ICOT Technical Memorandum: TM-0588

TM-0588

合成型知識システムにおける知識獲得支援

澤本潤，椿和弘，滝寛和，藤井裕一

August, 1988

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

合成型知識システムにおける知識獲得支援

総論

注　（未）の章・節は、本文未完成のために本報告書には記載していません。

総　論

知識システムの対象は、解析型問題と合成型問題とに分けられる。解析型問題では、システムの構造および構成要素の機能は所与であり、外部からの入力が与えられた場合、システムがどのような挙動を示すかを明らかにすることが主たるタスクとされる。診断問題、解釈問題、制御問題などは解析型問題の典型例である。一方、合成型問題では、システムに対する入力および出力の要求仕様が与えられた場合、これを実現するシステムの構造および構成要素の機能を決定することである。計画問題や設計問題は合成型問題の典型例である。合成型問題は解析型問題を内包しているともいえる。

知識システムの研究開発は、解析問題を中心に進められ、現在では研究の焦点は合成型問題へとシフトしつつある。合成型知識システムの開発がプロトタイプレベルに留っているのに対し、解析型知識システムの開発は実用化の時期を迎えようとしている。しかし実用化に際し、最大の障害は知識獲得過程にあることが現場の多くの技術者から指摘されている。人間の専門家がもつ知識を抽出し、変換する過程を支援する方法論およびツールの開発が強く要請されているといえよう。

これに対し、合成型知識システムにおいては、より上流側すなわち問題の定式化支援や問題解決の基本制御ループの設定において、これを支援する方法論の確立が要請されているのが現状とみられる。

したがって、解析型問題に関する知識獲得支援と合成型問題に対する知識獲得支援では、技術的および方法論的にみて、質的に異なった側面があることを認識しておく必要があると思われる。

このような現状認識に基づいて、本報告書は合成グループの作業結果をまとめたものである。本報告書の狙いは、合成型知識システムにおける知識獲得問題に焦点を絞り、合成型問題の持つ特徴、従来技術と人工知能技術との関連、さらに、合成型問題における知識獲得の事例を分析し、合成型知識の獲得の特徴、知識獲得支援機能を実現する課題についてまとめたものであるが、合成型知識システムのための知識獲得支援システムを提案するまでには至っていない。

東京大学工学部吉川弘之教授によれば、合成（シンセシス）型問題とは、抽象的なものから具体的なものへと概念操作を行って、最終的な実体に至る過程を扱うものである。これに対して、具体的な物（実体）から出発し、最終的に抽象的な概念へと、全体像を明らかにしていく過程は分析的な行為であり、解析（アナリシス）型問題と言われる。

合成型問題の代表例は、計画あるいは設計の問題であり、両者は制約条件下で所与の問題を解決するという点、一般的に探索空間が非常に大きい点など共通点が多い。

計画問題には、i)ある制約条件下で、所与の目的を達成するための手順を作成する問題（いわゆる、スケジューリングや工程設計がこれに相当する）、ii）ある制約条件下で、所与の対象についてのより適切な組合せを探し出す問題（運用計画、負荷配分、配置など

が相当）がある。

一方、設計問題は、前出の吉川教授によれば、次のように定義されている。すなわち、設計の対象となる物に対する概念には、いろいろな種類があるが、大きく分けると、実体（Entity）、属性（Attribute）、機能（Function）、価値（Value）の４段階が考えられる。ここで、設計というのは、これら４つの概念の中で、機能を与えられて属性を発見することである。このように機能の概念から、属性の概念への変換を行うことが、設計ということである。さらに、価値の概念から始まり実体の概念に至る一連の流れを広い意味での設計という。広い意味での設計には、開発ないしは基礎研究、さらには経営的要素も含まれる。一般に設計を支援するシステムというのは、狭い意味での設計、つまり機能から属性に至る部分を支援するものである。

すなわち、設計問題とは機能に対する要求仕様と、設計の対象となるモデルが満たすべき制約条件下で、適切な設計案を導出する問題であると言うことができよう。

さて、合成型知識システムでは、前述のように、人間の知的活動のかなり高度な部分を扱う必要があり、知識獲得は言うに及ばず、システム自体もプロトタイプレベルに留まっているものが多い。このような段階で合成型知識システムのための知識獲得支援システムのあり方について論ずるためには、必要とする情報があまりにも少ないとしか言いようがないが、２年間の活動の区切りをつけるため本報告書をとりまとめる運びとなった。

本報告書は、「合成型知識システムにおける知識獲得支援」と題し、全５章から構成されており、第１章以降の内容は以下のとおりである。

第１章は、合成型解決モデルの考察について整理している。

第２章は、合成型問題の解決手法として従来から適用されてきた技術との対比によって、人工知能技術、特に知識獲得に期待される機能について、人工知能技術への期待という形でまとめている。

第３章は、事例研究であり、毛筆文字の生成、事務処理システム、ディジタルＬＳＩ設計、アナログＬＳＩ設計、及び、航空機設計を事例として検討を加えたものである。

第４章は、４章での事例研究を踏まえつつ、合成型知識獲得の特徴についてまとめている。

第５章は、本来は知識獲得支援システムのイメージを提案するところであるが、今はその段階ではないので、オブジェクト指向アプローチ、マンマシンインターフェイス等の観点から、知識獲得支援機能を実現するための課題についてまとめたものである。

なお、本報告書の姉妹編として、ＴＭ－581 に、「解析型知識システムにおける知識獲得支援」がまとめられているので参考とされたい。

本報告書は、ＩＣＯＴ知識獲得支援システム作業グループ（ＫＡＳ－ＷＧ）の委員（表１）の内、合成型タスクグループのメンバーの労によるものである。

尚、森　啓（日電）、丸山文宏（富士通）、渡辺正信（日電）の諸氏にはＫＡＳ－ＷＧで講演をおこなっていただいた内容について、本報告書に含めるべく、執筆の労を取って

いただいた。

表1　　ＫＡＳ－ＷＧの委員

＜タスクグループ＞

諏訪　基	（電総研）	合成型問題	ＴＧ主査
寺野　隆雄	（電力中研）	〃	ＴＧとりまとめ
麻生　盛敏	（日本電気）	〃	
石井　暁	（東芝）	〃	
市川　雅也	（三菱重工業）	〃	
真田　英彦	（大阪大）	合成型問題	
渡辺　俊典	（日立製作所）	〃	
小林　重信	（東工大）	解析型問題	ＴＧ主査
国藤　進	（富士通）		ＴＧとりまとめ
田中　博	（東大）	〃	
長田　享一	（石油資源開発）	〃	
藤原　良一	（三菱電機）	〃	
門前　弘邦	（富士通）	〃	

なおＩＣＯＴの下記の研究者が討議に加わった。

岩下　安男	（第５研究室長）	合成型問題
椿　　和弘	（第５研究室）	〃
岡　　夏樹	（第１研究室）	〃
澤本　潤	（第５研究室）	解析型問題
滝　　寛和	（第５研究室）	

1．合成型問題

1.1　合成型問題のクラスと類型的タスク

1.1.1　解析型問題と合成型問題の比較

知識システムが対象とする問題のクラスは、大別して解析型問題（Analysis Problem ）と合成型問題（Synthesis Problem)に分けられる。解析型問題では、図1.1-1 に示すように、システムの構成要素の特性およびそれらのつながり方（システム構造）は所与であり、入力に対してシステムの特性がどのようになるのかを推定することが要請される。

一方、合成型問題では、図1.1-2 に示すように、システムの特性が仕様として与えられたとき、これを実現するようなシステム構造および構成要素の特性を決定することが要請される。

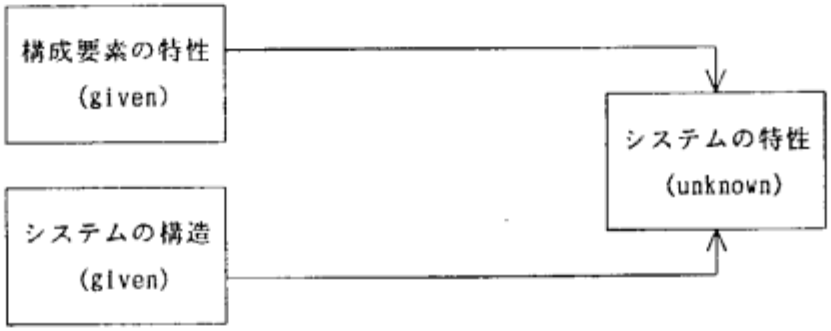


図1.1-1　解析型問題

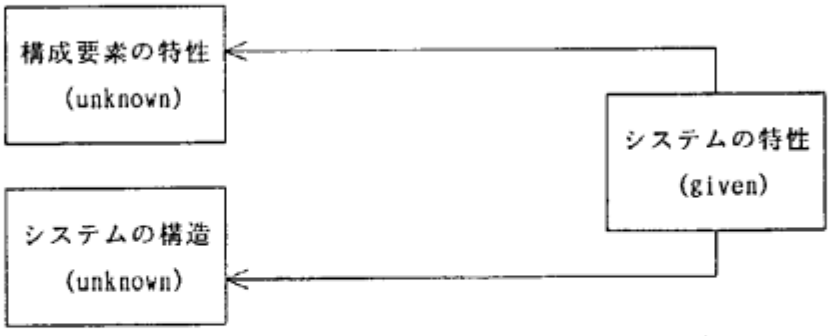


図1.1-2　合成型問題（クラス１）

システム構造と構成要素の特性が与えられたとき、システムの特性は一意に定まる。これに対し、システムの特性が与えられたとき、これを実現するシステム構造および構成要素の組み合わせは一般に無限に存在する。ここに、解析型問題と比べて、合成型問題の基本的な難しさがある。

合成型問題に対する基本的接近法のパラダイムは、生成検査法（Generate and Test）に求めることができる。生成検査法とは、システムを仮に生成（Generate）し、これを解析することにより、システムの特性を決定し、それが要求仕様を満足しているかどうかを検査（Test）することを繰り返して、最適なシステムを探索する方法である。この方法は解析による合成（Synthesis by Analysis)とも呼ばれる。この意味において、合成型問題は解析型問題を内包しているといえる。

1.1.2　合成型問題のクラス

もっとも一般的な合成型問題（クラス１と呼ぶ。図1.1.2 を参照のこと）は、つぎのような２レベル決定問題として定式化することができる。

［合成型問題（クラス１）］
　optimize　＜システムの構造＞
　　subject to
　　　optimize　＜構成要素の特性＞

すなわち、クラス１の合成型問題は＜構成要素の特性＞が最適に決定されるという制約条件のもとに最適な＜システムの構造＞を見出すことである。これら２つの副問題は独立ではなく、相互依存の関係にあることに注意されたい。＜構成要素の特性＞について最適決定するためには、＜システムの構造＞が事前に決定されていることを必要とし、また＜システムの構造＞について最適決定するためには、＜構造要素の特性＞が事前に決定されていることを必要とする。したがって、クラス１の合成型問題では、必然的に後戻りを必要とする試行錯誤的探索が不可避である。

より特殊化された合成型問題として、＜システムの構造＞または＜構成要素の特性＞のいずれか一方が所与の問題が考えられる。＜構成要素の特性＞が所与の合成型問題をクラス２の合成型問題と呼ぶことにすれば、つぎのように定式化される。

［合成型問題（クラス２）］
　optimize　＜システムの構造＞
　　subject to
　　　＜構成要素の特性＞　is given

クラス２の問題では、〈構成要素の特性〉は与えられているので、〈構成要素〉を空間的または時間的にどのように組み合わせるかが問題とされる。たとえば、レイアウト問題やスケジューリング問題などがこのクラスの問題に含まれる。

一方、〈システムの構造〉が所与の合成型問題をクラス３の合成型問題と呼ぶことにすれば、つぎのように定式化される。

［合成型問題（クラス３）］
　optimize　〈構成要素の特性〉
　　subject to
　　　〈システムの構造〉　is given

クラス３の問題では、〈システムの構造〉は与えられているので、構造の各部分に対して、競合する構成要素の中からもっとも適切なものを選択し、その特性を決定することが要請される。たとえば、機械部品や装置の定型的な設計問題などがこのクラスの問題に含まれる。

図1.1-3 と図1.1-4 に、クラス２とクラス３の合成型問題における〈システムの特性〉、〈システムの構造〉および〈構成要素の特性〉の関係を示す。

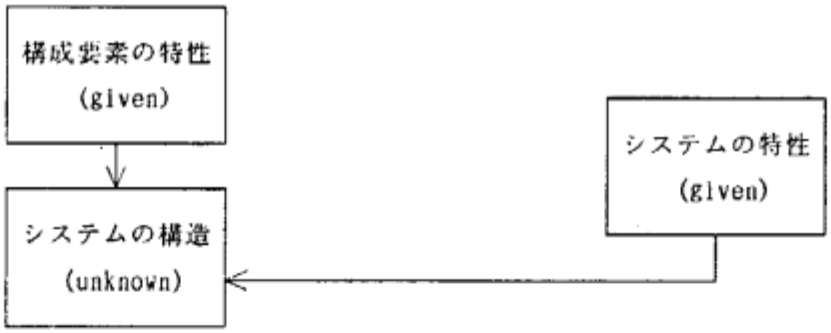


図1.1-3　合成型問題（クラス２）

図1.1-4　合成型問題（クラス３）

1.1.3　合成型問題における類型的タスク

Chandrasekaranは、従来の知識システムがルールやフレームのような実装レベルでの知識表現を中心に組織化されたことによる問題点を指摘したうえで、問題解決を中心に組織化することの必要性を指摘し、Generic Task（類型的タスク）という考えを提唱している。この考えは、知識処理の対象とされる問題の解決をいくつかの類型的タスクの組み合わせによって実現しようとするものである。

1.1.2 節で議論した合成型問題のクラスにしたがって、各クラスの問題解決において典型的と考えられる類型的タスクを考察することとする。

［1］クラス３の合成型問題における類型的タスク

Chandrasekaranは、２節の分類でいうクラス三の合成型問題を対象としてタスクの類型化を試みている。クラス３ではシステム構造は所与であるので、これを階層木に対応させ、計画選択・精密化による階層的設計（Hierarchical Desigy by Plan Selection and Refinement ）と呼ばれる類型的タスクを提案している。このタスクを実現するために、階層木の各ノードにはスペシャリストと呼ばれる推論モジュールが付加される。このモジュールは計画選択部と精密化部からなる。計画選択部ではそのノードに割りつけるべき構成要素の選択およびその特性の決定を行う。精密化部ではつぎに計画すべきノードを指示する。これらはスペシャリスト間のメッセージパッシング機構によって実現され、必要に応じて上位階層への後戻りが行われる。

階層的設計において構成要素の特性を決定するためには、それがシステム全体の特性にどのような影響を与えるかを推定する必要がある。ある階層における構成要素の特性の上位階層への定性的な影響を予測するために、Chandrasekaranは状態抽象化（State Abstraction)と呼ばれる類型的タスクを提案している。このタスクを実現するために、consolidation と呼ばれる定性推論の一種を利用している。

以上二つの類型的タスクは階層的生成検査法の１つの具体化を与えるものである。

［2］クラス２の合成型問題における類型的タスク

クラス２の合成型問題では、構成要素は所与であるが、システム構造は未知である。構成要素のつながりをすべて同時に考慮することが組合せ的複雑さのために困難である。普通に取られる戦略は、構成要素群の中で、空間的および時間的に重要と考えられるものをいくつか取り出し、他は考慮しないで、まずそれらの間の部分的システム構造を決定する。ついでこの部分的システム構造を制約条件として、つぎに重要と思われる構成要素群を考慮に入れ、それらの間のつながりを決定するとともに、すでにつくられた部分的システム構造の中に埋め込むことを行う。このよ

うな過程はすべての構成要素が考慮されるまで再帰的に繰り返される。この戦略はトップダウン精密化（topdown refinement）と呼ばれる。

上で述べたように、クラス２の合成型問題に対する基本的な問題解決戦略はトップダウン精密化である。トップダウン精密化による設計を可能とするためには、

１）構成要素の集合をいくつかの部分集合に分割すること。

２）部分集合内での部分的システム構造の決定が可能であること。

３）上位階層での部分的シテム構造の決定がそれより解の問題に対し、制約条件として機能すること。

４）上位レベルでの決定が下位レベルでの実行可能性を侵害しないこと。

１）を実現するためには、与えられた構成要素の集合をある評価基準のもとで最適に分割するためのタスクを必要とする。２）を実現するためには、部分的システム構造を見出すための探索タスクを必要とする。３）および４）を実現するためには、制約条件の伝播および実行可能性の評価を必要とする。

まとめると、クラス２の合成型問題に対する問題解決戦略としてトップタウン精密化を採用すると、これを実現するための類型的タスクとして、構成要素の階層化および部分構造の最適化（制約の伝播と実行可能性の評価を含む）などが必要とされる。

［３］クラス１の合成型問題における類型的タスク

クラス１の合成型問題では、システム構造も構成要素特性も未知である。したがって階層的生成検査法やトップダウン精密化によるアプローチは困難である。クラス１の問題に対しよくとられる方法は、過去の設計事例を参考にして、これに手を加えて要求仕様を満たすように改良ることである。このプロセスをシステム化するためには、設計事例を知識ベース化することとともに、現在の設計作業にもっとも参考になる類似の設計事例を効率よく検索し、利用するための類似および類推の機能をシステムにもたせる必要がある。このプロセスは１つの類型的タスクとしてまとめるとが可能と思われる。

既存の設計事例が守備よく利用可能になったとしても、これを改良するためには種々のパッチワークを必要とする。代表的なパッチワークは摂動法と呼ばれるものでシステム構造の部分的変更または構成要素の部分的変更がシステム全体に与える影響を評価すること、すなわち感度解析を行うこととにより、もっとも影響の大きな部分についてこれを代替的な構造や特性によって置き換えることにより、シテムの性能を段階的に高めていくものである。このプロセスも１つの類型的タスクとしてまとめることが可能と思われる。

以上三つのクラスの合成型問題における類型的タスクを考察したが、各問題に対する基本的な接近法および類型的タスクをまとめると、次表のようになる。

合成型問題	接　近　法	類　型　的　タ　ス　ク
クラス３	階層的生成検査法	計画の選択と精密化、状態抽象化
クラス２	トップダウン精密化	構成要素の階層化、部分構造最適化
クラス１	設計事例の手直し	類比と類推、パッチワーク

1.1.4　おわりに

合成型問題についての一般的な考察を行い、３つのクラスの問題を定義し、各問題における類型的タスクを考察した。ここでの議論はトップダウン的であり、抽象的な段階にとどまっているが、現実の合成型問題はもっと多様であり、問題のクラスをより詳細に分類し、事例との対応づけを行いながら、合成型問題の類型化を進めていけば、合成型問題に対する知識システム構築の方法が体系化されてくるのではないかと期待される。

参考文献

[小林 86]　小林重信：知識工学、昭晃堂 (1986)

[小林 87]　小林重信：知識工学方法論、システムと制御、Vol.31,No.4,pp.275-285 (1987) .

[Chandrasekaran 86]　Chandrasekaran,B.:Generic Tasks in Knowledge based Reasoning:High-Level Building Blocks for Expert System Design,IEEE Expert,Fall,pp.23-30 (1986) .

1．2　合成型問題解決のモデル

1.2.1　合成型問題解決の諸側面

1.2.1.1　はじめに

思弁の方法については、デカルトの演繹、ベーコンの帰納が古来より有名である。前者は対象の全体像を先ず仮定し、その妥当性検証を部分的データよりおこなう。後者は逆に部分から全体に至ろうとする。弱い精神の力で複雑な対象に対応するには前者が不可欠であるとしたデカルトの思想は、近代科学の中で様々な形で立証されてきた。生物の行動パターンの学習には生得的機構の存在が必要であるとしたローレンツやティンベルゲンの主張や、言語能力についてのチョムスキーの同様な思想を始め、様々な領域で「あらかじめ内在する物の存在」が知的なふるまいの前提要件であることが認識されて来た。合成的問題解決システム（以下、計画／設計型問題解決システムとよぶ）という高度に知的な活動と深くかかわる情報処理系にあっては、「内在する物」の研究が先ず重要である。これとは別に、全てのシステムは環境への適応という生態学的側面を持っていることも事実である。生物学や言語学で古くから主張されて来た、「システムは環境のコピーである」ということは、計画／設計型問題解決システムについても例外ではなく、設計対象物自身の性質のコピーとなっている点も見逃すことはできない。

さて、計画／設計型問題の計算機による解決は古くから研究されて来たものではあるが、ここ数年来の知識処理発展の中で新しい時期をむかえつつあるように感じられる。これは特に数値処理から記号処理への移行の進展によってもたらされつつあるもので、特に従来手つかずの状況にあった上流設計の計算機化傾向がみられるようになってきている。ここでは、計画／設計型問題解決法の代表的立場にあった数値計画法をひとつのガイドラインとして計画／設計型問題解決の仮説的モデルについての考察をおこなってみたい。

1.2.1.2　モデル考察の視座

計画／設計型問題は一般に、Ｄを集合、Ｒを集合要素間の関係とした時、Ｄの要素でありかつＲを満足するものを探索する問題とみなされる。場合によっては１次元あるいは多次元の評価尺度Ｊが存在し、Ｊの値のなるべく小さいものを探すという条件が課せられることがある。以上は問題の静的、表示的側面であり、これを実際に解くという動的過程すなわち操作的側面も重要である。対象とする製品の性質や時間的変化、あるいは設計の上・下流フェイズの差に依存してＤ、Ｒ、Ｊなどは様々に様相を変える。

以上より、計画／設計型知識システムを考察する場合の視座として下記をあげたい

と思う。（図1.2-1）。

・表示及び操作的側面

・環境適応的側面

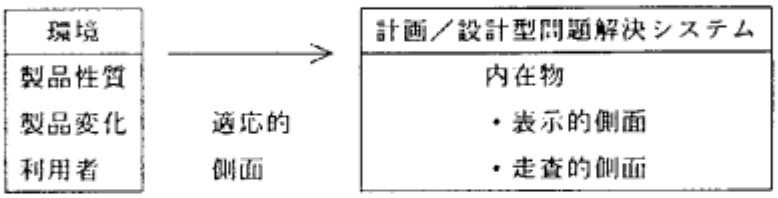


図1.2-1 計画／設計型システム分析視座

1.2.1.3　合成型問題解決の仮説的モデル

計画／設計型問題解決の仮説的モデルとして、ここでは次のものを考えてみたい。

計画・設計型問題解決

－　初期動機　・・・・・・・・・創造的

＋要素定義＋相関定義＋目標定義＋解・・・・表示的

＋求解　・・・・・・・・・操作的

＋適応　・・・・・・・・・生態学的

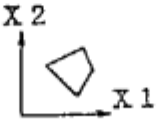
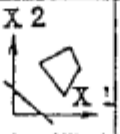
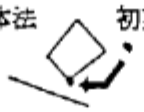
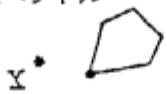
初期動機は、問題発見等の多分に創造的は、要素定義～解は表示的静的、求解は操作的動的、適応は生態学的な側面に対応するとみる。

このモデルは前述した通り、数理計画法の諸側面を規範にしたものであるため、それがよく適合する線形連続系に対しては各々側面の意味内容は明白であるが、離散記号系に対しての各々の側面の意味内容は必ずしも明白ではない。そこで線形連続系の代表として輸送問題を、離散系に対しては古典論理学にもとづく線分生成問題を例として各々の意味内容を表1.2-1 にまとめてみる。

線形連続系においては数値ベクトルが基本要素であり、それらの間の関係は線形不等式群、目標はひとつの線形式で与えられる。求解は単体法により、解は数値ベクトルとなる。古典論理学モデルの場合には記号間の関係は述語で、目標は目的述語で、求解は推論によっておこなわれる。推論法としては導出法が著名である。解は項とよばれる記号構造体となる。

計画／設計型問題解決の適応的側面としては、設計対象製品の性格、たとえば機械系、電気系の差や、同一系統内でも複雑度の差、製品のライフサイクルの様相などに

表1.2-1　計画／設計型問題解決システムの表示・操作的側面

	線形連続系	離散記号系
初期動機	創造など	創造など
要素定義	数値ベクトル$\in R^n$ 輸送量X，容量C	離散集合E
相関定義	不等式群 $X2$ $BX \leqq C$  $X1$	述語 point(p),point(q), line(A,B,line(A,B)):- point(A),point(B).
目標定義	目的関数 $X2$ Min AX  $X1$	目的述語 -line(A,B,M).
求解	単体法 初期解 	推論規則
解	数値ベクトル  X^*	項 M=line(p,q).

応じて多様なものとなる。また特定製品に限定した場合でも、上位設計、下位設計といった設計階層によって特徴も異なってくる。表1.2-2 に前述の諸側面が、製品の繰り返し性や複雑度あるいは設計階層などによって変化する状況を概括する。

表1.2-2　計画／設計型問題解決システムの適応的側面

	繰返し性		複雑度		設計階層	
	新規	繰り返し	単品	システム	上位	下位
初期動機	新製品	改良	ー	ー	ー	ー
要素定義	ー	ー	単調	多様	記号	パラメタ
相関定義	新規	部分変更	平面的	階層的	パターン 図面	数式
目標定義	新規	部分変更	一元的	多元的	仕様概略	仕様詳細
求解	新規	旧解変更	最適化 全域的	満足化 手順化 局所的	再利用 検索	パラメタ 最適化
解	新規	旧解	ー	保存 再利用	トポロジ カル	パラメト リック

1.2.2　諸側面への対処の方法について

計画／設計型問題解決の諸側面は以上のように多様なバラエティを持っている。ここでバラエティに応じた対処法と知識獲得という面からみた処方について概観する。尚、現実世界での合成問題解決は一般に相当程度複雑であり、知識獲得という言葉でニュアンスされる高度な自己組織化能力を期待することは、技術の現状では困難と思われる。従って、ここでは問題解決システムを計算機上に実現することの容易性という観点でこの言葉を解釈したい。（表1.2-3)。

1.2.2.1　初期動機について

高度に創造的な側面であり、対処法としては従来よりＫＪ法に代表される発想法の提案がある。知識獲得という点からは最も困難な側面であるが、物理データから背後の物理法則を発見するH．サイモンらの人工知能システム研究はこの方面への先駆的

なものである。

1.2.2.2　要素定義について

要素定義については、数値から記号、次元数小から大、時不変から事変へと様々になバラエティに富み、それぞれに応じてユークリッド空間での表現から記号体系での表現、平面的な表現から階層的な表現へ、固定的構造から・柔構造へ、特にメタレベル定義を用いて多様な変化にはインスタンスを生成してその都度対処するといった方法が重要である。

知識獲得という点からは、問題定義用新言語の開発によって問題自体を効率的に計算機上に定義することが現実的なシステムの場合への対処法といえる。線形計画法は線形最適化問題むき言語でありＰＳ（プロダクション・システム）は診断等のルール型知識が中心となってシステムの定義にむいた言語である。

1.2.2.3　相関定義について

要素間の関係条件は、線形から非線形にわたって多様である。対象が複雑になってくると制約条件の記述が完全に可能ではなくなり、問題解決に先立って不充分な情報しか与え得ない状況が発生する。輸送問題における新輸送路の発生など、要素間の関係条件が時間と共に変化する場合も多い。

対処法からみると、非線形性の増大と共に線形計画法から非線形計画法、さらには公理／代数系の使用が必要となる。不充分情報条件のもとでは旧解を再利用してそれに含まれる情報を最大限、再利用するといった方法がおこなわれることも多い。もちろん情報をおぎなうことは通常おこなわれている通りである。時変性に対しては関係条件の定義自体の柔構造化やメタレベル化が重要である。

知識獲得という観点からみると、関係定義の容易化、旧問題の再利用能力、関係に関するメタレベル知識を用いた知的対話能力の恒常などが有用である。

1.2.2.4　目標定義

目標指標もまた線形から非線形に、単一目標から複数目標に、最適化基準から満足化基準にと、多様なバラエティを持っている。一次元計量化できない場合には序数化するとか、重要でない指標は無視するという目標緩和操作をおこなう。

1.2.2.5　求解過程

求解に要するコストが小さいか大きいかによって対応は異なる。求解コストが小さい場合には求解操作を毎回おこなうが、求解コストが大なる場合には旧解を再利用してその摂動解を利用したり、分割と統合の原理によって求解コストを低下させることが重要である。

求解過程は、解の状態を変換する状態変換作用素と解釈できる。この作用素がどの程度明確であるかによって対応は異なってくる。不明な場合には候補解をしらみつぶしに生成して制約条件を満たすものを見出すという生成検証法が使用される。局所的作用素のみが明らかな場合には断片的なルールがその表現にむいている。大域的作用素が明らかな場合にはアルゴリズムの形で記述される。作用素が状態に及ぼす影響という観点からオンライン的な作用素とオフライン的な作用素とに分けてみることも興味深い。前者は新しい構造の状態が作用素によって時間経過と共に生成される場合にあたっており、状態の将来についての予測行為が求解の効率化に有効となる。後者は状態の構造が未来に渡って観測可能な場合にあたっており、動的計画法などの大域的最適解探索が有効となる。

知識獲得という面からみると、旧解を利用して求解コトを低下させる方法や、解法自体を生成、利用、改良する方法が有効となる。チャンキングとよばれる求解過程の自律的改良法は後者の例である。

1.2.2.6　解

線形計画法では解は多次元ベクトルとなるが、一般にはもっと複雑・多様であって、図面などの形式をとるため、多元的情報の記憶検索が重要である。解の寿命が長い場合には単に解を記憶保持して再利用すればよいが、寿命が短い場合には容易に解をモディファイできる点が重要となる。

2．従来技術と人工知能技術との関連

この章では従来技術との対比により人工知能技術の特質を考察し、人工知能技術、特に知識獲得技術に期待される機能を考察する。

以下の３つの節で、まず計算機構成決定問題を例にとり、手順のはっきりした合成問題における知識獲得、次にリソース割当て問題を例にとり、専門家の直感に大きく依存する問題における知識獲得、最後に最適化問題を例にとり、数理計画法の立場から見た知識獲得について論じる。

これらの議論により、人工知能技術に対する期待、その可能性を明らかにする。

2．1　計算機構成決定支援問題

計算機構成決定支援問題とは、計算機の構成の設計の支援を行わせようとする問題である。計算機の構成要素で接続不可の物を指摘し、またケーブル、シャーシなど付属品の選択等を行う事が代表的な機能である。

具体的に次の２つの問題を例にとる。

（１）ある計算機とある端末を使用する際、それらを接続する必要なケーブルを選ぶ。
（２）シャーシに２枚のプリント板を挿入する際、上下２枚分のスロットの各々にどちらのプリント板を挿入するかを決める。ただし、逆に挿入しても計算機は正常に作動するものとする。

各々、専門の設計者は次の様に解答を得ているとする。

（１）その端末についているコネクタの形に合わせてケーブルを選ぶ。
（２）どちらを上側に配置するかが決まっており、それに従って配置する。

2.1.1　問題の特徴

この問題の特徴は、ある決まった解答を作成する事ができ、更に作成しなければならないという点にある。その解答には、何ら設計者の創造性が反映されない。また、反映されてはならない。（１）の問題においては、そのケーブルでなければ接続できないので、それを選ぶのは当然である。（２）の問題においても、正常動作するからと言って勝手に決めてはならない。後の製造、保守の際の無用なトラブルを防止するためである。

工業の現場では、この主の“道の決まった”問題が多い。設計者は未知を踏みはず

さない様に進めばよい。システムをのけるのは、いかに道をはずさず、かつ早く解答を得るかである。この種の問題を軽視してはならない。優れた工業製品はこの種の問題を量的にも、場合によっては、質的にも限られた設計者で迅速に解く事によって得られている事が多い。

2.1.2　使用知識

２つの例題を解くための知識がどの様に表されるか、何通りかの方法を考える。まず、人間の読むマニュアルには次の様に書いてあるだろう。

（１）ＡＡ端末にはコネクタが付属していないので、ＢＢケーブルを用意せよ。
（２）ＣＣプリント板はＤＤプリント板よりも上側に配置せよ。

実際には、これらの記述が大量ののマニュアルのどこかにうめ込まれている。人間も見落し易いし、計算機の自然言語処理機能には荷が重い。そこで定式化された計算機言語で書き表すことになる。

プロダクションルールで書くと、次の様になろう。

（１）もし端末がＡＡ
　　ならば必要なケーブルはＢＢ
（２）もしＣＣプリント板があり
　　かつＤＤプリント板がある
　　ならばＣＣプリント板を上側に配置

この様な書き型で一応は知識を書き表す事が可能であるが問題が残る。即ち、端末なりケーブルなり、プリント板なりの新製品が出た時にどれだけの数のルールに変更が必要か、それにどの程度の広い知識が必要かを考えると、この書き型は優れた書き方とは言えない。

次の様に書く方が扱い易い。

（１）もし端末がＡＡ
　　ならばコネクタは××

　　もしコネクタが××
　　ならば必要なケーブルはＢＢ

（２）もしプリント板がある
　　ならば配置順にソート

即ち、（1）についてはコネクタの形状をインタフェースにして端末とケーブルの2つのルールに分ける事によりルールの局所性をふやし、分り易いルールとなる。また（2）についてはソートといった手続き的な処理を導入する方が同種の多量のルールを省く事ができ、見易い。

2.1.3　知識獲得

知識表現の改良で検討した事項は、従来の技術に通じる面がある。即ち（1）はリレーショナルデータベースにおけるリレーションの設計、また（2）は（たぶん配列やループを用いた）ソート技術を思い出させる。ルールでの記述を従来からの技術と対立した物と考えてはならない。それぞれ助け合って優れたシステムとなる。

以上の考察から計算機構成決定支援を行うための知識獲得の流れは第2-1 図の様になると考えられる。資料に書かれた知識は端末等の装置ごとに分離され、また知識の変型を行うことなくシステムに与えられる。以後はシステムが人間の介入なしに扱う。その際、データベースの活用と手続き型言語による処理が重要な要素になる。

2．2　リソース割当て問題

次にリソース割当て問題を例にとる。この問題は、第2-2 図に示す様に、例えば看護婦の勤務スケジュール作成の様にあるリソース（看護婦）に定められたジョブ（勤務）を割当てて、全ジョブをこなそうとする問題である。この際、リソースの持つ資格とジョブの要求する資格を適合させ、またリソースの制約（月刊勤務時間の制限や休暇予定）を満たさなければならない。

この種のリソース割当て問題は人員の配置だけでなく、機械の使用計画他、多くのバラエティに富んだ問題がある。現実の問題においては、割当ての巧拙が直接経済性に結びつくため、多くの場合、専門家による割当てが行われている。

2.2.1　問題の特徴

この種の問題は多くの特徴、困難な性質を持っている。

（1）解法の手順がない

数理計画法等により解析的に解ける事は、現実の問題においては期待できない。シラミツブシ法をとるには解空間が大きすぎる。また、解の存在も保証されていない。

（2）最適な割当ての評価基準がはっきりしない

単に全ジョブをこなしただけでは優れた割当てとは言えず、より良い割当てが

求められる。ところが、その評価基準が必要コスト等ではっきり求まるケースは少く、多くの場合、評価基準はあいまいである。しかしながら良い割当てか否かは専門家によれば判断可能であり、専門家はそれを求めている。

2.2.2　使用知識

専門家はこの問題を次の様な知識で解いている。

（1）制約のきついジョブから割当てる

制約のきつい、つまり難しいジョブから割当てていく。難しいジョブを担当できるリソースは少いので、それを有効に使おうとするものである。

（2）制約のきついリソースから割当てる

制約のきつい、つまり能力の少いリソースから割当ていく。能力の少いリソースを遊ばせない様にしようとするものである。

（3）時刻の早いジョブから割当る

時刻の早いジョブから（問題によっては遅いジョブから）割当ていく。

（4）多くのやり方を組み合せる

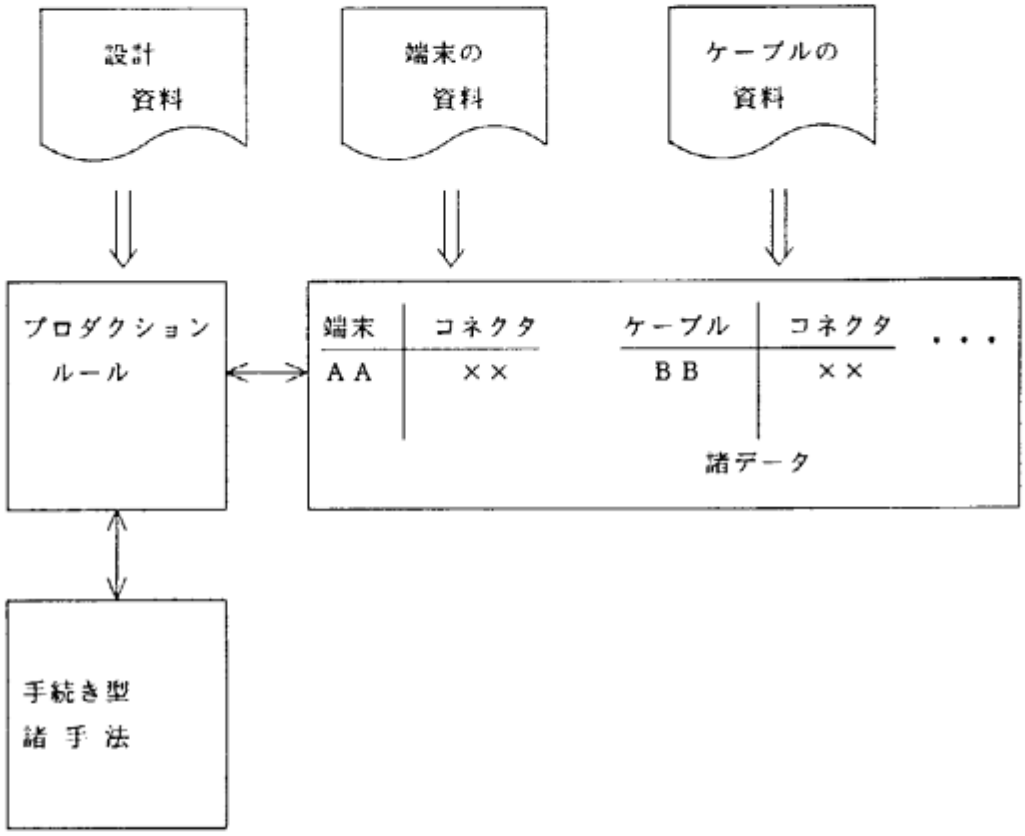
1つのやり方だけでは問題を解けず、多くのやり方を組み合せる。例えば、上記（3）のやり方は、まず（1）や（2）のやり方をある程度進めてから行う。どの程度まで進めてから（3）のやり方へ移るかは、ジョブやリリソースの状態を見て決定する。

2.2.3　知識獲得

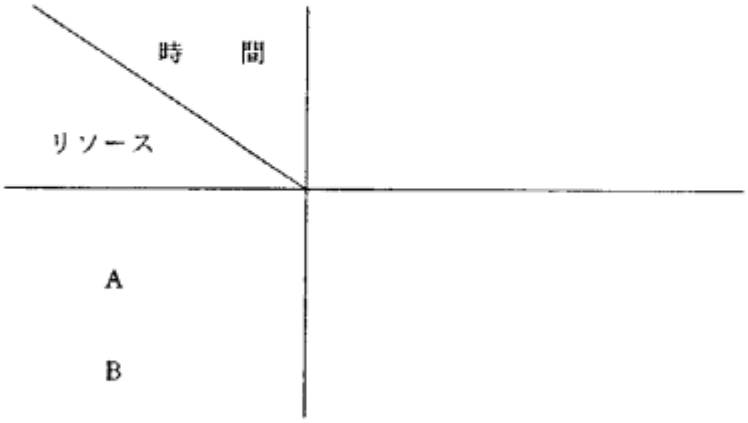
この問題の知識獲得は、人間が行うにせよ、計算機が行うにせよ、専門家がどの様な判断基準でどのやり方を選択したかを知る事に中心がある。この問題を専門家が解くには、第2-3 図に示す様に、まず問題の状態を評価し、それに従ってやり方を選択し、それを用いて数ステップ（数ジョブなど）の割当てを行う。それにより状態が変化するので、また状態の評価からくり返す。

このくり返しのなかで、中心となるのはどの様に状態を評価し、やり方を選択するかどある。それを明らかにするために、図で2重で囲った様に、問題の状態を示すパラメータを得て、それのいずれかが重要であるか、またそれだけでは不足かを判断していく事が必要となる。

この問題の知識獲得を全く自動的に行う事は事実上困難である。そこで、知識獲得の工夫の中心は専門家の割当て手順を容易に記録する手段、また、それを計算機によって再現するため、状態の評価ややり方の選択の基準を容易に変更できる手段を提供する事におかれる事になろう。



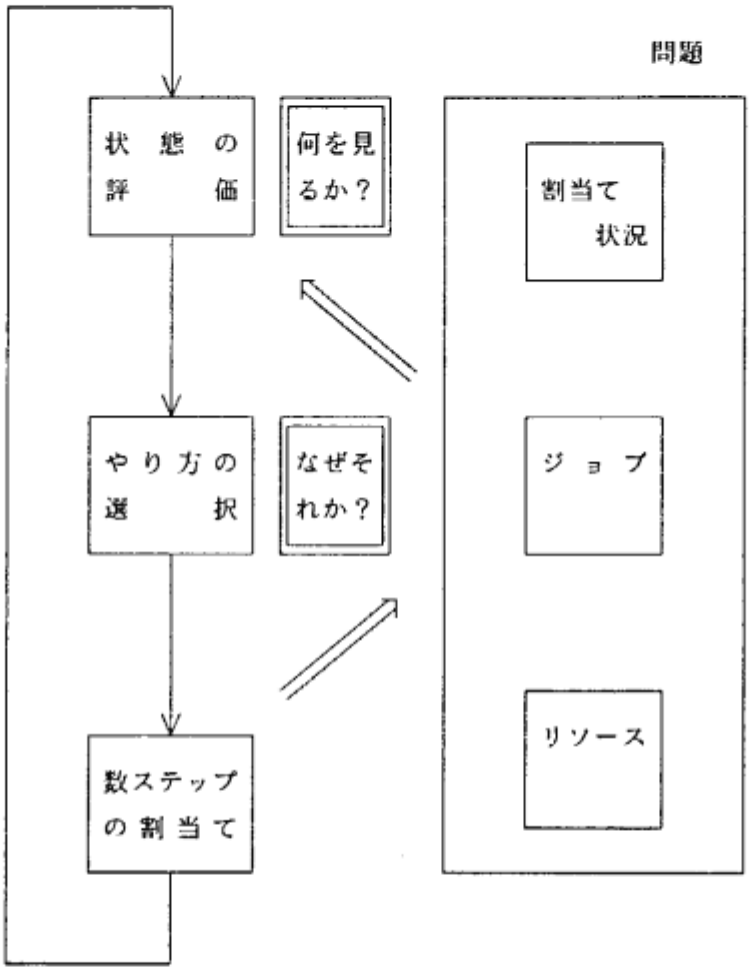
第2-1 図　計算機構成決定支援システム



ジョブ1

ジョブ2

第2-2 図　リソース割当て問題



第2-3 図　リソース割当て問題の知識獲得

2．3　最適化問題と人工知能技術

本節では、合成（計画／設計）型問題として、割当て問題、輸送問題、生産計画などの組合せ的、離散的な組合せ計画問題を考え、ある条件の下での、これらの最適（極値）問題などを扱う組合せ最適化問題を取り上げる。［茨木 79］。

次に、これらの問題を一般的な形（標準形）で定義した上で、その問題の最適解を求める一般的な計算手法（アルゴリズム）の一つとして、分枝限定法を考え、そこでの人工知能技術、知識工学的手法の応用など今後の方向について考える。

2.3.1　最適化問題とは

割当て問題、輸送問題、生産計画など、ある条件の下での最適（極値）問題の理論、解法、適用などに関する問題は、従来から数理計画法という名の下に研究されてきた。この数理計画法のうちでも、最もよく研究されて普及しているものが線形計画法であり、その他、条件や目的の変化に対応して、非線形計画法、整数計画法などがある。

この数理計画法で扱われる最適化問題の一般的定義を、ここでは次のように定める。

［最適化問題ＯＰ］　基礎空間：Ｘ
許容領域：$S \subseteq X$
目標関数：$f : x \in X \rightarrow f(x) \in C$
拘束条件：$x \in S$

最適化問題とは、以上の条件の下で、拘束条件を満たす許容解ｘのうち、目標函数$f(x)$を最小（大）にする最適解を求める問題である。（Ｃとしては、通常実数空間、あるいは整数空間を考える）。Ｘとしてｎ次元実数空間、Ｓとしての有限個の一次不等式を満たす領域（凸多面体）、$f(x)$として一次式Σ（$C_i \times X_i$）を考えれば、よく知られている線形計画問題となり、Ｘ及びＳをｎ次元整数ベクトルの集合とその部分集合に限定すれば整数計画法の問題となる。まはた、Ｓとして上記凸多面体とｎ次元０－１ベクトル集合との共通集合を考えれば、これもよく知られた０－１整数計画問題となる。

さて、ここで問題としているのは、人工知能あるいは知識工学のような非数値処理、記号処理などが中心となる分野であり、線形計画法、非線形計画法ではこのような問題は、一般的には扱わない。このような、どちらかといえば離散的、組合せ的な性質を持つ組合せ計画問題を対象としているのは整数計画法であるが、全ての組合せ最適（計画）問題を整数計画法で解くことは困難であろう。

前期の最適化問題ＯＰをもとに、組合せ最適化（計画）問題を次のように定義する。

［組合せ最適化問題ＣＯＰ］　基礎空間：空でない有限集合Ｘ
許容領域：Ｘの要素を有限個連接した要素からなる加算無限集合X^*の部分集合Ｓ
目標関数：ｆ：Ｓ→Ｃ

Ｘは具体的には、ある個々の“決定”（部分解）の集合“意味し、X^*はそれから生成される決定の系列すなわち計画を意味する。Ｓにより制限された許容解ｘは目標値ｃ(x)（通常、実数値）を持ち、これを最小にする最適解を求めることになる(-c(x)を考えれば最大化問題となる）。

この種の問題の具体的な例としては次のようなものがある。

（１）グラフネットワークに関する問題

最短経路問題、行商人問題、最大マッチング問題、節点（枝）カバー問題、最大フロー問題、最小費用フロー問題、彩色数問題、最大クリーク数問題、輸送問題、通信ネットワーク問題、最小木問題、スタイナー木問題、など。

（２）スケジューリング問題

何台かの機械上で、与えられたｎ個の仕事の処理順序を目標関数（例えば完了時刻）を最小にするように定める、ジョブショップ問題、フローショップ問題、ラインバランシング問題、並列プロセッサスケジューリング問題、など。

（３）最適選択・割当て・配置等に関する問題

集合カバー問題、集合分割問題、集合パッキング問題、ビンパッキング問題、２次割当問題、ナップザック問題、最適切断問題、取替問題、工場設置問題工場レイアウト問題、など。

（４）コンピュータの論理設計・符号理論等に関する問題

最小論理回路問題、順序回路の最小化問題、ＩＣチップのレイアウト・敗戦問題、最良符号の設計問題、など。

（５）その他

最適探索に関する問題、在庫管理に関する幾つかの問題、信頼性理論に関する幾つかの問題、等々。

ここでは、これら個々の問題の説明は省略し、以下では前記組合せ最適化問題ＣＯＰについて考えることにする。

2.3.2 問題の困難さと一般的解法

組合せ最適化問題の多くは有限の問題（Sが有限）であるから、基本的には全ての許容解を列挙し、その中から目標関数fを最小にするものを選べば最適解が得られるが、少し大きな問題を考えると許容解の個数がかなり大きくなる。しかし、一方では、相当大規模な問題でも解けるものもあり、それらの問題では、それぞれの問題のもつ特殊な構造を利用した、列挙法によらない計算法を用いているのが特徴である。

このような組合せ最適化問題ＣＯＰの、比較的広い範囲に適用可能な（その意味で、抽象レベルの高い）一般的なアルゴリズムとして、人工知能、ＯＲ等の分野で用いられてきた分枝限定法がある［茨木 79］。

分枝限定法とは一種の列挙法であり、その基本なる考え方は次のようなものである。X^* の全ての要素xを順に列挙していくと、無限個の節点を持つ木Tが得られる。分枝限定法ではTの根節点から始め、既に生成されているある節点xを一つ選び、その子節点を生成するという操作を繰返し、Tの節点を上から列挙していく。生成された各節点に対して、解の初期部分がxに限定されたという条件の下で、いわゆる部分問題をテストしこれを解くことになる。各部分問題（これを、cop(x)と書く）においては次のような可能性が考えられる。

(1) テストの結果、部分問題cop(x)が解け、その最適解とコストf(x)が求まる。
（ＣＯＰの最適解とは限らない）。
(2) 既に生成されているある部分問題cop(y)に対して、$f(x) \geq f(y)$が成り立つ。
(3) これまでに既にとかれたcop(y)のコストf(y)の最小値をzとするとき、$f(x) \geq z$ が成り立つ。
(4)cop(x) に関して何の結論も得られない。

(1) の場合には、cop(x)は解かれたのだから、cop(x)の部分問題を考慮する必要はない。また、(2),(3) の場合、$s(x) = \{xy \mid xy \in S,\ y \in X^*\}$ とすると、S－s(x)の中に最適解が存在することが結論できるから、cop(x)とそれから生成される部分問題を考慮しなくても、ＣＯＰの最適解を全て見失うことはない。

従って、(1) ～(3) が結論できるなら、cop(x)を終端してもよい。

(4) の場合には、cop(x)をさらに $|X|$ 個の部分問題cop(xy)$(y \in X)$ に分解し、更に計算を進めて行くことにな。。以上の手続きを続けて行けば、生成された部分の全ての葉節点が終端され手続きが終了する場合がある。このとき、zが最適解のコストに等しくなる。

このように、分岐限定法は各cop(x)をその部分問題cop(xy) に分解する分岐操作と、生成された部分問題を(1) ～(3) の性質を利用して終端する限定操作を基本とする手続きである。

実際にこの手続きを実行するには、(1) ～(3) の判定をどのように行うのかとか、部分問題の選択、終了条件の判定（X^* は無限集合）等の問題がある。上記の分岐限定法を用いるにせよ、従来からの各種計画法を用いるにせよ、設計の全プロセスを自動化、あるいはコンピュータ化することは不可能である。設計プロセスを数量的に表現することは常に可能なわけではなく、仮に数量的に表現することが可能であっても、そのモデルには自ずから限界がある。例えば、最も良く知られ普及している線形計画法にしても、次のような限界がある。［国沢　59］。

(1) 線形性の仮定、

(2) 方程式、未知数の等質性、

(3) 静的理論（動的変化の処理が困難）、等。

従って、実際には各種ヒューリスティックを用いた以下のようなアルゴリスムを用いて、その近似解を求めているのが実情である［茨木　86］。

(1) 欲張り法（貪欲法）、

(2) ランダム法（モンテカルロ法）、

(3) 緩和法、

(4) 分割法、

(5) 部分列挙法、

(6) 反復改善法、

(7)Simulated Annealing法。

2.3.3　人工知能技術の応用

人工知能技術、知識工学的手法の合成（計画／設計）型問題への応用は、解析（診断／解釈）型問題への応用より一般的に難しいとされている。その原因としては色々な要因が考えられるが、合成型問題の方法論が不明確である（確立されていない）ことも原因の一つであろう。従って、合成型問題への人工知能技術の応用は、解析型問題への応用より遅れており、現在始まったばかりといってよいであろう。

ここで、以下のような注意しておかなければならない点がある［大須賀　87］。

これまで我々が扱ってきたあらゆる分野の問題に対して、それぞれ固有の問題解決方法、理論が開発されてきた。例えば、計画問題に対しては、既に述べた各種計画法やグラフ理論、ＰＥＲＴ図等を用いる方法がある。これらの方法は、それぞれの問題に対してその時点時点で、人々がその能力の範囲内で最も有効なものとして開発してきたものであり、これより強力なものでない限り、他の方法で置き変えられるものではないということである。また、新しい方法が開発されたとしたら、それら従来の方法を包含するものでなければならない。少なくとも、新しいシステムは、これら問題毎に開発されてきた問題解決の方法を利用できるシステムとして統合されていなけれ

ばならない。残念ながら、現行のエキスパートシステムは必ずしもそのようになってはいない。

このような点から、最近の論文として情報処理学会第33回全国大会に発表された、合成型問題に属すると考えられるエキスパートシステム関係の論文を取り上げて眺めて見ると［中松　86］～［金井　86］、合成型問題の持つ特徴、即ち、情報量の多さ、目標の不完全さ、及び曖昧さ、求解コストの多さ等、によって生じる探索空間の大きさが、現状での一番重要な問題となっているようである。

合成型問題に限らず、試行錯誤的問題は一般的に木探索の問題に帰着できることが多く、結局この木探索を効率的に行う、即ち、無駄な枝をなるべく早くカットし、解の存在しそうな枝に的を絞り探索空間を狭めて効率的に探索し解を求めということが重要となる。この木探索型の問題に適用可能な一般的アルゴリズムの一つとして既に述べた分岐限定法がある。現状の合成型エキスパートシステムは、この部分、即ち、分岐限定法でいう(1) ～(3) の過程に、いわゆる“エキスパートのノウハウ”と言われるような各種技法を応用して、なるべく最適に近い知的探索を行うようなシステムであると言うことができよう。この種の技法、あるいは考え方として最近注目されているものを幾つか上げると以下のようなものがある。

(1) Intelligent Backtracking

(2) Dependency Backtracking

(3) Consdtraint Programming

(4) Model-Based Reasoning

(5) Qualitative Reasoning, etc.

さて、今までのシステムで用いられてきたアルゴリズム、あるいは手法は逐次型と考えてようであろう。その一方で、エキスパートシステム・アーキテクチャの流れとして分散処理指向化が考えられ始めている。ところで、上で述べた“無駄な枝をなるべく早くカットし、解の存在しそうな枝に的を絞り探索空間を狭めて効率的に探索する”ということは、PROLOGに代表される論理型言語の並列推論アルゴリズム、即ち、AND-OR木の効率的探索問題とも密接に関係する。従って、今後は合成型問題というよりは、より広く“ＡＩシステムにおける知的木探索型問題”という枠組みで捉えて、その一般的開発手法として、この分野に適した“知的分岐限定法”とでも言うような方法論を確立することが必要であろう。

【参考文献】

[茨木　79]　茨木俊秀、組合せ最適化の理論（電子通信学界、東京、1979）、p220

[茨木　86]　茨木、他、"特集　組合せ最適化" オペレーションズ・リサーチ、vol.31. No.1.pp.10-48(1986).

[国沢　59]　国沢清典　　リニア・プログラミング（日刊工業新聞社、東京、1959）p309.

[中松　86]　中松、他、"エキスパートシステム開発支援ツールＥＸＤ３－基本思想とプロトタイプの構成－"、情報処理学会第33回全国大会、pp.1141-1142.

[井上　86]　井上、他、"黒板モデルに基づくＥＳシェルとその設計問題への応用"、情報処理学会第33回全国大会、pp.1145-1146.

[玉野　86]　玉野、他、"プリント板設計工程管理支援エキスパートシステム"、情報処理学会第33回全国大会、pp.1165-1166.

[吉田　86]　吉田、他、"知識工学的手法を応用した変電所レイアウト・システム（３）－推論手法の特徴－"、情報処理学会第33回全国大会、pp.1167-1168.

[田中　86]　田中、他、"知識工学的手法を応用した変電所レイアウト・システム（４）－レイアウト・システムの機能－"、情報処理学会第33回全国大会、pp.1169-1170.

[都島　86]　都島、他、"知識工学応用・流通業向ワークスケジューリングシステム"、情報処理学会第33回全国大会、p.1189-1190.

[若林　86]　若林、"計算機による時間割編成支援に関する一考察"、情報処理学会第33回全国大会、pp.1243-1244.

[大須賀　86] 大須賀、"エキスパートシステムの現状と課題－次世代システムへ向けて－"、電学論Ｃ，Vol.107-C,No.2,pp.104-109.

[金井　86]　金井、"プランニング・システムPLANにおけるサブ・プラン間相互作用の解決方法"、情報処理学会第33回全国大会、pp.1327-1328.

3．合成型問題における知識獲得の事例

3．1　毛筆文字の美しさに関する知識獲得
― 芸術分野における知識獲得へのアプローチの例 ―

3.1.1　書道における知識の特徴

書道は東洋とくに日本、中国においてポピュラーかつ実用的な芸術活動の一つであり、多くの人にわかり易い対象である。また比較的限定された問題領域であり、データ（文字サンプル）を得ることも容易である。

しかしながら、芸術分野における知識伝承が、大部分徒弟的であるのと同様、書道に関する知識も、書道家である先生から、実技を通して直接経験的に教えられるものであり、したがってその知識を計算機言語で記述蓄積し、計算機によって活用することは不可能なこととみなされてきた。

ところが、毛筆文字の計算機による生成研究の進展によって美しい文字が、直観的に理解しやすいパラメータを調節することにより生成できるようになると、経験的に伝えられ定性的にしか表現し得なかった書道知識を、ある程度計算機言語によっても取扱えるようになり、全く新しい書道研究の方向が展望できるようになった。すなわち、毛筆ワープロとして既に市販されている多くのソフトが採用している、イメージとして文字パターンを記憶する方法ではなく、人が毛筆を用いて文字を置く過程を記憶し、コンピュータによりシミュレートする方法（階層分解合成法）【参考文献１～５】を用いれば、書道知識を計算機上に、記号的数値的に表現する具体的手段が実現された【参考文献６】。

また書道に関する知識源としては、書道書、書家、および書があり、いずれも容易に近づくことができる。したがって、書道は芸術分野における知識を扱う適切なドメインであるといえよう。

3.1.2　書道書による知識

書道書には、長年にわたって獲得されたノウハウの結晶とも言うべき書道の真髄が法として既に記されている【参考文献７，８】。その多くは漢字の部分パターン間のバランスをとり扱ったものであり、したがって階層分解合成法によくなじむと思われる。漢字の構成パターンは図3.1-1 のように分類され、これに従って、得られた９４法のルールにまとめられた書道知識も表3.1-1 のように分類される。これらのルールは、条件とその場の心得という形式で与えられており、プロダクション・ルールとして記述できる。しかしながら、当然のことながら表現の抽象度が高く、記述が困難である。ルールの適用条件としては、いくつかの文字例が与えられるのみであることが

多く、また比較的明確に指示されている場合でも、条件の一部が与えられているのみで、書道家の常識としてデフォルトにされている条件が難物である。ルールは、文字の部分パターンそのものに関する規則と、部分パターン間のバランスに関するものに大別される。９４法のうち２８法が既にプロダクションルールとして記述され知識ベース化されている。その一例として三水法をあげると、

三水法：三水の第三画の長さは、つくりの画数により変化する。画数が多いほど短くすればよい。

と書道家には定性的に与えられている。これを定量化して、

三水法：もし、つくりの画数が（Ｊ１，Ｊ２）の範囲ならば、三水の第３画を（Ｕ１，Ｕ２）の範囲の長さに収める。

となる。実際には、Prologによって図3.1-2 のように記述されている【参考文献９～１１，１４】。

一方、条件が字例によってのみ与えられ、抽象度の高い表現に終始している法はルールとしての記述が難しい。その例を次に示す。

譲横法：字例のような文字に対しては、文字の勢いを荷う一本の横画を伸ばし、他の第画は抑えぎみに書く。

このようなルールは２３法ある。

3.1.3 書家から得る知識

書道知識は書道書から得られるもののみでは不十分であるし、また既に書道書にある知識であっても、その洗練化には、書家の直接の協力が不可欠である。このためには、書家にわかり易いインタフェースをもつ、使いやすいシステムを提供しなければならない。このため図3.1-3 のような毛筆文字デザインシステムを作成し、このシステムに知識獲得支援機能をもたせる試みを行っている。

本システムを書家に使ってもらい、数多くのサンプル文字を美しく修正する過程を記録しておく。次に記録されたデータをもとに、分類し統計処理を行うことによって新しい知識を抽出し、知識ベースに蓄える。本システムが有用であるためには画面上でマウス等により、文字の形を自由に操作できるユーザ・インタフェースがもっとも重要な鍵である。

これらの支援機能の実現が可能となったのは、階層分解合成法によって美しい毛筆文字の記述が、少数のパラメータ・データでできるようになったこと、すなわち極めて直観的、効率的な、毛筆文字の表現手段を工夫できたことと、これらのデータを柔軟に取り扱う階層的データ構造をもつパラメータデータベースを考案したことにある【参考文献６】。

3.1.4 書から得る知識

文書の美しさは、個々の文字の美しさもさることながら、文字相互の大きさ、太さ、字配り、字形などのバランスによるところが大きい。人間は文字を並べて書くときに、美的感覚すなわちバランス感覚によって文字の大きさを調整していると考えられる。

書のサンプルをもとに実際に得られた知識の一例として、文字の感覚的大きさのバランスをとりあげる。書道書によるルールが大部分字種または部首のごとき文字カテゴリーを条件とするものであったのに対し、文字サンプルの測定値を条件とする、カテゴリーを越えた一般的計測量に基づくルールの例である。計測可能な大きさに関する物理量と、感覚として感じる大きさが一致しないことは、例えば書道書には、次のように記されている【参考文献７】。

書法に曰く、大きく見える字は小さく、小さく見える字は大きくすると自然にちょうどよい大きさになる。たとえば、"日"と"國"の字を同じ大きさに書いてはならない。吉村らは、"広がり量"を次のように定義し、感覚的大きさを表すファクターとしている【参考文献１３】。

$$\sigma 1 = \sqrt{\frac{\int\int \{ (x - a_{x1})^2 + (y - a_{y1})^2 \} \cdot f_1 (x, y)\, dx\, dy}{\int\int f_1 (x, y)\, dx\, dy}} \quad (1)$$

但し、$f_1 (x, y) = 1$（文字の黒領域）、0（それ以外）

(a_{x1}, a_{y1}) は文字の重心

しかしながら、書サンプルを用いた我々の追試実験では、満足しうる結果を与えなかった。そこで文字の勢力範囲を考え、図3.1-4 破線のように補正した広がり量σを用いた。

さらに、複雑度を次のように定義する。

$$C = \frac{\Sigma S_n}{\max (dx, dy)} \quad (2)$$

S_n ：n番目のストロークの長さ

$dx, dy : x_{max} - x_{min}, y_{max} - y_{min}$

これらの測定量を用いれば、多くの文字の大きさのバランスをとることができた。図3.1-5 に示されるようにσとCの関係がほぼ３次曲線で近似できる。この近似補正により文字の大きさを正規化した例を図3.1-6 に示す【参考文献１２，１５】。

書法の抽象的定性的知識を、書サンプルをもとに定量化した知識の例である。

3.1.5 今後の展望と課題

システムは現在、文字デザイン・エキスパート・システムとして、そのプロトタイプがほぼできている。本システムを実際に書家が自在に使いこなし得る程度にインタフェースを改善できれば、有効な毛筆日本文字に関する知識獲得支援システムになり得ると期待できる。平仮名も、同じ階層分解合成法によって生成できるが、かな文字に対する美しさ、バランスなどは、まだこれからの課題である。

【参考文献】

〔張　84〕　張、真田、手塚：“漢字楷書毛筆字体の計算機による生成”、電子通信学会論文誌(D)J67-D.5.pp.599-606(1984

〔張　85〕　張、真田、手塚：“階層分解合成法による隷書体漢字の生成”、電子通信学会論文誌(D).J68-D.8pp.148901496(1985)

〔張　86a〕　張、真田、手塚：“計算機による様々な書体生成に適合する筆触パターンの提案”、電子通信学会論文誌(D).J69-D.6(1986)

〔張　86b〕　張、真田、手塚：“階層分解合成法による毛筆書き平仮名の生成”、電子通信学会総合全国大会（昭和６１年）

〔内尾　86〕　内尾、張、真田、手塚：“漢字楷書毛筆自体生成パラメータの半自動決定システム”、電子通信学会論文誌(D).J69-D.6(1986)

〔北川　86〕　北川、井上、真田、手塚：“計算機授用毛筆自体漢字設計システムについての一考察”、電子通信学会パターン認識研究会資料、PRL86-61(1986)

〔鐘　63〕　鐘：“欧陽結体三十六法全釈”、人民美術出版社（1963）

〔藤原　83〕　藤原：“図解楷書事典”、省心書房（1983）

〔曽　85〕　曽、井上、真田、手塚：“ルールを用いた毛筆文字評価法の一試案”、電子通信学会情報・システム部門全国大会（1985-11 ）

〔曽　86a〕　曽、井上、真田、手塚：“知識を用いる毛筆字体評価システム”、情報処理学会第33回（昭和61年後期）全国大会（1986）

〔曽　86b〕　曽、井上、真田、手塚：“毛筆字体評価知識の洗練化についての検討”、電子通信学会技術報告、A186-25(1986-10)

〔樋口　87a〕　樋口、内尾、北橋、真田、手塚：“文字の感覚的大きさの定量化”、電子情報通信学会総合全国大会（昭和62年）

〔吉村　70〕　吉村、飯島：“文字字体の一設計法”、情報処理、11.3.pp.135-143(1970)

〔曽　87b〕　曽：“毛筆漢字字形評価エキスパートシステムに関する研究”、大坂大学工学研究科通信工学専攻修士学位論文（1987）

〔樋口　87b〕　樋口："文字の感覚的大きさの定量化とその正規化手法に関する研究"、大阪大学工学部通信工学科卒業論文（1987）

表3.1-1　書道規則分類表

	パターン間の位置関係	パターンの相対的な大きさの関係			
		文字構造の一般特徴に関する規則			
left-right型	1 譲左法(叩、幼) 2 譲右法(頒、蛾) 3 下平法(朝、知) 4 上平法(明、唯) 5 分疆法(顕、難)	6 左占地歩法(敬、剣) 10 減勾法(杯、村)	7 右占地歩法(鍵、鼓) 11 屈勾法(鳩、鶏)	8 応付法(誘、非) 12 右垂法(邪、邦)	9 左縮法(斜、竹)
up－down型		1 天覆法(宇、宇) 5 下占地歩法(装、爲) 9 中竪法(卓、辛) 13 蓋横法(害、安) 17 円　法(国、隣)	2 地載法(至、孟) 6 蓋下法(合、命) 10 堆　法(森、品) 14 横波法(是、足) 18 直長縮短(矣、異)	3 二段法(叢、留) 7 俯仰勾脚法(冠、宅) 11 減勾法(禁、森) 15 横戈法(思、息) 19 勾挙法(芳、菊)	4 上占地歩法(誓、響) 8 上小法(算、泉) 12 減捺法(条、爻) 16 排点法(赤、然)
over-below型		1 上大下小法(道、直)	2 伸勾法(勉、旭)		
rectangle型		1 平四角法(目、国) 5 遮蔽法(匹、医) 9 促波法(尺、文) 13 上短下長法(正、王) 17 横短直長法(才、斗)	2 短方法(由、田) 6 回両肩法(南、而) 10 促戈法(武、或) 14 長　法(自、月) 18 地載法(且、豈)	3 長方法(周、用) 7 懸針法(申、巾) 11 屈脚法(馬、鳥) 15 横長直短法(十、一上) 19 均横法(三、壷)	4 促脚法(鳳、風) 8 縮直法(甲、平) 12 横脚法(毛、元) 16 短肥長痩法(白、四) 20 均直法(川、州)
left-middle-right型		1 左右占地歩法(鍬、衍)	2 三勾法(御、謝)	3 中占地歩法(街、樹)	
below-over型II		1 勾裹法(句、勿)	2 促脚法(気、飛)		
up-middle-down型		1 三停法(草、意)	2 上下占地歩法(繋、賢)	3 中占地歩法(幸、華)	
below-over型I		1 遮蓋法(反、皮) 5 横撇法(考、省)	2 促挙法(庭、居)	3 直短横長法(左、在)	4 直長横短法(右、灰)
		特定の部分パターンに関する規則			
left－right型		1 散水法(湖、滿) 5 力旁法(勤、功) 9 联挙法(須、形) 13 禾偏法(和、私) 17 頁旁法(頂、顧)	2 イ偏法(作、侍) 6 阝　法(陽、部) 10 巾偏法(帆、幅) 14 米偏法(精、粉) 18 金偏法(鉄、錯)	3 木偏法(柱、横) 7 欠旁法(次、欣) 11 攵旁法(攻、政) 15 見旁法(規、親) 19 鳥旁法(鳩、鴨)	4 彳偏法(行、往) 8 日偏法(明、時) 12 卩旁法(印、即) 16 貝偏法(則、財) 20 糸偏法(統、細)
up－down型		1 上羊法(羨、美) 5 联挙法(参、公)	2 上羽法(習、翌) 6 下夕法(支、多)	3 上禾法(香、季) 7 上山法(岸、出)	4 下羽法(翁、翡)
over－below型		1 下走法(超、起)			

総計：９４個

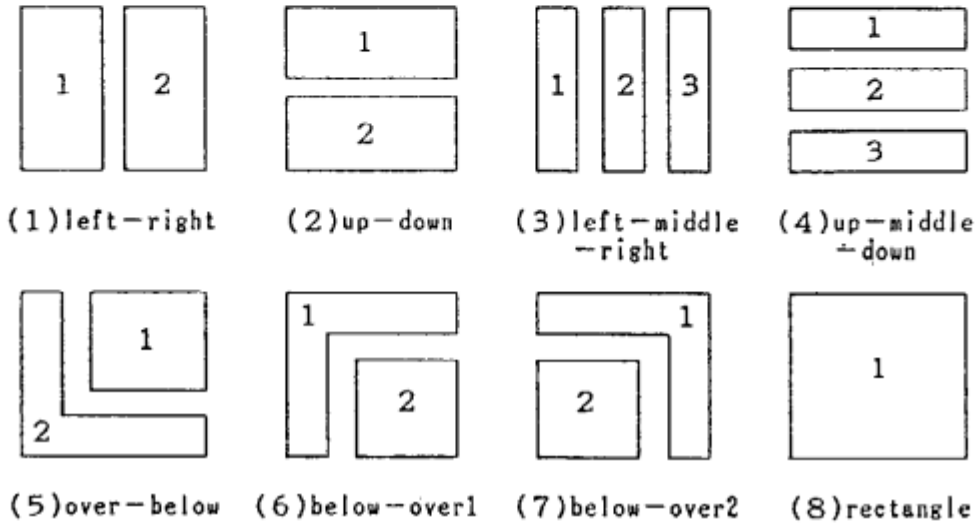


図3.1-1　漢字構成パターン分類

```
action_r(H,8,X,Z,E):-
        not(member(X,E)),
        list_l(Z,Z1),re_list_n(1,Z1,ZZ),
        ZZ=[1,1,18],!,action_e(H,8,X,Z,E).
action_e(H,8,X,Z,E):-
        list_l(Z,Z1),listn(Z1,3,Z33),
        list_r(Z,Z2),length(Z2,N2),
        listn(Z33,4,Z4),
        ( N2=<3,!,
          ( Z4<133,list_elem(Z33,4,133,ZZ);
            Z4>136,list_elem(Z33,4,136,ZZ);
           ZZ=Z33 ),list_elem(Z1,3,ZZ,H1);
         N2>3,N2=<6,!,
          ( Z4<115,list_elem(Z33,4,115,ZZ);
            Z4>118,list_elem(Z33,4,118,ZZ);
            ZZ=Z33 ),list_elem(Z1,3,ZZ,H1);
         N2>6,N2=<11,!,
          ( Z4<108,list_elem(Z33,4,108,ZZ);
            Z4>110,list_elem(Z33,4,110,ZZ);
            ZZ=Z33 ),list_elem(Z1,3,ZZ,H1);
         N2>11,N2=<16,!,
          ( Z4<98,list_elem(Z33,4,98,ZZ);
            Z4>100,list_elem(Z33,4,100,ZZ);
            ZZ=Z33 ),list_elem(Z1,3,ZZ,H1) ),
       H=[H1,Z2],
       write('三水偏の下点の長さを変更したいが,'),
       write('ご同意でしょうか(y/n)'),
       nl,nl,read(P),
       (  P=n,!,nl,new_rule(8,X,Z,E);
              !,result_print(8,H,X,E)  ).
```

図3.1-2　Ｐｒｏｌｏｇによる三水法

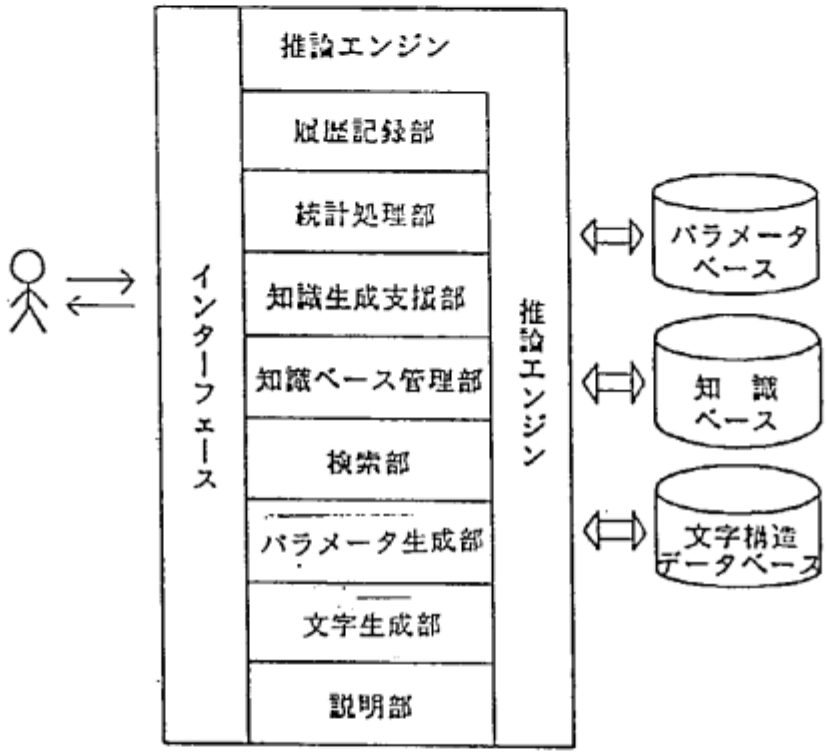


図3.1-3　文字デザイン・エキスパート・システム構成図

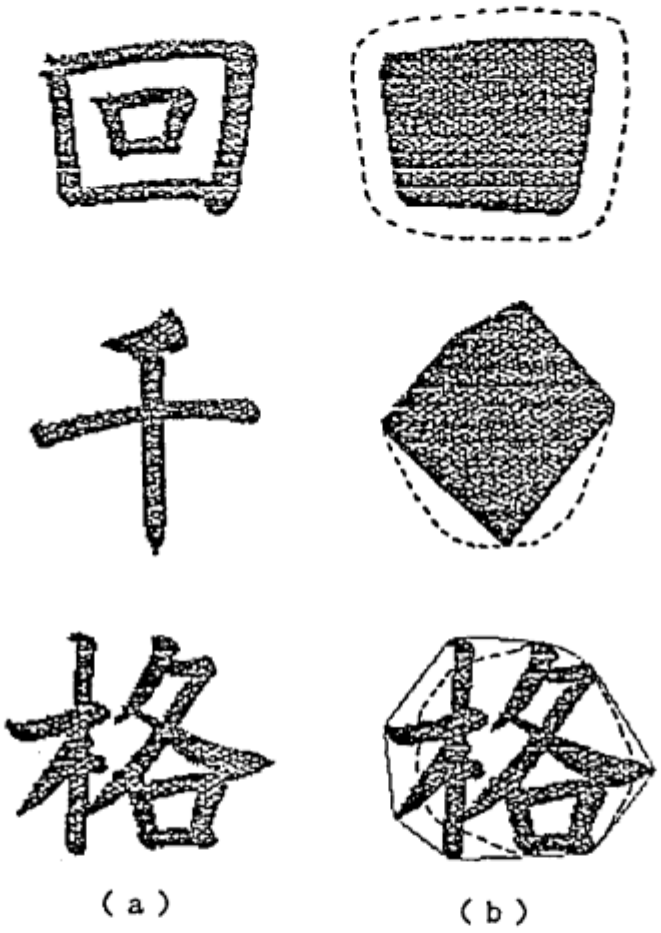


図3.1-4　文字の勢力範囲

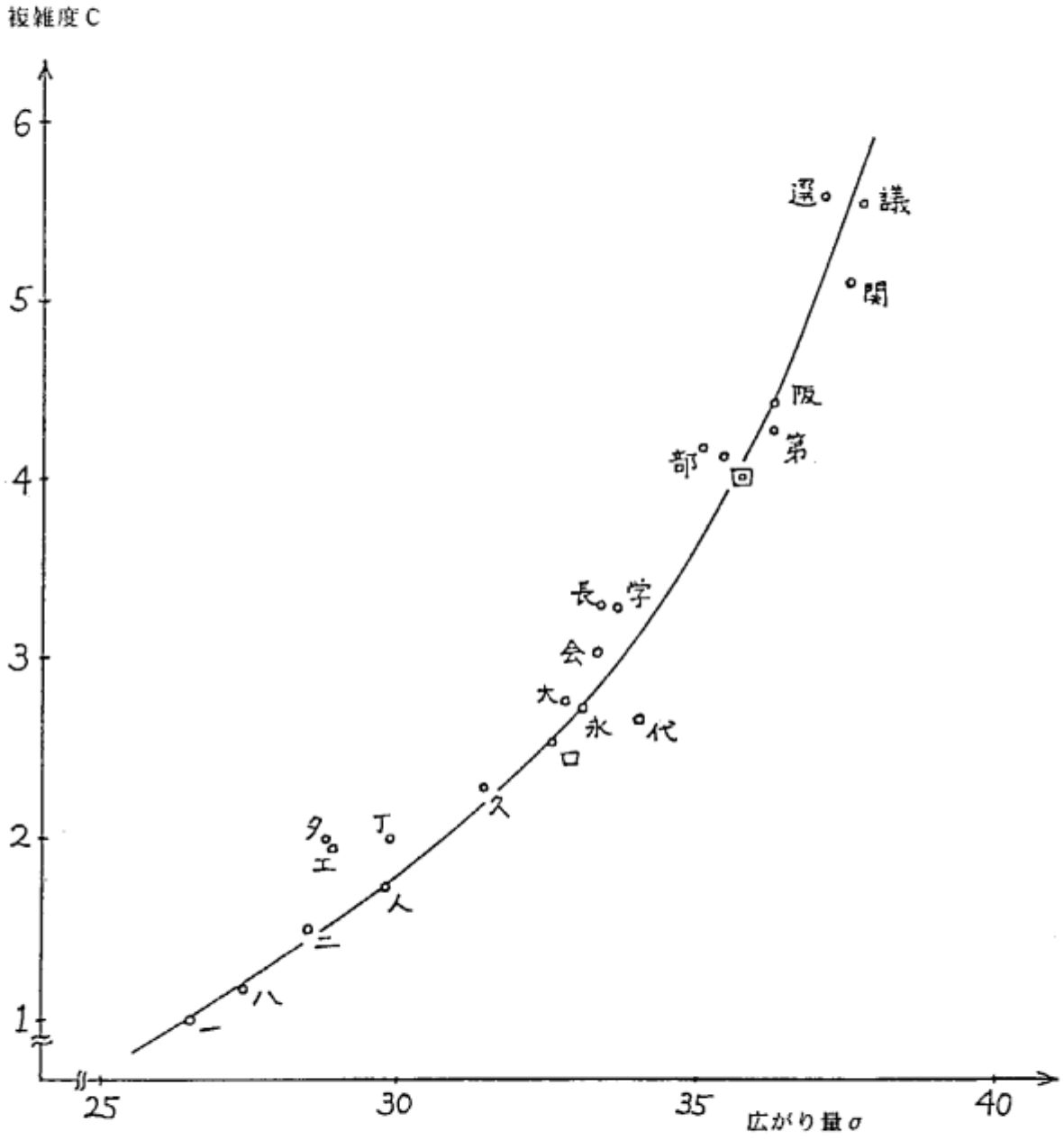
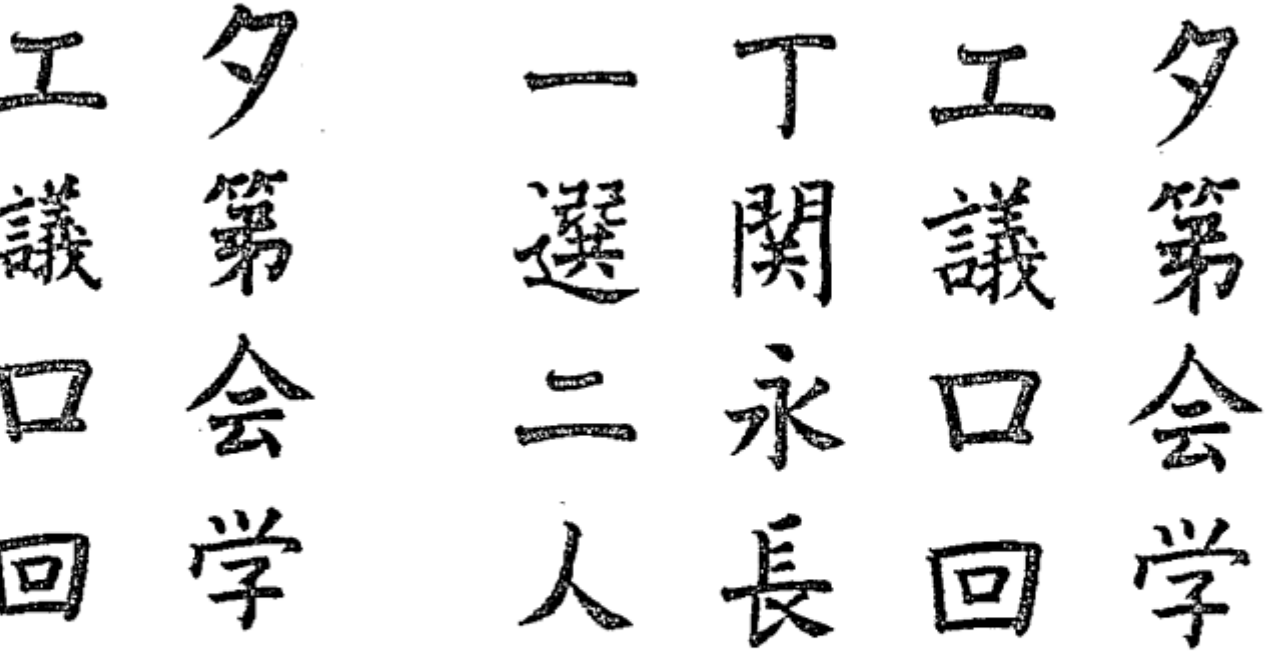


図3.1-5　σとCの関係

（a）広がり量のみの場合



（b）本手法による場合

図3.1-6　正規化結果の比較

3．2　事務処理システムの開発

3.2.1　大規模事務処理システムの現状と特性

大企業の情報処理部門はこれまで永年にわたって業務の機械化を推進してきている。そして、最近は、主要業務のシステム化は達成し、システムの統合化と個別の支援業務のシステム化の時代へと移行している。現在の大規模事務処理システムの構造は、図3.2-1 に示す通りである。すなわち、ハードウェアを含むＥＤＰシステム、個別システム、ワークステーションに情報収集、情報管理、情報処理、情報伝達の４つの基本構想を持たせ、それらを統合システム化している。

大規模事務処理システムの特性は以下の５点にまとめられる。

・保有プログラム数が多く、しかも寿命が長いために、その保守作業に多くの労力が必要となる。
・新規システムはほとんどが全社規模のデータベースに関連するうえに、要求が高度であるために、既存システムとの整合性に多くの労力が必要とされる。
・いわゆるＯＡ化の影響でエンド・ユーザが対話的に直接データを操作できるようなシステムの必要性が高まってきている。
・従来からなされていたシステム開発の標準化の枠組みにとらわれるあまり、必要以上に開発の手続きが複雑化・硬直化する傾向にある。
・特にＯＡシステムを対象とする場合などは、機器の高度化、対象範囲の全組織的な拡大、ホスト・コンピュータとの連携の必要性等の理由によって、常に新技術の導入を要求される。

また、最近は、情報処理部門を取り巻く環境、すなわち、システム化の対象業務とシステムを実現するツールとが大きく変化しており、情報処理部門は、エンドユーザ主導型の非定型業務を対象とするようなシステムにも対応していかなければならない。ところが、このようなシステムではエンドユーザの要求ををあらかじめ正確に把握することは難しい。そのため、今後の柔軟性のある情報システムを効率的に開発・保守・運用していくためには、「情報管理の効率化」の概念をさらに進めて、情報の収集・補完・利用のフェーズを明確に分離したデータ管理を基礎とするシステム構成にすることが重要である。このような体系のもとでは情報処理部門の役割は、さまざまなデータベースとそれの表わすビジネス・モデルの管理が中心となる［坂内１９８３］。

3.2.2 事務処理システムの開発プロセス

事務処理システムの開発形態には、従来から行われてきたウォーターフォール型の開発形態と、人工知能システム開発に起源をもつラピッドプロトタイピング・探査型プログラミング、ならびに、ソフトウェア部品を開発に利用する再利用手法などがある。

ウォーターフォール型の開発は、ソフトウェアのライフサイクルをシステム要求・仕様化・概念設計・詳細設計・コーディング・単体テスト・連結テスト・運用・保守などのフェーズにわける。そして、各段階ごとに作業が完結するように、トップダウンに作業を詳細化していく。この方式によると、特に大規模なシステム開発においては、開発の比較的早い段階で利用者の要求が確定するので、後のフェーズでの修正が生じにくく、ライフサイクルコストの低減に役立つとされている。そのため、事務処理システムの中心となる統合化システムにはウォーターフォール型の開発が適切である。また、大企業の情報処理部門のシステム開発は通常この方式にしたがっており、各社ごとに独自の開発標準を設定していることが多い。反面、ウォーターフォール型の開発においては、各フェーズごとに作成される文書の量が膨大になり、開発中で生ずる利用者要求の変化に対応することが難しいという欠点がある。

それに対し、ラピッドプロトタイピング・探査型プログラミングの手法はソフトウェアが絶えず変化することを前提とした開発手法といえる。この方法ではシステムのスクラップ・アンド・ビルドを頻繁に繰り返しながら、開発と利用者による評価とを交互に行い、システムを徐々に成長させていく。この手法は、各プロトタイプの性格をどのように位置付けるかによって、次の４種類に分類することができる。第１はプロトタイプを運用システムの模型と考え、その評価が終ればプロトタイプを捨てる方式である。第２は、プロトタイプを運用システムのシミュレーションに用いる方式で、利用者インタフェースの改善などにしばしば使われる。第３は、プロトタイプを実働モデルと考え、その機能を充実させることで連続的に運用システムの完成をはかる方式である。４は、人工知能のプログラムのように使用も実現手法も不明確なシステムの開発において、プロトタイプを研究開発用の道具として役立てる方式である。ラピッドプロトタイピング・探査型プログラミングという用語はほとんど同じ意味に使われることが多い。その差異はラピッドプログラミングは実現手法の明確化をねらうことにある。このような手法は、利用者指向の支援システムの開発に有効であるとされている。

再利用手法は、事務処理業務特有のタスクを汎用的な部品の形で整理・収集しておき、それを組合せることで開発の効率化をはかる方法である。ウォーターフォール型の開発にも、プロトタイピングの手法にも適用することができるが、環境の変化に耐えうるようなソフトウェア部品を蓄積することが難しいという欠点がある。

3.2.3 開発に必要な知識

事務処理システムの開発に必要な知識は、対象業務に関する知識、ビジネス・モデルに関する知識、プログラミング・開発方法論などに関する知識の３種類にわけることができる。

対象業務の知識はいわばワールドに関する知識であり、利用者のシステム要求を収集・整理・理解していく過程で獲得される。そして、この知識はシステムの完成の後は少なくとも部分的にはプログラムやＪＣＬに埋め込まれてしまう。このような種類の知識をある程度抽象化して得られるのがビジネス・モデルに関する知識である。これは、システム利用者の要求仕様を誤りなく理解するための知識であり、開発者の頭の中で整理された後に、通常は、データベースのスキーマや再利用部品などの形式で定式化される。ソフトウェア工学における要求仕様化技法はこのような種類の知識の獲得に有効であり、これと知識情報処理技術における知識獲得手法とを組合せることによってソフトウェア開発の効率化に役立つ可能性が高い。

プログラミングに関する知識は、使用するアルゴリズムやユーティリティプログラム、計算機の性能などに関する知識などからなり、マニュアルや開発者の頭の中に存在している。これらの知識は（大量ではあるが）明文化さえできれば、計算機上に蓄積することは比較的容易である。したがって、（浅い知識を扱う）エキスパート・システム技術の適用可能性が高いと考えられる。また、開発方法論に関する知識も、マニュアルや開発者の頭の中、あるいは、帳票の形で存在している。しかし、これらの知識は設計のノウハウに関する深い知識であるため、たとえ明文化できてもその使い型・制御方法に関する知識まで獲得することができなければ、計算機上に保持してもあまり役立たない。

3.2.4 知識情報処理技術の応用可能性

実際の事務処理システムは、対象業務からビジネス・モデル、コンピュータシステム、利用者インタフェースへというように、何回かの変換を経てようやく実現する。したがって、これを利用者のもつメンタル・モデルに一致させることは非常に困難である（図3.2-2 ）。知識情報処理技術を事務処理システムの開発に適用する最終的な目標はこのギャップを埋めることにあると考えられる。

ただし、現在の知識情報処理技術のレベルでは、前節に述べた知識をすべて取り扱うことは不可能である。さしあたって、有望なのはビジネス・モデルやプログラミングに関する知識（の一部）を計算機上で表現することであろう。このような知識は対象が比較的明確なのでオブジェクト指向のパラダイムで定式化できる可能性が強い。この形で定式化された知識は、オブジェクト指向のパラダイムのもつモジュール性から、環境の変化に対応しやすいので、ウォーターフォール型開発におけるビジネス・

モデルの改良・詳細化に役立てることができよう。また、これらを特定の視点から見直すことで利用者向けの支援システムの実現も効率化できるようになると考えられる。

ビジネス・モデル、より具体的にはデータと処理手続きの意味を計算機に与えることができれば、それによって情報システムの開発をある程度まで自動化することが可能となる。そうすれば、強力なプログラミング環境とこの自動化システムを利用することによって事務処理システムにおけるソフトウェア問題は解決されると考えられる。

【参考文献】

〔坂内１９８３〕坂内広蔵、鈴木道雄：データ管理を基礎とした業務処理システムの構築。電力中央研究所報告・研究報告：５８２０１８，昭和５８年８月。

〔坂内１９８３〕坂内広蔵、寺野隆雄：大規模事務処理システムにおけるプロトタイピング、情報処理学会要求定義とプロトタイプシンポジウム、（１９８６年４月）

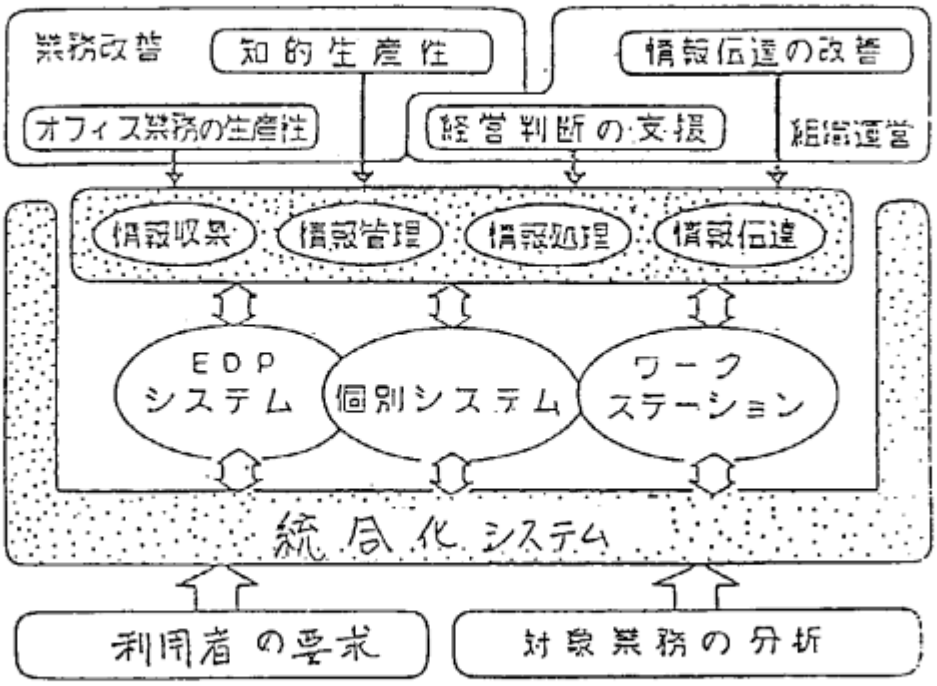


図3.2-1　事務処理システムの構造

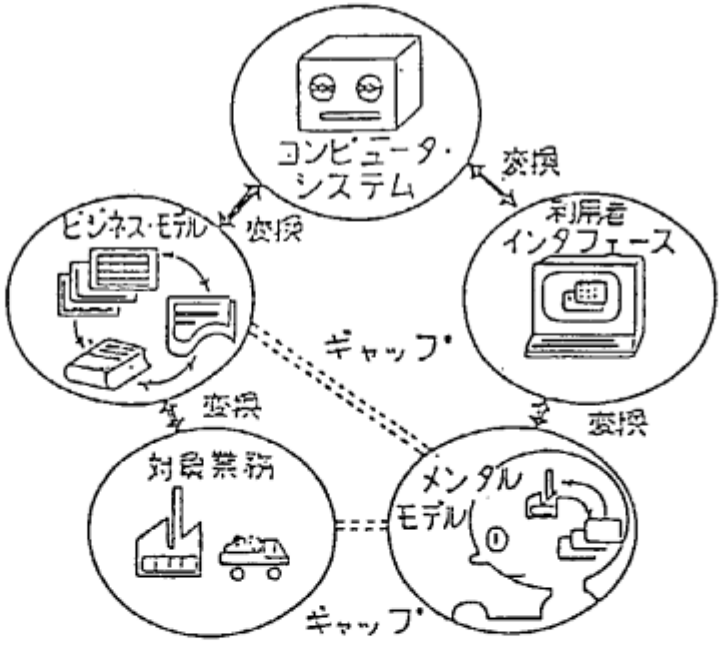


図3.2-2　ソフトウェア開発におけるモデル化とギャップ

3．3　ＬＳＩ設計における知識獲得支援

3.3.1　はじめに

ＬＳＩ設計における知識獲得支援システムの例としてVILLA(VLSI Design Intelligent Learning Apprentice System)〔渡辺　86〕、〔渡辺　87〕の紹介を以下に行う。　VILLA はExplanation-besed Learningと呼ばれる解析的学習方式と述語論理をベースにしたTMS を用いて専門家知識獲得における負担が出来るだけ少なくなるように専門家からVLSI設計のノウハウを演繹的に獲得するシステムである。VLSI設計エキスパートシステムは近年数多く開発が行われているが、システムの構築においては下記の問題点〔Politakis 84〕がある。

1. VLSI 設計を実現するためには、大規模な知識ベースが必要である。
2.一人ではなく複数の設計者のノウハウが必要である。
3. VLSI 技術の急速な変化に追随しなければならない。
4.設計者の本来の仕事を中断させること無く、設計ノウハウを知識ベースに組み込めることが望まれる。

従来の知識獲得支援システムでとられてきた獲得支援方法には、TEIRESIAS 等でみられる既存の知識ベースと獲得された新しい知識の整合性維持の支援、ETS やMORE等でみられる心理学やモデル論をベースにした専門知識の抽出・リファインの支援がある。各システムは主に一人の専門家が、経験則を抽出してリファインすることを支援したり、獲得された知識ベースの無矛盾性がないように成長させることを支援したりする。このため構築される知識ベースも相対的に小さい。ところが上記の問題点を解決するには膨大な量の知識を複数の専門家から抽出して知識ベース化することと大量の知識を既存の知識ベースに矛盾がないように統合することが特に重要となるのであるが、従来の支援方法をとるとすれば専門家による知識ベース構築は現状においては人数的にも時間的にも不可能である。このような訳でExplanation-based Learningを用いて人間の問題解決プロセスをシステムがモニターし、かつ演繹的に解析して必要な知識を一般化して抽出することとTMS による知識ベースの管理することをVILLA で支援することにした。VILLA のVLSI設計知識の獲得における主たる目的は

・設計者の仕事を中断することなく、そのノウハウを修得すること
・知識レベルに応じた知識獲得支援を行うこと

である。そしてVILLA の目標としてはExplanation-based LearningとTMS を用いて実用的VLSI設計知識獲得システムを実現することである。下記の図3.3.1-1 にVILLA の位置付けを示す。

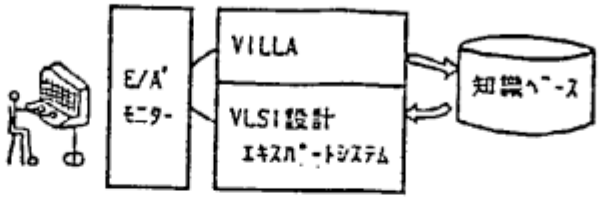


図3.3.1-1　VILLA の位置付け

E/A モニターとはエキスパートシステムがある時には専門家（Expert）としての役割を果たし、又、ある時は見習い（Apprentice）としての役割を果たすためのモニターである。上図のように、VILLA は "より賢い見習い" として振る舞い、設計者は特にVILLA を意識すること無く仕事を進めることができ、システムの知識ベースにない設計知識が獲得されていくことになる。

3.3.2　VILLA の知識獲得のアプローチ

VILLA の知識獲得方法の基本的な考え方は下記のようになる。

1.各知識レベルに応じて最適な知識獲得アプローチをとる。

2.設計者が遂行している仕事の中断を最小限に抑えて、知識獲得を行う。

(1) 各知識レベルに応じた最適な知識獲得アプローチ

獲得される設計知識は設計オブジェクト、モジュール詳細化ルール、ヒューリスティックルールの３種類に分けられる。設計オブジェクトは設計の対象となる回路（モジュール）のことである。モジュール詳細化ルールは各設計オブジェクトからモジュールを作成するルールのことである。ヒューリステックルールは専門家が通常の設計作業のさいに効率的に設計を行うために用いるルールのことである。

設計オブジェクトの獲得は不完全なモジュールライブラリーの動作仕様と各種制約条件が与えられた時にこれらを満足する動作仕様を持つ設計オブジェクトを求めることになる。即ち、不完全なモジュールライブラリーの動作仕様と各種制約条件から完全な動作仕様を導出することになる。この導出の作業は基本的に構築しようとするモジュールの表現モデルに従って各パラメータを対話的に埋めていくことになる。この時、パラメータの値又は制約条件のなかには他のパラメータの伝播により決定される場合があり、パラメータの探索空間の無矛盾性の保持を可能にするTMS を利用した支援が有効である。

モジュール詳細化ルールを獲得する方法には、は、直接設計者から獲得する方

法とシステムが設計者の設計過程をモニターして獲得する方法がある。前者は従来の通りの獲得方法であり、VLSI設計用の知識ベース構築の初期の段階においては有効である。しかしながら、知識ベースがある程度大きくなると後者の方法が構築の効率の面において有効になる。後者の獲得方法の実現のためには既存の知識ベースを用いて設計者の設計例からルールが導出される必要がある。そのために演繹的学習法であるExplanation-based Learningを方法論として用いる。このExplanation-based Learningの基本的な考え方は、何故訓練例（Training example ）が目標とする概念（Goal Concept）の例になるかを説明し、その後で、この説明に利用されたドメイン理論（Domein Theory ）を基にして、その訓練例をより効率的に認識できるような形式に目標概念を表現し直すことで訓練例を一般化することである。そしてこの方法論をベースにした解析学的学習法がモジュール詳細化ルールの獲得には有効である。

ヒューリステックルールを獲得する方法については具体的には検討が行われていないが、設計プロセスを大局的にみることによって獲得されため設計プランに基づいて獲得する方法が有効であると考えられる。

VILLA においてはこれら設計オブジェクト、モジュール詳細化ルール、ヒューリステックごとに最適な獲得アプローチがどられることになる。各知識レベルごとの最適な獲得手法についてまとめたものを表3.3.2-1 に示す。

表3.3.2-1　VILLA における知識レベルと獲得手法

知識レベル	獲得手法
〔知識レベル１〕 設計オブジェクト （モジュール、データストリーム）	設計オブジェクトに関する各種制約条件や動作仕様を対話的に獲得する。
〔知識レベル２〕 モジュール詳細化ルール	設計者の実現法をモニターし、可能なものは一般化する。（解析的学習法）
〔知識レベル３〕 ヒューリステックルール （制御ルール）	設計プランに基づき有効なヒューリステックルールを対話的に獲得する。

(2) 設計者の仕事の中断の短縮

設計者の仕事の中断を最小限にするためには、設計者から対話的に直接知識の獲得するよりは、上記のモジュール詳細化ルールの獲得手法と同様にシステムが設計過程をモニターしてシステムが必要とする知識を獲得する方が有効である。

3.3.3　知識獲得の概要

設計オブジェクト〔知識レベル１〕は、前述のように設計オブジェクトの要求仕様、動作仕様の一部、制約条件、評価基準から設計オブジェクトの完全な動作仕様が導出されることにより行われる。そして完全な動作仕様の導出過程においてTMSによる支援が行われることになる。

モジュール詳細化ルールの獲得はまず設計者により与えられた具体的な設計例が既存のドメインルールを用いて設計の要求仕様を満足することを証明する。次にこの証明に使用されたルールの前提条件を基に要求仕様の重要な項目の一般化と一般化された詳細化ルールの導出を行うことになる。

図3.3.3-1 と図3.3.3-2 にこの解析的学習法を用いて設計者の示した具体的モジュール詳細化の設計例から汎用的なルール（〔知識レベル２〕）を獲得する例を示す。ここでは、設計の要求仕様（図3.3.3-1 におけるFunction to be implemented とWhere Input Signals Satisfy ）と設計例（図3.3.3-1 におけるuser's Solution）からに示されるモジュール詳細化の汎用ルールが生成される。

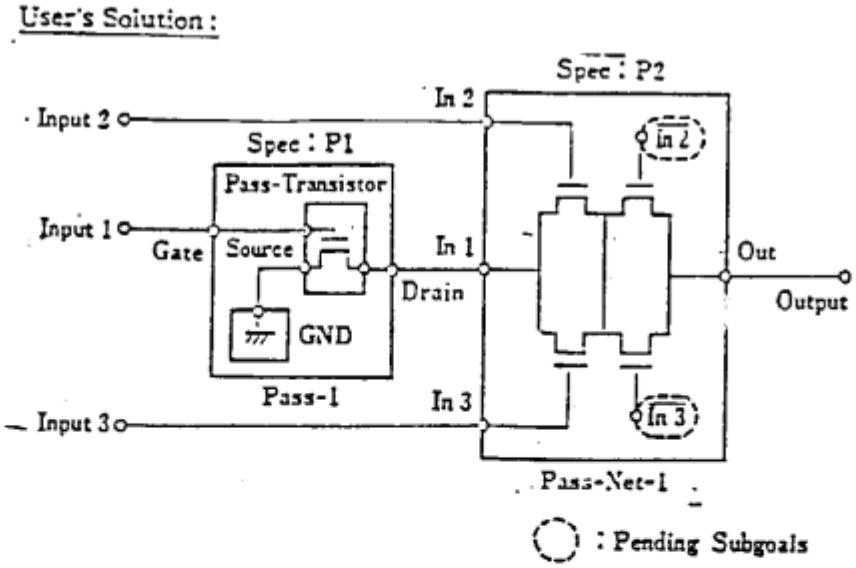


図3.3.3-1　設計例

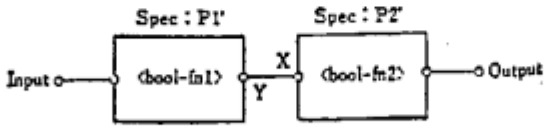
IF the function to be implemented is of the form:

Inputs: *any*
Outputs: Output
Function: (Equals (Value Output (i))
(If (And ⟨bool-fn1⟩ ⟨bool-fn2⟩)
Then ⟨any1⟩ Else ⟨any2⟩))

Where Input Signals Satisfy:

(Equals (Voltagelevel *any*)
(If (Equals (Voltagelevel ⟨any1⟩)
(Voltagelevel ⟨any2⟩)
Passing-Low-Level)
Then Switchable-High-Level
Else Restoring-Logic-Level))

THEN one possible implementation is:



With Specifications of the two modules as follows:

P1': (Equals (Value Y (i))
(If ⟨bool-fn1⟩ Then ⟨any1⟩ Else ⟨any2⟩))
P2': (Equals (Value Output (i))
- (If ⟨bool-fn2⟩ Then (Value X (i)) Else ⟨any2⟩))

図3.3.3-2　導出された汎用ルール

3.3.4　今後の課題

VILLA の今後の課題としては下記の３点が上げられている。

1.ヒューリステックルール〔知識レベル３〕の獲得手法の検討
2.VLSI設計エキスパートシステムに組み込んでのVILLA の評価
3.設計問題向き知識獲得方式の枠組みの構築とその枠組みのCLへの組み込み

ヒューリステックルールは設計者の経験から得られた設計作業に関する制御知識であり、この知識は作業効率に大きく影響する。このためヒューリステックルールの獲得方式の検討は重要である。また実際にVILLA をエキスパートシステムのなかに組み込んで知識獲得のアプローチの有効性を評価することも必要である。VILLA はCL上〔渡辺　85〕のVLSI設計エキスパートシステムのサブシステムとしてインプリメントされる予定である。そしてVILLA を通して一般の設計問題向き知識獲得方式の枠組みを構築し、その枠組みをエキスパートシェルの基本機能とそしてCLのなかに組み込むことが望まれる。

【参考文献】

〔渡辺　86〕渡辺、岩本、山野内、松田：VILLA ：VLSI設計知識獲得システム、電気通信学会、人工知能と知識処理研究会資料、AI-1(1986).

〔渡辺　87〕渡辺：エキスパートシステムにおける知識獲得、情報処理学会誌、VOL.28,1987,No.2,pp167-176(1987)

〔Politakis 84〕Politakis,P.and Weiss,S.M., "Using Empirical Analysis to Refine Expert System Knowledge Bases." Artifical Intelligence,22,1984,pp23-33.

〔渡辺　85〕渡辺、岩本、出口、"CL：概念ネットベース知識表現システム、"情報処理全国大会、1985年9 月、pp1221.

3．4　アナログＬＳＩレイアウト設計エキスパートシステム

ＬＳＩの設計、製造技術の進歩はめざましいが、それを支援するためのコンピュータ利用技術も著しく進歩している。しかしＣＡＤツールの大半はコンピュータの持つ高速演算能力を利用して、人手の単純作業の肩替わり、高速化をするものであり、人間の知恵や創造性に頼る部分は大きく残されている。

トランジスタ、抵抗等の素子レベルを考慮する回路設計においては、ＬＳＩの製造プロセス、回路構成手法等の深い理解に基づく高度な設計知識、ノウハウが必要である。とくにアナログ回路の設計においては、ディジタル回路と異なり、信号レベルの規格化が行われていないため、この特徴が際だっており、設計者に要求される熟練度が高く、非熟練者との設計品質の差が著しい。このため、このギャップを埋めるツールとしてエキスパートシステムに大きな期待がかけられている。

3.4.1　アナログＬＳＩ設計と知識処理

アナログＬＳＩの実態の調査を行ったが、設計全体を対象ととらえるにはあまりに広範にわたっている。アナログＩＣの種類はパワーＩＣあるいはＧＨｚの高周波用ＩＣ等の素子設計が中心になっているものから、素子数400 ～1000を越えるシステム的色彩が強いＬＳＩまでバラエティが多く、それぞれ設計の様相が大きく異なっている。またデバイス的にもバイポーラ、ＭＯＳ／ＣＭＯＳそれぞれ設計に応じた設計手法、設計ノウハウが存在する。ここでは典型的な例として、比較的設計過程がとらえやすく、作業イメージが明確なレイアウト設計過程における機能ブロック設計を取上げることにした。デバイスとしては、アナログＬＳＩでは中心的なデバイスであるバイポーラを取上げる。バイポーラは素子間分離する必要があるために、素子が占有する領域が明確であるという特徴がある。このために比較的単純なモデルでの知識の展開が可能である。

機能ブロックは４０～５０素子からなるＬＳＩの構成要素であるが、ディジタル回路のように標準ブロックをライブラリ化し、繰り返し使用されるということはあまりなく、その都度カスタム設計されるのが通常である。

アナログＬＳＩの設計は、回路設計とレイアウト設計とに一応区別することが出来るが、ディジタルＬＳＩの設計のように明確に分離出来ない。アナログ回路ではＬＳＩチップ上にレイアウトされる素子あるいは配線間の相互干渉が重要な問題となる。寄生素子の問題、基板の歪や熱分布による素子特性のづれ等の問題を考慮しなければならず、単に回路図上に素子を配置し、素子間を接続するだけでレイアウト設計を行うことは出来ない。

ＬＳＩ設計のＣＡＤとして現在使用されているプログラムシステムは、解析的な働きをするものか、あるいは配線設計のような比較的自由度の少ない、すなわち拘束条件の強い解空間での探索問題を解くものが多い。アナログＬＳＩ設計のように

非常に自由度が大きく、最適化問題となる部分問題に分解できないような設計問題に対しては、殆ど無力である。

設計のあらゆる面において仕様における“曖昧さ”が残されており、トップダウン的に設計していく際の設計候補の選択は、設計者の手に委ねられている。この際にキーとなるのは、バランスの感覚である。設計結果は単に、設計仕様に記述された条件を単純に満足すれば良いわけではない。仕様として明確に表現されていない条件も存在し、また必ずしも数知的な表現が出来ない要求項目もある。中でも相反する条件のバランスをどのようにとるかが重要である。Ａを良くすれば、条件Ｂの特性が悪くなるといったことはごく普通に起こり得る。設計問題を全て最適化問題のサブ問題に展開することができない。特にその時点での判断の正当性が即時に検証出来ない場合、その時点での判断が非常に重みを持っている。アナログＬＳＩ設計の場合、実際に製造まで行い、テストをしてみて始めて不都合を発見したとすると、その間の数週間、数ヶ月をさかのぼって設計のやり直しを行わなければならない。設計者にとっては、それぞれの時点での判断が、その後にどのような影響を及していくかを示すシュミレータが存在することは望ましいが、単純な回路解析でさえ数十素子レベルまでしか取扱えない今日の解析技術レベルでは、とても不可能なことである。そこで、過去の設計例等の経験をもとにした結果の予測という手法がとられることになる。人間の判断においても、“実際にこのようにした時には、このようになった”という情報が依りどころになっている。この時の知識には、大きく２つあるように思われる。第一は、過去にうまくいった設計例における記憶、知識である。１つのことを実現するための方法は幾通りも考えられるのが通常であるが、うまくいった方法が前例になって新規の設計の際の参考にされる。実際にこのようなやりかたで探索空間を非常にうまく絞っている。このような探索空間を絞りこむための知識をエキスパートシステムに蓄積することにより、経験のあるプロセス技術等を使用する設計に対しては、多いに役立つ。第二は、過去における失敗の知識である。テストにおいて不都合が発見されると、その原因は徹底的に調査されて、ドキュメントとして残される。これらの大部分は単純ミスであるが、中には、通常では見逃しやすいものもある。この種のミスのチェックサポートには、エキスパートシステムは設計者に強力な援助を与えてくれる。

3.4.2　設計知識の表現

アナログＬＳＩ設計の顕著な特徴は次のようなことである。個別部品と比較して、素子特性のバラツキは非常に大きい。しかし、レイアウトに注意することにより相対的な精度はかなり向上させることが出来る。例えば、絶対精度で±３０%程度バラつく抵抗でも相対精度を±１～３%程度のバラツキに抑えることが可能である。これはＬＳＩの大きな特徴であり、回路構成は素子の絶対値でなく、相対値で特性

が決まるような回路設計がなされ、そのための回路構成、レイアウト上のさまざまなノウハウがある。例えば、次のような知識がある。

(a) 整合性を満足させるための知識

回路設計において比精度を要求されている素子をどのようにレイアウトしたら良いかに関する知識である。

(b) 交差配線の知識

配線の引回しは、回路全体の特性に及す影響が大きいので、重要である。回路図上は配線の交差は問題ではないが、レイアウト設計では、一層配線を用いる場合、異なる信号がショートしてしまうので、交差配線（クロスオーバー）を用いてこれを回避する。

(c) 分離領域の知識

バイポーラＬＳＩの場合、Ｎ－Ｐ接合に逆バイヤスをかけて領域分離を行う必要があるため、チップ面積に及す影響が大きい。分離領域の設定のしかた、バイヤスのためのコンタクトの設定のしかた等、多くのノウハウがある。

以上に述べたような知識を状況に合わせて活用しながら設計者は、できるだけ小さいチップ面積で、要求された回路特性を満足するレイアウトを設計する。

レイアウト設計の実際をインタビュー調査したところ、次のような設計の流れが分った。レイアウトの初期段階で、ラフレイアウトが行われる。ラフレイアウトはレイアウト設計者が、各素子の形状、配置、配線の概略図を、机上でフリーハンドで描きながら、レイアウトの最適案を検討してゆく作業である。この際には、詳細な設計ルールや厳密な物理形状ではなく、だいたいのトランジスタの大きさ、抵抗、キャパシタの形状を頭におきながら、概略図を考えて行く。このラフレイアウトの作業を分析したところ、設計者はラフレイアウト図の初期情報として回路図を用い、そこでの配置レイアウトを出発点として、部分的に素子配置、配線経路を修正しながら、ラフレイアウト図を作成している。ただし、このような作業はかなりの部分が設計者の頭の中で行われている。

回路図は、単に素子間の配線接続を表現しているだけでなく、一種のシンボリックレイアウト図であり、素子の相対配置、配線の引回し方などが重要な意味を持っている。最上部に電源ライン、最下部にグランドラインを設定し、主たる信号の流れが、左から右へとなるように作成するといった一定のルールに従っており、また回路設計者が意識的あるいは無意識的に素子密度や配線密度の一定化を計っているからである。従って、レイアウト設計において回路図における素子配置、配線をもとに各素子、各配線線分を、物理的なレイアウドデータに変換してゆけば、最終的なレイアウトが得られる。この様な設計方法論もまた大きな知識である。本エキスパートシステムは、この設計手法に沿って支援するものである。

3.4.3 システム試作

対象として取上げたのは、非常にバラエティの広いアナログＬＳＩの中でも数の上から見ても標準的なバイポーラアナログＬＳＩの素子数４０～６０程度の機能ブロックのレイアウト設計である。アナログＬＳＩは回路素子パラメータがレイアウトに大きく依存するために、レイアウト設計と回路設計とを明確に切離す分けにいかない。また、レイアウトによる寄生効果をまで取込んでシミュレータは実用的には等分見込みが薄い。このために、回路設計結果に近いレイアウトを与えるためにこれらの諸条件は経験的知識に基づいてレイアウト設計時に考慮されねばならない。

［入力］：入力は回路設計の結果である。新たにレイアウト設計を行う場合あるいはレイアウト結果からフィードバックされ回路設計の修正を行った場合がある。入力項目として、最も重要なものは回路図である。回路設計に関しては、ＬＳＩ製造のためのプロセスを決定し、使用する各素子の設計も完成しているものとする。すわほち、トランジスタとしては、数種類のものがあらかじめ容易されており、いずれかを選択して用いる。その時のレイアウトの状況に合わせて、１つの種類のトランジスタの中にバリエーションをあらかじめ設計しておく。回路図の各トランジスタにはトランジスタ種類名が付加されている。回路図に陽な形で記されていない素子接続情報以外の情報が含まれている。これらは回路図を読む人が、回路の意味から読取るべき情報であるが、簡単のためここでは、素子間の相対的精度、バイヤス条件等の付加情報をシステムの入力情報とする。

［出力］：物理レイアウト、各素子の形状および位置、配線線分の幅および位置、素子間の相対精度条件からのレイアウト制限を満足する、なるべく少ない面積の機能ブロックのレイアウトを生成出力する。なお、機能ブロックの外形は矩形で、上辺は電源ライン、下辺はグランドライン、入力端子は左辺または右辺上にあるものとし、回路図での相対位置にならう。

本システムは、２つのフェーズに分れる。第１のフェーズでは会話型に回路図を編集して、素子および配線の相対位置を決めるラフスケッチを行う。このフェーズは、回路図のシンボルを用いてレイアウト表現を行う。すなわち、素子の相対位置および配線の引回し方法のレイアウト設計において本質部分を決定する。この設計部分では、設計を進めるのはあくまでも人間の設計者であり、それらに対してエキスパートシステムは回路特性を実現するめたのノウハウに基づいて設計者に助言を行うという形態を取る。

システム試作では、次に(a)(b)(c) の知識群のインプリメントを行った。

(a) 交差解消の知識

配線交差の解消は、図11. に示すように抵抗上の交差利用、トランジスタの電極間の通過配線の利用、キャパシターの電極間通過配線の利用等によって行われる。可能な手段の提案を行うが、素子配置、配線の相対位置がそのままであることを前提としたローカルな判断であるのでユーザのグローバルな判断が必要である。

(b) トンネル抵抗挿入の知識

交差する2つの信号線のどちらにトンネル抵抗を挿入すべきかの知識である。信号線の優先度に関する一般ルール、トンネル抵抗挿入の可不可の判断ルール等を含む。

(c) 素子グループ化の知識

同電位の分離領域はまとめることにより面積を小さくすることができる。分離領域の島の数は少ない方が望ましい。一般に、抵抗同志、トランジスタ同志をグループ化するとレイアウト上有利なことが多い。第2のフェーズではラフスケッチを詳細化しマスクレイアウトパターンを生成するフェーズである。このフェーズでは、回路図シンボルで表現されたレイアウトを物理的な大きさを持つ素子および配線のマスタパターンに展開する。これは物理的な制約条件を満たすようにチップ面積の最小化をはかるもので、物理的な展開のための知識を知識ベースに表現し自動的な処理を行い初期レイアウトを生成し会話型にエディットし最終レイアウトを定める。

回路図のエディットのフェーズでは、配線交差の解消と、整合あるいは相対精度の要求がある素子の相対位置の決定等を行う。

図3.4-1 にシステムの構成を示す。

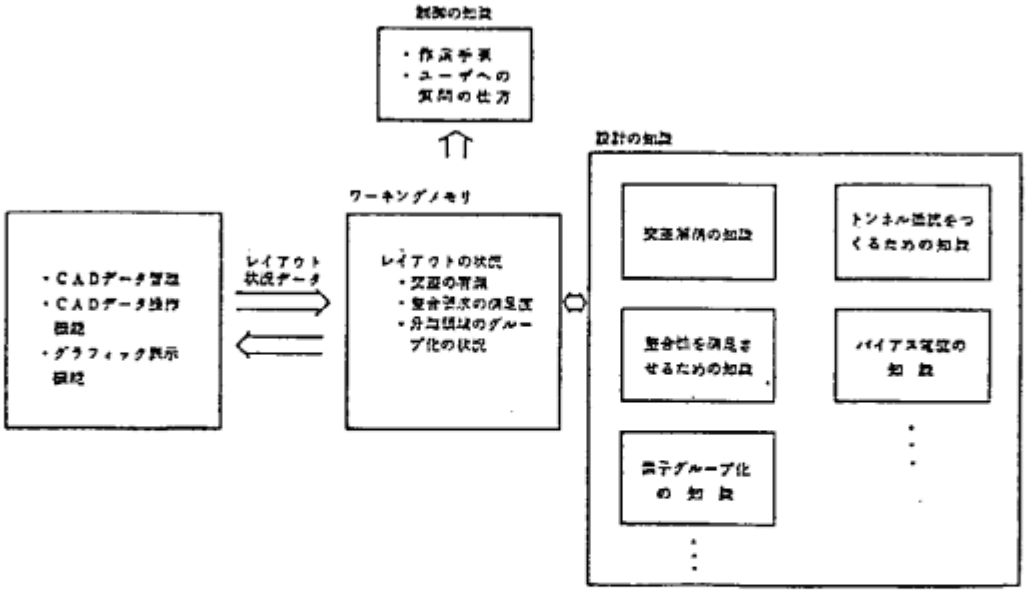


図3.4-1　システム構成

3．5　航空機設計における知識システムのニーズ

航空機は、あらゆる分野での最先端技術を統合した複雑な精密機械であり、その開発には、ＣＡＤ／ＣＡＭ等の計算機技術が大きな役割を果たしている。この傾向は、今後も急速に増大するであろうことは多くの人の予想するところである。航空機工業におけるＣＡＤ／ＣＡＭについては、参考資料の［大須賀　82］・［大須賀　83］に詳しく検討されており、航空機の性能向上、開発設計作業効率向上のため、将来のＣＡＤ／ＣＡＭシステムに知的情報処理システムを応用することが提案されている。ここでは、航空機設計作業の中で、空力設計作業を例にとりあげ、航空機設計における知識システムのニーズをさぐった。

3.5.1　航空機設計の流れ

航空機設計の流れについて簡単に説明する。航空機設計は、まず概念設計・基本設計と呼ばれるフェーズで開発機の大枠が設定される。

概念設計では、開発機のカバーすべき範囲を検討し、これを要求仕様として明確にする。続いて数多くの基本形状モデルについて性能・市場性の観点から評価が繰り返され、最終的には一つのモデルを選択する。（機体緒元策定）

次いで、基本設計では、要求仕様を満たす最適の航空機を製作すべく、外形形状、構造、装備及び装備品の配置案を設計し、重量見積り、性能評価、安全性の確認等によって設計モデルの最適化修正が繰り返され、機体の基本的な緒元が決定されていく。（空力設計、構造設計、装備設計）

このあと、詳細設計を経て、製造へと作業は移っていくのである。

3.5.2　航空機設計作業の例－空力設計［谷岡　85］

航空機の空力設計では、現在までに蓄積された各種の空力、性能データ等の知識を整理し、相互に関連づけてまとめられたデータベースを元にして、開発する航空機の諸性能を満たすように試行錯誤の手法で主要緒元が決められていく。次いで、求められた機体主要緒元を元に、風洞試験と空力計算により、航空機の空力特性を検討しつつ、機体各部の具体的な空力形状を決めていく。

例えば、航空機で重要な主翼の空力設計は、次のような進められる。

図3.5-1 に主翼空力設計の流れを示す。図中、太線で囲まれた部分は、計算機により実行される部分であり、それ以外は設計者の設計・試験作業である。

機体緒元策定で求められた主要緒元をもとに設計が始まる。航空機は広い速度域で飛行するが、音の早さを境にして空力的な諸現象が異なるため、音速以下及び音速付近の速度である亜・遷音速域と、それ以上の超音速域に分けて空力特性を検討する。最適形状は、両速度域での検討が終わったあと、両者の最適形状が付き合わされ全速度での最適形状へコンプロマイズする過程を経て、求められる。各フェー

ズでは、設計案策定・評価・設計案修正が満足するまで、繰り返し行われる。

3.5.2.1 知識システムのニーズ

上述したように、現在の空力設計は、計算機解析を設計の各フェーズで利用しながら進められているが、従来の計算機システムに任せることが難しく、経験豊富な設計者の能力にたよる部分も多い。これらを整理すると下記のような作業に大別できる。そして、これらは、ある程度他の分野の設計にもあてはまることと思われる。

(1) 初期設計案の策定：過去のデータ等をもとに、要求（空力特性）を満足すると思われる設計案（形状または、形状に変換し易い量）を策定する作業。

（例）圧力分布設定：設計条件（空力特性）から、過去の経験データをもとに圧力分布候補を設定する作業。

(2) 解析方針の設定：設計案を以下に評価するべきかを計画する作業。解析項目の選定等を含む。

（例）解析方法設定：解析対象の形状、速度、などに応じてプログラム選定、差分解析の節点設定、初期値・境界値の設定、差分化方法選定などを行う。

空力解析には、次のような特徴があり、解析方針の設定は複雑且つ重要なものの１つである。

（ア）使用する計算プログラムが多種にわたり、また計算プログラムの使用方法に自由度が大きいため、解析対象・条件に適したプログラムとその使用方法を選定する必要がある。

（イ）一つ前の計算結果を見て次の作業内容を決めるため、方針設定が作業効率に与える影響は大きい。フロータイムを短縮するために、多くのケースを並行処理することもできるが、コストが大きくなり、限度がある。

（ウ）設計者の目的に適した幾通りもの作業手順がある。

（エ）計算の規模が大きく、所要時間・計算コストが大きいので、無駄な計算回数を極力減らし、最短路で目標に達したい。

(3) 解析結果処理：解析結果と仕様とを比較し、仕様との違いの重要性を判断し、ＧＯ／ＮＯＧＯを判断する。ＮＯＧＯの場合、設計案の修正を検討する。解析方法に検討の余地がある場合は、解析方法の検討・修正をする。

（例）空力解析の結果を評価し、満足できる結果が得られるまで、解析方法・機体形状・圧力分布の修正をする作業。

(4) コンプロマイズ：相反する要求をコンプロマイズする作業。

（例）亜音速に最適な翼型と超音速に最適な翼型から、全速度域で最適な翼型を立案する。

3.5.2.2 設計支援システム

図3.5-2 に上記の設計作業を支援する知識システムを示す。設計は、既存の計算機解析システムを利用することが不可欠であり、設計支援知識システムも計算機解析システムと深く関連することが必要である。こうして、設計支援知識システムとして、図3.5-2 のように従来の計算機システムと設計者の間にあって、従来の設計者の作業を一部代行するシステムが考えられる。従って、システムに必要な知識としては、空力に関する知識、設計の流れに関するものなどの他に既存の解析システム利用法に関する知識が重要となる。機能的には、システムは、データ検索、設計案策定、解析方針設定、解析結果表の４つの機能を有する。これらの必要な内容は、下記のようなものものになる。

(1) データ検索は、設計の全フェーズに必要な機能であり、設計者がある作業を進めるに当って、必要または知っていた方が良いと思われる情報を適宜検索する機能である。

検索するデータの種類としては、設計標準・過去の戦訓などが想定される。設計者から支援システムへの指示はミニマムであることが望ましい。

(2) 設計案策定し、初期設計案策定段階で、初期設計案を設計者が設定する作業を支援する。この段階では設計支援の方法として、データ検索の機能を利用し、過去の設計例の提示、過去の設計例と今回の要求仕様との比較から考えられる変更点の提案などが考えられる。また、解析の結果に基づいて、設計案にフィードバックする作業を支援する。そして、相反する要求をコンプロマイズする作業もこの機能が支援する。

(3) 解析方針設定し、解析対象・解析フェーズなどに応じて、最適な解析プログラム、解析プログラムの仕様方法などを決める作業を支援する。また、過去の解析結果に基づいて、次の解析方法を考える作業を支援する。

(4) 解析結果評価は、解析結果から有意な情報を抽出する機能である。解析の結果は、膨大であり、これを設計者に分り易い形で提供することが必要である。有意な情報とは、要求仕様と解析結果との差を重み付けすることにより得られると思われる。

上記の各機能の具体的実現の他に、図3.5-2 には明確に現れていないが、全般的な点で、下記のような要検討項目がある。

一つは、利用対象がエキスパートか、平均的設計者か、初心者かにより支援する箇所、必要な機能が変わって来るためこの対応をどうするかという問題である。

即ち、エキスパートが対象の場合は、ユーザが立案する設計案について、マクロな指示で必要な解析を実施し結果を答えるという「知的な解析」機能、データ検索機能などが必要な機能になるであろう。

一方、平均的設計者が対象の場合には、エキスパートが彼に与えるであろう指

示（解析方針、解析結果に対するコメント、過去の戦訓など）を与えるような機能が適当と思われる。

また、初心者が対象の場合は教育的配慮も必要であり、指示の理由を教えながら進める機能も必要であろう。

二つ目の課題は、複雑な３次元形状処理の取り扱いである。現在の知識システムは、図形データの取り扱いが不得手であるが、設計支援システムでは３次元形状データが必須であり、これを如何に扱うかの問題を解決する必要があろう。

【参考資料】

[大須賀　82] 大須賀節雄　他、航空宇宙工業におけるコンピュータ利用高度化調査検討報告書（（社）日本航空宇宙工業会、１９８２）。

[大須賀　83] 大須賀節雄　他、航空宇宙工業におけるコンピュータ利用高度化調査検討報告書（その２）、（（社）日本航空宇宙工業会、１９８３）。

[谷岡　85] 谷岡忠幸　他、航空機の空力とＣＡＤシステム、三菱重工技報Vol.22 No.6.(1985-11)。

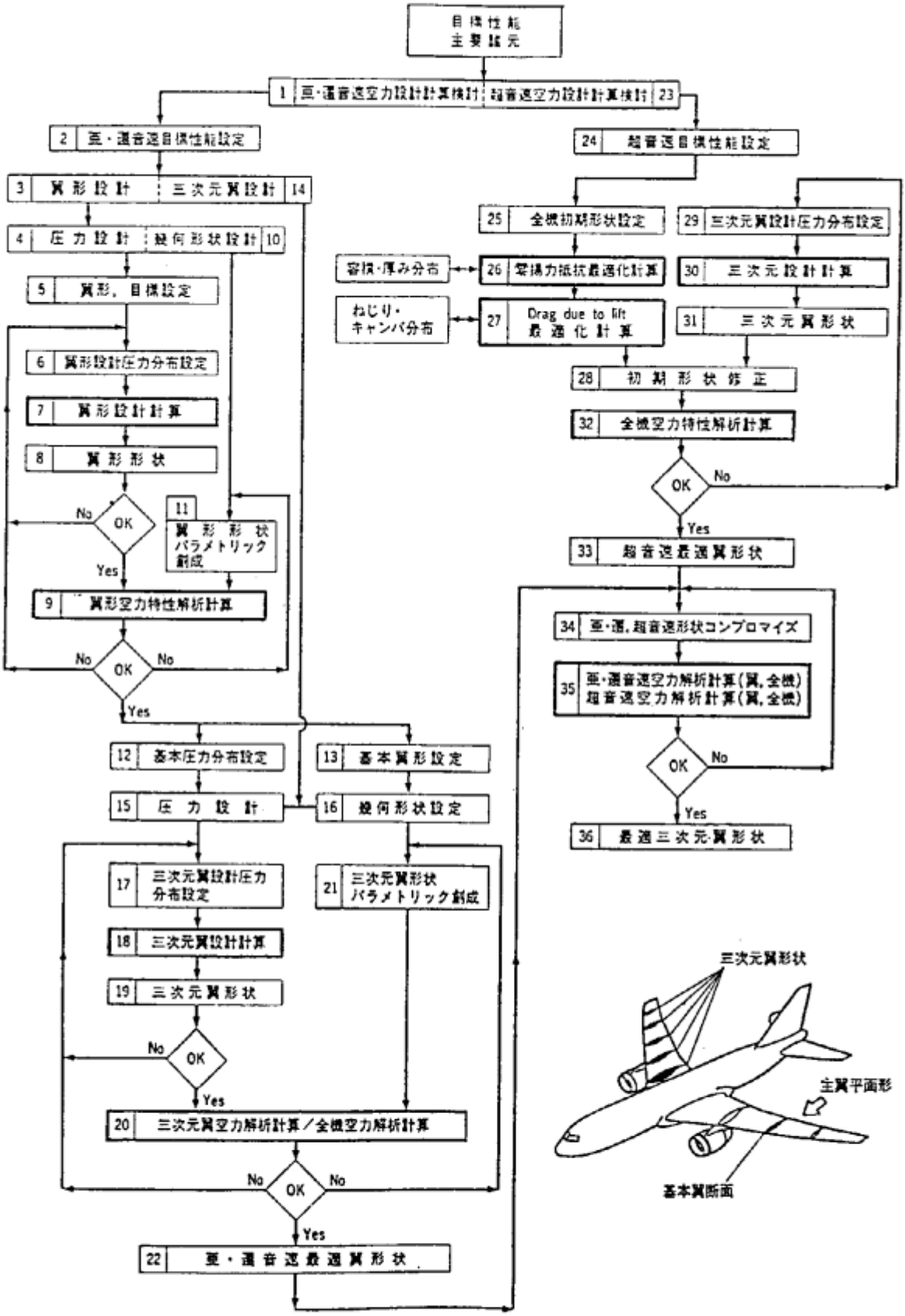
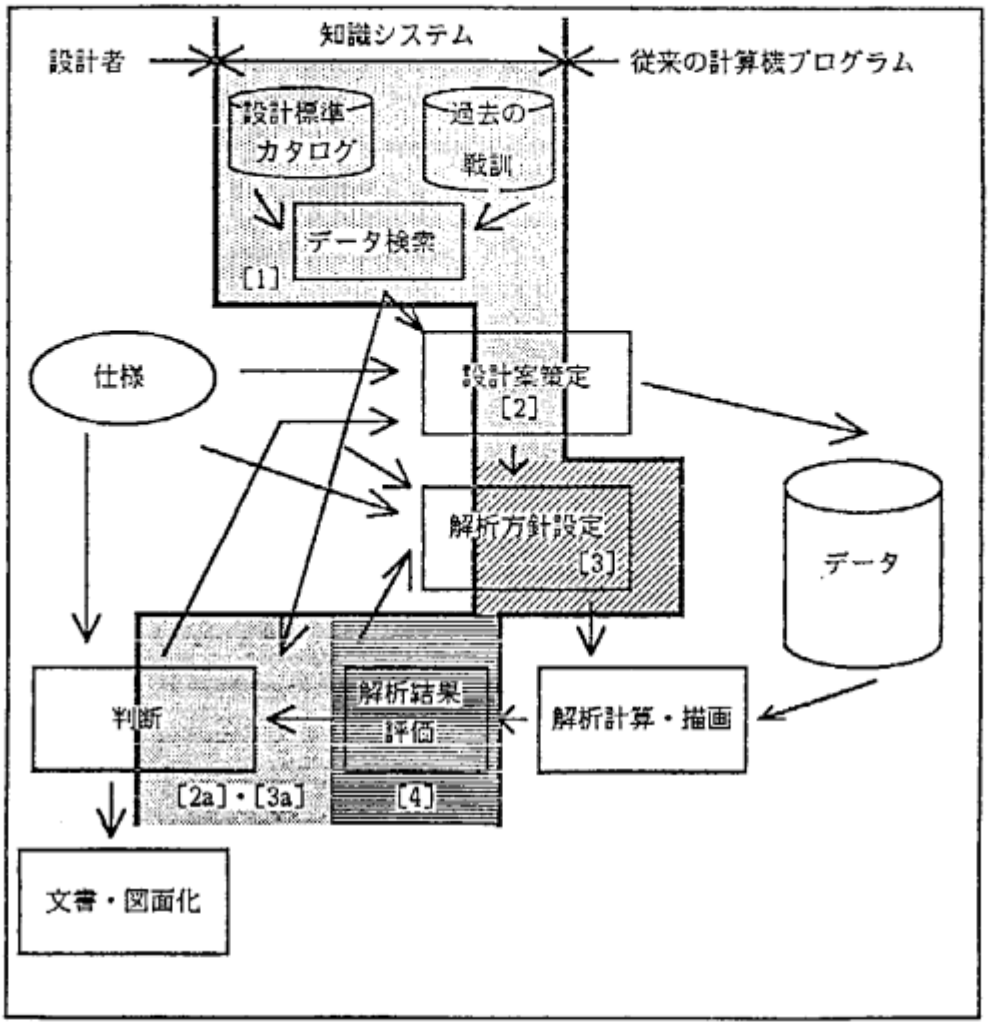


図3.5-1　主翼空力設計作業の流れ



応用箇所	上図との対応	具体的内容例
データ検索	[1]	設計対象に合った、必要データを検索する
設計案策定	[2]	仕様から設計案を策定する
	[2a]	解析結果と仕様の比較により設計案を修正する
解析方針設定	[3]	解析項目の選定 解析プログラムの解析条件を設定する
	[3a]	解析の結果により、解析の条件を変更する 解析の結果により、次の解析項目を設定する
解析結果評価	[4]	解析結果のなかで有意なものを選び出す

図3.5-2　設計支援知識システム

4．合成型知識獲得の特徴

4章では、3章で紹介した事例に基づいて、合成型問題の知識獲得の特徴について考察する。まず、4．1で事例について分析、考察し、ついで注目すべき特徴点として4．2と4．3で解空間の扱いと知識のレベルの問題について述べる。4．4ではまとめを述べる。

4．1　事例分析と考察

3章で紹介した事例は、具体的なエキスパート・システム開発の経験に基づくものから知識情報処理技術の今後の適用可能性を論じたものまで幅が広い。具体的にエキスパート・システム化されたものについてはポイントを絞った知識獲得がなされているが、そうでない事例については知識獲得の問題点－特に、深い知識の獲得ほど難しいこと－が指摘されている。以下では各事例に現れる知識獲得の問題について分析を行い、それらに共通する特徴を明らかにする。

(1) 毛筆文字の生成における知識獲得

創造的分野における知識情報処理技術の適用例である。この分野の特徴は、①高度な能力をもつ専門家の経験的知識が、抽象的ではあるもののある程度は定式化されていること、②それを人間が正しく理解するにはそれなりの訓練が必要となること、③文書化された知識（書道の教科書中の規則）には互いに矛盾する情報が多く、それらを計算機に移植する場合には、そこに明示されない経験則を専門家から補わなければ役に立たないことなどが上げられる。したがって、このような分野の知識獲得の問題を解決するには、まず、曖昧な経験的知識と現在利用可能な知識表現手法とのギャップを埋めるための研究が大きな課題であり、さらに、経験的知識の背後に存在する深い知識の明確化が必要となると考えられる。

(2) 事務処理システムの開発における知識獲得

大規模な事務処理システムの開発に知識情報処理技術を適用するには、①対象分野となる業務の知識、②それを計算機モデル化するための知識、③プログラム開発に必要な計算機の知識を定式化する必要が生ずる。第1の知識はいわば対象分野のワールドの知識であり、この獲得を効率化するためには毛筆文字生成の場合と同様に、深い知識を取り扱う手法を確立しなければならない。第2の知識はシステム利用者の要求仕様を誤りなく収集するための知識であり、ソフトウェア工学にみられる要求仕様化技法と知識情報処理技術で研究されている知識獲得手法とを組み合せることによってある程度発展が望めると考えられる。第3の知識は、アルゴリズムに関する知識などを含んでおり、ＬＳＩ設計、エンジニアリングシステム設計などに必要な知識と共通の性質が多い。したがって、このような知識の獲得には階層的

な手法との適用が可能と考えられる。

(3) ＬＳＩ設計における知識獲得

VILLA では、具体的問題とその解とが与えられた時に、その問題パターンを条件部に解を結論部にもつルールを生成する方法を取り扱っている。このシステムの特徴は、知識を定式化するのに、一般的に行われている複数の例に基づく帰納的な方法にかわって、Explanation Based Learningに基づいて単一例から知識を演繹的に獲得する方法を用いていることである。このような方法が有効なのは、回路合成のように解の性質が比較的明確で、また、知識獲得に必要なドメイン知識がよく整理された分野である。すなわち、このシステムで獲得される知識は、設計問題に必要とされる設計対象の知識、モジュール詳細化知識、設計効率化のための専門知識のうち、特に２番目のモジュール詳細化知識を中心としたものである。

(4) アナログＬＳＩレイアウト設計における知識獲得

本エキスパート・システムは、利用者との対話を続けることによって、自動配線プログラムでは結線できないＬＳＩの配線レイアウト設計作業を支援することを目的としている。この問題の特徴は、大規模な組合せ問題を解くこと、適当な解をひとつ求められればよいこと、ならびに、解を見出すためには人間－機械系による浅い知識－設計効率化のための専門知識－が重要である。このような知識の獲得には、通常、インタビュー方式、弟子入り方式などが用いられる。その自動化には、専門家の知識を計算機向きに自動的に変換する手法の確立が重要と考えられる。

(5) 航空機設計における知識獲得

航空機設計では、要求仕様を満たすべく、蓄積された空力、性能データ等の知識や、風洞実験、数値計算から得られる情報をもとに作業が進められる。ここでの特徴は、設計知識が大量の数値的なデータや数値計算プログラムとして表現される場合が多いことである。これらの獲得は、データベース開発やプログラム開発の作業を通じて、implicitになされるのが普通である。したがって、この種の知識獲得はプログラム開発の自動化における知識獲得とよく似た側面を持つと考えられる。一方、設計作業を効率化は計算機と設計者とのインタラクションの高度化によって達成されるという立場にたてば、そのための知識、すなわち、"知的な解析"機能を実現するための知識の獲得には、(4) で述べたような浅い知識を獲得するための手法が有効であろう。

合成型問題においては、設計過程のモデルをどのように把握するか、また、設計知識をどのように捉らえるかがポイントとなる。

［長沢 87 ］では、設計過程のモデルを、①設計情報をすべて保持しておいてそれを検索することで設計を行うと言う全数対応モデル、②数値計算あるいは最適化手法によって解を導く計算モデル、③ルールを問題解決に利用する生成モデル、④専門家の方法を規範とする凡例モデルに分類している。各モデルごとに知識獲得の問題は別々に考察できるが、そこに共通する問題は、解空間の大きさをどのように克服するかという点であろう。これについては４．２節で論ずることとする。

また、設計知識については、やはり［長沢 87 ］において次の３種類のレベルに分類できることが指摘されている。第１は設計対象についての拘束条件、構造、評価関数などの知識であり、第２は設計案を操作するための、要求から設計案へいたる詳細化知識、設計案の修正知識であり、第３は解の探索を行うための、経験則や実行順序、評価尺度などを扱う設計制御知識である。このうち、第１のレベル、ならびに第２のレベルの一部は設計データに関する知識と考えることができ、また、第２、第３のレベルは設計のノウハウに関する知識と考えることができる。そこで、４．３節ではこの２つの視点からあらためて考察を行う。

４．２　解空間の扱い

合成型問題の特徴のひとつは、その解空間の大きさである。これは、例えば、診断問題のそれと比べてもオーダーで上回る。従って、明らかに、generate-and-test（完全な解の候補を生成してテストするという意味で）あるいは探索といった手法は無力である。

設計者は、全体の仕様を睨みながら、徐々に設計を詳細化し、全体の回路を明らかにしていく、この過程は、副作用を起して環境を変化させることにより、計算機上に実現できる。

大きな問題に対処するひとつの手段は、部分問題に分割することである。合成型問題では、多くの場合、問題を局所的な部分問題に分割することが可能である。局所的な部分解を組併せて、どのように大局的に最適な解を得るか、が問題となる。

４．３　知識のレベル

4.3.1　知識と設計データの表現

設計システムが扱うものとして、知識（ノウハウ）と設計データを区別しなければならない。

設計データは、膨大であり、どのように効率良く表現するかが問題となる。すなわち、設計データの構造化が重要である。それとともに、設計データを操作する手続きをどのように実現するかが、知識（ノウハウ）の表現の問題とも絡んで、大切である。

ひとつの方法は、オブジェクト指向の考え型を導入することである。個々の設計データの表現の枠組みとしては、フレームが有効であろう。知識の中で出て来るような手続き（通常、設計データの表現とは独立）に対して、実際にはいくつものインプリメンテーション・レベルの手続きを実行しなくてはならない。前者はマクロな操作手続き、後者はミクロな手続きのみが関心事であり、システム側がマクロな手続きのミクロな手続きへの翻訳を管理すればその負担が軽減する。意味的にも、このような翻訳は、あくまで設計データの表現形式に依存する作業であり、ノウハウはこのようなものとは独立である。

図4.3-1 に設計データ操作手続きのオブジェクト指向による実現について示す。

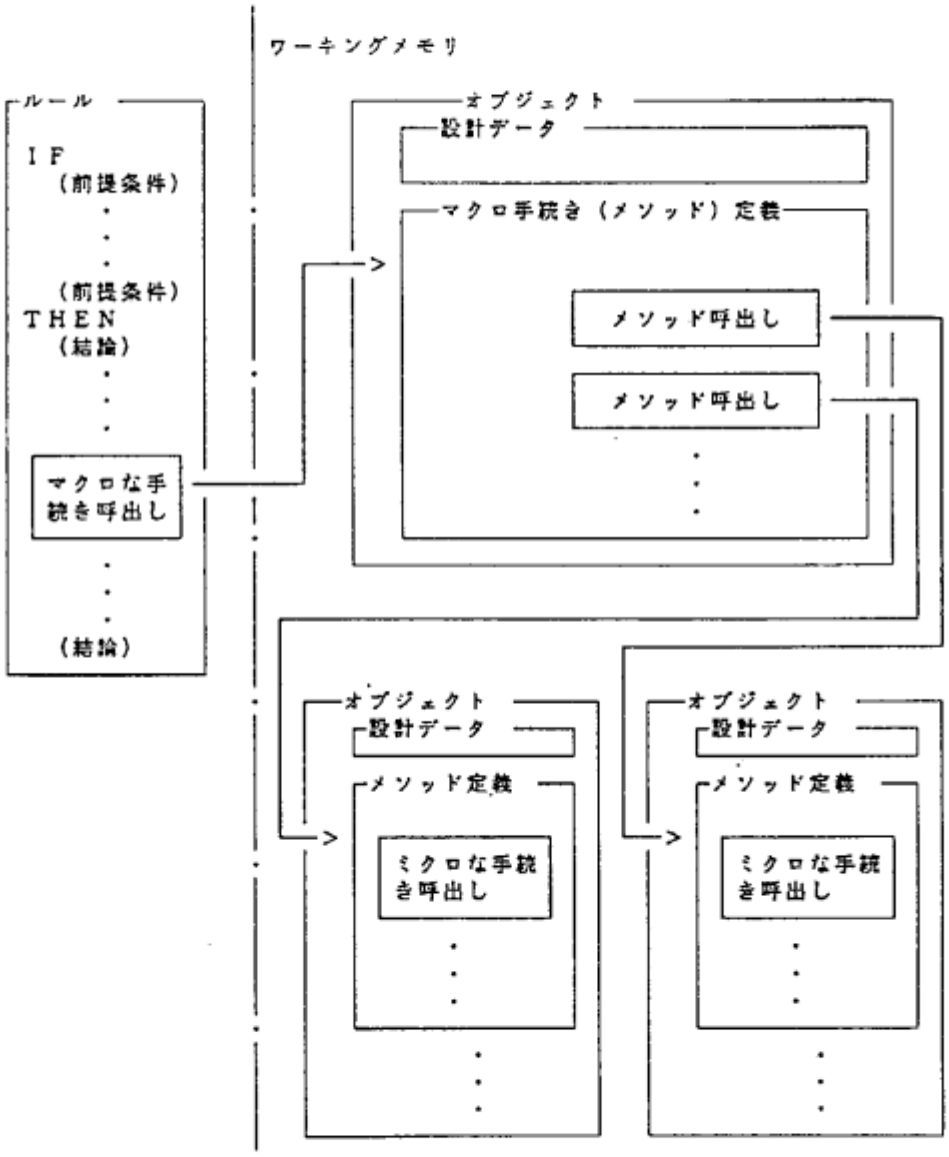


図4.3-1　オブジェクト指向による手続きの実現

この実現法では、オブジェクト指向の特徴である情報の隠蔽が重要な役割を果している。何段階かの隠蔽が実現されている。まず、マクロな手続きレベルでは、マクロな手続きがどのようにインプリメントされているのかはその使用者には隠されている。つまり、そこで使う手続きがどのように実現されているのかを意識せずにルールを記述することができる。また、ひとつの（ミクロでない）手続きのレベルでは、それが下位の手続きでどのようにインプリメントされているのかが使用者に隠されている。

知識（ノウハウ）の表現に関しては、設計という作業が、複数の対象間の関係を参照しながら進められていく、という特徴に注目しなければならない。このことから、設計の知識には、設計要素間の関係を参照するものが多い。「述語が因数間の関係を表現する」という論理型言語の性質は、設計作業における知識を表現するのには非常に都合が良い。

4.3.2 試行錯誤

設計という作業は、試行錯誤のプロセスを伴う。これは、ひとつには、設計者が個々の決定（design decision ）の及ぼす影響を正確に予測することができないことによる。繰り返しは、ひとつの設計過程（design stage）内で起る場合もあれば、何段階かの設計過程に戻ってやり直すこともある。いずれの場合も、やり直しを管理するメカニズムが必要であろう。

4．4 まとめ

本章では、3章で述べた事例を中心に合成型問題の知識獲得の特徴について考察した。現在の知識獲得ツールでサポートできるレベルは、［渡辺 87 ］にも記述があるように、エキスパート・システム開発の諸段階（要求確認・概念化・定式化・実現・テスト）のうち、ようやく、最後の段階－実現・テスト段階－で使える程度のものである。ところが、事例に即して考察すると、むしろエキスパート・システム開発の初期段階において、おのおのの合成型問題をもつ対象分野に特有な知識獲得の問題点が明らかになっており、これを解決するための研究開発が今後必要である。

【参考文献】

［渡辺 87 ］ 渡辺正信：エキスパート・システムにおける知識獲得。情報処理、vol.28,No.2,pp.167-176 (1987年 2月)。

［長沢 87 ］ 長沢 勲：設計エキスパート・システム。情報処理 vol.28,No.2,pp187-196(1987年 2月)。

5．知識獲得支援機能を実現するための課題

5．1　オブジェクト指向アプローチの有用性

5.1.1　合成型問題解決における知識獲得とオブジェクト指向アプローチ

現実世界での計画／設計型問題解決は一般に相当程度複雑であり、知識獲得という言葉でニュアンスされる高度な自己組織化能力を期待することは、技術の現状においては困難と思われる。そこで知識獲得問題をより広くとらえ、対象とする問題解決システム全体を計算機上にインプリメントすることの容易さ、別の言葉で言えば問題解決に係わる情報体系全体を計算機外部から計算機内部に移植することの容易さを実現する問題として把え、様々な観点からオブジェクト指向アプローチの有用性を考えてみたい。なお、ここでは考察のための基本視座として、既に述べた計算／設計型問題解決の仮説的モデルを用いることにする。

計画／設計型問題解決
＝　初期動機　--------- 創造的
＋要素定義＋相関定義＋目標定義＋解　--------- 表示的
＋求解　--------- 操作的
＋適応　--------- 生態学的

また、オブジェクト指向言語の特色としては、次の点を取りあげる。

オブジェクト指向
＝　概念体系記述力
＋上位概念継承
＋クラス概念
＋スロットとメソッド

以上の、計画／設計型問題解決及びオブジェクト指向概念の持つ諸側面との間の寄与の関係を表5.1-1 に記載する。

表5.1-1　知識獲得とオブジェクト指向思想

諸側面	知識獲得	オブジェクト指向思想			
		概念体系記述	上位概念の継承	クラス概念	スロットとメソッド
初期動機	—				
要素定義	体系的記述能力	○	○	○	○
相関定義	定義や変更容易化	○	○	○	○
	旧問題再利用能力		○	○	
	メタレベル知識保有				○
目標定義	同上	○			
求解過程	旧解再利用能力	○			○
	解法生成,再利用改良能力	○		○	○
解	解の知的保持・検索	○	○	○	
	メタ知識による解可塑化		○		○
適応	変更保守容易性	○	○	○	○

以下では、この表の内容について説明を行う。

5.1.1.1 初期動機での知識獲得側面とオブジェクト指向

どのような問題を解くかといった点を決定するという多分に創造的な側面については、知識獲得という機械化機構について語ることは難しいと思われる。

5.1.1.2 要素定義での知識獲得的側面とオブジェクト指向

オブジェクト指向思想における概念体系の記述機能は複雑な階層関係にある対象要素の記述に向いている。上位概念の継承機能は、別の見方からは、計算機に常識を持たせることに対応し、対象要素の定義にあたって、特異的情報のみを定義し、それ以外は背後の常識、上位の常識を継承させることでまに合わせることを可能にしてくれる。クラス概念は、通常言語のマクロ機能に相当し複雑な対象要素間の簡略記述を可能化する。スロットとメソッドの概念は、対象の持つ静的、

及び動的属性の両面の定義を可能にしてくれる。以上のように、複雑な対象要素の定義が、オブジェクト指向思想のもとでは容易化される。

5.1.1.3 相関定義での知識獲得的側面とオブジェクト指向

オブジェクト指向思想における概念体系記述は２面製を持っている。それらは個々にはモジュール化されたオブジェクトが相互に持つ階層関係の記述という静的関係定義、及びあるオブジェクトがある状態になった時に、メッセージ送信によって他のオブジェクトに所定の状態を取らせるという動的関係定義の２機能である。これらによって複雑な（といっても連立方程式に相当する複雑度のものまでは困難であるが）相互関係の定義や変更が容易となる。オブジェクト指向思想の持つ上位概念の変更を下位の全てに波及させることができる点で相関の定義や変更が容易化される。また、旧問題の一部を変更して新問題を作成する場合にも上位概念を変更し、下位から変更後の上位概念を継承することで対応できる点も長所である。オブジェクト指向言語の持つクラス概念は従来言語のマクロに対応するものであり、少量の情報によって複雑な概念体系をインスタンシエイトできるため、相関定義や変更の容易化に寄与する。また旧問題の一部を変更して新問題を作成することも容易である。スロットとメソッドの概念は静的、動的両側面をそなえた概念モジュールの構造的記述を可能化し、相関定義やその変更時のマンマシン性向上に寄与する。メソッドは使い方によってはスロット内の変数の間の拘束関係の自動調整機能の実現など、メタレベルでの知識の記述に寄与する。

5.1.1.4 目標定義での知識獲得的側面とオブジェクト指向

数理計画法の場合には目的関数が、述語論理学の場合には証明したい定理の否定が目標言語となる。オブジェクト指向思想の場合には特に明確な対応物は存在しない。

5.1.1.5 求解過程での知識獲得的側面とオブジェクト指向

複雑な問題を解く場合、旧解を再利用することが多い。解自体が複雑な概念体系となる点や、解の部分的変更に対しては、オブジェクト指向思想における概念体系記述能力やメソッドの利用が有効である。解法自身の生成、再利用、改良能力が実現できれば有効であるが、これには、解法という動的手続き自体の内部構造の明示化、構造化の研究（恐らく強度に分野依存的色彩をおびると思われる）と、明示化された構造のオブジェクト思考的思想での表現が第一歩として重要であると思う。求解過程を単純なルールとして記述することが可能な場合には、ルールの記述文法を明確化し、オブジェクト体系として再表現し、スロットフィリングの方式によってルールの蓄積や保守の容易化をはかることができる。

5.1.1.6 解について

計画／設計型問題解決によって得られる解は、通常は図面等の高度な複雑な情報形態をとる。図面には、要素定義から求解過程に至る全ての段階での情報が盛り込まれており、それを作り変えることは通常、相当のコストがかかる。そこで解の再利用が通常行われる。単なる記憶ではなく、解の知的な保持と検索、メタ知識による解の可塑化すなわち解の自動変更能力が重要である。前者に対しては、オブジェクト指向思想における概念体制記述機能による複雑な解の体系の表現、上位概念継承機能による解表現用情報量の圧縮、クラス概念におけるインスタンスとしての解の生成と保存などが有用である。後者に対しては上位概念クラスに解変更メソッドを保有させておき、これを用いて解を自動的に、かつ制約事項を守らせつつ変更させることなどが可能化される。

5.1.1.7 適応的側面からの考察

製品のイノベーションなど、環境変化に対してＣＡＤシステム自体を適応させゆくことは、広くソフトウェアエンジニアリングの観点からみてもきわめて重要であり、しかも未だ解決されていない問題である。計画／設計型問題解決システムの場合、要素定義から求解過程に至る諸側面についてオブジェクト指向思想の利用が、種々の観点からみて有益であることを考察した。問題解決システムを構成する、要素定義から求解過程に係わる様々な知識をオブジェクト指向概念によって記述することによって、体系の変更や保守が容易なＣＡＤシステムを実現することができると期待される。

古典的数理計画法の世界においては、ベクトル空間内の線形方程式群が要素間の相対関係や目的関数を表し、求解は単体法によって多次元シンプレクスのエッジ上をたどる過程として実現された。対象が水系のように無限連続可分であり、加減算を基本算法としうる場合には数理計画法の本質との相性も良く、成功裡に活用された。しかしながら、より一般的な計画／設計型問題解決においては、対象の要素や相関あるいは求解過程、さらには解までも、一般に複雑な情報体系となるため、これらを受容する器としては、オブジェクト思想は一つの魅力を持っているといえる。

5.1.2 まとめ

線形計画法をひとつのメタフォアとして、合成型問題解決の諸側面を抽出すると、代表的なものとして、図5.1-1 に示す①～⑤があげられる。これらはオブジェクト指向言語の上にそれぞれ、先に考察した様式によってうめこむことができる。合成型問題解決という複雑な対象において知識獲得ということを語る場合には、技術の現状においては、このうめこみϕを、その保守作業まで含めて容易化することと拡

大解釈することが妥当な知識獲得パラダイムとなっていると思われ、今後の発展が期待される。現実世界の合成型問題解決においては、過去においては単純な図形処理ＣＡＤが中心であった。この様相は、分野の差はあるにしても現在においてもよくみかけられる。人間は背後に保持した設計知識の体系によって新図面を作成したり、旧図面の一部を目的に応じて変更することを容易に実現できる。古い自体のＣＡＤは、この背後の常識を計算機な内在化させるわけである。オブジェクト指向思想におけるクラス概念を常識の記述に用い、それらを活用してインスタンスを作ることを設計作業とみなすと、オブジェクト指向思想は高次の常識まで含めた知的ＣＡＤの実現にむかう意味深いステップを実現していると思われる。

5．3　知識表現手法とシステム工学的手法の融合

エキスパートシステムの開発にあたって最も重要な点は、集められた知識を有効に生かしていかに優れた知識としてまとめるかである。知識ベースに収められたこのような知識が正しく機能するためには、その相互の関連性が明確に正しく定められている必要がある。広範な知識を知識ベース化するためには、このように知識相互の関連性をあらかじめ明確な形にまで整理しておくことが不可欠となる。ところが、多様な知識を整理して知識ベース化することに対して有効な技術はほとんどない。

実際的な問題を取り扱おうとすると、問題をトップダウンに与えられたルール、フレームなどの人工知能の概念が定める形式に整理・変換しなければならない。ところが、現在の知識表現手法は、計算機で推論処理を実行するために必要な十分に詳細化された情報を記述する手段を提供しているだけで、人間がそのような形に知識を整理していくプロセスに対してはほとんど配慮がなされていない。

一方、人間の問題解決手法をみると、このように上から概念を与えてその形に知識を収集するかわりに、個々の具体的な問題を整理して、その本質を明らかにするというアプローチも多く見られる。このようなアプローチを体系的に整理した手法は、一般にシステム工学的手法として知られている。ところが、これらの手法は基本的に人間が知識を整理して使用することを前提としており、その整理結果は計算機で直接操作できるほど意味が明確にされていない。したがって、これらのシステム工学的手法はそのままでは知識ベース開発に適用することは難しい。

対象システムに必要な知識の量が比較的少ない場合は、それでも、ある程度強引に人工知能のパラダイムに基づいて知識を獲得し、知識ベースを構築することは可能である。しかし、その際は、いわゆるプロトタイピング手法・探査型プログラミング手法によってシステムのスクラップ・アンド・ビルドを行いながら知識ベースの改善をはからなければならない。ところが知識が膨大な場合はその全体像が捉らえにくくこのような手法をとることは本質的な困難－人間では扱いきれない複雑さ－を含むようになる。このような知識獲得のあい路－計算機向きの知識表現と人間向きの知識整理手法のギャップ－を解消するひとつの方法は、問題解決の元となる情報から情報の整理の枠組みを発見するという、いわば、ボトムアップ型の知識獲得手法の開発である。このような研究はまだ数が少ないが、以下では、［篠原 86］にしたがってＫＪ法に基づく知識ベース構築支援手法について基本的に考え方を簡単に紹介する。

この方法は、ＫＪ法に基づいて、未整理な情報や知識の相互関連を分析していくプロセスを計算機で支援し、徐々に整理される情報を複数の視点から柔軟にアクセスできる方法と、知識の相互関連の意味付けを階層的に構成する方法として提供しながら、最終的には、それを知識ベースとして使えるような形式にまとめるものである。

ＫＪ法は、さまざまな見解をとりまめて、その中から一定の考えを見つけ出すという目的でしばしば利用される。この手法は、事前には相互の関連性が明らかでない群

の中から関連性を見出しい行くという点において有用である。しかし、ＫＪ法では、個々の情報を単一の見方でしか整理できず、情報が全体としてもつ多様な内容をほとんど捨てる結果となってしまう。このような弱点を克服するためには、各情報を多種の関連性に基づいて相互に関係付けることで情報のもつ多様な内容を捉らえる整理手法が必要である。

この整理手法は、基本的に以下の３フェーズからなる。

(i) 元となる情報を表わす項目のグループ化：
元となる情報を表わす項目のうち「親近感」を感じるもの同志を集めてグループ項目としてまとめる。このグループ項目についてもさらにグループ化を繰り返して、小グループ、中グループ、大グループ、……としてまとめていく。

(ii) 項目の間の関係付け：
項目（グループ化によってできた項目も含む）のうち「関係の深い」項目同志を結んで項目間の関係付けをする。

(iii) 項目間の関係の種類についての吟味：
(i)、(ii)で得られた結果から項目を箱で表現し、項目間の関係を入れ子や矢印で表現しいくつかの視点から図（構造グラフ）に描く。これらの図を参照して、そこに扱われている関係の混同や考え忘れている関係を付加し、その上で、「親近感」や「関係深さ」で表わされる関係をより厳密に捉らえなおす。このように捉らえなおされた関係の種類を関係の種類の階層に位置付け、再び、(i)～(iii)の作業を繰り返す。

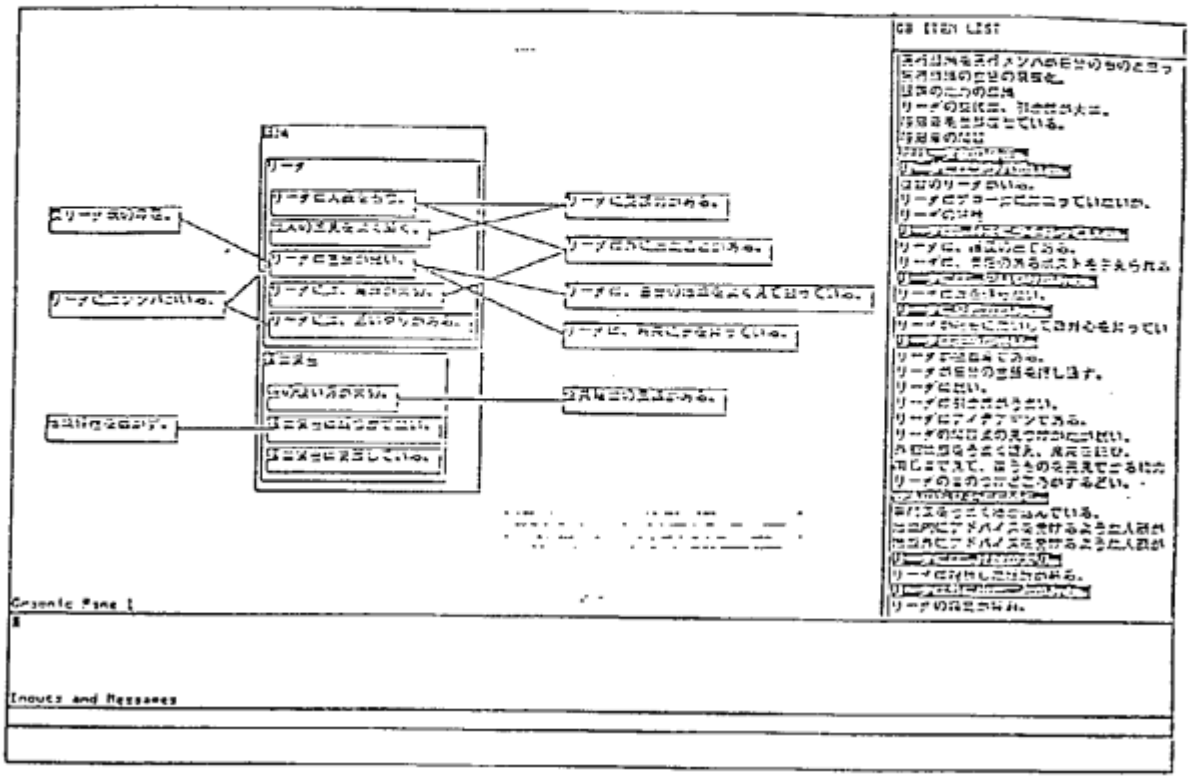
本手法は、多種の関連性を階層的に管理することで、「～するために～する」あるいは「～だから～になる」という（if-then-に近い）関係も因果的関連性として統一して見ることができるなど、多面的な見方を有機的に結びつけることができる。

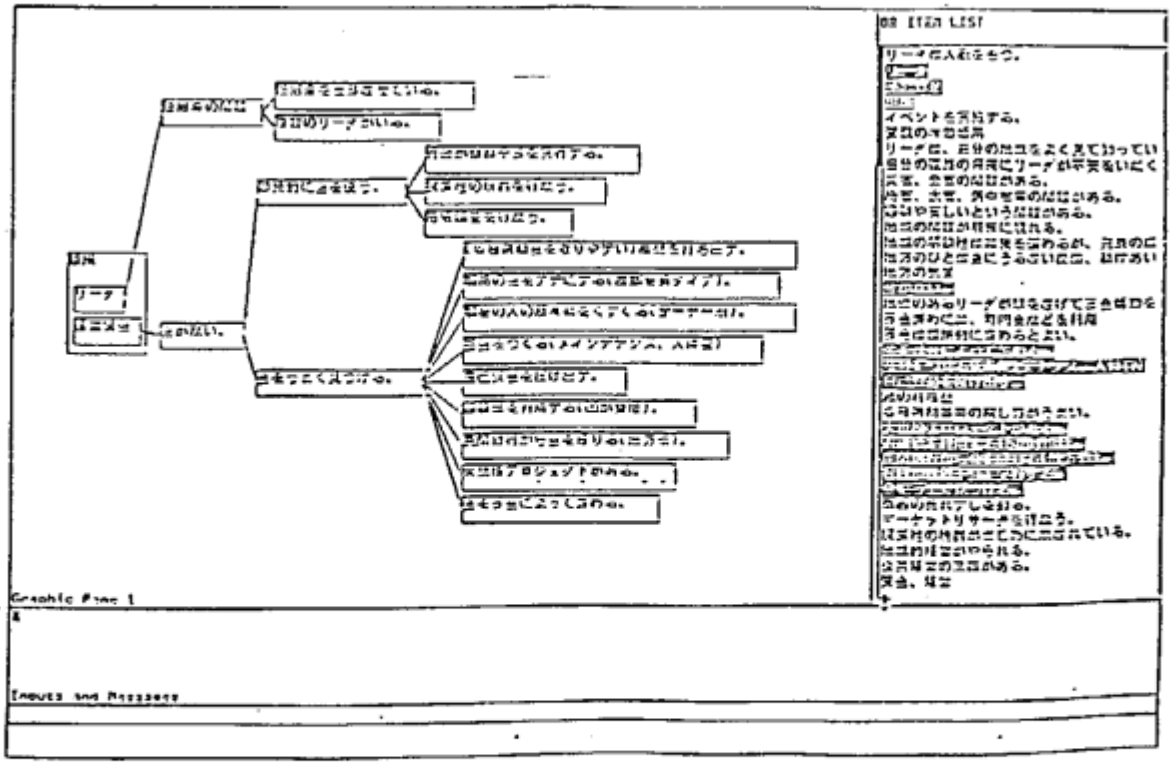
本手法が効果を発揮するのは、整理の過程で、それまでとは別の観点から知識整理を進めようとする場合、さらに、これに伴って新たな情報や関連性の追加を次々とおこなっていくような場合である。この事は、上で述べた３つのフェーズを繰り返し適用することを意味し、その作業は、コンピュータによる適切な支援がなければほとんど不可能である。これを実現したのがＣＯＮＳＩＳＴ（a system for CONstruction Support of Information STructure）である。（図5.3.1）。ＣＯＮＳＩＳＴでは、フレーム表現手法を拡張し、種々のリンクの情報までをフレームシステムで表現し、それを利用者が自由に変更できるようにしている。そのため、これを利用して整理した知識は、比較的容易に通常のフレームやルールなどの知識表現形式に変換することができる。ＣＯＮＳＩＳＴは現在のところ特別な推論機構、あるいは、自動的な知識獲得機能をそなえてはいない。しかし、このようなシステムを高度化していくことで、知識ベース作成上のボトルネックを解消し、優れた知識を組み込んだ知識システムの実現に役立てることができると考えている。

【参考文献】

[篠原　８６] 篠原　靖志、寺野隆雄：“関係の階層を利用した知識ベース構築支援システム．” 信学技報、Ａ１８６－３２．１９８６年１２月．

図5.3-1　知識整理支援システムＣＯＮＳＩＳＴの画面例





5．4　創造的支援、学習

人間のためのより良い道具として、人間の創造的な活動を支援する知識システムも研究されなくてはならないだろう。このような創造性支援の対象となる分野は、ＶＬＳＩ向きのハードウェア・アルゴリズムや3次元ＶＬＳＩの設計といった新しい可能性の領域である。そこでは、既存の知識が蓄えられた知識ベースを参照しながら推論を進めるだけのシステムは通用しない。

ユーザの発想を刺激するようなシステムが最終的な目標である。ユーザ・インタフェースがひとつのキーポイントになると思われる。既存のシステムでも、ユーザ・インタフェースの部分が、推論エンジンや知識ベースを含めた全体のコード量の４２％を占めたという例が報告されている。しかし、単なるインタフェース・プログラムではなく、知識処理をする部分と融合したものが望まれる。

従来のエキスパート・システムの欠点として、それ自身で進歩することがなく、いつでも同じ反応しか示さないため、ユーザが使用していてもすぐ飽きがくることがある。もし人間ならば、熟練した設計者の設計を傍らで観察する機械を得れば、設計について何かを掴むに違いない。つまり、経験を蓄積して賢くなっていく。知識システムの究極の最大目標は、このような学習の問題であろう。これは、知識獲得の面だけでなく、今述べたように、ユーザとのインタフェースの面、さらにノウハウのトランスファを行う際の説明能力の観点からも重要である。

5．5　ＴＥＮＴＡＴＩＶＥな推論

前にも述べたように、設計者は個々の決定の及ぼす影響を正確に予測することができない。従って、設計の可能性をtentative に探求するメカニズムが備わっていることが望ましい。具体的には、適当なところからのやり直しをサポートすること、決定の理由を示す情報（justification ）を保有しておくこと、がある。後者は、ユーザが前に下した決定について忘れてしまった時にも有効であるし、システムがユーザに説明をする時にも役立つ。

5．6　語ることのできない知識と知識獲得

専門家の知的活動を、意識的な逐次処理と無意識の大規模並列処理の組み合せととらえる。従来の知識獲得研究の視点は、専門家が語ることのできる知識を抽出し、それを逐次アルゴリズムとして実現することに偏っていた。いわゆる知識獲得のボトネックは、専門家が語ることのできない知識〔Polanyi 〕〔Kolers〕と並列処理を考慮することなしには解消されえない。

例えば、専門家がある例題を見て、「こういう場合は、Ａである。」と報告した場合を考えてみよう。ここに、Ａは、診断結果、適用されるルール、設計のプリミティブ等である。たいていの場合専門家は、「こういう場合」に対応する適切に一般化さ

れた条件を述べることはできない。これは、この判断が無意識のうちに並列的に行われていることが多いからである。つまり専門家にとって、特定の例題についてどうであるということは易しいが、一般的なルールを述べることは難しい。本報告書においても３．１節の書道の知識にこうした例が述べられている。

知識獲得に関して、いくつかの研究方向がある。

① 専門家の自らの知識の抽出／一般化を対話的に助けるシステム（knowledge engineerの代わり）

- 関係のある属性の発想を助ける
  - 例：ETS (personal construct theory) 〔Boose 84〕
- 属性間の関係付けを助ける
  - 例：ETS (personal construct theory) 〔Boose 84〕
- 似た例を検索する
- 一般化を助ける

② 専門家の示す個々の例に対する判断から帰納的に逐次ルールを学習するシステム

- similarity-based learning
  - 例：ARCH 〔Winston 75〕
  - 例：candidate elimination algorithm 〔Mitchell 83 〕
  - 例：ID3 〔Quinlan 79〕

③ domain knowledgeと一つの例から逐次ルールを学習するシステム

- explanation-based learning 〔DeJong 86 〕
  - 例：カップの構造の記述、
    カップの目的、
    構造と機能を結び付けるdomain knowledge
    から、カップの機能を説明するのに必要な観点を抽出する。

④ 類似の例をみつけ、それを現在のタスクに適用するシステム

- case-based reasoning
  - 例：dynamic memory 〔Schank 82 〕
- analogy

⑤ 現在の例とbest matchの例を選ぶシステム

- memory-based reasoning 〔Stanfill 86 〕
  （並列計算機上で実用的スピードが得られる）

手順

i) 各feature あるいはfeature の組み合わせの生起する回数を数える。

ii) これを使ってmetric (feature の重み×value difference) を求める。

iii) そのmetricで、現在の例とmemory中の各itemとのdissimilarity を計算する。

iv) best matchesを検索する。

特徴

"knows that it doesn't know"

"degrade gracefully"

"ルールは作られない"

"understadable " (connectionist model と比較して)

例：StanfiII & WaltzのMBRtalk (on the Connection Machine)

⑥ 逐次ルールを並列処理に変換して高速化を図るシステム

・chunking in SOAR [Laird 84]

・マクロ・オペレータ

⑦ 専門家の示す個々の例に対する判断から局所的な結合表現を学習するシステム

・connectionist systems のback propagation error learning [Rumelhart 86]

例：NETtalk

⑧ 例から分布的な結合表現を学習するシステム

・learning in Boltzmann machines [Hinton 86]

以上のようなさまざまな研究を、語ることのできない知識に注目して、整理し、マッピングすると以下のようになる。

「外部世界」、「専門家の世界」、「コンピュータの世界」の三つの世界を考える。「外部世界」は、対象とする世界で実際に起こる例や、本に書かれていた知識や、自分以外の専門家の応答を含む。「専門家の世界」の知識は、専門家の頭の中にある（体で覚えているものも含めて）知識全て（語ることのできるもの／できないもの）を含む。「コンピュータの世界」の知識は、コンピュータにインプリメントされた知識全てを含む。

それぞれの世界における知識を「語ることのできる知識（分節できる記号〔Goodman 68〕、共有的記号〔Kolers 86 〕で表現される）←→語ることのできない知識（分節できない記号、個人的記号で表現される）」と「例、エピソード等の知識←→一般化された知識」の観点で分類する。図5.6-1 －図5.6-4 では、三つの世界においてそれぞれ、上半分が語ることのできる知識、下半分が語ることのできない知識、左半分が例、右半分が一般化された知識を表わす。

分節できる記号（articulated symbol）は、文字や音符などのように正確に複製することが可能なものである。分節できない記号（dense symbol）は、絵画のように正確には複製できないものである。共有的記号（consesual symbol）は、自然言語などのように人々の間のコミュニケーションを可能にするものである。これらと、語ることのできる／できない知識とは、正確に対応しているわけではないが、ほぼ上述の関係にあると思われる。

5.6.1 専門家自身の学習

専門家は、語ることのできる／できない、すでに持っている知識を用いて、例と一般化された知識から、語ることのできる／できない知識を学習する。母国語の文法の修得は、語ることのできない知識の学習の例である。また、習熟により、意識的な処理が無意識の処理に変化する、タイピングの習熟はこの例である。

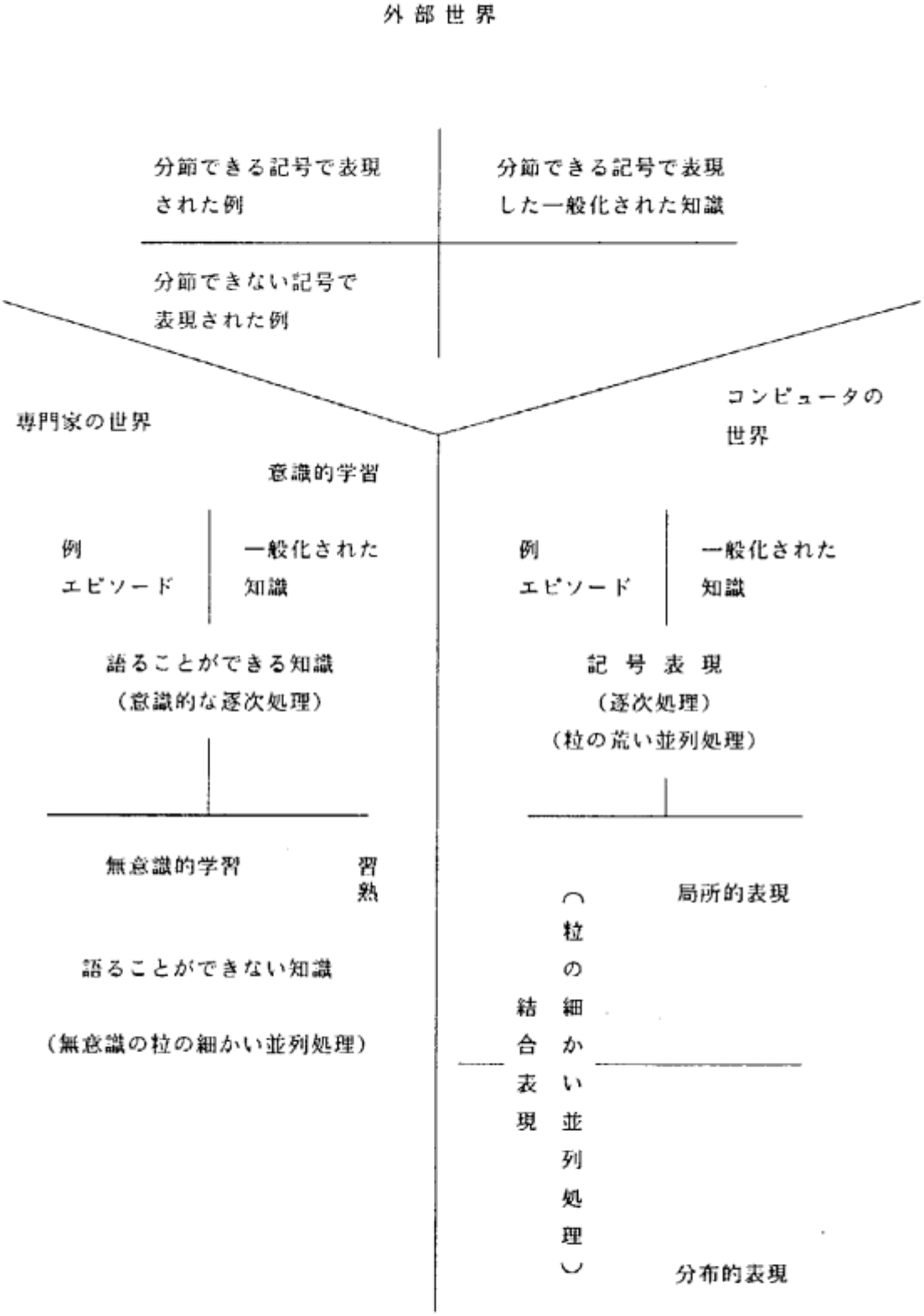

外部世界
分節できる記号で表現された例
分節できる記号で表現した一般化された知識
分節できない記号で表現された例
専門家の世界
コンピュータの世界
意識的学習
例 エピソード
一般化された知識
例 エピソード
一般化された知識
語ることができる知識
（意識的な逐次処理）
記号表現
（逐次処理）
（粒の荒い並列処理）
無意識的学習
習熟
局所的表現
語ることができない知識
（無意識の粒の細かい並列処理）
結合表現
（粒の細かい並列処理）
分布的表現

図5.6-1

5.6.2 計算主義（computationalism）のシステム

専門家が語れる知識と分節できる記号で表現された例からノイマン・マシン上の記号処理システムが作られる。一般化された知識と例の割合により次のように分類できる。専門家が語ることのできる一般化された知識をインタビュー等により抽出して、いわゆるエキスパートシステムが多数作られてきた。一方、例からの帰納推論（上記②）の研究の蓄積も多い。domain knowledgeと一つの例からの学習（上記③）の研究も盛んであり、これといわゆる帰納推論との中間的な方向も模索されている。

よく使われる逐次的処理をひとまとめに変換しようという研究（上記⑥）もあるが、これは、逐次処理→並列処理の変換と関連が深い。

類似の例をみつけ、それを現在のタスクに適用するシステム（上記④）や現在の例とbest matchの例を選ぶシステム（上記⑤）も研究されている。

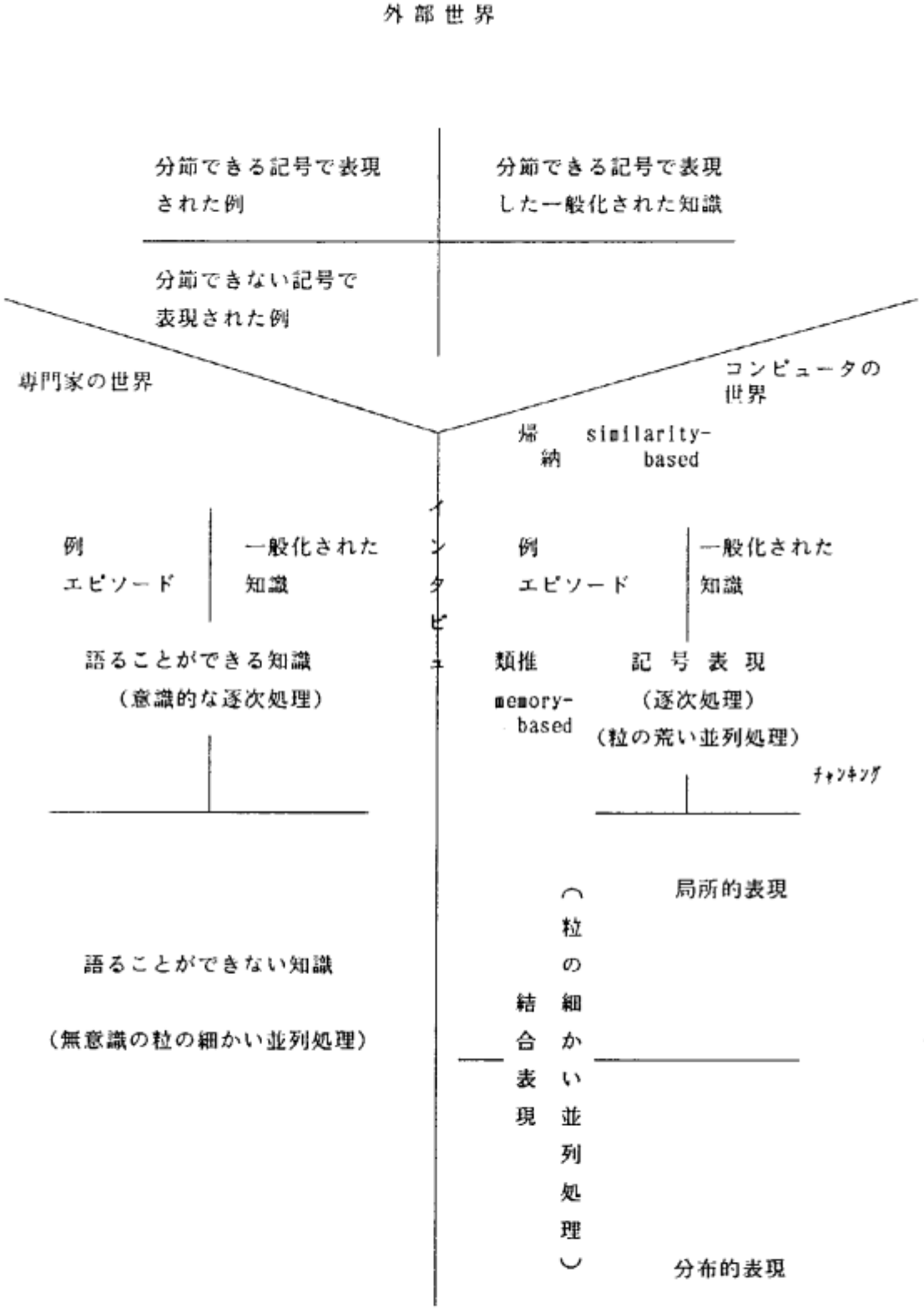

外部世界
分節できる記号で表現された例
分節できる記号で表現した一般化された知識
分節できない記号で表現された例
専門家の世界
コンピュータの世界
帰納
similarity-based
インタビュー
例
エピソード
一般化された知識
例
エピソード
一般化された知識
語ることができる知識
（意識的な逐次処理）
類推
memory-based
記号表現
（逐次処理）
（粒の荒い並列処理）
チャンキング
局所的表現
語ることができない知識
（無意識の粒の細かい並列処理）
（粒の細かい並列処理）
結合表現
分布的表現

図5.6-2

5.6.3 結合主義（connectionism ）のシステム

connectionist model には、局所的な表現をとるもの（上記⑦）と分布的な表現をとるもの（上記⑧）がある。

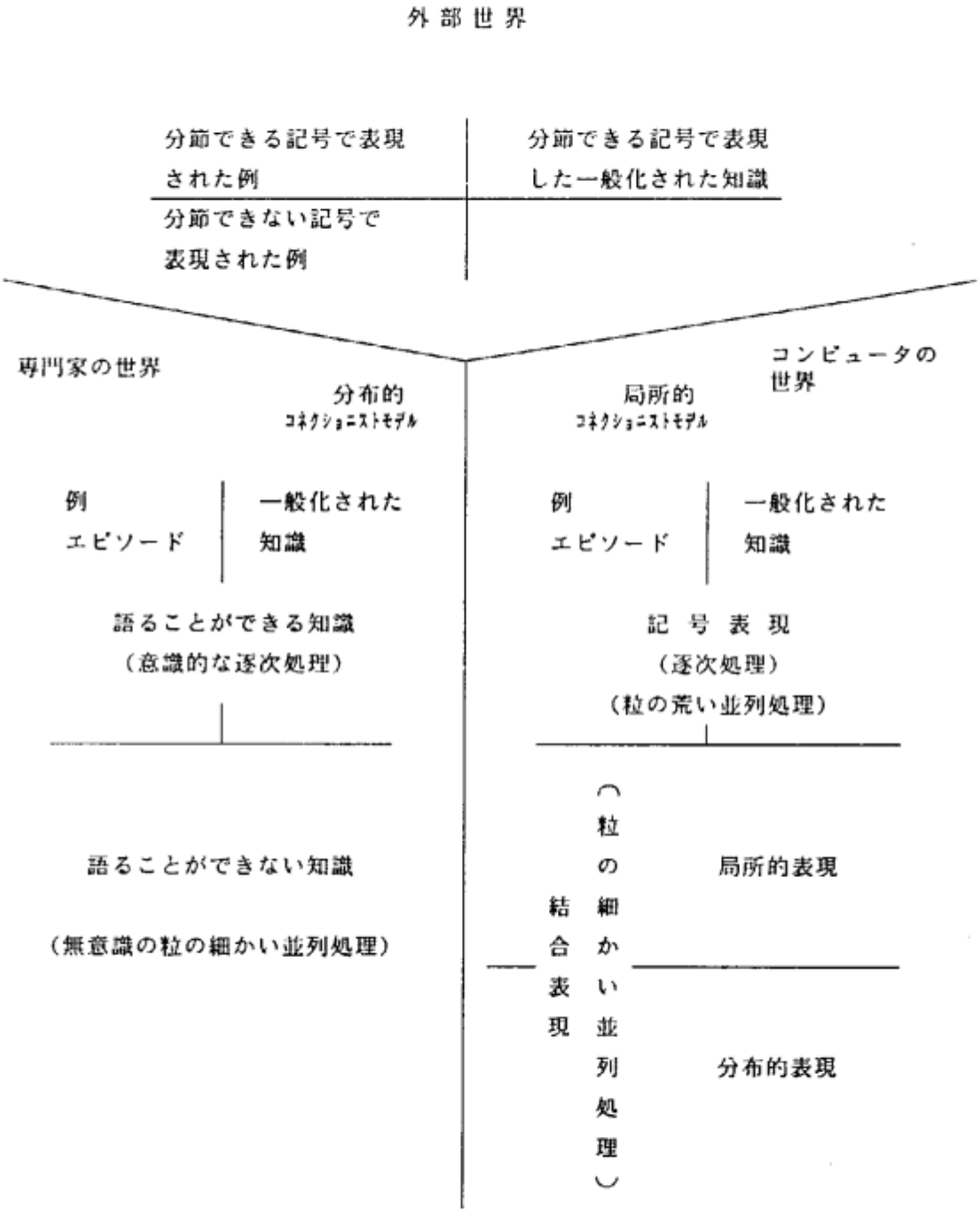


図5.6-3

5.6.4　記号表現と結合表現の結び付け

いくつかの挑戦的な課題がある。

・専門家の無意識の知識の抽出をいかに支援するか。

・connectionist model の動作を記号表現で説明する。

・connectionist model の学習に記号表現された明示的な知識も用いる。

・記号表現のシステムと結合表現のシステムが相互作用を持つシステム

例：パターン情報処理と推論のモデル〔安西　87〕

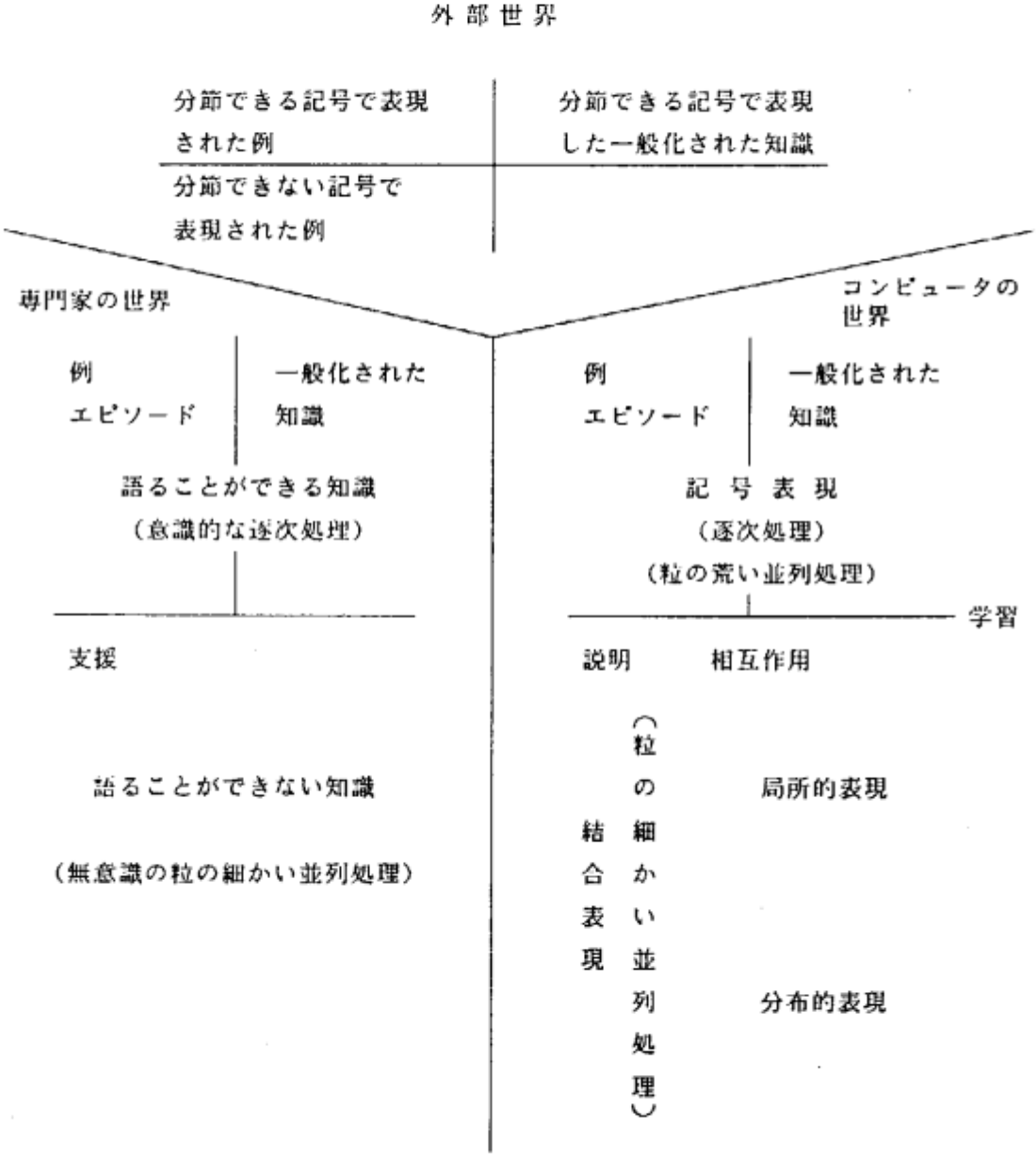


図5.6-4

【参考文献】

〔安西　87〕安西祐一郎、認識と推論の情報処理メカニズム、科学、vol.57 No.4, 1987,210-219.

〔Boose 84〕Boose,J.H.Personal construct theory and the transfer of the human expertise. AAAI-84.1984.27-33.

〔DeJong 86 〕DeJong,G.,and Mooney,R.Explanation-based learning : An alternative view. Mach. Learn. 1.2(Apr.1986),145-176.

〔Goodman 68〕Goodman,N.Languares of art. Indianapolis : Bobbs-Merrill,1968.

〔Hinton 86 〕Hinton,G.E.,and Sejnowski,T.J.Learning and relearning in Boltzmann machines. In Parallel Distributed Processing : Explorations in the Microstructure of Cognition,J.L.McCelelland and D.E.Rumelhart, Eds.MITPress, Cambridge, Mass.,1986.

〔Kolers 86 〕Kolers,O.A.,Smythe,W.E. 佐々木訳、記号操作：心の計算説を超えて、In認知科学の基底、産業図書、1986.

〔Laird 86〕Laird,J.E.,Rosenbloom,P.S.,and Newell,A.Towards chunking as a generallearning mechanism,AAAI-84,1984,188-192.

〔Mitchell 83 〕Mitchell,T.M.et al.Learning by experimentation : Acquiring and modifying problem-solving heuristics,In Machine Learning, Michalski,R.S.et al.,Ed.Tioga,Palo Alto,Calif.,1983.

〔Polanyi 80〕Polanyi,M.佐藤訳、暗黙知の次元、紀伊国屋書店、1980.

〔Quinlan 79〕Quinlan,J.R.Discovering rules from large collections of examples : Acase study. In Expert Systems in the Micro Electronic Age, D.Michie,Ed.Edinburgh University Press,Edinburgh,1979.

〔Rumelhart 86〕Rumelhart,D.E.,Hinton,G.E.,and Williams,R.J.Learning internal representations by error propagation. In Parallel Distributed Processing : Explorations in the Microstructure of Cognition,J.L. McCelland and D.E.Rumelhart, Eds.MIT Press,Cambridge,Mass.,1986.

〔Schank 82 〕Schank,R.C.,Dynamic Memory : A Theory of Reminding and Learning in Computers and People,Cambridge University Press,New York,1982.

〔Stanfill 86 〕Stanfill,C.,and Waltz,D.Toward memory-based reasoning. Communications of the ACM 29.12(Dec.1986),1213-1228.

〔Winston 75〕Winston,P.Learning structural descriptions from examples. In The Psychology of Computer Vision,P.Winston,Ed.McGraw-Hill,New York, 1975.