特集　知的情報処理

計測と制御
Vol. 25, No. 4
(昭和61年4月)

≪解　説≫

知的プログラミング

中所武司*・増位庄一*

1. はじめに

人工知能や第5世代コンピュータの研究の進展と共に，従来の数値処理，データ処理とは異なる知識処理を対象としたコンピュータの新しい応用分野が開かれようとしている．

このような知識工学に対してはおおむね2つの期待がかけられている．本特集の「知的情報処理」という表現を借りて説明するならば，第1には「知的情報の処理」，すなわち従来のコンピュータシステムが苦手としてきた人間の知識処理を模擬することにより専門家の業務の肩代りをするエキスパートシステムの実現がある．第2には「知的な情報処理」，すなわち従来分野の情報処理技術の壁を突破するブレークスルー技術として知識工学が期待されている．

このような状況は本論文の「知的プログラミング」についても同様であり，つぎの2つの側面からの技術動向について述べる．

- 「知的プログラムの作成方法」
- 「知的なプログラム作成方法」

前者はエキスパートシステムに代表される知識情報処理プログラムをどのように開発するかという問題であり，従来のプログラミング技法とは異なる新技術が要求される．後者は，従来分野のプログラム開発において，その手順の定式化とツールによる自動化を主体としたソフトウェア工学的アプローチに基づくソフトウェアの生産性と信頼性向上の限界を打破するためにどのように知識工学を応用するかという問題である．

本論文では，2章，3章でおのおのについて述べ，4章では双方を含んだ制御用応用システムの分野の実用例について述べる．

* (株)日立製作所　システム開発研究所
川崎市麻生区王禅寺 1099

キーワード：知識工学 (knowledge engineering)，知識表現 (knowledge representation)，エキスパートシステム (expert system)，推論 (inference)，プログラミング環境 (programming environment).

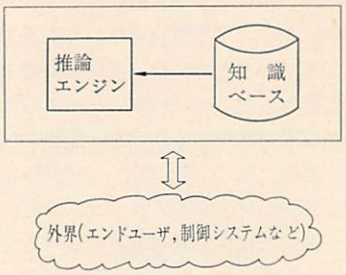

図 1　知的情報処理モデル

2. 知識情報処理プログラムの開発方法

2.1 知識情報処理のモデル

知識情報処理の簡単なモデルは図1のように表わされる．まず「太郎の母は花子である」という事実や「XがYの母で，YがZの親ならばXはZの祖母である」という規則などが知識ベースに貯えられる．

一方，この知識を利用して「太郎の祖母はだれか？」といった問題を解くのが推論エンジンである．基本的な推論方式としては前向き推論と後向き推論がある．今，つぎの2つの規則があったとする．

「AならばBである」
「BならばCである」

ここで，Aという事実が与えられてCの成立を導くのが前向き推論である．この方式は，あるデータがAの状態になったときにBやCの行動を起すようなデータ駆動型システムやある事象Aが生じたときにBやCの行動を起すような事象駆動型システムに利用される．

これとは逆に，後向き推論ではCという目標が与えられたときにその真偽を確かめるために，上記の規則をつぎのように適用していく．

「Cであるためにはであればよい」
「Bであるためにはであればよい」

そしてAが成立していれば目標Cが達成されたことになる．

なお実際の推論エンジンでは種々の工夫がされてい

る[1),2)].

従来の情報処理システムではアルゴリズムを記述した手続きプログラムが主体で，それにデータが付随して機能を果していたが，知識情報処理においては図1に示したように推論制御と知識ベースが完全に分離されており，しかも知識ベースが主要な役割りを果している点に大きな特徴がある．

2.2 知識表現言語

2.2.1 単一形式

知識情報処理は，その核となる知識ベースの中にどのような形で知識を蓄積するかによって大きく影響される．そこでまずその基本となる代表的な表現形式について述べる．なお，(1)〜(3)は文献1),2) に詳しく紹介されている．

(1) プロダクションシステム

エキスパートシステムの分野で比較的早くから用いられてきたのは，人間の認知モデルから導かれたプロダクションシステムである．これは，ある条件が成立したときにとるべき行動を指定する規則や，ある前提となる事実からその帰結として新しい事実を導く規則を「if〜then〜」というルールの形で表現するものである．このような簡潔な表現は，一般にコンピュータには不慣れな人にとっても自然で理解しやすい．この具体例は4章で図4を用いて説明する．

(2) 述語論理

人間の論理的思考の表現に適したものとして述語論理がある．その代表的なものは，日本の第5世代コンピュータプロジェクトの基本言語に選ばれて活発に研究されている Prolog である．その一般形はつぎのようなホーン節と呼ばれる述語の集合である．

$A \longleftarrow A_1, A_2, \cdots A_m$

これは「右辺の A_1 から A_m までのすべての項が成立すれば左辺Aが成立する」という規則を表わす．特に右辺のないものは無条件に成立するという意味で事実を表わす．

Prolog はこのようなホーン節の集合をプログラムとみなし，ユニフィケーションと呼ばれるパターンマッチングとバックトラッキングという探索機構を用いて後向き推論を行う．たとえば2.1節で述べた「太郎の祖母はだれか？」のような問題はこの方式でうまく解ける．

(3) フレーム

フレームの概念は，人間の記憶構造や推論過程の説明のために導入されたものである．先に述べたプロダクションシステムや述語論理では不十分であった知識の構造化や知識間関係の記述に適しているため，多くのエキスパートシステムに利用されている．その基本的構造としては，まず実世界の概念に対応する枠組みをフレームとして定義する．そして，その概念のもつ種々の属性や機能をスロットとしてフレームの中に埋め込む．一方，フレーム間に階層構造を導入し，下位のフレームは上位のフレームの属性や機能を継続できるようにしている．したがって，1つのフレームに対応する概念の実体は，そのフレーム自身のもつスロットの性質とそのフレームが自分の上位フレームから継承する性質の両者によって規定されることになる．このような仕組みのため，フレームを用いた知識表現では，知識の構造化と階層化が容易である．

(4) オブジェクト指向

オブジェクト指向の概念はもともとプログラミング方法論の研究の中で導入され，SIMULA，SMALLTALK-80 などのプログラミング言語に反映されてきたが，結果的にはフレーム型に似た構造を有している．すなわち，まず，実世界の概念に対応するオブジェクトを導入する．オブジェクトはその対応する概念の有する機能を果たすメソッドの集まりであり，必要に応じて内部データを有する．個々のオブジェクトは独立性が強く，相互作用は相手オブジェクトのメソッドを起動するためのメッセージを送信することによって行う．図2にカウンタのオブジェクトをハイブリッド型言語[3] S-LONLI (後述) で記述した例を示す．このオブジェクトはカウンタの値を保持する内部変数

```
define_frame counter ;
    attribute class ;
    supers root ;
    i_vars count (initial(0)) ;
    i_method clear ;
        clear :- set(0) ;
    end ;
    i_method up ;
        up :- New_count is count + 1 ,
              change(count,New_count),
              show ;
    end ;
    {method down もほとんど同じ}
    i_method show ;
        show :- write('Value --> '),
                write(count) ;
    end ;
    i_method set ;
        set(Value) :- change(count,Value),
                      show ;
    end ;
end counter .
```

図2 S-LONLI によるオブジェクト記述例

count とカウンタの機能を果たすためのメソッド群 (up, show, set, clear) から成り，このカウンタへの操作はいずれかのメソッドを指定したメッセージ送信によって行う．

このような基本構造に加えて，オブジェクト間の階層構造，上位オブジェクトから下位オブジェクトへのメソッドの継承，オブジェクト変数へのデフォルト値指定やメソッド呼出し機能を付加することによりフレーム型と同様の知識表現機能を有する．さらに階層構造化された個々のオブジェクトを原型（クラス）として複数の実体（インスタンス）を作り出す機能やインスタンス間のメッセージ送信機能などにより，知識表現と知識処理の柔軟な記述が可能となっている．

2.2.2　ハイブリッド形式

実際のエキスパートシステムでは対象とする知識が多種多様であるため，以上に述べたような単一の表現形式の1つだけを適用するのでは不十分である．そこで最近では複数の表現形式をサポートするハイブリッド型の知識表現言語が数多く提案されている．たとえば，フレームとプロダクションシステムを含んだ KEE[4]，あるいはオブジェクト指向と述語論理を含んだ ESP[5]などがある．しかしながらこのようなハイブリッド型言語では単一言語化のために必要な統一的意味仕様の設定が難しい課題である．われわれの開発したハイブリッド型言語 S-LONLI では，この問題を回避するために，実世界の知識処理モデル（全体の知識構造）の自然な表現に適するオブジェクト指向表現を上層部の記述に用い，規則と事実（個別の知識）の正確な表現に適する述語論理を下層部の記述に用いることにより，両者の境界を明確にするような融合方式[6]を採った．図2の例では，カウンターオブジェクトの各メソッドの本体が述語論理で定義されている．

2.3　開発環境

知識情報処理システムは知識表現言語と推論エンジンがあれば簡単に作れるというものではない．従来の情報処理システムに比べて歴史が浅く，本格的実用システム開発の実績が少ないため，開発技術についても多くの課題がある．特にここでは開発環境の面でのつぎのような課題をとりあげる．

（1）　応用分野の専門家が提示する断片的知識を構造化して知識ベースへ蓄積するための知識獲得技術

（2）　推論結果の妥当性に対する検証技術

（3）　知識ベースの継続的保守技術

第1の点は，効果的かつ効率的な推論を行うためにはその基になる知識がうまく構造化されていなければいけないことから必須の技術である．そのため，知識ベースエディタは，従来のプログラムエディタや構造エディタの機能に加えて，これらの機能のサポートが重要である．

第2の点は，入力と出力の対応関係が明確な情報処理プログラムとは異なり，推論結果の真偽が一意的に判定できないため難しい問題である．実際的には，不十分な推論結果に対しては知識ベースや推論方式の試行錯誤的変更を繰返しながら修正していくことになる．そのため，デバッガは通常の機能に加えて，推論過程の説明機能や知識ベースエディタとの連動機能などが必要である．

第3の点は，知識情報処理システムの開発当初の段階では必要十分な知識が用意されておらず，その後，継続的に追加修正されることに対応するものである．すなわち，まずプロトタイプが開発され，その継続的改良によって徐々に実用版ができていく過程で生じる頻繁な知識ベースの保守の効率が全体の開発効率に大きく影響する．

3.　プログラミング環境の高度化

1970年代のソフトウェアの大規模化に伴う「ソフトウェア危機」に対処するためにソフトウェア工学の研究が重視され，種々のプログラミングツールが開発されてきた[7]．これらのツールは熟練プログラマに使用され，その作業量を削減するのが主目的であった．しかしながら，今後の深刻なプログラマ不足を反映して，非熟練プログラマがプログラムを開発するケースが急増しつつあるため，最近では初心者向きのプログラミングツールが重要になっている．米国の TRW 社の調査[8]では，ツールの良し悪しによる生産性の差が1.49倍なのに対し，個人差は4.18倍であったと報告されているが，この個人差の解消が最大の課題である．

そのためには熟練プログラマの知識を計算機に組込んで初心者に利用させるプログラミングエキスパートシステムが有効であろう[9]．筆者らもその1つとして文法規則というマニュアル的知識を内蔵して段階的詳細化によるプログラミング法を誘導する構造エディタ PARSE[10]を開発，使用している．しかし，現在のプログラミング環境下では，個々のツールや TSS システムに関するノウハウ的知識が多く，熟練者には常識的な事柄でも初心者がマニュアルからだけでは修得できないことが多い．そこで，このようなノウハウ的知識を計算機に埋め込み，初心者の誘導に利用することができれば効果が期待できる．

たとえば，知識工学の応用例として，システムから

出力されたエラーメッセージをもとに周辺の状況（結果）から誤りの原因を特定するような故障診断システム，あるいはユーザが入力するコマンド列の履歴を監視しながら会話方式でシステムの利用方法を誘導するようなコンサルテーションシステムなどが考えられる．

4. システム制御分野での知的プログラミング

4.1 制御エキスパートシステム

知的プログラミングの具体例として，知識工学応用が最も進んでおり，かつ成功をおさめてきたシステム制御[11),12)]の分野を取り上げてみる．この分野での知的プログラム，すなわちエキスパートシステム開発の狙いは，従来の固定的でアルゴリズミックな自動制御システムに対して，専門の運転員のもつ経験知識を導入し人間の高度で柔軟な知的活動の一部を取り入れた制御システムを構築しようとする点にある．これら経験知識は，それが取り扱う問題の構造の不確定性あるいは変更に対する脆弱性から，全体の処理手順の体系化，アルゴリズム化を必要とする伝統的な手続き型プログラミングに馴染まず，制御自動化の過程においても切り捨てられてきたものである．知識工学技術は，これら知識自身をその知識をどのように用いるかの方法（推論）から分離することにより，体系化しにくい知識のプログラム化を可能とする点で注目されており，特にその試行錯誤を前提とするモデリングスタイルがこの種の応用に向いているとされる[13)]．この知識工学技術をシステム制御，大規模システムの運用制御や診断に応用すれば，つぎのような効果が期待できる．

（1） アルゴリズム的知識に加えてエキスパートのノウハウを活用することができ，従来にない高度な判断操作の実現が可能となる．

（2） 知識と推論の分離というソフトアーキテクチュアを基礎としているため，知識の追加や削除が容易で，継続した機能向上が可能である．

（3） ユーザ自身による知識の修正が可能であり，プロトタイプ開発の工数の低減が期待できる（これはシステム完成までの工数低減を必ずしも意味するものではない）．

（4） 推論過程の提示が容易であり，推論に用いた知識の意味，推論の流れの理解が可能で運転員の判断を確実ならしめることができる．

4.2 システム制御における知的プログラミングの要件

経験ノウハウを活用した（制御）エキスパートシステムと従来の制御用プログラムとの間には，その基礎とするモデリングスタイルの相違がある．従来の制御用プログラムは完全なアルゴリズムの存在のもとではじめて組み立て可能となるものであり，開発されたプログラムの実行がソフト開発の目的となるのに対して，エキスパートシステムではプログラムは常に修正され追加削除をうけるべきものであり，その生成過程そのものが実行過程同様に重視される．

このように試行錯誤がプログラミングの基本スタイルになると，それに適した優れたプログラミング環境を伴うことがプログラム開発における必然となり，その優劣がシステムの開発効率を決定するものとなる．すなわち，システム制御における知識の記述に適した知識表現の設定と，その知識表現を計算機に入力するための専用エディタ，検証修正用のデバッガなどいわゆる開発ツールの整備が必要不可欠となる．さらにこれらツールは，従来のような開発環境として独立したものでなく実行環境の一部として組み込まれることになる点も重要な相違点である．これは，プログラムの開発と実行が従来のように分離できないことを意味する．したがって，知的プログラミングのためには統合的環境—開発と実行が同時に可能となる環境—の存在が不可欠でエキスパートシステム構築ツールもこの必要性，機能性を満たすものでなければならない．以下には，われわれが開発したシステム制御用知識処理ソフトウェア EUREKA (Electronic Understanding and Reasoning by Knowledge Activation) の概要について述べ，知的プログラミング環境の例を示す．

4.3 知的処理ソフトウェア EUREKA[14)]

EUREKA は，システム制御のためプロダクションシステムとオブジェクト指向システムを機能的に融合した知識表現体系をもつエキスパートシステム構築ツールである．EUREKA の知識表現は，ルール表現のもつ制御ノウハウ（論理処理）の表現容易性と，手続き型言語の処理高速性，数値処理の容易性を融合させることに狙いをおいて設定してある．具体的にはシステム構成を自然な形で表現できるフレーム型データ構造と手続きプログラム（プロセデュア）を一体化したオブジェクト形式で対象プラントの設備，機器の状態，機能，関連性などを記述し，それらに関する運用制御の方策を，追加削除が容易で記述性の高い IF（条件）-THEN（行動指示）型プロダクションルール形式で記述する．このオブジェクトとルールは，メッセージパッシングの形式で抽象的に結合され，相互の独立性が保たれている．このため構成要素の変更は，その要素に関するオブジェクトのみの変更で，制御方

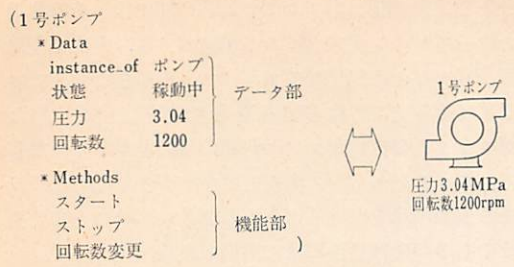

図3　EUREKA のオブジェクト例

```
[ 運転開始]  (揃速運転ルール
IF
  (一号ポンプ    の S回転数 を ?X とする)
  (二号ポンプ    の S回転数 が ?X である)
  (?あるポンプ   の S状態   が 停止 であり
                 S圧力   が 10以上であり
                           40以下である)
THEN
  (~  ?あるポンプ  スタート  (?X)))
```

図4　EUREKA のルール例

策の変更や，推論知識の追加などは，ルールだけの変更で対処でき，制御の水準をきわめて容易に向上できることになり，エキスパートシステムを容易にかつ柔軟に構成できる．

EUREKA のオブジェクトはプロセスを構成する設備，機器（実体）の計算機内のモデルである．図3に示すように，名称（オブジェクト名），実体を表現するデータ（*Data）を属性とその値で記述する．属性は任意で自由に対象に関する記述ができる．実体の機能（*Methods）はプログラム名称で表現する．このプログラムはメソッドと呼ばれ，そのオブジェクトに付随するものとみなされる．プログラム本体は，別途 C, FORTRAN 77 などで記述する．以上のように EUREKA のオブジェクトは，データ抽象化の単位として，プロセスエンジニアが見ている世界をそのままの形で記述することを可能とする．したがって，プロセスシミュレータの構築などもきわめて容易である．

一方，ルールは図4に示すように，条件部（IF 部）に，このルールが実行されるときのオブジェクトの状態を記述し，行動指示部（THEN 部）には，オブジェクトへの機能（メソッド）実行依頼メッセージを記述する．メッセージは EUREKA が自動的に当該オブジェクトの機能プログラムを実行する形で処理される．EUREKA のルール記述の特長は，条件部において変数（?で先行された文字列）記述が可能，属性値の二項比較（=，<，>など）が日本語的表記で可能な点にあり，これがユーザ知識の自然な表現を可能としている．このように EUREKA のルールは，プロセスエンジニアの頭の中のノウハウを表現する手段であるといえ，特にその日本語的記述はメンテナンスなどを容易化するのに役立つ．

EUREKA の構造は図5に示すように，知識ベース（ルールメモリー，オブジェクトメモリー）と統合的

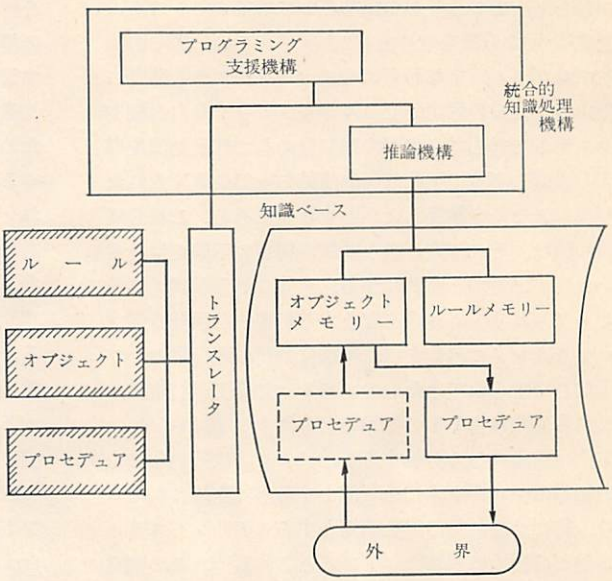

図5　EUREKA のソフトウェア構成

知識処理機構（推論機構，プログラミング支援機構）からなる．EUREKA の基本推論形式は，前向き推論であり，オブジェクトの現況にマッチする条件部をもつルールを選択しその行動指示部を実行する．また EUREKA は，ルールの実行順序を制御するため，ルールを群別けし，その群単位で実行指示ができるメタルール機構をもっており，自由かつ柔軟にルール実行順序や推論深度の制御ができる．またオブジェクトの属性データの書き換え，値参照時にあらかじめ設定したメソッドを自由に起動できるアクティブ属性機能をもち，プラントシミュレータが簡単に構成できる．EUREKA の特長は，強力なプログラミング支援機能[15]にあり，これを通して，知識ベース（ルール，オブジェクト）への任意アクセスや，ルール実行のトレース，実行の中断，再開などができ，容易にプログラムのデバッグや段階的構築が行える．このように推論機構としてプログラミング支援機構が統合的な実行環境として提供されることが従来のプログラム開発にみ

られない知的プログラミングの特徴である．

5. おわりに

コンピュータが世の中に登場しはじめた1950年代からプログラミングの方法は絶えず進歩してきた．最近の十数年を見ても，記述言語はアセンブラから高級言語へ，入力・修正はカードパンチからスクリーンエディタの利用へ，検証は16進ダンプリストからデバッガの利用へ大きく変化してきた．しかしながらコンピュータの処理手順を1ステップずつ正確に定義していくというプログラミングスタイルは旧態依然のままである．そして，この点での変革がない限り，コンピュータの応用範囲は限定され，ソフトウェア生産技術のブレークスルーもありえない．

本稿では，「知的プログラミング」をこのようなプログラミングスタイルの変革を志向するものとしてとらえ，知識情報処理システムを中心に解説を試みた．1980年代に入ってから研究活動は年々活発化しており，近い将来にその成果が期待される．

(昭和61年2月4日受付)

参考文献

1) 小林：知識工学の基礎と応用［第1回］，計測と制御，**24**-2，155/164 (1985)
2) 同上：同上［第2回］，計測と制御，**24**-3，242/250 (1985)
3) 芳賀，中所：知識情報処理システム記述言語S-LONLIの提案，日本ソフトウェア科学会第1回大会，167/170 (1984)
4) R. Fikes et al.: The Role of Frame-Based Representation in Reasoning, CACM, **28**-9, 904/920 (1985)
5) T. Chikayama: ESP Reference Manual, ICOT Technical Report TR 044 (1984)
6) 中所，芳賀：オブジェクト指向型言語と論理型言語の融合方式に関する考察，オブジェクト指向，共立出版，133/146 (1985)
7) 中所：プログラミング言語とその会話型支援環境，情報処理，**24**-6，715/721 (1983)
8) B. Boehm: Software Engineering Economics, Prentice-Hall, Englewood cliffs (1981)
9) E. Rich: The Gradual Expansion of Artificial Intelligence, IEEE Computer, **17**-5, 4/12 (1984)
10) 田中，中所，ほか：言語適応型プログラミング用マンマシンインターフェイスとしての構造エディタPARSE，情報処理学会，ソフトウェア工学研究会資料，**34**-8，43/48 (1984)
11) T. Tashiro et al.: Advanced Software for Constraint Combinatorial Control—Rule-Based Control Software for Factory Automation—, Proc. of Compint' 85 Sept. (1985)
12) 都島，ほか：流れ作業ライン制御へのルール型制御方式の適用—製鉄所のピレット精整ライン制御への適用—，計測自動制御学会論文集，**21**-10，1113/1120 (1985)
13) 辻井：プロダクションシステムとその応用，情報処理，**20**-8，735/743 (1981)
14) 増位，ほか：知識制御核ソフトウェア EUREKA の記述体系，情報処理，29回全国大会論文集 6Q-5 (1984)
15) 増位，ほか：知識処理核ソフトウェア EUREKA によるプラント管理制御のラピッドプロトタイピング，情報処理，30回全国大会論文集 2N-6 (1985)