エキスパートシステム構築支援ソフトウェア ESHELL/Xの紹介

大貝, 彰
九州大学工学部建築学科

桜井, 尚子
九州大学大型計算機センター

https://doi.org/10.15017/1468204

出版情報：九州大学大型計算機センター広報. 22 (3), pp.181-207, 1989-05-25. 九州大学大型計算機センター
バージョン：
権利関係：

エキスパートシステム構築支援ソフトウェア ESHELL／Xの紹介

大貝　彰*，桜井　尚子**

1. はじめに

人工知能（AI；Artificial Intelligence）という言葉が流行り出してから数年を経ている．この知識工学の実践的応用分野の一つとして，現在最も実用化が進んでいるのがエキスパートシステムである．エキスパートシステムとは，専門家がそれぞれの分野で長年培った専門的知識を整理してコンピュータに蓄積し，その知識ベースをもとに論理的推論機構を駆動させて，多様な現実の問題解決に役立てることを意図したシステムである．従って，コンピュータ適用の領域が拡張されるため，従来望み得なかった大規模な問題や危険な作業に対処する可能性が向上することになる．

今回紹介するESHELL／Xは，富士通㈱が開発したエキスパートシステム構築支援のアプリケーションソフトウェアである．本センターでは公開していないが，ESHELL／Xの前身に相当するESHELLも，富士通㈱が製品化しているソフトウェアである．ESHELLとESHELL／Xとの相違点として，ESHELL／Xは，単なる知識表現ツールではなく，知識の統合及び既存システムとの連携を狙った総合的なエキスパートシステム開発環境を提供することを目指す点が挙げられよう．ESHELL／XのESHELLに対する主な特徴は，以下の通りである．

- 図形，イメージ，文書などを含めた幅広い知識の利用が可能である．
- 知識の統合により問題領域のモデル化が容易になる．
- 既存システムとの連携機能が標準提供される．
- ワークステーション版ESHELL／Xの機能により，実行時の利用者インタフェースの構築が容易となる．

図1.1にESHELL／Xのシステム概要を示す．

本稿は，これからESHELL／Xを利用するユーザに対し，ESHELL／Xの概要，使用方法ならびに具体的な使用例について記述するものである．各研究分野での成果拡大に一役を買うことができれば幸いである．

2. ESHELL／Xの特徴

2.1 知識表現の統合

知識の表現方法としてフレーム，ルール，手続きがある．

フレーム　：問題領域に存在する"もの"の関係を体系化して階層的に表現したもの．一般に，関係の明瞭な知識を表現するのに適している．

ルール　：問題領域の断片的な因果関係などを"IF～THEN"形式で表現したもの．一般に，

平成元年3月25日受理

* 九州大学工学部建築学科
** 九州大学大型計算機センター

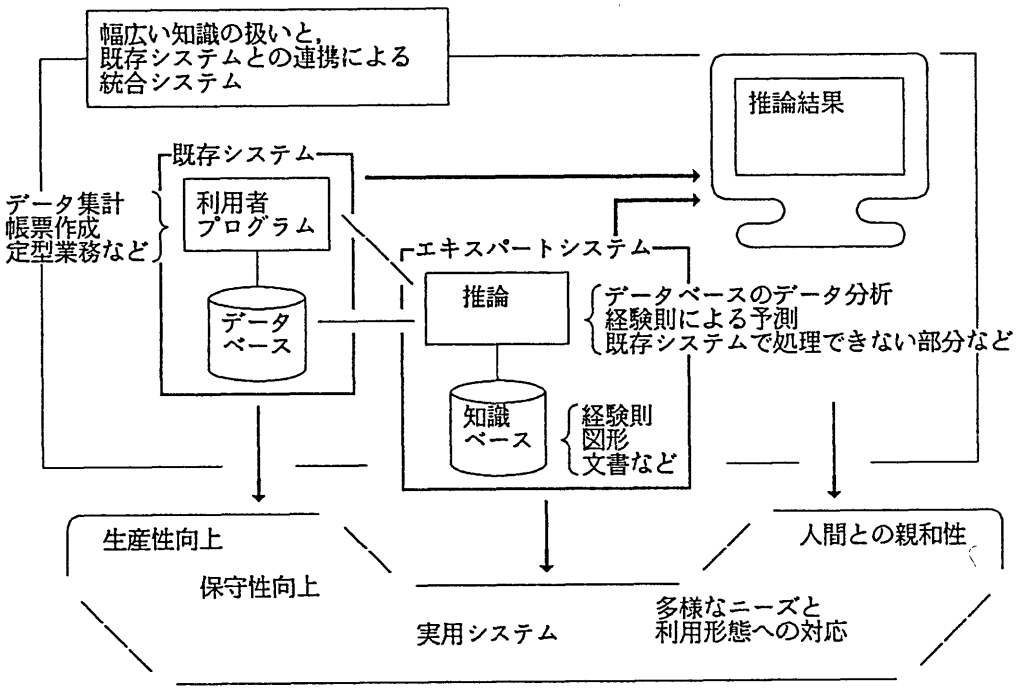


図 1.1　ESHELL／Xのシステム概要

専門家の経験的ノウハウ等の知識を表現するのに適している.

手続き　：定型的な処理手順をプログラムとして表現したもの.

しかし，現実の問題領域では，これらの他に設計書，図面，写真，マニュアル若しくは自然法則，常識といったいろいろなレベルの知識が存在する．これらの知識を一元的に扱うために，このシステムでは，知識オブジェクトという管理単位を用いる．ESHELL／Xの知識の管理単位には，次のものがある.

知識オブジェクト：フレームを主体として，それに付随するルール，手続き及びマルチメディアデータなどを一元的に扱うための管理単位.

ルール集合　：一まとまりのルールの管理単位.

関数　：手続きの管理単位

これにより，次のような効果が期待できる.

1）問題領域のモデル化に伴い，ルールや手続きもフレーム構造に合せてモジュール化できる．また，手続きなどの詳細処理をフレームの中に隠すことができ，見通しの良いモデル構築が可能である．この結果，モデル構築の生産性，保守性が向上する.

2）フレームシステム又はプロダクションシステムとして独立に利用することもでき，多様な利用形態に対応可能である.

2.2　知識表現言語

知識表現言語は，知識オブジェクト，ルール集合などを記述するための外部表現形式であり，次のような特徴がある．

1）入力形式と表示形式が同じであり，外部エディタによる一括編集が行える．知識表現が編集，入力，デバッグ，表示等ですべて同一形式であるため，保守性が向上する．

2）日本語，英語両方での記述が可能．

3）簡略記法により，複雑な記述や重複記述を避け，任意の構文での記述が可能．

2.3　推論制御

推論の制御方式としては，起動方法，方向，進め方等いくつかの要因があり，問題に合せて最適の制御方法を選ぶことが可能である．

起動方法：　ルール集合を待行列に繋ぎ，行列の先頭から順に起動していく方法とサブルーチン的にルール集合を呼出す方法があり，これらを単独又は組合せて使用することも可能である．大きな推論の流れはルール集合の待行列によって制御し，細かい処理はサブルーチン的にルール集合を呼出すといった選択ができる．

方向　：　前向きと後向きがあり，これらを単独又は組合せて使用することができる．最初は，前向きで推論を進め，ある程度の解答が得られたところで，後向きの推論を行い，解答の検証をすることができる．

進め方：　あいまいな仮定のもとに推論を進めて，あいまい性を残した結果を得ることができる．これには，事実やルール成立の確かさを数値で表した確信度と，事実の意味自体のあいまいさを関数で表すファジイがある．

高速性を要求される推論に於ては，黒板（Blackboard－BB）と呼ばれるデータ領域を使用できる．黒板には，推論のための入力データ，推論の途中結果，推論結果を書込むことができる．

2.4　既存システムとの連携

ESHELL/Xでは，既存システム連携機能として，データベース検索やファイル入出力が可能である．また，数値計算など既存のプログラムで処理する部分はそれらを呼出すことができる．これには，次のような特徴がある．

1）データベースやファイルの特徴を知識オブジェクトの中に記憶しているので，構造の詳細を意識せずに一様なインタフェースで利用することができる．

2）アクセス結果は，知識オブジェクトに反映されるので，通常の知識オブジェクトの操作により，容易に参照できる．

2.5　図形処理

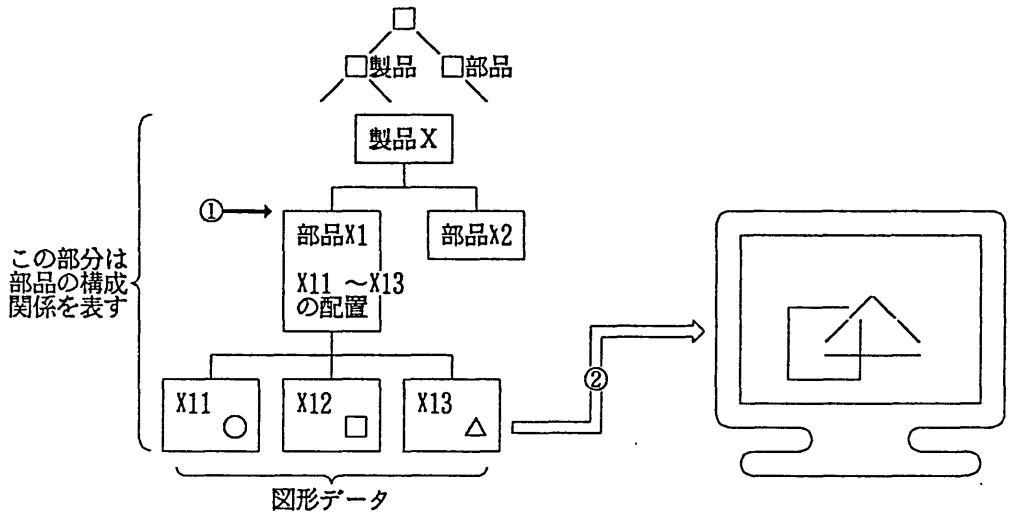
ESHELL/Xでの図形処理には，次のような特徴がある．

1）図形データを知識オブジェクトで管理しているため，表示，移動及び拡大などの図形操作を一

様なインタフェースで容易に行うことができる．

2）図形を構成関係に基づいて階層的に分割し，部分図形を知識オブジェクトで管理しているので，部分図形単位の操作が容易である．

図2.1に図形処理のイメージを示す．



① 部品X1に対して表示する旨の命令を出す．部品X1には，部分図形の情報(配置を含む）がある．

② 各部分図形の図形データが画面に表示される．

図2.1　図形処理のイメージ

2.6　開発環境

ESHELL／Xの開発環境としては，編集・デバッグ等の基本機能に加え，知識ベースの管理機能ならびに印刷機能を提供する保守支援機能がある．

1）知識の編集・デバッグ

知識編集・デバッグの対象は，ファイルではなく，メモリに読込まれた知識オブジェクト，ルール集合及び関数である．次のような特徴を持つ．

■ 編集作業は，知識表現言語の形でフルスクリーンによって行う．

■ デバッグの途中で，推論の中断時などに編集作業が可能である．

2）知識ベースの管理

知識ベースは，複数のファイルから成っており，その管理機能には次のような特徴がある．

■ 知識ベースは複数のファイルに分割管理されるので，大規模な知識ベースを分割して開発することができる．

■ 知識ベースは，ファイル単位でメモリへの読込み及びメモリからの削除が可能なので実行時に必要な部分だけをメモリ上に置くことができる．従って，メモリを有効に使用することができる．

3）知識ベースの保守支援

印刷機能により，知識ベースの内容，知識オブジェクトの関係図及びその参照関連を出力する．

表2.1にESHELL／Xの機能一覧を示す．なお，ESHELL／Xは，UTILISP V03の下で動作するため，知識ベースの一部はLisp言語で記述作成する必要がある．

表2.1 ESHELL／Xの機能一覧

機能			機能概要
知識管理／推論制御	知識管理	メッセージ送信	知識管理の基本インタフェース（メッセージ）の解釈／実行
		継承	知識オブジェクトの階層関係に伴う，性質の引継ぎ
		知識オブジェクト操作	値の参照／設定，知識オブジェクトの生成／削除など
		メソッド起動	知識オブジェクトに付随するルール又は手続きの起動
		デモン起動	知識オブジェクト操作に伴う付加的な手続の起動
	推論制御	ルール集合の実行	前向き又は後向きのルール集合の，推論実行
		制御機構の管理	ルール集合切替えのための，イベント管理
		あいまい性の処理	値の確かさを表す確信度，及び意味のあいまいさを表すファジィ
		黒板	推論を高速に実行するためのデータ領域
	知識表現言語	リーダ	知識オブジェクトやルール集合の外部表現形式を読み込み，内部形式に変換 簡略記法を，通常の表現形式に展開
		プリンタ	知識表現の内部形式を外部表現形式に変換
実行支援	既存システム連携	データベース検索	AIM／RDBのデータベース定義，テーブル定義を知識オブジェクトで管理 データベースの検索結果の，知識オブジェクトへの取り込み
		ファイル入出力	ファイル名，レコード定義など，アクセスに必要な情報を知識オブジェクトで管理 順編成又は区分編成のファイルに対して，データ単位又はレコード単位のアクセス可能
		利用者プログラム呼出し	Lispの外部プログラム呼出し機能により実現
	利用者インタフェース構築（図形処理）		図形データを知識オブジェクトで管理 知識オブジェクトの関係付けにより，部分図形に分割，及びそれらの組合せ 図形操作として，表示，消去，移動，拡大，縮小，ラインプリンタへの図形出力など

機　　能		機　　能　　概　　要
開発支援コマンド	編　　集	主記憶上の知識のフルスクリーンによる修正，編集
	デバッグ	推論実行の追跡，中断．中断点では状態参照，変更が可能
	読込み	知識ベースをファイル単位でメモリへ読込み
	消　　去	知識ベースをファイル単位でメモリから消去
	保　　存	知識ベースをファイル単位で書き戻す
	一覧表示	メモリ中に読み込まれた知識ベースのファイル名一覧 知識オブジェクト，ルール集合，関数の一覧
	印　　刷	メモリ中の知識の要約，関係図，詳細内容などをラインプリンタ又はファイルに出力

3. 知識ベースの記述と実行支援機能

3.1 知識表現

ESHELL/Xによる知識表現には，以下のようなものがある．

- 知識オブジェクトの定義 ： DEFOBJECT
- 手続きの定義 ： DEFUN （Lispの関数定義と同じ）
- ルール集合の定義 ： DEFRULES
- あいまい表現の定義 ： DEFFUZZY
- 簡略記法の定義 ： DEFSYNTAX

この他にも，特殊な知識オブジェクトを定義する記述文があるが，ここでは主要な言語の記述概要を述べる．知識表現にはキーワードとユーザ定義語を使用する．キーワードは英字でも日本語でも記述できる．

3.1.1 知識オブジェクト

知識オブジェクトは，エキスパートシステムが扱う具体的な事物や概念を表現するもので，問題領域の知識モデル（これを世界モデルと呼ぶ）を構成している個々の知識の単位といえる．具体的には，知識の属性，ルールや関数などの手続きを統合管理するもので，複数の属性項目＝スロットによって構成されている．知識オブジェクト定義の一般形式を以下に示す．

```
( DEFOBJECT    知識オブジェクト名
    SYSTEM         システムスロットの並び……
                 [ LEVEL（必須），SUPER-CLASS，SUPER-LEVEL ]
    PROPERTY       属性スロットの並び……
    INITIAL        初期化属性スロットの並び……
    METHOD         メソッドスロットの並び……
)
```

(1) システムスロット

一般的に問題領域の世界モデルは，複数の知識オブジェクトより構成される階層構造をもつ．知識

オブジェクトには，集合的な抽象度の高い概念を表した＂クラスレベル＂のオブジェクトと個々の具体的なものを表した＂インスタンスレベル＂のオブジェクトがある．簡単な階層構造の例を図3.1に示す．

階層関係にある複数の知識オブジェクトにおいては，下位あるいは下層のオブジェクトは，デフォルトとして上位あるいは上層のオブジェクトの属性や手続きをもっており，これらを自動的に受け継ぐことができる．この性質を＂属性継承＂という．

LEVELスロットには，この知識オブジェクトがクラスかインスタンスかを定義する．SUPER－CLASSスロットにはクラスオブジェクトの上位にあるクラスオブジェクト名を，SUPER−LEVELにはインスタンスオブジェクトの上層オブジェクト名を指定する．

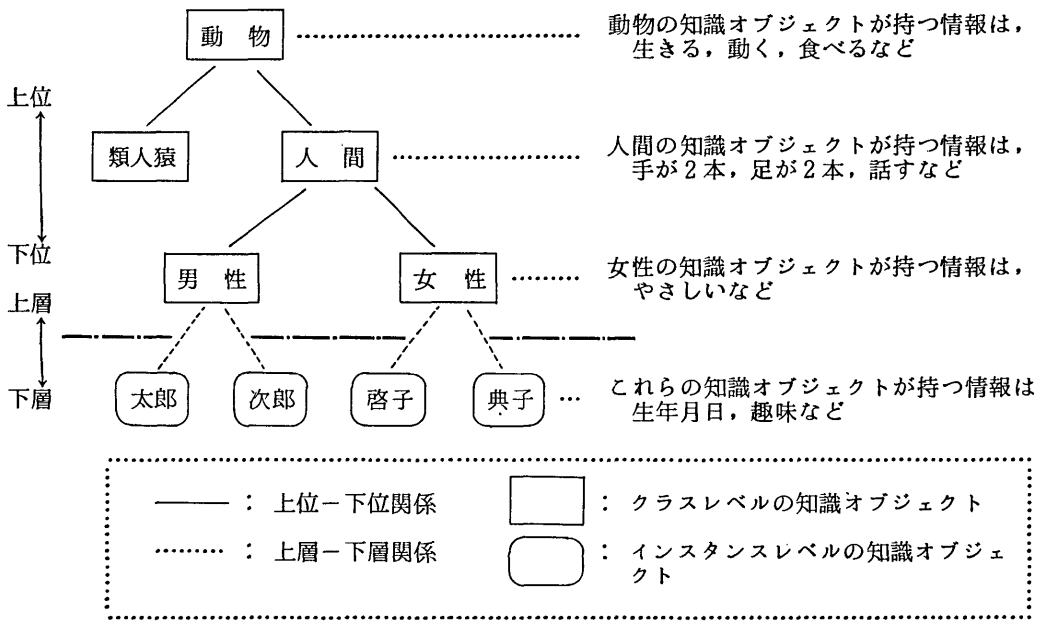


図3.1　簡単な世界モデルの例

(2)　属性スロット

知識オブジェクトの属性データ（スロット名；ユーザ定義語）とその値を記述する．スロット値はシステム実行中に設定，更新できる．またスロットには値以外に，ファシットと呼ばれる副属性やデモンと呼ばれる手続きを定義することができる．

(3)　初期化属性スロット

クラスレベルのオブジェクトは，メッセージNEWが送信されると，インスタンスオブジェクトを動的に生成する．ここでは生成するインスタンスオブジェクトにはじめから持たせるべき属性スロットを記述する．

(4)　メソッドスロット

ここでは，メソッドと呼ばれる手続き（関数やルール集合）との関連づけを記述する．そのスロッ

ト名は"選択子"と呼ばれ，知識オブジェクトに処理（関数やルール集合の起動）を依頼する際に，その処理の名前として指定されるものである．その値にはLisp関数名，LAMBDA式，ルール集合名のいづれかを記述する．

図3.2に知識オブジェクトの具体例を示す．

```
（オブジェクト定義　　LANDKOSOU
　　システム
　　　レベル　　クラス
　　　上　層　　%CLASS%
　　　上　位　　%ENTITY%
　　初期化属性
　　　TYPE2　　SINGLE　*
　　　X　　SINGLE　*
　　　Y　　SINGLE　*
　　メソッド
　　　TH2
　　　　HOURITSU
　　　EXCP
　　　　(LAMBDA (OBJ) (! OBJ 'EXR) (! OBJ 'ZH) (! OBJ 'HY))
　　　EX2
　　　　(LAMBDA (OBJ) (! OBJ 'ZH) (! OBJ 'HY))
　　　EXR
　　　　EXRULE
　　　ZH
　　　　ZAHYOSET
　　　HY
　　　　HYOUJI
）
```

図3.2　知識オブジェクト定義の例

3.1.2　メッセージ送信とメソッド

(1)　メッセージ送信

メッセージとは，知識オブジェクトに対する処理操作を行うためのインタフェースである．メッセージ送信により，処理対象の知識オブジェクトに対して手続き＝メソッド（ルール集合，関数，LAMBDA式）を起動することができる．メッセージはそれ自体Lispの関数である．

一般的なメッセージ送信の記述式を以下に示す．

```
（！　オブジェクト　　選択子　　引数　……）
```

！　　：メッセージ送信のキーワード

オブジェクト　：送信先の知識オブジェクト名

選択子　　：メッセージの名前，受信側オブジェクトはこの名前のメソッドを起動する
引　数　　：メッセージに付加する引数

メッセージ送信は，ルールの条件部，実行部，Lisp 関数内など，Lisp プログラムが記述可能な箇所ならばどこにでも記述できる．

(2) メソッド

メソッドには，ユーザ定義のメソッドとESHELL／Xが標準提供するシステムメソッドがある．ユーザ定義のメソッドはクラスレベルの知識オブジェクトのみがもつことができる．システムメソッドはすべての知識オブジェクトにデフォルトとして定義されている．

(3) メソッドの起動

メソッドの起動はメッセージ送信により行われる．システムメソッドの場合はただちに起動するが，ユーザ定義のメソッドの場合は，メッセージ送信された知識オブジェクト（属性継承の場合は，インスタンスレベルであれば上層及び上位，クラスレベルであれば上位の知識オブジェクト）からメソッドの獲得を行い，獲得できたメソッドを起動する．

(4) システムメソッド

システムメソッドには，知識オブジェクト中のスロットの値を操作する"スロット操作メソッド"，知識オブジェクト自体の生成や削除などを行う"知識オブジェクト操作メソッド"，指定された条件の知識オブジェクトまたはスロットのリストを得る"情報収集メソッド"などがある．以下に比較的よく使用されると思われるものを列挙する．

a. スロット操作メソッド
- REF ：指定スロットの値の参照
- SET ：指定スロット内のスロットへの値の追加／削除，新規設定

b. 知識オブジェクト操作メソッド
- NEW　　：知識オブジェクトの動的生成
- CLEAR-SUB-LEVEL　：下層オブジェクトの削除
- FIND　　：条件に合ったスロット（値）を持つ下層オブジェクトの検索

c. 情報収集メソッド
- TAKE-SUPER-LEVEL　：上層オブジエクトの収集
- TAKE-SUB-LEVEL　　：下層オブジェクトの収集
- TAKE-SUPER-CLASS　：上位オブジェクトの収集
- TAKE-SUB-CLASS　　：下位オブジェクトの収集

d. 検査メソッド
- OBJECTP（関数）：知識オブジェクトの判定

(5) デモン

デモンは，属性スロットに付加される手続きであり，属性スロットの値にアクセスするメッセージの受信を契機に起動される．これにより属性スロットの更新や参照などに付随する処理をアクセス側の手続きから分離することができる．

デモンには次の4種類がある．

- 欠損値デモン（￥IF−NEEDED）
  値が設定されていない属性スロットを参照しようとしたときに起動される．
- 正規化デモン（￥IN−FILTER，　￥OUT−FILTER）
  スロット値の正規化を行う．
- 制約デモン（￥IN−CONSTRAINT，　￥OUT−CONSTRAINTなど）
  スロット値が制約条件を満たしているか否かを調べる．
- 副作用デモン（￥IF−ADDED，　￥IF−REMOVEDなど）
  スロット値にアクセスした直後に起動される手続き．

3.1.3　ルール集合

ルール集合は，一まとまりのルールを管理する枠組みであり，図3.3のように環境記述部とルール記述部からなる．

ルール集合の一般記述式は以下のとおりである．

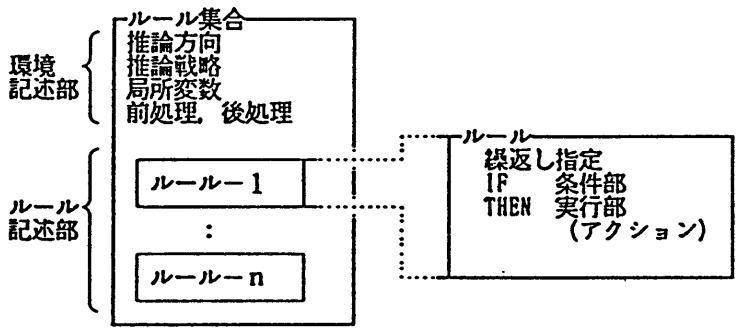


図3.3　ルール集合の構成

```
( DEFRULES     ルール集合名
    FORWARD［又はBACKWARD］
    STRATEGY  SINGLE［又はMULTI］
     ......
    RULES
     RULE−NAME  ルール名
              ［COMMENT    コメント］
              ［ITERATE    繰り返し式］
              IF        条件部
              THEN      実行部
     ......
     ......
)
```

環境記述部には，推論の実行戦略の指定，ルール集合内で使用するローカル変数の宣言，推論の前後に起動される処理などを定義する．

ルール記述部の条件部には，Lisp のＳ式を記述する．一般には条件部のすべてのＳ式がＮＩＬでない値を返したとき，そのルールの条件は成立する．

実行部には，条件部が成立したときに実行される処理を記述する．この処理のことをアクションという．以下の９種類のアクションがある．

- OBJECTアクション　：知識オブジェクトの生成，属性値の設定，更新を行う．
- EXECUTEアクション：Lisp のＳ式を評価する．
- CHAINアクション　：後ろ向き推論を行う場合に指定する．
- EVENTアクション　：即時型のイベントを発行する．
- EXPECTアクション　：保留型のイベントを発行する．
- COMMENTアクション：注釈を記述する場合に用いる．
- LOOPアクション　：あるアクションを繰り返して実行する場合に用いる．
- ＢＢアクション　：黒板ノードの生成，属性値の設定，変更を行う．
- BBCHAINアクション：黒板を利用した後ろ向き推論を行う場合に指定する．

アクションは複数定義することができる．

ルール集合定義の具体例を図3.4に示す．

```
(ルール集合定義    EXRULE    FORWARD
  実行戦略          SINGLE
  変数              LK    = (NTH 0 (! 'LANDKOSOU 'TAKE-SUB-LEVEL))
                    YOTO = (! 'LAND-OBJ 'REF 'YOTO)
  ルール
    ルール名      EX1
      もし          (>= YOTO 1)
                    (<= YOTO 3)
      ならば        <OBJECT>       ; ACTION-NO (1)
                      PROCESS     SET
                      OBJECT      LK
                      SLOTS       TYPE2 = "RESIDENTIAL"
    ルール名      EX2
      もし          (>= YOTO 4)
                    (<= YOTO 5)
      ならば        <OBJECT>       ; ACTION-NO (1)
                      PROCESS     SET
                      OBJECT      LK
                      SLOTS       TYPE2 = "COMMERCIAL"
    ルール名      EX3
      もし          (>= YOTO 6)
                    (<= YOTO 8)
      ならば        <OBJECT>       ; ACTION-NO (1)
                      PROCESS     SET
                      OBJECT      LK
                      SLOTS       TYPE2 = "INDUSTRIAL"
)
```

図3.4　ルール集合定義の例

その他の知識表現に，あいまい表現を定義するDEFFUZZYなどがある．詳細についてはマニュアル[1)]を参照されたい．

3.2　推論方式

本節では，利用者が作成した知識ベースをESHELL/Xがどのように解釈し，推論を実行するかについて簡単に説明する．

推論は，基本的には以下の処理の繰り返しである．

- ルール集合の起動
- ルール集合内のルールの解釈

3.2.1　ルール集合の起動方式

起動方式には以下の2種類がある．

(1)　メッセージ送信によるルール集合の起動

知識オブジェクト（メソッドスロットの値にルール集合名が設定されている）に対してメッセージを送信することにより，ルール集合は起動する．メッセージはルール集合内のルールにも記述できるので，ルール集合から別のルール集合を起動することが可能である．すなわち，この方法はサブルーチン的起動方法と言える．図3.5を参照されたい．

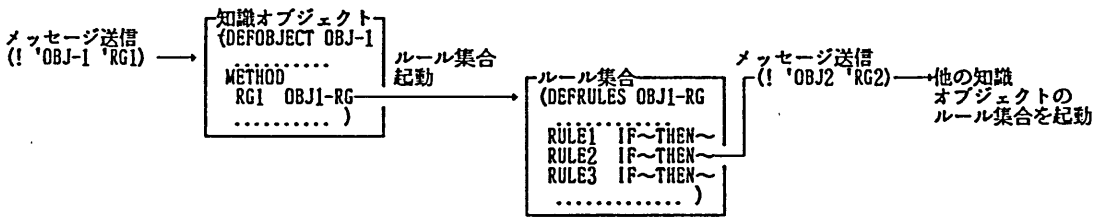


図3.5　メッセージ送信によるルール集合の起動

(2)　イベント発行によるルール集合の起動

起動順序が静的に決まらない場合，又は起動がサブルーチン的でない場合，イベントと呼ばれる実行依頼を発行することによってルール集合が起動される．この場合，推論の実行を制御する特別な知識オブジェクトが必要である．この推論制御オブジェクトに対して，メッセージRUNを送信することにより推論が開始される．図3.6を参照されたい．

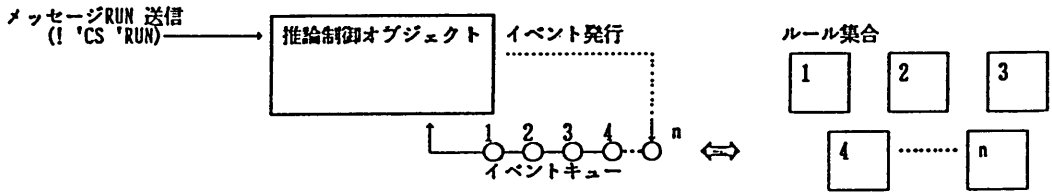


図3.6　イベント発行によるルール集合の起動

3.2.2　ルール集合内のルールの解釈

ESHELL／Xには，前向き推論と後ろ向き推論の二つの方式がある．さらに確信度付き推論，ファジィ推論，黒板を利用した推論方式が用意されているが，ここでは最も基本的な前向き推論方式によるルールの解釈を概説する．

前向き推論は，推論の種類FORWARDで指定する．はじめに環境記述部の解釈が行われ，次に各ルールの解釈が始まる．前向き推論用ルール集合では，ルールは記述されている順番どおりに一つずつ調べられていく．

1) ルールの条件部に記述されているLispのS式を評価する．
2) 条件部の評価結果がNILならば，そのルールの解釈は中止され，次のルールを調べる．
3) 条件部の評価結果がNIL以外ならば，実行部で指定されているアクションをすべて実行する．
4) ルールに繰り返し指定（ITERATE）があれば，繰り返しの指定にしたがって上記1) から 3) を繰り返す．
5) 実行戦略（STRATEGY）がSINGLEであれば，最初に条件が真になった実行部を処理し，以後のルールの解釈は行わない．MULTI であれば，条件が真になったすべてのルールの実行部を処理する．

3.3　実行支援機能

実行支援の構成機能は以下のとおりである．

- 図形表示
- ファイル入出力
- データベースとの連携

ここでは，とくに図形表示とファイル入出力機能の使用法を概説する．

3.3.1　図形表示機能

図形表示機能は，エキスパートシステムのMMI（マン・マシン・インターフェイス）を向上させるための重要な支援機能である．利用者がこの機能を使うためには，図形表示用の機能クラスのオブジェクト（DEFGRAPHIC）の定義が必須である．

一般に具体的な図形属性データは，DEFGRAPHICの下層オブジェクト内の図形属性スロット（GRAPH−DATA）に記述する．図形記述の種類を以下に示す．

[LINE, POLYLINE, POLYGON, BOX, CIRCLE
ARCH, ARCHC, CHARACTER]

さらに線の種類，太さ，色，面塗りの色，面塗りのパターン等も指定できる．

この図形表示用のオブジェクトの操作には，次のような専用の図形操作メソッドが用意されている．

- DISPLAY　：オブジェクトの図形データを表示する．
- UNDISPLAY　：表示図形を消去する．
- RESHOW　：消去したオブジェクトを再表示する．
- REWRITE　：画面全体を作画し直す．
- MOVE　：オブジェクトを指定された位置に動かす．
- ZOOM　：オブジェクトを指定量だけ拡大，移動させる．
- TERMINATE－GRAPH（関数）：図形表示機能のモジュールをメモリから削除する．

例えば図形を表示させるためには，次のようにメッセージを送信する．

```
（！　図形OBJECT名　'DISPLAY）
```

3.3.2　ファイル入出力機能

ESHELL/Xのファイル入出力の特徴は，ファイルのデータを知識オブジェクトに写像して，知識オブジェクトのスロットの値として参照，設定する点にある．扱うファイルは，順編成又は区分編成データセットである．ファイルの内容に応じて，レコード単位入出力とデータ単位入出力の処理形態がある．

本機能を利用するためには，以下の二つの知識オブジェクトが必要である．

- 機能クラスオブジェクト（DEFIO）
  これには処理対象のファイル名（スロットPATHに指定），処理形態（スロットPROCESSにRECORD又はDATAを指定），データ項目（スロットINITIALに指定），ファイルのレコード長やデータ長を指定する．
- 入出力域オブジェクト（DEFIOの下層オブジェクト）
  ファイルから読み込んだデータを，スロット値として設定したり，既に設定してあるスロット値をファイルに書き出すためのオブジェクト．

ファイル入出力用の操作メソッドには次のものがある．

- OPEN　：ファイル入出力の環境設定のための初期処理を行う．
- CLOSE　：処理環境の解除を行う．
- READ　：ファイル内のデータを読み込み，入出力域オブジェクトのスロットに格納する．
- WRITE　：入出力域オブジェクトのスロット値をファイルに書き込む
- TAKE-BUFFER　：入出力域オブジェクトの名前の参照を行う．

ファイル入出力の概要（レコード単位の場合）を図3.7と図3.8に示す．図3.8のオブジェクトY1は，メッセージ（！　'Y　'OPEN　'IN）により動的に生成されるインスタンスオブジェクトである．

```
( DEFIO   Y
    PROPERTY
       PROCESS              RECORD
       PATH                 "INFILE"
    INITIAL
        氏名      SINGLE  *
                  ¥LENGTH  16
        年齢      SINGLE  *
                  ¥LENGTH  8
        身長      SINGLE  *
                  ¥LENGTH  15
)
```

図3.7　レコード単位入出力用の機能クラス

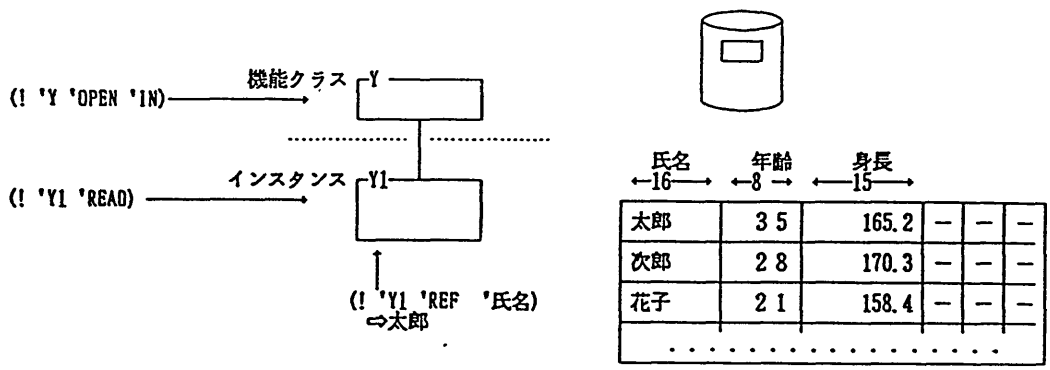


図3.8　レコード単位のファイル入出力処理概要

4. ESHELL／Xの起動と開発環境

4.1　起動方法

(1)　動作環境

ESHELL／Xの動作環境には，知識ベースの作成段階で選択する開発環境と，知識ベースが完成し，エキスパートシステムを運用する場合の実行環境がある．ESHELL／Xの動作環境を図4.1に示す．

① ESHELL／Xプログラムモジュール．

② メッセージファイル ： ESHELL／Xから表示されるメッセージを格納する．

③ 知識ベースファイル ： 知識オブジェクト，ルール集合，手続きなどの，利用者の作成した知識を格納するファイル．ファイルはLispのソースプログラムが記述できる形式であればよい．行番号があってはならない．ファイル全体の知識編集は外部エディタを使用する．

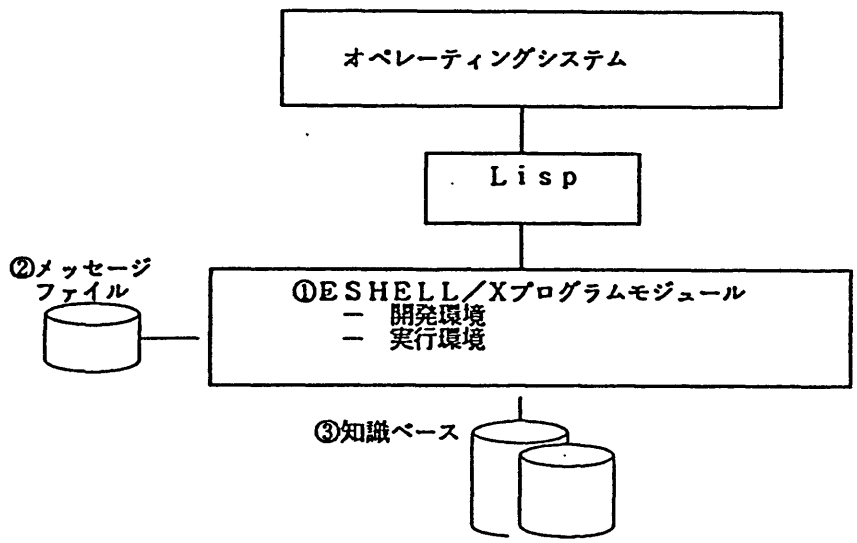


図4.1　ESHELL／Xの動作環境

(2)　起動方法

ESHELL／Xはコマンドプロシジャ ESHELLXにより起動される．開発環境か実行環境かの選択はオペランドで指定する．省略すると開発環境が選択される．また，図形表示機能，ファイル入出力機能等を使用する場合には，対応するオペランドが必要である．

```
READY
ESHELLX 'FILE'                    ①

>(ESHELL/X)                       ②
*コマンドモード開始，ESHELL/X
ES!
```

① ファイル入出力機能の使用を指定して開発環境を起動する．

② Lisp モードになり，プロント＞が表示される．関数ESHELL／X又はESHELLXを実行すると，ESHELL/Xのコマンドモードとなり，プロンプトがES！に変化する．

4.2 コマンド一覧

プロンプトES！でコマンドが入力できる状態になる．この状態をトップレベルモードという．表4.1にトップレベルモードのコマンド一覧を示す．

表4.1 ESHELL／Xトップレベルコマンド一覧

コマンド名	コ マ ン ド の 説 明
LOAD	知識ベースを読み込む．
RUN	知識ベースの実行を指示する．
SAVE	知識ベースを外部ファイルに保存する．
LISTKB	メモリ中の知識ベース一覧を表示する．
LISTKN	メモリ中の知識名を表示する．
BROWSE	知識内容を表示する．
RESET	メモリ中の知識ベースを消去する．
X	TSSコマンドを実行する．
EDIT	知識を編集する（＃MSP，＃FSP）．
DEBUG	知識ベースのデバッグを行う．
PRINT	知識ベースの内容を印刷する．
QUIT	ESHELL／Xを終了し，Lisp モードに復帰する．
QUITX	ESHELL／Xを終了し，READY状態に復帰する．
?	利用可能なコマンド一覧を表示する．
??	利用可能なコマンドを説明する．

LOADコマンドにより知識ベースが格納されているファイルを読み込むと，ファイル中の知識は主記憶中に展開され，推論実行の可能な状態になる．

EDITコマンド，DEBUGコマンド，PRINTコマンドにはサブコマンドが用意されているが，中でもDEBUGコマンドは，推論の実行の流れを追跡したり，一時中断する機能をもっており，システムの開発段階で知識ベース中のエラー個所を発見する手掛かりとすることができ，有効なコマンドである．

表4.2にデバッガのサブコマンドの一覧を示す．

表4.2　デバッグサブコマンド一覧

サブコマンド名	指定可能なモード	用　　途
TRACE	両　方	実行追跡のための情報を指定する．
UNTRACE	両　方	実行追跡情報の表示を解除する．
AT	両　方	中断点を設定する．
OFF	両　方	中断点を解除する．
LISTTRCS	両　方	表示対象のトレース情報の一覧を表示する．
LISTBRKS	両　方	中断点の一覧を表示する．
GO	ブレークモードのみ	推論の実行を継続する．
RUN	デバッグモードのみ	推論の実行を開始する．
X	両　方	TSSコマンドを実行する．
Q	両　方	上位のモードに移る．
?	両　方	利用可能なコマンド一覧を表示する．
??	両　方	利用可能なコマンドを説明する．

5. 使用例

筆者らはESHELL/Xを用いて地域メッシュを推論の単位とした土地利用構想支援エキスパートシステムを開発中である．このシステムは，都市を1/4地域メッシュ（250m×250m）に分割し，メッシュごとに推論を繰り返し，将来の都市の土地利用構想図を描こうとするものである．ここではESHELL/Xの具体的な使用例として，開発初期の1メッシュのみの土地利用タイプの推論を行うトイ・モデルを紹介する．

(1)　モデルの概要

最終的な推論結果は，そのメッシュの土地利用タイプ（住宅地，商業地，工業地）とし，メッシュのもつ情報としては，住，商，工の土地利用面積構成比と用途地域指定を想定した．つまり，これらの事実的知識をもとに一定の専門的知識であるルールを活用して土地利用タイプを推論するという問題を考える．

ここで用いたルールは以下の２種類である．

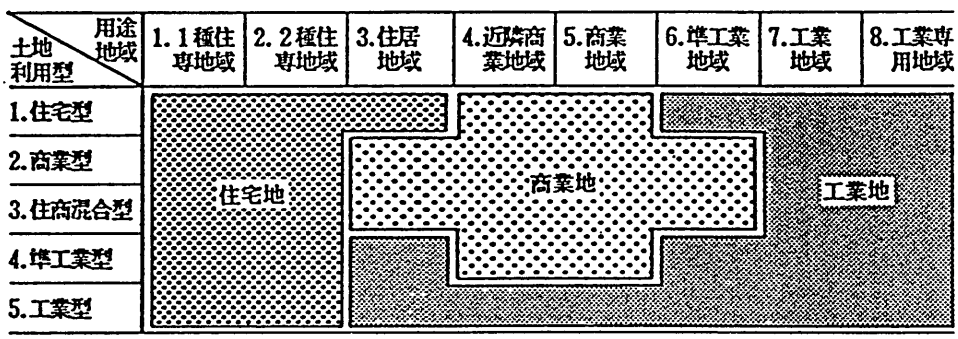
① 住，商，工の土地利用面積構成比による土地利用型の判定（表5.1参照）

② ①の土地利用型と用途地域指定による土地利用タイプ（表5.2参照）

表5.1 土地利用面積構成比による土地利用型

土地利用面積構成比	土地利用型 （番号）
住宅用地≧７５％	住宅型 （１）
商業用地≧７５％	商業型 （２）
住宅用地＜７５％かつ， 商業用地＜７５％かつ， 工業用地＜７５％	住商混合型 （３）
工業用地≧２５％かつ， 工業用地＜７５％	準工業型 （４）
工業用地≧７５％	工業型 （５）

表5.2 土地利用型と用途地域指定による土地利用タイプ



(2) プログラムの解説

以上の問題をESHELL／Xの知識表現言語を使ってプログラム化したのが図5.2である．土地利用面積構成比と用途地域指定は会話型で入力する方法を採用した．これらのデータはファイル入出力機能を利用して一般のデータセットから読み込むことも可能である．なおこの知識表現では，知識オブジェクトの階層構造は考慮していない．

プログラムは知識オブジェクト４つ，ルール集合２つ，手続き関数８つから成り，ルール集合の起動はメッセージ送信によって行われる．

システムの処理の流れを簡単に説明すると次のようになる（図5.1参照）．

ｉ）関数RUNの実行で推論開始．

ii）システム側から土地利用構成比を聞いてくるので順次入力する．

iii）ルール集合TRIANGLEが実行され，構成比による土地利用型の判定を行う．同時にその結果を表示する．

iv）次に用途地域指定を聞いてくるので，これを入力する．

v）ルール集合HOURITSUが実行され，土地利用タイプが決定され，表示される．

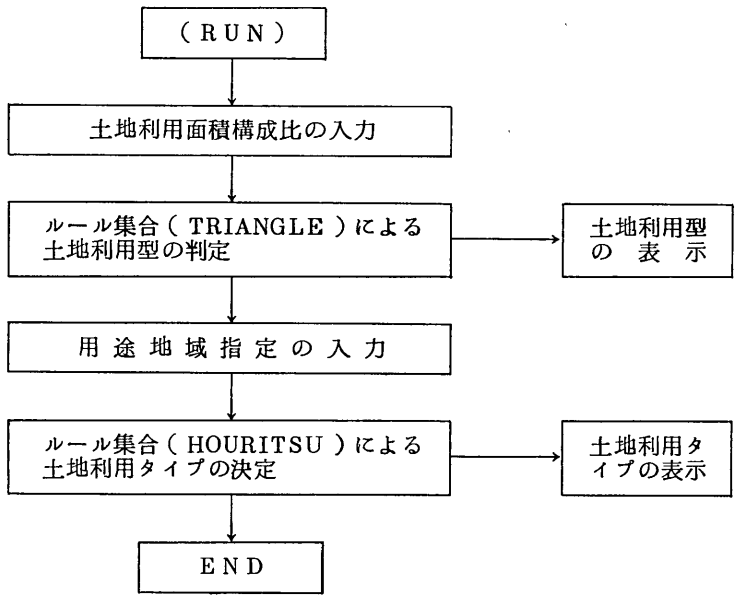


図5.1　処理の流れ

以下，プログラム内の知識ベースごとにもう少し詳しく解説する．

①　関数（RUN）の実行によってシステムが起動され，＊＊＊LAND−USE TYPE NO HANTEI ＊＊＊の画面表示の後，（TYPEHNT）が評価される．

②　この関数はシステムの処理の流れを制御するもので，知識オブジェクトLAND−USEとHANTEIに対して順次メッセージを送信し，各々のオブジェクトのメソッドを起動させる．

③　クラスレベルのオブジェクトで，メッセージ送信を受けてルール集合TRIANGLEを起動させ，その結果の土地利用型を属性スロットTYPE，NUMBに格納する．またその値を表示する手続きを起動させるメソッドも管理している．

④　構成比による土地利用型を画面表示する関数．

⑤　構成比による土地利用型の判定を行うルールを管理するルール集合である．オブジェクトLAND−USEのメソッドTHにより起動される．

ルールの解釈の前にローカル変数の右辺が順次評価される．＊FOCUSOBJ＊はシステム変数で，現在推論の対象となっている知識オブジェクト名に相当する．次に，メッセージ（！ 'SJK' NEW)

```
(DEFUN RUN NIL (FORMAT " *** LAND-USE TYPE NO HANTEI *** /N") (TYPEHNT))---①

(DEFUN TYPEHNT                                          ----------②
            NIL
   (! 'LAND-USE 'TH)
   (! 'LAND-USE 'EX)
   (! 'HANTEI 'TH2)
   (! 'HANTEI 'EX2))

(オブジェクト定義     LAND-USE                          ----------③
    システム
      レベル     クラス
      上層       %CLASS%
      上位       %ENTITY%
    属性
      TYPE       SINGLE  "MIXED-S-J"
      NUMB       SINGLE  3
    メソッド
      TH
                 TRIANGLE
      EX
                 EXPR
 )

(DEFUN EXPR
          (LAND-USE)
   (FORMAT "--LAND USE NO TYPE ; /S/N" (! 'LAND-USE 'REF 'TYPE)))  -------④

(ルール集合定義    TRIANGLE    FORWARD                  ----------⑤
  実行戦略          SINGLE
  変数              LAND-USE = *FOCUSOBJ*
                    RATE     = (! 'SJK 'NEW)
                    SR       = (! RATE 'REF 'SR '(/+))
                    JR       = (! RATE 'REF 'JR '(/+))
                    KR       = (! RATE 'REF 'KR '(/+))
  ルール
    ルール名     SYO
      もし          (>= SR 75)
      ならば        <OBJECT>      ; ACTION-NO (1)
                      PROCESS    SET
                      OBJECT     LAND-USE
                      SLOTS      TYPE = "COMMERCIAL"
                                 NUMB = 2
    ルール名     SYO-JYU
      もし          (<= SR 75)
                    (<= JR 75)
                    (<= KR 25)
      ならば        <OBJECT>      ; ACTION-NO (1)
                      PROCESS    SET
                      OBJECT     LAND-USE
                      SLOTS      TYPE = "MIXED-S-J"
                                 NUMB = 3
    ルール名     JYU
```

図5.2　プログラム例

```
        もし            (>= JR 75)
        ならば          <OBJECT>        ; ACTION-NO (1)
                          PROCESS     SET
                          OBJECT      LAND-USE
                          SLOTS       TYPE = "RESIDENTIAL"
                                      NUMB = 1
      ルール名      KOU
        もし            (>= KR 75)
        ならば          <OBJECT>        ; ACTION-NO (1)
                          PROCESS     SET
                          OBJECT      LAND-USE
                          SLOTS       TYPE = "INDUSTRIAL"
                                      NUMB = 5
      ルール名      JYUNKOU
        もし            (<= KR 75)
                        (>= KR 25)
        ならば          <OBJECT>        ; ACTION-NO (1)
                          PROCESS     SET
                          OBJECT      LAND-USE
                          SLOTS       TYPE = "SEMI-INDUS"
                                      NUMB = 4
)

(オブジェクト定義      SJK                                    ---------- ⑥
    システム
      レベル      クラス
      上層        %CLASS%
      上位        %ENTITY%
      下層        $JK0002
    初期化属性
      SR          SINGLE  *
                  ¥IF-NEEDED    SRIN
      JR          SINGLE  *
                  ¥IF-NEEDED    JRIN
      KR          SINGLE  *
                  ¥IF-NEEDED    KRIN
)

(DEFUN SRIN NIL (RIND "---SYOGYO NO RATE? >>"))        ---------- ⑦

(DEFUN JRIN NIL (RIND "---JYUKYO NO RATE? >>"))        ---------- ⑧

(DEFUN KRIN NIL (RIND "---KOUGYO NO RATE? >>"))        ---------- ⑨

(オブジェクト定義      HANTEI                                 ---------- ⑩
    システム
      レベル      クラス
      上層        %CLASS%
      上位        %ENTITY%
    属性
      TYPE2       SINGLE  "INDUSTRIAL"
    メソッド
      TH2
```

図5.2　プログラム例（つづき１）

```
                HOURITSU
      EX2
                EXPR2
 )

(DEFUN EXPR2                                                    ---------- ⑪
            (HANTEI)
   (FORMAT " *TOCHI-KOSOU* :(  /S )/N" (! 'HANTEI 'REF 'TYPE2)))

(ルール集合定義     HOURITSU     FORWARD                       ---------- ⑫
  実行戦略           SINGLE
  変数               LAND-USE2 = *FOCUSOBJ*
                     RATE2     = (! 'SJK2 'NEW)
                     TYPE      = (! 'LAND-USE 'REF 'NUMB)
                     YOTO      = (! RATE2 'REF 'YOTO '(/+))
  ＬＨＳ評価関数
                       @OR
  ルール
    ルール名      RR
      もし           (AND (= TYPE 1) (<= YOTO 3))
                     (AND (>= TYPE 2) (<= TYPE 5) (<= YOTO 2))
      ならば         <OBJECT>       ; ACTION-NO (1)
                       PROCESS     SET
                       OBJECT      LAND-USE2
                       SLOTS       TYPE2 = "RESIDENTIAL"
    ルール名      CC
      もし           (AND (= TYPE 1) (>=YOTO 4) (<= YOTO 5))
                     (AND (>= TYPE 2) (<= TYPE 3) (>= YOTO 3) (<= YOTO 6))
                     (AND (= TYPE 4) (>= YOTO 4) (<= YOTO 5))
      ならば         <OBJECT>       ; ACTION-NO (1)
                       PROCESS     SET
                       OBJECT      LAND-USE2
                       SLOTS       TYPE2 = "COMMERCIAL"
    ルール名      II
      もし           T
      ならば         <OBJECT>       ; ACTION-NO (1)
                       PROCESS     SET
                       OBJECT      LAND-USE2
                       SLOTS       TYPE2 = "INDUSTRIAL"
 )

(オブジェクト定義      SJK2                                     ---------- ⑬
    システム
      レベル     クラス
      上層       %CLASS%
      上位       %ENTITY%
      下層       SJK20003
    初期化属性
      YOTO       SINGLE  *
                 ¥IF-NEEDED    YTIN
 )

(DEFUN YTIN NIL (RIND "---YOTO-SHITEI? >>"))                    ---------- ⑭
```

図5.2　プログラム例（つづき２）

（記述方法は，3.1.2参照）が評価され，クラスオブジェクトSJKのインスタンスオブジェクトが動的に生成され，RATE変数にはその名前が入る．そして続く3つのメッセージ送信により，そのインスタンスオブジェクトの属性スロットSR，JR，KR（各々，商，住，工の土地利用面積構成比を表す）の値を参照して，ローカル変数SR，JR，KRに各々の値が設定される．なおその際，メッセージ内のキーワード（＋）によりオブジェクトSJKの属性スロットに定義されたデモンが起動される．

ルールの解釈は前向き推論で，実行戦略はSINGLEである（ルールの解釈は3.2.2参照）．例えば，ルール名SYOの条件が成立すると，その実行部のOBJECTアクションが実行される．すなわち，オブジェクトLAND−USEの属性スロットTYPEに文字例"COMMERCIAL"が，またNUMBには数値2が設定される．

⑥　クラスレベルのオブジェクトで，それぞれ欠損値デモンの付随した初期化属性スロットSR，JR，KRをもつ．ルール集合TRIANGLE内からメッセージ（！ 'SJK 'NEW）により動的に生成される下層オブジェクトを従えている．

⑦〜⑨　土地利用構成比を会話型で入力するための手続き関数である．この関数の起動は，SJKの下層オブジェクトがルール集合TRIANGLE内のメッセージ（！ RATE 'REF 'SR '(／+)）を受信して，属性スロットSRの値を参照しようとする際に，欠損値デモン¥IF−NEEDEDが起動されてなされる．

⑩　③と同様にクラスレベルのオブジェクトで，②のメッセージ送信によりルール集合HOURITSUを起動させ，その結果を属性スロットTYPE2に格納する．またその値を表示する関数を起動させるメソッドも管理している．

⑪　推論結果の土地利用タイプを画面表示する関数．

⑫　構成比による土地利用型と用途地域指定の関連より土地利用タイプを判定するルール集合である．オブジェクトHANTEIのメソッドTH2により起動される．

ローカル変数LAND−USE2はオブジェクトHANTEIに等しい．RATE2には，（！ 'SJK2 'NEW）により，クラスオブジェクトSJK2のインスタンスオブジェクトが動的に生成され，その名前が入る．YOTOの値は，そのインスタンスオブジェクトの属性スロットYOTOの値を参照して，設定される．またTYPEの値は，オブジェクトLAND−USEのNUMBを参照して，設定される．

ルールの解釈の仕方は⑤と同様であり，いずれかの条件が成立すると，その実行部のOBJECTアクションが実行される．すなわち，オブジェクトHANTEIの属性スロットTYPE2に文字列が設定される．

⑬　⑥と同様，欠損値デモンの付随した初期化属性スロットYOTOをもつクラスオブジェクトである．ルール集合HOURITSU内からメッセージ（！ 'SJK2 'NEW）により動的に生成される下層オブジェクトを従えている．

⑭　用途地域指定を会話型で入力するための手続き関数である．この関数は，SJK2の下層オブジェクトがルール集合HOURITSU内のメッセージ（！ RATE2 'REF 'YOTO '(／+)）を受信することにより，属性スロットYOTOの欠損値デモン¥IF−NEEDEDが起動され，実行される．

(3)　実行例

システムのトップレベルモードでの実行例を図5.3に，デバッグモードでの実行例を図5.4に示す．

```
READY
eshellx
> (eshellx)
*コマンドモード開始        ESHELL/X
ES] load "satori.data(ex3)"
*処理が正常終了した
ES] (run)
 *** LAND-USE TYPE NO HANTEI ***
---SYOGYO NO RATE? >> 15
---JYUKYO NO RATE? >> 65
---KOUGYO NO RATE? >> 20
--LAND USE NO TYPE ; "MIXED-S-J"
---YOTO-SHITEI? >> 5
 *TOCHI-KOSOU* :(  "COMMERCIAL" )
NIL
ES]
```

図5.3　実行例（トップレベルモード）

```
ES] debug
DEBUG] trace rulegroup *
*処理終了
DEBUG] trace rule triangle *
*処理終了
DEBUG] at rule triangle *
*処理終了
DEBUG] (run)
 *** LAND-USE TYPE NO HANTEI ***
++ TRIANGLE : START RULEGROUP
---SYOGYO NO RATE? >> 10
---JYUKYO NO RATE? >> 60
---KOUGYO NO RATE? >> 40
 + SYO : START RULE
 @ AT START RULE : RULE = SYO
BREAK? go
   * SYO : IF (@AND (>= SR 75))
  ==> NIL
 - SYO : END      RULE
 @ AT END RULE     : RULE = TRIANGLE
```

（中略）

```
BREAK? go
    * JYUNKOU : IF (@AND (<= KR 75) (>= KR 25))
  ==> T
    * JYUNKOU : THEN EXECUTED
 - JYUNKOU : END      RULE
 @ AT END RULE     : RULE = TRIANGLE
BREAK? go
-- TRIANGLE : END      RULEGROUP
--LAND USE NO TYPE ; "SEMI-INDUS"
++ HOURITSU : START RULEGROUP
---YOTO-SHITEI? >> 7
-- HOURITSU : END      RULEGROUP
 *TOCHI-KOSOU* :(  "INDUSTRIAL" )
NIL
DEBUG] q
```

図5.4　実行例（デバッグモード）

6. おわりに

現在開発中のシステムでは，MMIの向上を図るため既存システムの連携を検討している．すなわち，メッシュ情報をファイルから読み込み，推論結果，中間仮説，推論過程をさらに別のファイルに出力して，CGMS（Computer Graphics and Movie System）による図形表示を試みている．図5.5は画像情報システムFIVISを使用して推論結果を表示したものである．

ESHELL/X自身にも図形表示機能はあるが，利用者の要求に十分応えるだけの機能は用意されていないようである．ESHELL/X自体はあくまで知識ベースの構築ツールであると考えるのが妥当であり，実用的なエキスパートシステムを目指すには，周辺システムとの連携が重要になってくると思われる．

より多様な動作環境への対応と，より身近なエキスパートシステム構築環境を得るために，さらに以下の点の発展を期待する．

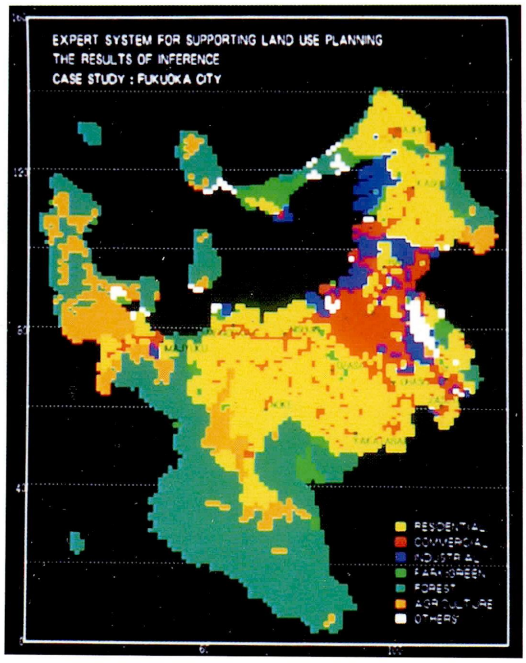
① 本来の性能の向上と共に，前述したシステム連携機能の強化をはかることでシステムの柔軟性と適用範囲を拡大する．

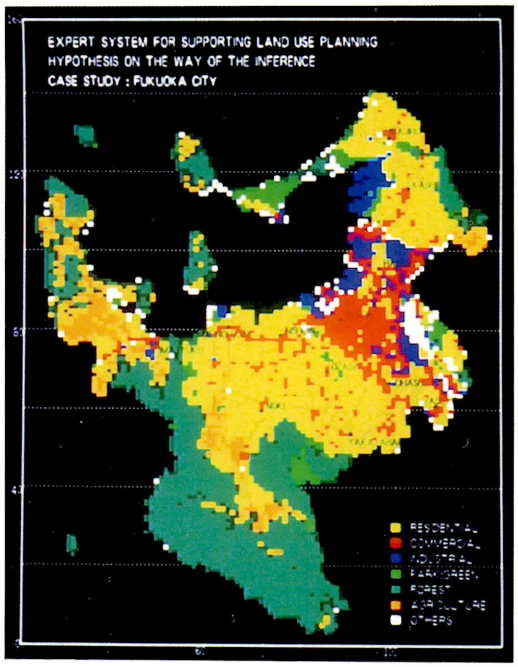
② 知識管理の機能を充実させることにより，知識の規模拡大に対処する．

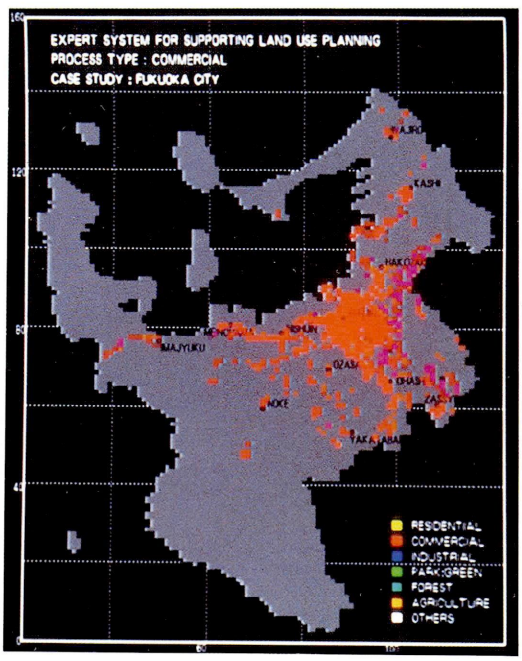
③ 診断，制御等の特定された問題に対する高レベルの知識表現と推論機構を整備することにより，知識獲得に要する様々なコストを削減する．

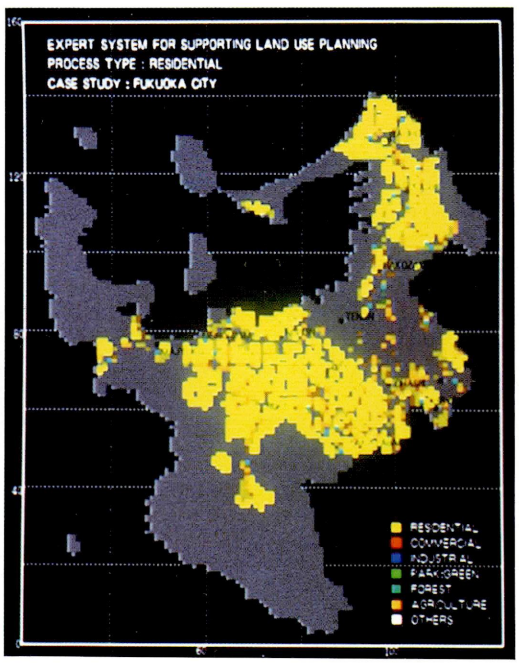
参考文献

1. 計算機マニュアル，ESHELL/X説明書V10用（99SP-4010-1），富士通㈱
2. 上野　晴樹　知識工学入門，オーム社，1985
3. OHM別冊，エキスパートシステムの実務，オーム社


EXPERT SYSTEM FOR SUPPORTING LAND USE PLANNING
THE RESULTS OF INFERENCE
CASE STUDY : FUKUOKA CITY
RESIDENTIAL
COMMERCIAL
INDUSTRIAL
PARK/GREEN
FOREST
AGRICULTURE
OTHERS


EXPERT SYSTEM FOR SUPPORTING LAND USE PLANNING
HYPOTHESIS ON THE WAY OF THE INFERENCE
CASE STUDY : FUKUOKA CITY
RESIDENTIAL
COMMERCIAL
INDUSTRIAL
PARK/GREEN
FOREST
AGRICULTURE
OTHERS


EXPERT SYSTEM FOR SUPPORTING LAND USE PLANNING
PROCESS TYPE : COMMERCIAL
CASE STUDY : FUKUOKA CITY
KASHI
MAJIYUKU
OZAKI
HOKE
RESIDENTIAL
COMMERCIAL
INDUSTRIAL
PARK/GREEN
FOREST
AGRICULTURE
OTHERS


EXPERT SYSTEM FOR SUPPORTING LAND USE PLANNING
PROCESS TYPE : RESIDENTIAL
CASE STUDY : FUKUOKA CITY
RESIDENTIAL
COMMERCIAL
INDUSTRIAL
PARK/GREEN
FOREST
AGRICULTURE
OTHERS