知的情報処理システムに関する調査研究報告書

第１分冊　計画・設計型知識システムの構築方法論

平成元年３月

JIPDEC

財団法人　日本情報処理開発協会
ICOT-JIPDEC AI センター

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて昭和６３年度に実施した「知的情報処理システムの導入・活用に関する調査研究」の成果の一部をまとめたものであります。

まえがき

通商産業省情報処理振興審議会は、１９８６年３月の答申において、１９９０年代を目標とする第４期「電子計算機利用高度化計画」を取りまとめましたが、この中で、従来のコンピュータ利用技術の延長線上では捉えきれない新しい概念として「知識情報処理」を取り上げています。また、情報処理振興審議会の答申から遡ること４年、１９８２年４月に通商産業省では、世界に先駆けて、知識情報処理機能を組み込んだ全く新しいコンセプトの「第五世代コンピュータ」の開発をスタートさせました。

欧米諸国もこの日本の第五世代コンピュータプロジェクトに刺激を受け、相次いで知識情報処理技術の研究開発をナショナル・プロジェクトとして取り上げています。

一方、産業界でも、各種の知識システムの開発、導入に積極的に取り組み始めていますが、構築方法論が確立されていないことや知識獲得の問題などがあり、知識システムの多くは、未だプロトタイプの段階に留まっています。

このような現状認識に基づき、ＩＣＯＴ－ＪＩＰＤＥＣ　ＡＩセンターでは、昭和６１年度から「知的情報処理システム調査研究委員会」を設置し、学界、産業界で活躍している方々により、３年間にわたって活発な調査研究活動を行い、ここに、全５分冊に及ぶ研究成果を取りまとめました。

本調査の実施にあたっては、委員会の委員の方々はもとより関係省庁ほか関係各方面から御指導・御協力をいただきました。ここに謝意を表する次第であります。

平成元年３月

財団法人　日本情報処理開発協会

会　長　　影山　衛司

「知的情報処理システムに関する調査研究」概要

わが国における知識システムの研究開発の数は 2,000を越えたとみられ，産業界のあらゆる分野への応用の裾野が広がりつつある。しかし，そのうちフィールドテストを経て，実用化の段階に達したシステムの数は，多めに見積もっても， 500以下とみられ，多くのシステムはプロトタイプの段階またはデモンストレーションの段階に留まっているとみられる。その理由として，いくつかの要因が考えられるが，その中で，特に，

1) 知識システムを構築していくことを支援する方法論が確立されていないこと，

2) 知識システムを試作・実装するためのツールが十分には整備されていないこと，

3) 知識獲得を支援する方法論やツールの研究開発が立ち遅れていること，

4) 知識システム技術者が不足し，育成の指針が確立されていないこと，

5) 開発事例を蓄積・分析し，結果を有効に利用する体制が確立されていないこと，

などが指摘されている。このような現状認識に基づき，ICOT-JIPDEC AIセンターは昭和61年9月に「知的情報処理システムに関する調査研究委員会」を設置し，つぎのようなテーマに関する調査研究活動を行ってきた。

1) 知識システム構築方法論を確立すること，

2) 知識システム構築ツールの評価体系を確立すること，

3) 知識獲得支援の動向調査を行い，今後あるべき支援のイメージを提案すること，

4) 知識システム技術者を育成するためのガイドラインを作成すること，

5) 知識システム開発事例の収集・分析を行い，今後の開発の資とすること，

調査研究活動の前半（昭和61年9月～昭和62年9月）では，知識システム構築方法論および知識システム構築ツールの評価に重点をおき，その研究成果は昭和62年9月の成果発表会で報告され，その詳細は「知識システム開発方法論」および「知識システム開発ツールの評価」と題する報告書にまとめられている。

調査研究活動の後半（昭和62年9月～平成元年3月）では，計画・設計型問題の構築方法論，構築ツールの性能評価，知識獲得支援の動向分析，知識システム技術者の育成方法および知識システム開発事例の分析に重点をおいて活動を展開してきた。

調査研究の成果は，以下の5分冊からなる本報告書にまとめられている。

第1分冊「計画・設計型知識システムの構築方法論」では，計画・設計型知識システムの特徴分析を行い，計画・設計型問題には探索的側面，意思決定的側面，事例利用的側面および知識獲得的側面の4つがあることを指摘，各側面に対して，問題解決的接近，仮説推論的接近，

事例ベース推論的接近および学習的接近の必要性と枠組みをまとめている。

第2分冊「知識システム構築ツールの評価」では，システム構築ツールを評価するための評価体系を整備，ツールの性能評価を行うための標準問題を作成，ＡＩセンターオープンハウスに導入されているツールに対し性能評価のテストを行い，評価結果を取りまとめるとともに，ツールの評価に関するいくつかの提言を行っている。

第3分冊「知識獲得支援」では，知識獲得支援ツールの位置づけ・分類を行い，文献調査とヒアリングに基づき，各ツールの機能と特徴を客観的にとりまとめ，知識システム開発事例における知識獲得の実際を分析，知識獲得支援のツールのあり方および知識獲得支援ツールに対する要求事項をまとめている。

第4分冊「知識システム技術者育成ガイドライン」では，知識システム技術者の役割を，知識システムの開発手順の段階に対応して，システムアナリスト，プロトタイピングアナリスト，知識プログラマ，保守・管理技術者の4つに類別した上で，修得すべき技術を人工知能技術，知識工学技術，システム技術の3つに分け，その概要をまとめている。

第5分冊「知識システムの事例」では，生体や社会などの解釈／診断問題事例，プラントや人工衛星などの制御問題事例，スケジューリング，レイアウト，ＬＳＩ設計などの計画／設計問題事例など全部で２３の代表的な事例を取り上げ，比較可能な同一の枠組みで紹介し，さらに各システムにおける基本タスクと問題解決機能を明らかにしている。

以上を要するに，本報告書は知識システムの研究開発においてその解決が緊急とされる課題を重点的に取り上げ，産業界と学会で活躍している技術者・研究者からなる横断的な組織によって調査研究を行った成果である。本報告書がわが国における知識システム研究開発の一層の発展に寄与する資として役立つことを願っている。

最後に，本調査研究を取りまとめる上で，数十回におよぶ会合を通じて，終始，熱心にご協力頂いた委員の各氏ならびにこれを支えてくれた事務局の皆様に感謝の意を表する。

平成元年3月

知的情報処理システム調査研究委員会　委員長　小林重信

昭和６３年度　知的情報処理システム調査研究委員会

委　員　名　簿

委員長	小　林　重　信	東京工業大学
幹　事	寺　野　隆　雄	(財)電力中央研究所
委　員	関　根　史　麿	花王(株)　知識情報科学研究所
〃	菊　地　一　成	キヤノン(株)　情報システム研究所
〃	山　崎　知　彦	(株)豊田中央研究所
〃	森　　啓	日本電気(株)　Ｃ＆Ｃシステム研究所
〃	千吉良　英　毅	(株)日立製作所　システム開発研究所
〃	中　島　俊　哉	富士通(株)　科学システム部
〃	小　林　慎　一	(株)三菱総合研究所
〃	福　島　正　俊	三菱電機(株)中央研究所
〃	森　谷　正　晴	新日本製鐵(株)　君津製鐵所　設備部
〃	桃　井　茂　晴	日本電信電話(株)　ＮＴＴ情報通信処理研究所
〃	畝　見　達　夫	長岡技術科学大学
オブザーバ	生　駒　憲　治	(財)新世代コンピュータ技術開発機構
〃	渡　辺　　敏	新日本製鐵(株)　君津製鐵所　設備部
事務局	平　井　吉　光	(財)日本情報処理開発協会　ＡＩ振興センター
〃	茂　呂　知　明	〃　　〃

類型的タスクワーキンググループ

委　員　名　簿

主　査	小林　重信	東京工業大学
委　員	関根　史麿	花王㈱
		知識情報科学研究所
〃	菊地　一成	キヤノン㈱
		情報システム研究所
〃	森　　啓	日本電気㈱
		Ｃ＆Ｃシステム研究所
〃	千吉良英毅	㈱日立製作所
		システム開発研究所
〃	中島　俊哉	富士通㈱
		科学システム部
ｵﾌﾞｻﾞｰﾊﾞ	生駒　憲治	（財）新世代コンピュータ技術開発機構
事務局	平井　吉光	（財）日本情報処理開発協会
		ＡＩ振興センター
〃	茂呂　　知明	〃

計画設計型知識システムの方法論・目次

1. 序　論 …… 1

1.1 計画・設計型問題の所在 …… 1

1.2 方法論の必要性 …… 3

1.3 報告書の構成 …… 6

2. 計画・設計型問題の特徴分析 …… 9

2.1 計画・設計型問題の分類と特徴 …… 9

2.2 計画・設計の過程のモデル …… 18

2.3 計画・設計型問題の多元的側面 …… 22

3. 計画・設計型エキスパートシステムの基盤技術 …… 31

3.1 問題解決手法 …… 31

3.2. 仮説推論 …… 39

3.3 事例に基づく推論 …… 46

3.4 学　習 …… 57

4. 計画・設計型エキスパートシステムへの接近 …… 67

4.1 問題解決的接近 …… 67

4.2 仮説推論的接近 …… 74

4.3 事例ベース的接近 …… 81

4.4 学習的接近 …… 94

5. 計画・設計型エキスパートシステムの事例分析 …… 103

5.1 配合エキスパートシステム …… 103

5.2 レンズ設計エキスパートシステム …… 110

5.3 ソフトウェア再利用システム …… 117

5.4 回路設計エキスパートシステム …… 126

5.5 ダイヤ編成エキスパートシステム …… 134

6. 結　論 …… 139

6.1 方法論としてのまとめ …… 139

6.2 支援ツールのイメージ …… 140

6.3 むすび …… 141

1．序　論

1. 序 論

1.1 計画・設計型問題の所在

解析問題と合成問題

知識システムが対象とする問題は，解析問題(Analysis Problem)と合成問題(Synthesis Problem)に分けられる．

解析問題の目的は，システム構造(System Structure)および構成要素特性(Components Properties) が与えられたとき，システム特性(System Property)を推測することにあり，解釈(Interpretation)，診断(Diagnosis)，制御(Control)などが解析問題に類別される．解析問題では，システム構造と構成要素特性が事前に設定されているので，環境が所与とすれば，システムの特性は，一般に，一意に定まる．

設計問題の目的は，要求仕様としてシステム特性が与えられたき，これを実現するようなシステム構造と構成要素特性を決定することにあり，計画(Planning)，設計(Design)などが合成問題に類別される．合成問題では，システム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特性は，一般に，無限に存在する．

解析問題でも，例えば複合故障診断問題や定性推論問題のように，少し問題が複雑になってくると，組み合せ爆発(Combinatrial Explosion)の問題が生じる．これに対し，合成問題では，組み合せ爆発の問題を内包しており，この問題をどう克服するかが本質的な課題である．

合成問題に対する基本的なアプローチは，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，システム構造と構成要素特性を仮に生成（決定）し，システム特性を検査（解析および評価）を行うことを繰り返して，より良いシステムを探索していく方法をいう．生成検査法は，解析による合成(Synthesis by Analysis)とも呼ばれ，この意味で，合成問題は解析問題を内包しているといえる．

計画・設計型問題の特徴

計画や設計の問題は，達成すべき目標が抽象的な（すなわち機能的な）記述で与えられたとき，これを実現可能とするような具体的な（すなわち操作的な）記述に変換する多段の問題解決プロセスとみなすことができる．

計画や設計の企画のように，問題解決プロセスの抽象度が高ければ，一般に，計画・設計の担当者には勘やひらめき，センスといった創造性の能力が要請される．また，詳細計画や詳細設計のように，問題解決プロセスの具象度が高ければ，計画や設計の担当者には手続き的な作業を忠実に遂行する定型性の能力が要請される．

計画・設計作業はつぎのような基本的なタスクに分解される．

①仕様の記述(Description of Specification)：
　計画・設計されるべき対象に関する要求仕様の記述

②目標の構造化(Structuring of Goals)：
　目標間の関係（目標－副目標の関係，優先関係，トレードオフなど）の構造化

③詳細化による代替案の生成(Generation of Alternatives by Refinement)：
　目標間の関係に従い，要求仕様が詳細化された代替案の生成

④代替案の解析(Analysis of Alternatives)：
　代替案の特性（機能や性能）の解析

⑤代替案の評価／検証(Assessment/Verification of Alternatives)：
　代替案の機能や性能の評価および検証

⑥代替案の修正(Modification of Alternatives)：
　要求仕様を満たすように代替案の修正

⑦代替案の選択(Choice of Alternatives)：
　最適性または満足性の基準に基づく選択

⑧選択案の正当化(Justification of the Choice)：
　選択された代替案の正当性の根拠づけ

計画・設計問題に対し，知識システム的接近を行うためには，まず，①対象の記述と②目標の構造化は，基盤として，必要である．

知識システム的接近の中心的課題は，③代替案の生成，④代替案の解析，⑤代替案の評価／検証および⑥代替案の修正，にある．③から⑥のサイクルにおいて，計画・設計の担当者がどのような形で係わりをもっているかを明らかにすることが必要である．すなわち，担当者の問題解決プロセスを分析することは知識システム的接近を行う上で重要な役割を占める．

1.2 方法論の必要性

これまで，計画・設計問題に対する接近の多くは，定型的な計画・設計問題を中心として，問題に対する十分な分析がなされないままに，探索すべき空間を限定し，最適に近い解を効率よく見出すのに有効と思われる経験則を集め，これを利用する方法が取られ，しかもシステムの構築が場当り的または試行錯誤的になされてきたといえる．このような接近法は，プロトタイプの作成段階では有用であっても，システムの維持・拡張を考えた場合適切な方法とはいえない．

計画・設計問題への組織的な接近を考える上で，計画・設計問題のクラスおよび問題解決プロセスという視点に着目した接近法が有用と思われる．

計画・設計問題のクラスに着目した接近法

もっとも一般的な設計問題をクラス１の問題と呼ばれる．クラス１の問題は，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．

［計画・設計問題（クラス１）］

optimize　〈システムの構造〉

subject to

optimize　〈構成要素の特性〉

クラス１の計画・設計問題は〈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．すなわち，〈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，〈構成要素の特性〉が事前に決定されていることを必要とする．

〈構成要素の特性〉が所与の計画・設計問題をクラス２の計画・設計問題と呼ばれる．

［計画・設計問題（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．

＜システムの構造〉が所与の計画・設計問題をクラス３の計画・設計問題と呼ばれる．

［計画・設計問題（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計画・設計問題などがこのクラスの問題に含まれる．

各クラスに対する基本的な接近法および問題解決に必要とされる類型的タスクは，次表のようにまとめられる．

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 制約伝播と構造最適化
クラス　1	事例ベース推論	特徴づけ，類比照合 修正および修復

なお，この項の詳細な議論は，[小林(1988)]および[AIセンター(1987)]を参照されたい．

設計者の問題解決プロセスに着目した接近法

計画・設計のサイクルを実行する上で，計画・設計者の問題解決プロセスに着目すると，つぎのような多元的な側面が観察される．

①探索的側面：

既に指摘したように，計画・設計問題に対する基本的な接近法は，生成検査法にある．多くの計画・設計問題では，代替案を何らかの方法で生成して，解析・検査することにより，はじめて代替案の適切さが確定するのが普通であり，したがって適切な代替案の追及は探索問題に帰着する．探索空間が小さければ網羅的な探索が可能であるが，探索空間が大きくなるにつれて，探索の効率を向上させるために，問題解決戦略を導入したり，可能性の低い代替案の探索を排除するための枝刈り規則を利用することが必要となる．問題解決戦略や枝刈り規則の導入は極めて問題依存的である．計画・設計問題の多くはＯＲ問題として定式化した場合，ＮＰ完全となるため，計算量の壁を克服するため，新しいアプローチの開発が望まれているのが現状である．

②意思決定的側面：

ＯＲ問題として定式化される探索問題では，代替案の生成に続いて検査が行われるので，もしも不適切であれば，その代替案は直ちに撤回されて，後戻りを行うことにより，べつの代替案を探索することが試みられる．しかし代替案を生成しても，それが適切であるかどうかを判定することが，その時点で不明の場合には，代替案を仮に選択することが必要になる．この場合代替案の選択は意思決定の問題となり，選択された代替案は仮説として扱う必要がある．仮説に基づいて導き出された推論結果は，やはり仮説であり，依拠した元の仮説が矛盾を生じないことが確認されている範囲内で正当化されるものであり，仮説間の依存関係の維持と矛盾が生じた場合には，依存関係に基づく後戻りを必要とする．仮説に基づく推論を形式的に取り扱う枠組みは意思決定的側面を支援する上で有用である．

③再利用的側面：

人間の問題解決においては，過去に経験した成功および失敗の事例は有効に利用されていると考えられる．計画・設計問題においても同じようなことがいえる．すなわち，計画・設計の過程とは成功・失敗の経験に基づくノウハウの蓄積と利用に他ならないとする見方も成立する．全く新しいタイプの計画・設計問題に直面するのはむしろ稀であり，新しい計画・設計問題が与えられたとき，これと類似の計画・設計事例を検索し，類似事例を

参考にしながら，問題解決の効率を向上させることは極めて普遍的行われていることと考えられる．事例の利用には，成功事例を修正して利用する場合と失敗事例を参照して同じような過ちを回避する場合の２通りが考えられる．

④知識獲得側面：

類似の事例が利用できない全く新しいタイプの問題においては，計画・設計者は問題解決に有用な知識を自ら獲得していくことが必要になる．新しい知識の獲得が全くの偶然によって得られる場合もあるが，一般的には，知識の獲得は多数の正例および負例の背後に成立する法則や原理を帰納的に導くことによってなされるか，既に獲得されている知識を使って新しい事例を説明しかつこれを一般化することにより，演繹的に導くことによってなされるかのいずれかとみられる．事例からの知識の獲得は学習の問題に密接に関係する．

本報告書では，計画・設計者の問題解決プロセスに観察される多元的側面に着目して，各側面を支援するようなＡＩの基盤技術を中心に，計画・設計型知識システムの構築方法論を論じることとする．

1.3 報告書の構成

本報告書は，全６章から構成されている．

第１章は序論である．

第２章「計画・設計問題の特徴分析」では，計画・設計問題を２つの視点，すなわち，システムの構造と構成要素特性による特徴分析および開発フェーズの違いによる特徴分析を行った上で，計画・設計プロセスの基本サイクルを考察し，このサイクルの中で計画・設計者の問題解決プロセスには，探索，意思決定，事例利用，知識獲得などの多元的側面があることを指摘している．

第３章「計画・設計型エキスパートシステムの基盤技術」では，計画・設計者の問題解決プロセスの各側面を支援する基盤技術として，探索的側面に対しては問題解決手法が，意思決定的側面に対しては仮説推論が，事例利用的側面に対しては事例ベース推論が，知識獲得的側面に対しては学習が，それぞれ重要な役割を担うとして，各技術の現状と課題などを論じている．

第４章「計画・設計型エキスパートシステムへの接近」では，問題解決，仮説推論，事例ベース推論および学習の各基盤技術に基づく接近を行うために，各モデルの同定および問題の定式化について論じている．

第５章「計画・設計型エキスパートシステムの事例分析」では，前章で述べた各接近法の有用性を示すために，５つの計画・設計問題を取り上げ，適用の実際および適用の可能性を試み，考察を行っている．

第６章「結論」では，方法論のまとめを行い，この方法論をベースとして，計画・設計問題領域での問題解決を支援するシステムのイメージを提案している．

参考文献

[小林(1988)]

小林重信：知識システムにおける類型的タスクに関する調査研究，新世代コンピュータ技術開発機構

[AIセンター(1987)]

知的情報処理システムに関する調査研究報告書：知識システム開発方法論，ICOT-JPDEC AIセンター

（１章担当 小林重信）

2．計画・設計問題の特徴分析

2. 計画・設計型問題の特徴分析

2.1 計画・設計型問題の分析と特徴

回路設計，機械設計等のような領域の設計の現状を考えると，一般的に使われているものは，各種図面図入力のためのエディタ，解析的にシステムの性能を知るシミューレータ，配線，配管等のルータ等といった機械的作業や数値演算を支援するものに限られており，”設計”本来の知的な活動を代替するものを，あまり見いだせないのが実状である．したがって，設計作業そのものを代替するという観点から，設計問題を分析する必要もあると思われる．本節では，設計対象と設計の流れの２面から計画設計型問題を分類し，その特徴を考えてみる．

2.1.1 システム構造と構成要素特性による分類と特徴

設計対象システムから見た場合，計画設計型問題は，そのシステムの特性が与えられたとき，これを実現するようなシステムの構造とサブシステムの特性を決定する問題と捉えられる．この定義から計画設計問題は，３種のクラスに分類することが可能で，それぞれのクラスごとに特徴的な汎化タスク(generic task)が存在すると考えられている［小林他87］．ここでは，まずそれぞれのクラスの定義を述べ，次にそれらクラスの特徴と汎化タスクを以下にまとめる．

2.1.1.1 計画設計型問題のクラスとのその定義

計画設計型問題は，設計対象システムから，以下の３つのクラスに分類される．

（クラス１）

”システムの特性”が与えられた時，”サブシステムの特性”および，それを構成要素とした”システムの構造”を求めること。

（クラス２）

”システムの特性”と”サブシステムの特性”が与えられた時，それを構成要素とした最適な”システムの構造”を求めること。

（クラス３）

”システムの特性”と”システムの構造”が与えられた時，最適な”サブシステムの特性”を求めること。

以下に，これらのクラスごとに計画設計型問題を分類し，その特徴を分析していく．

2.1.1.2　計画設計型問題クラス２の特徴

まず，クラス２のエキスパートシステムの例としては，コンピュータルームレイアウト，プリント基盤の配置配線設計，ＬＳＩマスクのレイアウト設計等といったフロアプランに関する設計問題がある．また，時間も多次元空間の一次元と見なすことで，製品の組み立てのための工程設計，ジョブショップスケジューリング，各種運行ダイヤ編成等といった計画型の問題も空間配置に関する設計問題に帰着し，このクラスへ分類可能となる．

これらの問題の特徴としては，

①基本的に，ある空間へどのサブシステムを置くかという探索問題であり，生成検査法の枠組みとして取り扱うことが可能である．しかしながら，サブシステムの空間的配置により出来るシステム構造の探索空間は，一般に膨大であり組合せの爆発を生じる．

②また，フロアプランの設計の様に，配置の終盤にこないと配置可能性が評価出来ない問題もある．したがって，設計作業においてシステム構造の評価が遅くなりがちで，効率良く探索空間が刈り込めない場合がある．

③したがって，戦略としてトップダウン精密化（topdown　refinement）［Chandrasekaran 86］のような，汎化タスクを使うことが有効である．これは，まず要求されているシステム特性によって強い制約をうける部分の構造を優先して決定し，この部分構造を新たに制約条件へ付加する．更に，システム特性によって次に強い制約をうける部分の構造を決定していく，これを次々繰り返しながら最終的なシステム構造を求めるものである．

④このような方法でシステム構造の探索が成功しても，種々の構造から最適なものを選択するための評価は難しい．例えば，配置問題の場合では，空間利用度による評価といっ

た単純な評価尺度による評価選択は困難である．したがって，最適な配置を行なうための，領域固有の知識も必要とされる．

2.1.1.3　計画設計型問題クラス３の特徴

クラス３の設計は，既存システムの仕様と要求仕様の間に比較的大きな相違がない場合に行われることが多い．要求仕様に満たすシステムの実現は，既存システム構造を前提とし，この構造内にある各サブシステムの特性パラメータをチューニングすることでなされる．特に，この様なパラメトリック設計は，レンズ設計，機械設計，回路設計等の中の定型的な設計部分のためのエキスパートシステムとして多く見られる．

これらの問題の特徴としては，

①システム構造が与えられているので，このシステム構造を階層的に表現し，上位構造から下位構造への順に，インスタンスを生成検査することで設計出来る．したがって，計画選択・精密化による階層的設計（Hierachical design by Plan Selection and Refinement）［Chandrasekaran 86］という汎化タスクによるアプローチが有効である．

②システムの骨組み構造の様に，抽象レベルの高い階層のサブシステムの特性パラメータとシステムの特性パラメータ間の関係は，定式化することが可能な場合が多い．したがって，これら抽象レベルの高い階層におけるサブシステムの特性は，システムの構造とシステムの特性から直接導出出来る．しかしながら，要求仕様として与えられるシステムの特性パラメータの種類は，通常設計回毎に変化するので，これらのそれぞれのバリエーションについて効率的にサブシステムの特性パラメータを導出する知識が豊富に必要とされる．

③実装部品レベルの様に，抽象レベルの低い階層のサブシステムの特性パラメータとシステムの特性パラメータ間の関係は，一般に非線形性が強いので，解としてのサブシステムの特性パラメータ値は多数存在する．したがって，これらの中から，システム性能等を考慮した最良な解を選択する必要がある．このため，まずサブシステムの特性パラメータ値へ過去の経験等から暫定値を与え，その上でこれらサブシステムの特性パラメータ値から解析的にシステムの特性パラメータの値を知る．このシステムの特性パラメータが要求仕様に近づく様に，サブシステムの特性パラメータ値を補正しつつ設計を行なう．

このように，システムを設計するためには，サブシステムの特性からシステムの特性を

知る解析に関する知識や，サブシステムの各特性パラメータの暫定値や補正の仕方をどの様にするかという戦略的知識が，探索を効率化する上で重要である．特に，あるサブシステムの特性が，より上位階層のサブシステムまたはシステムに与える影響の定性的予測評価のタスクの必要性は，状態抽象化（State Abstraction）［Chandrasekaran 86］と呼ばれる汎化タスクとして指摘されている．

④システムやサブシステムの特性パラメータは，連続値や多値の離散値をとることが多い．連続値は離散的に取り扱われ，組み合せ問題に置換される．また，例えばデータブックや各種設計標準等といった公的知識の活用によって，サブシテムの特性パラメータの範囲を制約し，探索範囲を刈り込むことも行われる．

2.1.1.4　計画設計型問題クラス１の特徴

クラス１は，要求されたシステムの特性が新規性の強い場合に多い．典型的な例として，機能仕様からのシリコンコンパイラの試みの様なものがあるが，まだ実用的レベルに到達していない．この理由は，新規な設計においては，設計者の創造性や決断力といった能力に依存するところが強く，その設計過程に未知の部分が多いことによると考えられる．

しかしながら，要求されたシステム特性に対して，システム構造を決定出来る範囲は制限されているが，クラス２のパラメトリック設計のシステムにおいても，要求されたシステム特性に近い過去のシステム構造をプロトタイプに選んだ上でパラメトリック設計を行う様になっていたり［溝口他　83］，クラス１のＬＳＩのレイアウトではサブシステムとしての配置するトランジスタをパラメトリックな形状として取り扱うことで，このクラスの要素を持っている．この様に，クラス２・クラス３に分類されている実用的なエキスパートシステムにおいても，システムの特性・サブシステムの特性の両面において最適化を図るためのさまざまな仕掛がなされているのが実状である．

このクラスの計画設計問題の特徴はすでに述べた様に未知の部分が多いが，一応その特徴を述べる．

①新たなシステム構造を決定するため最も多くとられると思われる方法は，過去の複数の設計事例を参照し，これらの流用可能な部分構造の部分の組み合せによって，システム構造を決定することである．この際，各設計事例のシステムの特性とシステム構造間の関係の仮説が立てられ，その上で要求されているシステムの特性によって必要な過去の設計

事例のシステムの部分的構造を切り出し，新しいシステムの構造を作り出す．

②この様に，基本的なシステムの構造を一応設計した上で，要求されたシステム特性要求が従来より厳しくなる場合は，等価的変換理論によってシステムの部分的構造の分割や統合が行われる．特に，システムの特性の要求として機能・性能向上が大きいときには，設計自由度を増加させる必要からその特性に関連の深いシステムの一部構造が適度に分割される．また逆に，コストを下げるときには，逆にシステムのある部分の構造が統合される．

③サブシステムの特性の決定は，クラス３のパラメータチューニングのように，過去事例のパラメータ値を参考にした暫定値から，パッチワークによって決定する．

④設計当初の設定したシステムの構造は，あくまで設計仮説である．このため，サブシステムの特性パラメータを決定した上で，要求されたシステムの特性が満たされない場合は，再度システムの構造に戻って再設計を行う知的バックトラック機構が必要とされる．

2.1.2 開発フェーズの違いによる分類と特徴

次に設計の流れから，計画設計問題の特徴を考えてみる．従来，設計の流れの上から見て，あるＣＡＤシステムが有効に使われるのは，設計作業ごく一部のフェーズだけであることが示している様に，設計フェーズ毎に要求されている能力や作業内容は異なっていると考えられる．したがって，ここでは設計の初期・中期・後期の３フェーズに分類し，こういった面から計画設計型問題を分析し，その特徴を述べる．

2.1.2.1 初期設計フェーズ

いわゆる方式，機能設計フェーズである．このフェーズでは，製品企画段階によって決められた機能や性能等のシステムの特性に基づき，設計可能性・開発日程・概略コスト等が見積得るレベルのシステムの概略構造を求めるため，システムをサブシステムへ分解し，個々のサブシステムの特性を求める．したがって，この設計フェーズの目標は，設計対象のモデル化理論に基づいて考えられる骨組み構造の候補の決定である．

この骨組みは，すでにクラス１やクラス２で述べたように，過去の設計事例の骨組みやその事例の設計時に得られた設計経験に基づき，特に要求仕様内の機能条件を充足する様に設計される．この様に設計された骨組みは，従来の設計経験に基づき，開発工数，コスト見積，性能が評価可能である．種々の骨組みについてこれらの評価を行い，以降の設計対象となる骨組みの選択を行う．この設計フェーズで決定される骨組みは，一般にそれ以降の設計の容易さや設計品質の良さに大きく影響するので，比較的多数検討され，その中から幾つかを選択するのが通常である．

このフェーズでのタスクの特徴は，

①要求されたシステムの特性による過去の設計情報（設計対象，工数，コスト等）の検索・参照

②要求されたシステムの特性のうち，主として機能に関するパラメータから，骨組みの特性パラメータを決定する明示的かつ効率的な設計手順基づく設計

③設計された骨組みから，機能・性能等の解析的手法による評価や過去の事例に基づく工数，コスト等の推定

④複数のプロトタイプシステムの設計可能性の並列的評価

等といったものが考えられる．特に，定型的設計分野においては，①②③については有効なアプローチとして属性モデリングの考え方が使われており，初期設計段階でのより正確なコスト見積や最適な骨組み選択を実現しようとしている［philips 87］．④に関しては，最適な骨組みを選択するために仮説推論の技術の適用が試みられている［宮崎 88］．②の効率的な設計手順は，要求仕様の与えられ方で一般に設計の度ごとに変わる．したがって，システムの構造，各構造間の動作，設計効率化のための経験的ノウハウに関するより深い知識からこれを発見するという試みである，知識コンパイルという手法が研究されている［Araya 87］．

2.1.2.2　中期設計フェーズ

いわゆる詳細設計フェーズである．初期設計フェーズで決定された骨組みを有するシステムを具体化するため，この設計フェーズでは数多くの実装部品の特性パラメータを決定しなければならない．したがって，設計全般としては横型探索による並列的検討と言うよりは，むしろバックトラックを利用した深さ優先探索である．

このフェーズでは，要求仕様中の性能条件で示されたシステムの特性を実現させること，使用可能な実装部品の特性の制約の中でコスト制約を満足させること，更にはこのような要求仕様を満足するのみならず，性能とコスト間や各性能パラメータ間の最適なバランスを追求すること，といった課題が課せられる．これらの実現においては，単に過去の設計経験に基づく知識だけでなく，設計者のセンスや決断力に依存するところも大きい．

また，このフェーズでは，設計図の入力支援，設計されたシステムの性能の解析的シミューレション，性能向上のための非線形最適化等といった多くの計算機支援が存在する．したがって，これら効果的なツールの使用が，開発効率や品質に，大きく影響を与えている．

このフーズでのタスクの特徴は，

①構成部品のカタログ等の公的知識とその参照

②サブシステムの機能を実現するサブシステム構造の知識に基づく設計

③性能条件やコスト条件を充足させるための設計ノウハウによる設計変更

④解析的手続きに基づく性能等の評価

⑤バックトラックによる深さ優先の設計

⑥コスト性能比性能間のバランスの最適性と設計困難時の”諦め”等の意志決定

⑦計算機支援ツールの使用

といったことがある．特に①④⑦については，エキスパートシステムがＤＢやＣＡＤとのインターフェース機能を有してきており，既存システムと統合化のための環境が整いつつある．また，⑥のコスト性能比，性能バランス等にかかわる意志決定は，単に工学的側面とはいえない面をもっているため，本来の意味で設計を代替するエキスパートシステムを作る上では，大きなボトルネックとなっている．

2.1.2.3　後期設計

生産設計のフェーズであり，設計されたシステムを製造機や品質基準等の都合に適合させるため，詳細設計で得られた機能や性能を大幅に変えない程度で設計変更を行う．これには，具体的に冗長部品の排除，企業毎の標準部品への置き換え，部品配置や支持のための部品変更，組み立て作業や調整作業のための部品の追加変更等といった設計作業がある．このように，この生産設計フェーズでは，企業毎の製造上ノウハウが特に重要な関わりを持っている．この様に多くの制約下において，比較的単純な繰り返しによる設計の修正が行われる．したがって，機械的作業のわりには細心の注意を要し，比較的工数のかかる設計作業も多い．

したがって，このフェーズでのタスクの特徴は，

①標準部品の制約等に関する公的知識による設計変更

②システムの形状寸法上の制約による設計変更

②製造機に関する知識による設計変更

③組み立てや調整に関する知識に基づく設計変更

等といったことがある．

［参考文献］

[小林その他　87]小林　重信他：知的情報処理システムに関する調査研究報告書
知識システム方法論，(ICOT-JIPDEC AIC,1987)

[Chandrasekaran 86]B.Chandorasekaran:Generic Task in Knowledge-based Reasoning:
High-level Building Block for Expert System Design,
IEEE Expert, Fall, 23-30(1986)

[Araya 87]A.A.Araya and S.Mittal:Compilig design plan from descriptions of
artifacts and problem solving huristics,
IJCAI-87, 552-558(1987)

[浅野　85]溝口　文雄他：　レンズ設計，ＡＩテクノロジー第３章，25-47(オーム社,1986)

[phillps 87]R.E.phillips and L.W.Rosenfeld:A Knowledge-Based System for
Design Automation, 6th international Symposium on
Offshore Mechanics and Artic Engineering(1987)

[宮崎　88] 宮崎　俊彦他：　人工知能を応用した論理回路変換システムの開発，
第２回人工知能学会全国大会６－２３，325-328(1988)

（2.1節担当　菊地）

2.2　計画・設計の過程のモデル

モデル化にあたっては，①まず該業務がどのような部分業務から構成されるかを明らかにし，②その後これを基本的なタスクに分解し，モデル化した．

なお，計画・設計の部分業務には設計・計画対象に依存した各業界，あるいは該会社固有の呼び方があり，全てを網羅することはできない．そこで，ここでは部分業務よりタスクレベルに近い中間段階（計画・設計マクロステップ）を想定した後，モデル化を試みた．

2.2.1　計画・設計と他業務との関係

計画・設計の入力と出力を明らかにしておくために，これら業務の上流と下流の業務との関係を図2.2－1に示す．

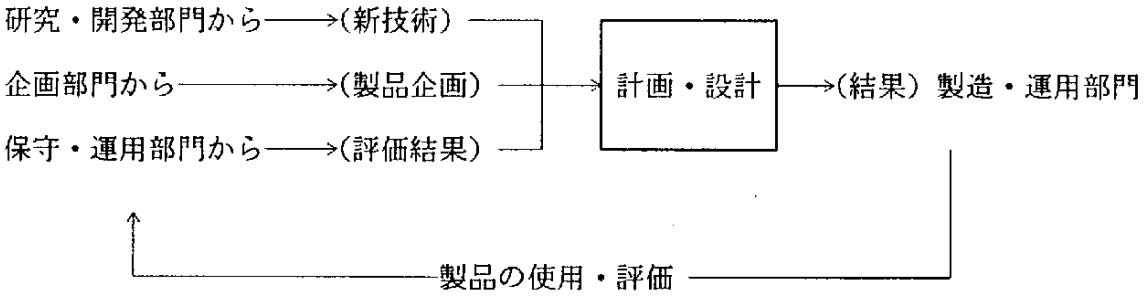


図2.2－1　計画・設計と他業務との関係

図2.2－1に示すように直前，直後の業務のみでなく，間接的な他の業務との間にも評価というような依存関係があるため，モデル化にあたっては，これら前後の業務に含まれる計画・設計に関連した業務をも含めることとした．

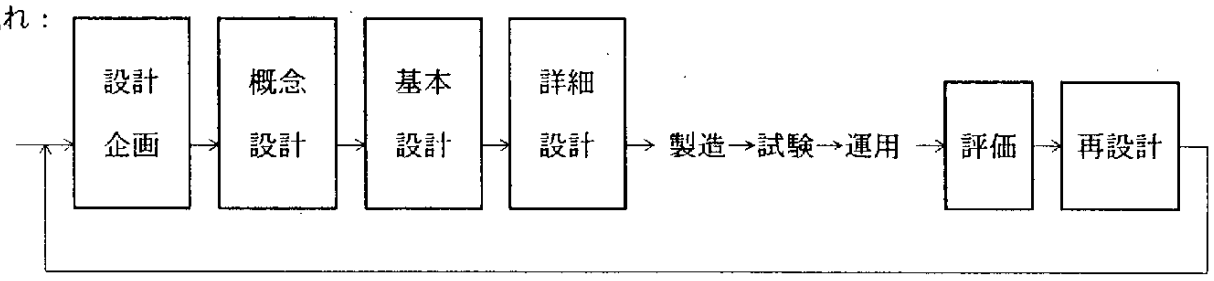
なお，計画・設計に関してはまったく新規に実施する場合と，既存の結果を一部手直しする場合とがあるが，決定すべき機能・詳細仕様の種類や量の差はあるものの，業務の実施内容そのものは同一と想定した．

2.2.2　計画・設計の流れ

図2.2－2に計画・設計の工程順に従った計画・設計マクロステップでの流れを示す．

なお，業種によりこれらの呼び方が異なるので理解を深めるため，例として論理ＬＳＩ設計，ソフトウェア設計での業務概要を併記した．

流れ：



用語

ﾏｸﾛｽﾃｯﾌﾟ	概要	論理ＬＳＩ開発	ソフトウェア開発
設計企画	・企画の分析 ・課題の設定	・外部仕様の設定 （性能・容量）	・外部仕様の設定 （機能・性能）
概念設計	・課題に対する解決策／代替案	・機能ブロック設計	・処理方式検討
基本設計	・解決策の展開 ・実現可能性確認	・レジスタトランスファレベル設計	・マクロフロー作成
詳細設計	・製造に必要なレベルまで詳細化	・回路レベル設計 ・レイアウト設計	・詳細ドキュメント作成
非設計 製造	――――	・プロセス	・コーディング
非設計 試験		・加速試験	・デバッグ
非設計 運用		・量産	・システム運用
評価	・再設計に必要な情報収集／判断	・フィールドデータ	・負荷試験等
再設計	・機能性能向上のための設計	・機能ブロック再利用／改良	・モジュール再利用／改良 ・仕様変更／機能拡張

図2.2－2　計画・設計業務の流れ

2.2.3　計画・設計の過程のモデル

図2.2－3に計画・設計の過程のモデルを示す．各工程は(a)図に示すような基本タスクから構成され，また，これら一連の基本タスク群は(b)図に示すように修正に関連する後戻りを組合せた流れとしてモデル化した．

なお，各タスク，および複数タスクの組合せに対する知識処理によるモデルの詳細については2.3節で述べる．

(1)詳細化

前工程の結果から，さらに製造・運用に近いレベルまでドキュメント記述を詳細化，具体化する．また，不足情報も補う．

(2)解析

次ステップの評価・検証ためにシミュレーションやトレース等を実施し，対象のシステムの動作・挙動・性能を見積もる．

(3)評価・検証

詳細化によって得られた結果と，前記解析結果とを比較し，その機能・性能・特性に関して目的の条件を満たすかどうかを判定する．また，「良」と判定した場合は次工程に進む．

なお，一般には，評価のための基準（評価基準・制約）がある．

(4)修正

目的の機能・性能・特性が得られなかった場合（「否」と判定），誤りの訂正や詳細化不足の部分の詳細化を行う．また，該ステップでの修正で対処できない場合には前工程へ戻り，前工程の詳細化を再実施する．

〔参考文献〕

〔長澤87〕　設計エキスパートシステム，情報処理，２８，２（１９８７）１８７．

〔Mostow85〕 J.Mostow,"Toward Better Models of The Design Process",AI MAGAZINE pp.44-57,Spring,(1985).

〔JIPDEC87〕 知的情報処理システム調査研究委員会資料　ＥＳ開発方法（１９８７）

〔ICOT88〕　第６回ＩＣＯＴシンポジウム予稿集　pp.20-21（１９８８）

（2.2節担当　桃井）

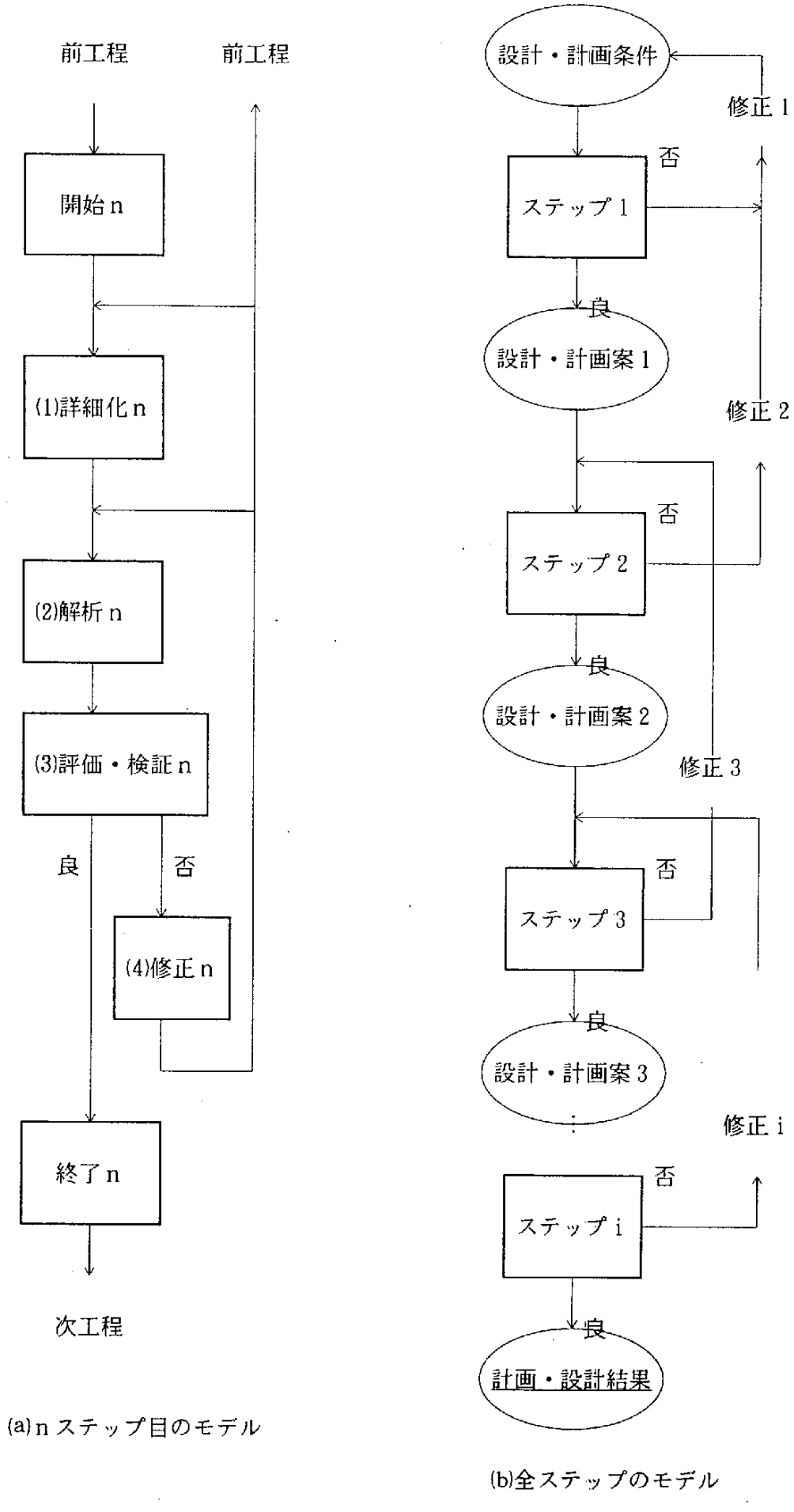


図2. 2 － 3　計画・設計の過程のモデル

２．３　計画・設計型問題の多元的側面

（１）探索的側面

一般に物事の計画・設計は難しい問題であり，その解法は従来からＯＲ（オペレーションズ・リサーチ）の分野において研究されている．その成果として線形計画問題で単体法やカーマーカ法などが開発されたことは良く知られている．これらは数式で表現された評価関数を最小（最大）化するアルゴリズムとして定式化される．一方，ＡＩ特にエキスパートシステムの分野における設計・計画問題は数式として定式化しにくいものを扱うために，探索（状態空間探索）を基本的な解法として用いており，各種の探索法が開発されている．ここでは計画・設計問題の探索問題としての側面を，例を用いて述べる．

例）３個の電灯Ａ，Ｂ，Ｃがある．これらは個別に点灯・消灯できるが，任意の時点でどれか２個以上が点灯していなければならない．ところがＡとＢの同時点灯は１単位時間以内であり，連続点灯可能なのはＡ，Ｃが２単位時間以下でＢは３単位時間のみである．また全ての電灯は一度消灯するとその直後１単位時間は点灯できない．各灯の点灯・消灯を単位時間毎に決めるものとし，１２単位時間の点灯・消灯を決定する．

探索を用いる場合，状態の概念が重要になる．状態とは問題を表現する可変パラメタが各時点で取る値の集合ということができる．１個のパラメタは各時点で変化する可能性があり，したがって状態も各時点で変化し得る．状態が変化することを状態遷移と呼ぶ．１個の状態が遷移するとき，発生する可能性のある状態（候補状態）は一般に複数存在し得る．この中からどれかの状態をある基準で選び，それが制約条件に抵触していなければ選んだ状態を遷移させ，抵触していれば別の候補状態を選ぶ．これらが全て抵触すれば１段前の候補状態の中から選び，同様の操作を行なう．これを最後の状態に到達するまで繰り返す．これが探索であり，一口でいうと探索とは状態遷移を追跡する過程のことである．

上記の例では何が状態になるであろうか．それは問題を見る視点，いいかえれば問題の定式化によって異なる．まず考えられるのは，１単位時間毎の各電灯の点灯・消灯の組み合わせである．これを用いると全部で８種類の状態が可能であり，ある状態から遷移できるのはこれらの部分集合になる．その中から制約条件に適合する状態を単位時間毎に追跡すれば探索を行なうことができる（図２．３－１）．

他の視点では，この問題は各電灯の点灯時間（Ａ，Ｃが１から３単位時間，Ｂが３単位

時間）をいわば部品と考え，点灯スケジュール表の中に配置する過程とみなすこともできる．この場合，状態は表中に配置された部品の位置関係であり，状態遷移は位置関係の変更によって生じることになる（図２．３－２）．

このように，同じ問題であっても探索するための状態の表現方法によって，計画問題的性質が強調されたり，設計問題的側面が浮かび上がったりする．いずれにしても計画・設計問題のエキスパートシシテムでの基本的解法は探索であり，適用可能な問題の範囲は広い．その一方，上述した程度の探索手法はプリミティブなものでありほとんど試行錯誤に近い．したがって解が出るまで通常かなりの時間がかかる．探索の本質は試行錯誤だとしても，効率的な探索方法を工夫する必要がある．そのための手法は各種開発されている（３章）が，重要な点は解こうとしている問題をそれらの手法が適用できるように変換することであり，そのためには問題認識・問題分析を通して問題全体を複数の小規模な部分問題へ分割することが必要になる．

（2.3節 (1) 担当 中島）

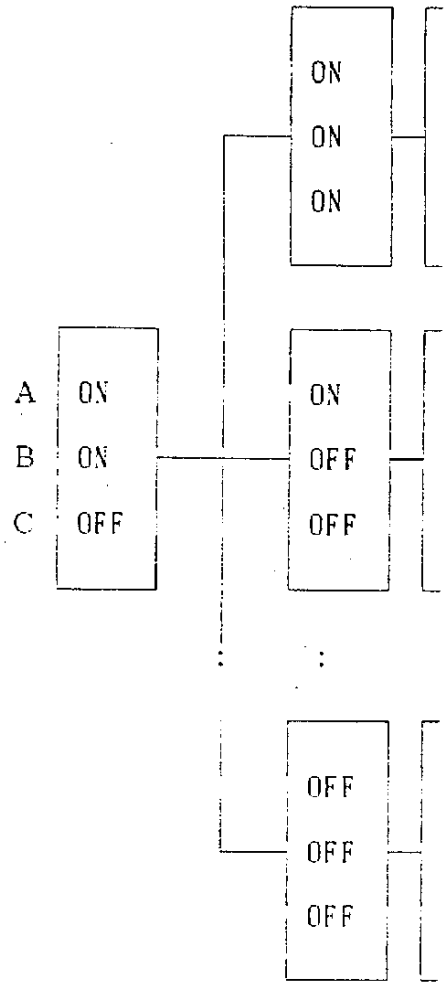


図 2.3-1 計画問題的な状態表現

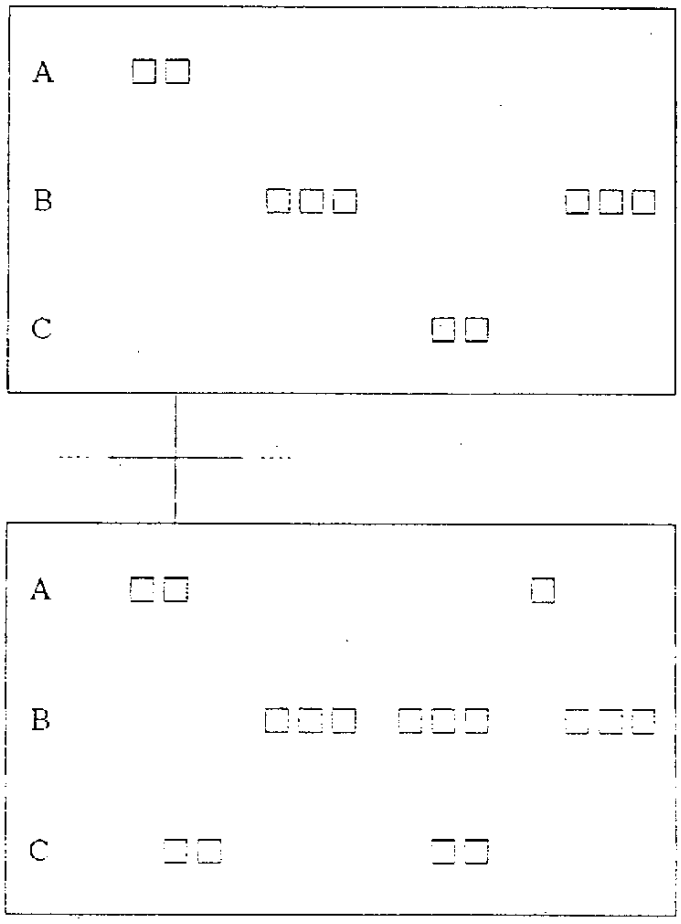


図 2.3-2 設計問題的な状態表現

(2) 意志決定的側面　（仮説推論的側面）

合成型問題すなわち計画型，設計型問題は与えられた制約を満たすパラメータの組を見つけ出す問題として捉えることができる．すなわち，パラメータ群の作り出す多次元空間は設計あるいは計画を表す点の集合であり，問題はこの中の適切な1点を決定することであると考えることができる．このとき，一般に解空間は非常に広く，設計あるいは計画はユニークには定まらない．つまり制約条件を満たす設計パラメータ，あるいは計画パラメータの選択には自由度があり，その中で，パラメータをどのように選んで行くかは意志決定問題と捉えることができる．設計問題や計画問題を解く過程において，試行錯誤の繰り返しが行われるが，この試行錯誤を取り扱う枠組みとして仮説推論が適切である．試行錯誤の中で，競合する知識を扱う場合が多く，矛盾を生じる可能性があるため，これらを仮説として扱う必要がある．計画や設計問題は，’生成－検証’パラダイムに基づく探索を必要とするので，仮説推論を必要としているといえる．

設計問題あるいは計画問題はまた，基本的には組合せ問題である．取り扱い不可能な大規模問題であるところに難しさがある．組合せ的爆発の問題を回避するために，計画過程を階層化するのが通例である．1)で探索的側面について述べたが，実際の問題においては探索すべき空間が広すぎるために，設計者（計画者）が，あらかじめ探索空間の絞り込みを行うのが通例である．この絞り込みは，階層的な問題の定式化に対応しており，一階層下の部分問題を定義することになる．ここに設計者の意志が寄与する部分が多いにある．

設計・計画の各階層では，1)にのべたような，組合せ的探索問題を解くことになるが，これでも組合せ的爆発問題の回避ができないときは，設計・計画者の意図によってさらに枝刈を行うことが必要である．設計・計画者は予測，あるいは推測に基づいて，この絞り込みを行うが，その予測は不確実性をもつために，リスクをうまく取り扱うためのメカニズムが要請される．仮説推論の枠組みはこの要請に答えられるものである．しかし現実の問題を取り扱うに当たっては，拘束条件のつよさ，解空間の大きさなど，個々の問題によって様相が大きく異なっており，何を仮説の集合として捉え定式化するかは大きな課題である．

例えば，階層設計を行う場合，階層を一段下りるときに定める条件，これは設計パラメータの一部であるが，は最終的に与えられた制約条件を満足するかどうかはその時点では不明である．これを仮説として，設計を進めることができる．この際どの様な仮説を置くかは意志決定の問題である．すなわち，”意志決定”とは”仮説を設定すること”となる．

(3)再利用的側面（事例推論的側面）

計画・設計問題に限らず，人間の問題解決は過去の成功及び失敗の経験に基づいて新しい問題を解決する事がよく行われる．一般に人間の問題解決行動には事例の再利用的側面がある．たとえば過去に試行錯誤の結果うまくいった事例があれば，同じ様な問題に直面したときには再度ルールによる推論を実行することなくその成功事例に基づいてただちに問題の解決を行おうとする．ここでは計画・設計のいくつかの例を取り上げ，計画・設計問題に再利用的側面のあることを指摘する．

①計画（スケジューリング）問題での再利用的側面

例として工場における生産計画の日程展開を考えてみよう．生産設備の増設や故障による停止などで生産資源が変化した場合や，生産品目や数量がこれまでの計画立案と大きく異なる場合を除くと，実際に稼働実績のある過去の計画（稼働実績）を基に，今回の生産条件に合わせて変更・修正を行い，新たな計画を作成するという問題解決のアプローチが取られる．

②設計問題での再利用的側面

設計問題を，機械設計に代表されるような物理的設計問題と，機能性材料の設計に代表される化学的設計問題に分けて特徴の分析を行う．

(i)物理的設計問題

機械設計やソフトウェア設計などのように，仕様や制約条件を充足するように部品や要素を組み合わせ最適化をおこなうことをここでは物理的設計問題と呼ことにする．

このような設計問題は，新製品の開発や，既存製品の改良や変更による仕様の変更に伴い発生する．このうち既存製品の一部改良や部分的変更に代表されるルーチン設計は，過去の設計事例を基に機能追加や修正を施して新しい設計案とすることがよく行われる．実際の設計業務の中でこのようなルーチン設計の占める割合は多い．また新製品の開発に伴う設計においても，まったく白紙の状態から問題領域の知識によってトップダウンに設計を行うことはまれで，一般的には過去の類似の設計事例を探しだしそれを参考にすることが通常行われる．

(ii)化学的設計問題

超伝導材料や触媒などの機能性材料設計，生理活性物質や医薬品などの新規化合物の分子設計，また界面活性剤や乳化剤，分散剤，香料などの配合処方設計などに代表される化学的設計問題では，物理的設計問題で要求される仕様・制約条件の充足に加え，要素の組合せによる相乗効果（シナジー）の発現が要求される．

これらの分野においては，設計のための知識がルールとして体系化されていないことが多いため，これまでに設計した材料・化合物の構造とその機能，また配合組成とその性能を蓄積したデータベースが設計に際して不可欠である．設計はこのデータベースを参照しながら行われる．そのような分野では成功例だけでなく失敗例もデータベース化しておくことにより，同じ失敗を２度繰り返さないようにするだけでも設計の効率化には役立つ．

これらの計画・設計問題の特徴として以下のことが言える．

①問題とする領域が開いている．

②計画・設計のための領域知識はルール化，体系化されていないことが多い．

③専門家は領域知識として，ルールにまで一般化されていない計画・設計事例を数多く持っており，それらを参照しながら問題解決を行っている．

④成功例には数多くの失敗例を背後に持っており，専門家はそれらの成功・失敗事例を経験として持っている．

⑤同じ失敗は予め予想し，回避するだけでも計画・設計の効率化には役立つ．

⑥まったく同じ問題を取り扱うことはまずなく，専門家は問題解決を行いながら学習し，その結果を以後の問題解決に反映させている．

⑦領域のルール化された知識に基づいて，白紙の状態から計画・設計を行うことはあまりなく，過去の類似の事例を基に修正・変更を行い新しい問題の解とすることが多い．特に業務の中で多くの割合を占めるルーチン設計では過去の事例を基にすることが多い．

以上述べたように，計画・設計問題においては探索をガイドする理論や知識が確立されていないことが多いため，過去の成功および失敗の事例に基づいて問題解決を行うという事例の再利用的側面のあることがわかる．

（2.3節(3)担当　関根）

(4) 知識獲得的側面（学習的側面）

類似の事例が利用できない全く新しいタイプの問題においては，計画・設計者は問題解決に有用な知識を自ら獲得していくことが必要になる．新しい知識の獲得が全くの偶然によって得られる場合もあるが，一般的には，知識の獲得は多数の正例および負例の背後に成立する法則や原理を帰納的に導くことによってなされるか，既に獲得されている知識を使って新しい事例を説明しかつこれを一般化することにより，演繹的に導くことによってなされるかのいずれかとみられる．事例からの知識の獲得は学習の問題に密接に関係する．

設計問題に使われる知識には，設計知識と制御知識の２とおりがある．設計知識とは，要求仕様を詳細化するための手段を与える知識をいい，制御知識とは，設計知識を問題解決の文脈に応じて適切に選択することをガイドする知識をいう．

[J. Mostow(1985)]は，計画・設計問題における学習の役割を論じている．人間が行う計画・設計活動では，問題解決と学習の間には相補的な関係がある．人間は計画・設計問題を解く過程において，失敗および成功を含めて，さまざまな種類の設計に関する知識を獲得しているとみられる．人間は初めて遭遇する問題であっても，探索空間を網羅的に探索するようなことはしない．探索の進行に従って，同じような失敗は回避するような行動を取るようになるし，一方，つぎに探索すべき方向を盲目的にではなく，これまでの探索情報を生かして，選択的に決定しているとみられる．これは発見的な探索過程であり，その背後には問題領域に関する知識が有効に働いているとみられる．人間が問題解決過程を通じて獲得する知識の多くは，既知の設計知識をどういう状況のもとで使うべきかを規定する制御知識と考えられる．

[Mitchell et al.(1984)]は，領域専門家が行う回路設計領域での設計活動を肩越しに観察して，設計仕様を回路上に実現するまでの一連の行動の妥当性を領域知識を使って証明し，さらにこれを一般化することにより，他の類似の状況にも適用できる制御知識を獲得できるような枠組みを提案し，これを見習い学習（Learning Apprentice）と名付けた．見習い学習のベースとなった学習は，その後，説明に基づく学習（Explanation-Based Learning)として発展しており，現在では，人工知能分野での学習研究におけるもっとも活発な研究領域となっている．説明に基づく学習は，領域理論を利用する演繹的な学習であるが，最近では，帰納的学習との統合化が試みられている．

計画・設計問題における知識獲得問題の所在を明らかにするために，論理回路設計における素子数最適化問題について考察しよう．この問題では，与えられた論理回路と等価な素子数最小の回路を見出すことが求められる．設計知識としては，つぎのような等価変換規則（オペレータ）が利用可能である．

①２重否定：　(not (not X)) <==> X

②ドモルガン則：　(not (and X Y)) <==> (or (not X) (not Y))

(not (or X Y)) <==> (and (not X) (not y))

与えられた回路に対して，素子数を最小とする等価回路は一般に複数存在し，しかも最適回路に至るオペレータの適用系列も複数存在する．与えられた回路に対して，一般に，すべてのオペレータが適用可能であり，しかも１つのオペレータを選択したとしても，その適用可能な場所は複数存在する．与えられた回路から最適解に至る状態変化において，素子数が単調には減少することはない．このため，探索問題として定式化する際に必要とされる有用な評価関数は見出されていない．

与えられた回路に対し，どのオペレータをどの箇所に適用すればよいかを指示する知識は制御知識と呼ばれるものであり，回路設計者は経験により獲得するしか手立てはないといえる．すなわち，設計者自身が経験からの学習を行っているとみられる．このような設計問題では，設計者が選択・適用するオペレータを肩越しに観察し，その正当性の根拠を証明し，かつその証明過程を一般化して，マクロルールとして蓄積することができれば，類似の問題が提示されたときに，マクロルールを適用することにより，問題解決過程を向上させることが期待できる．

このように，計画・設計者自身が，予め，設計する手段である設計知識をもっていても，それを制御するための知識が整理されていない状況のもとでは，見習い学習方式に基づく学習は有用と期待される．

一方，設計知識が不完全であれば，仕様を詳細化することができない．これは設計知識の洗練化問題と呼ばれる．事例からの説明に基づく学習は，設計知識を洗練化するためにも有用である．

参考文献

[Mitchell, T. M., Steinberg, L. and Amarel, S.(1984)]

A Learning Apprentice for Knowledge Acquisition in an Expert System, Rutgers Digital Design Project Working Paper, No.16.

[Mostow, J.(1985)]

Toward Better Models of the Design Process, The AI Magazine, Spring, 44/57

[Mitchell, T. M., Keller, R. M. and Kedar-Cabelli, S. T.(1986)]

Explanation-Based Generalization: Unifying View, Machine Learning, Vol.1, 47/80.

[Dejong, G. and Mooney, R.(1986)]

Explanation-Based Learning: An Alternative View, Machine Learning, Vol.1, 145/176.

(2.3節(4)担当 小林重信)

3．計画・設計型エキスパートシステムの基盤技術

3．計画・設計型エキスパートシシテムの基盤技術

3．1　問題解決手法

本節では計画・設計問題の解法として探索を基本とする場合，各種探索手法の特徴，それらを効率的に適用するための手法（［Hayes-Roth他　85］，［小林　85］）について述べる．また，最適化の一手法として近時研究が盛んなニューラル・ネットワークを説明する．

3．1．1　探索手法

探索とは問題を表現する可変パラメタとしての状態の遷移を，初期状態から終了状態まで追跡する過程といえる．遷移によって生じた新たな状態は更に何種類かの状態に遷移する．解が出るためには追跡は有限回のステップで停止しなければならないが，この過程は全体としてツリー構造で表わすことができる．ネットワーク構造の上を探索する場合には既に通過した状態に出会ったらそこでの追跡を止めることにすればツリーと等価な探索ができる．したがって以下はツリーの探索として話を進める．

（1）深さ優先（縦型）探索

ツリー構造において枝が伸びる方向を深さ方向とよぶ．ある状態Aの次に追跡する状態Bを，Aから伸びた枝につながる状態の中から任意に選ぶ方法が深さ優先探索（あるいは縦型探索）である（図3．1－1），したがってBを追跡している時点では図中の状態C，Dは未追跡の場合もある．Aにつながる状態が全て失敗したときC，Dの中から未追跡のものを任意に選ぶ．

この方法は探索がツリーの特定のレベルにおいて終了することが既知の場合に効果がある．例えば2．3節での例では1単位時間を深さ1段に対応させる探索方法にすれば12段の探索が必要十分であり，最小で12回の追跡で終了する．逆に，深さ方向に無限に遷移し得る状態の場合には追跡途上に終了状態がない限り，解は永久に見つからないことになる．このようなときは探索段数を制限することが必要になる．

（2）広さ優先（幅優先，横型）探索

ツリー構造において同一レベル内の方向を広さ（あるいは幅）方向とよぶ．ある状態A

の次に追跡する状態Bを，Aと同一レベルにある未追跡状態の中から任意に選ぶ方法が広さ優先探索（あるいは幅優先探索，横型探索）である（ 図 3．1－2 ）．選べなければ次の深さレベルの状態の中から選ぶ．したがってある状態を追跡しているときはそれより浅いレベルの状態は全て追跡済である．

この方法は探索終了のレベルが未知の場合に効果的であり，解が存在する場合には必ず見出すことができる．しかし（1）で述べたような例では深さ優先探索よりも時間がかかる可能性が大きい．

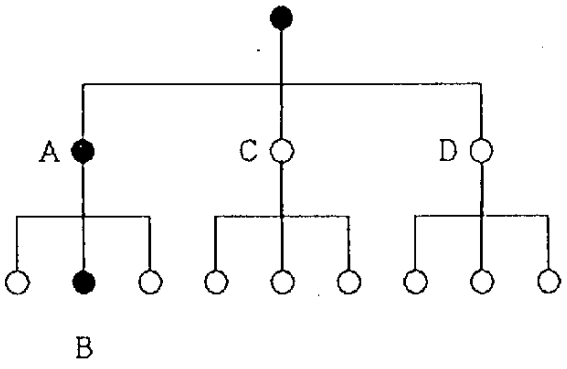


図 3.1－1　深さ優先探索

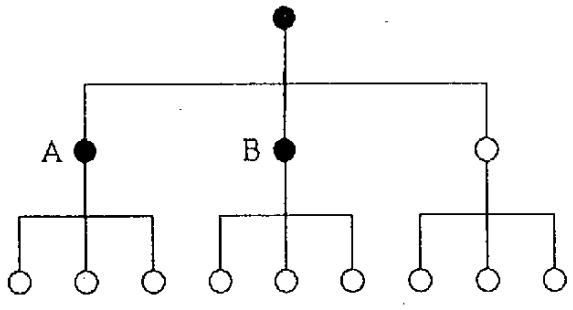
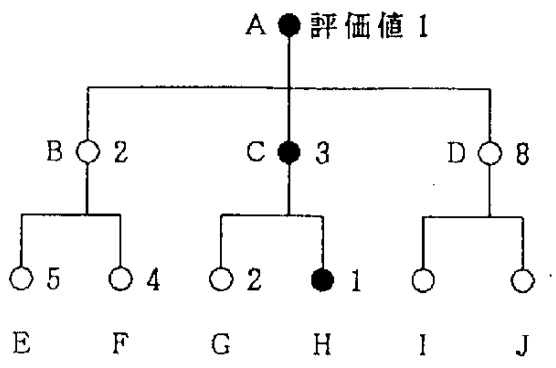


図 3.1－2　広さ優先探索

（3）最良優先探索

深さ優先探索でも広さ優先探索でも，次に追跡する状態はツリーのある範囲から任意に選ばれる．もし，拘束条件を満足するという意味での解が複数あり，実際にはその中からある評価基準にてらして最適な解を選びたい場合には，どちらの方法でもまず全ての（拘束条件を満足する）解を出さなければならない．その後各々の解から最適解を選択することになる．具体的には（1）の例で点灯・消灯の切り換えにコストがかかり，その累積コスト最小の解が欲しいといった場合である．

最良優先探索は，広さ優先探索に各状態での評価値を計算する処理を付加した方法である．評価値が最適値に近づく状態を常に追跡し，ある状態の追跡が失敗したら次に最適値に近い評価値になる状態を追跡する．常に最適に近づく状態を選んでいるのであるから，最初に得られた解が最適解になる（図3.1－3）．



・A＋B＝3，A＋C＝4，A＋D＝9

・A＋B＋E＝8，A＋B＋F＝7，
A＋C＋G＝6，A＋C＋H＝5，

枝を展開した時点で累積値最小の経路を通る

図　3.1－3　最良優先探索

（4）ヒューリスティック探索

次に追跡する状態を選ぶとき，個々の問題に依存した経験則を用いる方法である．この方法は，確たる根拠はとりあえず不問にして経験則（ヒューリスティックス）によって次の状態を選択するため，経験則の質が重要になる．特に計画・設計型問題の場合，経験則が問題の構成要素の数やその属性，あるいは初期条件などの変化に対して有効性が低下しない（頑健である）必要がある．そのような場合には非常に効果的であるが，問題依存性が強い（他の問題に流用しにくい）欠点がある．

3．1．2　問題解決戦略

ごく小規模な問題を除いては，状態空間の探索に前節で述べた個々の探索手法がそのまま用いられることは少ない．一般に問題全体の探索空間は広大なため，部分問題に分割し効率的な探索を実現する必要がある．本節では探索を効率化するための戦略について述べる．

（1）トップダウン精密化

計画・設計問題では拘束条件に優先順位が付けられることが多い．そのような場合には優先度の高い条件を満足させる粗い計画をまず作成し，その後他の条件を満足させるため

計画を詳細化する．例えば旅行計画を立てる場合，出発地から目的地へのルート決定において航空路線や高速道路のルートをまず決定し，ついで空港やインターへのルートを決めることに相当する．このような方法をトップダウン精密化と呼ぶ．優先度の高い条件のみ最初に考慮することで探索空間が限定され，詳細化の段階では粗い解自体が制約条件となって更に探索空間が縮小される．

（２）階層的生成検査

探索の実行は生成検査として定式化される．状態を遷移させて新たな状態を作り出す過程が生成，作り出された状態に対して制約条件との適合性を調べる過程が検査であり，制約条件に適合しない状態は検査段階で棄却することにより枝刈りが行なわれる．階層的生成検査では状態の一般的（抽象的）な記述を複数作成し，検査を通過した後より具体的な状態を生成・検査する動作を繰り返す．多くの場合，状態の記述が一般的であれば，最初から具体的に記述するよりも探索空間が狭くなる．

（３）拘束最小化

計画・設計作業においては，全体をいくつかの殆んど独立な部分問題に分割するすることが日常的に行なわれる．例えばソフトウェアの設計ではモジュール単位に分割し，更にサブルーチンに分割する等である．これにより全体を小規模な部分問題毎に扱うことが可能となるが，部分問題が完全に独立でなければその部分問題の外部の情報が必要になる．拘束最小化はそのような場合に，自前の情報だけでは決定できない部分については外部の情報が利用できるまでとりあえずその先の実行を保留して他の部分問題の検討に移り，必要な情報が得られた時点で，保留されている部分問題の実行を再開する方法である．保留がいつまでも解除されない場合にはヒューリスティックスなどで情報を決定する．

（４）依存関係後戻り

計画・設計ではある仮定を設定しそれに基づいて作業を進めなければならないことがある．探索の途中でも情報不足の場合，何らかの仮説を設定して探索を進めることが必要なことがある．この場合一度仮説を設定するとそれ以降の探索結果はその仮説に依存することになるため，もし探索結果が失敗したら仮説設定時点まで後戻りして仮説を変更し，新たな仮説の下で探索を再開する．この方法を依存関係後戻りという．深さ優先のような単

純な探索がツリーの１段上へしか後戻りしないのに対して，依存関係後戻りでは何段も上のレベルへ戻ることができ，しかも根拠のある後戻りが可能になる．

３．１．３　最適化手法としてのニューラル・ネットワーク

計画・設計では解が出るだけでは十分ではなく，何らかの基準に関して最適な解を要求されることが多い．このためＯＲ等の分野では各種の評価関数最小化手法が開発されてきたが，近時ニューラル・ネットワーク（ＮＮ）を用いた最適化法が注目されている．ＡＩシステムとしてのＮＮは分類・診断方面への応用が研究されているが，計画・設計問題への応用はこれからの課題である．本節ではＡＩシステムとはとりあえず区別し，独立した最適化システムとしてのＮＮについて述べる．なおＮＮの学習機能についてはふれない．

（１）ＮＮのモデル

ＮＮはニューロン呼ばれる素子が多数結合したネットワークである．生体の神経細胞のことをニューロンというが，その特性を単純化して数式で記述したモデルは形式ニューロンと呼ばれる．したがって形式ニューロンのハードウェアは特に規定せず，ソフトウェアによる仮想的な存在であってもよい．以下単にニューロンというときには形式ニューロンを含むものとする．

さて，生体ニューロンの特性を完全にモデル化することは無理であるから，抽象化して重要と思われる特性だけが数式化されている．神経生理学からの知見によれば，ニューロン間の情報伝達は軸策を通る電気的パルスによって行なわれる．１個のニューロンが他の多数のニューロンからのパルスを受信すると，その総和によって受信ニューロン内の活動度とよばれる内部状態が上昇し，これがあるバイアス値（しきい値）以上になると他のニューロンへのパルスが出力される．これを模式的に表わすと図３．１－４のようになる．すなわち１個のニューロンは総和特性としきい特性とから構成され，数式モデルはこの範囲を表現する．

ＮＮがｎ個のニューロンから成るものとすると，ｉ番目（$1 \leqq i \leqq n$）のニューロンの特性は式（ａ）で記述される．

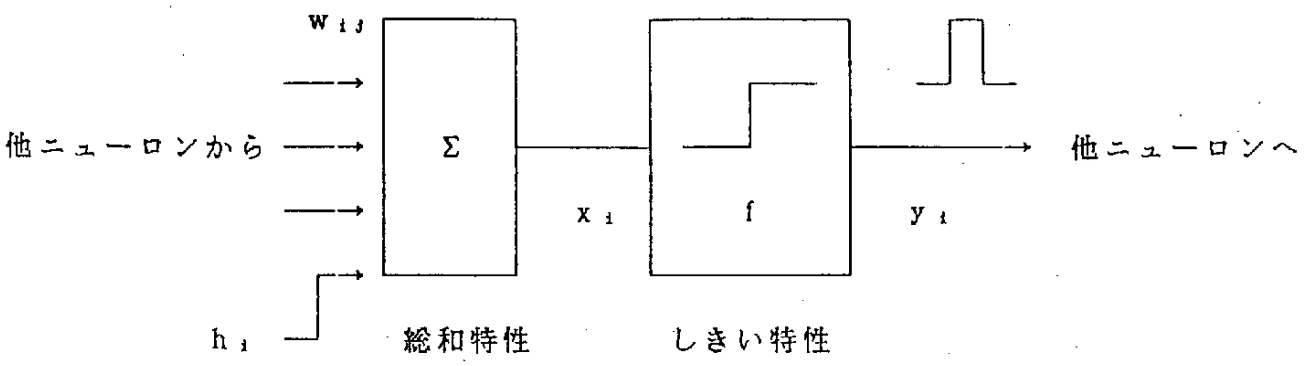


図　3.1－4　　ニューロンの模式図

$$\frac{dx_i}{dt} = \sum_{j=1}^{n} w_{ij} \cdot y_j + h_i \qquad (a)$$

$$y_i = f(x_i)$$

ここでx_iはニューロンの状態，y_iはニューロンの出力，w_{ij}はj番目のニューロンからi番目のニューロンへの信号伝達効率，h_iはバイアス値，fはしきい特性を表わす出力関数であり，通常 $f(x)=[1+\tanh(x/a)]/2$ と書かれる．これは定数$a \neq 0$のときS字状の滑らかな関数であり，$a \to 0$のとき単位階段関数である．

（2）NNの動作

各ニューロンの方程式は非線形であり，解析的に解くのは容易ではない．HopfieldはNNにポテンシャルの概念を導入し，ネットワークの全体的な挙動を考察した［Hopfield 82］．

次の関数Eを考える．

$$E = -(1/2)\cdot\sum_{i,j=1}^{n} w_{ij}\cdot y_i\cdot y_j - \sum_{i=1}^{n} h_i\cdot y_i$$

これを y_i で微分すると，$w_{ij}=w_{ji}$ のとき

$$\partial E/\partial y_i = -\sum_{j=1}^{n} w_{ij}\cdot y_j - h_i$$

であり（a）式から

$$dx_i/dt = -\partial E/\partial y_i$$

が成立する．すなわち各ニューロンの状態が変化するとき，Eは減少しその方向は最急降下方向であることがわかる．これはポテンシャルの性質と等価であり，そのためEはＮＮのポテンシャル関数（あるいは物理系とのアナロジーからエネルギ関数）と呼ばれる．

結局，上で述べたＮＮのダイナミックスは，ポテンシャルを下りその極小値に落ち着く動作であるといえる．

（３）最適化への応用と留意点

数理的な最適化手法は一般に評価関数の最小値を求める方法である．ＮＮを最適化手法として用いる場合は，ポテンシャル関数が評価関数に対応する．ある最適化問題に対する評価関数をＮＮのポテンシャル関数に変換すれば，ＮＮのダイナミックスによって自動的に評価値が改善されてゆく．

評価関数 ⇦――⇨ ポテンシャル関数 ――→ＮＮのダイナミックス

解 ⇦――⇨ ポテンシャルの極値←――――┘

この方法は基本的には評価関数の最急降下法と変わらない．メリットはネットワークが数式上ニューロン毎に並列動作しており，したがって原理的には極めて高速に解が出る点にある．現時点では通常の直列動作計算機でシミュレートされているが，並列計算機あるいはＬＳＩ化によって並列動作させることにより，大規模な組み合わせ最適化問題（ＮＰ完全問題など）が実用時間で解ける可能性がある．

ＮＮによる最適化で留意すべき点は，解が一般には準最適解であることである．これはポテンシャル関数が一般に多数の極値を持ち，ＮＮは初期条件によってどれかの極値に収束するためである．したがって複数の初期条件でいくつか準最適解を出し，その中から最も良い解を選択するか，あるいは極値から脱出してより最適に近い極値に移行する方法を考える必要がある．後者の場合にはネットワークにある確率分布を持つノイズを混入するボルツマン・マシンなどが提案されているが詳細は文献［倉田　87］にゆずる．

文献：

［Hayes-Roth他　85］　エキスパートシステム，産業図書

［小林　85］　知識工学の基礎と応用［第５回］，計測と制御，Vol.24，pp.730-738.

［Hopfield　82］ Neural network and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. USA(79), pp.2554-2558.

［倉田　87］　神経回路モデルとしてのボルツマン・マシン，数理科学，No.289，pp.23-28.

（3.1　(1)　担当　中島）

3.2 仮説推論

知識ベースシステムでは，ある時点で正しいと思われていた知識もさらに新しい知識が追加されると成立しなくなったり，あるいはその後その知識と矛盾するような新しい知識が得られたりする．その場合，知識ベースの論理的な整合性を保つためには，矛盾を生じた知識を取り除いたり，矛盾を導きだす根拠となった知識，それらの知識から導き出された他の知識を訂正する必要がある．このような知識ベースにおける論理的整合性をメンテナンスするために，ＤｏｙｌｅはＴＭＳ(truth maintenance system)を提案し，依存関係の修復を行うためのメカニズムを与えた［Doyle 79］．ｄｅＫｌｅｅｒはＴＭＳを多重世界に拡張して，ＡＴＭＳ(Assumption-Based Truth Maintenance System)を提案している［De Kleer 86］．ここではＴＭＳ，ＡＴＭＳについて述べる．

1) ＴＭＳ(truth maintenance system)

ＴＭＳでは，知識は信念(belief)とよばれ，ＴＭＳは信念とその支持理由を記録しておくことによって，信念の集合が変化した場合に信念のあいだの論理的整合性をメンテナンスするための処理を行う．ＴＭＳは，ノードと呼ばれる信念を表す識別子と弁明(justification)と呼ぶ信念の支持理由の２つのデータ構造から構成されている．それぞれのノードは弁明を通じて互いに依存しあい，全体として依存関係(dependency)によるネットワークを構成する．このネットワークのうえで新しいノードが作られたり，新しい弁明が付け加えられたりしてノードの状態が変化すると真実メンテナンス(truth maintenace)プロセスが実行される．また矛盾が生じたことを示す信念が付加されると，原因を調べて矛盾を解消する依存関係によるバックトラックが実行される．

このようすを，図3.2-1のような簡単な回路を用いて，見ていく．

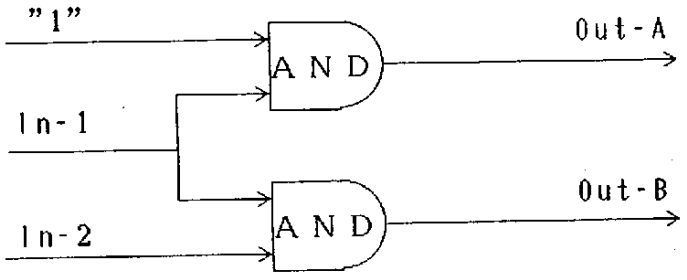


図3.2-1　簡単な論理回路

回路の構造が次の様なルールで表現される［寺野　87］.

R1:　IF　In-1(X)　THEN　Out-A(X)

R2:　IF　In-1(1) and In-2(X)　THEN　Out-B(X)

R3:　IF　In-1(X) and In-2(1)　THEN　Out-B(1)

R4:　IF　In-1(0) and In-2(X)　THEN　Out-B(0)

R5:　IF　In-1(X) and In-2(0)　THEN　Out-B(0)

ここでIn-1(X), In-2(X), Out-A(X), Out-B(X)はそれぞれ入力と出力の状態を表す述語である. 例えば最初のルールは, 入力1の状態と出力1の状態とが一致することを示している.

ここで図に示している回路では出力が0であるか, 1であるかの状態が設計の要求仕様として与えられ, その出力を実現する入力を設計する問題を考える. このような系の状態を想定するために, 上のような回路構造を表現するルールの他に, 仮説に基づくルールを使うこととし, 推論を進めるために, 以下の様なルール群を設定する.

A1:　IF　Maybe　IN-1(0)　THEN　IN-1(0)

A2:　IF　Maybe　IN-2(0)　THEN　IN-2(0)

A3:　IF　Maybe　IN-1(1)　THEN　IN-1(1)

A4:　IF　Maybe　IN-2(1)　THEN　IN-2(1)

ここで，Maybe P は，not P がいえないかぎり，P は正しいと信ずることを意味する．

さらに矛盾を導き出すルールを挙げる．

C1: IF In-1(1) and In-1(0) THEN contradiction

C2: IF In-2(1) and In-2(0) THEN contradiction

C3: IF Out-1(1) and Out-1(0) THEN contradiction

C4: IF Out-2(1) and Out-2(0) THEN contradiction

これらのルールは0状態と1状態とが両立しない概念であることを述べたものである．

ネットワークの各ノードはin，outのどちらかの状態をとる．in，outはノードが信じられていること，信じられていないことを表す．ノードは次のような3つ組で表現される．

ノード	信念	弁明
(N1	Out-B(0)	(SL(N1 N3) (N4)))

ここで (SL 〈inリスト〉 〈outリスト〉) はサポートリストとよばれる，このノードが信じられる根拠をまとめたものである．〈inリスト〉に含まれているノードがすべてinであり，〈outリスト〉に含まれているノードがすべてoutであるならば，かつ，その場合のみこの弁明は妥当であることを示している．

いまOut-B(0)という事実が満たすべき仕様として与えられたとき，矛盾の生じない状況の一つとして，推論システムが次の状況を見つけたとする．

In-1(0) かつ In-2(0) かつ Out-A(0) かつ Out-B(0)

このときの論理的導出のようすは次のようになる．

ノード	信念	弁明	状態
N1	A1		in
N2	A2		in
N3	not In-1(0)		out

N4	not In-2(0)		out
N5	In-1(0)	(SL(N1) (N3))	in
N6	In-2(0)	(SL(N2) (N4))	in
N7	R1		in
N8	R2		in
N9	Out-A(0)	(SL(N5 N7) ( ))	in
N10	Out-B(0)	(SL(N6 N8) ( ))	in

ここでは，Out-A(0)，Out-B(0)という事象を仮定しているが，このとき，さらにOut-A(1)という事実が要求仕様として与えられると，このままでは矛盾が生じる．さらに，次のようなノードが付け加えられる．

ノード	信念	弁明	状態
N11	Out-A(1)		in
N12	C3		in
N13	contradiction	(SL(N9 N11 N12) ( ))	in

そこで，システムは，上に示したネットワークをたどって，仮説In-1(0)の誤りを見いだし，それを棄却して推論をやり直す．そしてあらたにIn-1(1)，In-2(0)という仮説を採用することとなり，次のような観測事実と矛盾しない状況の解釈を保つことができる．

ノード	信念	弁明	状態
N14	A3		in
N2	A2		in
N15	not In-1(1)		out
N4	not In-2(0)		out
N16	In-1(1)	(SL(N14) (N15))	in
N6	In-2(0)	(SL(N2) (N4))	in
N7	R1		in
N8	R2		in

N17	Out-A(1)	(SL(N16 N7) ( ))	in
N10	Out-B(0)	(SL(N6 N8) ( ))	in

これが依存関係に基づくバックトラックである．ＴＭＳは，信念の変化に応じて矛盾のない状態を一つずつ作り出すという意味で，状態空間を縦形に探索していることとなる．

2）ＡＴＭＳ(Assumption-Based Truth Maintenance System)

ＴＭＳでは，矛盾を生じない状態を一つだけ保持していた．これに対して，ＡＴＭＳでは，事実と矛盾しない状態を同時にすべて保持する．そして，新たに与えられた事実と矛盾する状態を順次消去していく．これは，状態空間を横型に探索する方法といえる．

仮説とは，観測事実と矛盾しない限りは自由にその真偽を選択できるデータである．先の例で，仮説ルールの中にmaybeとともにでてくるIn-1(0)やIn-1(1)などがこれにあたる．ＡＴＭＳでは状態はおのおの仮説の組合せに対応させることができる．これを表現するために各ノードにＴＭＳにおけるデータと（inサポートのみからなる）サポートリストに加えてそのデータが真となるための前提となる仮説の組（環境：environment）のリスト（これをlabelともいう）を保持している．そしてあるデータが新たな観測事実と矛盾を生じた場合，そのデータの環境が許されない仮説の組合せであったと判断して，その組合せを棄却する．

すなわち，その組合せを含む環境をすべてのノードのラベルから消去する．このような矛盾を生じる仮説の組合せをnogoodsと呼ぶリストで管理して，無駄な推論を防いでいる．この操作によって，各データを真とするために必要な環境を維持できる．

各ノードは次の様な構造をもっている．

ノード	信念	環境	サポートリスト
(n1	Out-B(0)	{(In-1(0))(In-2(0))}	((n2 n3) (n6)))

例えば，さきに挙げた例において，事実　Out-B(0)　が観測された時点に於ける状況を，仮説の組ごとに表示すると図3.2-2のようになる．図に示された仮説の組の箱をビューポイ

ントあるいはワールドという.

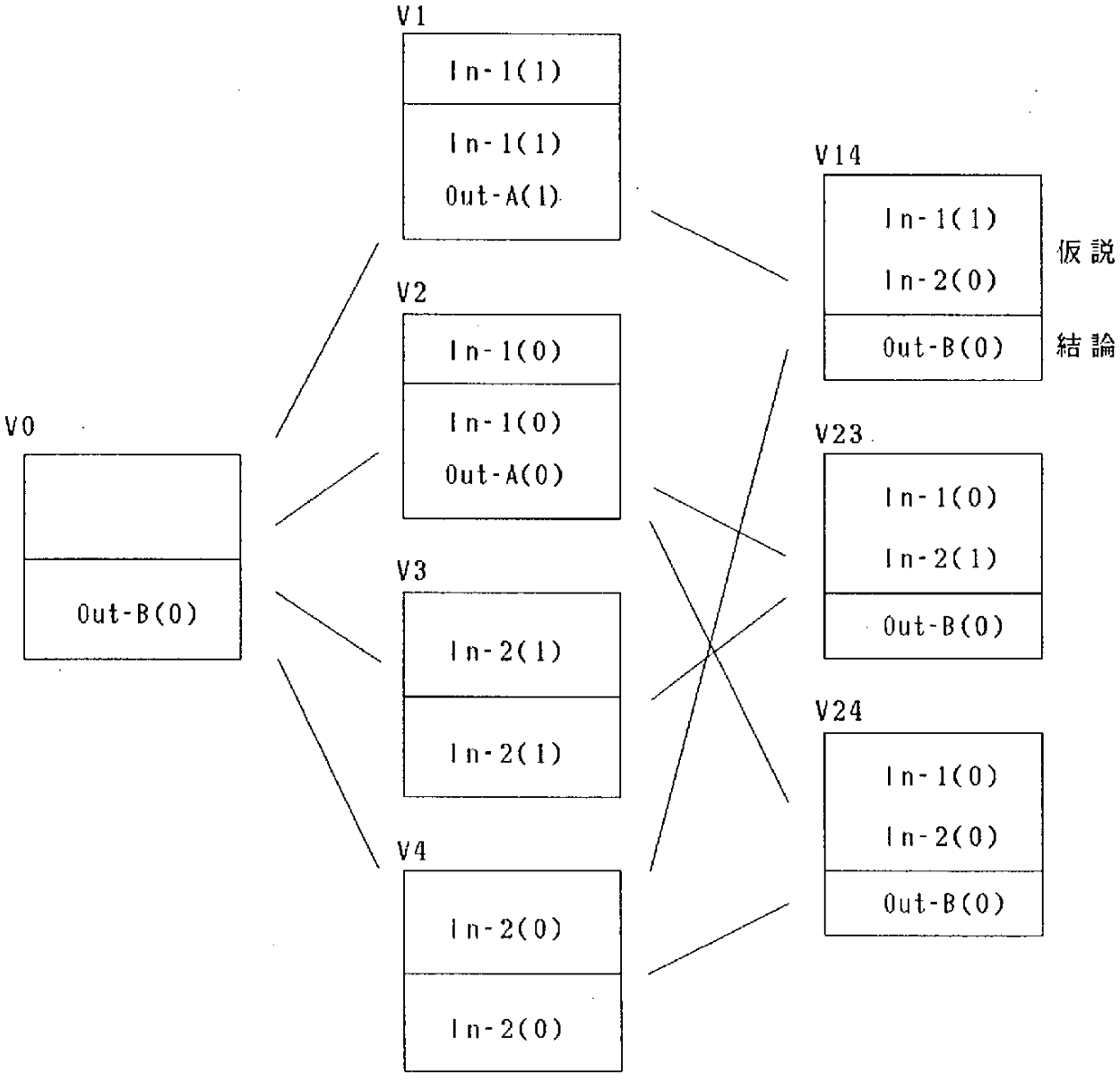


図3.2-2　事実Out-B(0)におけるビューポイントと状況

ここで，各箱の下側はそのビューポイントで新しく得られる結論を，上側はその前提として採用した仮説をしめす．なお，各ビューポイントからは，それ以前に得られた結論（すなわち図の上側にある箱の結論）はすべて参照できる．

この状況で，さらに，新事実Out-A(1)が要求仕様として与えられたとすると，Ｖ２では矛盾が生じる．すると，Ｖ２および，Ｖ２に従属するＶ２３，Ｖ２４は棄却され，同時にＶ３も棄却されてＶ１４のみが残る．このようにして，新しいノードが与えられるつど，矛盾しない解釈の維持ができる．

ＴＭＳ，ＡＴＭＳのモデルは，依存関係に基づく後戻りを自然に表現する枠組みを提供しており，設計・計画問題に限らず，診断型問題に対しても有効なほうほうである．しかし，これらは解決すべき問題に本質的に含まれている，組合せ的爆発問題をいかに克服するかという課題を解決するものではく，したがってこれらのモデルの上に知的な制御機構，問題解決機構を実現することが不可避である．

［参考文献］

［Doyle 79］J. Doyle : A Truth Maintenance System, Artificial Intelligence, Vol. 12, No. 3, pp. 231-272 (1979).

［De Kleer 86］J. De Kleer : An Assumption-Based Truth Maintenance System, Artificial Intelligence, Vol. 28, No. 2, pp. 126-162 (1986).

［寺野 87］寺野 隆雄，篠原 靖志：仮説的な推論－ＴＭＳ，ＡＴＭＳとその応用，オペレーションズ・リサーチ，Vol. 32, pp. 606-611 (1987).

(2.2節担当： 森)

3.3 事例に基づく推論

ルールに基づく推論では，専門家から獲得した知識をルールとして整理し，そのルールに基づいて推論を行う．そのような推論に対して，専門家の知識がルール化される前の情報，すなわち過去の成功および失敗の事例を組織化して蓄積しておき，それらを直接利用して問題解決を行う推論のことを事例に基づく推論（Case-Based Reasoning）という．

事例に基づく推論では，①対象とする問題領域は開いた領域であること，②そこで利用できる世界モデルは不十分であってかつ不完全であること，③健全性にたいする解析や証明を与えることはできないこと等を仮定している．

これらの仮定は現実世界の領域，また我々が現実に持ち得る知識についての考察から導入されたものである．人間は複雑な領域で，しかも領域の本当の原理について深い知識を持たなくても問題解決を行うことができる．

1) 事例ベース推論の必要性

事例に基づく推論のメリットとして考えられることとしては，事例の獲得のほうがルールの獲得のよりも比較的容易であること，また類似の問題に対して同じような計算を繰り返す必要が無いなどがある．以下では事例に基づく推論の必要性について述べる．

①ルールベースシステムの限界

現在ほとんどの知識システムで用いられている方法であるルールに基づく推論では，専門家から得られた知識の正当性および一般性が検証されていることが前提となっている．しかしながら，専門家から得られる知識は断片的知識や具体的な事例であることが多く，一般的にはルールとして体系的に保持されているわけではない．したがってルールとしの知識獲得は困難な作業となっており，知識システム構築における主要な問題点の一つである．

またそのようにして獲得されたルールは，専門家にとっては常識的知識だけであることが多い．したがってそれらを知識源とする知識システムは初心者用システムであり，本当真の意味での専門家システムには程遠いのが現状であろう．

②計画問題の効率向上

物流における配送スケジューリングや，製鉄所での鋼板の板取りなどはその問題解決のため大型計算機を毎日長時間稼働させている．ほとんど条件の変わらない問題を取り扱う時でも，毎日同じような計算を繰り返しているのが現状である．

このように探索に膨大なコストがかかる問題領域では，事例を利用することにより問題解決の効率を向上させることが期待できる．

③化学的設計問題の方法論

機能性材料や触媒の開発，生理活性物質の分子設計，また界面活性剤や香料の配合設計等の分野では，経験がルールとして一般化されておらず，問題解決は成功および失敗の事例を参考に行われている．このように探索をガイドする理論や知識が確立されていない問題領域においては事例に基づく推論は唯一の方法論である．

④物理的設計問題の効率向上

[小林 88]は設計問題を，最も一般的な設計問題であるクラス1から，いわゆるルーチン設計であるクラス3の3種に分類している．このうちクラス1の問題解決における類型的タスクとして事例に基づく推論の必要性を指摘している．またクラス3における類型的タスクとしては階層的生成検査法であるとしているが，このようなルーチン設計の問題解決には，過去の設計例を修正・変更して新しい問題に適用するという事例に基づく推論も有用であると考えられる．

⑤工学的問題の質的向上

工学的領域における解釈・診断問題のような閉じた問題においても，専門家の知識のおちでルールとして一般化できない部分を，事例として蓄積し直接利用することで，問題解決の効率向上だけではなく，質的向上が期待できる．

⑥社会的問題での有用性

一般的な知識があっても事例が優先する領域は多数存在する．社会，経済，法律など非工学的分野では事例に基づく推論が基本であり，ルールに基づく推論は補助的な存在に過ぎない．

⑦同じ失敗の回避

一般に成功事例の陰には数多くの失敗事例が存在する．成功事例に加えてこれらの失敗事例についても適切にインデキシングし，蓄積できれば，同じ失敗を回避することができる．このように同じ失敗を回避できるだけでも問題解決の効率向上が期待される領域は少なくない．

2) 事例ベース推論の枠組み

事例に基づく推論の基本的なアイデアは単純である．

①過去にうまくいった事例があればそれをそのまま使う．
②失敗した事例は同じ失敗を繰り返さないように固定する．

このためには，現在の問題の特徴を理解すること，その特徴に基づいて過去の類似事例を検索し，その解を現在の問題に利用できるように変更することが基本的に必要である．これを具体化するために，事例に基づく推論の枠組みとして以下のプロセスが必要とされている．

①事例の定義（Definition）

メモリに蓄積すべき事例の構造を定義する．事例は成功及び失敗例を含む．事例をどうのように表現するかは，次の事例の特徴付けとも関連し重要な問題である．

②事例の特徴付け（Indexing）

将来の問題解決に事例を再利用するためには，事例の特徴付けをする必要がある．すなわち，事例を他の事例と区別し，後の利用に際して適切に検索できるようキーを付与しなければならない．

③事例の検索（Retrieval）

蓄積された事例の中から，現在の問題解決に最も適切な類似事例を検索するプロセスが

必要である.

④事例の変更（modification)

検索された過去の事例が，そのまま現在の問題に利用できる場合はほとんど無い．したがって，過去の事例を現在の問題へ適用可能とするために，事例の変更を行うことが必要となる.

⑤事例の修復（repair)

過去の事例は，変更を行った後現在の問題に適用を試みられるが，適用に失敗した場合は，さらに事例の修復を行うプロセスが要求される.

⑥事例の記憶（storing)

膨大な数の成功および失敗の事例を，組織化して効率よく記憶するにはどうすればよいかを扱う.

⑦事例の獲得（Aquisition)

新しい事例をどのように獲得して，蓄積していくかを扱う.

⑧信用割当（Credit Assignment)

事例に基づく推論では，現在の問題への事例適用の失敗を，事例の変更の失敗としてだけでなく問題の特徴理解の誤りとして取り扱う．したがって事例適用に失敗した場合は，同じ失敗を繰り返さないように，失敗の理由を理解し特徴付けのプロセスを修正する必要がある.

⑨事例の一般化（Generalization)

数多くの事例が蓄積された場合，類似事例を典型例として抽象化したり，事例の共通部分をルールとして一般化することができれば望ましい.

4)事例に基づく推論の流れ

上述した各プロセスを用いた事例に基づく推論の流れを図3.3-1に示した．事例に基づく

推論の基本的制御ループは以下のようになる.

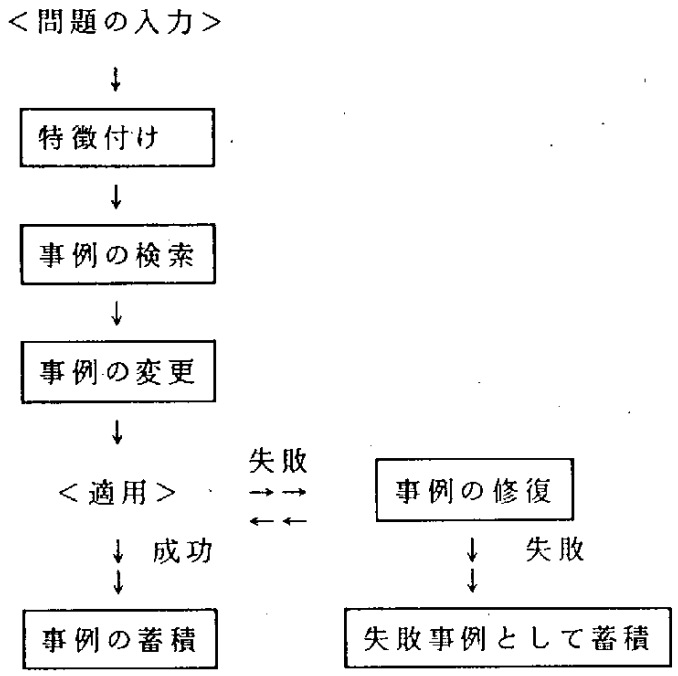


図3.3-1　事例に基づく推論の流れ

①問題の入力

②特徴付け

入力された問題を理解し，問題解決に適当な過去の事例を検索するための検索キーを設定する.

③検索

検索キーに基づいて事例ベースを検索し，最も類似している事例を選択する.

④変更

検索された事例に含まれている解が，現在の問題の目標や制約条件のすべてを満足していない場合は，解の変更を行い目標，制約条件を充足させる.

⑤現在の問題への適用

⑥事例の蓄積

問題の解決に成功したならば，その解を問題の特徴とともに新たに獲得した事例として蓄積する．

その他，事例の適用に失敗した場合には事例の修復プロセスが必要となる．

4）事例に基づく推論の問題解決における位置づけ

領域知識の一般化の程度から問題解決の方法を考えると，知識が十分に一般化されている場合はルールに基づく推論，領域知識があまり利用できない場合は汎用問題解決器による探索による問題解決が利用できる．事例に基づく推論はその中間に位置すると考えられる．

問題解決システムとしてはこれらの方法の組合せも考えらる．汎用問題解決器による探索による方法は問題解決の最後の手段として位置づけられる．ルールに基づく推論と事例に基づく推論のどちらを優先的に問題解決に適用するかは，ルールに基づく探索に必要な計算量と，事例に基づく推論において過去の事例を現在の問題に適用可能とするための事例の変更を行うのに必要な計算量との比較になる．与えられた問題とほとんど類似な事例があれば，事例に基づく推論を先ず適用すべきであろう．反対に事例の変更が容易でなければ，ルールに基づく推論を優先して適用したほうが効率的に問題解決がおこなえるであろう．一般的には事例の変更や修復のための知識を専門家から抽出することは容易でないので，その場合は図3.3-2のようになると考えられる．

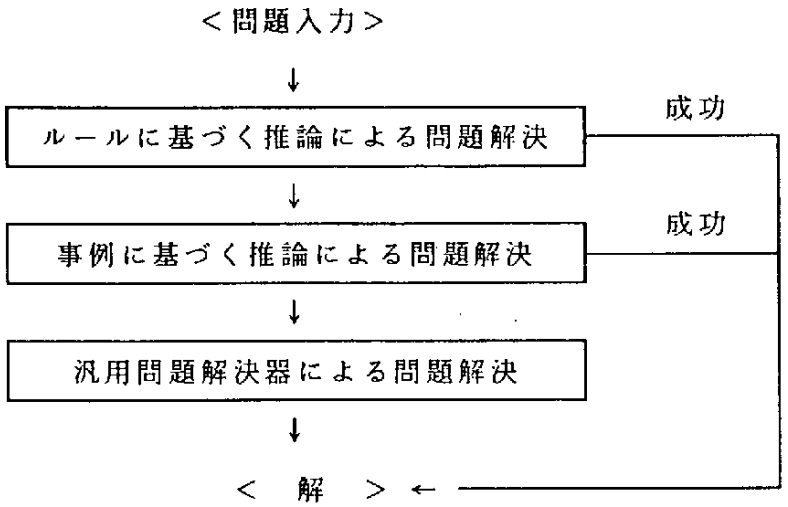


図3.3-2　事例に基づく推論の位置づけ

5)研究の現状

事例に基づく推論の基本的なアイデアであるエピソード記憶の利用については，Schankが1982年に言及しているものの，事例に基づく推論の実質的な研究の歴史は浅い．
また事例に基づく推論は多くの情報を組織化する問題や，経験から学習する問題に対して，極めて強力なアプローチであると考えられるが，現在の研究レベルでは具体的な解決法や問題解決の具体的な手順を与えるにいったていない．

事例に基づく推論の技術的基礎は学習における類推研究であるが，ここでは先に定義した事例に基づく推論の各プロセスで用いられている手法についてだけ述べる．

①事例の定義（Definition）

事例を将来の問題解決に再利用するためには，問題の特徴を表現した検索のためのキーワード部と，その問題に対する解（初期状態から解にいたるオペレータの系列）が少なくとも必要である．さらに事例の変更，修正のためには問題解決の各過程における意志決定理由を保持する必要がある．[Carbonell 88]

②事例の特徴付け（Indexing）

事例の特徴付けの目的は，事例を他の事例と区別し，問題解決に関係のない属性を排除し，さらに問題の解決に適用できる事例を検索するための索引を付与することであり，事例の検索とも密接に関連している．事例に基づく推論においては，過去のどの事例を基にして変更を行うかによって，問題解決のための負荷が大きく異なる．不適当な事例を基に変更を行うのに必要な負荷は，ルールに基づく推論で誤ったルールを選択した時よりも大きい．したがって過去の事例を有効利用するための特徴付けは，極めて重要である．しかしこれまでに報告されたシステムの多くはこのプロセスをユーザに依存しており，事例の特徴付けの自動化に関するものは少い．

事例の特徴付けに要求される事項としては，問題に関係のある特徴であること，一般化されていること，過剰一般化されていないことが必要である．機械の故障診断を例に説明すると，ある故障が温度３０度で発生した場合，その温度が故障と関係ある時はそれを特徴としなければならない．関係があったとしても特徴とするには，ある温度の範囲内でその故障が起こるとして温度の適当な一般化を行わないと，その事例は温度がちょうど３０度である時にしか適用可能とみなされなかったり，過剰一般化の場合はどの温度でもその事例が検索されてしまうことになる．

インデキシングの自動化は基本的には学習問題である．アプローチの方法には，帰納的な方法と説明に基づく方法についての報告がある．

帰納的な方法の問題としては，帰納によりインデキシングを行うためにはそれ以前に数多くの事例を獲得しておく必要があること，問題解決に無関係な属性までも特徴としてしまうことがあげられる．

[Barletta 1988]ではEBL（説明に基づく学習）を利用して機械の故障診断事例のインデキシングを行っている．不完全な領域知識（故障原因に無関係な属性までも特徴としていしまうような知識）をEBLで洗練化（無関係な属性は排除する）する方法について論じている．この方法では，故障原因に関係ある属性を説明する知識は領域知識としてすべて予め与えられていなければならない．すなわち故障に無関係な属性の排除は行うが，領域知識として与えられていない属性の取り込みは行わない．

③事例の検索（Retrieval）

事例の検索は，与えられた問題から特徴抽出した特徴付けを利用して事例ベースを検索

し，過去の事例と照合を行い，類似事例を選択することが目的である．照合には3レベルがある．

・直接照合

同じ特徴付けを持つ事例は照合する．

・概念階層を利用する照合

概念階層（検索キーワードのシソーラス）を予め定義しておき，それを利用して検索を行う．

・類比照合

このレベルの照合には一般的な手法はまだない．類似性尺度の導入[Hammond 86]や因果関係構造の利用[Navinchandra 88]が提案されている．

このプロセスは事例の変更，修正とともに学習における類推（analogy）研究の進展を待つところが多い．

④事例の変更，修正

事例の変更，修正には領域知識を利用している報告が多い．したがって取り扱う領域に依存するところが多く，手法として一般化されてはいない．

[Carbonel 88]は類推における誘導形類推（derivational analogy）の立場から，事例として問題の特徴とその解を記憶しておくだけでは不十分であり，問題解決過程の各段階での意志決定理由を保持しておくことによって変更，修正を行うことを提案している．

[Sycara 88]では事例の変更，修正の際に領域知識をもちいるだけではなく，ベースとなる過去の事例をヒューリスティクスによって変更した後，変更によって生じる副作用的な問題点を予想し，その問題点の解決のためにまた過去の事例を用いる方法を提案している．このような方法はCYCLOPS[Navinchandra 88]でも使われているが，いずれのシステムでも事例を変更することで生じる新たな問題点の解決するための手続きを付加するだけであり，すでに変更を施した部分を再度変更するものではない．

以上事例に基づく推論の技術について述べてきたが，基本的プロセスは提案されてはいるものの，研究の歴史は浅く技術といえるものは少ない．事例に基づく推論における主要

な研究課題をとしては以下の項目があげられる.

・いかに事例を表現するのか

・いかに類似の事例を検索するのか

・いかに現在の問題に適用できるように変更，修正するのか

・新たな事例の蓄積に伴い事例の構造自体をどうダイナミックに変化させるか

7)応用事例

事例に基づく推論を利用した応用としては，研究段階であるが次のようなものがある.

・分類型問題

法律判例 [Ashley 87]

機械故障診断 [Barletta 88],[Hammond 88]

音声合成

・設計，計画型問題

プランニング [Carbonell 88],[Hammond 86]

景観設計 [Navinchandra 88]

自動プログラミング [Williams 88]

【参考文献】

[小林 88]

小林重信

知識システム技術の現状と将来

計測と制御,vol.27,10, pages 859-868,1988

[Ashley 87]

Ashley,K.,Rissland,E.

Comapre and Contrast,a Test of Expertise

In Proceedings of AAAI-87,1987

[Barletta 88 a]

Barletta,R., Mark,W.

Explanation-Based Indexing of Cases

In Proceedings of DARPA Workshop on Case-Based Reasoinig,1988.

[Barletta 88 b]

Barletta,R., Mark,W.

Explanation-Based Indexing of Cases

In Proceedings of AAAI-88, 1988.

[Carbonell 88]

Carbonell,J.,Veloso,M

Integrating derivational analogy

into a general problem solving architecture

In Proceedings of DARPA Workshop on Case-Based Reasoinig, 1988.

[Hammond 86]

Hammond,K.J.

CHEF:A Model of Case-Based Planning.

In Proceedings of AAAI-86, pages 267-271. 1986.

[Hammond 88]

Hammond,K.J.,Hurwitz,N.

Extracting Diagnostic Features form Explanation

In Proceedings of DARPA Workshop on Case-Based Reasoinig, 1988.

[Navinchandra 88]

Navinchandra,D.

Case Based Reasoning in CYCLOPS,a Desing Problem Solver

In Proceedings of DARPA Workshop on Case-Based Reasoinig, 1988.

[Sycara 88]

Sycara,K.

Using Case-Based Reasoning for Plan Adaptation and Reair

In Proceedings of DARPA Workshop on Case-Based Reasoinig, 1988.

[Williams 88]

Williams,R.,S.

Learning to Program by Examinig and Modifying Cases

In Proceedings of DARPA Workshop on Case-Based Reasoning, 1988.

（3.3節担当　関根）

3.4 学　習

本節では，計画・設計問題における知識獲得的側面を支援するための基盤技術として，学習の理論と方法について，特に，説明に基づく学習に重点おいて，現状を紹介する．

3.4.1 学習研究の概観

学習とは，「対象の理解や問題解決に必要な知識を獲得し，技能を習熟していく過程」と定義される．学習の前半の過程は知識獲得(Knowledge Acquisition)，後半の過程は技能洗練化(Skill Refinement)と呼ばれる．学習は，帰納的学習，演繹的学習および発見的学習の３つに分類される．

(1)帰納的学習(Inductive Learning)

帰納的学習は，類似性に基づく学習(Similarity Based Learning)とも呼ばれる．例題からの学習(Learning from Examples)は，帰納的学習における基本形であり，外部の教師が訓練集合および各例題が属するカテゴリーを陽に与える．バージョン空間法はその代表例であり，負例を排除し，正例を含む一般化ルールを帰納してこれをバージョン空間上に維持することにより，概念の獲得を行なう．

観察からの学習(Learning from Observation)では，訓練集合の設定は学習システム自身の判断に委ねられ，例題間の差異や類似性を分析することにより，例題集合を自律的に探索することが求められる．ＩＤ３的接近法は，観察からの学習の代表例であり，訓練集合を自律的に更新しながら，最適な決定木を同定する．

コネクショニストによる学習(Connectionist Learning)は，認知処理レベルでの帰納的学習であり，ニューラルネットの自己組織化能力に基づいて例題からの学習や観察からの学習を行なうもので，逆伝搬学習や競合学習などの学習アルゴリズムが提案されている．

(2)演繹的学習(Deductive Learning)

教示からの学習(Learning from Instruction)は，演繹的学習のもっとも素朴な方法であり，教師が与えた教示や助言に基づいて有用と認められた情報を知識に変換するものである．

説明に基づく学習(Explanation Based Learning)は，演繹的学習における新しいパラダイムとして，注目を集めている．説明に基づく学習では，領域理論，目標概念および操

作性規範が事前に定義され，目標概念の具体例として１つの例題が与えられると，その例題が目標概念の具体例であることを領域理論を使って証明することを試みる．ついで証明木を一般化することにより，操作性規範を満たす操作的かつ一般的な知識を導出する．説明に基づく学習は，新しい知識の獲得手法というよりは，むしろ既存知識の洗練化手法と考えるのが妥当である．

再定式化による学習(Learning by Reformulation)は，知識洗練化の別の手法である．再定式化による学習では，問題解決の分野を主たる対象として，問題解決における探索過程を分析して，チャンキングや知識コンパイルと呼ばれる手法を用いてマクロオペレータの生成を行なう．再定式化による学習と説明に基づく学習は，その方法論は見掛け上異なるものの，本質的には同じことを学習しており，いずれも問題解決の性能の向上に寄与する知識を求める点において共通しており，両者の関係は密接である．

(3)発見的学習(Heuristic Learning)

発見的学習とは，発見的方法により新しい概念や法則の発見を行なうもので，演繹や帰納の枠を超える飛躍を伴う．

観察による発見(Discovery by Observation)では，例題集合を分析することにより，背後に成立しているとみられる法則や原理を推測するもので，理論形成とも呼ばれる．

実験による発見(Discovery by Experiment)では，学習システム自身が自律的に実験を計画し，新しい知識の発見を行なうことを目的とする．

類比による学習(Learning by Analogy)では，基準対象と目標対象間の類似性を利用して，基準対象に成立する関係や過程を目標対象に写像することにより，帰納や演繹を行なわずに，一挙に問題解決を行なうことを目的としている．

帰納的学習，演繹的学習，発見的学習は，それぞれ人間の問題解決過程に観察される帰納(induction)，演繹(Deduction)，発想(Abduction)に対応するものである．

学習研究の発展段階としては，帰納的学習に関する研究が先行したものの，その限界が認識され，現在は演繹的学習の研究，特に説明に基づく学習の研究が盛んである．発見的学習については，類推に関する研究が事例に基づく推論との関連で注目を集めいる．

3.4.2 バージョン空間法

バージョン空間法[Mitchell,1977]は，外部の教師から逐次的に与えられる例題から目

標とする概念を獲得することを目的とする．各例題は属性（特徴）記述および正例または負例のいずれかに属することが与えられる．例えば，

例１：（脊＝低い，髪＝ブロンド，目＝青；クラス＝正）

例２：（脊＝高い，髪＝黒髪，　　目＝青；クラス＝負）

のように与えられる．

バージョン空間法では，学習の対象である目標概念は属性の連言形（ａｎｄ結合）で表現されるクラスであることが仮定される．このように目標概念のクラスを陽に制約する条件のことをバイアス(bias)と呼ぶ．目標概念が属性の連言形に制約されることから，学習システムが探索すべき概念空間は概念間の半順序構造を構成する．

バージョン空間法における学習とは，すべての負例を排除する概念の中でもっとも極大なもの(Most General Version;ＭＧＶ)ともっとも極小なもの(Most Specific Version;ＭＳＶ)を維持，管理することであり，ＭＧＶ＝ＭＳＶとなったとき，学習が完了する．ＭＧＶおよびＭＳＶによって張られる概念空間の半順序構造のことをバージョン空間という．

バージョン空間を継続的に維持するための学習アルゴリズムとして，候補消去アルゴリズム(Candidate Elimination Algorithm)がある．

候補消去アルゴリズム

1)正例が入力されると，

①正例をカバーするＭＧＶ要素をバージョン空間から除去する．

②正例とＭＳＶ要素の間で一般化を行ない，ＭＳＶを更新する．

2)負例が入力されると，

①負例をカバーするＭＳＶ要素をバージョン空間から除去する．

②負例とＭＧＶ要素の間で特殊化を行ない，ＭＧＶを更新する．

バージョン空間法を応用した学習システムとして，ＬＥＸ[Mitchell et al, 1983]がある．ＬＥＸは不定積分の問題を解くためのヒューリスティックスを獲得することを目的とするもので，つぎのようなステップで学習を行なう．

ＬＥＸの学習手順

1)与えられた問題に対し，適切なヒューリスティックスがないとき，

利用可能なオペレータを順次適用して試行錯誤的に問題を解く．

2)与えられた問題に対し有用であったオペレータを正例とみなす．

3)あらかじめ設定された概念の汎化階層（バイアス）を利用して，
オペレータに対するバージョン空間を生成する．
4)新しい問題を解くことにより，バージョン空間を更新する．
（バージョン空間の更新は候補消去アルゴリズムに従う）．
5)バージョン空間が収束（ＭＧＶとＭＳＶが一致）すれば，
これを新しいヒューリスティックスとして登録する．
1)～5)のステップを繰り返す．

バージョン空間法は非常にエレガントな帰納的学習法であるが，つぎの問題点を有する．

1)バイアスが強すぎること

目標概念が属性の連言形であることは，制約として余りにも強すぎる．目標概念が属性の選言形で表現される場合にも適用可能とするために，バージョン空間を複数個設定し，これらを並列に維持する方法[Mitchell,1978]が提案されているが，バージョン空間法の限界を克服するものではない．

2)ノイズに対して頑健でないこと

ノイズが存在する状況下で，バージョン空間法をそのまま適用すると，ＭＳＶ要素がＭＧＶ要素よりも一般的となってしまうことが起り得る．このような問題を回避するために，ノイズの程度に応じてＭＧＶ要素とＭＳＶ要素を多重化してバージョン空間を維持する方法[Mitchell,1978]が提案されているが，バージョン空間の管理を複雑にする恐れがある．

3)大規模問題には対処できないこと

属性数が２桁を超えると，バージョン空間法は組合せ的な爆発の問題を招き，バージョン空間を維持することが事実上困難になる．

このように，は“elegant, but not robust”な方法であるが，その考え方は有用であり，演繹的学習と統合することにより，その欠点は克服することが期待できる．

3.4.3 説明に基づく学習

バージョン空間法に代表されるＳＢＬ(Similarity-Based Learning)では，一般に，多くの訓練事例を必要とし，さらに帰納的飛躍(Inductive Leap)を起こすために強いバイアスを導入することを余儀なくされる．バイアスは探索すべき概念空間に制約を課すとともに，意味のある一般化を行なわせる効果をもつ．しかし一般化の正当性を保証する根拠と

しては弱いものである．ＳＢＬの欠点を克服する試みとして，学習すべき目標概念に関する背景知識を積極的に利用してはどうかという考えが生じる．ＥＢＬはこのような考えに基づくきわめて新しい学習の方法である．ここでは，知識洗練化手法としてのＥＢＬの特徴や課題を考察する．さらに，ＥＢＬを拡張するいくつかの試みを紹介する．

ＥＢＬはつぎのような枠組みをもつ[Mitchell et al, 1986]．

＜given＞

①目標概念(Goal Concept)

学習すべき概念の抽象的な記述

②訓練事例(Training Example)

目標概念の具体例

③領域理論(Domain Theory)

訓練事例が目標概念の具体例であることを証明するための知識

④操作性規範(Operational Criterion)

学習によって得られる概念の有用性を判定する基準

<determine>

操作性規範を満足する一般的な知識の生成

ＥＢＬにおける学習はつぎの２つのステップから構成される．

1)説明(Explanation)

訓練事例が目標概念の具体例であることを領域理論を用いて証明する．

証明木は説明構造と呼ばれる．

2)一般化(Generalization)

説明構造に対し，目標概念を逆向きに回帰することにより，一般化の操作を施し，操作性規範を満足する一般的なルールを抽出する．

次ページにＥＢＬの動作例[Mitchell et al, 1986]を示す．①の目標概念で記述されている cup(x) の記述は，機能的な記述(liftable, stable, open_vessel)であることに注意されたい．②の訓練事例は構造的な記述で表現されている．③の領域理論は機能的な記述と構造的な記述の関係を表現している．④の操作性規範は構造的記述の範囲を規定している．⑤の説明構造は領域理論を用いて，訓練事例が目標概念の具体例となっていることを示している．⑥の一般化概念は説明構造に対し一般化を施して得られたものである．

①目標概念

```
cup(x) <==>  liftable(x) & stable(x)
           & open_vessel(x)
```

②訓練事例

```
owner(Obj1, Edger)
color(Obj1, Red)
part_of(Obj1, Concavity-1)
is_a(Concavity-1, Concavity)
is(Concavity-1, Upward_Pointing)
part_of(Obj1, Bottom-1)
is_a(Bottom-1, Bottom)
is(Bottom-1, Flat)
part_of(Obj1, Handle-1)
is_a(Handle-1, Handle)
length(Handle-1, 5)
part_of(Obj1, Body-1)
is_a(Body-1, Body)
is(Body-1, Small)
```

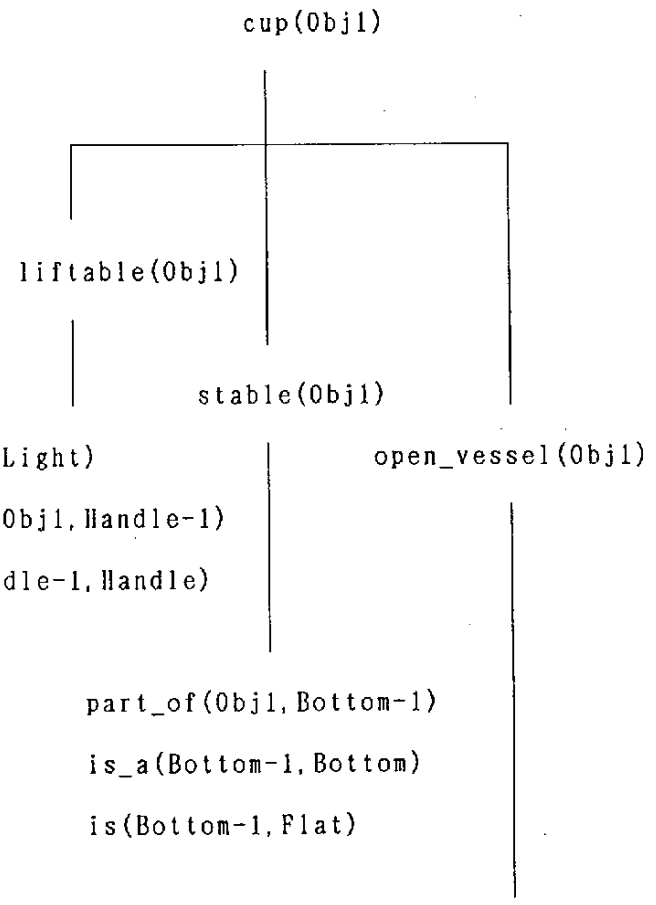
③領域理論

```
  is(x, Light) & part_of(x, y)
& is_a(y, Handle)   ==> Liftable(x)

  part_of(x, y) & is_a(y, Bottom)
& is(y, Flat)   ==> Stable(x)

  part_of(x, y)
& is_a(y, Concavity)
& is(y, Upward_Pointing)
  ==> Open_Vessel(x)
```

④操作性規範

構造的記述（Light, Handle, Flat, Concavity, Upward_Pointing, Bottom）を用いて，目標概念を記述せよ．

⑤説明構造



⑥一般化概念

```
part_of(x, xc) & is_a(xc, Concavity)
& is(xc, Upward_Pointing) & part_of(x, xb)
& is_a(xb, Bottom) & is(xb, Flat) & part_of(x, xh)
& is_a(xh, Handle) & is(x, Light)   ==> cup(x)
```

説明構造を一般化するために，cup(Obj1) における定数 Obj1 を変数 x に置き換える操作が行なわれる．この操作は目標概念から逆向きに再帰的に行なわれる．その結果，定数 Concavity-1, Bottom-1, Handle-1 はそれぞれ xc, xb, xh に変数化される．

ＥＢＬにおける諸問題を以下に議論する．

一般化問題

説明構造に基づいてこれを一般化する方法には，いくつかのバリエーションが存在する．[Mitchell et al,1986]らの目標回帰(goal regression)による方法は過少一般化の恐れがある．[DeJong and Mooney,1986]らの単一化の解除(unification retraction)はより妥当な一般化をもたらすが，決定的とはいえない．いずれも，一般化は定数を変数に置き換える変数汎化に留っているが，一般化を説明構造の下部を裁断する構造汎化［山村，小林,1988]にまで拡張すれば，１つの例題から任意個の一般化を生成することが可能で，しかも演繹の範囲を超えることはない．

操作性規範問題

[Mitchell et al,1986]らは，操作的と呼ばれる述語のクラスを事前に定義している．[DeJong and Mooney,1986]らは，操作性規範は固定すべきではなく，実行時に動的に設定すべきであると主張している．[Keller,1987]は，操作性規範には利用可能性(usability)と効用(utility)の２つの側面があることを指摘している．

操作性規範については，学習という閉じた世界を考える限り，これを一般的に議論することはできても，陽に定義することは困難と思われる．操作性規範は問題解決システムからの要請に応じて決定されるべき性質のものと考える．

不完全性問題

ＥＢＬでは，その前提として，領域理論は訓練事例が目標概念の具体例であることを証明するのに十分であることを仮定している．このことは説明構造の一般化によって知識を生成することが領域理論に全く新しい知識を加えることにならず，マクロルール化された知識を加えることを意味する．ＥＢＬが知識洗練化の手法であって，知識獲得の手法ではないといわれるのはこの理由による．しかし，領域理論が不完全な場合には，知識洗練化において注意を必要とする．

不完全性問題はいくつかの副問題に分けて考えることもできるが[Mitchell et al,1986]

大局的に言えば，目標概念に対して，正例であるにもかかわらず，説明することができない場合および負例であるにもかかわらず，説明してしまう場合とに分けられる．前者の場合，領域理論を一般化する必要があり，後者の場合，領域理論を特殊化する必要がある．

ＥＢＬとＳＢＬの融合

ＥＢＬは１つの例題から領域理論を用いて演繹的な学習を行なうのに対し，ＳＢＬは複数の例題からバイアスの制約の下で帰納的な学習を行なうものであり，前者は領域理論が豊富な場合に適しており，後者は例題集合が豊富な場合に適しているといえる．したがって領域理論がほどほどにあり，かつ例題集合もほどほどにある状況のもとでは，ＥＢＬとＳＢＬを融合して利用することは当然考えられる[Mitchell et al,1986]，[山村，小林(1989)]．ＥＢＬとＳＢＬの融合にはいろいろな形態が考えられるが，まだ真の意味での融合は達成されているとは言えないが，いくつかの先駆的試みがなされている[Bergadano, F. & Giordana, A.(1988)]，[Dietterich, T.G. & Flann, N.S.(1988)]．特に，領域理論が不完全な場合には，ＳＢＬ的接近法を適切に導入することにより，領域理論の洗練化を行うことができる[白井，小林(1989)]．

ＥＢＬは計画・設計問題における知識獲得的側面を支援する基盤技術として極めて有望な枠組みを有しており，今後の技術的発展に大いに期待できる．

参考文献

[Bergadano, F. & Giordana, A.(1988)]

A Knowledge Intensive Approach to Concept Induction,

5-th Int. Conf. on Machine Learning, 305/317.

[DeJong, G. & Mooney, R.(1986)]

Explanation-Based Learning: An Alternative View, Machine Learning, Vol. 2, 145/176.

[Dietterich, T.G. & Flann, N.S.(1988)]

An Inductive Approach to Solving the Imperfect Theory Problem,

AAAI Spring Symposium Series: Explanation-Based Learning, 42/46.

[Keller, R.M.(1987)]

Defining Operationality for Explanation-Based Learning, AAAI-87, 482/488.

[Mitchell, T.M.(1977)]

Version Spaces: A Candidate Elimination Approach to Rule Learning, 5-th IJCAI, 305/310.

[Mitchell, T.M., Keller, R.M. & Kedar-Carbelli, S.T.(1986)]

Explanation-Based Generalization: A Unifying View, Machine Learning, Vol. 1, 47/80.

[山村, 小林(1989)]

ＥＢＬの複数例題下への拡張, 人工知能学会誌, Vol. 4, No. 4（掲載予定）.

[白井, 小林(1989)]

複数の説明構造の階層化と変数汎化に基づく知識の獲得と洗練化, 第10回知識工学シンポジウム, 計測自動制御学会

（3.4節担当 小林重信）

4．計画・設計型エキスパートシステムへの接近

4. 計画・設計型エキスパートシステムへの接近

4．1　問題解決的接近

4．1．1　問題解決モデルの同定

問題の具体例が与えられたとき，それがそのまま教科書的事例に引き写せることはまずない．実用化を目指すシステムであれば，その問題は現実世界の複雑さを直接反映していることが多く，世の中に同じ問題は二つとないようにも思われる．しかし実際には問題を分析し部分問題に分割することにより，他の典型的問題に類似した部分が抽出され得る．

問題を解決するためには，当然のことではあるが問題自体を理解しなければならない．その後，あるいは並行して問題の性質に基づいて，部分問題への分割や解法を適用するための分析が行なわれる．前者を問題認識，後者を問題分析としてそれらの内容を次の図に示す．

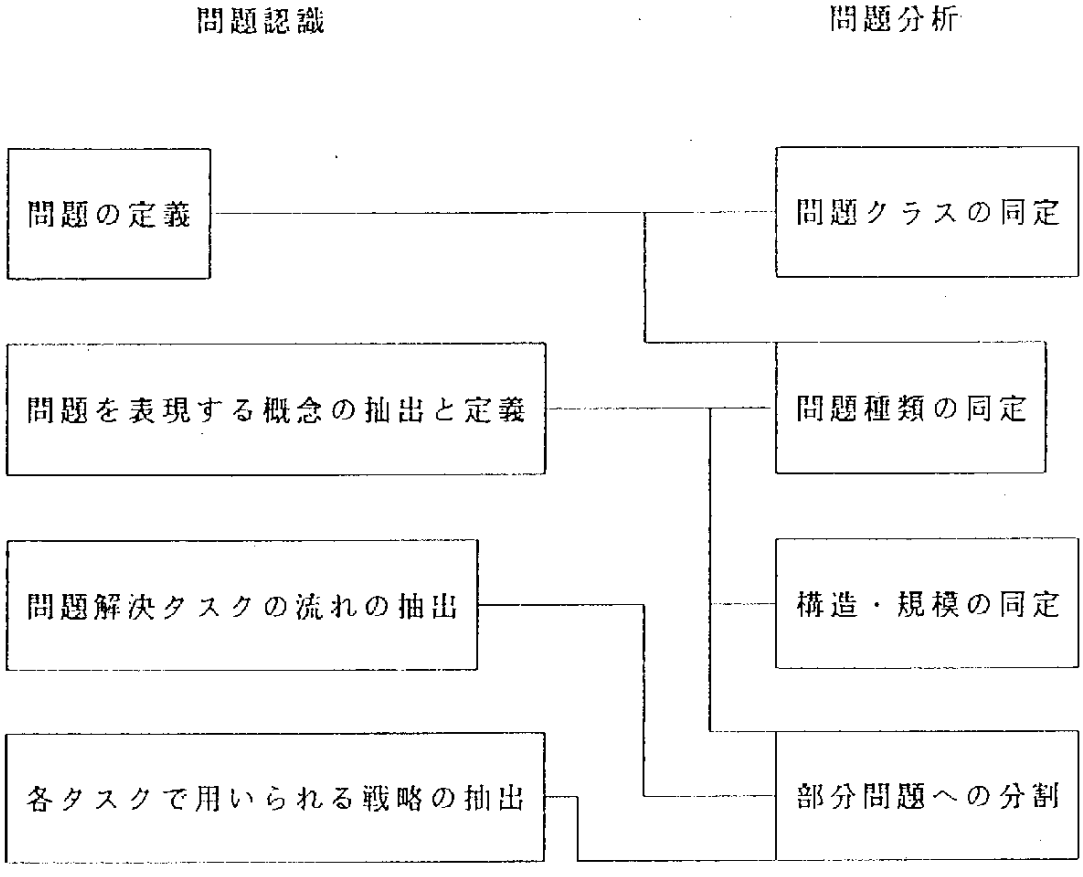


図　4.1－1　問題認識と問題分析

上図は問題認識と問題分析とが並行して，基本的には上から下へ流れる様子を表わしている．また，認識と分析と相互に関連する項目を結んでいる．以下では各要素について考察する．

（１）問題の定義

システム化しようとしている問題が全く新規なものであるか，あるいは既存の各種の問題の色々な側面が関連して現われたものであるかを判定するために行なう．定義といっても最初から文章で厳密に表現されることは少ないから，具体例の提示・専門家による問題の解説・見学・デモ等を通して問題のイメージを把握することがまず必要になる．この段階で少なくともその問題が分類・診断型と計画・設計型のいずれに属するか，両方に属するならどちらの比重が大きいかを見極めなければならない．本稿は計画・設計型を考察しているので，以下では問題が設計・計画型であるとする．

（２）問題クラスの同定

ここでは問題がクラス１，２，３［小林　87］のいずれに属するかを決定する．クラス１は計画・設計されるものの構造・構成要素が未知であり創造的性格が強く，システム化する場合には意思決定や仮説設定などの高度な機能が要求される．クラス２は構造未知・要素既知であり材料を用いて用件を満足するものを作るというタイプで，現実の問題としては事例が多い．クラス３は構造既知・要素未知であり，どの構成要素を選択するかという観点で構造の各パラメタをを決めるタイプの問題である．比較的ルーチンワーク業務に多く，そのため過去の事例を参照する場合も多いと考えられる．仮説推論・事例ベース推論は他稿で解説されるので，以下では問題がクラス２であるとする．クラス１，２，３を図４．１－２に示す．

（３）問題を表現する概念の抽出と定義

新しい問題に接したときは，問題を表現する用語でさえ理解困難なことが多い．しかし計画・設計問題では，それらの用語や概念は共通なカテゴリに属するものがあると考えられる．それらのカテゴリの例を図４．１－３に示す．用語や概念をカテゴリに分類することにより，問題全体を俯瞰する役に立つ．

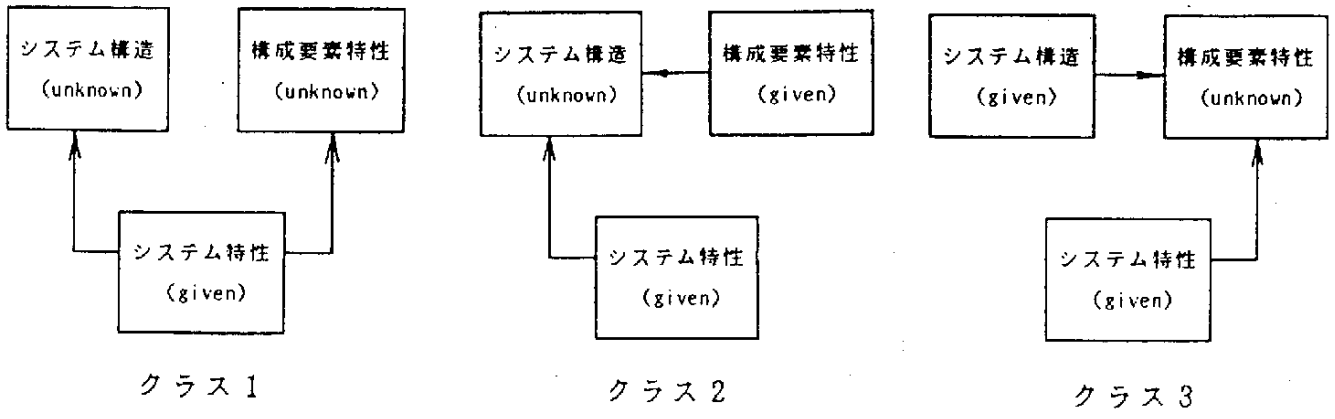


図 4.1－2 クラス1，2，3

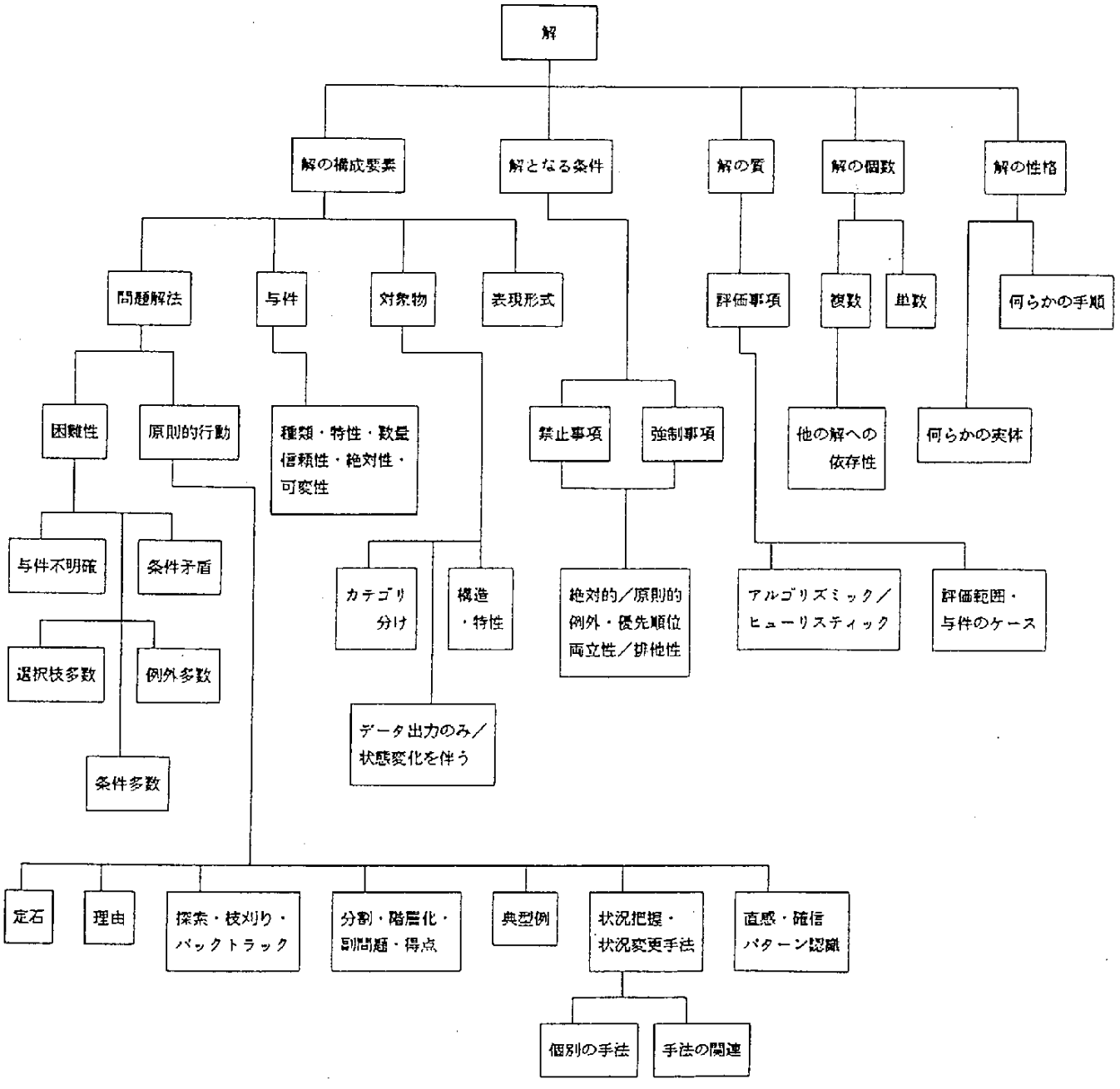


図 4.1－3 概念が属するカテゴリの例

（４）種類の同定

問題の全体がほぼ把握できた段階で，その問題がテキストレベルでの各種の典型的問題の性質を含むかを調べる．何種類かの典型例の複合であることは当然考えられる．クラス２の計画・設計問題の場合，典型例としては以下のようなものが挙げられる．

①初期状態から最終状態への遷移系列の決定（例：宣教師と人食い人種）

②空間への要素配置（例：時間割り，８クイーン，ビンパッキング，レイアウト）

③ネットワーク経路（閉路）作成（例：最短経路，オイラー閉路，ハミルトン閉路）

これらは規模は小さくとも問題が純粋な形で現われているため，内容・解法も理解し易い．実際の問題の本質的部分を抽出したとき，これらに似ているかがポイントになる．また，典型例でなくとも社内開発事例集なども参考にすべきである．

（５）構造・規模の同定

問題の詳細な構造を把握し，最大規模を推測する段階である．これらが単純・小規模ならば典型例と類似の解法が使えるかもしれない．しかし一般には構造は複雑で規模も大きいことが多い．したがって部分問題に分割することが必要であり，逆に，分割するためには問題構造を明確化しておく必要があることも明らかである．また，ここで規模を問題にする理由は，小規模な問題を短時間に解く解法が，規模が増加すると実用時間で解くことができなくなる可能性があるためである．部分問題に分割することによって問題規模の増加をおさえ，したがって小規模問題の解法の適用可能性を残しておくことが望ましい．参考までに，問題規模をパラメタｎで表わし，各解法が問題を解くに要する時間をｎの関数で表わすとき，ｎの増加に対して解法によってどれだけ差がつくかを表４．１－１に示す[Garey,Johnson 79]．

（６）問題解決タスクの流れの抽出

問題規模が大きいほど，専門家は無意識にせよ問題解決をいくつかのタスクとして分割している場合が多い．計画問題なら制約条件の優先度の順に計画を詳細化したり，規模を縮小して「あたり」をつけたりする等である．設計問題でも，局所的・階層的に設計を進めることが見出される．このように作業を複数のタスクの流れとしている理由は，各タスク毎に問題を限定し部分問題を解いているためである．タスク毎に問題に対する視点が変わり，比喩的にいえば３次元の物体を２次元に投影して扱っているといえる．

表　4.1－1　多項式解法と指数解法の比較

Time complexity function	Size n					
	10	20	30	40	50	60
n	.00001 second	.00002 second	.00003 second	.00004 second	.00005 second	.00006 second
n^2	.0001 second	.0004 second	.0009 second	.0016 second	.0025 second	.0036 second
n^3	.001 second	.008 second	.027 second	.064 second	.125 second	.216 second
n^5	.1 second	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	13.0 minutes
2^n	.001 second	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 second	58 minutes	6.5 years	3855 centuries	2×10^8 centuries	1.3×10^{13} centuries

（７）各タスクで用いられる戦略の抽出

専門家は自分の流儀にこだわるものである．計画・設計作業の各タスクにおいて，行なうことが大抵決まっており，それが個々の計画・設計対象によって変化しないものであれば，それは戦略レベルまで抽象化されたものと考えられる．それらを抽出することは必ずしも容易ではないが，例として空間配置問題に帰着された問題で抽出された戦略の一部を挙げる．この例では配置要素の形状・大きさはほぼ同一である．

①優先度の高い要素・制約から実行する　：　階層化
②過去の事例からのデフォルト的位置に配置する　：　類似例の検索
③配置要素をパタン化（グループ化）する　：　パタン単位の配置
④９割程度の要素をとりあえず配置して「あたり」をつける：　実行可能性のチェック
⑤最終結果の概略イメージを予測する　：　過去の類似例での学習
⑥等間隔的に初期配置する　：　制約条件の平均化
⑦影響が広範囲に及ぶ修正はしない　：　修正影響の局在化
⑧「混んでいる」領域の要素を「空いている」領域へ移動　：　配置密度のパタン認識

⑨優先度の低い制約条件の変更　　：　制約条件の緩和

⑩最初からやり直すときは与件をラジカルに変更する　　：　上位レベル後戻り

（８）部分問題への分割

これまで考察してきたことは，最終的には問題を部分問題に分割することを目的にしている．部分問題に分割することによって問題の内容を既存の問題として表現し，既存の解法を応用し，さらに問題規模の増加を抑えることによりプリミティブな解法（単純な探索など）の適用余地を残せる可能性がでてくる．最後に，各サブタスクも更に分割または階層化するときに着目する観点の例を表４．１－２に示す［松本　87］．

表　4.1－2　問題分割の観点

観　点	レベル分け	応　用　例
抽象度	抽象的　⇨　具体的	・生物分類 生物→動物→脊椎動物→哺乳類→人間
複合度	複合体　⇨　単体	・ダイヤ編成 ループの決定→経路の決定→路線の決定
制約	強い制約　⇨　弱い制約	・生産計画 機種割り付け→号機割り付け
頻度	一般的　⇨　特殊	・計算機構成設計支援 一般的な構成例→ユーザ用件の反映

4．1．2　探索問題としての定式化

計画・設計型問題を探索をベースとした解く場合は，問題全体に対して高度な探索方式を適用するよりも，問題をできるだけ多くの互いに独立な部分問題に分割し，更に各々の部分問題を階層化して各階層でシンプルな生成検査に基づく探索を行なうほうが良いと思われる．部分問題の規模が十分小さくなり，階層化が適切になされていれば単純な深さ優先・広さ優先探索でも役に立ち，またそれだけ小規模になれば単純な探索問題として定式化することは容易である．問題の認識と分析とを通して問題を分割し，構造を単純化し，規模を縮小し，特殊と思われていた問題を典型例に帰着させることが，計画・設計のような従来から困難とされてきた問題をエキスパートシステム化する場合の基本的立場と考えられる．

具体的な問題に対する問題分割と探索問題への定式化についての例が5．5節に述べられている．

文献：

[小林　87]　知的情報処理システムに関する調査研究報告書・知識システム開発方法論　日本情報処理開発協会，pp.55-64.

[Garey,Johnson 79]　Computers and Intractability, A Guide to the Theory of NP-Completeness, Freeman.

[松本　87]　ES/SDEM，富士通㈱.

（4.1節担当　中島）

4.2 仮説推論的接近

設計や計画における人間の問題解決・推論の過程は，要求仕様として与えられた事実を説明する仮説を生成する過程，得られた仮説から論理的に導出される帰結を得る過程，および，そのようにして得られた帰結を何らかの実験あるいはシミュレーションによって検証し，最初に生成した仮説を検定する過程からなると考えることができる．この設計や計画の過程には，多くの経験的知識と専門的知識を必要とすることはだれしも認めるところであるが，これらの知識は，どの分野においても，十分に体系づけられていないのが実状であり，人間は種々の知的技法を用いて，何らかの意味のある仮説を生成したり，合理的な仮説検定を行っている．このような過程の表現に，仮説推論の枠組みが利用できる．

1）設計問題の仮説推論問題としての定式化

まづ，論理に基づく仮説推論システムの枠組みについて述べる［松田　88］．知識ベースを，事実(fact)の知識（対象世界で常に成り立つ知識）の集合Ｆと，仮説(hypothesis)の知識（対象世界で常に成り立つとは限らない知識）の集合Ｈとに分ける．Ｈの部分集合をｈ，すなわちｈ⊆Ｈとする．ある観測Ｏが与えられたとき，次の条件を満たすｈを求めることが仮説推論の基本動作である．

$$F \cup h \vdash O$$

$F \cup h$　は無矛盾

すなわち，事実の知識の集合Ｆだけから Ｏを証明できればそれでいが，Ｆだけでは Ｏを証明できないとき，仮説の集合Ｈの中から無矛盾な部分集合ｈを切り出し，Ｏを証明するように動作する．図4.2-1はこのような仮説推論の動作を図示したものである．

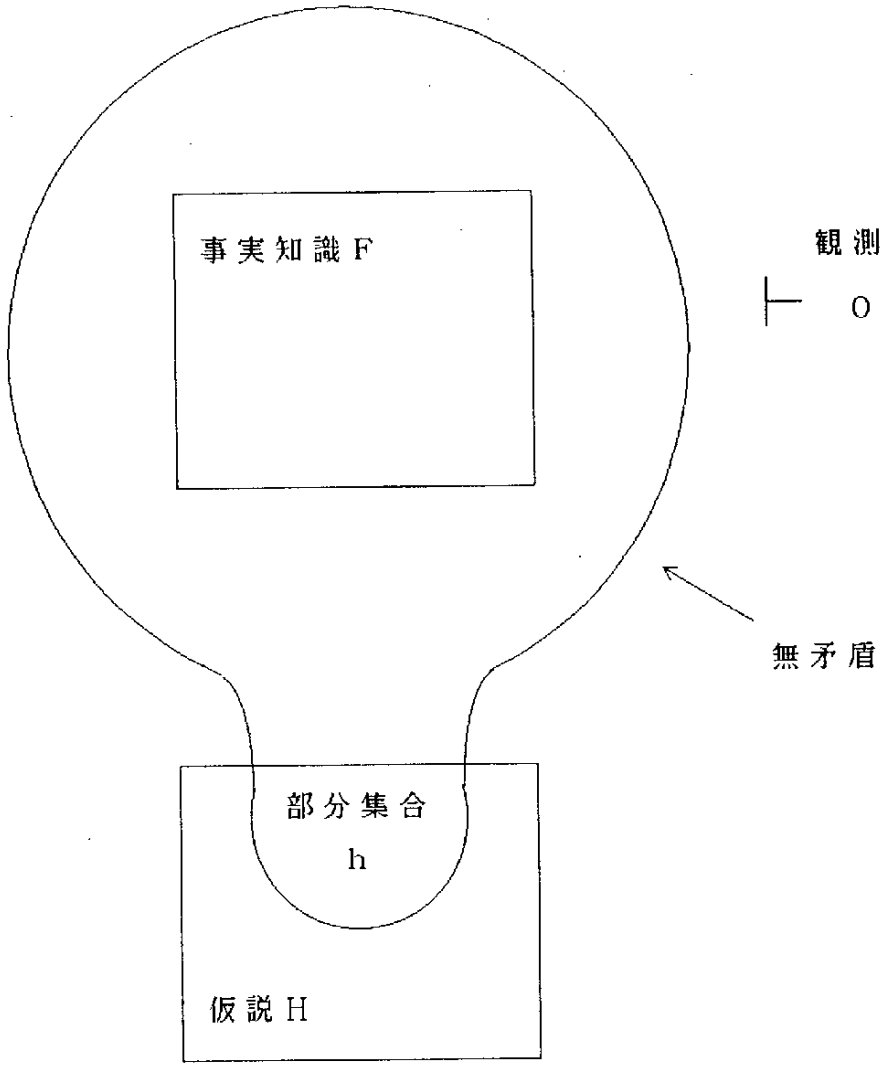


図4.2-1　仮説推論システム

使用可能な部品，実現可能な構造を仮説として与えることにより，このような枠組みに基づいて，仮説推論からのフォーマルな定式化を行うことができる．このようなアプローチにResidueのアプローチ［Finger　85］がある．Residueシステムはロボットのプラニング，回路設計，回路のパラメータ設計やプログラムの合成と行った問題を解決する演繹的な解の合成という考えに基づく設計システムである．このシステムは仮説推論の枠組みが設計システムにどのように適用できるかを示している．

Residueのアプローチでは，設計問題を与えられた設計のゴールに対するResidueを計算することによって解く．ここでResidueとは次のようなものである．

Wを初期世界のモデル，Gを設計のゴールとするとき，Rが設計の解すなわちResidueであるとは，次の三つの条件を満足する場合である．

R1. ゴール達成可能	W ∪ R ⊢ G
R2. 無矛盾性	W ∪ R はconsistent
R3. 実現可能性	Rに含まれる事実は真にすることができる

上のR3. の条件はRが設計者により実現できるものであることを表しており，設計問題を意味づけている．Residueを求めるアルゴリズムは証明を行うプロセスから得られる．

図に示す例を用いて説明を行う．

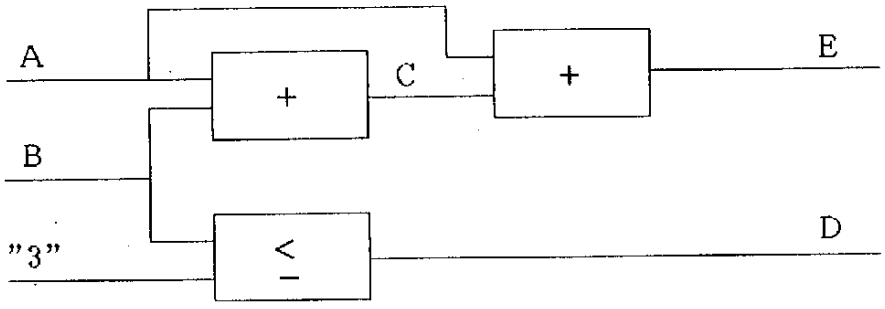


図4.2-2　Residueの例

図の回路において，E＝１４，D＝１となるようなA，Bを求めることを設計目標とする．E＝１４というのEで示す信号ライン上の点で値が１４であるという意味である．初期データベースWは次のようなルールを含んでいる．

C4:　A=x1 ∧ B=x2 ∧ x1+x2=x3 ⇒ C=x3

C5:　A=x1 ∧ C=x2 ∧ x1+x2=x3 ⇒ E=x3

C6:　B=x1 ∧ x1≦3 ⇒ D=1

C7:　2x + y = z ∧ even(z) ⇒ even(y)

(E=14 ∧ D=1) を証明する過程で用いられた仮定をもとめればそれがRである．この目標の否定から矛盾を導き出す．目標の否定を左側に，それまでに生成した過程を右側に組に

して記録する.

E1: < E≠14 ∨ D≠1 | {} >

E1とC5とのClause-Clause Resolutionから,

E2: < A≠a ∨ C≠b ∨ a + b≠14 ∨ D≠1 | {} >

C4とE2とのClause-Clause Resolutionから,

E3: < A≠d∨B≠e∨d + e≠b∨A≠a∨a + b≠14∨D≠1 | {} >

ここで A=d の仮定をおくと,

E4: < B≠e∨d + e≠b∨A≠a∨a + b≠14∨D≠1 | { A=d } >

E4に対するAssumption-Clause Resolution から,

E5: < B≠e∨d + e≠b∨d + b≠14∨D≠1 | { A=d } >

ここで B=e を仮定すると,

E6: < d + e≠b ∨ d + b≠14 ∨ D≠1 | { A=d, B=e } >

はじめの２つの節のClause-Clause Resolution から,

E7: < 2d + e≠14 ∨ D≠1 | { A=d, B=e } >

ここでC7からのContraint Propagationによって, １４は偶数なので,

E8: < 2d + e≠14 ∨ D≠1 | { A=d, B=e, even(e) } >

E8とC6とのClause-Clause Resolution から,

E9: < 2d + e≠14 ∨ B≠f ∨ f≦3 | { A=d, B=e, even(e) } >

ここでAssumption-Clause Resolution から,

E10: < 2d + e≠14 ∨ e≦3 | { A=d, B=e, even(e) } >

ここで e≦3 を仮定して,

E11: < 2d + e≠14 | { A=d, B=e, even(e), e≦3 } >

ここで Ｂ＝２を選択（仮定）すると,

E12: < 2d + 2≠14 | { A=d, B=2, even(2), 2≦3 } >

これを簡単化し, ｄ＝６とおくと矛盾が導かれる,

E13: < NIL | { A=6, B=2, even(2), 2≦3 } >

これから求めるRediduが, { A=6, B=2, even(2), 2≦3 } であることが分かる. このようなアルゴリズムはgoal-directed enumeration とよばれておりblind enumeration とよぶ総当たり的探索より速いアルゴリズムである. 設計問題の特徴は, 設計者が仮定可能な命題集合からなる探索空間が膨大になることであるが, そのためにこのアルゴリズムが

考えられた.

多くの場合，上記の条件を満たす複数個の仮説が生成される．このとき妥当な仮説を選定するための質問を生成し，質問の回答を通して追加の要求仕様を得て，仮説を１個に絞り込むことも仮説推論の動作である．上では制約条件を満たす設計パラメータを求める問題として定式化したが，石塚［石塚　88］は次の様な設計問題をあげて仮説選定の説明を行っている．

下図のような計算回路において，内部の計算モジュールｘ１，ｘ２の機能はplus（加算），minus（減算），mult（乗算），div（除算）のいずれかであるが，定まっていないとする．

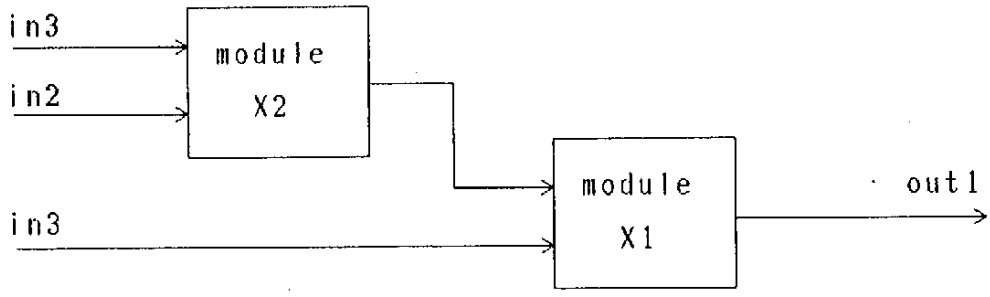


図4.2-3　設計対象回路

ここで入力ノードin1,in2,in3に１，２，２の値を入れると，出力ノードout1の値は４であるという仕様が与えられたとする．

in1＝１，in2＝２，in3＝２　　⇨　　out1＝４

すると石塚らのシステムにより可能な４個の仮説が次のように形成される．

＜function (x2, plus)　&　function (x1, mult) ＞
＜function (x2, mult)　&　function (x1, mult) ＞
＜function (x2, plus)　&　function (x1, div) ＞
＜function (x2, mult)　&　function (x1, div) ＞

仮説選定の質問応答プロセスが，をもちいて，追加の仕様を要求することにより，最終的にモジュール機能を決定する．システムは，

in1＝1，in2＝1，in3＝1　⇨　out1＝2

が正しいかどうかを聞いてくる．ｎｏと答えると，次に，

in1＝2，in2＝1，in3＝1　⇨　out1＝2

が正しいかどうかを聞いてくる．これもｎｏと答えると，最終的にモジュールx1の機能がdivであり，モジュールx2の機能がmultであると決定する．

2）現実場面への応用

以上に述べてきたような設計問題をフォーマルに捉える見方は現実の複雑な設計問題に対するアプローチの示唆を与える．しかし，現実の設計場面とのギャップを考えると次のような解決すべき課題がある．

1. モデルを完全には記述できない
2. 隠れている仮定，暗黙の前提がある
3. ゴール達成可能性が完全には不明なことが多い
4. 制約条件，要求仕様を途中で変える
5. 探索範囲が広すぎる

仮説推論によるフォーマルなアプローチを意識することはモデル化を図る上でおおいに役立つが，設計問題としてフォーマルに記述するためにはかなりの単純化を行わねばならず，対象問題によっては問題の本質を落としてしまうことにもなりかねない危険がある．現実的に有効であるような対象モデルを作ることが最重要課題である．

設計実行段階におけるゴール達成可能性の検証には，シミュレーションなどのツールを用意されていることが考えられるが，これはかなり設計プロセスの解明が行われている分野に限られる．また，実際の設計では要求される仕様が明確になっているとは限らず，設計の満たすべき条件そのものを試行錯誤的に変化させていくことはしばしば行われる．これを取り扱うためには仮説推論の枠組みがうまく適合するように思われる．

フォーマルな定式化からは探索範囲を極端に制限するようなヒューリスティックは引き出すことはいまのところ期待できない．階層設計はこのギャップを埋める重要な手段である．階層設計は，仮定を置く，つまり，設計パラメータを部分的に決めることによって問題のサイズを小さくする，仮定をおいていく順序（設計手順）を決めているなどの意味をもっている．すなわち，問題の設定時に，すでに，探索のための枝刈を行っておくという

ことである.

機械設計において比較的早く実用になると考えられるものに，パラメトリック設計がある．パラメトリック設計は，設計対象の諸元や寸法をパラメータで表しておいて，設計仕様が与えられたのちパラメータの値を定める設計方法であり，シリーズ化した製品の設計に有効な設計方法である．同様な方法は電気回路設計においても用いられているが，構造そのものを作り出すことは，非常にむずかしいため，このようなパラメータ探索問題を試みるのが現実的であるように思われる．

［参考文献］

［松田　88］松田　哲史，石塚　満：仮説推論システムの拡張知識表現と概念学習機構，人工知能学会誌，Vol.3, No.1, pp.94-102 (1988).

[Finger　85] J.J.Finger and M.R.Genesereth : RESIDUE - A Deductive Approach to Design Synthesis, Stanford Heuristic Programming Project, Memo HPP-85-1(1985).

［石塚　88］石塚　満，松田　哲史：知識の獲得と管理－仮説知識を含む知識ベースでの実例，知識プログラミング，共立出版，pp.125-156 (1988).

(4.2節担当：　森)

4.3 事例ベース的接近

4.3.1 事例利用モデルの同定

3.3で述べたように設計・計画型問題の中には，事例の再利用的側面がある．たとえば機能性新規化合物の設計では，化合物の構造から理論的にその機能を予測することはほとんどの場合できない．したがってそのような化合物の設計では，通常以下の手順で行われる．

①化合物が持つべき機能をキーとして，過去に開発された化合物データベースを探索し，その機能を特性としてもつ化合物を検索する．
②新しく要求される機能，スペックをすべて満足するように，検索された化合物の構造の一部を変更，修正する．
③性能評価を行い，不十分であれば②または①に戻る．

このような設計方法は，機能性化合物の設計だけではなく，機械設計問題でのいわゆるルーチン設計などでよく見られるアプローチである．

これらのように過去の計画・設計事例に基づいて，変更や修正を行い新たな問題を解決する過程は事例利用モデルとして同定できる．

4.3.2 事例に基づく推論としての定式化

事例に基づく推論は，その必要性や枠組みとして必要なプロセスの提案はなされてはいるものの，まだ技術として確立されたものはあまりない．したがってこれまでに報告された研究事例も対象とする領域に大きく依存しており，技術として一般化されてはいない．

本節では事例に基づく推論で最も重要なプロセスである事例の特徴付け，検索，変更について，設計・計画分野への応用事例で用いられている手法について説明する．

1)設計への応用[Navinchandra 88]

①システム名

ＣＹＣＬＯＰＳ

②対象領域

景観設計（Landscape design）分野でのレイアウト問題を取り扱っている．

③事例に基づく推論手法

以下では「傾斜面に家を建てる」問題を例にとり説明する．これまでに獲得された事例として以下の二つをもっているとする．

ブロックの事例：「おもちゃのブロックを斜面においたら，そのブロックが傾斜した．」

タイの事例：「タイでは洪水による浸水を防止するために地面に支柱を打ち込みその上に家を建てている．」

・事例の定義／特徴付け

CYCLPOSが持つ事例には２種類ある．ひとつは状況だけで記述されている事例であり，ブロックの事例がそれにあたる．他方はタイの事例にように問題状況とその解から成り立っている事例である．

状況だけからなる事例は，意味ネットを用いた因果関係構造で表現されている．ブロックの例では，「ブロックが地面の上にあり，かつ地面が斜めであるので，ブロックは傾いた」という因果関係構造が表されている．（図4.3-1）

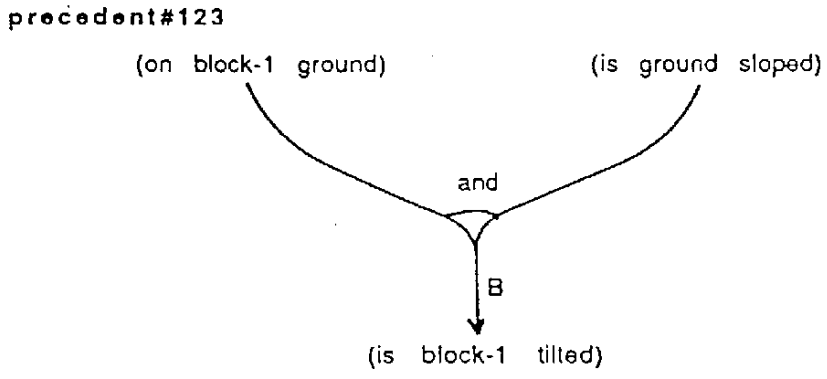


図4.3-1 ブロックの事例の表現

問題の状況とその解を含む事例も同様に，それぞれが因果関係構造で表現されており，さらに状況と解との間の因果関係構造も記述されている．タイの事例を図4.3-2aに，その因果関係を図4.3-2bに示した．CYCLOPSではこのような因果関係構造が特徴として用いられている．

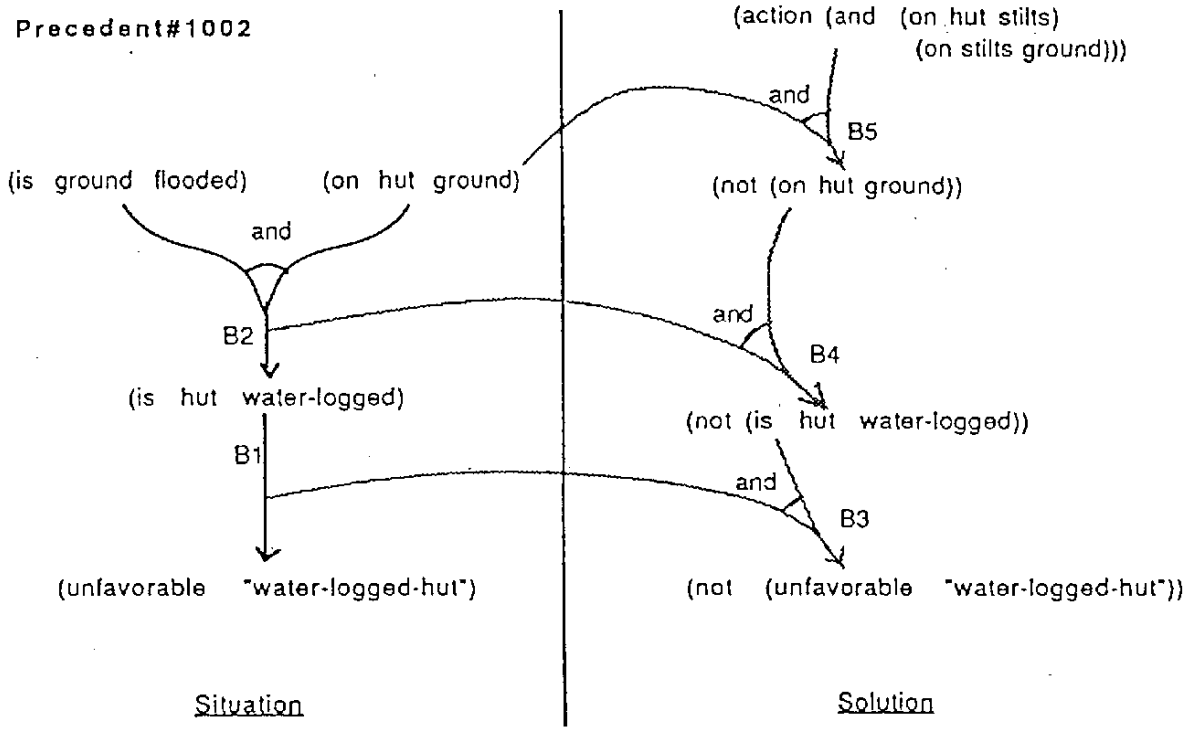


図4.3-2a　タイの事例の表現

B1: (unfavorable "water-logged-hut") because (is hut water-logged)

B2: (is hut water-logged) because (and (is ground flooded) (on hut ground))

B3: (not (unfavorable "water-logged-hut")) because (and (B1 is true) (not (is hut water-logged))

B4: (not (is hut water-logged)) because (and (B2 is true) (not (on hut ground)))

B5: (not (on hut ground)) because (and (on hut ground)
(action (and (on hut stilts) (on stilts ground))))

図4.3-2b　タイの事例の因果関係

・検索

いまここで「傾斜面に家を建てる」という問題が与えられると，(on house ground) と (is ground sloped)がシステムへの入力として与えられる．この特徴がブロックの事例（図4.3-1）と照合し，さらに「傾いた家は不都合である」という推論規則から図4.3-3に示したような与えられた問題に対する因果関係構造が作成される．

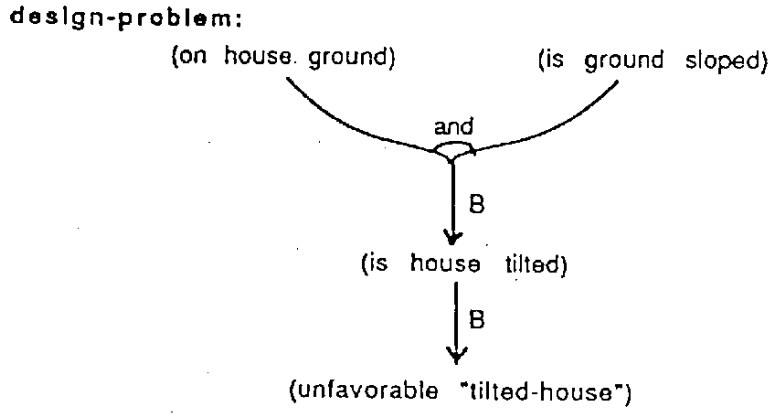


図4.3-3　問題の因果関係構造

家（house）とブロック（block）との照合は図＊に示した対象の概念階層を利用して行われる．

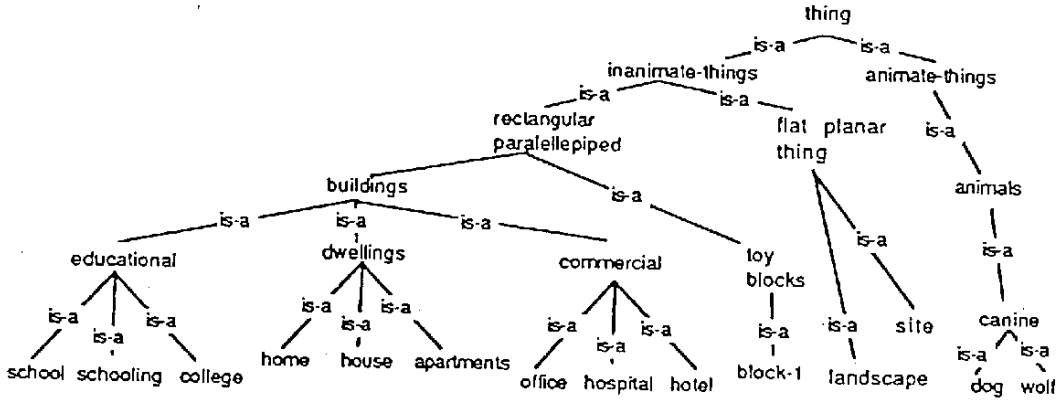


図4.3-4　概念階層構造

CYCLOPSは類似事例の検索にあたり，デマンドポスティング（Demand Posting）手法と呼

ぶ，上述した事例の因果関係構造を階層的にたどり，より根本的な原因に照合する事例を検索するという特徴をもっている．

デマンドポスティング手法を用いた事例の検索は以下のように行われる．（図4.3-5 参照）

①問題状況の因果関係構造のなかで直接の問題点である(unfavarable "tilted-house")) を解決できる事例を検索するための質問をポストする．（図4.3-5　A）

②事例が見つからないため，問題状況の因果関係リンクとたどり原因を問題点の原因を解決するための質問をポストする．（図4.3-5　B）

③やはり事例が見つからないのでさらにその原因を解決するための質問をポストする．（図4.3-5　C）

④質問Q3a:(not (on house ground))が図4.3-2に示したタイの事例と照合する．

このような方法を用いてCYCLOPSでは類似事例の検索を行うことを可能にしている．いまの例では「浸水を解決するためのタイの事例」が「傾斜面に家を建てる問題」と照合している．

・事例の変更

照合した事例の解は，与えられた問題の解決案としてコピーされる（図4.3-6）．こうして得られた問題の解決案は，新たな問題点が生じないかテストされる．問題点があったならば，再度デマンドポスティングによりその問題を解決するための事例が検索され，解決策が解決案に付加される．なにも問題がなければそれが解となり，事例ベースに記憶される．

CYCLOPSで用いられている因果関係構造を利用したデマンドポスティング手法は，事例が与えられた問題に直接適用できない時の有用な解決方法のひとつであるが，問題点としては事例に記述されている因果関係構造は予め決定されていなければならないことがあげられる．より柔軟な推論のためには，必要に応じて適切な因果関係構造を生成するメカニズムが必要とされる．

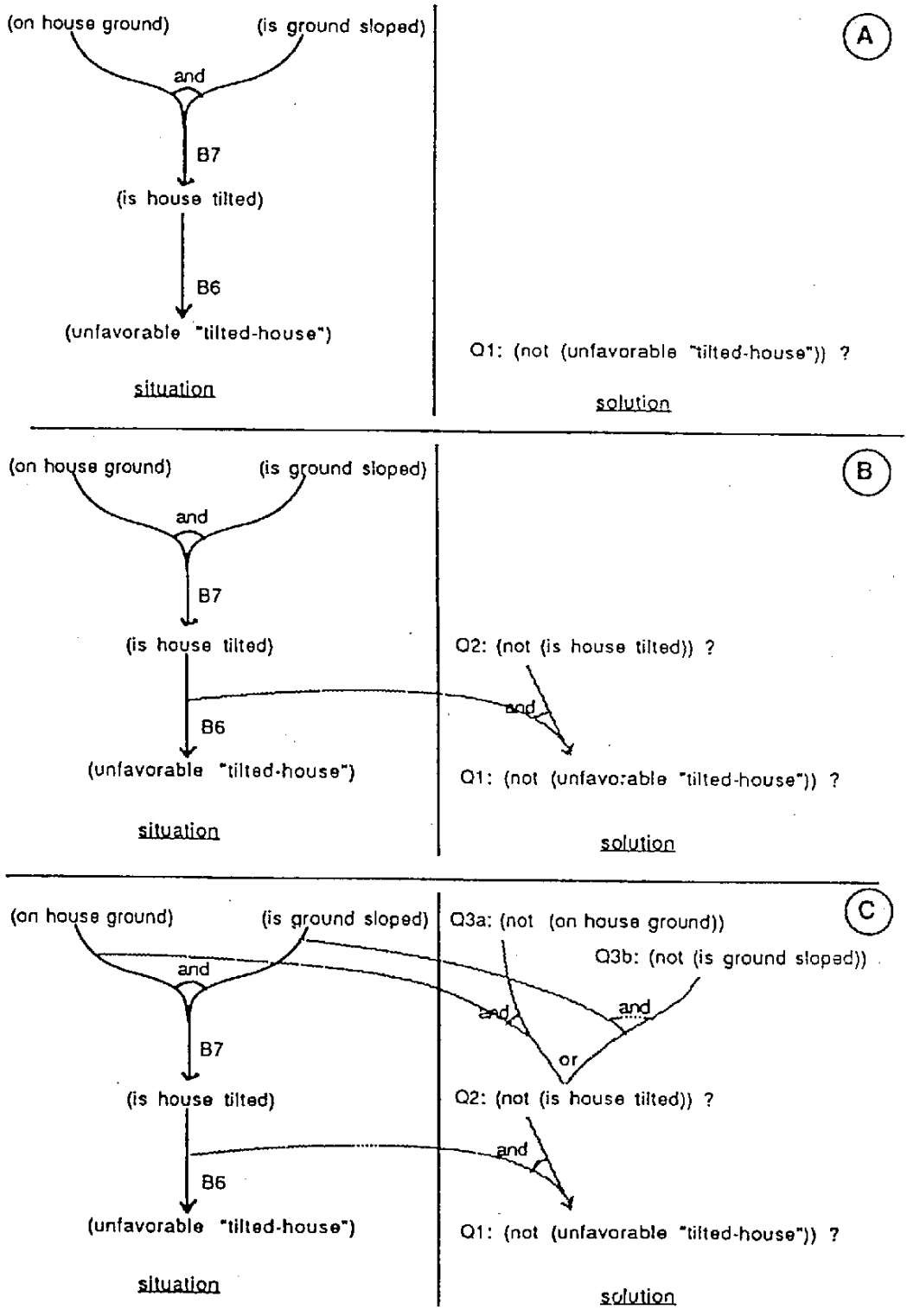


図4.3-5　デマンドポスティング手法

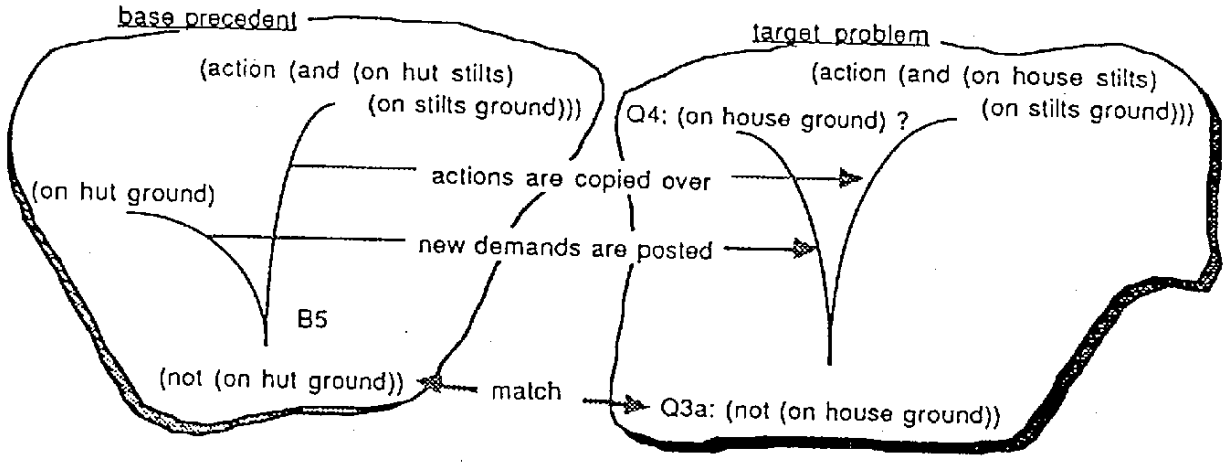

base precedent
(action (and (on hut stilts)
(on stilts ground)))
(on hut ground)
B5
(not (on hut ground))
target problem
(action (and (on house stilts)
(on stilts ground)))
Q4: (on house ground) ?
actions are copied over
new demands are posted
match
Q3a: (not (on house ground))

図4.3-6　事例の変更

2)プランニングへの応用

①システム名

CHEF

②対象領域

中華料理の調理法作成

③事例に基づく推論手法

・特徴付け

入力: 料理の味（hot,spicy,...），材料（beef,chicken,broccoli,...）
料理法（fry,souffle,pasta,...）

CHEFでは上記の属性が入力として与えられると，これらを充足すべき目標として設定するだけでなく，さらにプランニングの過程で発生するであろう問題点を予想し，それらを回避すべき問題点として目標に付加する．このように問題点を予め予想しておくプロセスを問題予想（Problem Anticipation）と呼んでいる．

例えば入力が「牛肉とブロッコリの炒めもの」であるとする．システムは先ずそれらの入力を目標として設定する．さらにシステムに過去の事例で「牛肉とブロッコリを一緒に炒めたら牛肉からでた肉汁のせいでブロッコリの歯ざわりを悪くした」という失敗例があったとするとそれを想起し，今回のプランニングで回避すべき問題点であるとして目標に付加する．

事例の特徴付けは，充足すべき目標と回避すべき問題点の２種類の特徴で構成されている．このように失敗例に着目し，同じ失敗を繰り返さないように設計者に注意を喚起することは設計計画問題において重要である．

・事例の検索

最良照合（Best Match）と呼んでいる手法で適切な事例を検索している．最良照合は，「プランニングにおいて最も重要な目標のなるべく多くを充足する」と定義されている．

このためにCHEFでは次の知識を利用している.

・類似性尺度: 目標（対象）間の類似度を与えている.

（例 "牛肉"と"鶏肉"はそのままでは照合しないが，類似性尺度からそれらは両方とも"肉"であることがわかり，照合が成立する.）

・価値階層: プランニングすべき目標集合の相対的重要性を与えている.

このような検索で得られた事例を基にするという前提には，過去の問題の状況が現在与えられた問題の状況と似ていれば，過去の問題の解は現在の問題を解決するために役立つという考えがある.

・事例の変更

このプロセスは対象領域依存であり，CHEFにおいても領域知識を用いて変更をおこなっている. 実例として「牛肉とブロッコリの炒めもの」レシピを基になる事例として利用し，「鶏肉とsnow pea（ブロッコリと同じ様な歯ざわりを持つ野菜）の炒めもの」レシピに変更する過程を図4.3-7に示した. この例の場合，事例の変更は対応する料理の材料を置き換えればすむので比較的容易である.

CHEFでは事例の変更後，検査をおこない変更にともなって副次的に発生する手続きをレシピに付加している. 例では牛肉を鶏肉に変更したのにともない，鶏肉を切る前に骨を抜くという手続きが加わっている.

```
Modifying new plan to satisfy:
        Include chicken in the dish.
Substituting chicken for beef in new plan.

Modifying new plan to satisfy:
        Include snow pea in the dish.
Substituting snow pea for broccoli in new plan.

Considering critic:
Befor doing step:Chop the chicken
 do: Bone the chicken. -- Critic applied.
```

図4.3-7 CHEFでの事例の変更実施例

3）プログラミングへの応用

①システム名　ＴＡ

②問題領域　　プログラミング方法の学習システム

③事例に基づいた推論手法

・動作概要

ＴＡの動作概要は以下の通りである。

ステップ１　　：プログラム事例ベースを検索し、プログラムを得る。

ステップ２　　：プログラム仕様から予測した出力と、プログラム事例ベースから得たプログラムとの出力を比較する。

ステップ３'　　：比較結果が同じであれば、プログラムをユーザに提示する。

ステップ３''　：比較結果が異なれば、バッチ事例、デバッグ事例を用いて、プログラム修正を行う。新規プログラムをプログラム事例ベースに登録する。

ステップ３'''：比較結果が異なり、且つ、利用可能なバッチ事例、デバッグ事例がない場合は、ユーザと対話を交えつつデバッグを行う。この際システムは、バッチ事例、デバッグ事例を学習し、該当事例ベースに登録する。また、新規プログラムをプログラム事例ベースに登録する。

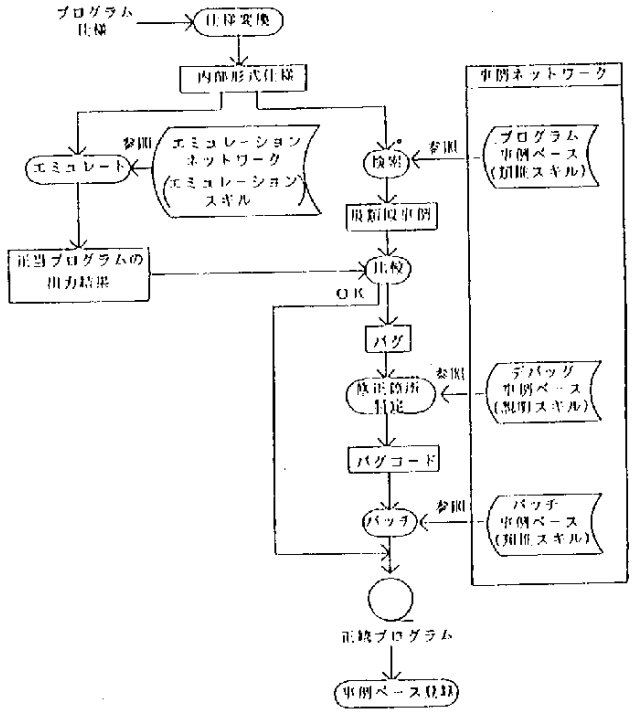


図４．３－８　ＴＡの動作概要

・事例の特徴付け

全ての事例に、検索インデックスが付随している。プログラム事例は、プログラム仕様をインデックスとする。パッチ事例は、新旧プログラム仕様の差をイデックスとする。デバッグ事例は、仕様を一般化しインデックスとする。

例えば、システムが、事例ベースのなかに、(∀ ?X ?L (ATOMIC ?X))という仕様を持つプログラムを備えていたとしよう。プログラムを書くために、次のような仕様を、さらに仮定しよう。(∃ ?X ?L (ATOMIC ?X))

新しいプログラムを書こうとすると、最初のプログラムを新しい問題の解として利用することによってバグが発見できる。バグをとるためにパッチを当てる。システムは新しい、バグのないプログラムにするためにパッチする。

旧：(∀ ?Z)

新：(∃ ?Z)

バグは一般化した仕様(?P ?X ?L (ATOMIC ?X))をインデックスとする。

・事例の検索

プログラムの仕様やバグの状況など、プログラムの生成工程で生じた問題を事例にメッセージとして伝送する。各事例は、問題の状況に適合し得る割合（プログラムの仕様が問題として与えられた場合は、仕様と適合する割合）を調停ノードに伝送する。検索時のインデックスは図４．３－９である。調停ノードは、適合した割合の高いメッセージを選択し親調停ノードに伝送する。以上のアルゴリズムにより、ルートに達したメッセージは、現在の問題に最適な事例を表現していることとなる（図４．３－１０）。

プログラム仕様：

```
(∀ ?X ?L (ATOMIC ?X))
```

↓ 変換

内部形式の仕様：

```
(SEARCH
    (SEARCH-SPACE-TYPE CONS)
    (SEARCH-SPACE ?L)
    (SEARCH-TYPE LIST-SEARCH)
    (SEARCH-ELEMENT ?X)
    (STOP-AFTER ALL-SEARCHED)
    (ON-TERMINATION SUCCEED)
    (TEST
        (TESTER (TEST (ISA ?X ATOM))
                (ON-SUCCES SUCCEED)
                (ON-FAIL FAIL)))
    (ON-SUCCESS CONTINUE)
    (ON-FAIL FAIL))
```

変換 →

検索インデックス：

```
(CONS SEARCH-SPACE-TYPE SEARCH)
(?L SEARCH-SPACE SEARCH)
(LIST-SEARCH SEARCH-TYPE SEARCH)
(?X SEARCH-ELEMENT SEARCH)
(ALL-SEARCHED STOP-AFTER SEARCH)
(FAIL ON-TERMINATION SEARCH)
(?X ISA TEST TESTER TEST SEARCH)
(ATOM ISA TEST TESTER TEST SEARCH)
(SUCCEED ON-SUCCESS TESTER TEST SEARCH)
(FAIL ON-FAIL TESTER TEST SEARCH)
(SUCCEED ON-SUCCESS SEARCH)
(CONTINUE ON FAIL SEARCH)
```

図４．３－９　プログラム事例で用いる検索インデックス

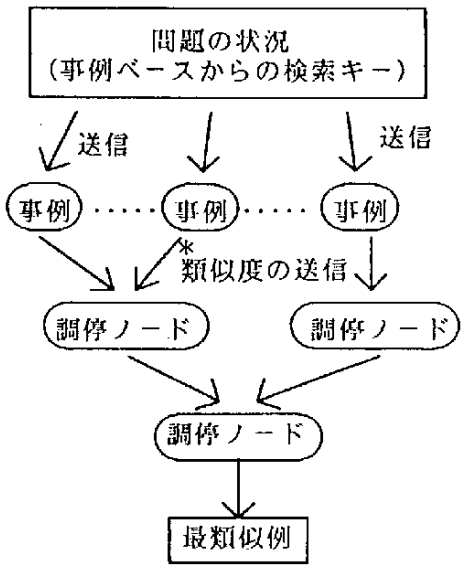


図４．３－１０　ＴＡにおける事例ネットワークの構造

＊類似度：キーと事例のインデックス
との適合割合

・事例の獲得

新規プログラム及び、ユーザと対話を交えつつデバッグを行なった際に獲得したパッチ事例、デバッグ事例を、該当事例ベースに登録する。

・事例の記憶

事例ネットワークには、既に開発したプログラム、パッチ方法、バグを蓄積する。事例は、逆２進木の形状で蓄積され、ルート及び各ノードには、調停ノードが存在し、葉の部分に事例を配置する。

事例ネットワークとして蓄積される事例ベースには、以下の３種類がある。

１）プログラム事例ベース

既存のプログラムを体系化した事例ベース。

２）パッチ事例ベース

バグを含んだコードを正常に動くコードに変換するパッチ方法を体系した事例ベース。新旧のコードを記憶している。

３）デバッグ事例ベース

プログラムを生成した際に生じたバグを体系化した事例ベース。夫々のバグ事例は、バグを含んだプログラムのバグに関する情報を含んでいる。

（4.3.2－3　担当　千吉良）

【参考文献】

[Hammond 86]

Hammond, K. J.

CHEF:A Model of Case-Based Planning.

In Proceedings of AAAI-86, pages 267-271. 1986.

[Navinchandra 88]

Navinchandra, D.

Case Based Reasoning in CYCLOPS, a Desing Problem Solver

In Proceedings of DARPA Workshop on Case-Based Reasoinig, 1988.

[Williams 88]

Williams, R., S.

Learning to Program by Examinig and Modifying Cases

In Proceedings of DARPA Workshop on Case-Based Reasoning, 1988.

（4.3節担当　千吉良，関根）

4.4 学習的接近

4.4.1 知識獲得モデルの同定

2.3節(4)項で議論したように，計画・設計問題には，知識獲得的側面がある．対象とする問題に対して，専門家自身が完全な知識をもっていない場合には，問題解決プロセスを通じて，問題解決に有効であることが判明した一連の操作を知識として一般化することにより，類似の新しい問題解決状況に対しても対処することが可能になる．

計画・設計問題において必要とされる知識には，設計知識と制御知識の2通りがあることを2.3節(4)項で指摘した．設計知識とは，仕様を詳細化すること，すなわち，要求仕様である初期の機能的記述を実行可能または実装可能な操作的記述に変換するために使われる知識であり，通常，変換規則とかオペレータと呼ばれるものである．これに対し，制御知識とは，初期記述に対して，複数の設計知識が適用可能であるかまたは1つの設計知識でも適用箇所が複数存在するときに，どの設計知識をどの箇所に適用すべきかを指示するような知識である．

設計知識は，一般に，問題領域について陽に知られている知識であるのに対して，制御知識はいわゆるノウハウに相当する知識であり，専門家自身が問題解決活動を通じて獲得する必要のある知識であり，また既に獲得しているとしても，それを陽に表現することが困難な知識でもある．

設計知識が不十分であると，特定の問題が与えられたとき，仕様を満たすような詳細化を見出すことができないことが起こり得る．これに対し，設計知識が十分であって，制御知識が不十分であると，仕様を満たす詳細化を見出すことは理論的には可能であるが，後戻りを頻繁に行うことになり，試行錯誤的な探索を余儀なくされる．制御知識が十分に蓄積されてくれば，後戻りすることなく，設計知識を適用することが可能となり，結果的に手続き的な問題解決を実現できることになる．このように，制御知識の獲得は問題解決の性能を向上させることに直接寄与する．

以上の議論から，計画・設計問題における知識獲得モデルの同定とは，「専門家の問題解決行動を観察して，あるいは，学習システム自らの問題解決行動の経験を通じて，特定の問題解決において，どのような知識がどういう状況のもとで使われたのかを分析して，これを他の類似の問題解決においても利用できるように一般化すること」と定義される．

問題解決を通じての知識獲得を行うために，3.4節で紹介した学習を基盤技術として利用することができる．ここでは，まず，帰納的学習の１つであるバージョン空間法に基づく知識獲得の例として，ＬＥＸと呼ばれる不定積分を解くシステムを紹介する．

ＬＥＸにおける知識獲得

ＬＥＸは，与えられた積分問題に対して，どのオペレータをいつ適用すべきかという制御知識を獲得することを目標とする．ＬＥＸの積分オペレータには，例えば，つぎのようなものがある．

(op1)　∫ r・f(x) dx　=>　r ∫ f(x) dx

(op2)　∫ u dv　=>　u v − ∫ v du

(op3)　1・f(x)　=>　f(x)

(op4)　∫ (f1(x) + f2(x)) dx　=>　∫ f1(x) dx + ∫ f2(x) dx

(op5)　∫ sin(x) dx　=>　− cos(x) + C

3.4節で述べたように，ＬＥＸの学習はつぎの手順に従って行われる．

1)与えられた問題に対し，適切なヒューリスティックスがないとき，
　利用可能なオペレータを順次適用して試行錯誤的に問題を解く．
2)与えられた問題に対し有用であったオペレータを正例とみなす．
3)あらかじめ設定された概念の汎化階層（バイアス）を利用して，
　オペレータに対するバージョン空間を生成する．
4)新しい問題を解くことにより，バージョン空間を更新する．
　（バージョン空間の更新は候補消去アルゴリズムに従う）．
5)バージョン空間が収束（ＭＧＶとＭＳＶが一致）すれば，
　これを新しいヒューリスティックスとして登録する．
　1)〜5)のステップを繰り返す．

例題として，

∫ 3 x cos(x) dx

が与えられたとする．直接，適用する知識がなかったとすれば，手順1)に従い，探索が行われ，その結果，

①(op2), ②(op1), ③(op5)

の順にオペレータを適用して，解が得られたとする．

つぎに，手順2)に従い，学習のための正例として，

$\int 3x \cos(x)\, dx$ => Apply (op2), where u = 3x and dv = cos(x)

がつくられる．手順3)に移り，関数に関する概念の汎化階層を利用して，オペレータに対するバージョン空間が生成される．この例の場合，バージョン空間の極大元および極小元はつぎのようになる．

MGV：

$\int f1(x)\, f2(x)\, dx$ => Apply (op2), where u = f1(x) and dv = f2(x)

MSV：

$\int 3x \cos(x)\, dx$ => Apply (op2), where u = 3x and dv = cos(x)

つぎに，新しい例題として，

$\int 3x \sin(x)\, dx$

が与えられ，つぎの正例および負例が生成されたとする．

(正) $\int 3x \sin(x)\, dx$ => Apply (op2), where u = 3x and dv = sin(x)

(負) $\int 3x \sin(x)\, dx$ => Apply (op2), where u = sin(x) and dv = 3x

同様の手順が繰り返され，その結果，バージョン空間はつぎのように更新される．

MGV：

(G1) $\int poly(x)\, f2(x)\, dx$ => Apply (op2), where u = poly(x)
and dv = f2(x)

(G2) $\int f1(x)\, transc(x)\, dx$ => Apply (op2), where u = f1(x)
and dv = transc(x)

MSV：

$\int 3x \, trig(x)\, dx$ => Apply (op2), where u = 3x
and v = trig(x)

なお，このバージョン空間の更新においては，関数に関する概念の汎化階層がバイアスと

して，積極的に使われ，汎化と特殊化が適切に制御されている．

ＬＥＸのような帰納的な学習システムでは，適切な知識を獲得するまでに，多数の正例および負例を必要とする．しかしこの条件は，実際には，かなり強い制約となり，適用の範囲を制限することになる．少数の例題しか利用可能ではない状況では，説明に基づく学習が有効である．ＬＥＸの後継であるＬＥＸ２では，説明に基づく学習を利用した知識獲得を行っている．

つぎに，ＬＥＸおよびＬＥＸ２では，問題解決システムが組み込まれているので，自らが問題を解いた結果を学習システムに引き渡し，学習が行われている．問題解決システムの部分を領域専門家に置換えれば，観察に基づく知識獲得システム，すなわち，見習い学習システム として利用することができる．そのような例として，ＬＥＡＰを紹介する．

ＬＥＡＰにおける知識獲得

ＬＥＡＰはＶＬＳＩの設計を支援する学習システムであり，説明に基づく学習をベースとしている．ＬＥＡＰはＶＬＳＩ設計支援システムＶＥＸＥＤを利用する領域専門家であるユーザの問題解決行動を肩越しに観察して，設計知識の適用を制御するための制御知識を獲得することを目的としている．

ＶＥＸＥＤは与えられた設計問題に対し，適用可能な設計知識をユーザに提示する．ユーザは提示された設計知識の１つを選択し，仕様を詳細化することを試みる．ＬＥＡＰはユーザの設計活動を観察して，詳細化に使われた知識とその適用状況を分析して，詳細化のモジュールとして認識し，その利用度を高めるために，一般化することを試みる．

一般化に際して，領域理論による証明，すなわち，説明に基づく学習が使われている．ユーザがつくった回路が要求仕様を満たしているどうかを，回路の等価変換規則を使って証明する．つぎに証明過程を一般化することにより，マクロルールの獲得が行われる．

このように，ＬＥＡＰでは，知識獲得の手法として，演繹的学習を採用しているため，少数の例題からの学習が可能である．

4.4.2 学習問題の定式化

前節では，観察からの学習による知識獲得の可能性を明らかにするために，不定積分問題領域における学習システムＬＥＸおよびＶＬＳＩ設計問題領域における学習システムＬＥＡＰを取り上げ，知識獲得の概要を考察した．

例題からの学習および観察からの学習に対して，帰納的な学習も演繹的な学習のいずれも可能であるが，どちらを選択するかは，利用可能な例題および領域知識の程度に依存する．そのことをつぎの表に示す．（表の中で，ＥＢＬは説明に基づく学習，ＳＢＬは類似性に基づく学習を表わすものとする）

利用可能な知識獲得の方式		領域理論	
		rich	poor
例題	rich	ＥＢＬ＆ＳＢＬ	ＳＢＬ
	poor	ＥＢＬ	類推

例題が rich であれば，ＳＢＬ的接近が可能であるが，適切な帰納的飛躍を起こさせるためには，強力なバイアスを必要とする．このバイアスは領域理論の１種とも考えられる．領域理論が rich であり，例題が poor であれば，ＥＢＬ的接近が適切である．領域理論が不完全な場合には，ＥＢＬ的接近を補完するために，ＳＢＬ的接近を導入する必要がある．領域理論も例題も poor であれば，ＳＢＬ的接近およびＥＢＬ的接近のいずれも不可能であり，類推による推測以外に適切な方法はない．

以下に，計画・設計問題領域における学習の応用例として，説明に基づく学習の応用を中心に現状を紹介する．

[Shavlik, J.W. and DeJong, G.F.(1987)]は，問題解決過程の知識を獲得するBAGGERと呼ばれるシステムを開発している．BAGGERは，問題解決過程において現われる繰り返しの過程を一般化することを目的としている．BAGGERはそのような一般化をＥＢＬの枠組みで行なう．説明中で繰り返し現われるルールをFocus Ruleと呼び，このルールを中心に繰り返しをなすルールの系列を見出す．一般化の結果は，任意長の繰り返しを畳み込んだルールとなっている．一般化ルールは，１ステップを証明することにより，そのステップを任意回適用した場合のルールを学習している．

[Minton, S. & Carbonell, J.G.(1987)]は，問題解決の性能を向上させるための戦略学

習システムPRODIGYを開発している．PRODIGYは，Explanation-Based Specializationを用いて，種々の現象(解法，失敗，目標の相互作用）から戦略を学習する．advanced STRIPS-like problem solverは領域独立あるいは領域依存な制御規則によって探索をガイドしている．ＥＢＬ-facilityは，制御の決定がなぜ適切であったのかを説明し，これを制御規則に変換する．

[Segre, A.M.(1987)]は，機械を組み立てることを学習する見習い学習システムARMSを開発している．ARMSは外部エージェントが同じような問題を解くことを観察することにより，組み立て作業のシーケンスをプランニグする能力をもつ．基本的オペレータは５つあり，値は連続的であるので，制御するためのヒューリスティックスを獲得する必要があり，ここにＥＢＬを利用している．

[Hammond, K.J.(1986)]は，料理のプランニングを行う事例からの学習システムCHEFFを開発している．CHEFFは，プランナー自身の失敗を解析することによって，プランニング問題を予測することを学習するための接近法を提案する．失敗を予測することの必要性を論じたうえで，プランニングの失敗を因果的に解析することにより，問題を予測する特徴を学習する．この接近法を事例に基づくプランニングの包括的理論に自然に統合できることを示している．

[Barletta, R. & Mark, W.(1988)] は，事例に基づく推論のための事例のインデキシングを行うために，説明に基づく学習を応用している．ここで，事例とは機械の故障復旧に際して取られる一連の行為をいい，この行為に関連性があるとみられる観察特徴を同定することが目標である．生成されるインデックスは，ある特殊な事例の中の初期状態の一部を一般化したものとして表現することができる．

[Wilkins, D.C.(1988)]は，エキスパートシステムの知識ベース洗練化のために見習い学習システムの手法を利用している．見習い学習は観察による学習の１形式であり，学習は人間の問題解決行動の説明を失敗した場合にこれを完全なものとすることによって行なわれる．見習いは医学のように知識強調型領域では人間が自分の専門知識を洗練化するうえで強力な方法であるとして，診断のような一般的な問題クラスを解くうえで戦略知識を陽に表現することが重要であると指摘している．

[DeJong, K.A. & Schultz, A.C.(1988)]は，オセロゲームを対象として性能を改善するこ

とを学習するシステムGINAを開発している．GINAに対する経験のソースとして多くのオセロゲームプログラムを収集して利用している．技能の洗練化と技能の改善の観点から経験に基づく学習の利点と欠点を明らかにすることを試みた．実験から経験に基づく学習はプレイのスピードと質の両方を改善するうえで有効なことが明かとなったとしている．

[Mooney, R.J.(1988)]は，従来の研究はオペレータの時間的順序を一般化する問題および半順序なアクションを伴うマクロオペレータの学習にはあまり注意を払ってこなかったとして，半順序構造をもつマクロオペレータを学習アルゴリズムを提案し，これをEGGSに組み込んだ．このシステムの性能を示すために，プログラミング問題への応用を試みている．一般化アルゴリズムの時間的複雑さの理論的解析も併せて示している．

[Mostow,J.(1988)]は，フロアプランニング問題を解きながら　失敗からの学習を行なう学習システムFailsafeを開発している．人間は，失敗した問題解決行動を解析することにより，類似の失敗を回避するように以後の問題解決をガイドすることを参考にして，この種の実行しながら(while doing)の学習は複雑な領域では本質的なこととしている．そのような能力を達成する手法として，失敗からのＥＢＬ（ＥＬＦ）を提案している．

以上，計画・設計問題領域における見習い学習システムの適用例として，説明に基づく学習を中心に応用例を紹介した．

参考文献

[Mitchell, T.M., Utgoff, P.E. and Banerji, R.(1983)]
Learning by Experimentation: Acquiring and Refining Problem Solving Heuristics,
Machine Learning: An Artificial Intelligence Approach, 163/190.

[Mitchell, T.M.,Mahadevan, S. and Steinberg, L.I.(1985)]
LEAP: A Learning Apprentice for VLSI Design, IJCAI-85, 573/580.

[Barletta, R. & Mark, W.(1988)]
Explanation-Based Indexing of Cases,
AAAI-88, pp.541-546.

[DeJong, K.A. & Schultz, A.C.(1988)]
Using Experience-Based Learning in Game Playing,

5-th Int. Conf. on Machine Learning, pp. 284-290.

[Hammond, K.J.(1986)]

Learning to Anticipate And Avoid Planning Problems through the Explanation of Failures, AAAI-86, pp. 556-560.

[Minton, S. & Carbonell, J.G.(1987)]

Strategies for Learning Search Control Rules: An Explanation-Based Approach, IJCAI-87, pp. 228-235.

[Mooney, R.J.(1988)]

Generalizing the Order of Operators and its Relation to Generalizing Structure, AAAI Spring Symposium Series: Explanation-Based Learning, pp. 84-88.

[Mostow, J.(1988)]

Failsafe: A Floor Planner that Uses EBG to Learn from its Failures, IJCAI-87, pp. 249-254.

[Segre, A.M.(1987)]

A Learning Apprenteice System for Mechanical Assembly, 3-rd Conf. on AI Application, pp. 112-118.

[Shavlik, J.W. and DeJong, G.F.(1987)]

BAGGER: An EBL System that Extends and Generalizes Explanations, AAAI-87, pp. 516-520.

[Shavlik, J.W. and DeJong, G.F.(1987)]

An Explanation-Based Approach to Generalize Number, IJCAI-87, pp. 26-238.

[Wilkins, C.(1988)]

Knowledge Base Refinement Using Apprenticeship Learning Techniques, AAAI-88, pp.646-651.

(4.4節担当 小林重信)

5．計画・設計型エキスパートシステムの事例分析

5. 計画・設計型知識システムの事例分析

5.1 配合エキスパートシステム

1) 序

設計型問題の知識システム開発事例はこれまでにある程度報告されているが，そのほとんどは機械設計やＬＳＩ設計などの分野におけるいわゆるルーチン設計問題への対応である．

一方，設計問題の所在はきわめて広範囲の分野にわたっており，上記の分野以外にも新素材や機能性材料の設計，生理活性物質や医薬品の分子設計などにおいても設計の自動化あるいは計算機による設計支援のニーズは大きいが，これらの分野での知識システムの開発事例はほとんどない．この理由はこれらの分野では設計をガイドするための理論や知識が確立されていないことが多いことにあると考えられる．またこれらの分野では設計部品（素材）の持つ各々の機能の組合せで要求仕様や特性を満足するだけではなく，部品の組合せによる複合効果を期待することが多い．その種の設計問題を化学的設計問題と呼ぶ．

本節では化学的設計問題のひとつである配合処方設計問題を取り上げ，その設計手順の分析を行い，事例に基づく推論としての定式化を試みた．

2) 配合問題

配合問題とは，要求特性のすべてを満足する素材が存在しなかったり利用できない時，各々の特性をもつ複数の素材を組み合わせて配合することにより要求特性を満足する処方を設計することである．この場合，機械設計における部品間の物理的な相互作用ではなく，複数の素材間の分子レベルの化学的相互作用が発生するためその挙動はきわめて複雑である．以下では配合問題の具体例として界面活性剤の配合問題を取り上げ，配合問題の特徴を説明する．

①界面活性剤（素材）の化学構造とその機能

まず界面活性剤の基本的な性質について概要を説明する．以下では界面活性剤という用語をひとつの配合素材としての界面活性剤と，そのような配合素材を組み合わせて配合し

た結果としての界面活性剤との両方の意味に用いる．

素材としての界面活性剤は親水基と疎水基からなる化合物であり，ほとんどの場合複数の界面活性剤を組み合わせて配合し，特定の用途（例　シャンプー，洗剤，・・・）に応用する．配合の主な目的は界面活性剤にスペックとして要求される機能をすべて充足するような単一の界面活性剤はほとんど存在しないため，それぞれ特徴ある機能を持つ界面活性剤を組合せ，要求される機能の役割分担をさせることにある．界面活性剤の構造とその作用の関係を図5.1-1に示した．化学構造は界面活性剤を構成する種々の因子（疎水基の種類，親水基の種類，対イオンの種類など）からなり，これらの構造因子の組合せによって界面活性剤に特有な性質である界面への吸着性，分子の配向性，ミセルの形成がきまり，この結果としてその界面活性剤のもつ乳化，分散，あわ立ちなどの諸機能が発現される．しかしこれらの性質や機能は界面活性剤の使用される環境すなわち溶媒などのよっても変化する．

専門家は経験を積むことにより，これらの構造因子の組合せをみてその機能の大小をかなりの程度まで推定できるようになる．しかし界面活性剤の機能は上記の構造因子の総合的な組合せだけではなく，同時に配合されている他の界面活性剤（素材）や溶媒などによって決定されるので，ある構造因子だけをとりあげてその因子と機能の関係をルールとして知識化することは困難である．

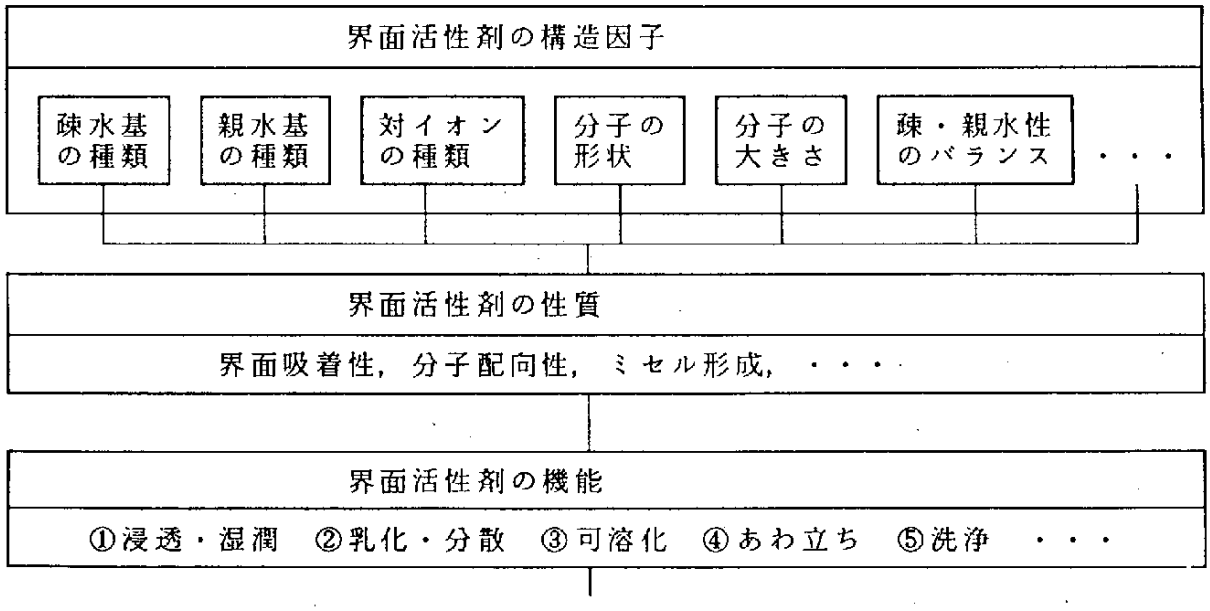


図5.1-1　界面活性剤の基本的構造因子とその性質／機能

② 複合効果

配合の主目的は単一の素材で要求特性を満足できない場合，複数の素材を用いて機能分担することであるが，他の重要な目的として複数の素材を配合することによる相乗効果の発現がある．ここでは配合問題における複合効果について分析する．

素材ａのもつある機能をＡで表すと配合問題の複合効果には次のようなものがある．

(i)相乗型（量的）：

機能Ａ（素材ａ1）＋機能Ａ（素材ａ2） ＜ 機能Ａ（素材ａ1＋素材ａ2）

(ii)相乗型（質的）：

機能Ａ（素材ａ）＋機能Ｂ（素材ｂ） → 新機能Ｘ（素材ａ＋素材ｂ）
＋機能Ａ（素材ａ＋素材ｂ）
＋機能Ｂ（素材ａ＋素材ｂ）

(iii)相補型：

機能Ａ（素材ａ）＋機能Ｂ（素材ｂ） ＝ 機能Ａ（素材ａ＋素材ｂ）
＋機能Ｂ（素材ａ＋素材ｂ）

(i)の相乗型は複数の素材ａ1，ａ2がもつそれぞれのある機能Ａとすると，それらの素材を配合した結果えられる機能の大きさが，素材を単独で用いたときの機能よりも大きい場合であり，それは量的な相乗効果といえる．

(ii)の相乗効果は複数の素材を配合した結果，別の新しい機能が発現する場合であり，これは質的な相乗効果である．

(iii)の相補型は配合した結果としてえられる機能は，単にそれぞれの素材の和にすぎないが，互いに欠けている機能を相補って要求特性を満足するような場合で，いわゆる機能分担である．

これらの複合効果のなかで相補型は比較的効果を予測しやすい．しかし配合することによって逆にそれぞれの機能が発現しなくなるという負の効果の場合も多くある．またその他の複合効果，なかでも質的な相乗効果を予測することは専門家にとっても極めて困難なことであり，専門家から体系的な知識として獲得することはほとんど不可能である．一方，

そのような複合効果を利用した過去の配合処方開発事例は，成功例及び失敗例を含め数多くあり，配合問題はそれらの過去事例を参考に解決されているのが現状である．

③配合問題の特徴

上述の配合問題の特徴をまとめると以下になる．

- 系に要求される複数の特性を同時に満足させるため，複数の素材を用いる．
- 複合効果の発現が要求されるが，その予測は困難である．
- 素材のもつ機能を完全に記述することができない．
- 配合する素材の種類を決定するとともに，それらの配合比率を決定する必要がある．配合比率を考慮すると組合せの状態空間は無限となる．
- さらに配合比率を変化させると系の性能が変化する．
- 同じ素材であってもおかれる環境でその性質，機能が変化する．
- 設計のためのおおよその規則はあるが，理論や知識として体系化されておらず，細部の設計まではできない．
- 配合事例は成功例および失敗例を含めて数多くある．

これらの特徴をもつ配合問題へのアプローチとしては，過去の配合事例に基づいて新たな問題を解決するという方法が行われている．

3)　事例に基づく推論としてのモデル化

本節では専門家による配合問題の問題解決過程のモデル化について検討する．専門家の問題解決過程を図5.1-2に示した．この問題解決過程は事例に基づく推論ですなおにモデル化することができる．以下では図5.1-2の各プロセスについて説明する．

①要求特性の入力

システムへの入力として，要求特性がスペックとして与えられる．

②特徴付け

要求特性に基づいて，過去の類似事例を検索するための特徴抽出を行う．具体的には，

(i)要求特性を問題解決のための重要度で順序づけること
(ii)過去の事例ベースを検索するためのキーワードに要求特性を変換すること
(iii)要求特性には陽に記述されてはいないが問題の解決には必要な概念を付加すること
などが行われる．このプロセスを事例に基づく推論では事例の特徴付けと呼ぶ．

③事例の検索

特徴付けを利用して事例ベースを探索し，問題解決に関係のある事例を検索してくる．結果として通常複数の配合事例が検索される．現在の問題の解決の基となる事例という意味でベース配合事例（処方）とよぶことにする．

④ベース配合処方の選択

事例の検索の結果えられた過去の配合事例群から，現在の問題に最も適した事例を選択する．ベース配合処方としては，現在の問題となるべく類似の事例を選択すれば，効率的に現在の問題を解決できると考えられる．しかし現実には”類似”の事例とは何かを定義することは困難であり，専門家に委ねられることが多い．

⑤ベース配合処方の変更

ベース配合処方を基に，要求特性をすべて満足するように変更する．新製品の開発などでは，新機能をもつ新素材の導入がこのプロセスで行われる．また既存製品の改良，例えばコストダウンを目的とするのであれば，ベース配合処方に含まれる最もコストの高い素材を，同じ機能をもつ安価な素材に置き換えるなどの操作が行われる．これらの操作をへて新処方案が生成される．

⑥評価

新処方案の評価を行う．ここでは先ずベース処方を変更したことにより，新たに生ずる問題点がないかチェックする．例えば素材の置換を行った場合，新たに系に導入された素材が既に配合されている他の素材に悪影響を及ぼすことがないかなどを，ルールあるいは過去の事例を用いて検査する．問題点があれば修復プロセスで修復する．問題点が予想されなければ実際に適用し評価する．適用に成功すればそれが新処方となる．適用に失敗した場合は⑦修復プロセスに行く．

⑦処方案の修復

評価の結果が失敗であれば，その処方案の修復を行う．このプロセスで専門家の行っている作業は極めて複雑であり，モデル化は困難である場合が多い．修復は新処方案に対して行われれる他，大きな修復が必要とされる場合はベース配合処方選択のプロセスや特徴付けのプロセスに戻って，再度ベース配合事例が選択される．

⑧事例ベースへの登録

要求特性をすべて満足した新処方は，新しい事例としてその特徴付けとともに事例ベースに登録する．

4) まとめ

計画・設計型問題の典型的な問題のひとつである配合問題について，問題の分析を行い，その特徴を指摘し，配合問題は事例に基づく推論としてすなおにモデル化できることを示した．しかしながら事例に基づく推論の各プロセスを具体化するには問題点が多い．特に事例の変更や修復については，対象領域に依存するところが多く，技術として一般化することは容易でない．現実としては，このような問題解決プロセスにどのように人間の専門家を取り込むかが重要な点になろう．

しかし配合問題のように，設計をガイドする知識が体系化されていない領域については，ルールに基づく推論によるアプローチは不可能であり，事例に基づく推論が唯一の問題解決手段であると考えられる．さらに化学は本来帰納的な学問であり，過去に極めて多くの事例が蓄積されており，データベース化も進んでいる．これらの過去事例をいかに知識化して有効利用するかが今後の課題であると考えられる．

［参考文献］

常盤文克，
界面活性剤とその応用，油化学，第２８巻第８号５７８頁（１９７９）

（5.1節担当　関根）

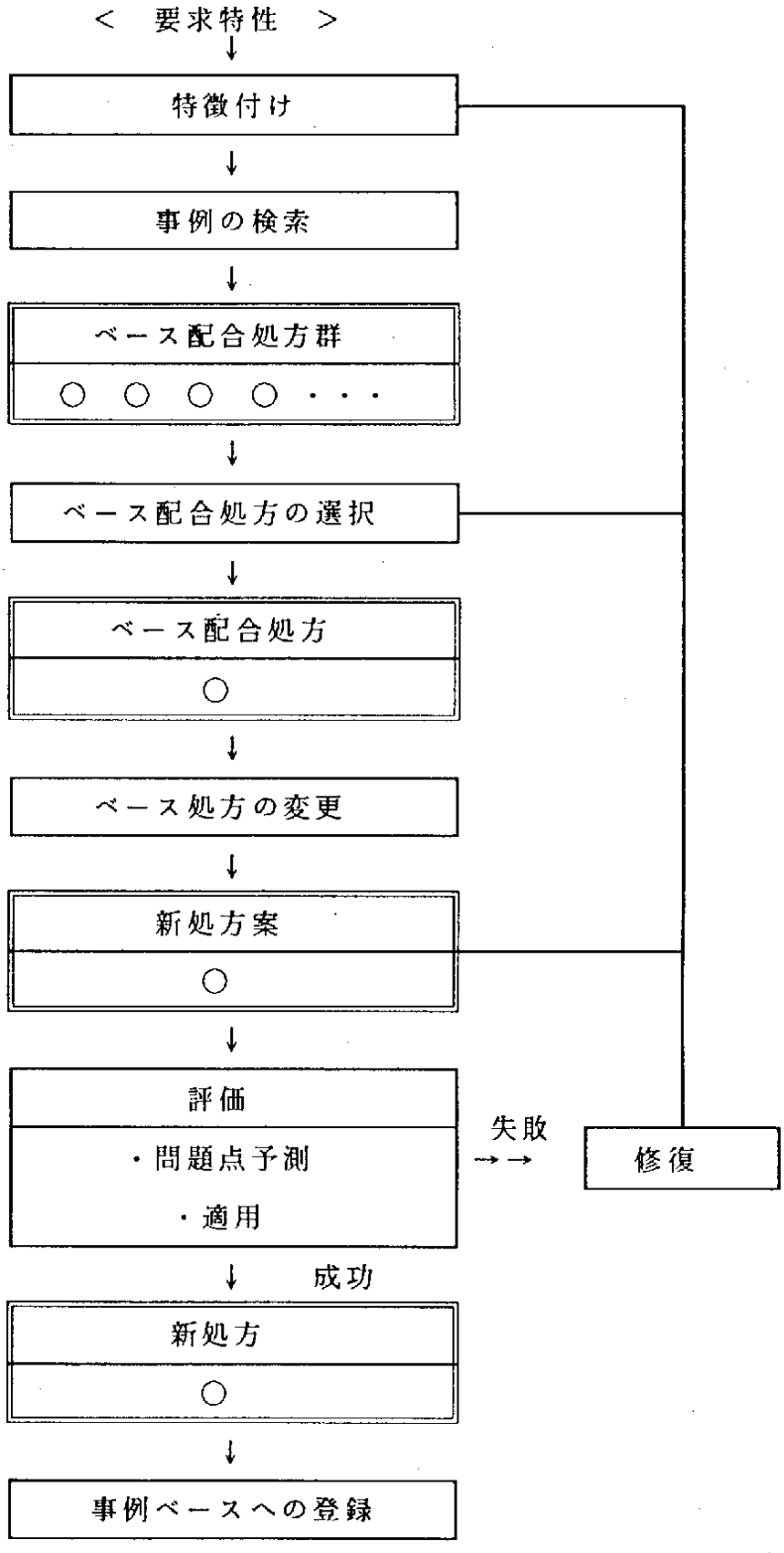


図5.1-2 配合問題解決モデル

5.2 レンズ設計エキスパートシステム

本節では，レンズ設計分野において，次世代のエキスパートシステムを構築するための技術的課題について述べる．まずレンズ設計とこの分野の自動化の難しさについて簡単に紹介し，次に現状のエキスパートシステムの役割と課題を述べる．その上で，現状のＡＩ技術のシーズ面から，近い将来実現が期待されるエキスパートシステムの機能について述べる．

5.2.1 レンズ設計問題の定義

まず，レンズ設計について一応の説明を行う．レンズ設計の目的は，与えられた要求仕様を満足するレンズシステムの形状を決定することである（図5.2-1）．要求仕様では，機能・形状寸法・コスト・性能に関しての制約が与えられる．一方，設計によって決定されるパラメータは，各レンズ面の曲率，ガラスの種類，厚みや，各レンズ間の距離といったものからなる５０～１００個程度の構成パラメータと呼ばれるレンズシステムの形状を規定するものである．

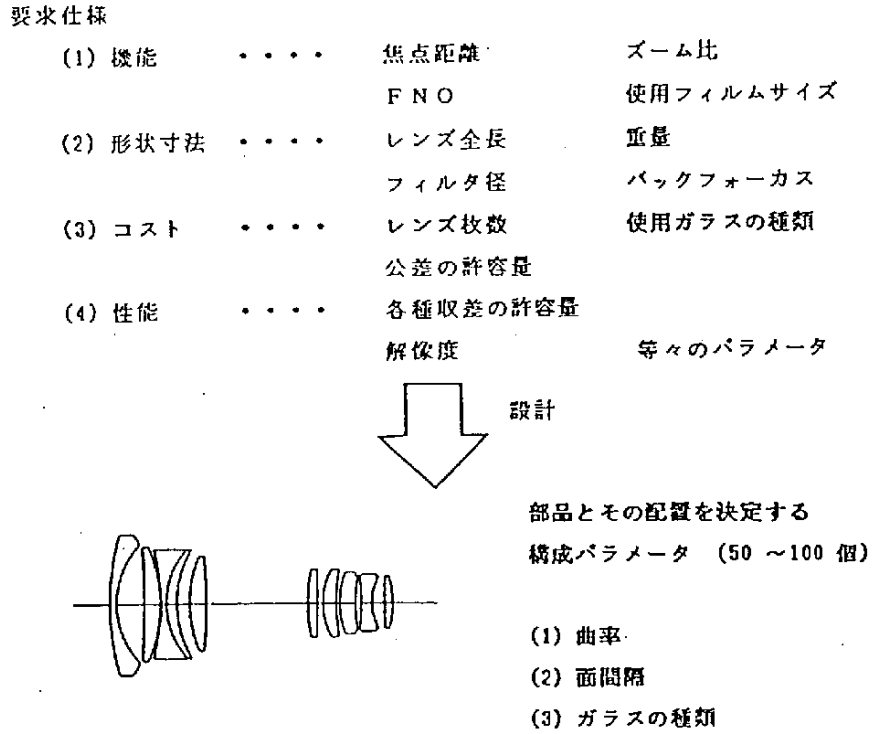


図5.2-1 レンズ設計の目的

このようなレンズの形状を決定する上での根本原理は，電気設計で言えばオームの法則に相当するスネルの法則に基づいている．これは，２つの屈折率の異なる媒体間の境界面へ光線が入射したとき，その光線の出射の様子が，光線の入射角と媒体の屈折率の比によって決まるというものである．この法則によって，被写体の一点から発生する無数の光線が，レンズシステムを通過した後，像面へどんな状態で結像するかを解析的に知ることが出来る．レンズ設計では，これを光線追跡シミュレーションと呼ぶ．このシミュレーションで得られた結像結果からは，レンズ収差を知ることが出来る．更に，レンズシステムの各構成パラメータとこの収差間の感度解析を行うことによって，レンズ形状を最適化することが可能となる．実際の設計では，設計途上のレンズシステムの性能を評価するため，計算機上の光学設計ＣＡＤによって，このシミュレーションや最適化が頻繁に行われている．

しかしながら，このような光学設計ＣＡＤがあるにもかかわらず設計が困難な理由は以下による．

①要求仕様で与えられるパラメータと設計で決定しなければならない構成パラメータ間には，強い非線形性が存在している．このため，非線形最適化の問題となっている．具体的には，先のＣＡＤによる最適化を実行しても，最適化するための初期値となったレンズシステム形状に依存し，最適化後の形状が異なってしまう．

②また，しばしば要求仕様の全パラメータを満たす妥当な解が存在しないことも有り得る．

③単に要求仕様上の制約を満足すれば良いわけでない．例えば，各種収差がバランス良く存在し，結像状態が視覚的により自然である必要がある．

等といった問題が存在する．このため，レンズ設計における設計者の役割は，

①要求仕様としての制約を満たすであろうと思われるレンズシステムの形状を，ＣＡＤへの初期値として設計経験に基づき設定する．

②設計途上のシミュレーションの繰り返し結果，最悪の場合はこの要求仕様を満足する設計を諦めるか否か・・・等といった意志決定を行う．したがって，設計途上もノウハウを学習過程であり，この知識を積極的に活用し設計上の意志決定を行っていると言える．

③性能間バランスのあり方については，結像状態とその認知の関係から一般的な知識が存在する．しかしながら，設計されるレンズシステム毎独特な特性があり，このような一般的に言われる理想的バランス通りにはならないかもしれない．したがって，設計途上レンズシステムの持っている特性を正確に見極め，最も望ましい性能バランスに収束させる必要がある．この過程は，設計者のセンスや創造性に依存するところが大きい．

レンズ設計には，特に②③のようにそのプロセスが十分に究明されていない面がある．しかしながら，これらは設計の品質や付加価値を決定している大きなファクターでもある．現状のエキスパートシステムに関わる技術においては，これらの実現を望むべくもないのが実状である．しかしながら，近い将来現状のエキスパートシステム技術開発の延長上で設計の一部代替可能となるところもあり，以下この面から議論をすすめる．

5.2.2 現状のエキスパートシステムの課題

前節で簡単に述べた様に，現状の光学設計エキスパートシステムは，ルーチン的設計の一部を代替しようするに過ぎない．したがって，未だ解決されなければならない課題が山積している．ここでは，こういったエキスパートシステムにおいて解決されなければならないいくつかの課題について述べる．

①並列的設計検討の問題

エキスパートシステムによる設計の初段階は，まず要求仕様中のズーム比・焦点距離・Ｆｎｏ等といった基本的な機能を満たすと考えられるレンズシステムの骨組み（図5.2-2）やそれぞれ骨組みを構成するレンズ形状を，データベースより検索してくることである．要求されている仕様は，つねに過去の設計事例の持っている仕様と異なる．したがって，検索においては，要求仕様中の各パラメータ値を許容範囲内でトレードし，要求仕様に近い過去の設計事例の複数持ってきている．以降，これらの検索された候補のレンズシステムは，候補の刈り込みが可能な時点まで，それぞれ独立かつ並行的に設計検討される．

現状では，候補のレンズシステムの刈り込みの時点やその基準は，設計者自身が判断している．また，この刈り込み作業は，それまでの候補の設計履歴（候補レンズシステムへ

の改良作業内容とその性能シミュレーション結果）の比較から，適切な設計時点で行なわれている．現状エキスパートシステム内で，このように並列的かつ協調的に設計検討を行うことは難しい．これは，各候補を刈り込むための知識が明かでないことも大きい原因であるが，並列的かつ協調的に推論制御を行う枠組みがＡＩシェルに無い（あっても性能的に不十分）ことが大きく影響している．

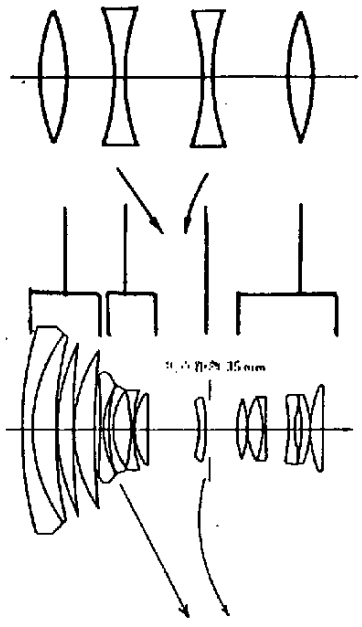


図5.2-2．レンズの骨組みとレンズ形状

②最適な設計戦略の発見の問題

骨組みの決定の様な単純な設計では，要求仕様から骨組みのパラメータを決定する最適な設計手順が明示的に存在するケースは多い．このためエキスパートシステム上，種々の設計ケースで予測される要求仕様のパラメータの与えられ方毎に，骨組みパラメータの最適な決定手順に関する設計知識を，骨組みの属性として表現していた．その結果，要求仕様のパラメータの与えられ方毎に類似した設計知識が豊富に必要とされたり，最悪これらの知識が属性として登録されていないケースの場合には，折角要求仕様から妥当な骨組み構造が検索されても，要求仕様へ適合する様に各パラメータ値が決定出来ないケースもあった．したがって，要求仕様に対してのエキスパートシステムの適用範囲の制限を緩和するため，より高次な推論に基づいたこれらの知識の導出の枠組みが必要とされている．

③過去の設計経験に基づく設計戦略の問題

エキスパートシステムは，過去に設計されたレンズシステムのレンズ形状を強制的に変

更し，その上ですでに決定された骨組みの中へ割り当てる．その後，この様にして出来たレンズシステムの収差等の性能評価は，ＣＡＤによるシミュレーションによって行なわれる．この評価結果に基づき，各レンズ形状の変更・レンズのガラス交換・レンズの付加削除・骨組みパラメータ値の変更等といったレンズシステムへの改善方法が決定される．この性能改善に関わる最適な設計手順は，先の骨組みの決定のケースの場合とは逆に，非明示的で定式化出来ない．したがって，過去の設計経験に基づいて決められる．具体的には，性能シミュレーションの結果を過去の同一タイプの骨組み構造をもったレンズシステムの設計経験と照らし合わせ，その設計時点で無視してよいと思われる性能項目の欠陥な後回しにして，現在最も重要と思われる性能項目上の欠陥を優先して取り除く方法で行なわれている．

現状のエキスパートシステムでは，このような設計手順の知識の適用において，期待された性能改善の効果が十分に出来ない場合や，あるいは性能シミュレーションの自体の結果が既に過去の事例と違った傾向のため適用すべき知識が存在しない場合等が発生し，設計困難に陥ることがある．したがって，エキスパートシステムの適用範囲をより広げるため，過去の設計経験によって得られた設計手順が必ずしも妥当性がない場合においても，設計を可能にする必要がある．

5.2.3　将来のエキスパートシステムの機能

現在の知識システム技術のシーズ面から考えると，これらの従来の問題点に関して，以下のような解決が期待される．

①並列的設計検討の問題

仮説推論の枠組みの中で，複数の設計対象による設計検討を並列的かつ協調的に実行可能とする．特に，仮説推論は比較的設計データが少ない骨組みの設計では，有効であろう．レンズ設計では，開発初期に比較的多数の骨組みの中から，収差補正に耐え得るものを探索することが多い．したがって，これら多数の骨組のモデルを個々の仮説とし，骨組みの設計やシミュレーションを並行的に行い，更にこれら骨組み間の比較のための知識に基づ

いた仮説刈り込みを行うことが出来る．このように見落としのない骨組みの設計検討が出来れば，設計品質の向上に大きく貢献する．

しかしながら，現状のＡＩシェルの仮説推論機構では，設計に要求されている実用規模の問題をこの枠組みの中で取扱にくい．これは，１つの仮説としての骨組みの正しさの検証は，性能データ等のそれに付随した多くの設計パラメータを決定した後でないと行えない．これによって個々の仮説のデータが膨大になる傾向があるためである．現状の仮説推論機構はどちらかといえば機能重視の理論先行型であって，実際のこの様なアプリケーションのためにはまだパーフォーマンスが不十分である．また，骨組み設計以降のレンズシステムの性能向上を目指した設計は，むしろバックトラックに基づく思考錯誤となる．したがって，推論機構がいま以上に自由に制御出来る枠組みも必要である．

②最適な設計戦略の発見の問題

要求仕様で与えられているパラメータから，その骨組みの各パラメータを導出する最適な手順が知られていない場合，設計者は，通常それを理論式などから導き出す．現状のエキスパートシステムは，既にこれらが導かれたケースだけを適用範囲としていたが，Arayaらによる知識コンパイルといったアプローチで，より高次な知識からこれらを導くことも考えられる．しかしながら，現状の課題としては，これらの知識を主としてパラメータ間の代数的関係の推論から導こうとしているが，実際問題与えられた要求仕様のパラメータと骨組みのパラメータ間の関係には数値計算によってしか求められないケースもあり，こういった問題に関し如何に対処するかといった点でまだ検討要素が存在すると思われる．

しかしながら，このような技術によって，エキスパートシステムの設計の適用領域が拡大されると同時に，表現しなければならない知識の量の大幅な減少が期待される．

③過去の事例に基づく設計戦術の知識の問題

性能向上のために従来使っている過去の設計経験で得られた設計戦術の知識は，表層的でありその適用範囲は狭い．したがって，性能シミュレーションで出てきた結果に対して，その性能劣化の原因をより理論に近い知識によって分析し，その結果得られたを新たな知識に基づき設計する必要がある．しかし，このような推論の枠組みは未だ明かになっておらず，また実際に設計者もすべてのケースでこの様にしているとも思えない．より現実的なアプローチとしては，そのレンズシステムを等価変換によってメタ的なモデルへ抽象化

することで，そのレンズシステムの骨組みのタイプ以外で，最も近いの骨組みを持った他の過去のレンズシステムの設計戦術の知識を参照することである．このような方法で，既存知識の適用範囲をより広くすることが可能であると思われるし，またより実際的である．したがって，今後事例推論的なアプローチが，こうしたシミュレーション結果に基づく性能改善のような設計へ有効な技術となる可能性があろう．

5.2.4 まとめ

以上の様な課題は，将来ＡＩ技術のシーズ面からの各種アプローチで解決が可能となると思われる．しかしながら，例えば，レンズシステムの種々の収差残存量のバランスといった様な点を考えても，本来如何にあるべきかと言う点が十分科学的に究明されていない．この点を考えても，設計とは単に”要求仕様を満足するものを作る”とは言えず，実際はこの過程での様々なトレードオフの結果”最適とは何か”を見つける過程である言える．ここに，本来の意味での設計の代替の難しさがあると思われる．また，レンズ設計の世界は要求仕様が年々厳しくなっており，設計当初の設計要求の全てを満足させることが困難な状況に直面することが発生する．したがって，その一部仕様を断念せざる追えないこともままある．したがって，この意味で意志決定を行なう設計者の役割がますます大きくなっている．

こうした状況下では，近未来のレンズ設計のためのエキスパートシステムも，設計自動化というにはほど遠いものとなるだろう．しかしながら，これらのエキスパートシステムは，スクリーニング的設計のための自動設計システムとして使われ，設計上のトレードオフ等の意志決定上の有効な情報の提供において重要な役割を果たすだろうと思われる．

（5.2節担当　菊地）

５．３　ソフトウェア再利用システム

本節では、ソフトウェア再利用技術を中心に、次世代のソフトウェア再利用エキスパートシステムを構築するために必要となる技術について論ずる。

始めに、ソフトウェア生産におけるソフトウェア再利用技術を概観する。次にソフトウェア生産におけるソフトウェア再利用システムの運用モデルについて述べる。最後に、実用性のある、再利用エキスパートシステムについて論ずる。

５．３．１　ソフトウェア生産における再利用技術

ソフトウェア再利用技術とは、ソフトウェア開発に当り、再利用できる既開発ソフトウェアを最大限流用し、新たに生産するソフトウェアの量を、必要最小限にとどめる技術と考えることができる。目的とするところは、ソフトウェアの信頼性・生産性を向上させることである。

一般に企業内でのソフトウェア生産は、対象業種別に分けて実施されている。すなわち、ビジネスソフトウェア、交換ソフトウェア、端末機組込みソフトウェアなど、業種別に組織化され、生産している。さらに、ビジネスソフトウェアの中でも、金融、流通、製造、病院等々、端末機組込みソフトウェアでも、流通端末、金融端末、自動販売機など細分化してソフトウェア開発が行われているのが実状である。

このような方法は、対応した業種ごとに専門家（システム設計者、ソフトウェア設計者など）が育成でき、業種に対応した知識と経験が、それぞれの技術者に蓄積できるメリットがある。同時に類似ソフトウェア生産も一箇所に蓄積できる。これがソフトウェア生産の現状である。

以上のような組織で、ソフトウェア生産性・信頼性を維持し、向上させていくために必要となるソフトウェア開発支援システムは、各環境に最適なシステムでなければならない。現場では、汎用性よりも個別性を重視する。業種に対

応したソフトウェアを、同一業種について次々と生産した経験・知識から、その業種のソフトウェア開発の関して、最適化された”標準的”な設計技術を構築し、”標準的”な生産物を設定している。各現場で導入できるソフトウェア開発支援システムは、各現場で構築した標準設計法に基づいてソフトウェアが生産できるものでなければならない。このことは、支援システムが適応対象ごとに経験や知識を吸収する機能が必要でありまた適用対象ごとにソフトウェア生産物を再利用するための、支援機能を具備すべきことを示している。ソフトウェア生産現場から観ると、ソフトウェア再利用システムは、きわめて実用性の高いツールであると考えられる。

５．３．２　ソフトウェア再利用システムの運用モデル

ソフトウェア生産は、非形式的な記述による要求仕様から、形式的に厳密な仕様・プログラムへ、段階的に仕様を詳細化してゆく過程ととらえることができる。この変換過程は、さらに「何を開発するかを決定する」という仕様構築の段階と、「どう実現するか」という設計・製造の段階に分けて考えることができる。現在、実用的な再利用システムとして実現しているのは後者を対象としたものである。

設計・製造工程は、各ソフトウェアの生産手順ごとに最適な中間工程を設定し、分割運用されている。各部分工程では、前工程の生産物を入力とし、これを加工し新たな、中間生産物を、次工程へ出力する。

このような設計・製造工程で繰り返し再利用できるものは次の２項目である。

１）　生産物（仕様書、プログラム、テストデータなど）

２）　開発手順（対象業務の知識、開発経験・知識など）

ソフトウェア開発作業で蓄積してきた開発手順を再利用し、開発手順に含まれた知識を集約したものが生産物である。生産物は、それ自体知識の塊りと考えることができる。

ソフトウェアの生産で有効となるソフトウェア再利用システムは、各工程ごとに上記２項目の知識を一体化して使用できるものでなければならない。このような前提条件のもとで、ソフトウェア再利用システムのモデルを考察する。（図５．３．１参照）前工程の生産物（仕様書）を入力とし、次工程へ渡す生産物（仕様書、プログラムなど）を出力する再利用エンジンには、たとえば、

プロダクション・ルール表現などで記述した開発手順知識を実行解釈し、入力を出力に変換する機能（知識インタプリタ）と、各ソフトウェアの生産で現実に使用している生産物を編集するエディタの機能が最低限必要である。

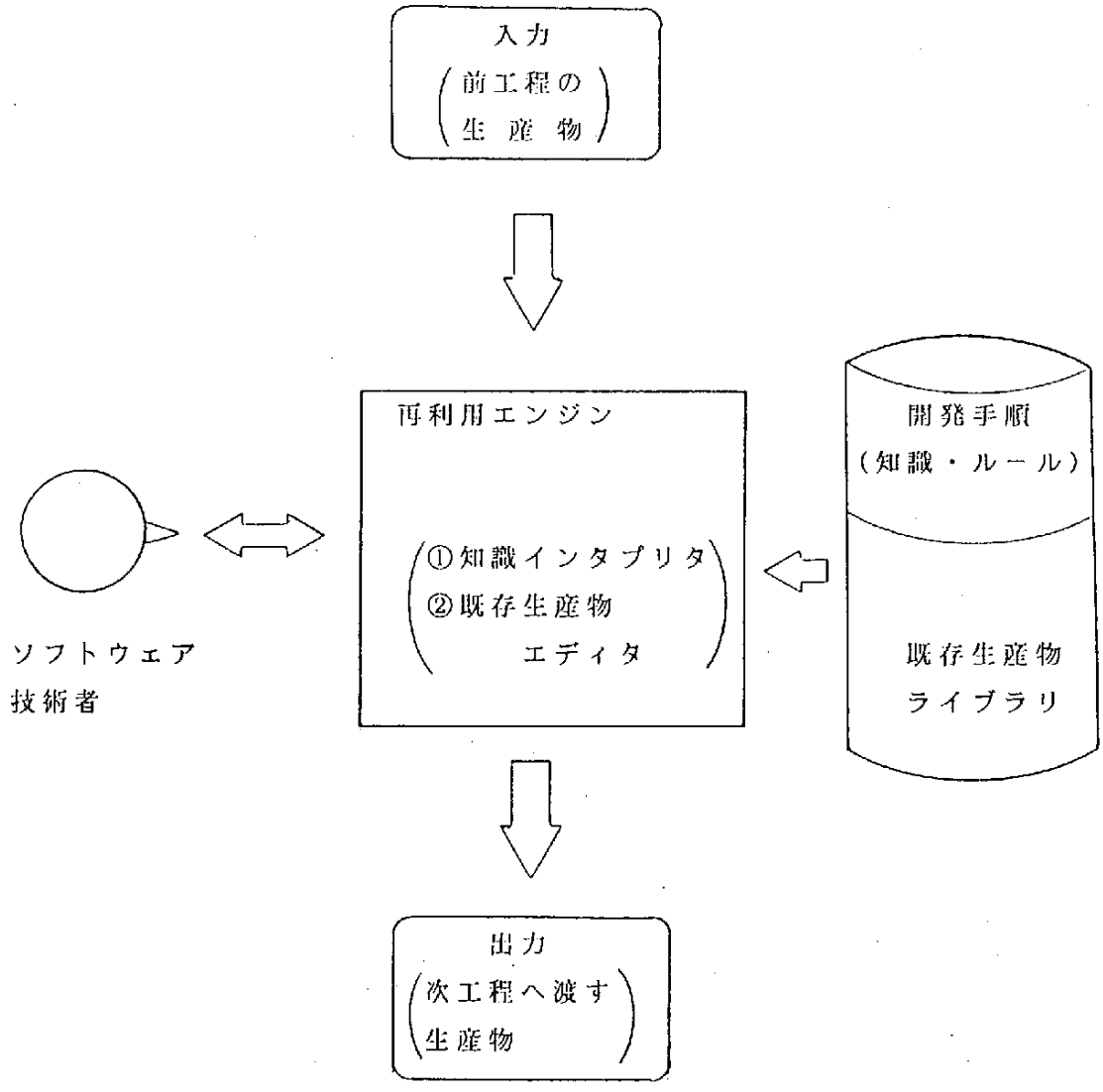


図５．３．１　ソフトウェア再利用エンジンのモデル

ソフトウェアの生産にソフトウェア再利用エンジンを導入する場合、以下のような問題に対処できることが必要である。

１）ソフトウェア生産現場ごとの異なる中間生産物の形式

２）顧客仕様の差異による生産物の構造の変更

ソフトウェア生産現場では、対象業種ごとの生産物および開発手順の標準化を行っている。たとえば、交換ソウトウェアの開発では、状態遷移図を中心とした標準化が行われているが、ビジネス系の開発は、データダイアグラムを中心とした標準化がすすんでいる。

ソフトウェア再利用エンジンのエディタは、中間生産物も含め、すべての生産物を、１枚ごとの図面と見なす。図面上で使用する表記法（シンボル、シンボル使用規則、図面構成文法など）の定義と図面間の関係の定義およびエディタの操作法の定義を与えることにより、各ソフトウェアの生産に合致した生産物エディタが実現できる。

顧客仕様による差異は、さらに２つの問題に分けて考えることができる。

一般に顧客が異なれば、要求仕様は異なることを見込んで開発手順を標準化してある場合と、標準化した開発手順を超えるような要求仕様に対処しなければならない場合である。

前者は、各現場ごとの開発手順・知識をプロダクション・ルールやフレームなどの知識表現を用いて記述し、再利用エンジンにある知識インタプリタを用い、開発手順を再利用できる。入力となる前工程の生産物から、次工程へ渡す生産物への変換のためのルールが完備している場合、ルールをアルゴリズム化し、再利用エンジンに埋め込み、ジェネレータとして、開発手順・知識を効率よく再利用することも可能である。

標準化した、開発手順を超えるような要求仕様に対処するケース、または、標準化した開発手順の一部分を知識化してあるようなケースでは、知識化の範囲内の作業を再利用エンジンに対応させる。残る試行錯誤的要素の多い作業は、ソフトウェア技術者が主体となり、既存のソフトウェアの事例を参照しながら、対話型変換作業を実行できる。

ソフトウェア再利用エンジンは、ソフトウェア生産の状況に合うよう、知識処理、編集処理、対話環境、類似例表示機能を整えておくことが必要である。

５．３．３　ソフトウェア再利用エキスパートシステムに関する考察

工業化されたソフトウェア生産では、既生産物、開発ノウハウを利用する方式を用いている。開発するソフトウェアの機能を中心に詳細化する過程をライフサイクルとして把えることができる（表５．３．１）。

表５．３．１　ソフトウェアライフサイクルの各フェースの作業概要

ライフサイクルフェーズ	作業の目標
1. システム分析計画	●システム化の目的と解決手段の明確化 ●プロジェクトの到達点及び目標の設定
2. システム設計	●ユーザー要求の過不足のない仕様化 ●ソフトウェア構成要素の決定
3. ソフトウェア設計	●ソフトウェア機能要求を実現するソフトウェア方式の決定 ●設計の経済的な実現 （モジュール化，モジュール間インタフェース モジュール内論理）
4. ソフトウェア製造	●ソフトウェア設計仕様の計算機言語への翻訳（コーディング） ●計算機言語への翻訳誤りの除去（デバッグ）
5. システムテスト	●作成されたシステム全体の要求との合致を検査（機能，品質）
6. 運用・保守	●完成システムの使用と誤りの補修 ●使用環境変化に応じた修正・改良

各フェーズ間に、前節に述べたソフトウェアの再利用モデルを、あてはめて考えることができる。ライフサイクルの各フェーズ間の仕様変換は、技術的に次のように分類することができる。

（１）　非形式的要求仕様から形式的機能仕様への変換技術

（２）　形式的機能仕様からプログラムへの変換技術

（３）　プログラム正統性の検証技術

現在、ソフトウェアの機能設計を中心に、これらの変換技術にについて研究開発が行なわれている。

非形式的要求仕様から形式的機能仕様への自動変換の試みとしては、米国スタンフォード大学とケストレル研究所（Kestrel Institute）によるＰＳＩおよびＣＨＩ、米国南カリフォルニア大学情報科学研究所のＳＡＦＥなどがある。

形式的機能仕様からプログラムへの自動変換は、ソフトウェアの働き（Ｗｈａｔ）を規定する仕様から、計算機上での実現手段（Ｈｏｗ）であるプログラムへの自動変換技術である。仕様通りに正確に稼動すると同時に、性能上要求を満たすプログラムが生成されなければならない。現在、事務処理ソフトウェアを対象に、第４世代言語の商業化が活発に行なわれている。これは、事務処理ソフトウェアの対象業務が、仕様と実現アルゴリズムとの差が比較的小さいことによる。一方、科学技術計算、記号処理などは、仕様と実現アルゴリズムが一般には単純に対応しない。一仕様に対し、複数のアルゴリズムが存在する。またアルゴリズムにより性能や、精度が異なってくる。このような技術としてＰＳＩ、ＣＨＩ、米国ＭＩＴのProgrammer's Apprentice がある。

これらの試みとして共通している点は、変換の知識として、ルールを用いていることである。これらのルールとは、対象とする業務の専門家から得たものと考えられる。しかし、ルールを記述した専門家にとっては、極めて常識的な知識をルールとしたものが多い。

ソフトウェア生産における再利用技術（５．３．１）で述べたように、対象業種別に分化したソフトウェア生産現場では、各技術者が専門家である。これまでのルールによってのみ生産を支援するシステムは、経験年数の少ない技術者にとっては有効であっても、専門家にとって必ずしも役立つものではない。

ソフトウェア設計は、機能設計の他に、性能設計、メモリ容量推計、運用設計などがある。機能は、要求仕様をを過不足なく満たすものでなければならないが、要求性能を満たすこと、及び、想定されている計算機の記憶容量を超えないことも当然要求される。さらに、エンドユーザにシステムを提供した後、計算機システムとして運用できることも必要である。（例えば、夜間小人数になった計算機オペレータに対し、頻繁にディスクの入換作業を発生させてはならない）

各設計要因に対し、ソフトウェア生産現場では、表５．３．２に示すような、事例の利用傾向がある。

機能設計に関しては、多くの場合、成功事例を参照しているが、性能、メモリ容量、運用に関しては、ほとんど失敗例に基づいて、失敗を回避する方向で仕様を決定する。また各設計要因が独立な問題として扱えることは稀で、性能上問題のある仕様を改善すると、メモリ容量が許容限度をこえたり、運用上の問題を改善することは、機能設計に影響を与えたりする。

表５．３．２　成功・失敗事例の利用傾向

事例 ＼ 設計（要因）	成功例	失敗例
機能設計	◎	
性能設計		◎
メモリ容量	○	◎
運用設計		◎

このようなソフトウェア生産現場で、実用性の高いソフトウェア再利用エキスパートシステムを構築するためには、事例ベース推論技術に対する期待が大きい。

ソフトウェア再利用に適用する事例ベース推論の技術課題として次のような項目が考えられる。

１）　事例の定義

ソフトウェアライフサイクルの各工程での機能仕様は階層化されているので、階層関係も含めて事例とすることが必要である。また、記憶容量の問題から、オブジェクト指向モデルを適用するなどして事例の抽象化を行なうことが有効とも考えられる。各仕様の構造、仕様を決定するに至った要因なども保持する必要がある。性能、メモリ、運用等に関する事例は、現状の技術者の行動から推して、失敗事例が主に利用される。この場合、失敗の原因となった最初の機能仕様から、失敗が判明し、リカバリーを行なった過程までも事例とできることも必要である。

２）　事例の特徴付け

機能仕様全体を再利用するケースよりも、解を得ようとする問題を分割し、小さな問題に対して、適合する仕様を再利用するケースが多い。ソフトウェア開発は一般に、機能分割を方法論として採用しているため、技術者には、"なじみ易い"と考えられる。一つの事例に対して、機能、性能、メモリ、運用という最低４つの観点からの特徴付けが必要である。

3） 事例の検索

現在の問題解決のために、最類似事例を検索するためには、問題を機能、性能、メモリ、運用の観点から明確に定義する必要がある。類似事例に基づいて修正作業が発生するため、修正工数が最も少ないものを検索できると実用性が高まる。

4） 事例の変更

機能仕様決定という観点から事例を再利用、修正した場合、他の観点（性能、メモリ、運用）からも、同等の修正を行なう必要がある。また、事例の変更を中止した場合、中止した原因を、第3項にフィードバックするプロセスを用意することは有効である。

5） 事例の修復

事例の変更を中止した場合、変更前の事例を残し、事例の変更過程を廃棄する。

以上のような事例ベース推論は、ソフトウェア再利用エンジン（図5．3．1参照）の知識インタプリタ機能の大幅な拡張と、開発手順の中に、既存生産物の利用ノウハウを知識として組込むことにより達成される。この場合再利用エンジンは、次のような運用フローになる。

対象を限定したソフトウェア生産過程では、対象とする業種の専門家である技術者から見て、多くの問題は、常識的な解法を適用することによって解が得られる。従って事例ベース推論を用いて試行錯誤的に解を求める前に、スクーリング的意味も含め、ルールベース推論は必要である。

事例ベース推論を実行するためには、問題自体の分割が必要である。従って、問題を解くための戦略知識が必要となる。この問題分割を実行する機構が、いわば事例ベース推論のマネージャーとなり、機能設計問題解決器、性能設計問題解決器、メモリ設計問題解決器、運用設計問題解決器を協調推論させる方式が考えられる。各問題解決器による協調推論の結果は、すべての設計要因に関し要求仕様を満たした解が得られた時終了する（図5．3．2）。

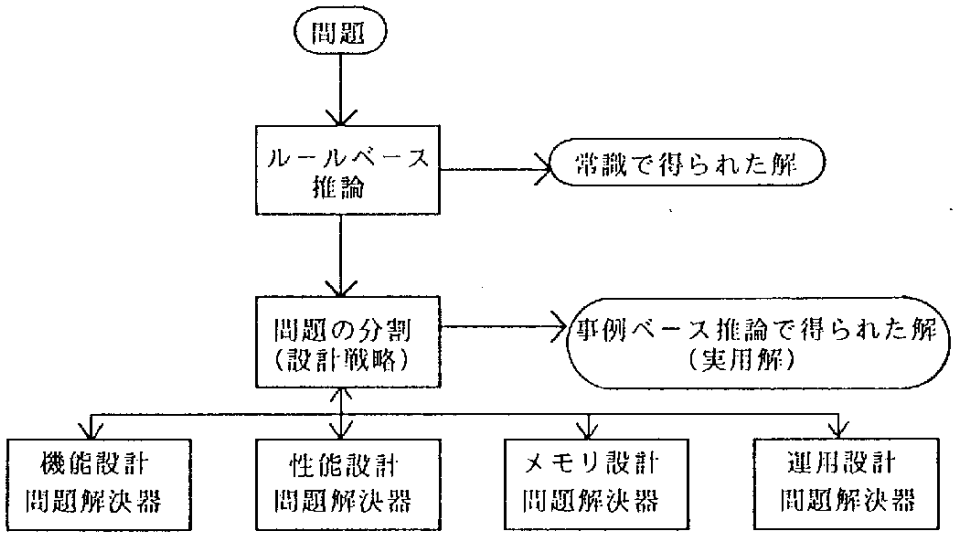


図５．３．２　ソフトウェア再利用エンジンにおける
事例ベース推論制御フロー

（５．３　担当　千吉良）

5.4 回路設計エキスパートシステム

回路設計問題とくにアナログＬＳＩの設計の場合について述べる．演算増幅器や基準電圧回路といったビルディングブロックの設計には労働集約的な労力を必要とし，アナログＩＣ設計のボトルネックとなっており，知識システムはそのボトルネックを解消するものとして期待されている．つまり，知識ベースＣＡＤ（計算機援用設計）ツールによって，システム設計者が，演算増幅器とか基準電圧回路といった共通的なアナログブロックをその動作の詳細をそれほど理解しなくても自由に使って設計を進めることができる，あるいは回路設計者もまた設計の中で頻繁に用いる部品を既設計のパーツとして持つことによりメリットを得ることができる．人間の設計エキスパートの戦略を知識システムに具体化することによって設計過程をスピードアップするだけでなく，システムの全体的なパーフォマンスの改善を可能にすることが期待される．

アナログ構成部品知識ベース設計システムとして，スイスＣＳＥＭ（Centre Suisse d' Electronique et de Microtechinique）のＩｄａｃ［Degrauwe　87］，ＣＭＵのＯａｓｙｓ［Harjani　87］，カリフォルニア大学バークレーのＯｐａｓｙｎ［Koh　87］などがある．ここではこれらのシステムを通して回路設計における知識システムを考える．

1）アナログ回路設計支援へのアプローチ

Ｉｄａｃ，Ｏａｓｙｓ，Ｏｐａｓｙｎシステムは，いずれも共通ビルディングブロックを採用し，サイズ決定回路図を生成する方式を取っている．サイズ決定回路図とはトランジスタやキャパシタ等がどのように接続しているかを示すとともに，各コンポネントのデバイスパラメータ値，例えばＭＯＳトランジスタのゲート幅と長さを備えているものである［Carley　88］．これらのビルディングブロックはセミカスタムのアナログＩＣ設計に使われるような固定された設計ではなく，人間のエキスパートがツールに与えたルールに従って無限に変化するものである．

アナログ回路に対するＣＡＤの登場が遅かったのはアナログ設計のプロセスがディジタル設計よりもはるかに複雑であったからである．ディジタルの論理設計では限られた種類の固定されたビルディングブロックしか必要としない．ＡＮＤゲート，ＯＲゲートといった単純な要素，そして情報蓄積のレジスタ，あるいは演算や論理回路ユニット等さらに複

雑なブロック，メモリ回路等である．アナログ設計では，最良の回路パーフォーマンスを得るために，数多くののカスタム化されたビルディングブロックを必要としている．

アナログ設計支援の知識ベースシステムが，設計し，検査し，多く準最適な設計を棄却する能力を持てば，システム設計者，回路設計者の両者ともその恩恵を被ることができる．システム設計者は内部の働きを理解することなしにアナログビルディングブロックを使用することができ，また回路設計者はシステムの非常に難しい部分の設計に集中することができる．さらに，ブロックの設計が早く自動的に行なわれると，同一のブロックのバリエーションを比較検討して特性値間の最適なトレードオフ，例えば，ゲインとバンド幅とのトレードオフを計ることが可能になる．

2）設計解を求める戦略

設計知識をどのように利用するかは大きな問題であるが，これら3つのシステムにおける知識利用は，だいたい次のような図式にしたがっている．例えば，オペアンプといったビィルディングブロックを合成することを要求されると，システムは知識ベースから特定の回路トポロジー，その回路トポロジーを構成しているデバイスの簡単化したモデル，およびその回路の挙動を記述する方程式群を呼び出す．これらを用いて，ツールは各コンポーネントに最適な可能値を割り当てて，サイズ決定回路図を生成する．

回路トポロジーそのものは設計知識の一部である．システムは新しいトポロジーを発明するのではなく，回路トポロジーのライブラリからあるトポロジーを自動的に一つ選びだして各要素のパラメータ値を決定すればよい．システムは設定可能なパラメータに対してブロックの性能を表現するために知識ベースから解析方程式を取り出して使用する．要素パラメータとはトランジスタのサイズ，閾値電圧等の簡単化されたデバイスモデルのパラメータである．

このような解析方程式を解くこと自体が単純にはいかない．往々にしてそれらは非線形であり，それぞれの変数が複雑に絡んでいる．Ｉｄａｃ，Ｏｐａｓｙｎ，Ｏａｓｙｓそれぞれのシステムは，この方程式をそれぞれ違う方法で取り扱っている．

図5.4-1はＩｄａｃシステムの戦略である．ユーザに選択されたサイズ未決定回路図を用いて，解析方程式を知識ベースにあるより簡単な方程式群にブレークダウンして解に到達する．解が直ちに見つからない場合には，システムは解が見つかるか，あるいは解が存在

しないと判断するまで，徐々に仕様に厳しい条件を付加していく．

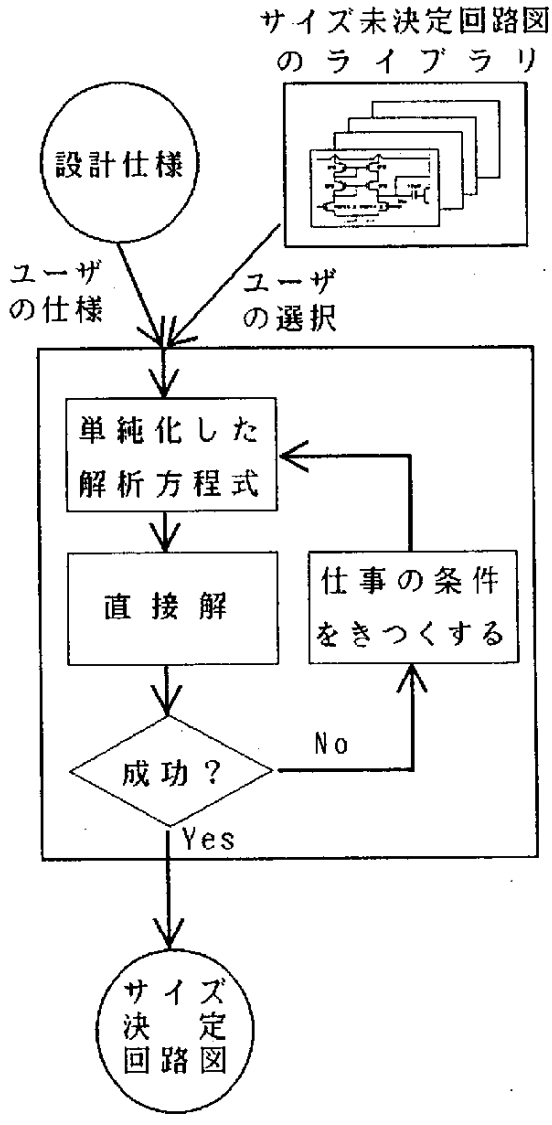


図5.4-1　Idacシステムの戦略

Idacシステムは非線形方程式を直接解いているわけではない．その代わり，設計知識を作る人は設計方程式のすべてについて陽な形で解いておかなければならない．言い換えれば，人間のエキスパートが，それぞれの方程式を解くプログラムコードを逐一詳細に与えなければならない．これは，対象とするすべての事柄について，それを表現する簡略化された設計方程式を解くことができる手続きを逐一与えるように，解析方程式の組を分解，分離する作業をエキスパートがしなければならないということである．

もしIdacが，要求された回路性能を達成できない場合には，システムは新しい性能

仕様に対して別の解を求めようとする．例えば，もし位相マージンについての要求仕様を満足できなければ，システムは内部の位相マージンをきつくして計算を繰り返す．但し，このタイプの繰り返しでは，性能を常に改善できる保証はない．

図5.4-2にＯｐａｓｙｎシステムの方法を示す．このシステムも回路トポロジーはユーザが選択するサイズ未決定回路図によっている．

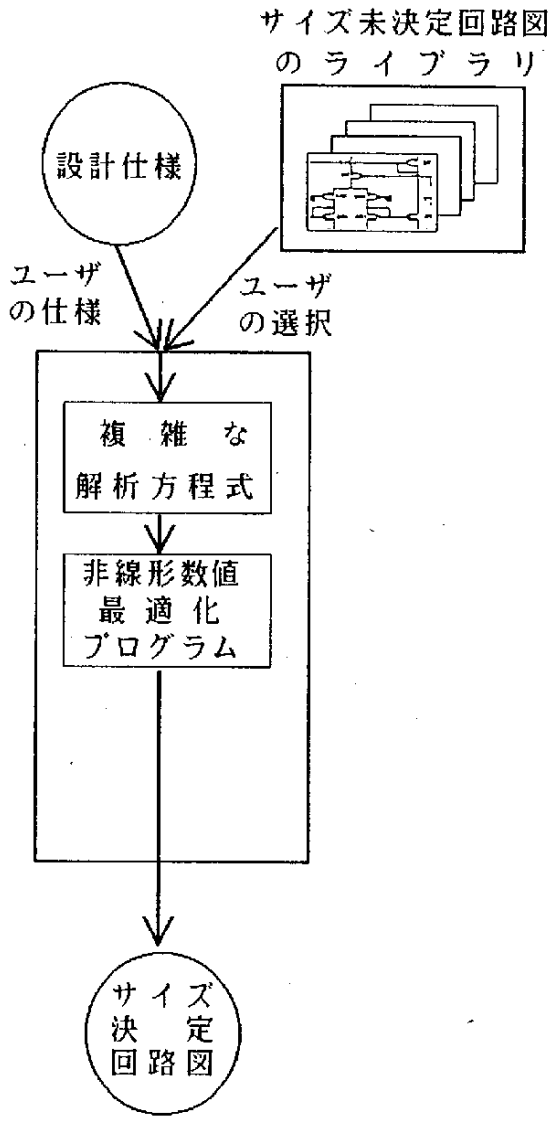


図5.4-2　Ｏｐａｓｙｎシステムの戦略

Ｏｐａｓｙｎは非線形の設計方程式群を最適化機能を使って解いている．最適化機能は，いかに目標とするパラメータ値に近いかという意味で設計解のよさを評価する．Ｏｐａｓｙｎは方程式を解析的に解いているのではなく，数値計算的な非線形問題解決プログラム

を用いて設計対象となる回路の数学的に完全な記述から結果を計算している．システムは，ユーザの要求仕様と設計性能との差が最小になるまで変数を調整する．

最適化のアルゴリズムを用いているためにＯｐａｓｙｎはＩｄａｃよりはるかに遅い．したがってＯｐａｓｙｎはＩｄａｃやＯａｓｙｓに比較して設計の代替案を探索するという意味合いからは魅力に乏しいが，Ｏｐａｓｙｎによるオペアンプは人間のエキスパートが設計したものに匹敵するほどの性能であるという．

Ｏａｓｙｓはシステムが，図5.4-3に示すように，階層的なアプローチをとっている点で，ＩｄａｃともＯｐａｓｙｎとも異なっている．

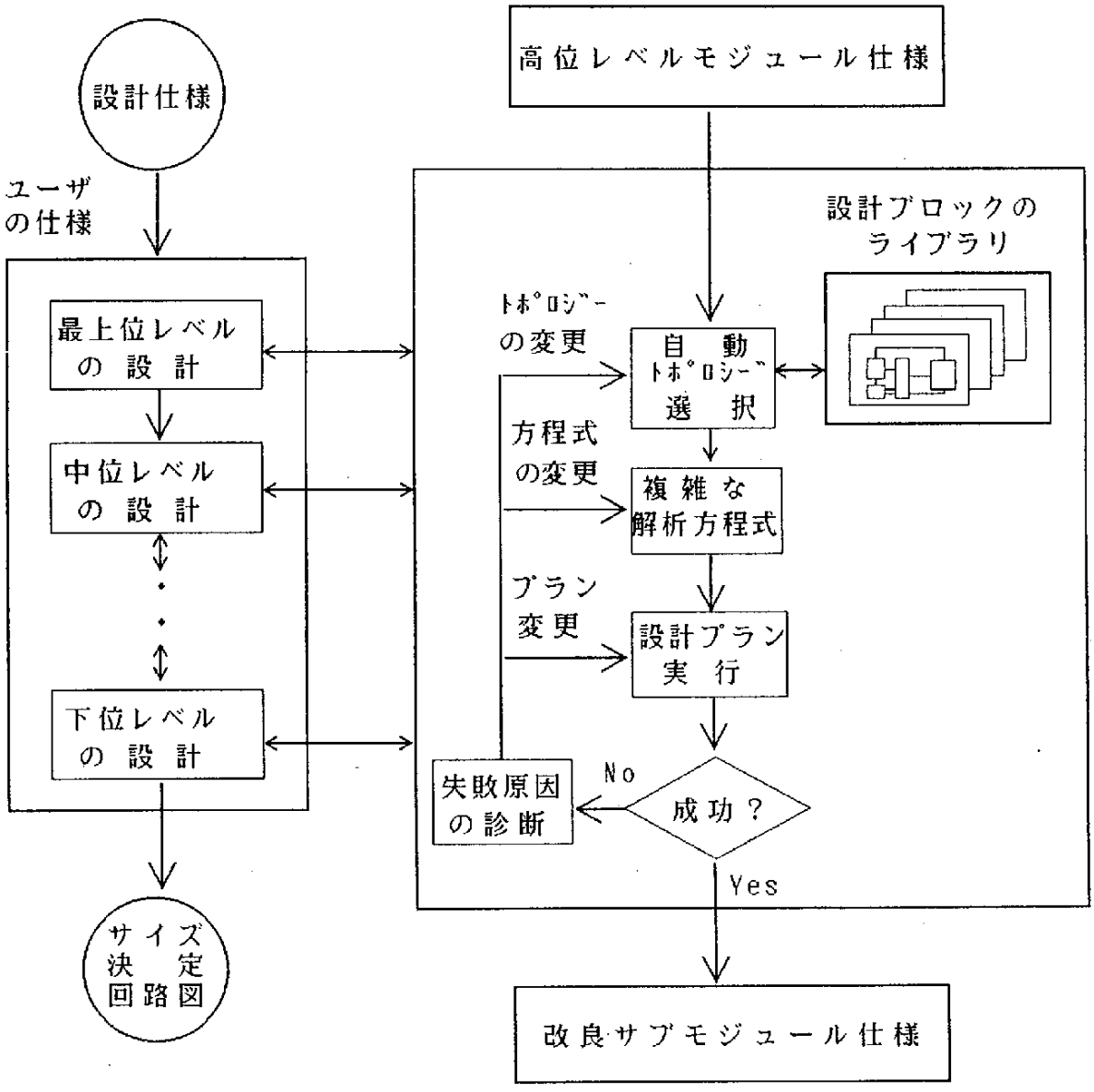


図5.4-3　Ｏａｓｙｓシステムの戦略

Ｏａｓｙｓシステムは個々の設計をサブブロックにブレークダウンし，それらは段階的により単純な方程式群で表現される．予め構成された完全な回路図を使用するのではなく，Ｏａｓｙｓは設計ブロック間の接続を作っていく．もしＯａｓｙｓがある特定のサブブロックに対して適切なパラメータ値を発見できなくなったら，システムは方程式を解くためのプランを変更する．それでもうまく行かないときには，困難を取り除くために設計過程の一段上のレベルの階層に立ち戻って，設計プランの修正を行い解を発見する．

Ｏａｓｙｓではオペアンプのような単純なアナログモデュールでさえも数個のサブブロックに分解される．それぞれのブロックは同時に解くべき設計方程式の比較的小さな集合をもっている．Ｏａｓｙｓは設計知識に基づく仮説からスタートし，方程式を通して，その仮説を調整する計画メカニズムによって方程式群を解く．

Ｏａｓｙｓの階層構造は，設計の多様性という点で有利である．例えば，Ｏａｓｙｓでは現在オペアンプに対して２ブロックレベルのトポロジーしか提供していないが，７２種類のデバイスレベルでのトポロジーの設計が可能である．選択可能となるトポロジーが非常に多いので，熟練した人間の設計者に近い設計を行う事ができる．これに対して，ＩｄａｃやＯｐａｓｙｎでは個々のデバイスレベルでのオペアンプのトポロジーに対し完全な設計知識を要求している．

3）設計空間の探索

設計を進めているアナログ回路は，それぞれの性能仕様が各々の次元軸をもつ多次元空間の点としてみることができ，その多次元空間における一つの点は設計の性能仕様の完全な集合を表現している．ある特定の回路に対する設計可能空間は，設計ツールが実際に回路を作る事ができる点からなっている．種々のアナログビルディングブロックに対する設計可能空間の形状が分かると，システム設計者は，システム内の構成ブロックの間にどのようなトレードオフがあるのかを知ることができる．

例えば，Ｉｄａｃシステムはアナログーディジタル変換回路に要求されたシリコンの面積とパワーとのトレードオフ，その変換回路を支えるディジタル間引きフィルターの面積，パワーのトレードオフを探索した．実験では，Ｉｄａｃはあるシステムレベルの設計者がトータルシステムの面積とパワーが最小になる点を選択するのに大きな役割を果たした．

設計空間を探索する方法の一つはある性能仕様を固定して，残りをすべて変化させることである．この手続きは固定された仕様対変化する仕様の多次元空間におけるプロットを生成する．Ｏａｓｙｓの場合には設計の変化に応じてトポロジーを変えることがでるが，ツールがトポロジーにどのような変化を生じさせたかを示すことができる．経験ある設計者はトレードオフに対する定性的な感覚を持っているが，ツールはそれらの正確な定量的な像を与える．

探索のもう一つの道は設計可能解と不可能解との境界面を求めることである．ツールは，オペアンプのバンド幅と利得といったふたつのパラメータに集中し，残りをすべて固定に保つ．ツールはまず利得の低い値に対して可能な最大のバンド幅を計算する．そして段階的に利得をあげていき，それぞれに対して最大可能バンド幅を計算する．ツールは利得の関数としてバンド幅の包絡曲線を求める．この曲線の下にあればその設計パラメータの組合せは実現可能であり，上にあればその組合せは実現不可能である．ここでもまた，エキスパートの設計者は包絡線の形状についての感覚をもっているものである．しかし設計ツールは設計の実現可能性の限界について極めて正確に指摘することができる．

4）アナログ設計における知識処理機能

回路設計は，トランジスタなどの基本素子を用いて，電子装置を構成するコンポーネントである部分回路をＬＳＩとして実現する設計である．この際，電源電圧，動作スピード，消費電流といったＬＳＩの仕様と，トランジスタ性能，素子のバラツキといったデバイスの条件を考えながら，性能，品質と設計コスト，プロセスコストとのトレードオフを図っていく作業である．アナログ設計の特徴は，条件もアナログであり要求仕様も曖昧性をたぶんに含んでいることである．上で見てきたように，アナログ設計は，回路トポロジーを決定するフェーズと回路パラメータを決定するフェーズからなる．前者は，2.1節で分類した，クラス１の合成型問題，後者はクラス３の合成型問題と見ることができる．回路パラメータ決定フェーズについては，上で見たように，あるクラスの回路については実用的段階になっている．

知識システムにさらに求められる機能としては，回路トポロジー決定フェーズでは，回路全体をまとまった単位のブロックに切り分ける機能，過去の設計事例を設計資産データベースとして蓄積し，要求された仕様に近いものを提示する機能，要求により近づくよう

な回路構成のバリエーションをつくり出す機能がある．回路パラメータ決定フェーズでは，回路定数を要求仕様に近づくように変更する機能が求められる．

［参考文献］

[Degrauwe 87] M.G.R.Degrauwe, O.Nys, E.Dijkstra, J.Rijmenants, S.Bitz, B.L.A. G.Goffart, E.A.Vittoz, S.Cserveny, C.Meixenberger, G.van der Strappen and H.J. Oguey : IDAC:An Interactive Desgin Tool for Analog CMOS Circuits, IEEE Journal of Solid State Circuits, Vol.SC-22, pp.1106-1116 (1987).

[Harjani 87] R.Harjani, R.A.Rutenbar and L.R.Carley : A Prototype Framework for Knowledge-Based Analog Circuit Synthesis, Proceeding of the 1987 IEEE Design Automation Conference, pp.42-49 (1987).

[Koh 87] H.Y.Koh, C.H.Sequin and P.R.Gray : Automatic Sysnthesis of Operational Amplifiers Based On Analytic Circuit Models, Proceedings of the 1987 IEEE International Conference on Computer-Aided Design, pp.502-505 (1987).

[Carley 88] L.R.Carley and R.A.Rutenbar : How to automate analog IC designs, IEEE SPECTRUM, Vol.25 No.8, pp.26-30 (1988).

（5.4節担当：　森）

5．5　ダイヤ編成エキスパートシステム

本節では国内線旅客機のダイヤ編成システムを例にとり，問題の構造と制約条件との関連性によって全体を部分問題に分割し，個々の部分問題を探索問題として定式化するまでの過程を考察する．

5．5．1　問題の概要と制約条件

国内線では1日当たり数百便が各航空会社によって運航されているが，そのダイヤは路線・旅客機の機種と機数・飛行時間・営業要件・整備要件・勤務要件・空港の規模と運用時間帯・他便との関連などの各種条件を満足した上で，必要な便数を各路線に設定したものである．制約条件の数が多いことと，それらが相互に関連していることとが，ダイヤ編成を困難な作業にしている原因である．なお，ダイヤは機種毎に（機種間の制約条件も考慮して）作成されるが，その一例を図5．5－1に示す．全体ではこの約5倍の規模と考えてよい．

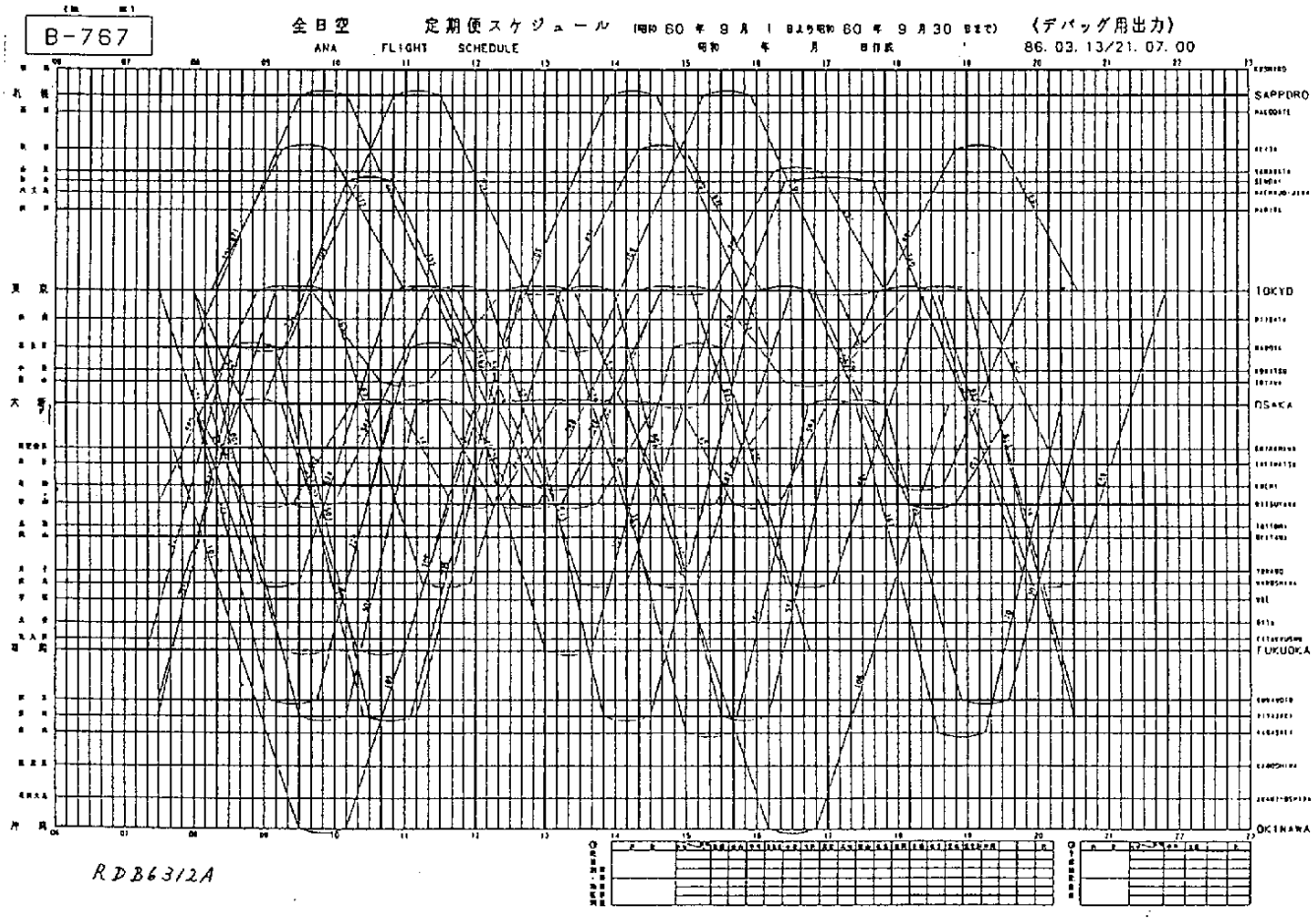


図5．5－1　航空ダイヤの一例

ダイヤ編成における制約条件は多岐にわたるが，大別すると数量的・時間的およびこれらの複合した制約条件になる．主なものを以下に示す．

(1)便数・機数

作成対象ダイヤに必要な，路線毎の便数と機種毎に運用可能な機数．

(2)各空港の運用時間帯

空港毎に運用開始・終了時刻が異なる．また，空港はオープンしていても時間帯によっては発着できない空港もある．

(3)飛行所要時間

路線毎に飛行時間が設定されている（往路と復路とで異なる場合が多い）．

(4)空港最小滞在時間

乗客・貨物の乗換えなどのために必要な滞在時間が空港・機種毎に設定されている．

(5)運航開始・終了時の駐機状態

一日での運航開始時と終了時とで各空港に駐機する機数が機種毎に同一であること．個々の機体は異なっていてもよい．

(6)特定機種の特定空港に対する出発・到着時間帯

機種によっては何機かが特定空港に出発・到着・滞在する時間帯がある．

(7)時刻指定便

特定の便については出発時刻があらかじめ決まっている．

(8)指定日数以下での特定空港への帰還

(5)により各空港を一日毎に巡った結果，幾つかの特定空港間の所要日数が全ての機体について指定日数以下であること．

(9)空港の駐機数

各航空会社に駐機スペースが割り当てられており，運航中はここに駐機する．

(10)便同士の相対時間間隔

同一路線の便は輸送の効率化のため，出発間隔の最小値（厳密ではない）が決まっている．

以上のような条件を満たしつつ，営業が希望する時間帯になるべく近い時刻に便を設定することがダイヤ編成作業である．

5．5．2　問題の分割

(1)タスクレベルでの分割

この問題は全体でおよそ 400便，60機，30空港，15時間の運航，5分単位での時刻設定という規模である．これを部分問題に分割する方針としては，数量を分割することと，質で分割することとが挙げられる．専門家は両方を用いており，数量の面では機種毎に分割し，質の面では制約条件の優先順位で分割している．後者の前提となるのは，少数の制約条件を考慮したラフスケジュールに対して徐々に修正を施すという基本方針である．したがって制約条件は空港運用時間や飛行時間のような物理的な絶対条件，あるいは修正が困難で実運用でも対処しにくい条件を優先的に満足させることになる．もっとも，ある程度ルーチンワーク的になるほど習熟すると先読みができ，一度に考慮する条件が増加する．この問題では次のように分割される．

数量的分割：　機種毎にダイヤを作成する．これによって便数・機数が約5分の1に削減される．

優先順位での分割：　①前説での条件(1)から(7)，②条件(8)，③条件(9)，④条件(10)　の順にダイヤを修正する．

①，②は機種独立に考慮できるが③，④は機種間にわたる条件である．したがってこの順位は機種毎の作成という分割の結果でもあり，機種毎に②までの条件を満足した後，全機種同時に③，④の条件を満足させることになる．

ここまでの分割は，探索の効率化を陽に意識してはいない．数量的分割によって結果的に規模の大幅な縮小がなされるが，作業手順を明確化し，一時点での作業量を削減することにより，間接的に探索の効率を上げることになる．その意味でこの段階はタスクレベルでの分割ということができる．

問題の整理という観点からすると，問題を表現する概念はなるべく少なく，一般化されていた方がよい．この問題のように制約条件が複雑な場合にはそれらの整理・統合を行なうことが重要である．例えばここでいう「空港運用時間帯」は規定によって空港がオープンしている時間帯ではなく，実際には「路線上の航空標識施設が使用可能な時刻以前には飛行できない」という条件を考慮した，実質的な空港使用可能な時間帯のことである．これを徹底すると問題固有に見えた制約条件が一般化され，既存の問題との類似点がうかび上がってくる．

(2)探索効率化手法としての分割

問題がいくつかのタスクとして分割された後，各タスクを具体的な探索問題として定式化する際には，探索効率の直接的な向上を目的とした問題分割を工夫することが必要になる．ここでは例として前説の条件(8)での分割方法を述べる．

条件(8)：　指定日数以下での特定空港への帰還

各空港を一日毎に巡った結果，幾つかの特定空港間の所要日数が全ての機体について指定日数以下であること．

（各機体は一日の運航開始時に，ある空港から出発し，当日の運航終了時に出発地とは一般に異なる空港に到着する．したがって一日毎に各空港を巡ることになるが，特定の空港から再びその空港に戻るまでの所要日数が全機体について指定日数以下でなければならないという条件．）

この条件は前項での優先順位は２番目であるため，上位の優先度の条件を満足したダイヤ（ラフダイヤ）を修正する形で実現することになる．したがってラフダイヤの修正箇所を探索する問題とみなすことができる．修正の１単位は，ある２機体について同一時間帯に同一空港に滞在している部分である．この部分で双方の機体を交換することで一回の修正が行なわれ，その結果空港の一日毎の巡り方が変化する（図５．５－２）．

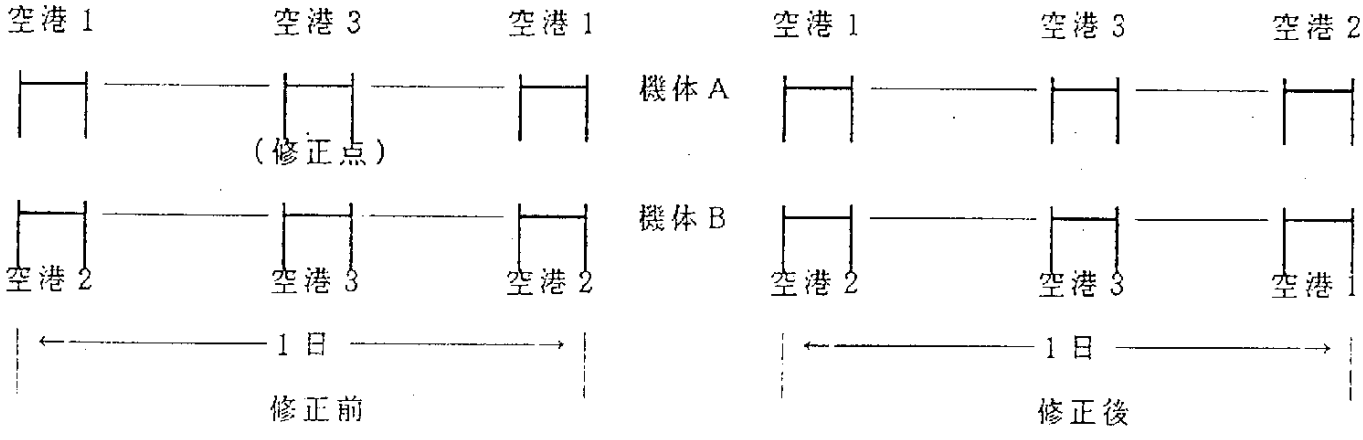


図５．５－２　１回の修正

このような修正点（単に機体を交換できる場所）はダイヤ上に多数あり，しかも条件を

満足するためには一般に複数の修正を行なう必要がある．この探索を効率化するために問題構造を階層的に分割する方法を考える．

前節の条件(5)から，ラフダイヤができた時点での空港の１日毎の巡りを継ぎ合わせるといくつかのループになることがわかる．これらの中には一般に条件(8)を満足するものと満足しないものとがあるが，上で述べた修正を一回実行することは２個のループが修正点で結合して１個のループに変化することに対応する．条件(8)を満足するためには，結合するループは条件を既に満たしているループと満たしていないループとの組合せでなければならない．したがって満足しているループ同士，あるいは満足していないループ同士は結合する必要はなく，それゆえそれらのループを構成する各機体に対する探索は不要になる（図５．５－３）．

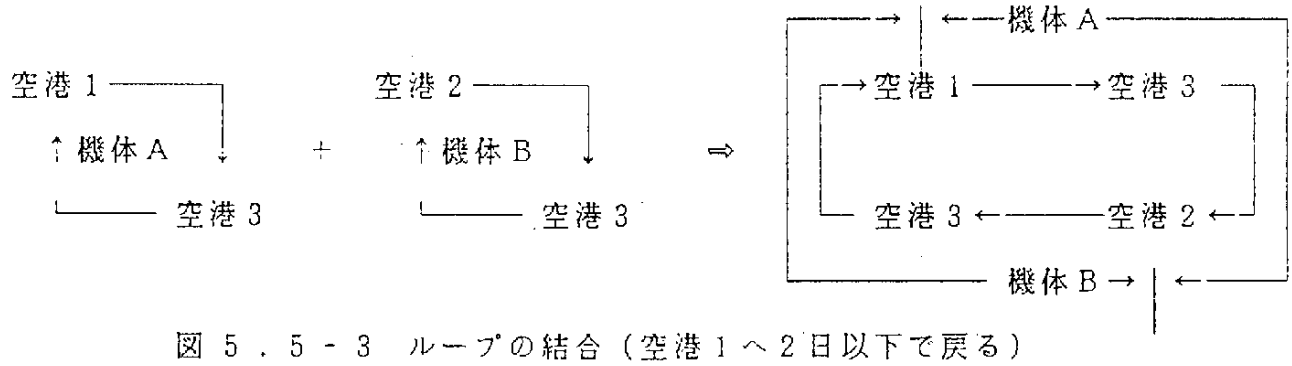


図５．５－３　ループの結合（空港１へ２日以下で戻る）

一般にループは複数の機体で構成されているため，結合する２個のループには結合して条件が改善される機体の組合せと，そうでない組合せとがある．したがって条件が改善される機体のみが修正点探索の対象になる（図５．５－４）．

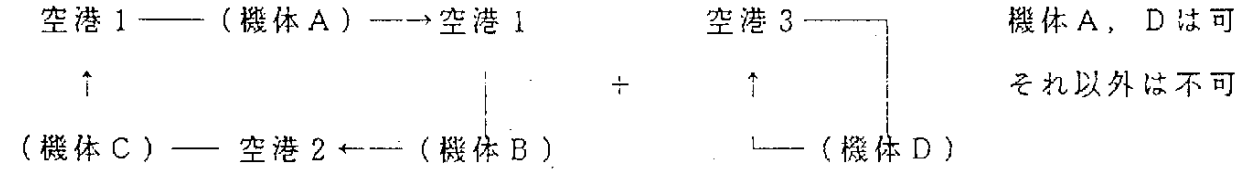


図５．５－４　機体の組合せ（空港１へ２日以下で戻る）

機体の組合せが得られた後，図５．５－２のような修正点を探せばよい．以上のように問題を階層化し，ループレベル，機体レベル，空港レベルの順に上位レベルから探索することにより，探索効率を上げることができる．また，個々のレベルでの探索は小規模になり，単純な縦／横探索でも十分用いることができる．　**(5.5.節担当　中島)**

6．結　論

6. 結 論

6.1 方法論としてのまとめ

本報告書では，方法論が確立されていない計画・設計問題領域を対象として，計画・設計のサイクルにおける計画・設計者の問題解決プロセスに着目して，①探索的側面，②意思決定的側面，③再利用的側面，④知識獲得的側面，の４つがあることを指摘し，各側面を支援する基盤技術として，探索，仮説推論，事例ベース推論および学習を中心に方法論を確立することの必要性を述べ，各側面への接近法および適用事例の紹介をおこなった．ここでは，方法論としてのまとめを行うこととする．

①探索的側面と問題解決的接近：

多くの計画・設計問題では，代替案を何らかの方法で生成して，解析・検査することにより，はじめて代替案の適切さが確定するのが普通であり，したがって適切な代替案の追及は探索問題に帰着する．探索の効率を向上させるために，問題解決戦略を導入したり，可能性の低い代替案の探索を排除するための枝刈り規則を利用することが必要となる．しかし問題解決戦略や枝刈り規則の導入は極めて問題依存的であることを指摘した．計画・設計問題の多くはＯＲ問題として定式化した場合，ＮＰ完全となるため，計算量の壁を克服するための新しいアプローチが要請されており，その中でニューラルネットによる組み合わせ問題の最適化が有望な技術であることを指摘した．

②意思決定的側面と仮説推論的接近：

代替案を生成しても，それが適切であるかどうかを判定することが，その時点で不明の場合には，代替案を仮に選択することが必要になり，この場合，代替案の選択は意思決定の問題となり，選択された代替案は仮説として扱う必要がある．仮説間の依存関係の維持と矛盾が生じた場合には，依存関係に基づく後戻りを必要とするので，仮説推論的な接近が要請される．別の見方をすれば，仮説推論は制約充足の１つの方法を与えるものであり，計画・設計問題への適用においては，既存の探索手法や問題解決戦略を適切に組み合わせることが必要であり，それによって冗長な探索が回避され，結果的に問題解決の性能が向上することが期待できることを指摘した．

③再利用的側面と事例ベース推論的接近：

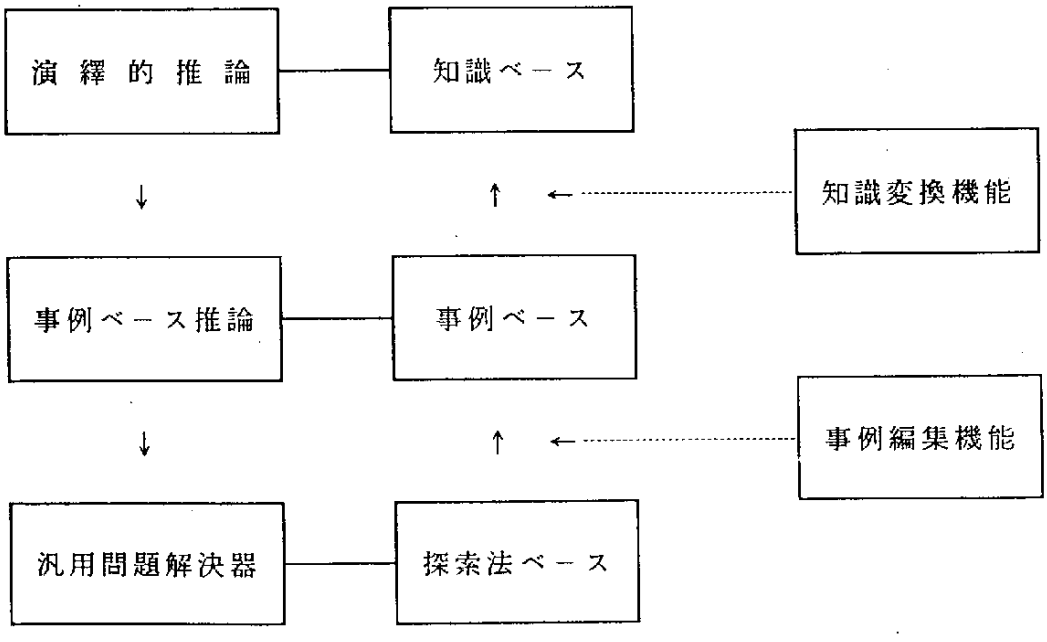
新しい計画・設計問題が与えられたとき，これと類似の計画・設計事例を検索し，類似事例を参考にしながら，問題解決の効率を向上させることは，人間の日常の問題解決において，極めて普遍的行われていることと考えられる．事例を再利用するためには，成功事例および失敗事例に対して特徴づけを行い，与えられた問題にもっとも類似した事例を検索し，これを修正して利用する枠組みが必要であるとして，事例ベース推論の意義を述べ，知識システムを補完する機能として有用であることを指摘した．

④知識獲得的側面と学習的接近：

既存の知識や類似の事例が利用できない新しいタイプの問題においては，計画・設計者が問題解決に有用な知識を自ら獲得していくことが必要になる．そのような過程を観察することにより，知識を獲得する枠組みとして，学習的接近の必要性を指摘し，特に，説明に基づく学習の有用性を指摘した．

6.2 支援ツールのイメージ

計画・設計問題がもつ多元的側面を考えれば，この問題を支援する問題解決システムは柔軟な構造をもつ必要がある．下図にその基本的構成を示す．



計画・設計問題を対象とした問題解決支援システムのイメージ

すなわち，本システムによる問題解決はつぎのような図式で行われる．

①知識ベースを利用した演繹的推論が試みられる．

演繹的推論が失敗に終われば，②へ．

②事例ベースを利用した事例ベース推論が試みられる．

事例ベース推論が失敗に終われば，③へ．

③汎用問題解決器を利用した探索が試みられる．

また，事例ベースおよび知識ベースの更新はつぎのように行われる．

①汎用問題解決器による探索の履歴は，事例編集機能により解析され，

その一部が事例ベースに登録される．

②事例ベースが更新されるごとに，知識獲得機能により解析され，

一般化された知識が知識ベースに登録される．

計画・設計問題領域のように，半ば開いた世界を対象とする場合には，このような柔軟な枠組みの下で徐々に問題解決システムの機能と性能を向上させていくことが有用と考えられる．ここで述べた枠組みは，イメージとして述べたに過ぎず，試作システムが存在しているわけではないが，次世代の知識システムのアーキテクチュアを考えていく上で参考になるものと思われる．

6.3 むすび

以上，本報告書では，計画・設計問題を対象として，知識システムを構築するための方法論の確立の必要性を指摘し，計画・設計のサイクルにおける計画・設計者の問題解決プロセスに着目して，各側面を支援する技術を中心に接近することの意義と事例を通じての考察を行った．

本調査研究では，ツールやマニュアルのレベルにまで詳細化された方法論を展開することまでは試みなかった．しかし，本報告書の内容は，今後，計画・設計問題を取り扱う上で参考になるいくつかの指針や手掛りを与えることができるのではないかと自負している．読者諸氏に何か資するところがあれば幸いである．

（6章担当 小林重信）

平成元年３月発行

発行所　　財団法人　　日本情報処理開発協会

東京都港区芝公園３丁目５番８号

機械振興会館内

電話（０３）４３２－９３９０

６３－Ａ００１