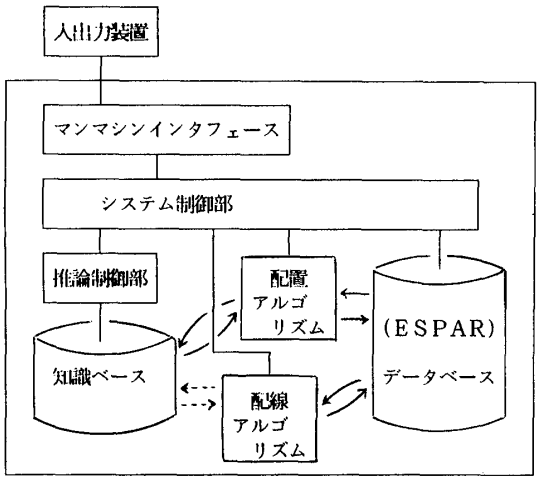
6G-1

C言語埋め込み型エキスパートシェル CeiShell

吉村　宏之　　土田　雅之　　阪本　清美
大岩　陽子　　植村　博一　　青江　秀雄
松下電器産業株式会社　情報システム研究所

1．はじめに

このエキスパートシェルはプリント基板自動配置配線エキスパートシステムＥＳＰＡＲ（ Expert System for Placement And Routing ）の配置・配線知識を推論するアプリケーションソフトとして開発した。プリント基板自動配置配線エキスパートシステムＥＳＰＡＲ（以下ＥＳＰＡＲと呼ぶ）はプリント基板の配置・配線をアルゴリズムのみでなく設計者の知識を取り入れることにより配線率の高いシステムを実現しようとするものである。我々は最初このエキスパートシステムのプロトタイプとして配置の知識処理をＬｉｓｐ言語で記述されたエキスパートシェルを用いることによって推論を行い，ある程度の結果を得る事ができた。しかし，推論部をＬｉｓｐ言語のエキスパートシェルで行っているために部品の自動配置に非常に時間がかかりまた多くのメモリを必要とした。そこでこのエキスパートシステムの実用化を考えた場合，上記の問題を克服するためにＬｉｓｐ言語ではなくＣ言語により記述し，またＥＳＰＡＲの配置・配線の知識を表現，処理するのにもっと適したエキスパートシェルを開発したので報告する。



第1図　ESPAR構成図

2．システムにおけるエキスパートシェルの位置付け

今回開発したエキスパートシェルはそれのみで推論を行わせるような閉じた汎用的なエキスパートシェルではなくＥＳＰＡＲの配置・配線の知識処理を行うのに必要なシステムにあったアプリケーションソフトである。よってこのエキスパートシェルはＥＳＰＡＲのシステム内に埋め込まれた形になっており，これがこのエキスパートシェルの特長となっている。

このエキスパートシェルの位置付けを示すＥＳＰＡＲの構成図を第１図に示す。

3．自動配置・配線にあった設計型推論機能

このエキスパートシェルはルール主導で前向きの推論を行う。基本的にプリント基板の自動配置・配線にあったアプリケーションソフトという立場で開発されたので以下のような特長を持つ。

a．汎用ワーキングメモリを持たない

このエキスパートシェルは汎用のワーキングメモリを持たない。なぜなら汎用のワーキングメモリにはＣＡＤのような複雑で大量のデータ構造が記述しきれないからである。そこでこのエキスパートシェルをシステムに埋め込むことにより推論のためにデータが必要ならば直接データベースを操作できるようにした。よって汎用的なワーキングメモリが必要でなくなるのでルールの各要素はマクロ（後述）を通じてすべて配置・配線に関する手続き関数（アプリケーション関数と呼ぶ）を呼び出すという形をとっている。

b．配置の知識にあったルール及び制御

第２図に配置・配線のうち配置に関するルールが記述されているＫＳ（Knowledge Source）例を示す。例にあるようにルールは基本的に条件部（if 部）で部品の選択をし，実行部（then部）で部品の配置を行うという形をとっている。また実行部の最後のエクスクラメーションマークはそのルールが発火して部品の配置が実行されたら今のＫＳを抜けるという配置にあった制御を行っている。

CeiShell : C-language Expert Imbedded Shell
Hiroyuki Yoshimura, Masayuki Tsuchida, Kiyomi Sakamoto
Youko Ohiwa, Hirokazu Uemura, Hideo Aoe
Information Systems Research Laboratory
Matsushita Electric Industrial Co.,Ltd.

4．エキスパートシェルの知識ベース

このエキスパートシェルは知識ベースとして，プロダクションルールが記述されているＫＳ、ＫＳのルールの各要素のマクロ定義が記述されているマクロ、推論に必要なデータが記述されているフレーム（Ｍ．Ｍｉｎｓｋｙの提唱するフレームとは異なる）の３つの知識が必要となる。これらの知識（ＫＳ，マクロ，フレーム）をここでは知識ユニットと呼ぶ。それぞれの知識は日本語での表現が可能である。

１）ＫＳ

ＫＳには複数のルールがルール名とともに記述されユーザーが自分で考えた表現で自由に知識が記述される。このためＫＳは見やすいものとなり，どのような推論が行われているか一目でわかる。

ＥＳＰＡＲではこのＫＳにはいくつかの基板に共通な知識を記述している。

２）マクロ

ＫＳで記述されている条件部，実行部の各要素はすべてマクロ定義されていなければならない。このマクロにはＫＳで記述しきれない細かな知識やＫＳに書くとかえって繁雑になるような知識を記述する。このマクロの中では既にシステム側に用意してある配置・配線のためのアプリケーション関数のどれを呼び出すかを定義することになる。

第３図にマクロ例を示す。

３）フレーム

フレームとは推論に必要と思われる知識を格納しておくところである。フレームには知識の単位ごとにフレーム名を与え，それをひとつのユニットとする。フレームはいくつかの知識よりなっているのでそれぞれの知識にスロット名と値を与える。フレームからの知識の取り出しは実際にはアプリケーション関数の中で行われる。第４図にフレーム例を示す。

ＥＳＰＡＲではこのフレームにはその基板固有の知識を記述している。

```
[ KS: ＩＣ配置ＫＳ
      •
      •
 CPU配置:
 ( CPU選択 )
  -->
  ( CPU配置 )!
      •
      •
      •
]
```

第2図 ＫＳ

```
[ MACRO:
  ( CPU選択 )
  ==
    {
    int ans ;
    call sel_cpu ( ans ) ;
    return ( ans ) ;
    }

  ( CPU配置 )
  ==
   {
    call put_cpu ( ) ;
   }
 ]
```

第3図 マクロ

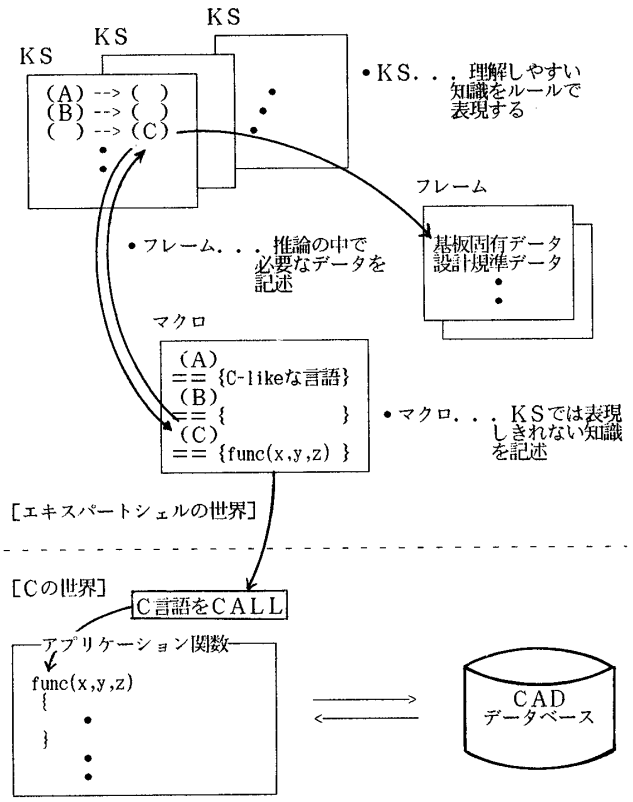
```
[ FRAME: CPU

  cpu1 : 80386

  cpu2 : 68030

  cpu3 : MN1617

       •
       •
       •
]
```

第4図 フレーム

5．知識ベースとシステム側との関係

上で説明した３つの知識ユニットとシステム側との関係をあらわす概念図を第５図に示す。一つのＫＳが実行されるとそのＫＳ内のルールの順番に評価されていく。このルールが評価されていく中で知識が必要な場合は自由にフレームから知識を取りだされる。

ルールの各要素はマクロ定義されているのでマクロで記述されているシステム側のアプリケーション関数がＣＡＬＬされる。このアプリケーション関数の中で推論に必要なＣＡＤデータベースやフレームの内容を参照したり，アルゴリズムによる配置・配線の手続き処理が行われる。このアプリケーション関数の処理が終わると再びマクロに戻りマクロが終わるとＫＳ内の次の要素に移る。



第5図 知識ベースとシステム側との関係概念図

6．まとめ

このエキスパートシェルにより

１．Ｃ言語によるエキスパートシェルのため少ないメモリで高速な推論が可能

２．知識の記述が見やすく，特にＥＳＰＡＲの自動配置にあった知識表現に向いている

３．システムに埋め込まれたエキスパートシェルのためシステム内を自由に操作可能

などが実現できた。

今後ＥＳＰＡＲにあった設計型のエキスパートシェルとしての機能をさらに強化していく予定である。