C++による汎用フレーム型知識工学環境 ZERO の開発

～知識ベースファイルの分散・共有化による協調動作～

2M－3

台良 剛*　柴田昌宏**
東京電機大学大学院理工学研究科

上野晴樹***
東京電機大学理工学部

1. はじめに

知識ベースシステムの開発、あるいは知識工学の分野において、対象となる問題領域の専門家(エキスパート)の知識をどのような形式で表現するかという知識表現(Knowledge Representation)の問題は、知識の獲得・利用と並ぶ問題解決上の最重要課題の一つである。当研究室では、M.Minskyにより提案されたフレーム理論に基づく知識表現方式を採用した知識工学環境 ZERO の研究、開発および, その利用を進めている。

ZERO はこれまで、InterLisp 環境や CommomLisp 環境および C++[1][2]において実現され、知的プログラミング環境 INTELLITUTOR[3]や、自律型人型知的サービスロボット HARIS[4]をはじめとした様々な応用に用いられている。

C++版の ZERO は、C や C++で記述された手続きやオブジェクトと、知識ベースとしてのフレームシステムを融合する、という要求を満たすために、フレームシステムと類似点を持つ C++によってインプリメントすることで、実用性の向上を実現したシステムである。

ただし、94 年度に開発されたこの C++版の ZERO[1]は、いくつかの理由から部分的な利用にとどまっている。

本報告における ZERO システム（以下、ZERO++と記す）は、94 年度に開発された C++版の ZERO に改良を加え、設計方針とそのモジュール構成を明らかにし、さらなる改良をしやすくすることがその目的である。また、現在我々が開発中であるロボットシステムは、C++版 ZERO の本格的な応用システムであり、このシステムが本来必要としている、複数モジュールからの単一知識ベースの共有、実行時の並列動作といった機能を ZERO++で実現することもその大きな目的の一つである。

本研究の目的をまとめると主に次の４つである。

- インタフェースと知識表現の分離
- ネットワーク化による資源の共有と並列処理
- 分散環境化
- 自律型人型知的サービスロボット HARIS のソフトウェアプラットフォームとしての利用

なお、ZERO++は、Sun WS 上で、X-Window ｼｽﾃﾑ (X11R6)と GNU g++を用いて開発している。

Development of Frame-based Knowledge Engineering Environment ZERO in C++
*Takeshi Daira
Tokyo Denki Univ. Graduate School of Science and Engineering
**Masahiro Shibata, ***Haruki Ueno

2. ZERO++の設計・方針

(1)　フレーム・システムの設計

フレーム・システムをオブジェクト指向で構築する際に問題となる点は、フレーム・システムを知識表現としている知識ベースを用いた推論においては、そのフレームは常に変化するということである。このため、C++を用いてフレーム・システムを構築する際、次の点に注意した。

- フレーム・システムで用いられるオブジェクトはすべて定義しておく
- フレーム・システムを構築するデータ構造をすべてオブジェクトで表現する

(2)　ZERO++のシステム構成

ZERO++は三層構造を持つ。図１に概念図を示す。

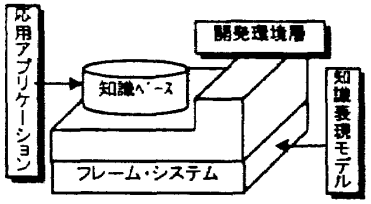

図１： ３層構造

- 知識表現モデル層

知識を表現するモデルを提供する層である。

- 開発環境層

知識ベースを構築し、また、推論を行なうための環境を提供する層である。また、推論を簡単に行なうためのシステム関数群をもつ。

- 応用アプリケーション層

知識ベースエディタを用いて記述された知識ベース(KB)である。

(3)　モジュール構成

ZERO++のモジュール構成を図２に示す。

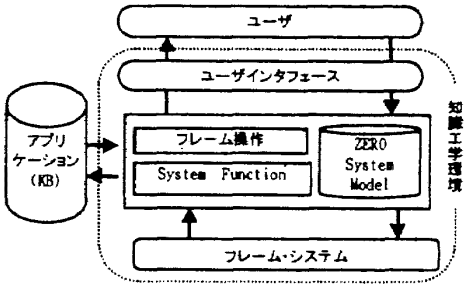

図２：モジュール構成

- ZERO System Model

知識工学環境 ZERO としての知識ベースである。ZERO はこの知識ベースにより動作する。

- インタフェース

X-Window システムベースのインタフェースと、C 言語のライブラリ関数群とで構成する。

- アプリケーション

知識ベース(KB)である。

(4) ZERO++による応用システムの開発

ZERO++で応用システムを開発する例として、ロボットシステムとの関係を図３に示す。

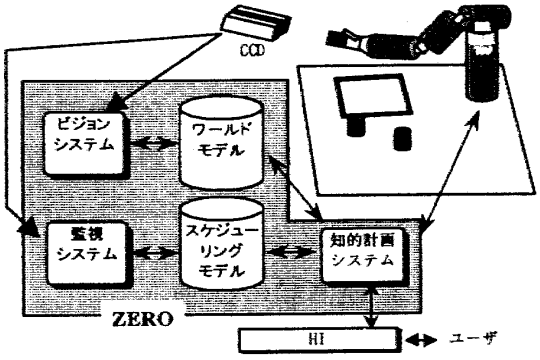

図３：応用システムの例

3. RPC による ZERO++のネットワーク化

ＺＥＲＯで分散処理を行なう目的は

- 資源の共有
- 並列処理
- 共同作業

などであるが、RPC でネットワーク化した場合には、同時に複数のクライアントがサーバにアクセスした場合でも、サーバはそれぞれのクライアントが要求した知識ベースを開き、処理しなければならない。特に、同じ知識ベースに対して接続する場合は、知識ベースを二重に開かないように処理し多重アクセスによる矛盾が起こらないようにしなければならない。

具体的には、ＫＢマネージャに現在開いている知識ベースの一覧をもたせておき、クライアントは、使用するＫＢをＫＢマネージャに問い合わせる。ＫＢマネージャは複数のクライアントから同じ知識ベースへの接続要求がきても、知識ベースを二重に開かないようにするなどの仕組みが必要である。本研究では、一覧表によって、クライアントと知識ベースの接続や操作を管理する（図４）。

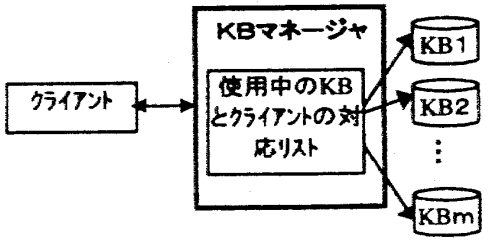

図４：一覧表の概念図

また、各ＫＢへの実際のアクセスはＫＢ毎に別プロセスで行う。これは、一つのサーバプロセスがシステムダウンした場合に、他のＫＢへのアクセスの影響を減らす為である。また、後で述べる並列処理とも関係する。

- RPC による並列処理について

次に既存のアプリケーションを分散化し、並列処理をする方法であるが、RPC を利用する。

並列処理は、ZERO の推論制御の一つである「付加手続き」を実装するために不可欠な機能である。特に、ロボットシステムでは複数の対象を並列に制御する必要があるため、欠く事ができない機能であるといえる。

4. まとめ

本報告では、汎用フレーム型知識工学環境の C++による実装方法と、知識ベースファイルの分散・共有化について示した。

フレームシステムは、従来は、LISP で開発されてきたが、C++でのフレームシステムの実現は、LISP による実現とは異なった技法が必要であった。中でもデータタイプの制限は非常に厳しく、全てのデータタイプをオブジェクトで表現することにより表現した。また、LISP では比較的容易にインプリメントできる不可手続きも、C++では困難であり、現段階では、ZERO++に精通していなければ使用するのは困難であり、環境もまだ整っていない。

次に、知識ベースファイルの分散・共有化への対応に関しては、現在発展途上であるが、実用的なシステムを実現するために非常に重要な部分である。

今後は、ロボットを制御するための機能として、専用のインタフェースを付加あるいは組み込むための機構を用意することも必要とされていくものと考えている。また、ユーザインタフェースの統一、優れたユーザインタフェースの実現、開発コストの削減などの理由から、開発環境として PC を考慮している。

参考文献

[1] 上野 晴樹、小暮 慎一、大熊義嗣:：C++による汎用フレーム型知識工学環境 ZERO の実現.、ソフトウェア工学の基礎Ⅱ 日本ソフトウェア科学会 FOSE '95 p101～110、株式会社近代科学社 1996.1.20

[2] 大熊義嗣、上野晴樹：C++による汎用フレーム型知識工学環境 ZERO の開発、東京電機大学理工学研究科修士論文、1995

[3] 上野 晴樹：知的プログラミング支援システム INTELLITUTOR について-背景と開発思想-、情報処理学会 知識工学と人工知能 37-5 1984

[4] 上野晴樹：自律型人型サービスロボットの研究開発、日本ロボット工業会機関誌「ロボット」１０９号、pp.101-108, 1996.

[5] 柴田 昌宏、台良 剛、上野 晴樹：C++による汎用フレーム型知識工学環境 ZERO の開発、第５５回情報全大論文集 6AE-4