U.D.C. 681.3：007.51：681.3.06：681.518.5

清水建設研究報告
第46号昭和62年10月

Prolog による前向き推論システムの開発とその応用

村越修平
（技術研究所）

§1. はじめに

近年，建設業界においても人工知能，特にエキスパート・システムへの関心が高まり，いくつかのプロトタイプが開発されている．このような知識工学的手法を応用したシステムは，従来の手続き指向のプログラムが不得意とした人間の思考に近いシステムを実現する上で，有効な手段となることが期待されている．当社でも，大型計算機を用いて杭工法選択システムや官公庁提出書類システムなどを開発した．一般に，このような知識工学手法を応用したシステムは，大型計算機や専用のエンジニアリング・ワークステーションを用いて開発されていて，機能的には優れている．しかし，使用に際して電話回線を使用しなければならなかったり，高価なエンジニアリング・ワークステーションを必要とするなど，実際の使用環境は必ずしも良くない．一般ユーザーの使用環境を整備することは，ユーザーにそのシステムを使ってもらい問題点を洗い出す意味からも，またそのシステムの有効性・実用性を検証する意味からも重要である．

本報告では，実際の使用環境を考慮し，パーソナル・コンピュータを用いた Prolog による前向き推論システム（Excel）とその開発支援用システム（Excel 2, Elint）の詳細について述べる．さらに，これらのシステムの有効性・実用性を検証するため，実際に応用した例についても述べる．これらのことから，大型計算機を用いて開発されたエキスパート・システムを，容易に一般ユーザーに提供できることが確認できた．さらに，本システム単独によるアプリケーション・システムの開発も可能である．これらの意味から，一般ユーザーの使用環境は飛躍的に向上したといえる．

§2. システムの概要

本システムのような知識工学を応用したシステムは，従来のプログラムとは，システム設計の手法が多少異なる．基本的には小さな機能の集合体であり，集合体としてのバランスをとることがシステム設計となる．また，このようなシステムではプログラムの制御部とデータとが明確に分離される点も大きく異なる．

ここでは，Prolog による実用的な実行用前向き推論システム（Excel）とその開発支援用システム（Excel 2, Elint）の概要について，知識工学的手法の立場から述べる．

2.1 前向き推論システム（Excel）

ここでは，本システムの機能と構成，分類・診断型としての前向き推論システム，推論エンジンの認知・行動サイクル，知識表現の枠組み，不確実性の取り扱い，データ構造（事実や症状としてのデータと，プロダクション・ルールとしてのデータおよびそれらの書式）と推論の実行アルゴリズムについて述べる．

2.1.1 システムの機能と構成

本システムの提供する機能は，基本的に以下のようなものである．

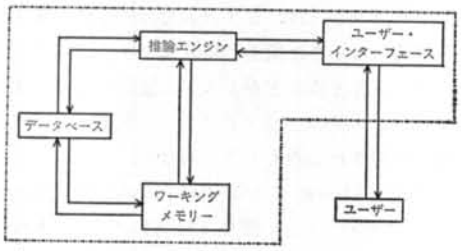
(1)高速な前向き推論による分類・診断結果の提示
(2)グラフィック機能を用いた視覚的効果の高いユーザー・インターフェース
(3)プロダクション・ルールによる比較的簡単なアプリケーション・システムの構築

実際の使用環境を考慮すると，推論速度は高速であることが望ましい．本システムの推論速度は，後述のアプリケーション・システムによって実用レベルを確保していることが実証できた．また，グラフィック機能を用いたユーザー・インターフェースは，一般ユーザーに対して十分な視覚的効果を与えるものである．さらに，知識表現の枠組みにプロダクション・ルール型のものを採用していることから，一般ユーザーでも比較的簡単にアプリケーション・システムになじむことができる．

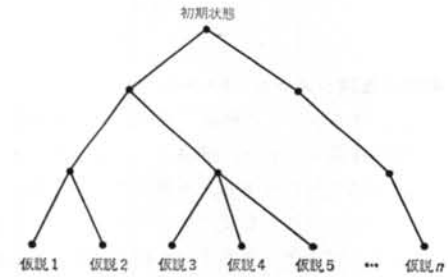
本システムは，大きく4つの部分に分けられる．推論

エンジン本体，データベース，ワーキングメモリーとユーザー・インターフェースである（図—1）．

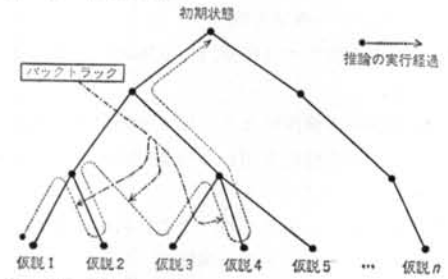
データベースは，専門家の知識をプロダクション・ルールとして，また推論に必要ないろいろなデータを保持しておくところで，Prolog の持つデータベース機能を利用している．ワーキングメモリーは，事実や現在の状況を保持しておくところで，同じく Prolog のデータベース機能を利用している．ユーザー・インターフェースはユーザーとのインターフェース部分であり，必要な質問を発したり，結果を提示したりする．本システムにおいては，特にグラフィック機能をサポートすることで，効果的なインターフェース機能を実現した．これら3つの部分を制御し，推論を実行するのが推論エンジン本体である．



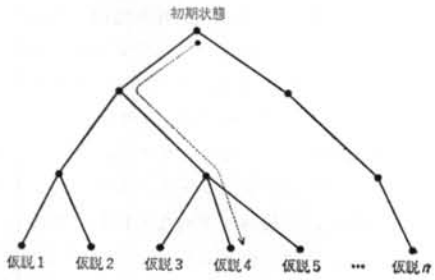
図—1　システムの構成



図—2　データの木構造



図—3　後向き推論の実行



図—4　前向き推論の実行

一般に，知識工学を応用したシステムでは推論エンジンと，データベースやワーキングメモリーが明確に分離されている．アプリケーション・システムの構築は，これらのうちデータベースを作成することを意味する．

2.1.2　分類・診断型前向き推論システム

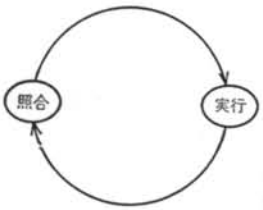
本システムは，そのアプリケーションの対象となる分野を分類・診断型のシステムとしている．ここでいう分類・診断型システムとは，あらかじめ推論の全ての結果（本システムでは，仮説という）を用意しておき，システムの指示に従って必要なデータ・事実や症状を入力することで，分類結果や診断結果を導き出すものである．

このようなシステムには，一般的に前向き推論システムと後向き推論システムの二つがある．推論の結果である仮説と，推論の経過を示すプロダクション・ルールまたは推論の途中の状態は，通常一つの木構造として表現できる（図—2）．このとき，木の枝葉の先が仮説を表わし，根が初期状態を表わす．後向き推論システムは木の枝葉の側から根に向かって推論を実行するシステムである（図—3）．すなわち，ある仮説を仮定し，データ・事実や症状により裏付けを取っていくシステムである．

後向き推論システムでは推論機構はいわゆる探索機構であり，後戻りの機能（バックトラック）を必要とする．このために推論の効率が多少悪くなる．しかし，予期せぬ事態に対しても比較的柔軟に対応ができ，柔軟度の高いシステムを作り上げることができる．一方，前向き推論システムは木の根から枝葉に向かって推論を実行する（図—4）．すなわち，初期状態からスタートし，データ・事実や症状により裏付けを取りながら，ある結果を導く．このようなシステムは推論の実行効率は良いが，予期せぬ事実に対応できないという欠点を持つ．より人間の思考に近いシステム，例えば学習機能を持つシステムなどには，多少実行効率が悪くとも柔軟性のある後向き推論システムが適していると思われる．本システムは，ある完成されたアプリケーション・システムを一般ユーザーが簡単に使用できるようにするという立場から，前向き推論システムを採用した．

2.1.3　認知・行動サイクル

知識工学を応用したシステムは，認知・行動サイクルと呼ばれる手続きを終了条件が満たされるまで行なうことで，推論の実行を制御する．本システムの推論エンジンは，次のような認知・行動サイクルによって，推論の



図—5 認知・行動サイクル

実行を制御する（**図—5**）.

(1)照合

ルールを一つ取り出し，条件部とワーキングメモリーの内容を照合し，条件部が満たされるルールを選び出す．必要があれば，ユーザー・インターフェースによる入出力が行なわれる．

(2)実行

ルールの行動部を実行する．この実行により，ワーキングメモリーの内容の更新やユーザー・インターフェースによる入出力が行なわれる．

一般的に，純粋プロダクション・システムと呼ばれるこのようなシステムでは，ある状況においてプロダクション・ルールの競合，すなわち一度に複数のプロダクション・ルールが適用可能となる状態が起こり得る．このようなとき，適用するプロダクション・ルールを決定するためのルール（メタ・ルール）により，プロダクション・ルールの競合解消を行なう．しかし，本システムは分類・診断型の推論システムであるため，基本的にはプロダクション・ルールの競合はないと仮定している．したがって，純粋プロダクション・システムのような適合ルールの競合における競合解消は行なっていない．

2.1.4 知識表現の枠組み

本システムは，知識表現の枠組みとして最も単純なプロダクション・ルール型の知識表現を採用している．すなわち，人間の持っている知識を if～then～型ルールとして記述し，さらにこのルールの記述は推論エンジンにより逐次的に処理される．逆にいえば，純粋プロダクション・システムのような局所的な状況におけるルールの記述はできない．

本システムのプロダクション・ルールは，推論結果である仮説，仮説に対する対応措置や推論の途中結果を示す中間仮説などの仮説定義と，事実やデータを表わす各種の事象定義から，その条件部と行動部が構成される．基本的には，ルールの条件部においてその時点で想定される状況を仮説定義と事象定義により表わし，ルールの行動部ではその状況における行動を，仮説定義に確信度を与えることや事象定義に新たな値を与えることにより表わす．

このような知識表現の枠組みは，表現の簡潔さと可読性の良さから，従来の計算機プログラムにあり勝ちな極端な可読性の悪さからくる不自由なメンテナンスを軽減するものである．さらに，人間の思考を単純化したモデルを提供するものでもある．

2.1.5 不確実性の取り扱い

人間の持つ知識の多くは不確実である．したがって，より人間の思考に近いシステムを構築する際，不確実性の取り扱いが必要不可欠となる．本システムでは，確信度と呼ばれる測度を導入している．確信度は，感染症診断システム MYCIN で初めて採用された不確実性を測る測度である．確信度は，閉区間［−1，1］の範囲の値を取り，次のような規則に従って生成される．

(1)全ての確信度は，正負の値に分けて計算される．

(2)全ての確信度の初期値を 0 とする（式(1)）.

$$\left.\begin{aligned} x_p{}^0 &= x_n{}^0 = 0 \\ x_0 &= x_p{}^0 + x_n{}^0 \end{aligned}\right\} \quad \cdots\cdots(1)$$

(3)プロダクション・ルールにより第 n 回目の確信度の更新は，更新される確信度を a とすると，次式のようになる．

$$\left.\begin{aligned} &a > 0 \text{ のとき,} \\ &\quad x_p{}^n = x_p{}^{n-1} + (1 - x_p{}^{n-1}) * a \\ &\quad x_n{}^n = x_n{}^{n-1} \\ &\quad x_n = x_p{}^n + x_n{}^n \\ &a < 0 \text{ のとき,} \\ &\quad x_p{}^n = x_p{}^{n-1} \\ &\quad x_n{}^n = x_n{}^{n-1} + (1 + x_n{}^{n-1}) * a \\ &\quad x_n = x_p{}^n + x_n{}^n \end{aligned}\right\} \quad \cdots\cdots(2)$$

x_i：第 i 番目の確信度

$x_p{}^i$：第 i 番目の正の確信度

$x_n{}^i$：第 i 番目の負の確信度

このような確信度は，プロダクション・ルールの実行順序に影響されず，否定的な不確実性も扱えるという長所を持つ反面，一度与えた確信度を元に戻せないという短所も持つ．本システムでは，仮説定義の持つ値を確信度とすることで，推論途中や推論結果において不確実性を扱えるようにした．

2.1.6 事実としてのデータ

本システムで扱う事実としてのデータ（事象定義データ）は，大きく分けると論理型と数値型の二つである．

論理型データは，値として真・偽の二つの値を取り得る．真はある事実・事柄が成り立つことを示し，偽は成り立たないことを示す．数値型データは，値として実数を取る．これらのデータは，初期値としていずれも“未知”という値を持っていて，システムは必要な質問を発

することにより値をユーザーから獲得する．また，ルール内の記述により値を獲得・更新することもできる．これらはワーキングメモリー上にあって，システムの推論実行中に参照される．

2.1.7 ルールとしてのデータ

ルールとしてのデータは，基本的には if～then～型ルールの集合体であり，一つのルールは条件部と行動部に分けられる．推論エンジンは，条件部が満たされる場合にのみ行動部を実行する．ルールの条件部ではすべてのデータのチェックが可能であり，行動部ではすべてのデータの更新が可能である．また，必要に応じて OS の機能も呼び出すことが可能である．

本システムは，知識表現の枠組みとしてプロダクション・ルール型の表現を採用しているが，さらにその表現能力を高め可読性を良くするために，ルールを階層化できるようにした．ルールを階層化することで，ルール群の持つ意味が把握しやすくなり，さらには推論実行時における無駄な探索を排除して，実行効率の向上にも貢献している．

2.1.8 データの書式

本システムを構成するデータ・ファイルは，

(1)仮説定義ファイル

(2)事象定義ファイル

(3)知識ベース定義ファイル

の三つである．

仮説定義ファイルでは，推論結果となる仮説，推論の途中結果を示す中間仮説と推論結果に対する措置を示す対応措置の３種を定義する．各々の書式は**表―１**のとおりである．

事象定義ファイルでは，ユーザーに対する質問項目としての事象を定義する．事象は，はい／いいえ型，選択型と数値型の３種である．このうち，はい／いいえ型と選択型は前述の論理型に属す．はい／いいえ型は質問事項一つに対して，はいまたはいいえで答えるのに対し，選択型は質問項目一つに対して，一つまたは複数の答えを選択するものである．各々の書式は**表―２**のとおりで

```
仮説        name_of_kasetu : 'format 付き出力文'.
中間仮説    name_of_chukan-kasetu : 'format 付き出力文'.
対応措置    name_of_taio-soti : 'format 付き出力文'. 'Dos_Command'.
```

表―１　仮説定義の書式

```
はい／いいえ型   name_of_jisho : 'format 付き質問文'.
選択型   'format 付き質問文',
         name_of_jisho-1 : 'format 付き内容文',
         name_of_jisho-2 : 'format 付き内容文',
                 ⋮                 ⋮
         name_of_jisho-n : 'format 付き内容文'.
数値型   name_of_jisho : 'format 付き質問文'.
```

表―２　事象定義の書式

```
知識ユニット
適用条件　(Condition),
[(if  (Condition)
  then  [(Condition  --> Action),
         (Condition  --> Action),
             ⋮               ⋮
         (Condition  --> Action)
        ]
 ),
 ⋮
 (if  (Condition)
  then  [(Condition  --> Action),
             ⋮               ⋮
         (Condition  --> Action)
        ]
```

表―３　知識ベース定義の書式

仮説	hg (name_of_kasetu, 数１：数２)
中間仮説	hm (name_of_chukan-kasetu, 数１：数２)
対応措置	hc (name_of_taio-soti, 数１：数２)
はい／いいえ型	fb (name_of_jisho, True_False)
選択型	fb (name_of_jisho, True_False)
数値型	fn (name_of_jisho, 数１：数２)
処理	mg (Dos_Command)
AND 結合	&
COUNT 結合	(Number : : FH(,), …, FH(,))
条件なし	nil

(注)　数１：数２は，閉間区間［数１，数２）を表わす
数１，数２は，＊により制限を外すことができる
True_False は，t（=True）または f（=False）のどちらかである
COUNT 結合は，以下のリストから該当するものを一つ以上選択するものである

表―４　知識ベースの条件部

仮説	hg (name_of_kasetu, 数)
中間仮説	hm (name_of_chukan-kasetu, 数)
対応措置	hc (name_of_taio-soti, 数)
はい／いいえ型	fb (name_of_jisho, True_False)
選択型	fb (name_of_jisho, True_False)
数値型	fn (name_of_jisho, 数)
処理	mg (Dos_Command)
AND 結合	,
行動なし	nil

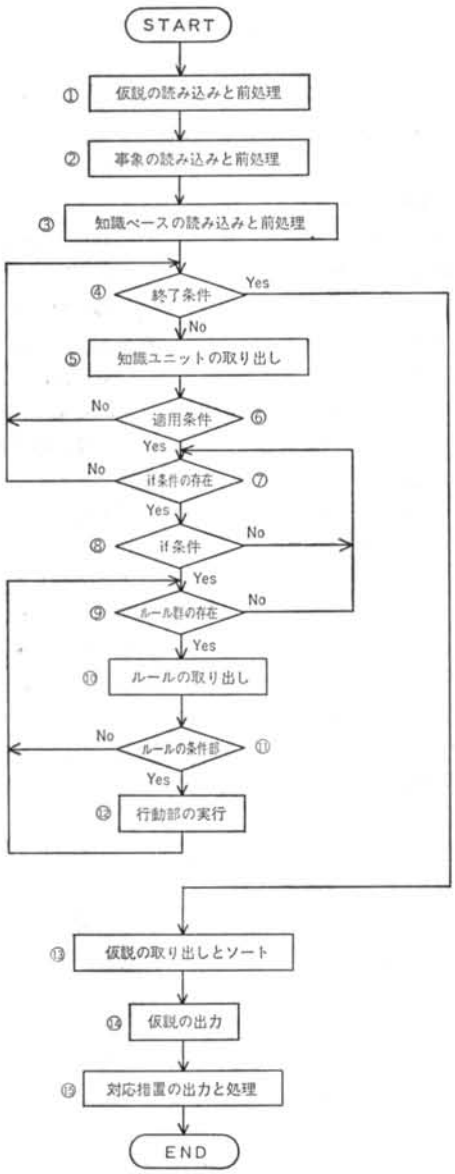
表―５　知識ベースの行動部

ある．

知識ベース定義ファイルでは，プロダクション・ルール群である知識ユニットを定義する．この書式は**表—3**のとおりである．知識ユニットにおけるConditionおよびActionは，各々**表—4**，**表—5**のようなものである．

2.1.9 推論の実行アルゴリズム

本システムにおける推論は，以下のようなアルゴリズムに従って実行される（**図—6**）．



図—6 推論の実行アルゴリズム

①仮説定義ファイルを読み込み，仮説，中間仮説と対応措置に対して初期値0を与え，ワーキングメモリー上に保持する．

②事象定義ファイルを読み込み，各種の事象定義をデータベース上に保持する．

③知識ベース定義ファイルを読み込み，ルールをデータベース上に保持する．

④実行の終了条件（ルールが存在しない状態）が満たされたら⑬へ，満たされないなら⑤へ．

⑤ルール群の単位である知識ユニットを一つ取り出す．

⑥第1階層の条件（適用条件）をチェックし，満たされたら⑦へ，満たされないなら④へ．

⑦第2階層の条件（if条件）があれば⑧へ，なければ④へ．

⑧第2階層の条件をチェックし，満たされたら⑨へ，満たされないなら⑦へ．

⑨第3階層のルール群があれば⑩へ，なければ⑦へ．

⑩ルール群からルールを一つ取り出す．

⑪ルールの条件部をチェックし，満たされれば⑫へ，満たされなければ⑨へ．

⑫ルールの行動部を実行し，⑨へ戻る．

⑬ワーキングメモリー上の仮説で，確信度がある基準値以上のものを取り出し，ソートする．

⑭結果として最大確信度の仮説を出力し，他の仮説を候補として出力する．

⑮ワーキングメモリー上の最大確信度の対応措置を出力し，何らかの処理があれば実行する．

各種条件のチェックにおいて，データベース上の事象に対しては，ユーザー・インターフェースによりその値を獲得し，データベースからワーキングメモリーへ保持場所を変更する．それ以降，ワーキングメモリーを参照して推論を実行する．

2.2 開発支援用システム

本システムを用いたアプリケーション・システムを構築するために，開発支援用システムとして二つのサブシステム（Excel 2, Elint）を開発した．ここでは，これらのサブシステムの概要について述べる．

2.2.1 Excel 2について

開発支援システム（Excel 2）は，実行用前向き推論システム（Excel）とほぼ同じ機能を持つ．違いは，Excelがすべてのデータをバイナリー形式でシステムのデータベース内に保持することで，実行速度の高速化を図って

いるのに対し，Excel 2 はデータをテキスト形式のまま扱うことである．このため，Excel 2 ではデータの修正・変更がエディタを用いることで比較的楽にできる．したがって，Excel 2 は Excel を用いてアプリケーション・システムとして実用システムとする前に，開発支援用としてすべてのデータをテストするためのものである．

2.2.2 Elint について

本システムにおいて仮説定義・事象定義や知識ベース定義（ルール群）は，ある一定の形式で表現しなければならない．これらの表現は Prolog のプログラムとしては一般的な形式であるが，Prolog を知らない一般のユーザーにはなじみにくい形式である．そこで，これらのデータを事前にチェックするためのサブシステム（Elint）を開発した．

このようなサブシステムとしては，

(1)シンタックス・チェック機能を持つ構造エディタ

(2)エディタとは独立したプリ・プロセッサー型のシンタックス・チェッカー

の2通りが考えられる．構造エディタは，そのユーザー・インターフェースを十分に検討した上で実現すればかなり使いやすいものとなるが，システムとしては複雑になってしまう．一方，プリ・プロセッサーはシステムの実現も比較的楽であり，かなりの機能を期待できる．アプリケーション・システムとして大規模システムを想定するときには，作業効率などからもプリ・プロセッサーの方が簡単かつ有効であると考えられる．

本システムの開発支援用システム（Elint）は，Prolog によるプリ・プロセッサーとして実現した．

§ 3. 論理型言語 Prolog の特徴

システムの実現に当たり，プログラミング言語として Prolog を採用した．Prolog は，1972年フランスで生まれた比較的新しい言語である．その言語機能は，従来の手続き型言語と大きく異なり，パターン・マッチ，バックトラックの機能を持ち，柔軟なリスト処理が可能である．これらの機能が，人間の思考に近いシステムを実現する上で非常に役に立つ．

ここでは，Prolog の閉世界仮説，パターン・マッチやバックトラックと枝刈りの機能について，システムの実現を踏まえて述べる．

3.1 Prolog の閉世界仮説

Prolog は **Pro**gramming in **Log**ic の略であり，論理学を基にする言語である．論理学では，その値として真理値を用いる．Prolog においても同様であり，真と偽の二つの値によってプログラムの実行が制御される．このとき，Prolog では閉世界仮説によりデータベース中に存在する事実のみを真とし，それ以外を全て偽として扱う．

本システムにおいては，システムの初期状態として特に“未知”という状態が必要であるため，これを偽として扱い，各事象の成否はともに真として，明示的に扱っている．すなわち，ある事象が未知のときにはユーザー・インターフェースによる入出力によって値を獲得し，ワーキングメモリーに事実として追加する．以後は，ワーキングメモリー内の値を用いて推論を実行する．

3.2 パターン・マッチ

パターン・マッチとは，ある構造化された文字列同士の比較である．構造化された文字列とは，文字列の並びを括弧でくくったもの（リスト）や，それらの入れ子構造化されたものをいう．従来の手続き型言語では数値化されたデータか，単なる文字または文字列の比較しかできない．また，比較する対象は基本的に同じ型でなければならない．これに対し Prolog では，変数に型の概念がないためパターン・マッチの際に型にとらわれることなく，どのような構造の文字列も一つの変数とパターン・マッチが可能である．

一般に，人間の思考は数値的というよりはむしろ記号的であるといわれる．このような思考過程に対して，数値的比較はほとんど用をなさない．型に束縛されず，自由に構造を変化させることのできる柔軟なリスト処理とパターン・マッチの機能は，このようなより人間の思考に近いシステムの実現に大いに役に立つ．具体的には，ルールの条件部のチェックと，データベース上からのルールの取り出しにこの機能を用いている．

3.3 バックトラックと枝刈り

バックトラックは，探索過程において探索の失敗を補うために，必要なところまで後戻りすることである．また，枝刈りはバックトラック時の不要な探索を行なわないように，強制的に木構造の枝葉を刈り取って捨ててしまうことである．Prolog 自身は後向きの探索機構とバックトラックの機能を持っているため，探索機構を利用したシステムの実現が非常に楽である．さらに，Prolog ではすべての変数がローカル変数であり，グローバル変数は存在しない．加えて，変数への代入という概念を持た

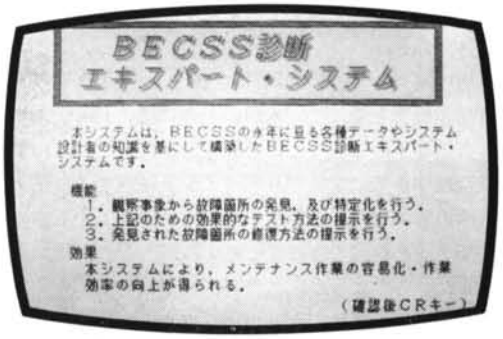
ない．変数は，いったんある値に束縛（ユニミァイ）されるとバックトラックに依らない限りその値を変えることはできない．

本システムは，前向き推論システムを採用しているので，大きなバックトラックは基本的には起こり得ない．むしろ，バックトラックに依存する戻り場所の決定に枝刈り機能を多く用いている．不要な枝を刈ることはシステムの見通しを良くし，実行効率の向上にもつながる．具体的には，ルールの条件部が満たされないときの行動部のキャンセルなどに用いている．また，変数への代入ができないことから，ある特殊なバックトラックによる変数の値の更新を行なっている．

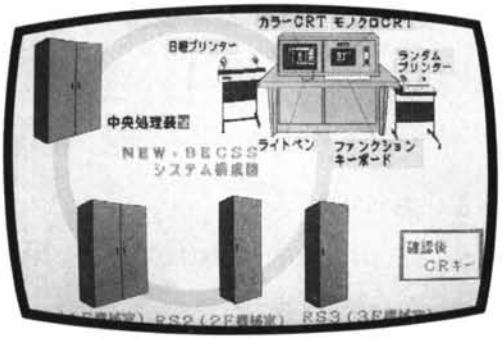
§ 4. BECSS 診断システムへの応用

ここでは，本システムの実際の応用例として BECSS 診断システムについて述べる．

BECSS は，当社が自社開発したビル環境制御システムで，マイクロ・コンピュータによる分散型の監視制御システムである．BECSS 診断システムは，BECSS の永年にわたる障害データやシステム設計者の知識を基にして，経験の浅い建築設備保守員が障害要因の探索を楽に行なえるように支援するシステムである．このシステ

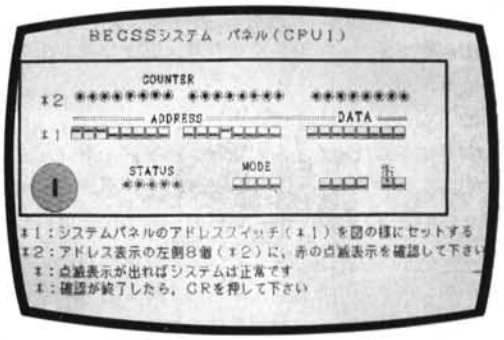


写真—1 Excel による BECSS 診断システムの実行例(1)

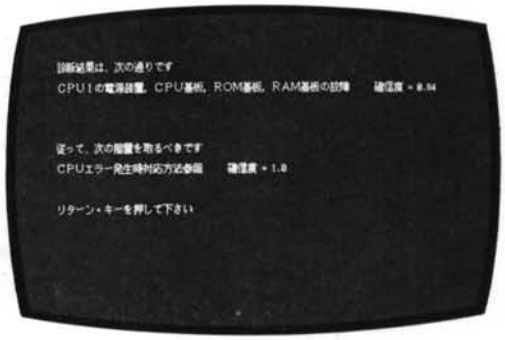


写真—2 Excel による BECSS 診断システムの実行例(2)

写真—3 Excel による BECSS 診断システムの実行例(3)

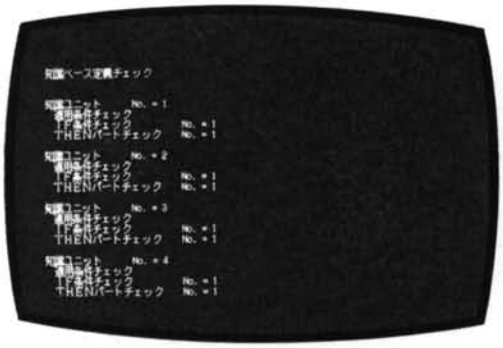


写真—4 Excel による BECSS 診断システムの実行例(4)



写真—5 Excel による BECSS 診断システムの実行例(5)

写真—6 Excel による BECSS 診断システムの実行例(6)



写真－7　Elint によるデータチェックの実行例(1)

写真－8　Elint によるデータチェックの実行例(2)

ムは，建築設備保守員との対話を通して以下の三つの機能を提供する．

(1)観察事象から故障箇所の発見，および特定化を行なう．

(2)上記の効果的なテスト方法の提示を行なう．

(3)発見された故障箇所の修復方法の提示を行なう．

制御システムとしての BECSS はバックアップ機能を備えているが，故障の修復には建築設備保守員が当らねばならない．このようなとき，経験の浅い建築設備保守員が本システムに障害状況を入力することで故障箇所の発見を行ない，さらに修復方法を知ることができる．

このシステムは，システム設計の段階で必要な機能を見きわめ，知識の獲得・体系化を行なうのにかなりの人工を費やした．また，このシステムは当初大型計算機を用いて開発されたが，実際の使用環境を整えるのにコストがかかり過ぎることから，パーソナル・コンピュータ上に移植されたものである．移植に当たり本システムの前身であるプロトタイプ・システム（DoT）を用い，その後知識の見直しなどに本システムを用いた．**写真**－1～6 に，BECSS 診断システムの実行例と開発支援用システム（Elint）によるデータチェックの実行例を示す．

§ 5.　まとめと考察

本システムは，BECSS 診断システム用に開発されたプロトタイプ・システム（DoT）が原型である．DoT では，大型計算機で開発されたシステムを，パーソナル・コンピュータ上で稼働させるという実際の使用環境のみを考慮したため，新たな別のアプリケーション・システムを構築することが困難であった．特に，シンタックス・チェック機能を持たないことが，アプリケーション・システムの検証を困難にしていた．本システムは，それらの経験を踏まえて知識表現の枠組みに若干の修正を加え，ユーザー・インターフェースを拡張し，さらにシンタックス・チェッカーを備えることでシステムとしての完成度を高めたものである．特に，開発支援用システム（Excel 2, Elint）を備えることで，本システム単独によるアプリケーション・システムの開発が可能となった．また，実行速度の面でも実用化レベルの速度が確保されている．

本システムの開発とアプリケーション・システムへの応用を通して，次のような知見が得られた．

(1)論理型言語 Prolog の有効性

(2)知識工学を応用したシステムのパーソナル・コンピュータ上での実現の実用性

(3)グラフィック機能を用いたユーザー・インターフェースの効果

Prolog は，従来の計算機主体というプログラミング言語の概念を変えて，人間の思考に近いシステムのための有効な手段であることが確認できた．これは，Prolog が論理学を基にするプログラミング言語であるために，人間の思考過程を計算機上でシミュレートする際に有効なモデルを構築できることによる．

また，そのようなシステムがパーソナル・コンピュータを用いても開発でき，実用レベルをも確保できることが，実際のアプリケーション・システムへの応用を通じて確認できた．さらに，ユーザー・インターフェースとしてのグラフィック機能は実用性を高め，ユーザーに十分な効果をもたらすことも確認することができた．

§ 6.　おわりに

本システムは，実行用・入門用として十分に実用に耐え得るシステムである．しかし，技術の進歩は日進月歩である．今後に向けての課題も多い．具体的には，

(1)知識表現の枠組みを拡大する（例えば，フレーム表

現など),

(2)開発支援環境を強化する（例えば，構造エディタを用いた知識エディタ，トレース機能など),

がある．今後もこれらの課題に取り組み，パーソナル・コンピュータ上におけるユーザー環境を整備・充実させていくつもりである．

<参考文献>

1) 小林重信："知識工学の基礎と応用（第1回）知識の表現と利用(1)" 計測と制御 Vol. 24, No. 2 (1985年) pp. 155～164
2) 小林重信："知識工学の基礎と応用（第2回）知識の表現と利用(2)" 計測と制御 Vol. 24, No. 3 (1985年) pp. 242～250
3) 畝見達夫："知識工学の基礎と応用（第3回）Prolog による知識ベースシステムの実現" 計測と制御 Vol. 24, No. 5 (1985年) pp. 439～448
4) 小林重信："知識工学の基礎と応用（第5回）エキスパートシステム(1)" 計測と制御 Vol. 24, No. 8 (1985年) pp. 730～738
5) 小林重信："知識工学の基礎と応用（第6回）エキスパートシステム(2)" 計測と制御 Vol. 24, No. 9 (1985年) pp. 833～840
6) 小林重信："知識工学" 昭晃堂 (1986年)
7) 上野晴樹："知識工学入門" オーム社 (1985年)
8) 中島秀之："Prolog" 産業図書 (1983年)
9) 中島秀之："知識表現と Prolog/KR" 産業図書 (1985年)