

6 A E - 4

C++による汎用フレーム型知識工学環境 ZERO の開発
 ~ネットワーク環境におけるリアルタイム型共有知識ベース機能~

柴田昌宏* 台良 剛**
 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 CommonLisp 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い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++

*Masahiro Shibata

Tokyo Denki Univ. Graduate School of Science and Engineering

Takeshi Daira, *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点である。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。

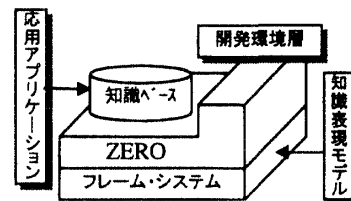


図1: 3層構造

- 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。

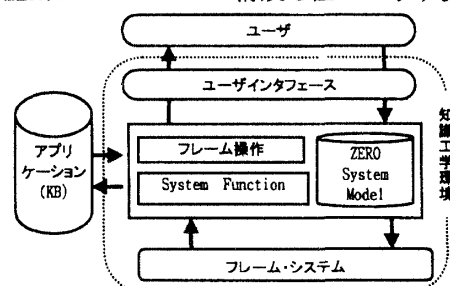


図2: モジュール構成

- 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。

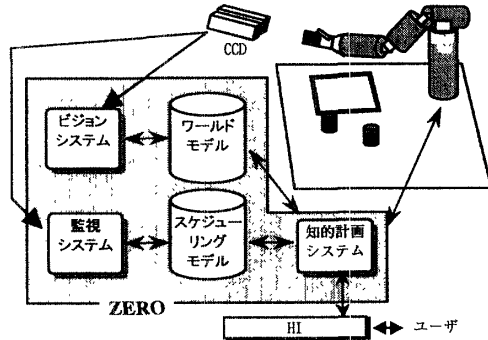


図3：応用システムの例

3. RPCによるZERO++のネットワーク化

ネットワーク上で分散処理を行なう目的は

- 資源の共有
- 並列処理

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なければならない。

具体的には、サーバに現在開いている知識ベースの一覧をもたせておき、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)

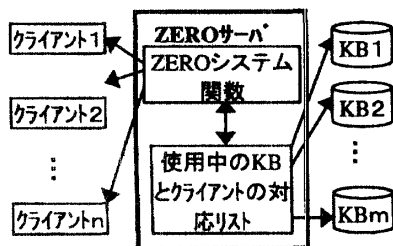


図4：ZEROサーバとクライアント(インタフェースも含む)との接続図

- 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する方法であるが、ここでもRPCを利用する。

これは、RPCでは、アプリケーションがしっかりモジュール化されていれば、モジュールのインタフェース部分を書き換えるだけで実装できる、

という利点から判断した。ただし、いくらかの制限として、ブロックを回避する仕組みが必要となる。これは、次のような方法を検討している。

1) 古典的な方法

UNIXでブロックしてしまう処理を行なうときにいつも利用されてきたもので、子プロセスを利用する方法である。

サーバサービスを利用する時、専用のプロセスをforkによって作成し、そのプロセスがRPCを利用してサーバにアクセスするというものである。

2) タイムアウトを使う方法

単方向RPCと呼ばれるタイムアウトを利用したブロック回避の方法である。

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++による実装方法を示した。

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。

次に、ネットワークへの対応に関しては、現在、設計段階であるが、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。知識ベースについても、今回は共有について論じたが、分散共有も盛り込んでいくべきと考えている。

さらに、今後は、ロボット制御のための機能として、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、PC化版の計画として、ユーザのサポート、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、PC(DOS/Vマシン)への移植も予定している。

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。