ソフトウェア工学 100－7
（1994．9．8）

C++ による汎用フレーム型知識工学環境 ZERO の実現

小暮慎一　　大森康正　　上野晴樹

東京電機大学　理工学部

〒350-03 埼玉県比企郡鳩山町石坂

あらまし

本システムは，従来のZEROの汎用フレーム型知識工学環境としての長所はそのままで，高速化を図る事を目的にしている。フレームとC++のオブジェクトは，類似点があるが操作対象が異なる為，C++のオブジェクトをそのままフレームの表現に用いる事は出来ない。本稿では，C++のクラスをテンプレートとして利用する事によって，フレームや ISA 階層，推論制御を管理する方法について述べる。

和文キーワード　フレームシステム，知識ベースシステム，知識表現

Implementation of Frame-based Knowledge Engineering Environment ZERO in C++

Shinichi Kogure　Yasumasa Oomori　Haruki Ueno

Tokyo Denki University

Abstruct

This report discusses the implementation of the ZERO system which is a frame-based knowledge engineering environment,in C++. The aim of the implementation is to increase the processing speed. Although tthere are similarities between frames in ZERO and objects in C++,but operating methods on subjects are different each other. Therefore,objects in C++ can not be used in the frame expressions. In this paper, we describe the way of management methods for frames,ISA hierarchies and reasoning controls by usage of C++'s classes as temprates.

英文キーワード　Frame-based system,Knowledge-based system,Knowledge representation

1　はじめに

ZEROは，我々が既に開発した汎用フレーム型知識工学環境であり，Common Lispによって実現され，INTELLITUTORや対象モデルをはじめとした様々な応用に用いられている[6][7]．現在，ZEROは，Sun，News，NTT ELIS，XEROX 1121等のワークステーション上，及びFACOM M380等汎用コンピュータ上で稼働している[1][2][3][4]．

エキスパートシステム(ES)の技術が1970年代に出現し，これまでに多くのES構築ツール及びESが開発されてきた．従来，ES構築ツールについては，LOOPS[8]やKEE[9]などのLispベースのハイブリッド型のものが主であったが，最近はCやC++などの環境での開発で処理速度の向上を図ろうという試みが多く行われている．また，オブジェクト指向型のES構築ツールも多く開発されている．しかしながら，フレームシステムについては，Lisp中心である．

本論文におけるシステム（以下，ZERO++と略す）の実現は，従来のZEROの汎用フレーム型知識工学環境としての長所はそのままで，高速化を図る事を目的にしている．また，推論制御を行う為の付加手続きの記述は，従来のLispに加えて，CやC++で記述する事が出来る為に実用性が高まる．また，これによりユーザは従来と同様の操作方法でシステムを構築する事が出来，かつ高速な推論システムの開発を行うことが可能となる．

一方，フレームとC++ におけるオブジェクトの間には，フレームがクラス及びインスタンスを操作対象とするのに対して，C++ のオブジェクトはインスタンスのみを操作対象とするといった相違点がある．したがって，C++ のオブジェクトをそのままフレームの表現に用いる事は出来ない．我々は，C++ のクラスをテンプレートとして用いることでフレームシステムを実現した．なお，本システムはSun ワークステーション上で，Xウインドウ（X11R5）の環境でSun C++を用いて実現されている．

2　汎用フレーム型知識工学環境ZEROの概要　[1][2][3][4]

2.1　ZEROの位置付け

ZEROを用いて知識型システムを構築するには，ZEROによって直接知識型システムを構築する方法と，ZEROを用いて応用モデルを開発しそれによって知識型システムを構築する方法の二通りが考えられるが,汎用な研究開発用ツールとして後者のアプローチをとることを前提に開発されている．

2.2　システム構成

ZEROは，以下により構成される（図1）．

Knowledge Base Editor(KBE),
Knowledge Base Checker(KBC),
Activator(AVT),
SystemFunctions（システム関数群），
Knowledge Base Manager(KBM),
Prolog interpriter, Production System

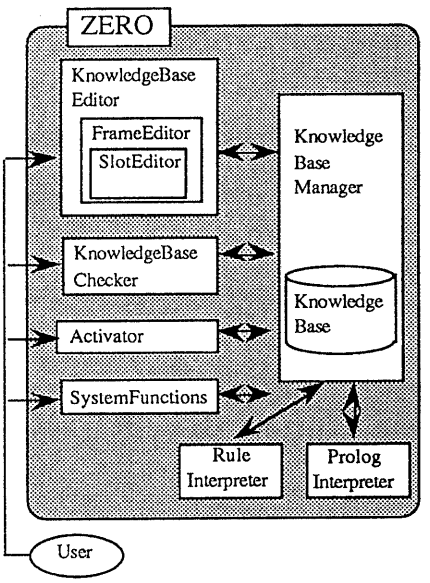

図1:ZEROのシステム構成

KBEは，知識ベースの構築・修正等を行う為のユーザ・インタフェースであり，推論制御の為のプログラムもこのインタフェースにより記述される．KBCは，構築された知識ベースの規約上のミスを検出する為に用いられる．部分的なミスはKBEでも検出されるので，これは主に全体的な矛盾などの検出を行う．AVTは，推論制御のトリガーとなるユーティリティであり，システム関数群は，推論制御の為のattac-hed proce-dures等を書き易くする為のLisp関数群である．KBMは，知識ベースの管理を行う．Prolog inte-rpreterは，attached clausesに記述された述語表現の解釈，ルール・インタプリタは，attached rulesとして記述されたルール表現を解釈する為の機構である（但し，効率等の問題の為，ワークステーション上のZEROに，現在はPrologを組み込んでいない）．

2.3 知識表現

ZEROの知識ベースは，複数のフレームから構成され，各フレームは，抽象－具体(a-kind-of)関係で階層化されるフレームシステムを構成する．

各フレームは，概念対象を記述したものであり，その知識ベースにおいてユニークな名前を持ち，複数のスロットから構成される．個々のフレームの具体的情報は，そのスロットに記述される．

スロットは，全てのフレームが共通に持つシステム・スロットとユーザ定義による可変長のユーザ・スロットとがある．

各フレームに付加されるタイプとしてクラス(class)及びインスタンス(instance)があり，クラス・フレームは抽象的概念を，インスタンス・フレームは具体的な事象あるい概念対象を記述する．フレームを構成する各スロットは，複数のフィールドから構成され，スロットの属性が記述される（図2）．

2.4 インヘリタンス

フレーム型システムの基本的な機能としてインヘリタンス（属性継承）の管理機能を持っている．

開発当初は，インヘリタンス管理の種類が複数存在していたが，現在では，最も標準的なものの一種に整理されている．このインヘリタンスは，上位フレームのスロットの値は下位のフレームにそのまま継承されるが，階層の途中で例外が生じた場合には，その例外値がそれ以後継承される（overrideと呼ばれる）ものである．

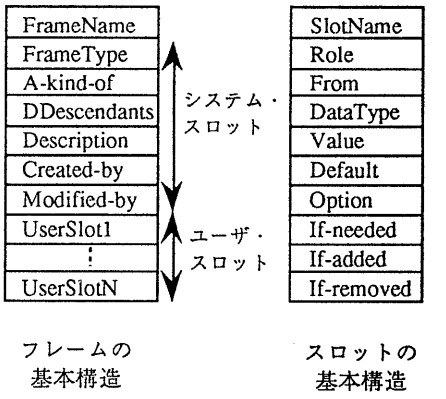

図2:ZEROの知識表現

2.5 推論制御

ZEROでは，推論メカニズムの設計において，フレームに付加された手続きが中心的役割を果たしている．これが，柔軟な推論メカニズムをフレーム・モデルを用いて実現することが可能であると考えられる理由の１つである．

ZEROには特定の推論制御機構を持たないが，ユーザが目的に応じて自由に推論機構を設計出来，フレーム間のメッセージ送信機構により柔軟で強力な推論機構が可能である．しかし，その他に，単なる知識あるいはデータ管理用の手段として，他のモデルに対する補助的ツールとする事も可能である．

3 ZERO++の実現

3.1 背景と目的

本研究の背景は，主に次の2つである．

●対象モデルの概念に基づく知識システムの開発をZEROで行って来たが，実用面から大幅な高速化が求められている．

●自律型ロボットシステムの開発に実用性（C, C++によるプログラミング）と知識処理（フレーム記述）の結合が必要となっている．

以上のような背景から，従来の汎用フレーム型知識工学環境ZEROの長所はそのままに，高速化を図る事を目的として，ZERO++が開発される事になった．

3.2 フレームとC++のオブジェクトとの違い

フレームとC++ におけるオブジェクトの比較を表1に示す．構造や推論制御の方法でいくつかの類似点が見られるが，フレームがクラス及びインスタンスを操作対象としているのに対し，C++はインスタンスのみを操作対象としている．例えば，フレームシステムではクラス・フレームにメッセージ送信が可能であるが，C++では，生成したインスタンス・オブジェクトのみが可能である．したがって，C++ のオブジェクトをそのままフレームの表現として用いることは出来ない．この違いは，フレームが知識表現と推論制御の為のパラダイムとして設計されているのに対し，C++がプログラミング言語として設計されている事によると考えられる．

そこで，我々は，C++のクラス・オブジェクトを用いたシステム構成やフレーム表現，ISA階層を管理する為のテンプレートを用意している．このテンプレートを基に生成したインスタンス・オブジェクトにより，ZERO++を実現している．

表1:フレームとC++のオブジェクトの比較

	ZEROのフレーム	C++のオブジェクト
表現方法	●クラス・フレーム ●インスタンス・フレーム	●クラス・オブジェクト ●インスタンス・オブジェクト
構造	●スロット ●付加手続き	●メンバ（データメンバ，メンバ関数）
階層	ISA	ISA
推論制御	●メッセージ送信 ●インヘリタンス ●デーモン	●メッセージ送信 ●インヘリタンス
操作対象	●クラス・フレーム ●インスタンス・フレーム	●インスタンス・オブジェクト
情報隠蔽	持たない	持つ
ルール記述	可能	不可能

3.3 ZERO++のシステム構成

C++のクラス・オブジェクトによるテンプレートで各モジュールを表現し，インスタンス・オブジェクトを生成して，システムを実現している．各クラス・オブジェクトは，各モジュールの実現に必要なデータメンバ及びメンバ関数を持っている．Editorモジュールについては，FrameEditorとSlotEditorに分けて管理している（図3）．

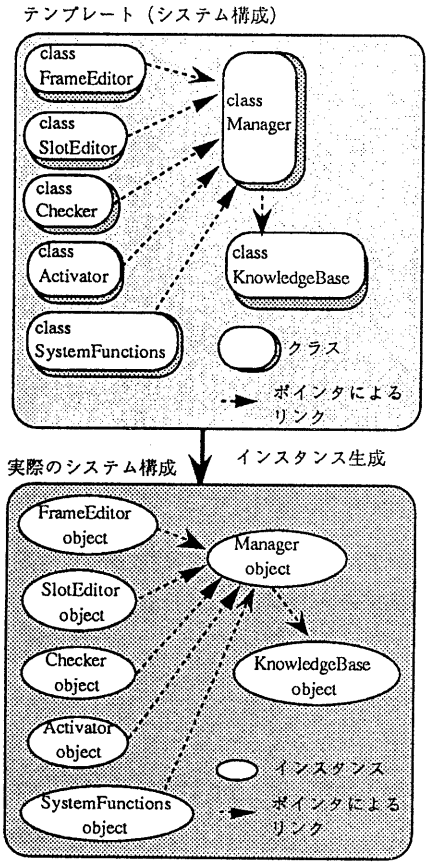

図3:ZERO++のシステム構成

3.4 ZERO++の知識管理

3.4.1 フレーム管理

C++ のクラス・オブジェクトを用いてZERO (ZERO++) におけるフレーム管理の為のテンプレートを用意している（図4）．フレームは，このテンプレートを基にインスタンス・オブジェクトを作成して管理している．フレームは複数のスロットにより構成されるが，システムが予め持っているシステムスロットと，ユーザにより定義される可変なユーザスロットとを分けて管理している．そして，複数のフレームから構成されるZEROの知識ベースは，フレームの追加や削除などを行い，複数のフレームを管理するオブジェクトを用意する事により，知識ベースの管理を行っている．

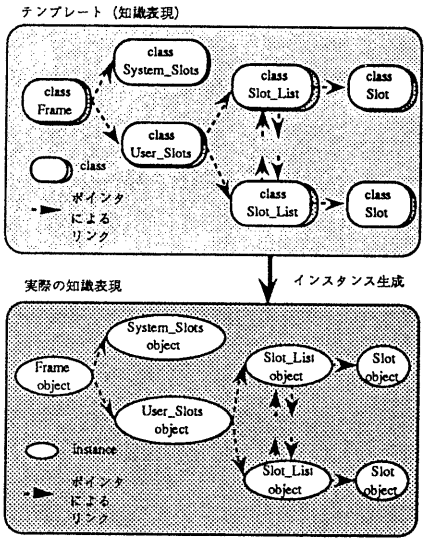

図4:C++によるZEROのフレーム管理

3.4.2 スロット値の管理

スロットは，属性値を記述する複数のフィールドにより構成されるが，その中でオブジェクト指向言語の特徴である派生クラスや多態 (poly-morphism)を用いたスロットの値（Valueフィールド）及びデフォルト値（Defaultフィールド）の管理を行っている（図5）．スロットの値及びデフォルト値は整数型や文字列，フレームなど様々なデータ型を持つ事が出来る．多態により，クラスDataTypeを通じて各データ型にアクセスすることが出来るといった動的結合を実現している．スロットの値及びデフォルト値にDataType型の値をもつことで，値及びデフォルト値がどのようなデータ型も持てることになる．この管理方法により，DataType型のオブジ

ェクトへメッセージを送ることにより，データの操作を始めとして文字列への変換や文字列からの変換などが可能になっている．なお，実際には，図6のテンプレートを基に生成したインスタンス・オブジェクトに対して操作を行う．

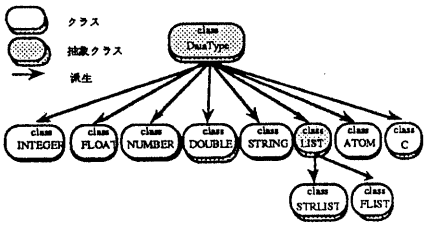

図5:スロット値の管理の為のテンプレート

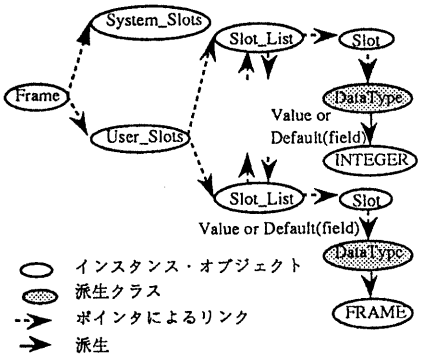

図6:スロット値の管理方法

3.4.3 インヘリタンスの管理

インヘリタンスは，図7に示すように継承するスロットへポインタによるリンクにより実現している．

継承スロットに上書きがなされた場合には，継承スロットを複製し，複製したスロットに上書きを行う事で対応している．

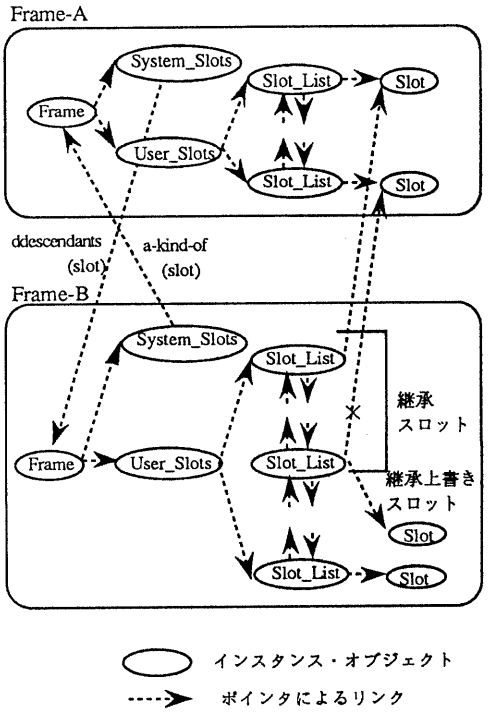

図7:インヘリタンスの管理

3.5 付加手続きの管理

従来のZEROは，推論メカニズムの設計において，中心的役割を果たしている付加手続きの記述に，データ構造が柔軟であり，システムのインプリメント言語であるLispで行っていた．ZERO++では，従来のLispによる記述に加えて，CやC++で記述出来るようになっている．これにより，実用性の向上が実現される．

ZERO++における付加手続きの管理については，コンパイラ処理系であるCやC++で記述した付加手続きについては，付加手続きをコンパイルする作業が必要となる．

また，付加手続きを起動する為に，従来のLispのZEROのおいては，付加手続き名（文字列）から付加手続きを起動する事が出来たが，

ZERO++（Cの処理系）では，付加手続き名から付加手続きを直接起動する事は出来ない．そこで，文字列で与えられる付加手続き名から付加手続きを起動する仕組みを作成する事で対応している（図8）．なお，IF-THENルールの記述も許す予定であり，これにより実用性と記述容易性はかなり向上する．

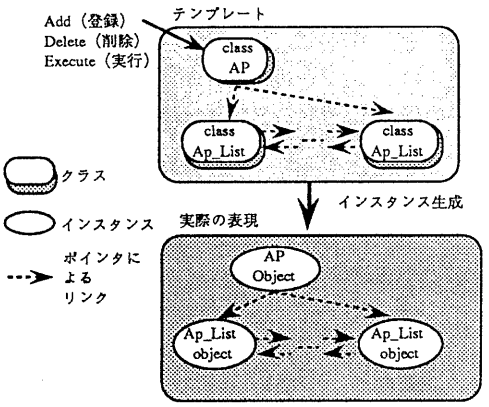

図8:付加手続きの管理

3.6 C++によるZEROのフレームの操作

3.6.1 フレームの作成

フレームの作成は次の順序で行う．

- ●作成するフレームの名前が，知識ベース内で重複していないかどうかを調べる（知識ベースを管理するオブジェクトに，定義されているかどうかを調べる為のメッセージを送る）．
- ●フレーム管理の為のテンプレートを基に，インスタンス・オブジェクトを生成する．
- ●親フレームのユーザスロットを継承する（親フレームのユーザスロット全てに対してポインタによるリンクを張る）．

3.6.2 フレームの削除

フレームの削除は次の順序で行う．

- ●作成するフレームの名前が，知識ベース内で重複しているかどうかを調べる．
- ●削除するフレームにおいて定義されたユーザスロットは，削除するフレームの子フレーム以下のフレームから削除する．
- ●削除するフレームにおいて上書きされたユーザスロットは，上書き前の上位フレームのユーザスロットに置き換える．

ZERO++の管理方法をまとめると，図9のようになる．

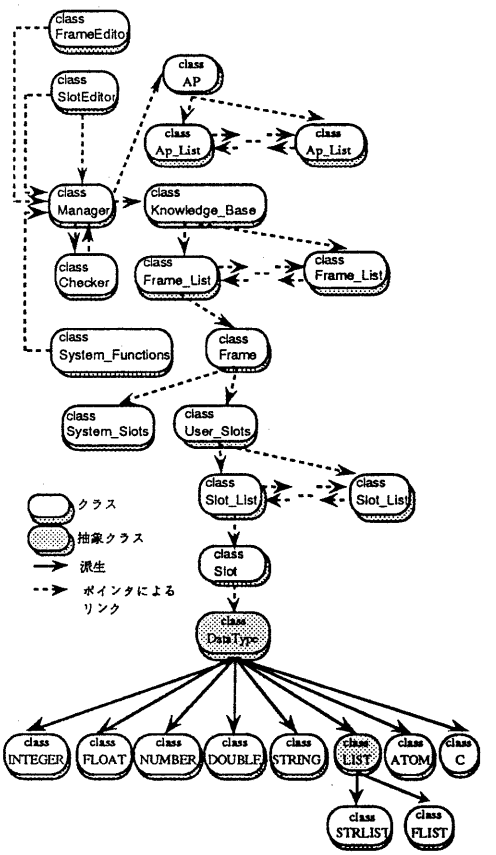

図9:C++によるZEROの全体的なシステム構成

3.7 ユーザ・インタフェース

ZEROは，フレームの階層構造表示やタイピングの量を減らす為の機能，各モード（モジュール）がインタラクティブに作用する，という事を基本方針として，ユーザのシステム構築を支援出来るように，設計及び実現されている．ZERO++のユーザ・インタフェースは，従来の基本方針を受け継ぎ，X-Window（X11R5，アテナウィジェット）によって実現されている．

4 評価

ZEROを用いた応用であるINTELLITUTOR，の知識ベースを用いた知識ベースのloadやsaveの時間を比較した結果を表2に示す．

表2:ZEROとZERO++の比較

	ZERO	ZERO++
ALPUS(load)	156.514 [sec]	6.801[sec]
ALPUS(save)	968.165 [sec]	4.317 [sec]
TUTOR(load)	115.208 [sec]	10.391 [sec]
TUTOR(save)	763.101 [sec]	1.822 [sec]
対象モデル(load)	1.158 [sec]	0.890 [sec]
対象モデル(save)	7.410[sec]	1.504 [sec]

5 おわりに

C++ による汎用フレーム型知識工学環境ZERO(ZERO++)の実現について述べた．C++のクラス・オブジェクトによるテンプレートにより，フレームを管理し，CやC++を用いた付加手続きの記述が可能となったことで，汎用フレーム型知識工学環境としての長所はそのままで，処理の高速化と実用性の向上が実現された．今後は，本システムの機能充実や高速化の検討，ユーザ・インタフェースの改良を行う．現在は，INTELLITUTOR（リバースエンジニアリング），自律型ロボット開発などのプロジェクトに試用を開始したところである．

謝辞

本研究を進めるにあたり，様々な面で御協力頂いた橋本庄太君をはじめとして上野研究室の皆様に感謝を致します．

参考文献

[1] 今井健志，伊藤浩之，吉村貞徳，上野晴樹：汎用フレーム・システムZERO －その概要とユーザ・インタフェースについて－，電子通信学会技術研究報告AI87-22，1987

[2] 伊藤秀昭，上野晴樹：フレーム型知識表現システムZEROにおける付加手続きとしてのProlog，人工知能学会誌，vol.3，No.3，1988

[3] 伊藤秀昭，上野晴樹：フレーム型知識表現言語FMSの構造について，情報処理学会知識工学と人工知能30-4，1983

[4] 伊藤秀昭，上野晴樹：フレーム型知識表現言語の設計及びインプリメンテーションについて，東京電機大学理工学部紀要，Vol.5，1983

[5] 上野晴樹：知識工学入門，オーム社

[6] 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

[7] 上野晴樹：対象モデルに基づく知識表現について－深層知識システムへのアプローチ－；電子通信学会技術研究報告AI86-5，1986

[8] Stefik, M.,D.G. Bobrow,S. Mittal,and L. Conway:Knowledge Programming in LOOPS.,AI Magazine,Vol 4.,No.3,3:14,1983

[9] 高橋清一：KEE，Computer Today；サイエンス社，No.1，1986