数理解析研究所講究録
第 495 巻 1983 年 87-107

フレーム型知識表現言語の設計およびインプリメンテーションについて

東京電機大学　理工学部　　上野晴樹（ Haruki Ueno ）
伊藤秀昭（ Hideaki Ito ）

1. はじめに

問題領域の専門知識を用いて、コンピュータにより問題解決を行なうシステムを知識型システム（Knowledge Based System ）と呼ぶが、この種のシステムに対する社会のニーズが最近急速に高まってきた。この様なシステムの開発に必要なプログラム言語は、従来の手続き中心のものではなく、専門知識を定義し、利用するために特別に設計された、いわゆる、知識表現言語である。

代表的な知識表現言語には、知識をルールとして表したルール型システムEMYCIN（20）、ルールと階層仮説構造を組み合せたブラック・ボード型システムAGE（21）、Minsky のフレーム理論に基づくいわゆるフレーム型システムUNITS（ 7,9）、述語論理に基づく論理型言語PROLOG（23）等がある。

FMS（Frame Manipulation System ）（22）は、フレーム型知識表現言語の一つであり、AIもしくは知識工学を研究するコンピュータサイエンティストのための実験ツールとして開発された。システムのユーザは、FMSを用いて標準的な階層型フレーム・システムを表現し操作するだけでなく、プロダクション・システム、因果ネットワーク、セマンティック・ネットワーク等の色々な知識表現モデルおよびシステムを実験することができる。FMSがこの様に極めて汎用な機能を持っているのは、このシステムが汎用性の高い階層型データ構造およびユーザ指定の付加手続（attached procedures）から成り、特別な推論機構を持たないことに由来している。FMSを用いて応用エキスパート・システムを開発し利用することは勿論可能であり、複雑で大量の階層知識を取り扱うシステムには特に強力かつ実用的である。

我々は、研究開発用の知識表現言語としてはMinsky のフレーム理論に基づくシステムが適していると考え、この理論に基づいて、フレーム操作システムFMSを開発した。これを用いてCOMEXモデルを書くことにより、システムの柔軟性と汎用性を確認した。なお、FMSはUNITS等を参考にして設計され、UTILISP（19）で書かれており、FACOM M－ 160上で稼動している。ここでは、設計の概念、システムの概要、インプリメンテーション等について述べる。

2. 知識の表現

一般に知識型システムでは、知識は推論機構（ Inference Engine ）から独立した知

識ベース（Knowledge Base ）内に置かれる。この両者を切り離すことにより、知識定義や更新を容易に行なうことが可能となり、システムの柔軟性および汎用性を達成することが可能となる。

フレーム・モデルは階層的データ構造である。一つのフレームは一つの概念対象を表わし、その概念の具体的性質や事実の記述はフレームの構成要素であるスロットに記入される。知識概念には抽象的な階層性があるので、全体の知識は一つの階層構造を持ったフレーム・システムとして構築される。これが知識ベースを構成する。フレーム型の他の特徴は、特定のフレームに特定の推論手続きをattached procedure として定義できることである。これもスロットの一種として取りあつかえる。更に、他のモデルと違って特定の推論機構を持っていない点が重要である。システムの利用者が目的に応じて自由に推論機構を設計できるが、このときattached procedure は柔軟な推論を実現する上で極めて有効な手段として、他のモデルに対する補助的ツールとすることも可能である。

以上の様な点から、研究開発ツールとして、フレーム型システムが最適であると考えられる。FMS開発の目的はここにある。但しモデルが柔軟かつ汎用であるということは、裏を返すと利用者の負担が重いということでもある。また、FMSはＬＩＳＰで書かれているが、これは、データ構造が極めて柔軟なことと、利用者がattached procedure として自由に手続きを定義できることを可能にすること、の理由による。

3. FMSの概要

フレーム型システムが研究用汎用ツールとしてすぐれていると考えられるのは、次にあげるような知識表現に関する要求事項をほぼ満たすと思われるからである。

- 複雑で大量の知識を体系的に表現し管理するためには、各知識は概念対象の周辺に組織すべきである。
- 柔軟性を高めるためには、定義型知識ばかりでなく、手続き型知識もそれと関連する概対象の記述として、表現できることが望ましい。
- 概念は一般に階層構造を持っているので、知識表現にも階層構造を導入すべきである。
- エキスパートによる問題解決においては、複雑の論理手順が、局面に応じて、見合わせて適用されると思われる。知識表現もこれに対応できる機能を持つ必要がある。
- 適切な推論の種類は、対象問題の性質、システムの目的、および問題解決のプロセスによって異なると考えられる。従って、推論の方法が固定されていたり、特定の問題領域のみに向いているというツールは、試行錯誤をともなう研究開発用ツールに適さない。

3.1 FMS設計の基本方針

設計の基本方針として考えたことは次のようである。

- 汎用知識ベースに必要と考えられる上記の要求事項を基本的に備えているか、もしくは容易に追加できること。
- フレーム型システムとして必要な基本的機能を備えたコンパクトなシステムであること。
- 知識獲得の機能はエキスパート・システムにとって最も重要な要素である。従って、これを支援するためのインターフェイスを持つこと。
- 特定の推論技法を固定せず、当面は利用者定義の推論手続きとattached proceduresによって推論機能を実現する。
- ケース・スタディを通じて、システムの拡張部分の開発を行なう。

データ構造の設計に関して考慮した点を次に述べる。フレーム型知識ベースは、一つの階層構造を持ったフレーム・システムとして構築される。各フレームはそれぞれ一つの概念対象（conceptual objects ）を記述したものであり、記述の具体的情報はフレーム構成要素であるスロットに記入される。階層の上位の方は抽象度の高い概念であり、下位の方はより具体的な概念を表わす。従って最下位のフレームは個々の対象を具体的に記述したものである。

このような考え方に基づいて知識体系を記述するために、一般に、上位フレームのスロットには、そのフレームの下位に位置するフレームに共通の事実、あるいはそのスロットに対応する下位フレームのスロットの属性もしくは値に対する制約条件が記入される。これは、知識の定義、更新や利用をスムーズにするとともに、矛盾や混乱を防止するのに役立っている。他に望まれる機能として、フレーム・システムは一つの階層構造を持ったフレームの集合であるが、フレーム間にも参照関係が成立し、ポインタを用いることによって、ネットワーク構造を持った知識体系を表現することができる。

これらの機能を組合せることにより、複雑で大量の専門知識を知識ベースとして構築でき、しかも極めて柔軟で強力な推論を実現できると考えられる。

3.2 フレームの構造

図1にフレームの基本構造を示す。各フレームは、そのフレーム・システムにおいてユニークなフレーム名をもち、複数のスロットから構成されている。その中の9つは、全てのフレームがもつ共通のスロットである。：Frame－Type スロットはそのフレームが最下位に位置するinstantiation か否かを示す情報、A－kind－ofスロットは親フレームへのポインタ、D. Descendentsスロットは子フレームへのポインタ・リスト、Description－information スロットは、そのフレームに関する備考などのテキスト値、Create スロットおよびModify スロットは、そのフレームがいつ誰により生成・修正されたかを示す。If －needed・If －added ・If －removed スロットは、それぞれあるスロットの値が必要となった時・書き込まれた時・削除される時にデモンとなる関数を示したもので

ある。上記以外のスロットは、目的に応じて自由に定義されるものであり、数の制限はない。各スロットは、同図下に示されたように、現在のところ、6つの要素から構成されている。すなわち、スロット名、インヘリタンス・ロール、データ型、データ値、デフォルトおよびオプションから成る。

スロット名： そのフレーム内においてユニークなスロット識別名である。

インヘリタンス・ロール： 現在、インヘリタンス・ロールは、U（Unique ）、S（Same ）、R（Restriction）、I（Independent）、M（Member ）、およびO（Option ）の6種を持っている。インヘリタンス・ロールの概念および具体的機能は後述する。

データ型： このスロットがもつデータ値のタイプの指定である。現在、ATOM、TEXT、TABLE、BOOLEAN、LISP、FRAME、LIST、RANGE、INTEGER、NUMBER、NUMERIC、STRING、EXPR、FSYSTEM、FLIST、の15種を持っている。データ型は、各スロットの値部がとる値の属性を指定するものである。ここでは特に重要な4種のデータ型の意味について述べる。詳細は文献（22）参照。

LISP： attached procedures。定義されると関数名が値部に記入され、本体はLISPPROCフレームにLAMBDA式として登録される。

FRAME： 他のフレーム名。これを用いてフレーム間のリンクを表現することができる。A-kind-ofスロットのデータ型もFRAMEである。

EXPR： LAMBDA式本体を格納するためのデータ型。LISPPROCフレーム中の関数本体を格納するためのスロットのデータ型はEXPRとなっている。

FLIST： フレーム名のみから構成されるリスト、NILも可。もちろん、D.Descendentsスロットのデータ型はFLISTとなっている。

デフォルト部： そのスロットにおいて値が決定されていない場合にデフォルト値を記入することができる。しかし値とデフォルトを同時に記入することはできない。

なお、オプション部は、システム利用者が自由に用いることができるように融通性をもたしたものである。システムは何のチェックもしないので十分注意しなければならない。短期記憶（STM）の代用や、そのスロットに対するメッセージ・コメント等の格納、等のために利用できる。

3.3 推論の制御

FMSは特定の推論制御機構を持たない。FMSにおける推論の制御は一般に次のように行なわれる。図2がそのメカニズムを示すが、ここでKBは知識ベース（フレーム・システム）を、IEは利用者が書いた推論制御プログラムを意味する。推論はIEを起動されることによって開始される。IEが特定のフレームに、attached procedure を持ちスロット名をパラメータとしてメッセージを送ると、そのprocedure が起動され、更に必要

Frame-name	Frame-type
A-kind-of slot	
D.Descendants slot	
Description-information slot	
Create slot	
Modify slot	
If-needed slot	
If-added slot	
If-removed slot	
Slot 1	
Slot 2	
...	
...	
Slot n	

Slot-name	Inheritance-role	Data-type	Value	Default	Option

図1　　フレームの基本構造

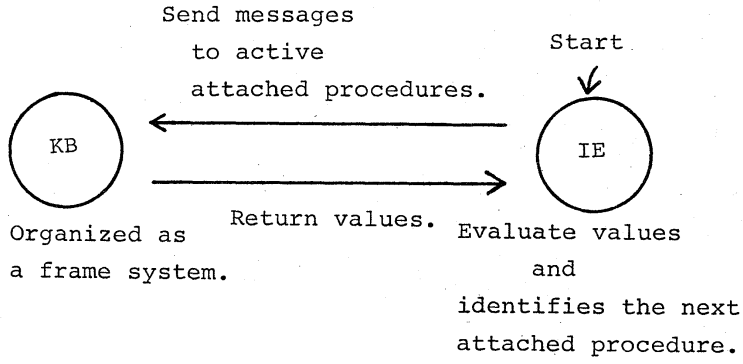

図2　　汎用フレーム・モデルにおける推論のメカニズム

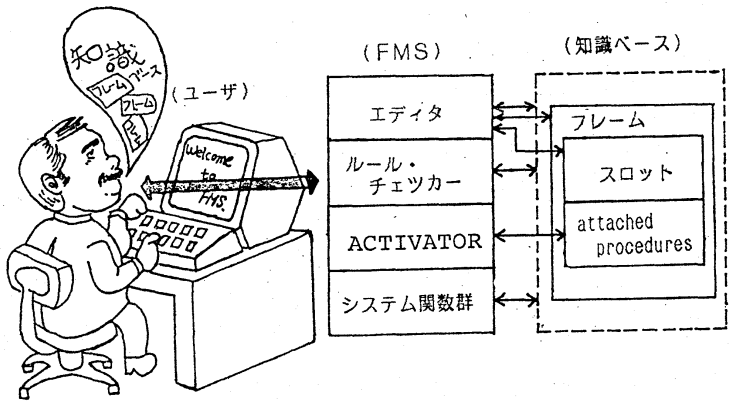

図3　　FMSシステムの構成

なフレームにメッセージを送る、というとこを繰り返してgoalに近付いていく。

ローカルな推論のための情報と手続き、注視点の制御（focus of attention）のための情報、終了状態の評価のための情報や手続きなどを、ＫＢとＩＥにどのように分担させるかは、問題の性質やシステム利用者の方針によって大きく異なる。一般的には、なるべく定式化してＫＢにもたせる方が、ＩＥがスッキリしたものとなり、システム全体の構成も解りやすくなると思われる。

3.4 プログラムの構成

ＦＭＳシステムは、４つの主なモジュールから構成されている（図３）。それらは、フレーム・エディタ、ルール・チェッカ、ＡＣＴＩＶＡＴＯＲ、およびシステム関数群である。

フレーム・エディタは、知識ベースの定義や更新など、フレーム・システムの構築のために用いられる。推論制御プログラムもこれを用いて書かれる。これは知識ベースの管理において、ユーザ・インターフェイスとなる。

ルール・チェッカは、フレーム・エディタを用て構築された知識ベースの規約上のミスを検出するために用いられる。部分的なミスは、フレーム・エディタでも検出されるので、これは主に全体的な矛盾などの検出を行なう。

ＡＣＴＩＶＡＴＯＲは、ＩＥを起動するためのものである。システム関数群は、attached procedures等を書き易くするために備えられているＬＩＳＰ関数群である。但しこれらは共通関数であるので、システム自身も使っている。

4. インプリメンテーション

4.1 インプリメンテーションの基本方針

インプリメンテーションの基本方針は次のようなものである。

- ＦＭＳ設計の方針を順守する。
- 一つの関数は、機能的に独立なものでなければならない。
- システムの拡張に柔軟に対応できる関数使用とする。
- 人工知能研究用システムとして絶え得るようなリスト処理用関数を整備する。
- ユーザとのインターフェイスの統一。
- 知識ベースの構造が変更されても、柔軟に対応することができる。

また上記以外に考慮したことは、“ＦＭＳは、ソフトウェア・システムとしてすべての関数、プログラムが階層的に構成されていなければならない。”という点である。

ＦＭＳの機能的な中心部となるのはフレーム・エディタである。我々はこの中で用いられる関数を中心にＦＭＳシステムの関数の設計を進めていった。このような開発手順を取ったのは、フレーム・エディタの各機能が個々に独立なものであり、かつフレーム・エデ

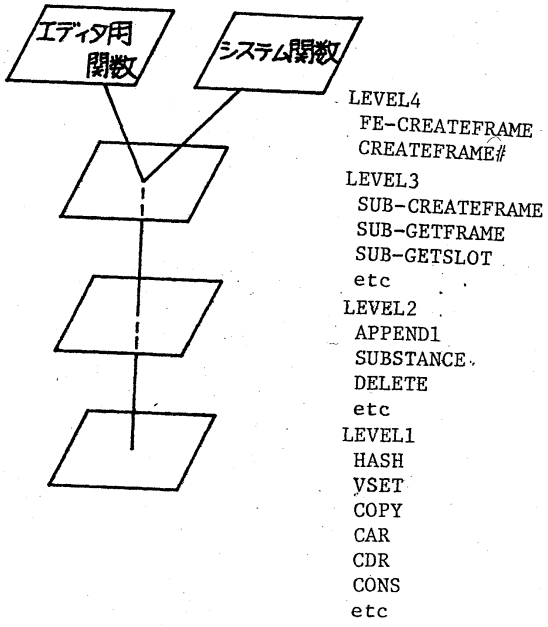

図4　　関数の階層的分類、およびその例

```
(MCTD SUBCLASS
 (A-KIND-OF U ROOT FRAME DISEASE *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (DDESCENDENT
    U
    ROOT
    FLIST
    (MCTD-DEF-A MCTD-PRO-A MCTD-PRO-B MCTD-POS-A MCTD-POS-B MCTD-POS-C)
    *DEFAULT-NOT-DEFINE*
    *OPTION-NOT-DEFINE*)
 (DESINF U ROOT TEXT "Mixed connective tissue disease." *DEFAULT-NOT-DEFINE*
  *OPTION-NOT-DEFINE*)
 (CREATE U
         ROOT
         STRING
         ("G00126" "8212272114")
         *DEFAULT-NOT-DEFINE*
         *OPTION-NOT-DEFINE*)
 (MODIFY U
         ROOT
         STRING
         ("G00126" "8212301716")
         *DEFAULT-NOT-DEFINE*
         *OPTION-NOT-DEFINE*)
 (IF-NEEDED U ROOT LIST NIL *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (IF-ADDED U ROOT LIST NIL *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (IF-REMOVED U ROOT LIST NIL *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (CONCLUSION U DISEASE LIST *VALUE-NOT-DEFINE* *DEFAULT-NOT-DEFINE*
  *OPTION-NOT-DEFINE*)
 (MACR S
       *TOP*
       LIST
       (RYES MYOS DC07 SW00 SCLD ENAP)
       *DEFAULT-NOT-DEFINE*
       *OPTION-NOT-DEFINE*)
 (MICR S
       *TOP*
       LIST
       (ALOP WBC4 NMC3 PLEU PERI ARTH TRIG MALR PLAT MYOM SWOH)
       *DEFAULT-NOT-DEFINE*
       *OPTION-NOT-DEFINE*)
 (MAJOR U *TOP* LIST NIL *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (MINOR U *TOP* LIST NIL *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (SEROLO U *TOP* ATOM NIL *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (EXCLU U *TOP* ATOM NIL *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (NOTINV S *TOP* BOOL NIL *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (CONCL U *TOP* ATOM NIL *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (LOGIC U DISEASE LISP DISEASELOGIC *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (UNKLOGIC U DISEASE LISP UNKDISLOGIC *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*)
 (UNKDISP U DISEASE LISP UNKDISP *DEFAULT-NOT-DEFINE* *OPTION-NOT-DEFINE*))
```

図5　　知識ベースの内部表現（フレーム　MCTD）

ィタには知識ベースを操作する関数が全て用意されていなければならないと考えたからである。このようにすることによって、システムの構造化、モジュール化が達成され、またインプリメンテーション上の具体的な考慮点が浮び上るものと考えたからである。

4.2 関数の階層化

FMSで用いられる関数を組み立てる場合において、それぞれの関数について機能的に階層的分類を施した。ここで言う機能とは、次に示すようなFMSのシンプリメンテーションの立場からの分類である。

LEVEL1： CAR、CDR、CONS等のLISPシステム自身が持っている関数、原始関数。

LEVEL2： 原始関数を組み合わせて作成された関数であり、リストの和、積を求める関数のようなものである。

LEVEL3： 実際に知識ベースの各要素に対して、ある操作を実施するものである。このレベルの関数群はLEVEL4の関数群において利用されている。

LEVEL4： このレベルの関数群は2種類の関数群に分類することができる。

（i）エディタ用関数

（ii）システム関数

エディタ用関数は、主としてフレーム・エディタにおいて用いられている関数であり、FMSユーザが容易に知識ベースを構築することができるよう知識ベースとのインターフェイスとなるものである。システム関数はattached proceduresを容易に記述することのできるようにFMSが用意している関数群である。尚、attached proceduresは全てのレベルの関数を用いて記述されるものである。また知識ベースの各要素の操作に関しては各種のチェックが行なわれるので、そのチェックにかかるような操作が行なわれようとした場合にはエラー処理が働くような関数も含んでいる。勿論、FMSシステム記述に対してもシステム関数は用いられている。

以上の4レベルに分類したが、厳密に区別されるものではない。図4は、この概念図である。現在、各レベルの関数群の整備が行なわれつつある。

4.3 関数の機能的分類

関数は、その機能により次の5つに分類することができる。：1）フレームを操作する関数、2）スロットを操作する関数、3）他のフレームを呼び出す関数、4）種々のチェックを行なう関数、5）その他。ユーザはこれら5種類の関数群を用いて、attached proceduresを記述する。システム関数は上記の5種類のうちどれかに分類することができる。また、FMSシステム自身を記述している関数も上記5種類に分類することができる。表1はそれらの関数の代表的なものを示したものである。

Attached proceduresおよびフレーム・エディタの具体的動作として最もよく利用さ

表１　関数の機能的分類表

（ｉ）フレーム操作関数

CREATEFRAME＃	フレームの生成
DELETEFRAME＃	フレームの削除
GETFRAME＃	知識ベース中のフレームを返す
GETFRTYPE＃	Frame－Type を返す
GETSLOTNAMELIST＃	フレームに定義されているスロットの名前のリストを返す
INSTANCE＃	Frame－Type をInstance に付け替える
SUBCLASS＃	Frame－Type をSubclass に付け替える
SUB－HIERLIST	知識ベースの階層構造を求める

（ii）スロット操作関数

CREATESLOT＃	スロットの生成
DELETESLOT＃	スロットの削除
GETSLOT＃	フレーム中のスロットを返す
GETVAL＃	スロットの値部を返す
GETOP＃	スロットのオプション部を返す
SETVAL＃	値部の設定
SETOPTION＃	オプション部の設定
PRINTSLOT＃	スロットの印刷

（iii）他のフレームを呼び出す関数

MESSAGE＃	Attached procedure の起動

（iv）種々のチェックを行なう関数

CHK－TYPEVAL	データ型と値とのチェック
FRAMEP	フレームとして定義され得るかどうかを調べる
EXPRP	関数として登録されているかどうかを調べる

（v）その他

ATOMLIST	深さ１のリストにして返す
PPRINT	あるリストを清書してファイルに出力する
INTERSECTION	リストの積

れる機能は、フレームとスロットの参照およびスロットの値の参照・格納である。高度な例として知識ベースの階層構造を変更してしまうような、フレームの生成・削除を行なう場合もある。また、単なる一つのフレームだけでなく、サブフレーム・システムの変更およびスロットの属性（インヘリタンス・ロール、データ型）の変更・修正等が考えられる。これらの操作はユーザが、ＬＩＳＰシステムおよびＦＭＳにおける知識ベースの内部表現に精通しているならば、単なるＳ式の処理として行なうものである。しかしながらこの様な関数の作成・利用はユーザにとって負担となるばかりでなく、知識ベースを破壊してしまう恐れがある。そこで我々は、 １）～５）の関数を用意し、ＬＥＶＥＬ４システム関数群を用意した。また、 ３）の機能を持つ関数はattached proceduresの実行、フレームの評価を行なうために用意されたものである。 ４）の関数は知識ベースの管理・チェックのために用いられる。この関数群の中で最も重要なのは、あるスロットに値を記入しようとした時働くものである。 ５）その他の関数としては、4.2で述べたＬＥＶＥＬ２の関数が主となる。これは、人工知能研究には不可欠な高度なリスト処理を行なうために開発された。

１）～４）の関数は具体的には次のようなものである。１）フレームの生成・削除および出力等を行なう関数、２）スロットの生成・削除、値の記入・変更、および出力等を行なう関数、３）他のフレーム（具体的にはattached procedures）の起動を行ない、その値を返す関数、４）スロットの値とデータ型との対応、インヘリタンス・ロールとデータ型・値が順守されているか等のチェックを行なう関数。現在、１）～３）の関数の基本的部分はインプリメントされているが、４）の関数群については種々の難しい問題を含んでおり検討中である。

4.4 知識ベースの内部表現

知識ベースは、有限個のフレームから成り立っており、フレーム・システムを形成している。ＦＭＳにおいて知識ベースは、“ＫＮＯＷＬＥＤＧＥ－ＢＡＳＥ”という名前を持つ連続した領域に格納されている。“ＫＮＯＷＬＥＤＧＥ－ＢＡＳＥ”の各要素には、個々のフレームが与えられている。例えば、フレームＭＣＴＤ（後述）は、次のような内部表現を持っている（図５）。

“ＫＮＯＷＬＥＤＧＥ－ＢＡＳＥ”中には、フレーム名をkey としてハッシュ関数により格納位置が求められる。図５からわかるように、ＦＭＳではフレームは次のように表現されている。

（＜フレーム名＞ ＜Ｆrame－type＞
（スロット１）
（スロット２）
……
（スロットＮ））

```
FE: CHOOSE A COMMAND:cr
=CREATE
NOW, YOU CAN CREATE A NEW FRAME.
*CREATION PROCEDURE*
    1.DEFINE THE PARENT(A-KIND-OF) FRAME,
    2.DEFINE THE NEW FRAME-NAME, THEN
    3.DEFINE THE FRAME-TYPE OF THE FRAME JUST DEFINED.
YOU SHOULD TAKE CARE OF THE TYPE OF THE NEW FRAME.

*CURRENT FRAME HIERARCHY*
<<<< HIERARCHY >>>>
 ROOT
      LISPPROC(*)
      TABLE
      FINDING(*)
      IH-AGG(*)
      DISEASE

ASSIGN THE PARENT(A-KIND-OF)-FRAME-NAME:disease
DEFINE A NEW-FRAME-NAME:mctd
ASSIGN THE FRAME-TYPE:su

*CREATE-AFTER*
<<<< HIERARCHY >>>>
 ROOT
      LISPPROC(*)
      TABLE
      FINDING(*)
      IH-AGG(*)
      DISEASE
           MCTD

END OF CREATE SESSION.
FRAME < MCTD > HAS BEEN CREATED.
YOU SHOULD CREATE INFORMATION OF THE FRAME < MCTD > BY < EDIT > SESSION.
```

図6　　フレーム・エディタの実行例

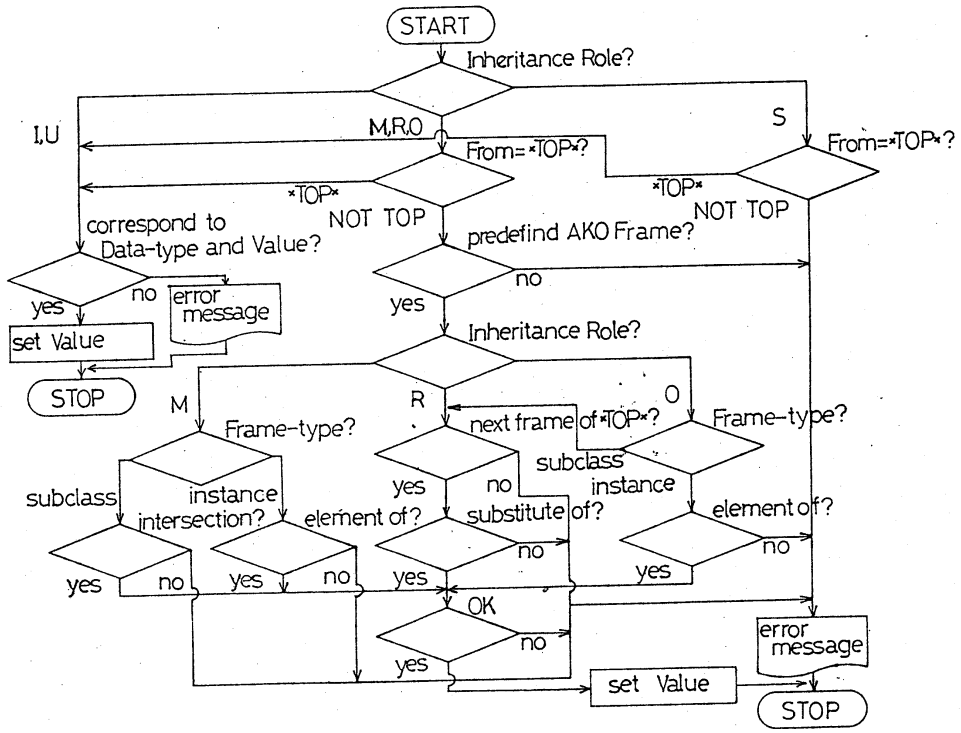

図7　　スロットの値を設定する操作に関するチエックの流がれ

このリストに対して関数ＣＤＲを２回適用することにより、一つのフレームはa －listとして表現される。つまり次のものである。

（（スロット１）（スロット２） …… （スロツトＮ））

このa －listは、スロット名をkey としている。以上のように、フレーム名およびスロット名より知識ベースに対してアクセス可能となる。

各スロットの一般的な構造は、次のものである。

（＜スロット名＞ インヘリタンス・ロール
ＦＲＯＭ部
データ型
値
デフォルト
オプション ）

ここでＦＲＯＭ部は、そのスロットがフレームをノードとする階層構造の中でどのフレームで最初に定義されたかを示している。

4.5 フレーム・エディタ

ＦＭＳのフレーム・エディタは狭義のフレーム・エディタ（Ｆrame editor：ＦＥ）とスロット・エディタ（Ｓlot editor：ＳＥ）から成り立っている。フレーム・エディタに知識獲得支援の機能を持たせているので、ユーザが楽に知識型システムを構築できるような設計がなされていなければならない。このことにより、フレーム・エディタには、統一されたインターフェイスおよびＧＵＩＤＥ機能等が必要である（ＦＭＳには、ＧＵＩＤＥ機能の一部がインプリメンテーションされている）。

フレーム・エディタ（ＦＥ）、スロット・エディタ（ＳＥ）は、階層化されている。つまり、フレーム・エディタ（ＦＥ）により知識ベースの枠組を定め、スロット・エディタ（ＳＥ）により具体的情報を設定していくという作業の流れに沿ったものである。この流れは、ユーザにとって自然なものであるべきことは言うまでもない。

フレーム・エディタ（ＦＥ）には、フレームの生成・削除等の機能が備わっており、知識ベース全体の管理が行なえるようになっている。また、知識ベースをファイルに格納したり、全体を見渡すことのできるようなコマンドも備わっている。スロット・エディタ（ＳＥ）は、個々のフレームに対して具体的に情報を決定したり変更を行なうことができる。

現段階において、フレーム・エディタは複雑な処理を除いて一応完成している。図６はフレーム・エディタを用いてフレームを生成するＣＲＥＡＴＥコマンドを実行したものである。

4.6 インヘリタンス・ロールの実現

あるフレームは、ある概念対象を表現し、知識ベースはフレームをノードとするような階層構造となる。この階層構造は概念の階層構造を表現しているので、親フレーム・子フレームとの間には上位概念・下位概念という関係が生じる。例えば、生物学における哺乳類と犬のような関係である。この場合には哺乳類が上位概念、犬が下位概念である。FMSは、このような2つの概念間の関係を表現することができる。各概念には個々の知識が記述されており、FMSにおいては個々の知識はスロットとして定義できる。当然ながら、親フレームの知識の中には、多少値が変更されても子フレームに受け継がれるべき性質の知識が存在することは我々の経験上明らかである。このように、親フレームから子フレームへ何らかの性質が受け継がれることは、インヘリタンスと呼ばれている。

フレーム・モデルにおいてインヘリタンスの問題は、特に重要である。なぜなら、あるフレームにおけるスロットの値が、どのようにしてなぜ他のフレームから決定されたかを示すからである。このインヘリタンスの仕方を表わしたものが、インヘリタンス・ロールである。FMSにおいては、インヘリタンス・ロールはスロットの属性となっている。インヘリタンス・ロールは、多くの汎用フレーム・システムで採用されており、その実用性・有効性が確認されている。この理由は、インヘリタンス・ロールの指定により、親フレームのスロットから子フレームのスロットの値に対して制約条件が生じるからである。これは、知識の定義、値の設定および更新や利用を矛盾のないものとするとともに、知識の定義の経済化にも役だっている。つまり、インヘリタンス・ロールの役割は、子フレームのスロットの値に対して何らかの制約条件を生じさせることである。

Brachman が指摘しStefik（7）が実現したように、Instance mode frame（最下位のフレーム）においてスロットの値は一意に決定できるが、Subclass node frame（途中の階層のフレーム）においてはその親フレームでのスロットの値を大域的な値として考えることにより、値はその範囲内の物でなければならないとしている。我々もインヘリタンス・ロールの実現に際して、この思想を採用した。

FMSを用いて、スロットに知識を設定しようとした場合には、常にトップ・ダウン的に値を設定しなければならないということである。これはあるスロットの値の設定に際しては、親フレームでのスロットの値がさだめられて始めて子フレームでの値が設定できるという、知識の定義順序を意味している。これは、ある概念対象を常に、その上位概念から明らかにしていくという立場に立ったものである。

FMSにおいては、6種のインヘリタンス・ロール（S、M、R、O、U、およびI）が準備されている。インヘリタンス・ロールの指定に際しては、十分にそのスロットの値の特性を考えなければならない、このことは以下に述べることにより明らかである。以下、各インヘリタンス・ロールの役割およびそのスロットに記入しようとしている値に対してどのようなチェックが行なわれるかを述べる。尚、親フレームから子フレームのスロットへ必ず受け継がれる属性は、スロット名・データ型およびインヘリタンス・ロールである。

1）S（Same ）：親フレームと同じ値を持つスロットであることを意味する。設定に際しては、常にＦＲＯＭ部が“＊ＴＯＰ＊”でなければならない。子孫フレームへの設定は自動的に行なわれる。値は、データ型との一致を見なければならない。

2）M（Member ）：“Subset　of”の関係にある値のために用いられるロールである。Subclass　node　frame には親フレームとの値との関係が常にSubset　ofの条件を満たさなければならない。当然データ型とのチェックが行なわれる。また、Ｉnstance　node　frame では親フレームでの値の一つの要素となっていなければならない。このロールとの組合わせが可能なのは、リスト形式を値に持たなければならないデータ型である。

3）R（Restriction）：“Substute　of”の関係にある値のために用いられるロールである。このロールに対しては、Subclass　node　frame おいては一意にその値を決定できる。値は、親フレームでの値の範囲内になければならない。

4）O（Option ）：Rと同じようなものである。しかしながら、Ｉnstance　node　frame においては一意にその値が決定される。もちろん親フレームでの値の範囲内になければならない。

5）U（Unique ）：値は、その親フレームでの値の制限が子フレームの値に対して何ら制限を与えるものではない。スロットの値は、知識ベースの階層構造の中でそれぞれのレベルについてユニークなものである。値には、データ型との一致が求められる。

6）Ｉ（Ｉndependent）：その概念対象の中だけに、存在すればよい知識のための指定であり、下位フレームへの影響はない。チェックは、データ型との一致だけである。

上記のように、インヘリタンス・ロールがどのようなものであろうと最低限のチェックとしてデータ型と値の対応関係は調べなければならない。但し、例外的に｀Ｍ´と｀Ｏ´については、データ型との不一致が生じる。なぜなら、Ｉnstance　node　frame においては、値が一意に決定することができるからである。しかしその値に対しては、それぞれのデータ型がとるべき値の各要素についている条件は守っていなければならない（例えば、データ型がＦＬＩＳＴである場合には、値はフレーム名でなければならない）。図７は、スロットに値を設定しようとした場合の基本的なチェックの流れを表している。

スロットに対するインヘリタンス・ロールの選択は、｀Ｍ´、｀Ｒ´および｀Ｏ´を指定した場合には、ＦＭＳは親フレーム・子フレームの値のチェックまでをも行ない、値に矛盾がある場合には値が設定されない。当然の事ながら、常に親フレームでの値の設定は事前に行なわれていなければならない。

これら４つのインヘリタンス・ロールは、｀Ｓ´をのぞいてとり得る値が常に親フレームの値よりも小さな範囲を示したり、制限をより厳しくするようなものである。我々がこのようにしたのは、上位概念が下位概念を包んでいるとしうる定説によるものである。ユーザは、常にこのことを意識して知識モデルを作成しなければならない。よって親フレー

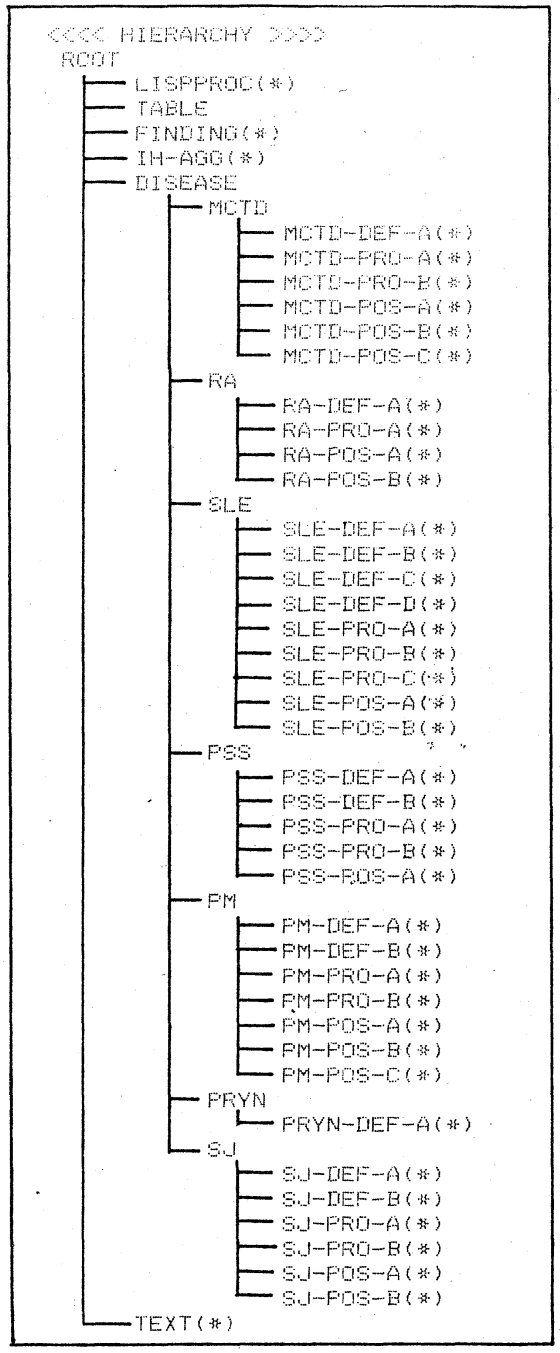

図8　　知識ベースの階層構造の例

```
 FRAME-NAME: IH-AGG                    FRAME-TYPE: INSTANCE
A-KIND-OF        (U)  ROOT              FRAME    ROOT
DDESCENDENT      (U)  ROOT              FLIST    NIL
DESINF           (U)  ROOT              TEXT     "Intermediate hypotheses and aggregat
ions."
CREATE           (U)  ROOT              STRING   ("G00126" "8212272114")
MODIFY           (U)  ROOT              STRING   ("G00126" "8301052031")
IF-NEEDED        (U)  ROOT              LIST     NIL
IF-ADDED         (U)  ROOT              LIST     NIL
IF-REMOVED       (U)  ROOT              LIST     NIL
AG1              (I)  *TOP*             LIST     (1 OF PMWM MBXM)
AG2              (I)  *TOP*             LIST     (2 OF AMEM AG1)
AG3              (I)  *TOP*             LIST     (2 OF AMEM EMG)
MYOM             (I)  *TOP*             LIST     (1 OF AG2 AG3)
AG4              (I)  *TOP*             LIST     (2 OF AMES PMWS)
AG5              (I)  *TOP*             LIST     (2 OF MYOM MBXS)
MYOS             (I)  *TOP*             LIST     (1 OF AG4 AG5)
AG6              (I)  *TOP*             LIST     (2 OF FEM ANE1)
AG7              (I)  *TOP*             LIST     (2 OF MALE ANE2)
NMC3             (I)  *TOP*             LIST     (1 OF AG6 AG7)
RYES             (I)  *TOP*             LIST     (1 OF RAYN ESOP)
RNPE             (I)  *TOP*             LIST     (2 OF RNP ENA4)
RNPN             (I)  *TOP*             LIST     (2 OF RNP ENA1)
RNPA             (I)  *TOP*             LIST     (2 OF RNP ENA0)
NOSM             (I)  *TOP*             LIST     (1 OF *SM)
RAFL             (I)  *TOP*             LIST     (1 OF *RAFH)
RAEX             (I)  *TOP*             LIST     (11 OF *SCLD *MALR *PHOT *HELI *GOTP
*DSCL *ERYN *MYOS *RYES *GOUT *RNPN)
NENA             (I)  *TOP*             LIST     (1 OF *RNPN)
NEPR             (1)  *TOP*             LIST     (1 OF FGLO MESG DGLO MEMG RBCC)
CNS              (I)  *TOP*             LIST     (1 OF OBSY COMA SEIZ PSYC)
SERO             (I)  *TOP*             LIST     (1 OF PLEU PERI)
HCMP             (I)  *TOP*             LIST     (1 OF HCM3 HCM4)
SLSE             (I)  *TOP*             LIST     (1 OF ANAP LE SM DNAP)
AG8              (I)  *TOP*             LIST     (3 OF RNP ENAP NOSM)
SLEX             (I)  *TOP*             LIST     (3 OF *SCLD *EART *AG8)
PLCO             (I)  *TOP*             LIST     (1 OF PLID DCO7)
SWHO             (I)  *TOP*             LIST     (1 OF SWOH SWDO)
RYED             (I)  *TOP*             LIST     (1 OF RAYN ESOP DGUL)
ANRE             (I)  *TOP*             LIST     (2 OF RNP ENAP)
PSEX             (I)  *TOP*             LIST     (2 OF *EOSI *ANRE)
RRES             (I)  *TOP*             LIST     (1 OF RAYN ESOP)
PMEX             (I)  *TOP*             LIST     (4 OF *SCLE *HEMA *RNPN *DNA6)
PMXX             (I)  *TOP*             LIST     (3 OF *HEMA *RNPN *DNA6)
RYEX             (I)  *TOP*             LIST     (5 OF *MCTD *PSS *SLE *RA *PM)
SJEX             (I)  *TOP*             LIST     (4 OF *MCTD *PSS *SLE *PM)
LOGIC            (I)  *TOP*             LISP     IHAGGLOGIC
UNKLOGIC         (I)  *TOP*             LISP     UNKIHAGG
```

図9　　IH-AGG フレーム

```
    FUNCTION-NAME: IHAGGLOGIC
    (LAMBDA (SNAME)
     (PROG (ANS)
        (SETQ ANS (GETOP# SNAME 'IH-AGG))
        (COND ((NOT (EQUAL ANS '*OPTION-NOT-DEFINE*)) (RETURN (CAR ANS))))
        (SETQ ANS (GETVAL# SNAME 'IH-AGG))
        (SETQ ANS (INVEST (CAR ANS) (CDDR ANS) 'IH-AGG))
        (SETOPTION# SNAME 'IH-AGG ANS)
        (RETURN (CAR ANS))))
```

図10　　Attached procedure の例

```
FRAME-NAME:DISEASE                FRAME-TYPE:SUBCLASS
A-KIND-OF      (U)  ROOT            FRAME    ROOT
DDESCENDENT    (U)  ROOT            FLIST    (MCTD RA SLE PSS PM PRYN SJ)
DESINF         (U)  ROOT            TEXT
                                               *VALUE-AND-DEFAULT-NOT-DEFINE*
CREATE         (U)  ROOT            STRING   ("G00126" "8212272114")
MODIFY         (U)  ROOT            STRING   ("G00126" "8212301715")
IF-NEEDED      (U)  ROOT            LIST     NIL
IF-ADDED       (U)  ROOT            LIST     NIL
IF-REMOVED     (U)  ROOT            LIST     NIL
CONCLUSION     (U)  *TOP*           LIST
                                               *VALUE-AND-DEFAULT-NOT-DEFINE*
CONSIST        (I)  *TOP*           LIST
                                               *VALUE-AND-DEFAULT-NOT-DEFINE*
UNKNOWN        (I)  *TOP*           LIST
                                               *VALUE-AND-DEFAULT-NOT-DEFINE*
DISEASES       (I)  *TOP*           LIST     (MCTD RA SLE PSS PM PRYN SJ)
LOGIC          (U)  *TOP*           LISP     MAINLOGIC
UNKLOGIC       (U)  *TOP*           LISP     UNKMAINLOGIC
UNKDISP        (U)  *TOP*           LISP     UNKDISPMAIN
```

図11　　階層フレームの例
(DISEASE フレーム)

```
FRAME-NAME:MCTD                   FRAME-TYPE:SUBCLASS
A-KIND-OF      (U)  ROOT            FRAME    DISEASE
DDESCENDENT    (U)  ROOT            FLIST    (MCTD-DEF-A MCTD-PRO-A MCTD-PRO-B MCT
D-POS-A MCTD-POS-B MCTD-POS-C)
DESINF         (U)  ROOT            TEXT     "Mixed connective tissue disease."
CREATE         (U)  ROOT            STRING   ("G00126" "8212272114")
MODIFY         (U)  ROOT            STRING   ("G00126" "8212301716")
IF-NEEDED      (U)  ROOT            LIST     NIL
IF-ADDED       (U)  ROOT            LIST     NIL
IF-REMOVED     (U)  ROOT            LIST     NIL
CONCLUSION     (U)  DISEASE         LIST
                                               *VALUE-AND-DEFAULT-NOT-DEFINE*
MACR           (S)  *TOP*           LIST     (RYES MYOS DC07 SW00 SCLD ENAP)
M1CR           (S)  *TOP*           LIST     (ALOP WBC4 NMC3 PLEU PERI ARTH TRIG M
ALR PLAT MYOM SWOH)
MAJOR          (U)  *TOP*           LIST     NIL
MINOR          (U)  *TOP*           LIST     NIL
SEROLO         (U)  *TOP*           ATOM     NIL
EXCLU          (U)  *TOP*           ATOM     NIL
NOTINV         (S)  *TOP*           BOOL     NIL
CONCL          (U)  *TOP*           ATOM     NIL
LOGIC          (U)  DISEASE         LISP     DISEASELOGIC
UNKLOGIC       (U)  DISEASE         LISP     UNKDISLOGIC
UNKDISP        (U)  DISEASE         LISP     UNKDISP
```

(MCTD フレーム)

```
FRAME-NAME:MCTD-DEF-A             FRAME-TYPE:INSTANCE
A-KIND-OF      (U)  ROOT            FRAME    MCTD
DDESCENDENT    (U)  ROOT            FLIST    NIL
DESINF         (U)  ROOT            TEXT     "Mixed connective tissue disease.  De
finite-A."
CREATE         (U)  ROOT            STRING   ("G00126" "8212272114")
MODIFY         (U)  ROOT            STRING   ("G00126" "8212301717")
IF-NEEDED      (U)  ROOT            LIST     NIL
IF-ADDED       (U)  ROOT            LIST     NIL
IF-REMOVED     (U)  ROOT            LIST     NIL
CONCLUSION     (U)  DISEASE         LIST
                                               *VALUE-AND-DEFAULT-NOT-DEFINE*
MACR           (S)  MCTD            LIST     (RYES MYOS DC07 SW00 SCLD ENAP)
M1CR           (S)  MCTD            LIST     (ALOP WBC4 NMC3 PLEU PERI ARTH TRIG M
ALR PLAT MYOM SWOH)
MAJOR          (U)  MCTD            LIST     (4 0 NIL)
MINOR          (U)  MCTD            LIST     (0 0 NIL)
SEROLO         (U)  MCTD            ATOM     RNPB
EXCLU          (U)  MCTD            ATOM     NOSM
NOTINV         (S)  MCTD            BOOL     NIL
CONCL          (U)  MCTD            ATOM     DEFINITE
LOGIC          (U)  DISEASE         LISP     CASELOGIC
UNKLOGIC       (U)  DISEASE         LISP     UNKCASELOGIC
UNKDISP        (U)  DISEASE         LISP
                                               *VALUE-AND-DEFAULT-NOT-DEFINE*
```

(MCTD-DEF-A フレーム)

ムの値よりも子フレームの値の範囲を拡大したり、より制限を緩めたりするようなインヘリタンス・ロールは必要でないと考える。

FMSにおいては次のようになっている。知識ベースとしてのフレーム・システムは、フレーム・エディタを用いて、対話的に定義し構築される。このとき、定義しようとするフレームの親フレームは事前に定義されていなければならない。但し、最上位のフレームROOTは、システムに備えられている。従って、全てのフレームはROOTの子孫となる。ここで、あるフレームで新たにスロットが定義されると、それはその子孫となるフレームを意義するとき、上記に述べたインヘリタンス・ロールの機能に従って、自動的に定義される。

5. 例題：COMEXモデルの表現

ここではFMSの応用例として、我々が先に開発した汎用エキスパート・システムCOMEXを書き、更にこれを用いてリューマチ診断支援モデルRHEUMを表現した例について極く簡単に述べる。COMEXおよびRHEUMモデルについては、文献(4,16,17,24)を参照されたい。これはプロダクション・システムと決定クライテリア・フレームから構成された、一種の簡易フレーム・システムである。FMSを用いて表現するにあたっては、なるべく元のモデルが累進し易くなるように心掛けた。なお、以後の図はフレーム・エディタの表示コマンドを用いて出力したものである。

図8はフレーム・システムの階層構造を示す。LISPPROCはattached proceduresを、FINDINGは所見質問リストを、IH-AGGはすべての中間仮説の定義を各スロットに持つフレームである。7つの病気に対する診断クライテリアは、DISEASEフレームMCTD-DEF-AはMCTDのdefinite-criteria-case-Aを意味する。図9はIH-AGGフレーム(中間仮説を定義したAND/ORトリー)を、図10はここで定義されているAND/ORトリー評価のattached procedure IH-AGGLOGICのLISP関数を示す。図11は階層フレームDISEASE、MCTDおよびMCTD-DEF-Aを示す。なお、各スロット表示の第3例は、このスロットが定義された先祖フレームを示す。これはスロットのFROM部に格納されている。また、このモデルの階層構造(図8)中に示されているフレーム名の後に`(*)´を持つフレームは、Instance node frame である。

このモデルはCOMEX(FORTRANバージョン)と同様な応答をするが、少し遅い。

6. まとめ

FMSは、汎用のフレーム型知識表現言語であり、フレーム型エキスパート・システムの研究開発に使用されるものである。FMSの長期的目標は、複雑で高度な問題解決のための専門知識を表現するツールの開発、およびこれを通じて意思決定過程のメカニズムを

究明することにある。FMSの評価を行なうためには、FMSを用いてエキスパート・システムの研究・開発を試ることが必要である。さもなければ、toy researchに陥ってしまうことになるであろう。現在は一応第一段階が達成されたものと考えられる。

現在の状況をふまえて、FMSの問題点を以下にあげる。

1) フレーム・モデルが持っている短所を克服していない。

フレーム・モデルは、すでに述べたように柔軟な構造を持つので研究開発用ツールとしては強力である。裏を返せば、汎用性は極めて高いが利用者の負担が大きすぎることである。これの解決方法としては、FMSの上に問題向きのエキスパート・モデルをのせ、これを用いて具体的なエキスパート・システムを構築するのが一つの対策である。また、フレーム・モデルの持つ柔軟性が知識の矛盾や一貫性のチェックを困難にしている（後述）。

2) 知識のチェックが不十分である。

知識型システムが対象としている領域は、曖昧で不完全である開いた世界から切り出された一部の知識を取り扱うものである。これは知識を論理的にチェックすることは困難であり意味のない場合もあり得ることを示している。しかしながら、可能な限りシステムがチェックすることは重要な機能である。知識のチェックは、構文と意味からチェックされなければならない。構文チェックは、インヘリタンス・ロールに基づくものが最も重要であり、FMSはそれを行なっている。意味論的チェックには2つの側が考えられる。一つはモデルの構文則であり、もう一つはモデルが持っている知識についてのものである。従って一部を除いては、論文的にチェックが困難なものである。知識のチェックは最終的にエキスパートの責任となる。ユーザが、知識ベースの構造に対して見通しが適切であるならばかなりの部分の曖昧さが論理的にチェックすることが可能となる。

3) FMSには欠落している機能がある。

FMSの機能的な中心部は、フレーム・エディタである。現段階において、知識ベースを操作するための機能は備わっている。しかしながら、複雑な処理を行なおうとすると、やや問題が生じる。例えば、スロットの属性（データ型、インヘリタンス・ロール）の付け替えを行なうような機能は備わっていない。また、知識の統合化がではないのも問題の一つである。知識の統合化とは、独立に開発された知識ベースを融合して一つの知識ベースを作ることである。この機能が備わることにより知識の共有を行なうことができる。

1) ～ 3) の問題は、FMSに対する根本問題を多く含んでいる。それぞれ、(1) いかにすれば、ユーザに負担の少ない知識表現言語となるか、(2) 意味的チェックの方法および知識のチェックはどこまで可能であるか、(3) フレーム操作システムにはどのような機能が備わっていればよいのであろうか、である。今後は、これらの問題に答えるようにケース・スタディを通じてFMSの充実を計らなければならない。このためには、実際の応用を通じて、エキスパート・システムを構築することが望まれる。

最後に、UTILISPはその実行速度の速さとともに、lisp objectに対する豊富な

data－type、各種のシステムの持つ大域変数がユーザに解放されていることにより、非常に操作しやすいＬＩＳＰシステムである。しかしながら、プログラミング上の変数の束縛が不明確であった場合、その実行がＬＩＳＰシステムが都合のつくように解釈がなされてしまう場合がある。また、一般的にＬＩＳＰシステムで問題となっている内装エディタに若干使用し難い点があったり、ファイル操作およびチェックポイント機能が弱い等の問題がある。

謝辞

システムのインプリメンテーションを行なってくれた小沢健司、高島伸彦の両君、並びにＵＴＩＬＩＳＰの使用を許していただいた東京大学　和田研究室に感謝する。

参考文献

1) E. Feigenbaum ; The Art of Artificial Intelligence －The termes and Case Studies of Knowledge Engineering, Proc . 5th ＩＪＣＡＩ (1977)
2) M. Minsky ; A Framework for Representing Knowledge, in P. Winston (ed.) The Psichology of Computer Vision , Mc Graw－Hill (1975)
3) 上野晴樹；フレーム理論に基づく知識型システム、ＩＰＳＪ人工知能と対話技法 21-3 (1981)
4) H. Ueno , D. Lindberg et al. ; Design of a Criteria －Based R hematology Consulting System for a Microcomputer, Proc . MEDＩＦＯ 80 (1980)
5) P. Winston ; Representing Knowledge in Frames , in Artificial Int elligence , Adisson－Wesly (1977)
6) Ｉ. Goldman ; NUDGE : A Frame－Based Scheduling System , Proc . 5th ＩＪＣＡＩ (1977)
7) M. Stefik ; An . Examination of a Frame Structured Rep re sentation System , 6th ＩＪＣＡＩ (1979)
8) D. G. Bobrow and T. Winograd ; An Overview of KRL, a Knowledge Representation Language , Cognitive Science, Vol. 1、No . 1 (1977)
9) R. G. Smith and P. Friedland ; ＵＮＩＴ Package User ´s Guide, Stanford Heuristic Programming Project Memo HPP-80-28 (1980)

10) D. Smith and E. Clayton；A Frame−Based Production System Architecture , Proc . 1st AAAI (1980)
11) J. Aikins ；Prototype and Production Rule ：An Framework to Knowledge Representation for Hypothesis Formation, Proc . 6th IJCAI (1979)
12) 井上昌計；論理学、成文堂
13) 園田義直；知識と論理、富士書房
14) 坂本百大；坂井秀寿、現代論理学、東海大出版会
15) 田中幸吉他；知識工学とその応用、情報処理、Vol. 23、No . 11、1982
16) 上野晴樹；COMEX：汎用知識型エキスパート・システム開発言語、情報処理学会知識工学と人工知能研究会資料、1982
17) 上野晴樹、COMEXマニュアル、東京電機大学経営工学科、1982
18) 上野晴樹、データベース技術と知識ベース・システム・シンポジューム資料、1982
19) Chikayama, T；UTILISP MANUAL, 1981、東京大学
20) Melle, W. ；A Domain −Independent Production −Rule System for Consultation Programs , Proc . 6th IJCAI (1979)
21) Nii, H. P and Aiello , N；AGE (Attempt to Generalize) ：A Knowledge−Based Program for Buildings Knowledge−Based Programs , Proc . 6th IJCAI (1978)
22) 上野晴樹、伊藤秀昭； FMS： フレーム理論に基く汎用知識表現言語、AL−82−64、電子通信学会オートマトンと言語研究会資料、1982
23) 中島秀之；Prolog とその処理系、情報処理、Vol. 23、No . 11, 1982
24) D. Lindberg , G. Sharp, H. Ueno . et al. ；Computer −Based Reheumatology Consultant 、Proc . MEDIFO 80 (1980)
25) G. Beretta, et als . ；XS−1：An Integrated Interactive System and Its Kernel 、Prco . 6th ICSE (1982)
26) S. J. Greenspan, J. Mylopoulos ；Capturing More World Knowledge in the Requirement Specification、Proc . 6th ICSE (1982)
27) Mark S. Fox；On Inheritance in Knowledge Representation 、Proc . 6th IJCAI (1979)
28) P. H. Winston, B. K. P. Horn ；LISP、Addison−Wesley (1981)