマンドを選出する知識ベースは、上記のそれぞれの階層型知識ベースから得た知識を表形式にまとめ、さらにそれをキーワードの述語別にクラス定義をし、その交わるところに該当するコマンドリストを格納するマトリックスを用いた方法で表現した。

このとき階層型知識ベースの継承機能を利用することができるので、この知識ベースで全てのマシンのネットワーク構成・処理対象物のマトリックスを作成する必要はなく、それぞれの階層型知識ベースの中で基本となる上位オブジェクト同士の組合せについてのみ記述する。

また、同義語の述語に関しては、マトリックスを語数分作成するのでなく、1つにまとめることで知識ベースの簡略化をはかった。

このように知識をモジュール化して追加・修正を容易すると同時に、階層型の知識ベースと併用することによってより少ない記述で広範囲の知識に対応できた。

3 CESP の評価

UNIX コンサルテーションシステム CESP で開発してみての評価を以下に示す。

- オブジェクト指向パラダイムを全面的にとりいれたシステムの構想が可能である。
- X-Window 上で稼働するメニュー・ウィンドウ, ウィンドウの機能のライブラリがあり実現が容易である。
- クラス階層により、階層構造の知識ベース記述に優れている。また階層構造の機能を生かすライブラリが豊富である。
- 日本語入出力が可能である。
- emacs 上での開発環境が充実している。
- Prolog のように宣言的記述できるのに加えて、機能ごと、知識ごとにクラス分けすることにより、プログラムの部品化・再利用ができ追加・修正が容易にできる。

4 課題

以下に、CESP および、UNIX コンサルテーションシステムに関する今後のの課題を示す。

1. CESP に関して
   - コンパイル多少時間がかかるので、短縮が必要
   - ウィンドウ機能の拡充
   - 知識ベース構築エディタの検討

   尚、本システムは CESP 基本版 (V2.3) で開発した。来年度に完成予定の CESP フルセット版ではこれらの課題は解決されることが期待できる。
2. UNIX コンサルテーションシステムに関して
   - 問い合わせ文の言い回しを多様化する
   - コマンド名からも直接、情報を呼び出せる機能の実現
   - 環境設定エラー解析機能の開発

参考文献

[1] R.Wilensky,D.N.Chin,M.Luria,J.Mayfield,and D.Wu, "The Berkeley UNIX Consultant Project" Computional Linguistics ,Dec,14,35-84,1988
[2] 近山 他,"ESPer への道" ,bit Vol.22,No.1-12,1990
[3] "CESP 言語",AI 言語研究所,1990
[4] "CESP ライブラリ",AI 言語研究所,1990