人工知能学会誌 Vol. 4 No. 1 (Jan. 1989)〔別刷〕

論 文 Technical Paper

マルチパラダイム型知識処理言語における対立概念の融合方式

Amalgamation of Conflicting Concepts for Knowledge Processing Language Supporting Multiparadigms

中所 武司* Takeshi Chusho
増位 庄一* Shooichi Masui
芳賀 博英* Hirohide Haga
吉浦 裕* Hiroshi Yoshiura

* (株)日立製作所システム開発研究所
Systems Development Laboratory, Hitachi Ltd., Kawasaki 215, Japan.

1988年6月17日 受理

Keywords: knowledge representation, multiparadigms, object-oriented, production system, predicate logic.

Summary

A language supporting multiparadigms is required for development of expert systems because various models of inference processes and various representation of knowledge are necessary to describe expertise as naturally as experts use. In the language design, however, it is difficult to amalgamate conflicting concepts among paradigms without metamorphosing pure semantics of these paradigms.

This paper presents a solution of this problem. First, a two-layer model is introduced for describing expertise. In the upper layer, a global structure of a system is described in an object-oriented paradigm. In the lower layer, expertise is modularized and represented in such paradigms as frame, production system and predicate logic. This two-layer model is effective to keep pure semantics of each paradigm.

Secondly, conflicting concepts between an object-oriented paradigm and other paradigms are amalgamated as follows:

(1) For introducing a frame paradigm, slots and methods are equalized. They become visible and are inherited to descendant objects only by declaring as public.

(2) For introducing a production system paradigm, a cooperative inference system with a data-driven model and a message-driven model is built while limiting data-driven inference to a virtual sub-world called a meeting room.

(3) For introducing a predicate logic paradigm, closed world assumption is kept by inheriting a group of methods written in Prolog as a closed Prolog database.

This language was developed as an expert system building tool (ES/X90) and runs on Hitachi workstation 2050. It has already been applied to development of several expert systems.

1. まえがき

社会組織の高度化に伴い，さまざまな分野で種々の専門家が重要な役割を担っているが，このような専門家の育成は容易ではなく，優れた専門家の数は限られている．そこで，計算機をこれまでのような定形業務ばかりでなく，専門知識と問題解決ノウハウを必要とする非定形業務へ応用するエキスパートシステムの開発が活発になっている．当初は，専門家の知識の抽出と推論過程のモデル化が比較的容易な産業，医療分野が中心であったが，最近では，多種多様な専門家が活躍するビジネス分野へ展開されつつある．

その中から幾つかのエキスパートシステムが実用化

され始めているが，プロトタイプの段階のシステムが多い．特に，知識の抽出が十分には行えない複雑な問題や複数の専門家が協力して解決しなければならない問題については，対象範囲を限定したプロトタイプでの成功がそのまま実用版にはつながっていない．このようなプロトタイプから実用への壁を乗り越えるためには，次の2点が重要である．

(1) 専門家の推論過程を素直にモデル化する．

(2) 専門家の知識をそのモデルの中で素直に表現する．

そのため，エキスパートシステム構築用の知識処理言語としては，柔軟な計算モデルと豊富な知識表現機能を備えたマルチパラダイム型言語が望ましい．この種の言語としては，プロダクションシステムとフレームを融合したものが多いが，上記(1)の機能が弱い．一方，柔軟な計算モデルであるオブジェクト指向概念[1]を含んだマルチパラダイム型言語としては，Lispを基本言語とした TAO[2]，LOOPS[3]，OPHELIA[4]や Prologを基本言語とした ESP[5]，鏡[6]などがあるが，上記(2)の機能が必ずしも十分とは言えない．しかしながら，いかに多くのパラダイムを融合しても，その言語仕様が「多機能，即，複雑」となっては実用的ではない．

そこで，我々は先に知識処理の全体的な記述にオブジェクト指向概念を用い，個々の知識の記述には種々の表現形式を用いる2階層モデル[7]を提案した．本論文では，この2階層モデルに基づいて開発したエキスパートシステム構築ツール ES/X90[8] (Expert system tool for 90's) の知識処理言語における，オブジェクト指向パラダイムとフレーム，プロダクションルール，述語論理の各パラダイムとの間の対立概念の融合方式について述べる．

2．マルチパラダイム型言語の設計思想

2・1 推論過程のモデル化

エキスパートシステム構築の最大の課題は知識獲得であるとよく言われるが，専門家からの知識の抽出を容易にするためには，まずはじめに専門家の実際の推論過程（問題解決プロセス）に合った推論モデルを作成する必要がある．この推論モデルの重要性は，もう一つの本質的課題である知識検証の観点からも言える．エキスパートシステムの検証技術が未熟な現段階での安全で確実な方法は，推論過程を把握可能にしておくことである．そのためには，問題領域の知識のほかに，その知識の利用の仕方を規定する推論制御に関する知識が必要である．このような意味で，推論モデルも一つの計算モデルであると言える．

そこで，我々は，大規模システム向きの分散協調型計算モデルとして柔軟性のあるオブジェクト指向パラダイムを推論制御のマクロなモデルとして採用した．そして，プロダクションシステムに代表される，もう一つの推論制御モデルであるデータ駆動型計算モデルをオブジェクト指向の基本的枠組の中で実現することにした．

2・2 知識表現のマルチパラダイム化

エキスパートシステム構築に際して，専門家からの知識の抽出を容易にするためには，推論過程のモデル化とともに，その推論過程の各段階で用いる知識がそのモデルの中で素直に表現できることが重要である．そのためには，専門家の知識の多様性に応じた複数の表現形式（パラダイム）が必要である．しかしながら，一つの知識が幾つかの表現形式で記述可能になるような冗長性は，言語の複雑化を招くことになる．

我々は，各パラダイムはお互いに相補的な役割を果たすべきであるという考えから，以下の3種類を必要最小限の表現形式として採用した．

① フレーム

② ルール

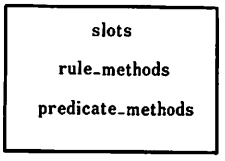
③ 述語論理

フレームは知識の構造化と階層化による知識管理に適する．ルールは，ノウハウ的な知識を if-then 形式で簡単に記述できる．述語論理は，事実や論理的規則などの宣言的知識と手順やアルゴリズムなどの手続き的知識を同一形式で簡潔に記述できる．この3種類の組合せの狙いは，事実的知識をフレーム，規則的知識をルールで表現することを基本とする一方，この単純な枠組では素直に表現できない手続き的知識などを述語論理で表現することである．

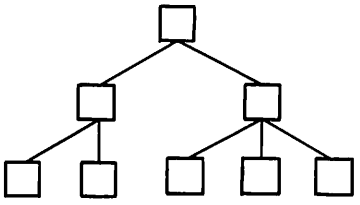
2・3 言語の基本構造

全体的な知識処理の記述は，オブジェクト指向パラダイムを基本とし，個々の知識はフレーム，ルール，述語論理の各表現形式を組み合せて記述するという2階層モデルに基づいて，知識処理言語を設計した．

知識表現の基本単位は，Fig. 1 (a) のように，スロット，ルールメソッド，述語メソッドからなるオブジェクトとする．スロットはフレームのスロットとオブジェクト指向言語の変数を兼ねたものである．ルールメソッドはプロダクションシステムのルール群に対応する．述語メソッドは Prolog プログラムである．このオブジェクトは，Fig. 1 (b) のような階層構造を構成し，継承機能を有する．



(a) Object



(b) Object hierarchy

Fig. 1 Basic structure of knowledge representation for ES/X90.

```
class <Class name> ;
  inherit <Parent class name> ;
  <Database connection> ;
    class_part {
      <External procedure name> ;
      <Slot definition> ;
      <Method definition> ; };
    instance_part {
      <External procedure name> ;
      <Slot definition> ;
      <Method definition> ; };
end .
```

Fig. 2 Syntactic structure for object definition.

具体的な言語仕様は次の方針で設定し，「多機能，即，複雑」とならないように心掛けた．

(1) パラダイム相互の境界を明確にし，個々のパラダイムをできるだけ純粋な形で導入する．

(2) 複数のパラダイムを必要としないシステムの記述には，単一パラダイム型言語として使用可能とする．

このような方針の下で，マルチパラダイム型言語に単一言語としての仕様の統一性を持たせるためには，パラダイム間の対立する概念の融合方式が重要である．

3．オブジェクト指向とフレームの融合

3・1 類似概念の融合

オブジェクト指向パラダイムは，計算モデルあるいはプログラミング技法として，フレームは記憶モデルあるいは知識表現技法として発展してきた．両者の目的は異なっていたが，類似概念は多い．

まず，フレームをオブジェクトの一種とみなし，基本単位をオブジェクトで統一した．さらに，オブジェクトの階層構造と継承機能を導入し，多重継承も可能とした．オブジェクト指向言語の変数はフレームのスロットの一種とみなし，スロットで統一した．

次に，手続きに関して，オブジェクト指向言語のメソッドは外部からのメッセージの受信によって起動され，フレームのデーモン（付加手続き）はスロットへのアクセスに付随して自動的に起動される．両者は矛盾しないので両方ともに可能とした．なお，手続き定義はメソッドで統一した．

最後に，一つのオブジェクトをひな型（クラス）として，同じ機能を持つオブジェクトの実体（インスタンス）を複数個生成する機能は，オブジェクト指向言語では一般的である．フレームにも類似の概念があるので本機能を導入した．

以上の結果から，オブジェクト定義の基本的構文をFig. 2のように定めた．class-partにはそのクラスオブジェクトに固有の知識，instance-partにはそのクラスオブジェクトから生成されたインスタンスが持つべき知識が記述される．なお，本論文では言及しないが，Fig. 2の「データベース結合指定」は，既存のデータベースと本システムのオブジェクトの間の自動変換指定，「外部手続き宣言」は，既存の手続き型言語プログラムの呼出し指定である．

3・2 対立概念の融合

オブジェクト指向とフレームは，メソッドとスロットのどちらをより主要な概念とみるかについて大きな違いがある．オブジェクト指向では，データ抽象化や情報隠ぺいの考えに基づき，外部からメソッド名は見えるが，スロット名（変数名）は見えない．オブジェクトを代表するメソッドは継承の対象となるが，スロットは関連するメソッドの継承に付随して継承されるだけである．一方，フレームでは，スロットが中心なので，スロット名は外部から見えるし，継承の対象もスロットである．

そこで，このような対立概念に関して，言語設計の観点から次の項目の仕様を決める必要がある．

① スロットとメソッドの名前の有効範囲とアクセス法

② スロットとメソッドの継承規則

〔1〕 **オブジェクト指向を重視した方式**

オブジェクト指向を重視した仕様は次のようになる．

① オブジェクトの外部からはメソッドだけが見える．

② 継承の対象はメソッドだけとする（付随的なスロットの継承は生じる）．

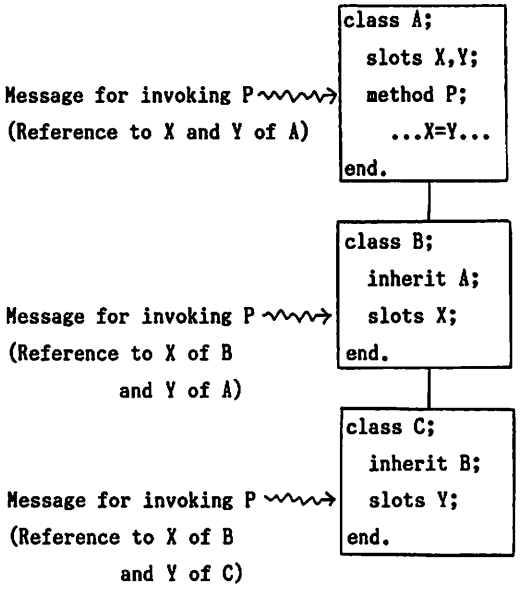


Fig. 3 Cross-reference of slots in object hierarchy.

この方式では，スロットへの制約が強すぎるため，フレームシステムを作成できない．

〔2〕 **フレームを重視した方式**

フレームを重視した方式として，メソッドをスロットの一種とみなし，両者を対等に扱う仕様が考えられる．

① 両者ともにオブジェクトの外部から見える．

② 両者ともに継承の対象とする．

この方式では，スロットが外部から見えるためにオブジェクト指向の大きな特徴である情報隠ぺいの利点が失われること，およびスロットとメソッドの継承が独立に行われるために，それらの定義参照関係を静的に決定できないことから，プログラム全体の理解容易性と保守性が損なわれる．

例えば，3個のオブジェクトA，B，CがFig. 3のような階層関係にある場合を考える．継承規則に関して，同じ名前のものが下位オブジェクトで再定義された場合は，上位オブジェクトのものは継承されないという一般的規則に従えば，BはAからYとPを継承し，CはAからはPを，BからはXを継承する．そこで，Aで定義されているメソッドPの中で参照しているスロットXとYは，AのXとYのように見えるが，実際には，Pの起動メッセージがBに送信されたときはBのXとAのY，またCに送信されたときはBのXとCのYを参照することになる．このように継承機能がテキストマクロ機能に近い働きをするため，プログラムの信頼性と生産性に与える悪影響は大きい．

〔3〕 **融 合 方 式**

オブジェクト指向とフレームの間でのスロットとメソッドの概念の対立は用途の違いによるものである．そこで，その用途をユーザが明示する方式を考えた．

① スロットとメソッドの外部アクセス可否の明示

② スロットとメソッドの継承可否の明示

この方式では，オブジェクト指向プログラミングのときはメソッドだけを外部アクセス可，継承可とし，フレームシステム作成時は，その逆の指定を行えばよい．

しかしながら，スロットやメソッドの定義ごとに2種類の指定を行う仕様は複雑である．そこで，2種類の指定の組合せは用途ごとに可と可または不可と不可の場合が多いことに着目し，継承の可否は外部アクセス可否で兼ねることにした．すなわち，上記②を次のように改める．

②′ スロットおよびメソッドは，外部アクセス可のものだけを継承可とする．

結局，言語仕様上は，外部アクセスを可とするスロットとメソッドはその定義時に「公開宣言」をすることとした．信頼性の面から「非公開」をデフォルトとした．

この融合方式に基づく具体的構文について，Fig. 4の記述例を用いて以下に説明する．

(1) **公開スロット宣言**

Fig. 4 (a) の会員オブジェクトに示すように，スロットのファセット属性としてvisibleという可視属性を指定することにより，公開スロット宣言を行う．

(2) **スロットアクセス法**

Fig. 4 (b) の述語メソッドに示すように，スロットの値の参照，代入，追加，削除を行うときは，各々に対応した組込み述語 (read-value, write-value, add-value, delete-value) を用いて次の形式で記述する．

組込み述語名 (スロット指定，項)

スロット指定は，対象とするスロットが外部のものか，上位オブジェクトから継承したものか，自分のオブジェクト内で定義したものかに応じて次の形式で記述する．

① オブジェクト名#スロット名

② self#スロット名

③ #スロット名

なお，頻繁に使用される参照と代入については，スロット指定のみの直接参照や :=オペレータによる代入などの簡略構文を設けた．Fig. 4の入会メソッドは次のように簡潔に記述できる．

```
入会 (Name, Address) :-# 会員数<# 定員,
        # 会員数 :=# 会員数 + 1,
        会員名簿登録 (Name, Address) ;
```

```
class 会員 ;
 class_part {
  slots (会員名簿(visible)); };
 instance_part {
  slots (名前(visible),
         住所(visible)); };
end.
```

(a) A "member" object

```
class 会員登録 ;
 inherit 業務 ;
 instance_part {
  slots (会員数 (initial(0), on_read(会員数チェック)),
         定員 (default(100))); };
 predicate_methods (入会);
  入会(Name,Address) :- read_value(#会員数,X),
                        X < #定員,
                        Y is X + 1,
                        write_value(#会員数, Y),
                        会員名簿登録(Name,Address);
  入会(Name,Address) :- write("入会出来ません"),
  会員名簿登録(N,A) :- send(会員,create(N)),
                       add_value(会員#会員名簿,N),
                       write_value(N#名前,N),
                       write_value(N#住所,A);
 rule_methods ;
   rules 会員数チェック forward(once);
     if #会員数 = 0 then write('会員未登録です');
     if #会員数 >= #定員 then write('満員です'); };
end.
```

(b) A "member registration" object

Fig. 4 An example of knowledge representation in multi-paradigms.

(3) 公開メソッド宣言

公開メソッド宣言は，述語メソッドおよびルールメソッドの定義の先頭のキーワード predicate-methods および rule-methods の直後にメソッド名を記述して行う．Fig. 4 (b) の例では，入会は公開メソッド，会員名簿登録と会員数チェックは非公開メソッドである．

(4) メソッド起動法

メソッドの起動は，対象とするメソッドが外部のものか，上から継承したものか，自分のオブジェクト内で定義したものかに応じて，次の形式で行う．

① send (オブジェクト名，メソッド呼出し)
② send (self, メソッド呼出し)
③ メソッド呼出し

ここで send はメッセージ送信用の組込み述語であるが，:オペレータによる簡略構文も設けた．Fig. 4 (b) の send (会員, create (N)) は次のように記述してもよい．

会員 : create (N)

なお，Fig. 4 (b) の会員数スロットの on-read ファセットは3・1節で述べたデーモンの例である．

3・3 融合方式に関する考察

この融合方式は，オブジェクト指向およびフレームの典型的使用法を前提にしているため，その前提に合わない次のような使い方を意図したときに不都合が生じる．

① スロットとメソッドの公開かつ継承不可の指定
② スロットとメソッドの非公開かつ継承可の指定
③ 定義参照関係にあるスロットとメソッドの公開

まず，①については，例外事項の記述に必要なので，親オブジェクトを指定する inherit 文に上位からのスロットおよびメソッドの継承拒否機能，またスロット定義のファセット属性に下位への継承拒否指定を設けた．次に，②については，「公開，即，継承可」の原則に沿って公開宣言せざるを得ない．この場合，ユーザは不法な外部アクセスを発生させない注意がいる．

最後の③は3・2節〔2〕項の Fig. 3 の問題である．これはスロットのアクセス法を適切に選択すればよい．メソッドPの中で参照しているスロットが常にAのXとYを意味する場合は，「X＝Y」と記述する．一方，Fig. 3 で示したように動的に決定する場合は，「self#X=self#Y」と記述すればよい．

4．オブジェクト指向とプロダクションシステムの融合

4・1 類似概念の融合

プロダクションシステムは，認知モデルあるいはデータ駆動型計算モデルとして発展してきたため，オブジェクト指向のメッセージ駆動型計算モデルとは大きな相違があるが，まず言語としての類似概念について述べる．

プロダクションシステムでの知識の最小単位であるルールは，原則的には個々に独立したものなので，これを手続きの最小単位とみなしてメソッドに対応させ，ルール群をオブジェクトに対応させるのが自然である．しかしながら，実際には関連するルールの集まりによって一つの機能を果たす場合が多いので，このようなルール群をメソッドに対応させ，ルールメソッドと呼ぶ．オブジェクトはルールメソッド群から成る知識源に対応する．

次に，グローバルなデータエリアの役割を持つワーキングメモリについては，固定的なデータ構造を必要

とする場合はスロットに対応させる．その他の場合は後で導入する黒板を対応させる．

4・2 対立概念の融合

オブジェクト指向のメッセージ駆動型モデルでは，スロットは外部から見えないのに対し，プロダクションシステムのデータ駆動型モデルでは，ワーキングメモリに対応するスロットはそれを参照するルールメソッドから見えていなければならない．そこで，このような対立概念に関して，言語設計の観点から次の項目の仕様を決める必要がある．

① メッセージ駆動型計算モデルの枠組の中でのデータ駆動型計算モデル（データ駆動型推論）の実現

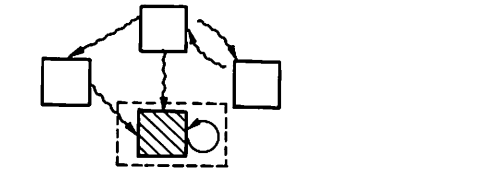
② ルールメソッド用スロットの名前の有効範囲

〔1〕 **オブジェクト指向を重視した方式**

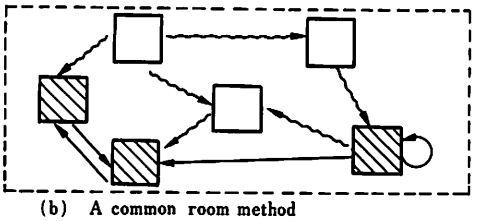
オブジェクト指向を重視した仕様は次のようになる．

① 一つのオブジェクト内でデータ駆動型推論を許す．

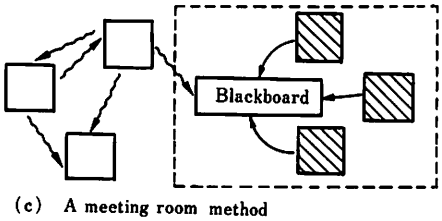
② そのオブジェクト内で定義されるか，あるいは上位から継承したルールメソッドとスロットだけがデータ駆動型推論に参加できる．



(a) A private room method



(b) A common room method



(c) A meeting room method

[Notation]

An object without production rules

An object with production rules

A production system area

Message passing

References to slots

Fig. 5 Three methods to embed production system into object-oriented system.

この方式は，OPHELIA[4]の方式に近いが，オブジェクト指向の枠組をくずさないで，特定の領域に限定してデータ駆動型推論を可能にしているので，Fig. 5 (a) に示すように小部屋方式と呼ぶ．この方式では，ルールの管理や制御を静的に決めるため，段階的にルール数が増加していくようなシステム構築には適さない．

〔2〕 **プロダクションシステムを重視した方式**

大規模なプロダクションシステムの構築には，オブジェクトをルール群の管理単位とする方式が考えられる．

① データ駆動型推論の範囲を限定しない．

② すべてのオブジェクトのルールメソッドとスロットがデータ駆動型推論に参加できる．

この方式は，Fig. 5 (b) に示すように，全体の知識処理が行われているオブジェクト指向の世界と全く同じ世界でデータ駆動型推論を可能としているので，大部屋方式と呼ぶ．これは，必然的にスロットへの外部からのアクセスを可能とするため，オブジェクト指向の情報隠ぺいや分散協調による利点が損なわれる．

〔3〕 **融 合 方 式**

小部屋方式および大部屋方式の欠点を回避した融合方式を次のような考えに基づいて設定した．

「多数のオブジェクトから成るメッセージ駆動型計算モデルの中で，データ駆動型推論の必要が生じたときは，そのために必要なルールメソッドの集合から成る小世界でのみそれを可能とする．」

この方式は，あたかも会議室に必要なオブジェクトを集めて，会議室内では自由に発言したり，どのデータを参照してもよいことにしているので，会議室方式と呼ぶ．Fig. 5 (c) に示すように，会議室の中はプロダクションシステムであるが，外に出ればオブジェクト指向の世界である．

この会議室方式を実現するため，黒板による制御方式を導入し，次のような言語仕様とした．

① 黒板に記入された起動条件キーワードと同じものを持つルールメソッドがデータ駆動型推論に参加する．

② データ駆動型推論に用いるスロットを明示する．

まず，①については，Fig. 6 の例のように，ルールメソッド定義の先頭で on (住宅) という起動条件キーワードを指定しておき，データ駆動型推論の開始時に黒板に“住宅”が記入されていれば，この起動条件を持つすべてのルールメソッドを起動する．このとき，②で述べたように，ルール条件部で参照するスロットを明示する必要があるが，この機能は，外部からのアクセスを許すことを意味するので，公開スロット宣言

```
class 住宅評価 ;
 inherit 評価;
 instance_part {
  rule_methods ;
    rules 中古住宅評価 on(住宅) ;
      if 報告書#建築年数 >= 15        then +(中古の度合, 0.8) ;
      if 15 > 報告書#建築年数 >= 5    then +(中古の度合, 0.5) ;
      if 報告書#工法 = 木造           then +(中古の度合, 0.2) ;
      if 報告書#集中暖房設備 = 無     then +(中古の度合, 0.1) ;
      if 報告書#屋根付き駐車場 = 無   then +(中古の度合, 0.1) ;
 };
end.
```

Fig. 6 An example of a rule method for a production system.

で兼ねることにした.

本機能の実現のために，黒板への起動条件キーワード（アジェンダ）の記入と削除のために write-ag と delete-ag，データ駆動型推論の開始と終了のために bb-mode (on) と bb-mode (off) という組込み述語を設けた．なお，プロダクションシステムに固有の機能は本論文では述べない.

4・3 融合方式に関する考察

〔1〕 協調型推論システムの構成法

この会議室方式は，基本的には小部屋方式と大部屋方式を包含しており，柔軟なシステム構成が可能である．これは，データ駆動型推論の範囲を限定する会議室の概念が仮想的なものであることによるので，ユーザの使い方次第でシステム構成が複雑になる危険性もある．したがって，システムの理解容易性や保守性の観点からは，データ駆動型推論に用いるオブジェクトの集合を知識ベース内のオブジェクト階層の中で部分木に対応させておくのがよい.

その具体例として，不動産評価エキスパートシステムのシステム構成について述べる．本システムは，Fig. 7 に示すように，各担当者へ指示，依頼する業務をメッセージ駆動型オブジェクトに対応させ，会議に参加して遂行する業務をデータ駆動型オブジェクトに対応させている．調査オブジェクトが黒板に“住宅”というキーワードを記入し，黒板を on 状態にすると，“住宅”を起動条件とするルールメソッドを持つ都市計画予測，住宅地評価，住宅評価の各オブジェクトが参加してデータ駆動型推論が実行される．このシステム構成を Fig. 8 に示す．データ駆動型推論に参加する評価業務関係のオブジェクトおよびそのデータとなる不動産関係のオブジェクトがオブジェクト階層の部分木を構成している.

〔2〕 逐次型ルールメソッド

この会議室方式に限らず，一般にプロダクションシステムは，ルール条件部判定，競合解消処理，ルール行動部実行の繰返し処理を行うため，ルール数が多くなると，条件部判定に時間がかかる，適切な競合解消戦略の選択が難しい，推論結果の検証と不良箇所の検

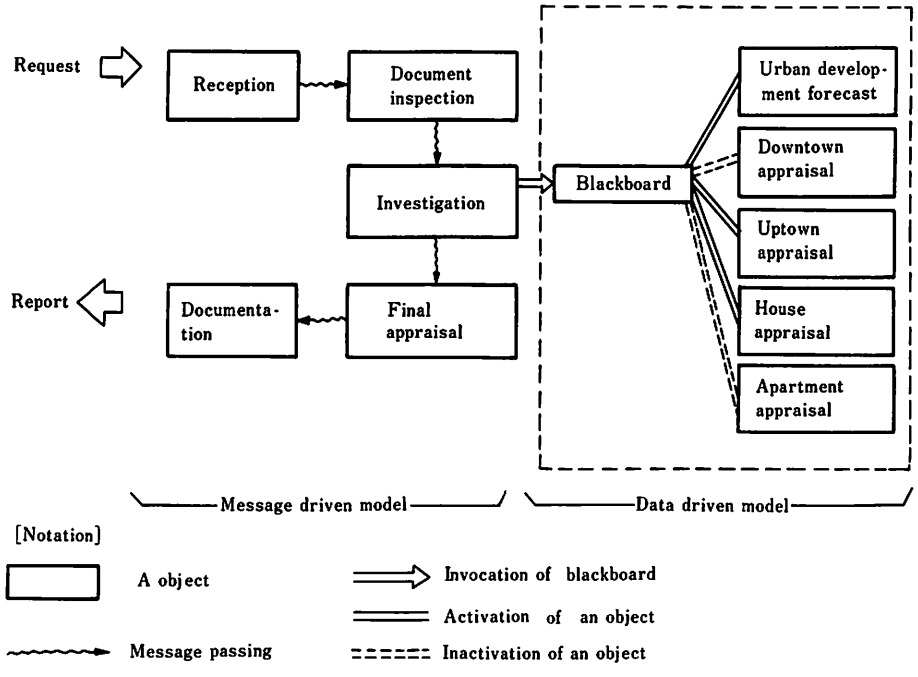
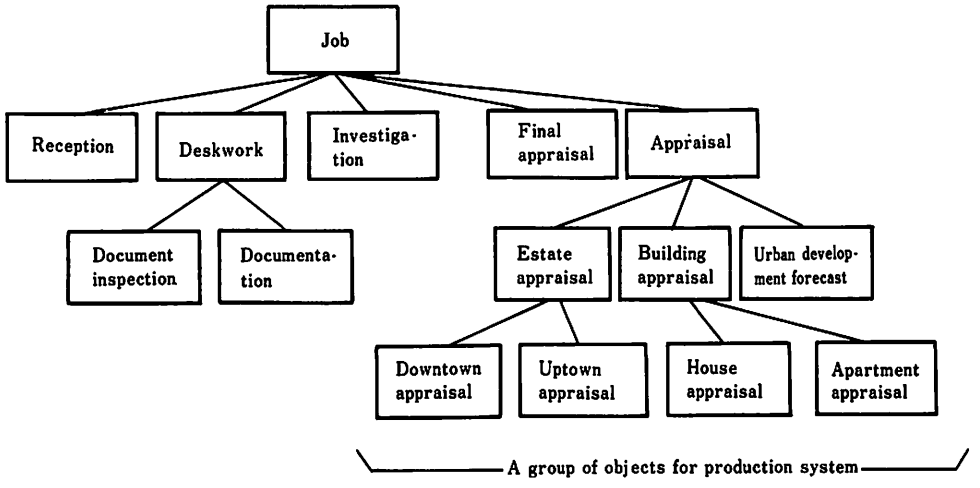
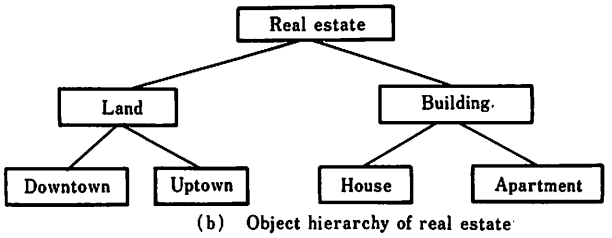


Fig. 7 A cooperative inference system with a message-driven model and a data-driven model in an estate appraisal expert system.



(a) Object hierarchy of jobs



(b) Object hierarchy of real estate

Fig. 8 Object hierarchies of an estate appraisal system.

出が難しいなどの問題がある．そこで，本機能の使用を必要最小限に抑えるために，ルールを記述順に実行する逐次型ルールメソッドを新たに導入した．これは，ルール形式の知識表現が持つ簡易プログラミングとしての特徴に着目したもので，次のような仕様とした．

① メッセージによって起動される．

② メソッド内のルールは記述順に実行され，前向き推論と後向き推論の指定ができる．

特に前向き推論については，ルール群を1回だけ実行するときは once 指定，繰返し実行は multi 指定，さらに1回の実行において条件を満足するルールをすべて実行するときは all 指定ができるようにして，簡易プログラミング言語としての機能を充実させた．Fig. 4 の会員数チェックは逐次型ルールメソッドの例で，once 型の前向き推論が指定されている．

5．オブジェクト指向と述語論理の融合

5・1 類似概念の融合

ここでは，代表的な述語論理型言語である Prolog を取り上げる．オブジェクト指向の分散協調型問題解決モデルに対し，Prolog は論理学を基礎とした三段論法による問題解決モデルを与えている．両者の間には異質な概念が多いが，まず，言語としての類似概念について述べる．

Prolog では，節集合からなるプログラム全体が一つのデータベースとして扱われ，モジュールの概念はない．そこで，このデータベースをオブジェクトに対応させる．手続きに相当するものは，Prolog では節であるが，実際には節の左辺の述語名の同じものが幾つか集まって一つの機能を表す場合が多いので，この節集合をメソッドに対応させ，述語メソッドと呼ぶ．Fig. 4 の例では，入会や会員名簿登録が述語メソッドである．その起動法は，3章で述べたように，公開メソッドは，

send (会員登録，入会 (中村，横浜市))

と記述し，非公開メソッドは通常の Prolog と同じく，

会員名簿登録 (中村，横浜市)

と記述する．

5・2 対立概念の融合

オブジェクト指向と Prolog の間の異質な概念の融合に関する課題は，言語設計の観点では次の2項目にまとめられる．

① 述語メソッドの名前の有効範囲
② バックトラッキング機能の制約条件

Prologでは，節集合の全体から成るデータベースに含まれる知識のみが真であるという前提がある．この閉世界仮説の観点からオブジェクトの階層構造と継承機能をどう解釈するかが上記①の課題である．

さらに，Prologのバックトラッキング機能はオブジェクト指向における変数の副作用および並列実行の概念に合わない．Prologの変数は，一つの節の中だけで有効であり，単一代入の原則もあるため，バックトラッキング処理が容易であるが，オブジェクトのスロット変数はオブジェクトの状態変数として用いられ，自由に値が変えられるため，バックトラッキング処理が複雑になる．この問題は，オブジェクト指向の並列実行性の下ではさらに複雑化し，いわゆる分散バックトラッキングが生じる．そこで，バックトラッキング機能にどのような制約を設けてこの問題を回避するかが上記②の課題である．

〔1〕 **オブジェクト指向を重視した方式**

オブジェクト指向を重視した仕様は次のようになる．

① 各オブジェクトの述語メソッドデータベースは，自分の中で定義したものと上位から継承したものを含む．

② バックトラッキングは同じ述語メソッドデータベース内でのみ可能とし，オブジェクト指向用組込み述語は常に成功とみなし，副作用も回復しない．

上記①は，ESP[5]の方式に近いが，Fig. 9 (a) のように自分の中で定義した述語メソッド群の後に上位から継承した述語メソッドを追加して実行するため，モジュール性が低下する．例えば，次のような親子関係の二つのオブジェクトを考える．

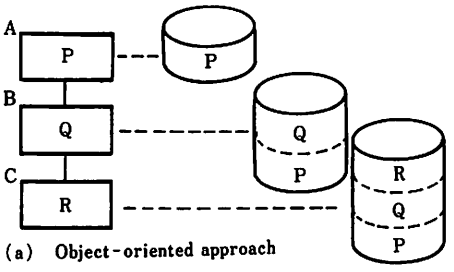
```
class A ;
  predicate-methods ; p :- … q …
                      q :- …          end.
class B ; inherit A ;
  predicate-methods ; q :- …          end.
```

述語メソッドpが起動されたとき，その起動メッセージがAに送られたときはAのq，Bのときは Bのqが実行されることになってしまう．さらに，上記②の方式では，他のオブジェクトの述語メソッドを利用した縦型探索が行えないなど，述語メソッドの使用上の制約が強すぎる．

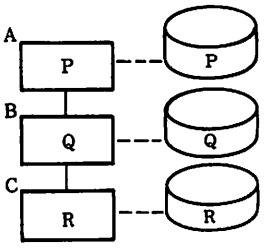
〔2〕 **述語論理を重視した方式**

述語論理を重視した仕様は次のようになる．

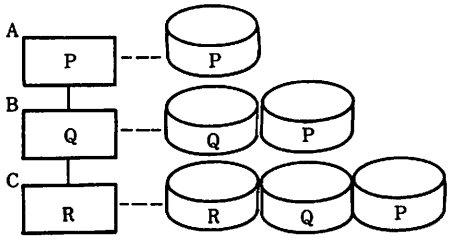
① 各オブジェクトの述語メソッドデータベースは，自分の中で定義したものだけを含む．



(a) Object-oriented approach



(b) Logic oriented approach



(c) Mixed approach

[Notation]

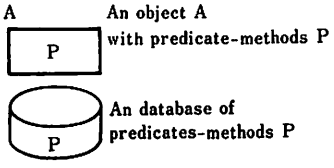


Fig. 9 A scope rule and an inheritance rule of predicate-methods.

② バックトラッキングはプログラム全体を対象とし，オブジェクト指向用組込み述語の副作用を回復する．

上記①は，Fig. 9 (b) にも示すように，オブジェクト指向やフレームに特徴的な継承機能を妨げている．さらに上記②では，スロットの値の変更やインスタンスの生成，削除の履歴を管理する必要が生じ，実行系が複雑になる．例えば，Fig. 4の入会メソッドを呼び出す入会受付メソッドが次のように定義されていたとする．

入会受付 (N, A) :- send (会員登録，入会 (N, A))，確認．

このとき，最後の確認メソッドの失敗に備えて，入会メソッドの実行では，会員数スロットの変更や会員オブジェクトのインスタンス生成を元に戻す準備が必要である．

〔3〕 融 合 方 式

上記2案の欠点を補う融合方式を次のようにした．

① 各オブジェクトは，自分の中で定義した述語メソッドだけから成るデータベースと上位から継承したデータベースを別々に保持し，下位優先で適用する．

② バックトラッキングはプログラム全体を対象とするが，オブジェクト指向用組込み述語の副作用は回復しない．

上記①は，Fig. 9 (c) に示すように，Prologの閉世界仮説とオブジェクト指向の継承機能を両立させる方式である．5・2節〔1〕項で述べた例に関して，この方式では，オブジェクトBが述語メソッドpの起動メッセージを受信したときBの中にpの定義がないので，Aから継承した述語メソッド群 {p, q} のデータベースを用いて実行する．したがって，このときにBのqが実行されることはなく，Aのqが実行される．Bのqを実行したいときは，2章で述べた規則に沿って，Bのqを公開メソッドとするとともに，pの定義内のqの呼出しを send (self, q) または self : q と記述すればよい．

次に上記②は，Prologのバックトラッキング機能を保存する一方，オブジェクト間にわたるマクロレベルでのオブジェクト指向パラダイムの意味仕様と実現方式の複雑化を避けた方式である．これに関連して以下の仕様も設定した．

(1) メッセージは逐次的に実行されるものとし，分散バックトラッキング問題を回避した．

(2) 述語メソッドはその実行結果として成功または失敗を返すが，ルールメソッドの実行は常に成功とみなす．

5・3 融合方式に関する考察

本方式はオブジェクト指向用組込み述語の実行による副作用を無視したバックトラッキングを行うため，複雑な動きをするプログラムが作成される可能性がある．この問題を避けるには，述語メソッドを次のいずれかの目的に合った使い方をする必要がある．

① バックトラッキング機能による後向き推論を行う．

② 通常の手続き的処理を行う．

すなわち，①の場合は，副作用の生じる組込み述語を用いないか，または副作用が回復されなくてもよい範囲で用いる．②の場合は，バックトラッキング機能を併用しないか，または局所的な探索処理に限定して用いる．

6. むすび

マルチパラダイム型言語の設計においては，その言語が与える計算モデルとそれに対応する言語仕様が複雑にならないように，異なるパラダイム間の対立概念を融合することが重要である．本論文では，オブジェクト指向を全体の基本的枠組とする2階層モデルを用いて，オブジェクト指向とフレーム，プロダクションシステム，述語論理との間の対立概念を融合した．

本言語を組み込んだエキスパートシステム構築ツールES/X90は，言語処理系と推論実行系をProlog，その他をCで記述しており，日立のワークステーション2050上で稼働する．これまでに，計算機システム構成設計支援システムなど，幾つかのエキスパートシステム開発に適用し，本システムの特徴である2階層モデル，協調型推論，マルチパラダイム型知識表現などの有効性を確かめた．

謝 辞

最後に本研究の機会を与えていただいた(株)日立製作所システム開発研究所の堂免信義所長，石原孝一郎部長，有益な御討論をいただいた同社ソフトウェア工場の高橋栄部長(工学博士)，開発に御協力いただいた同社システム開発研究所の古賀明彦，増石哲也ほかの関係各位に感謝する．

◇参 考 文 献◇

(1) 米澤：オブジェクト指向型プログラミングについて，コンピュータソフトウェア，Vol. 1, No. 1, pp. 29-41 (1984).

(2) Takeuchi, I., *et al.* : TAO-a Harmonic Mean of Lisp, Prolog and Smalltalk, *ACM SIGPLAN Notices*, Vol. 18, No. 7, pp. 65-74 (1983).

(3) Bobrow, D. G. and Stefic, M. : The LOOPS Manual, Xerox Co. (1983).

(4) 安西，近藤：階層構造を持つルール・ベース型表現を埋め込んだオブジェクト指向型知識表現言語 OPHELIA とその応用，コンピュータソフトウェア，Vol. 3, No. 3, pp. 43-60 (1986).

(5) Chikayama, T. : ESP Reference Manual, Technical

Report TR-044, ICOT (1984).
(6) 溝口，本田，片山：オブジェクト指向概念を導入した知識表現言語：姿と鏡の設計とその応用，Proc. Logic Programming Conf. '84, 2.1 (1983).
(7) Chusho, T. and Haga, H.: A Multilingual modular Programming System for Describing Knowledge Information Processing System, 10th World Computer Congress IFIP '86, pp. 903-908 (1986).
(8) 中所，ほか：エキスパートシステム構築ツール ES/X 90 (1)－(9)，情報処理学会第35回全国大会論文集，pp. 1733-1750 (1987).

〔担当編集委員・査読者：石塚　満〕

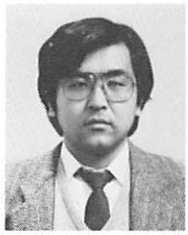
著者紹介

中所　武司（正会員）

1969年東京大学工学部電子工学科卒業．1971年同大学院修士課程修了．工学博士（東京大学）．同年（株）日立製作所入社，現在，システム開発研究所主任研究員，東京工業大学非常勤講師．1983年度情報処理学会論文賞，1986年度大河内記念技術賞受賞．知識工学とソフトウェア工学の研究に従事．著書：「人工知能」，「プログラミングツール」（昭晃堂，共著）．情報処理学会，日本ソフトウェア科学会，IEEE Computer Society 各会員．

増位　庄一（正会員）

1972年京都大学工学部電子工学科卒業．1974年京都大学大学院修士課程電気工学第2専攻修了．同年，（株）日立製作所入社．現在，同社システム開発研究所主任研究員．1981年～82年カーネギーメロン大学客員研究員．制御システム，知識工学の研究に従事．著書：「ビジネスマンのための AI 入門」（オーム社，共著）情報処理学会，電気学会，IEEE，AAAI 各会員．計測自動制御学会知識工学部会委員．



芳賀　博英（正会員）

1978年同志社大学工学部電子工学科卒業．1980年同大学院修士課程修了．同年，（株）日立製作所入社．現在，同社システム開発研究所第5部研究員．論理型プログラミング，AI 技術のソフトウェア生産技術への適用の研究に従事．情報処理学会，日本ソフトウェア科学会，The Association for Logic Programming 各会員．

吉浦　裕

1981年東京大学理学部情報科学科卒業．同年，（株）日立製作所入社，日立研究所勤務．1985年より同社システム開発研究所勤務．自然言語処理，CAD，知識工学の研究に従事．情報処理学会，電子情報通信学会各会員．