日本機械学会論文集(C編)
59巻568号(1993-12)

論文 No. 93-0429

対象モデルを適用した知識ベースのデータ構造*

長坂保美*1, 大滝英征*2
石川義雄*2, 綿貫啓一*2

Data Structure of Knowledge Base Based on Object Model

Yasumi NAGASAKA, Hideyuki OTAKI,
Yoshio ISHIKAWA and Keiichi WATANUKI

With application of an expert system for practical purposes, knowledge representation has become increasingly complicated, and current Artificial Intelligence (AI) tools cannot represent the knowledge of experts very well. Therefore, systems built by these AI tools are not adaptable in cases of actual application. Hence, we have investigated optimal development of an appropriate knowledge representation in consideration of actual applications. As a result, by modeling "function" and "structure" on traditional AI tools based on the concept of the object model, we have developed a system which creates a source code (language) automatically. This paper describes knowledge representation for building of AI tools and the method for creating a source code automatically.

Key Words: Modelling, Expert System, Artificial Intelligence, AI Tool, Object Model, Attribution Function, Daemon Function

1. 緒　　言

エキスパートシステム(ES)の知識表現や推論機構が自由に定義でき，しかも知識表現の異なる知識ベース間でも自在に推論可能となるAIツールができれば，機械工学分野でのES開発に弾みがつく．

それを可能にするためには，種々の専門分野の知識表現に随時対応できるようにAIツールそのものを，対象モデル(問題解決のための機構を備えているモデルで，機能と構造の組合せで表現)[5]における「機能」と「構造」でモデル化しておく．しかも，その「機能」と「構造」でモデル化された表現(AIツールのモデル表現)を自由に変更でき，AIツールのソースコードを自動的に生成できる機能を備えてやればよい．加えて，この新たなAIツールで構築された知識ベースと，既存の知識ベースと結合できる管理システムが構築されれば万全となる．後者については，ルール表現や従来の手続き表現(C言語など)を付加手続きとして呼び出せるようAIツール構築者(以後，ユーザと呼ぶ)側で定義，管理できるようにしてやればよい．これについては前報[6]にて述べた．前者については，知識ベースのデータ構造を「機能」と「構造」でモデル化し，ユーザ側で容易に定義でき，システム側で管理できるようにしてやれば達成できる．

これを実現するには，従来の知識表現を包含でき，知識表現の記述性や推論機構の柔軟性に優れているフレーム表現[1]を用いるのが有利である．しかし，フレーム表現による従来のAIツールKEE[2]では，知識ベースのデータ構造が固定的なフォーマットとなっているため，上記の要求には対処し難い．一方，STROBE[3]やKRINE[4]は，知識ベースのデータ構造を自由に定義できるが，これを管理するための機能を有していないため，データ構造の定義を逐次行う必要がある．さらに，ユーザは，ツールの内容についても熟知していなくてはならず，汎用性が高いとはいい難い．

そこで，これら従来のAIツールをふまえた上で，前述の機能を新たに付加するため，KEEの知識ベースのデータ構造「ユニット-スロット-ファセット-バリュー」に準拠し，ファセットに代わり「アトリビューション」という考え方を導入した．この際，スロットはシステムが管理するシステムスロットと，使用者が自由に定義できるユーザスロットとで構成するようにす

* 原稿受付　平成5年3月26日．
*1 正員，(株)ゼクセル(〒355 東松山市箭弓町3-13-26)．
*2 正員，埼玉大学工学部(〒338 浦和市下大久保255)．

る．そして，システムスロットは対象の知識表現を管理して，継承関係やユーザが定義した専門分野固有の知識表現を管理するようにする．アトリビューションは，ユーザスロットに関する付属情報を格納する領域とし，そこに特定の「機能」をもたせる．これにより，ユーザスロットそのものを定義，管理できる．

そこで，本稿では上記の根幹となるAIツールのモデル表現，およびこのモデル表現をもとにソースコードを生成する方法について述べる．

2. AIツールのモデル表現

2・1 AIツールのモデル化　図1は，AIツールそのものを，対象モデル[5]における「機能」と「構造」とで示したものである．これは，フレーム表現を基本とするKEE[2]やKRINE[4]などを参考に，ES開発に必要な三要素(推論機構，知識獲得，知識表現)から整理したものである．図中の「機能("FUNCTION")」はAIツールの要求される機能を，また「構造("STRUCTURE")」はこの機能を実現するための領域を示している．

ここで，推論機構の「機能」は，継承機能，デーモン機能，付加手続き機能などが含まれる．継承機能(図中の下線部分)は，フレーム理論における知識間の関係(抽象-具体関係，全体-部分関係など)を示す機能である．そしてこれに対応する「構造」は，継承機能の関係を管理する領域として示される．

同様に，知識獲得の「機能」は，システム環境(既存システムとの結合)，ユーザ環境(パスワードの登録など)，インタフェース環境[8]などが含まれる．

知識表現の「機能」は2・4節で述べるユニット内表現や，スロット内表現などが含まれる．

このように，AIツールは，要求される「機能」と，これを実現する「構造」から表現するようにした．

2・2 知識ベースのデータ構造　図2は，2・1節の「機能」と「構造」を実現するための知識ベースの基本的なデータ構造を示したものである．ここで，ユニット内表現は，システムスロットとユーザスロットから構成した．

システムスロットは，スロット名，スロットタイプ，スロット値から構成される．これに対し，ユーザスロットは，スロット名，スロットタイプ，スロット値，そして複数のアトリビューションから構成されるようにした．アトリビューションは，2・4節で述べるモデル表現を用いて，その「機能」と「構造」を自由に定義できる．この「機能」を，アトリビューション機能と呼ぶこととする．

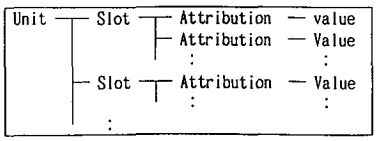
2・3 アトリビューション機能　図3は，アトリビューション機能の例として，アトリビューション「From」の動作例を示したものである．

アトリビューション「From」の「構造」は，ユニット名(図例の「UnitName」)が格納される領域である．一方，この「機能」は，継承機能の一種で，抽象-具体関係(ISA関係)にあるスロット値の探索時間を短縮する機能で，スロット生成時とスロット参照時に動作する．ここで，この機能の動作について，図中の番号順に追ってみる．

(1)　スロット生成(Creation)時では，ユーザによってスロット名，スロットタイプ，スロット値の入力が行われる．この際，スロット内に組込まれたアトリビューション「From」の「機能」が動作し，(2)が行わ

Subjects of Expert System	"FUNCTION" of AI tool	"STRUCTURE" of AI tool
Reasoning Mechanism	·inheritance-role ·daemon ·attached procedure	·area(slot) of ISA relation, HAS-PARTS relation, etc. ·area(attribution) of IF-NEEDED, IF-ADDED, IF-REMOVED, etc. ·area(unit) of user's functions .. .
Knowledge Aquisition	·system environment ·user environment ·interface environment	·area(unit) for managing connection between systems ·area for managing usernames, passwords, user group, etc. ·area for managing windows in system, commands in window, etc. .....
Knowledge Representation	·hierarchy representation ·unit(frame) representation ·slot representation ·data representation	·area for managing hierarchy structure in frame theory ·area for managing unit structure (system-slot and user-slot) ·area for managing slot structure ·area for managing data type

図1　AIツールのモデル化



(a)　知識ベースのデータ構造

```
system-slot
(SlotName_1   SlotType_1   SlotValue_1)
(SlotName_2   SlotType_2   SlotValue_2)
  :
-----------------------------------------
user-slot
(SlotName_U1  SlotType_U1  SlotValue_U1
              (Attribution1  Att_Value1)
              (Attribution2  Att_Value2)
               ...)
  :
```

(b)　ユニット内表現のスロット構成

図2　知識ベースのデータ構造

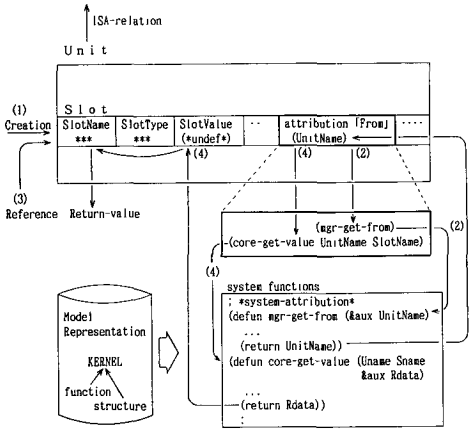


図 3 アトリビューション「From」の動作例

れる．

（2） あらかじめ定義された関数「mgr-get-from」が起動する．この関数は，従来の ISA 関係による探索方法(上位の概念を一つずつ探索する方法)を用いて，スロット値が実際に存在するユニット名「UnitName」を探索し，これをアトリビューション「From」に格納する．

（3） スロット参照(Reference)時では，ユニット名とスロット名の指定が行われ，該当するスロット値の評価値が返される．ここで，もしスロット値が未定義の場合は，従来の ISA 関係による探索が行われず，アトリビューション「From」の「機能」が動作し，（4）が行われる．

（4） 「From」内のユニット名と（3）で指定されたスロット名を引数として，関数「core-get-value」が起動する．この関数は，2・4 節で述べている CORE 関数の一つで，指定されたユニット内のスロット値を評価値として返すものである．

このように，アトリビューション「From」は，スロット値が未定義の場合，ただ一度の探索で目的とするスロット値を評価できるので，探索時間を短縮することができることがわかる．

さて，ここで提案したアトリビューション機能は，アトリビューション「From」のように，スロット内の「構造」に対応する「機能」という形式で定義できる．この定義については，2・4 節で述べる．そして，このアトリビューション機能によって，上記のような探索(推論)時間を短縮する機能，ある特定のスロット名に対してのみ起動する機能(スロット管理機能)，単なるデフォルト値を格納する領域に対して起動する機能(デフォルト機能)というように，知識ベースの構築性を

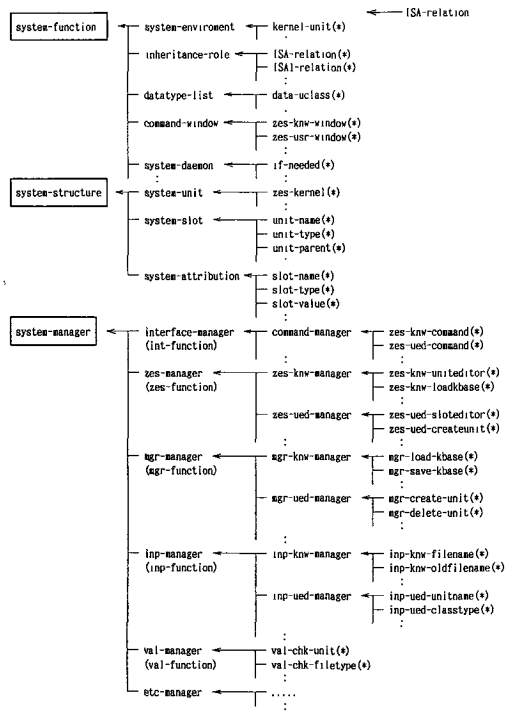


図 4 AI ツールにおけるモデル表現の階層の例

高めるための機能を随意付加できる．また，従来的なデーモン機能の拡張[7]としても機能し，スロット値に数学的な記述を可能にして，記述性を高めることもできる．このように，ここで提案したアトリビューション機能は，利便性の高いツールを可能にするものである．

2・4 AI ツールのモデル表現 図 4 は，実際に構築した AI ツールにおけるモデル表現の階層の例を示したものである．

AI ツールのモデル表現は，AI ツールの「機能」を表すシステム機能ユニット群(system-function)，「構造」を表すシステム構造ユニット群(system-structure)，およびこれらを実現するためのシステム管理ユニット群(system-manager)から構成されている．図中の実線は，フレーム表現の ISA 関係を示している．以下に，各ユニット群について述べる．

2・4・1 システム機能ユニット群 システム機能ユニット群は，専門分野固有の知識表現(ユニット内における名称や，この名称の機能，および名称間の関係をいう)による「機能」を管理する．ユーザは，このインスタンスユニットに，図 1 に示した各種「機能(“FUNCTION”)」を実現する．なお，基本的な「機能」については，あらかじめモデル表現内に登録されてい

るので，ユーザは必要な「機能」を選択し，必要に応じて修正を加えるだけでよいようにした．

例えば，システム環境(system-enviroment)では，使用する知識表現言語(Prolog 言語，C 言語など)との結合関係が記述される．このほか，継承機能(inheritance-role)，デーモン機能(system-daemon)，入出力機能(command-window)などが該当する．これら機能の実現例として，3章で継承機能を取り上げ述べている．

2・4・2　システム構造ユニット群　　システム構造ユニット群は，専門分野固有の知識表現による「構造」を管理する．ユーザは，このインスタンスユニットに，システム機能ユニット群で定義した「機能」に対する「構造」(図 1 の「構造("STRUCTURE")」)を定義する．

例えば，システム環境を管理するシステムユニット(system-unit)，ユニット間の関係を管理するシステムスロット(system-slot)，2・3節で述べたスロット内を管理するシステムアトリビューション(system-attribution)が該当する．ここで，知識ベースのデータ構造を示すユニット内表現，およびスロット内表現がどのように定義されるかを述べる．

(1)　ユニット内表現：図5はモデル表現(system-slot)とユニット内表現との関係を示したものである．モデル表現のインスタンスユニットのスロット「SlotName」の値は，ユニット内表現のスロット名を示している．同様に，スロット「SlotType」の値はスロットタイプを，スロット「SlotValue」の値はスロットバリュー(スロット値)を示している．

例えば，図5は図面管理に関する知識ベースを構築する場合のユニット内表現の例である．ここでは，ユニット名を「図面名(DrawingName)」[図中(1)]としている．新たにモデル表現に追加したユニット「全体-部分(Has-Parts)」と「部分-全体(Part-of)」では，上位の概念を利用してアセンブリと部品の関係が自動的に組込まれる[図中(2)]．また，ユニット「作成日付(Created-By)」では，ユニットが生成された日付が自動的に組込まれる [図中(3)]．

このように，専門分野固有の知識表現(図例では，DrawingName, DrawingType, …)や，ユニットの情報を自動的に組込む機能[図中(2)と(3)]など，モデル表現(system-slot)によって，ユニット内表現を容易に定義できるようにした．

(2)　スロット内表現：図6は，モデル表現(system-attribution)とスロット内表現との関係を，図3で示したアトリビューション「From」を例に示したものである．

スロット内表現は，図5と同様にスロット名「AttributeName」，スロットタイプ「AttributeType」，そしてスロットバリュー「AttributeValue」によって，2・3節で述べたような「機能」と「構造」を自由に定義できるようにした．

2・4・3　システム管理ユニット群　　システム管理ユニット群は，専門分野固有の知識表現による「機能」と「構造」を実現する AI ツール内のプログラム部品群(関数群)を管理する．

図7は，3章で述べるシステム管理ユニット群とその依存関係(has***-relation)を示したものである．

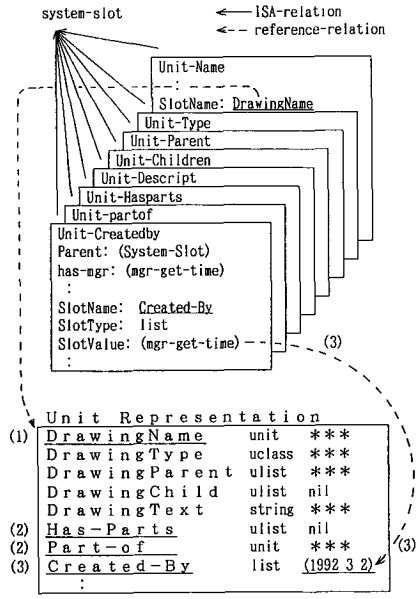


図 5　モデル表現とユニット内表現の関係

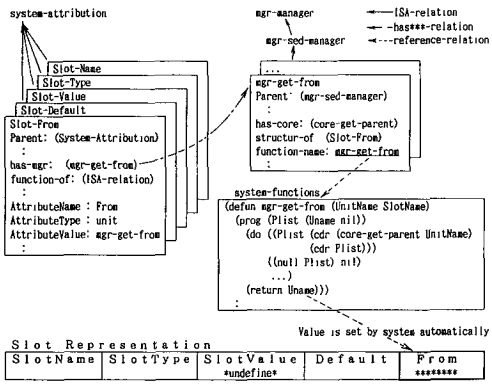


図 6　モデル表現とスロット内表現の関係

これは，AIツールのモデル表現で，システム機能ユニット群とシステム構造ユニット群を定義した際，ユニット内に登録されている依存関係を参照して，これらの関数群を必要に応じて定義することを示している．

なお，図中のCORE関数(CORE-function)は，図4のモデル表現によって新たなAIツールを生成するための基本関数である．例えば，インタフェース(Lisp言語とX-windowとの結合)関数，知識ベースの書込みや読込み関数，ユニットの生成や削除関数，さらに推論に関する基本関数など117の基本関数から構成されている．代表的な関数として，図8の「core-get-slot」はユニット内の該当するスロット本体を返す関数で，「core-get-parents*」は上位ユニットリストを返す関数である．

（a） mgr関数(mgr-function)：図4のユニット「mgr-manager」の下位ユニット群に相当する．知識ベースの構築や推論の基本的機能をもつCORE関数［図中(1)］を用いて，専門分野固有の知識表現を実現する．AIツールの「機能」と「構造」を直接実現する関数で，「zes関数」によって管理［図中(2)］されている．

（b） zes関数(zes-function)：図4のユニット「zes-manager」の下位ユニット群に相当する．mgr関数が起動するための入力条件を定義し，評価を行う．入力条件はinp関数を用いて定義［図中(3)］される．この関数は，int関数によって管理［図中(4)］される．

（c） int関数(int-function)：図4のユニット「interface-manager」の下位ユニット群に相当する．システム機能ユニット群をもとに，AIツールのインタフェース環境[8]や推論などを，zes関数を用いて実現［図中(5)］する．

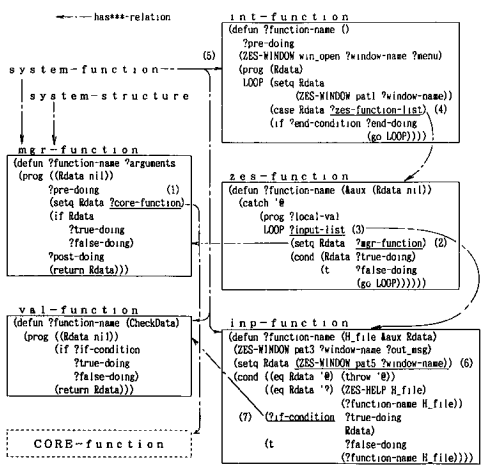


図7 システム管理ユニット群とその依存関係

（d） inp関数(inp-function)：図4のユニット「inp-manager」の下位ユニット群に相当する．システム機能ユニット「command-window」とzes関数をもとに，ユーザ入力［図中(6)］とデータの検証を実現する．データの検証は，val関数を用いて定義［図中(7)］される．

（e） val関数(val-function)：図4のユニット「val-manager」の下位ユニットに相当する．システム機能ユニット「datatype-list」とinp関数をもとに，入力データの検証を行う．

従来のAIツールでは，形宣言(float他)のみの定義であったが，図7の依存関係に従って，継承関係を考慮したデータの整合性を管理できるので，機械工学分野特有の単位(Pa, rad/sec他)が容易に定義できるようになった．例えば，断面(m^2)と力(N)，および単位面積当たりの応力(Pa)の関係が，全体-部分関係に依在する場合，その管理が容易である．

このほかにも，継承機能の範囲の規定や，特定のスロット値の参照時のみに起動する特殊な推論機能などを定義して，モデル表現を拡張することができるetc関数(etc-function)がある．この関数は，例えばユーザが他システムなどとの結合を要望する場合に，起動させるものである．

3. ソースコードの生成機構

図8は，2章のmgr関数のソースコードを生成する例を示したものである．この関数「mgr-get-slot」は，フレーム表現の推論機能の一つであるISA関係の継承機能を実現するものである．

ユニット「mgr-manager」のスロット「function-format」の値は，mgr関数を生成するテンプレートであり，テンプレート中の「?＊＊＊」は関数の変更箇所を示している．例えば，ユニット「mgr-get-slot」スロット「function-name, arguments, …」の値が，この変更箇所「?＊＊＊」に代入される．図中の番号は，ソースコードの生成手順を示し，順を追って述べる．

（1） システム管理ユニット群のインスタンスユニット(例えばユニット「mgr-get-slot」)が選択される．

（2） このユニットの関数生成のためのテンプレート(「function-format」の値：点線枠)が参照される．

（3） テンプレートの変数箇所が探索される．

（4） 変数箇所をスロット名として，このインスタンスユニットからスロット値が参照(あるいは評価)される．そして，その結果はテンプレートの変数箇所に代入される．例えば，スロット「false-doing」の下線部は評価を必要とする部分である．そこで，関数「core-

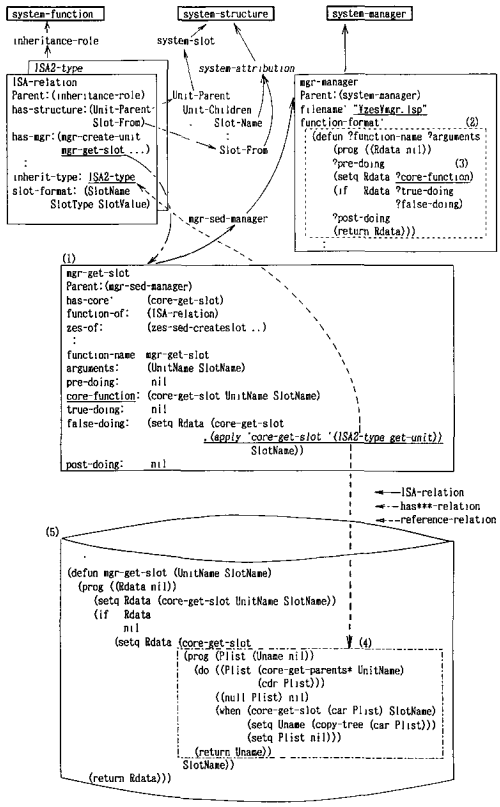


図 8　ソースコードの生成機構

get-slot」が実行され，ユニット「ISA 2-type」のスロット「get-unit」が参照され一点鎖線枠のように代入される．

（5）　新たなAIツールの関数として，ユニット「mgr-manager」で指定されたファイルに格納される．

このように，図7のテンプレート内のスロットが定義され，ソースコードが生成される．このテンプレートは，「前処理→判断→後処理」を基本構成とし，関数内が容易に理解できるようにした．また，ユニット「ISA 2-type」が示すように，基本的なAIツールの「機能」や「構造」は，あらかじめモデル表現内に登録されているので，専門分野固有の知識表現に応じて容易にAIツールが構築できる．ちなみに，図8中「ISA 2-type」を「ISA 1-type」に変更し，それをスロット「true-doing」の値とし，スロット「false-doing」の値をnilとすれば，図6のスロット値の探索時間を短縮した継承機能をもつAIツールが生成される．

図9～11は，図4のモデル表現を用いて，新たなAIツールを構築しているウィンドウ表示を示したものである．図9は，図5のユニット内のユニットの名

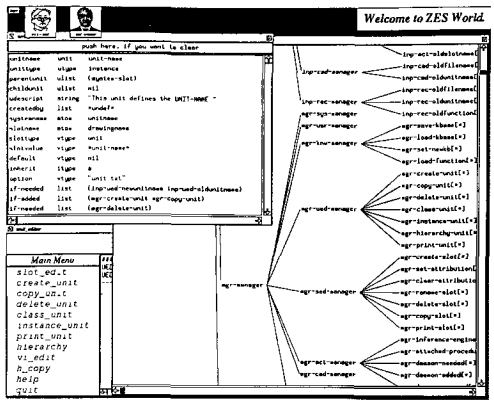


図 9　モデル表現を用いたユニット内表現の定義

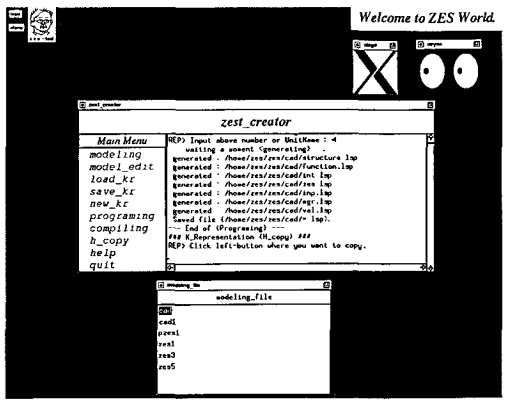


図 10　ソースコード生成の実行例

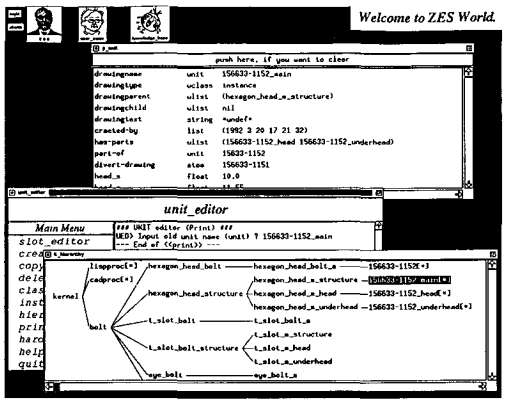


図 11　新たなAIツールによる知識ベースの構築例

称を「drawingname」に変更している例で，右部は図4のモデル表現の階層構造を表示している．図10は，図8のソースコード生成の実行例で，システム管理ユニット内の関数すべてが生成されていることを示している．また，図11は，図9でユニットの名称を変えた

際の新たなAIツールの例である．上部のユニット内表示は，ユニットの名称が「drawingname」と変更されているのが知れる．また，知識ベースの構築に際し，全体-部分関係(「has-parts」と「part-of」)が自動的に組込まれる．ちなみに，図9～11のAIツールの構築では，約1時間程度の作業時間(図10のソースコード生成時間35分含む)を要した．

4. 結　　言

本研究をまとめてみると以下のようになる．

（1） フレーム表現によるAIツールのモデル表現を，対象モデルの概念をもとに構築し，このモデル表現の有効性を示した．特に，モデル表現を用いてユニット内表現，スロット内表現が自由に定義できることを示した．これにより，専門分野固有の知識表現が容易に定義できるので，専門家の知識獲得に大いに期待できる．

（2） モデル表現の構築性では，依存関係，テンプレート，そしてあらかじめ登録されている基本的な関数により，容易に新たなAIツールが生成されることを示した．

（3） 知識ベースのデータ構造に「アトリビューション」の考え方を導入し，スロット内の管理について示した．これにより，知識ベース構築者の負荷が軽減でき，知識ベースの構築性を高めることができる．

文　　献

（1） Minsky, M., A Framework for Representating Knowledge, *Haugeland, J.,* (1981), 95-128, Mind Design, MIT.
（2） Kehler, T. P. and Clemenson, G. D., *Knowledge Eng. Enviroment for industry,* (1983), 573-581, IntelliCorp.
（3） Smith, R. G., Support for Structured Object Knowledge Representation, *STROBE,* (1983), 855-858, IJCAI.
（4） Ogawa, Y., Shima, K., Sugawara, T. and Takagi, S., Knowledge Representation and Inference Enviroment, *Int. Conference on Fifth Generation Comput, Systems,* (1984), 643-651.
（5） 上野，通学会技法，AI 86-4(1986)，21-28.
（6） 長坂・ほか2名，機論，58-547，C(1992)，975.
（7） 長坂・ほか2名，機論，58-549，C(1992)，1680.
（8） 長坂・ほか4名，機論，59-560，C(1993)，1321.