62—A001

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

財団法人
図書
62.9.30
日本情報処理開発協会

昭 和 62 年 9 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６２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。

はじめに

産業界での人工知能応用の現状は，ユーザレベルで実用化に手が届く知識システムがいくつか出現してきたが，これまでに開発された知識システムの多くはデモンストレーションの段階，せいぜいプロトタイプの段階に留っているとみられる．その理由として，

1）研究開発に着手してまだ間がなく，開発途上にあるものが多いこと，

2）マシン，言語，ツールなどプログラミング環境が整備されていないこと，

3）人工知能研究者や知識技術者の数が不足し，教育体制も確立されていないこと，

4）知識システム構築の基盤技術が未熟であること，

5）知識システム構築の方法論が確立されていないこと，

などを指摘することができる．

現在の技術水準のもとで，知識システムの開発を成功に結びつけるには，問題の注意深い選択と知識技術者の献身的な努力が不可欠である．知識システムの構築に従事している現場の技術者にとっては，ＡＩツールが利用可能であっても，システムを構築していくうえでの方法論がないために，試行錯誤的あるいは場当たり的なシステムづくりを行っている事実を見逃すわけにはいかない．

このように，現在は，知識システムの研究開発において１つの転回点に差しかかっており，次世代知識システムのあり方を技術的および方法論的に見直す格好の時期と思われる．

このような現状認識に基づいて，ＡＩセンタでは，昨年９月に，知的情報処理システム調査研究委員会を設置し，つぎの２つのテーマを調査研究の対象として選定した．

（1）知識システム方法論の確立

知識システム方法論の確立に向けて，つぎの４つの視点からのアプローチをとった．

1）知識システムの基盤技術

まず第１世代知識システムの限界について，問題の所在を明らかにした．ついで次世代知識システムの方法論の基礎を与える基盤技術について，知的な問題解決機能を実現するうえで，高次推論技術の必要性を指摘するとともに，それらの技術的課題を考察した．

2）知識システムの開発の手順

知識システム開発の一般的な手順を“問題の設定”から“性能の評価”に至る９段階にまとめるとともに，開発に際し留意すべき事項として，プロトタイピング，システム開発の条件，従来技術との接続，知識源，システム関与者の役割などを考察した．

3）応用分野の特徴とアプローチ

知識システムの対象を解析型問題と合成型問題に分け，基本的な接近法の違いを論じた上で，解析型問題を解釈問題，診断問題，制御問題に，合成型問題を計画問題と設計問題に，それぞれ分類し，各問題の特徴，基本タスクおよび問題解決機能を明らかにした．

4）知識システムの事例分析

７つの知識システム開発事例を取り上げ，それぞれについて，システムの目的と機能，システム構成と実現環境，システム開発の手順およびタスクと問題解決機能を比較可能な形でまとめ，2）および 3）との対応を明らかした．

（2）システム開発ツールの評価

システム開発ツールの評価に向けて，つぎの４つの視点からのアプローチをとった．

1）ツールの評価項目の体系化

ツール評価の枠組みを，①ツールの概要，②ツールの機能／知識表現，③開発者／利用者インタフェース，④システムインタフェース，⑤ツールの性能，⑥ツールの利用目的，⑦ツールが得意とする分野の特性，の７つに分類し，この下に評価項目を詳細化した．

2）性能評価用テスト問題の提案

ツールの性能評価を行うためには，適切に定義された問題が必要との認識から，いくつかの人工的問題を設定した．第１の型の問題は実行性能の評価に係わる問題である．第２の型の問題は応用適合性という視点からつくられた解釈型，計画型，設計型の問題である．

3）標準問題によるツールの評価

ツールの総合的評価を行うための標準問題として，レンズの骨組み設計問題（キャノン提供）および電気設備のトラブルシューティング問題（新日本製鉄提供）を取り上げ，３つのツール ART，KEE3.0 および Knowledge Craft によるモデリングと評価を行った．

4）知識システム開発方法論とツールの評価方式とのリンク

今年度後半より，知識システム開発方法論ＷＧの成果の一部をツールの評価方式の枠組みに反映させることを計画している．

最後に，本調査研究に当たって，ご協力いただいた委員はじめ関係各位に感謝の意を表する．

昭和62年9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	東京工業大学・大学院　総合理工学研究科　システム科学専攻助教授
委　員	関根史麿	㈱花王　知識情報科学研究所
委　員	菊地一成	キヤノン㈱　情報システム研究所　知識工学研究部　第２研究室
委　員	湯井勝彦	新日本製鐵㈱　君津製鐵所　設備部　電気計装技術室　室長
委　員	寺野隆雄	㈶電力中央研究所　経済研究所　情報システム部　知識処理研究部
（主査）		主査研究員
委　員	山崎知彦	㈱豊田中央研究所　７１研究室　７１２グループ　研究員
委　員	千吉良英毅	㈱日立製作所　システム開発研究所　第２部　１０３ユニット
委　員	森　啓	日本電気㈱　Ｃ＆Ｃシステム研究所　応用システム研究部　主任
委　員	中島俊哉	富士通㈱　教育事業部　教育部　第４教育課
委　員	小林慎一	㈱三菱総合研究所　情報システム部　知識システム室　主任研究員
委　員	福島正俊	三菱電機㈱　中央研究所　システム研究部　第２グループ
		グループマネージャー
委　員	森谷正晴	新日本製鐵㈱　君津製鐵所　設備部　電気計装技術室　部長代理
委　員	桃井茂晴	日本電信電話㈱　ＮＴＴ情報通信処理研究所　知能処理理研究室
		主任研究員
委　員	畝見達夫	東京工業大学　総合理工学研究科　システム科学専攻助手
オブザーバ	生駒憲治	㈶新世代コンピュータ技術開発機構　研究計画部　調査役
オブザーバ	大野泰治郎	㈱情報数理研究所　ＡＩ開発部
事務局	平井吉光	㈶日本情報処理開発協会　ＡＩ振興センター

システム開発方法論ワーキンググループ

委員（主査）	小林重信	東京工業大学・大学院　総合理工学研究科　システム科学専攻助教授
委員	湯井勝彦	新日本製鐵㈱　君津製鐵所　設備部　電気計装技術室　室長
委員	山崎知彦	㈱豊田中央研究所　７１研究室　７１２グループ　研究員
委員	森　　啓	日本電気㈱　Ｃ＆Ｃシステム研究所　応用システム研究部　主任
委員	千吉良英毅	㈱日立製作所　システム開発研究所　第２部　１０３ユニット
委員	中島俊哉	富士通㈱　教育事業部　教育部　第４教育課
委員	小林慎一	㈱三菱総合研究所　情報システム部　知識システム室　主任研究員
委員	福島正俊	三菱電機㈱　中央研究所　システム研究部　第２グループ　グループマネージャー
オブザーバ	藤井祐一	㈶新世代コンピュータ技術開発機構　第５研究室　室長
事務局	平井吉光	㈶日本情報処理開発協会　ＡＩ振興センター

ツール評価ワーキンググループ

委員（主査）	寺野隆雄	㈶電力中央研究所　経済研究所　情報システム部　知識処理研究部　主査研究員
委員	関根史麿	㈱花王　知識情報科学研究所
委員	菊地一成	キヤノン㈱　情報システム研究所　知識工学研究部　第２研究室
委員	森谷正晴	新日本製鐵㈱　君津製鐵所　設備部　電気計装技術室　部長代理
委員	桃井茂晴	日本電信電話㈱　ＮＴＴ情報通信処理研究所　知能処理研究室　主任研究員
委員	畝見達夫	東京工業大学　総合理工学研究科　システム科学専攻助手
オブザーバ	生駒憲治	㈶新世代コンピュータ技術開発機構　研究計画部　調査役
オブザーバ	大野泰治郎	㈱情報数理研究所　ＡＩ開発部
事務局	平井吉光	㈶日本情報処理開発協会　ＡＩ振興センター

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－

目　次

1. 知識システムの基盤技術

1．知識システムの基盤技術

1.1 問題の所在

ここ数年，人工知能の基礎研究は新しい研究のパラダイムを模索して，ある意味で混迷状況にあるにもかかわらず，産業界ではやや加熱気味のＡＩブームが続いてきた．これは産業界のやや過大な期待とマスコミのやや誇大な宣伝の相乗効果によるところが大きい．産業界における人工知能応用の現状は，ユーザレベルで実用化に手が届く知識システムがいくつか出現してきたものの，これまでに開発された知識システムの多くはデモンストレーションの段階，せいぜいプロトタイプの段階に留っていると認識するのが妥当であろう．その理由として，研究開発に着手してまだ間がなく開発途上にあること，計算機，ＡＩ言語およびＡＩツールなどプログラミング環境が整備されていないこと，知識技術者が不足していること，などが指摘されている．しかしもっとも大きな理由は，知識システム構築の基盤技術が未熟であり，方法論的に十分整備されていないことにある．

現在の技術水準のもとで，知識システムを成功させるには，問題の注意深い選択と知識技術者の献身的な努力が不可欠である．その意味で，現在の知識工学は未だアートの段階に留まっているといえよう．アートの段階を脱却して，知識工学が真の意味での工学の地位を確立するには，知識システムの研究開発の現状を冷静に分析したうえで，次世代知識システムの基盤となる技術を発展させるとともに，知識システム開発の方法論を整備することが必要と思われる．

知識システムを実際に構築することに従事している現場の技術者にとっては，ＡＩツールが利用可能であっても，システムを構築していくうえでの方法論がないために，試行錯誤的あるいは場当たり的なシステムづくりしかできず，小規模システムでは通用しても，大規模システムにおいては，従来のプログラミングにおけるソフトウエア危機と同じような問題が生起することは必死とみられる．

その意味で，現在は，知識システムの研究開発において１つの転回点に差しかかっているとみられる．これまでに開発された知識システムを第１世代システムと呼ぶことにする．

次世代知識システムのあり方を方法論的に見直す格好の時期と思われる．

1.2 第1世代知識システムの限界

本節では，第1世代知識システムの方法論的側面について，その限界を考察し，次世代知識システムにおいて考慮すべき事柄を考察する［小林 86-2]，[Kobayayshi 86]．

1.2.1 領域選択の問題

システム科学的接近では，対象を数理的なモデルとして明確に定式化することが前提とされる．したがって，システム科学的接近の対象は well-defined，well-structured な性質をもつことが要請される．これに対し，知識工学的接近では，従来のシステム科学的接近では困難とされていた ill-defined，ill-structured な対象が問題とされる．とはいっても，それは相対的なことであり，基礎研究の対象としてでなく，実用化の対象としては well-defined，well-structured に限りなく近い問題を注意深く選択する必要がある．現在の知識システムの基盤技術のもとで完全に取り扱い可能な領域はそれ程広くはない．その限られた範囲内での問題の選択が知識システムの成否を左右するといっても過言ではない．したがってシステム化の対象とする領域の選択は極めて重要であるが，この段階に対し，現在のところ人工知能や知識工学が貢献するものは何もない．現状では，試行錯誤的に問題の取捨選択がなされているとみられる．この段階では，システム工学的方法論に基礎をおいたシステム分析が重要な役割を果たすと思われる．

1.2.2 知識表現の問題

最近のＡＩツールの多くはいくつかの知識表現パラダイムを提供するようになっている．しかしそれはあくまで記号レベルでの知識表現に限られており，感覚データやアナログデータあるいはシステム構造の表現については未だ確立された表現形式をもたない．電動機の故障診断を例にとれば，診断に必要なデータとして，異常音，におい，焼け焦げのパターンなど感覚データが重要な役割を占めている．現状では，感覚データの解釈はそれ自体知識システムの研究対象であり，これらが知識工学の技術の中に取り込まれなければ，現実世界への真の応用は困難とみられる．

知識表現を記号レベルに限ったとしても，どのようなパラダイムを選択するかは知識技

術者の載量の問題とされ，最適な知識表現とは何か，より柔軟な知識表現はどうあるべきかなどについては未だよく分かっていない．

さらに大規模人工システムを対象とする場合，システム構造や構成要素を階層的あるいは抽象的に表現することが必要になる．しかし具体的に階層化や抽象化をどのように行うべきかの指針となる基準は確立されておらず，試行錯誤的になされているのが現状である．

1.2.3 推論制御の問題

現在までに開発されている知識システムの多くは，均質で完全な知識ベースと演えき的推論の組み合わせをベースにしている．すなわちデータ構造としては，ルール，フレーム，意味ネットワークなどの違いがあっても，推論の制御方法としては，プロダクションシステムの基本制御ループ，すなわち認識－行動サイクルを採用している．

知識ベースと推論機構の分離は，知識のモジュール化を促進し，システムの拡張性を容易にするというのは，建て前論であって，実際のところ制御的知識を知識ベース上におかなければ，推論過程をうまく制御することは不可能である．これは“制御の飽和問題”と呼ばれるもので，システムの保守可能性と拡張性を著しく損なう．制御の飽和問題を最小化するようなアーキテクチュアの開発が望まれている．

知識ベースの完全性という要求は演えき的推論にとって絶対的条件であるが，これは知識獲得に大きな負荷を課すことになる．人間がもつ知識にはあいまいさ，不完全さ，誤りなどが伴うことは不可避である．にもかかわらず，完全性を要求することは基本的に無理があるとみられる．あいまいで，不完全で，誤りを含んだ知識を前提として，これらを推論の側で吸収するような柔軟な推論メカニズムを開発することが望まれる．

1.2.4 知識獲得の問題

知識獲得は狭義には与えられた知識表現の枠組みのもとで，知識源から知識を抽出し，これを定められた知識表現形式変換するとともに，知識の管理を行うことである．しかし広義には問題の設定，既存技術の評価，知識源の同定，専門家モデルの同定，ユーザモデルの同定をも含めて考慮すべきである．

専門家からの知識獲得はシステム構築において重要な役割を果たす．しかしこれを重要視する余り，専門家の知識に全面的に依存することにはリスクが伴う．なぜなら知識獲得によって抽出される専門家の知識は彼がもつ知識の一部であり，知識の断片はそれ自体独

立した存在というよりは，むしろ相互依存の関係にある傾向が強いため，部分的に抽出された知識の中にはそれを正当づける根拠をもたない表層的な知識も含まれることになり，問題解決の文脈によっては，思わぬ副作用を引き起こす恐れがあるからである．特に，人工システムの場合には，設計仕様書や保守報告書など他の知識源からの情報を有効に活用すべきである．

人間の専門家からの知識の抽出には，インタビューやプロトコル解析などが採用されている．しかしこれらの方法では問題解決の典型的な局面についての経験則が抽出されるだけであり，知識をシステマティックに抽出するための方法論の開発が望まれる．

1.2.5 知識ベース管理の問題

制御システムのように，高信頼性が要求されるシステムでは，知識システムの完全性や無矛盾性の確保はきわめて重要である．知識ベースの完全性や無矛盾性の検証は，ホーン節のように非常に限定された知識表現形式を除いては，理論的な方法が確立されていないために，実際上極めて困難であり，シミュレーションに頼らざるを得ないのが現状である．しかしシミュレーションでは自ずとその検証範囲が限定される．

均質な知識ベースと単純な演えき的推論の組合せという枠組みを採用する限りにおいて，この問題の根本的な解決はない．知識ベースが不完全であること，そして矛盾が存在することを前提として，そのもとでも妥当な推論結果を導き出せるような頑健で，知的な推論制御の枠組みを確立することが望まれている．

1.2.6 合成型問題への接近

計画や設計問題の多くは，システム合成問題として扱われる．システム合成では，システムに対する要求仕様が与えられたとき，これを実現するサブシステムの特性およびシステム構造を規定することが要請される．一般にシステム構造は無限に存在する．システムの構造を決定しない限り，サブシステムの特性の最適決定はできない．一方，サブシステムの特性が与えられない限り，システム構造の最適決定はできない．そこにこの問題の本質的な難しさがある．現在のＡＩツールはこれらについてはほとんど何もサポートしてくれないといってよい．

計画や設計型問題では，推論の制御方式よりも，問題解決の基本制御ループの設計が重要である．専門家がもつ問題解決戦略を参考にすることが最初になすべきことであるが，

それをそのまま採用することが知識システムの問題解決にとってベストである保証は何もない．システム論的観点から最適な問題解決戦略を選択することが必要である．

どのような問題を対象とするにせよ，その問題がどのようなクラスの問題に属するかが明らかになれば，基本的な問題解決戦略は自ずと定まってくるはずである．現在の方法論では，システム構築がボトムアップ的になされるために，問題解決の本質的な部分が知識ベースの中に埋め込まれてしまう恐れがある．

システム合成問題に対し，トップダウン的接近を可能にするには，対象とする問題のクラスを構造化しておくことが必要である．

1.2.7 ＡＩツールの問題

LISPやPROLOGを用いて直接知識システムを構築することに比べれば，ＡＩツールは非常に便利な手段といえる．しかし現在のＡＩツールは，もっとも進化したツールであっても，ＡＩにおける問題解決のための方法論のごく一部を提供しているに過ぎず，総じて低レベルでのモデル化能力しか提供していない．ＡＩツールは，通常のプログラミング言語と同様に，知識システム構築の最終局面で利用すべきものである．にもかかわらず，ＡＩツールをあたかも対象領域のモデルであるかのごとく錯覚して，ＡＩツールに合うように，対象をモデル化してしまう危険性がある．より高次の推論能力および問題解決能力をＡＩツールに付与することがこれからの課題と考えられる．

1.3 基盤技術の技術的課題

前節での議論を踏まえて，本節では，次世代知識システムの方法論の基礎を与える基盤技術について考察を加える［小林 86-2],[小林 87].

1.3.1 知識表現

図1.3-1 に知識表現のパラダイムとモデルを示す．図1.3-1 の外側の扇形はパラダイム，内側の扇形はモデルをそれぞれ表す．知識表現の主たるパラダイムは論理指向・対象指向・手続き指向・ルール指向の４つであり，この下に種々の知識表現モデルが提案されている．意味ネットワークは論理指向かつ対象指向，フレームシステムは対象指向かつ手続き指向，プロダクションシステムは手続き指向かつルール指向，Prologはルール指向かつ論

理指向であることを図1.3-1 は示す．

LOOPSやARTなど現時点でもっとも進んでいるとみられるツールは，図1.3-1 に示されるパラダイムやモデルをハイブリッド形式でサポートしている．しかし，問題解決のためにどのパラダイムを選択するかの判断は知識技術者に委ねられている．パラダイムの選択は要請される問題解決の型，ユーザの利用の形態，推論効率，記憶効率，知識獲得および保守の容易さなど種々の観点からの評価に基づいてなされるべきである．各種モデルを組み合わせて使うにせよ，知識獲得，問題解決および保守の各局面を通じて，固定した形式である必要はなく，何らかの経済原理にしたがってシステム自身が知識表現を動的に再構成できる柔軟な枠組みが要請される．

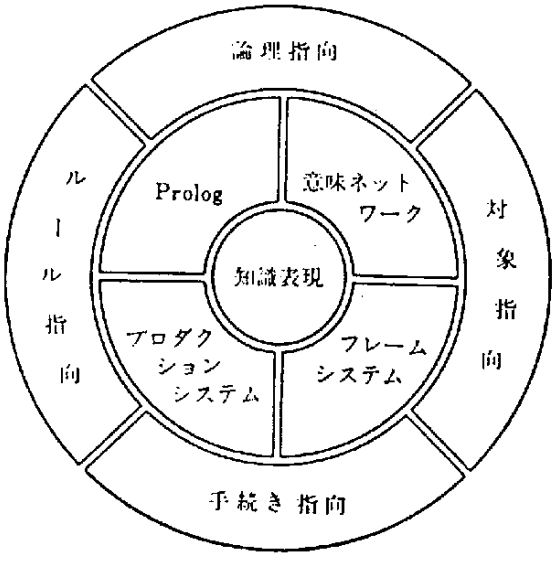


図1.3-1 知識表現のパラダイムとモデル

第１世代知識システムは，対象を狭い問題領域に限定化することにより，扱う知識のクラスを領域に依存した固有の専門知識と専門家の経験的知識だけに絞ってきた．そうすることにより常識の問題を回避してきたといえる．しかし限られた専門知識といえども，その背後の知識として常識を必要とすることが，第１世代システムの経験から指摘されている．知識システムの適用範囲を広げるためにも，常識の利用は必要と考えられる．今後は常識に基づく知識ベースと専門知識に基づく知識ベースの統合が重要な課題となろう．

1.3.2 高次推論

図1.3-2 に推論制御のパラダイムとモデルを示す．推論制御の主たるパラダイムはあいまい性・不完全性・非決定性・並列性の４つであり，この下に種々の推論制御モデルが展開される．デフォルト推論はあいまいで不完全な知識を対象とする．不確実性推論はあいまいな知識を対象とするもので，必然的に並列処理の問題に関係する．協調型推論は制御の非決定性の問題を克服することを意図しており並列処理の問題に関係する．あいまいで不完全な知識を対象として，制御の非決定性の問題を回避するうえで，図1.3-2 に示す推論制御モデルは重要な役割を担う．知的な推論制御の導入は，結果的に知識獲得問題の緩和を促進する．

第１世代知識システムでは知識の表現が中心的課題であったのに対し，次世代知識システムでは知的な推論の制御が中心的課題になるものと見込まれる．

知的な推論の制御とは，従来の演えき的推論に基づく単調推論ではなく，非単調推論を基本とするもので，不確実性推論，デフォルト推論，仮説推論，分散協調型推論，定性的推論，類推などが主たる内容を形成する．これらは現在，興味深い基礎研究分野を形成しているが，第１世代知識システムにおいてはその結果の一部が導入されたに過ぎず，技術的には未成熟の段階にある．これら推論の制御に関連する諸理論および諸技術が次世代知識システムの基盤技術を形成するのは必然とみられる．

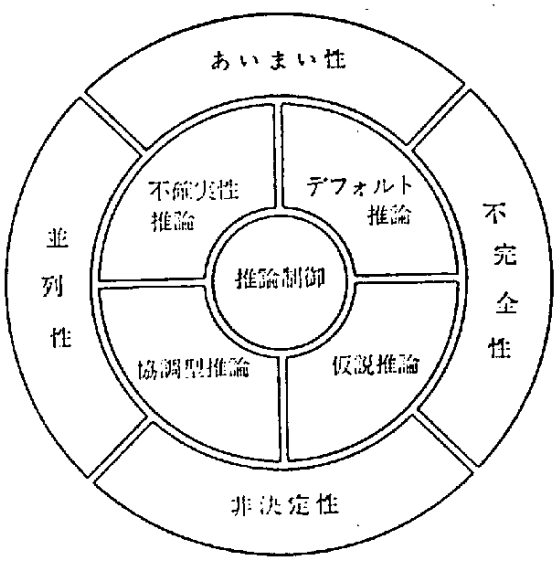


図1.3-2 推論制御のパラダイムとモデル

これらの問題については，以下1.3.3～1.3.8 において詳しく議論する．

1.3.3 不確実性推論

従来，ＯＲの問題などにおける不確実性の問題は，ベイズ確率に基づくアプローチが主流であったが，MYCINにおいて不確実性の取り扱いに対する新しい方法が提案されたのを契機に種々の理論や方法が開発され，現在ではMYCINの方法， Dempster-Shaferの方法およびファジー理論に基づく方法などが広く使われるようになってきた．ここではこれらの方法に共通する問題点を考察する．

第１に，確実度の割り当ての困難さの問題がある．確実度のレベルを言語的表現に対応させて，人間に表現しやすい形式にしたところでそれは見掛上のことであって本質的な難しさは変らない．人間は基数的情報を述べることがもっとも不得意である．したがって人間の専門家から取得した確実度に関する知識は初期値的な意味しかもたず，システム側で確実度に関するパラメータのチューニングを行わなければ使いものにならないのが実際である．パラメータのチューニングが行えるためには，十分なテストデータを必要とする．したがってテストデータをもたない未知の状況に対する推論結果の妥当性を主張する根拠は極めて乏しいことになる．

第２に，複数の根拠に基づく統合化の問題がある．いずれの方法においても統合化規則は約束事であり，強い理論的裏付けをもつものでもなければ，また人間の直観を正確に反映したものでもない．仮説を肯定する根拠が複数存在する場合，その仮説の確からしさが増すのは直観に合う．しかし仮説を肯定する根拠と否定する根拠が共存する場合，統合化によって確実度が両者の中間的な値になり，“未知である”と推論してしまうことについては問題がある．人間であればどちらかの根拠が誤っているのではないかと考え，それぞれの根拠の正当性を別の知識を使って検証することをするであろう．ここに機械的な統合化の基本的な問題が存在する．

不確実性の取扱いを可能にするために，確実度またはそれに類似の概念を導入することは必要なことであるが，確実度の最適化を行わなければ実用に耐えないとすれば，この枠組みは小規模な知識ベースにしか適用できないであろう．事実，実用化例が数多く報告されているファジィ制御分野では，知識ベースの規模はたかだか数十ルールであり，しかも推論が多段に及ぶケースは稀である．

不確実性推論を大規模知識ベースに対しても利用可能とするためには，もっと頑健な枠組みを構築する必要がある．前述したように，人間にとって基数的判断は困難なことであり，一方序数的判断，特に一対比較判断は比較的容易である．“知識Ａは知識Ｂよりも確かである”とか“前提Ａが成り立つとき，結論Ｂの方が結論Ｃよりも確実に主張できる”といった確実度に関して序数的な仮定のもとでのあいまい推論の枠組みを構築することが今後必要とされよう．もし現在の枠組みのもとでの展開を考えるならば，von-Neuman-Morgensternの効用測定法に類似の確実度計測法を確立すべきである．また統合化についても理論的な基礎を固めなければならない．

1.3.4 デフォルト推論

不完全な知識体系のもとでは，デフォルト推論が重要な役割を果たす．UNLESSオペレータ，閉世界仮説，サーカムスクリプションなどデフォルト推論のためのプリミチブは数多く提案されている．ここでは,性質継承を基本とするフレーム表現について，特に多重継承の問題について考察する．

フレームのクラス階層構造がツリー構造を有するとき，すなわち子のフレームが必ず１つの親フレームしかもたない場合には，親フレームからの性質継承は唯一であり，デフォルト推論における問題は存在しない．問題はクラス階層構造がネットワーク構造を有する

とき，すなわち子のフレームが複数の親フレームをもつことを許した場合，多重継承の問題が生じる．

まず継承戦略の問題である．縦型探索戦略，横型探索戦略，荷重和最短パス戦略およびいくつかのバリエーションが考えられるが，戦略の違いにより，当然のことながら性質の継承も異なってくる．いずれの戦略も一長一短あり，優劣をつけることは困難である．問題はいずれの戦略を採用するにせよ，知識ベースの管理に対し強い制約を課すことにある．

フレーム表現は均質な表現方法に比べてトップダウン的な知識表現であるために，知識ベースの管理という観点からは好ましい性質を有する．しかし組み込まれた継承戦略による制約のために，モデル化の意図とは異った予期せぬ副作用を生じる危険性を内在している．これを抑えるために，スロット単位で継承の範囲を記述するなどのアイデアが提案されているが，局所的には便利な機能であっても，大局的にみて妥当な機能かどうかは疑問視される．

このような問題は常識を含む大規模知識ベースにおいては重要かつ深刻な問題を提起する．多重継承という文脈の中でデフォルト推論による問題点を見直し，理論的および技術的な解決策を見出すことが必要と思われる．

1.3.5 仮説推論

矛盾する知識を含む知識ベースでは，矛盾する知識は競合するために，これらは仮説として扱われる．仮説はそれを支持する根拠仮説の状態により，“信じられている”または“信じられていない”の２通りの状態をとりうる．仮説間の支持関係は依存関係と呼ばれる．仮説に基づく推論が失敗したとき，依存関係に基づく後戻りが必要になる．依存関係に基づく後戻りを知的に行うためには，これを制御するためのメタ知識を必要とする．Doyle のＴＭＳは依存関係の修復を行うための理論的基礎を与えるものであるが，後戻りを知的にする方法は与えていない．

一方，de KleerはＴＭＳを多重世界に拡張して，無意味な後戻りをできる限り回避するＡＴＭＳを提案しているが，組み合わせ的爆発の問題を内在している．依存関係に基づく後戻りを探索制御の問題と考えれば，ＴＭＳは縦型探索であり，ＡＴＭＳは横型探索に相当する．

計画や設計問題は基本的には“生成－検証”パラダイムに基づく探索を必要とするために，仮説推論を必要とする問題領域である．

ＴＭＳや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ただけであって，計画や設計問題の組合せ的爆発問題を克服するという本質的な課題を解決するものではない．したがって彼らのモデルの上に知的な制御機構を実現することがこれからの研究課題と考えられる．

1.3.6 分散協調型推論

分散協調型推論とは，独立した制御機構およびデータ構造をもつ知識源と呼ばれる自律的構成要素が分散され，それらが疎に結合された集りによって協調的に問題解決を行う形の推論をいう．どの知識源も単独で問題全体を解決するのに十分な知識や情報はもたない．各知識源は情報を相互に共有したり，通信し合うことによって大局的な問題解決に向けた協調を必要とする．

現在の知識システムは知識の表現がルール型，フレーム型，関係型などいくつかのパラダイムに基づいているが，制御機構については基本的にはプロダクションシステムの枠組みを採用している．プロダクションシステムの最大の短所は，すべての状況に対応できる単純で強力な競合解消戦略の設計が困難なことにある．これは制御の飽和問題と呼ばれる．

協調型推論は，制御の飽和問題を克服するうえで有効な枠組みを提供する．協調型推論の利点は,1)確かな知識やデータの誤りを吸収できるので高い信頼性が得られること，2)モジュラリティが高く，概念的に明解であるので，システムの拡張を容易にすること，3)本来分散的な性質をもつ問題を自然にモデル化できること，4)ハードウェア化により，高速処理が期待できること，などにある．

知識システムにおける協調型問題解決のモデルとして，黒板モデルや契約ネットモデルがある．

分散協調型推論では，制御の分散と通信の局所化のバランス，いいかえれば各知識源の自律性の確保と知識源間の通信のボトルネックの回避のバランスを保つことが重要な課題である．各知識源が展開する局所的な問題解決活動が大局的な問題解決に一貫して貢献することが望ましいことであるが，そのために調整のための強力なプランナーを配置し，問題解決状況に応じて各知識源に指令を出すような枠組みでは通信に大きな負荷がかかり，柔軟な問題解決は実現されない．

プランナーの通信負荷を増大させることなく，問題解決の一貫性を向上させることが，協調型推論の本質的課題である．そのためには大局的な目標の局所的な目標への分解およ

びこれを具体化するための戦略的な計画から戦術的な計画への詳細化について，これらを動的に展開するためのメカニズムの解明とアーキテクチュアへの反映が課題とされよう．

1.3.7 定性的推論

物理的法則や化学的法則にしたがって動的に変化する対象の大局的な理解を得るために，de Kleer，Forbus，Kuipersらによって定性的推論のモデルが提案され，またメンタルモデルとしての観点からも近年注目を集めている．原子力発電プラントのように大規模人工システムではシステムの挙動を知るには大規模なシュミレーションを実行することが必要であり，また実際にシュミレーションを行うためには，いくつかの仮定をおかなければならないという問題点を有する．ここに定性的推論の工学的ニーズが存在する．

定性的推論では，対象を記述するためのパラメータは定性的変数として記述され，変数間の関係は定性的微分方程式として記述される．この定性的微分方程式に基づいて，システムの局所的な動特性を与える手続きをルール形式で記述する．これらの知識を用いて，システムの大局的な動特性を推測することが定性的推論の目的である．

定性的推論が首尾よく成功するのは，局所的な相互作用がシステムの大局的な挙動に直接反映されるような場合である．いいかえると，時定数の大きい要素がシステムの大局的な挙動に対して支配的である場合，定性的推論では状態を一意に確定することができず，不確定性が，増大する．また時定数が異なる要素が共存する場合には，ある時点において過渡的状態にある要素数が一部であるにもかかわらず，あらゆる状態の組み合わせを考慮してしまうという冗長性を招く．

したがって，現在の定性的推論の枠組みのもとで，実システムをモデル化するためには，予め知られているシステムの大局的または局所的な定量的知識を使って，定性的推論の方向をガイドすることが不可欠となる．

定性的推論に基づく対象についての深層的な知識の利用と定性的な状態の組み合せ的爆発を抑えるための表層的な知識の利用の効果的な組み合わせによるバランスを図ることが，実用化における鍵であり，これらをシステム化するための技術を開発することが今後の課題と考えられる．

定性的推論が微分方程式で記述される連続系を対象としているのに対し，時制的推論は事象の生起によってシステムの状態が変化する離散系を対象としている．定性的推論では状態遷移だけが問題とされ，時間という概念は排除されているが，異なった時定数を含む

パラメータの取扱いには時間概念が不可欠であり，これらを時制的推論の枠組みで扱い，両者を融合する方法を検討すべきである．

1.3.8 類　推

同一の問題領域であっても，対象が少しでも異なると，その違いに応じて，専門家の知識を追加しなければ使えないという状況が生じる．これと同じことはシステム設計者が想定する使用範囲とユーザが実際に作業する使用範囲がズレた場合にも生じる．これは現在の知識システムが非常にもろい性格をもつことを意味する．知識システムを頑健なものとするには，類似の知識を少し違った局面においても利用可能とすることが必要である．それには類推の機能をシステムに付加する必要がある．

類推は人間が新しい知識を獲得して，これらを同化する際に重要な役割を果たしているとみられる．人間は多くのことを知れば知る程，新しい知識の獲得が容易になるのは，類推が働いているためと考えられる．また日常の問題解決においても類推はごく自然に使われているとみられる．

原口らは類推を帰納推論との関連で把握し，ホーン節集合の枠組みのもとで数学的な定式化を試み，いくつかの基礎的知見を得ている．さらにその成果に基づいて，演えき的推論と融合した推論システムを試作している．彼らの研究は，従来のインフォーマルな推論に関する研究と比べて，厳密な理論的基礎をもつ点で評価されるが，これを実用化レベルにまでもっていくためには，克服すべき技術的課題が山積している．

類比とは何か，類推とは何か，についての一般的な見解が確立されていない現在において，類比や類推の原点に立ち返って再考することも重要と思われる．ここでは類推システムが実用化されるための技術的課題について考察する．

２つの対象が与えられたとき，この間に適切な類比が成立するためには，各対象に関する知識が必要かつ十分に与えられていることが必要である．必要以上の知識が付加されていたならば，可能な類比の数はいたずらに増え，極大類比に限っても無意味なものが多くなる恐れがある．また十分な知識が与えられていなければ，適切な類比を見出すことが困難になる．ここで“必要”とか“十分”を規定する要因は何であろうか．それは問題解決の文脈や制御の焦点と考えられる．それをここでは視点と呼ぶことにする．与えられた視点のもとに，基準対象と目標対象の間での類比を最適化するために，必要な知識を追加し，必要な知識を排除することが必要になる．ここに演えき的推論との結合が不可欠になる．

目標対象が与えられたとき，基準対象を知識ベースの中からどのようにして切出してくるかという問題（Analogical Access）がある．これは対象がどのように表現されているかに依存するが，対象がすべてフレーム形式で表現されていることを前提とすれば，フレーム間での照合プロセスが要請される．フレーム間での類似性をどのように定義するかという問題が生じる．ここでは粗い照合により，基準対象の候補を選択するメカニズムが必要であり，問題解決の文脈に関する知識が使われよう．

基準対象における問題解決すなわち変換規則が目標対象に対しても適用できるためには，変換規則の条件部が類比においても保存されなければならない．もし満たされていなければ，変換規則の条件部を一般化または特殊化することが必要になる．そのためには，一般化および特殊化を行う戦略およびその妥当性を評価する基準が必要となる．

類推はより広義には探索問題と考えるべきである．類比がとられる対象はフレームのように構造化された知識表現が適切である．類推の第１段階は基準対象の候補をフレーム集合から見い出すことで，ここでは粗いマッチングで十分と思われる．第２段階は選択された基準対象と目標対象の間での詳細なマッチングが必要で，最適化のために構造化されていないフラットな知識が使われる．そのような知識は分類階層的な知識と知識間のセマンティクスを与える推論規則である．

類推システムの実用化という観点からは，ＭＣＣで展開されているLenatが率いるＣＹＣプロジェクトの動向をモニターしておく必要がある．このプロジェクトでは類比を非常に柔軟な枠組みで把握し，大規模知識ベースの構築を支援するために類推を積極的に活用することを意図している．

常識に基づく柔軟で，知的な推論を実現することが，これからの知識システムの大きな課題とされよう．類推のシステム化の時機が到来しているとみられる．

1.3.9 問題解決

図1.3-3 に問題解決のパラダイムとモデルを示す．問題解決の主たるパラダイムは生成検査・抽象化・分解統合・制約伝播の４つであり，この下に種々の問題解決モデルが展開される．階層的生成検査法は生成検査をベースとして，抽象化による階層性を導入したものである．トップダウン精密化は抽象化と分解統合を組み合わせて，トップダウン的に問題解決を行うものである．抱束最小化は分解された副問題間での制約伝播において上位レベルでの決定を先送りするものである．依存関係後戻りは仮説推論において制約を充足し

つつ，生成検査に基づく探索を進め，失敗した場合，依存関係に戻づいて，制約条件を元に復帰させつつ，後戻りするものである．

問題解決のモデルはＡＩにおいて古典的かつ重要な分野であるにもかかわらず，知識パワーに押されて，片隅に追いやられている感じがするが，計画や設計などシステム合成型問題では基本的に重要なことであることを再認識すべきである．

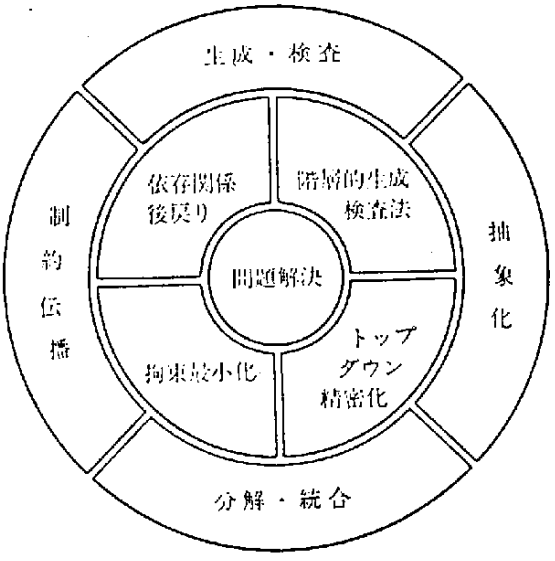


図1.3-3 問題解決のパラダイムとモデル

現在市場で入手可能なＡＩツールの多くは問題解決機能を提供していない．もっとも進んだツールでも，依存関係に基づく後戻りおよび制約伝播についてある種のモデルを提供しているに過ぎない．問題解決の原点に立ち返って，各種問題解決モデルを開発するとともに，ＡＩツールの中に組み込んでいくことが必要と思われる．

1.3.10 知識獲得

第１世代知識システムでは専門家からの知識獲得が強調されたあまり，知識獲得ボトルネックという問題を引き起こしたが，知的な推論制御を導入したとしても，知識獲得の問題は次世代知識システムにおいても重要な問題としてクローズアップされよう．

図1.3-4 は知識獲得のパラダイムとモデルを示す．知識獲得の主なパラダイムは仮説の生成と検証および知識の検索と管理である．帰納推論はデータから仮説を生成し，これを検証するもので，理論的にはもっとも基盤が整備している．類推は知識ベースとの類比を利用して仮説を生成するもので，理論的基礎が固まりつつある．知識同化および知識調節は知識ベースの完全性および無矛盾性を維持

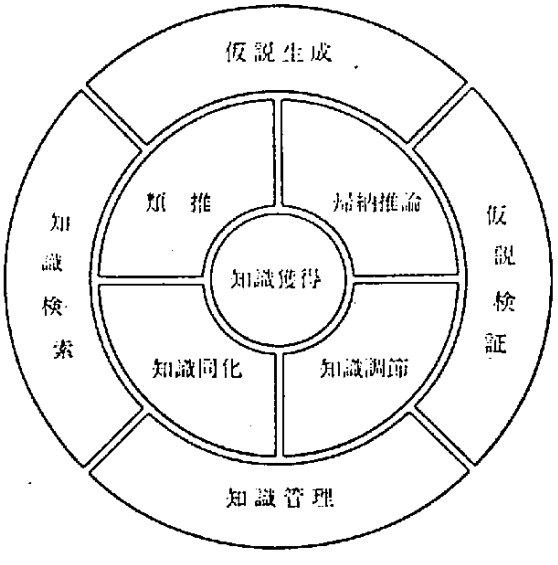


図1.3-4 知識獲得のパラダイムとモデル

するための管理技術で，より一般的な知識表現に対しても適用できるよう今後の技術開発が要請される．

1.4 まとめ

本章では，第１世代知識システムの限界となっている諸問題を考察し，次世代知識システムにおいて考慮すべき事柄をまとめた．ついで次世代知識システムの方法論の基礎を与える基盤技術について，特に高次推論に重点をおいて技術的な課題を考察した．

次世代知識システムでは，高次推論技術に支えられた問題解決機能の付与が中心的課題とされよう．そのためには，知識システムの対象となる問題のクラスを構造化し，要請される基本的タスクを抽出し，各種問題解決機能と併せて，類型化しておくことが必要である．

（小 林 重 信）

［参考文献］

［小林 86-1］ 小林重信：知識工学，昭晃堂，1986.

［小林 86-2］ 小林重信：知識システム方法論の確立に向けて，知識システム方法論シンポジウム，富　士通国際情報社会科学研究所，1986.

［ICOT 86］ ICOT応用システム第２専門委員会：昭和60年度新世代コンピュータに関する技術開発動向及び適用分野等の調査研究報告書－エキスパートシステムにおける知識獲得の諸相，新世代コンピュータ技術開発機構，1986.

［Kobayashi 86］ S.Kobayashi: The Present and Future in Expert Systems Technology, Control Theory and Advanced Technology, Vol. 2, No. 3, pp.329-344 1986.

［小林 87］ 小林重信：知識システム方法論，システムと制御，Vol. 31, No. 4, 1987.

2. 知識システム開発の手順

2．知識システム開発の手順

2.1 知識システムの一般的な開発手順

本節では，知識システム開発における一般的な手順および開発に際し留意すべき関連事項について論じる[小林 86]，[小林 87]．

2.1.1 システム開発の手順

知識システム開発の一般的な手順を図2.1-1 に示す．

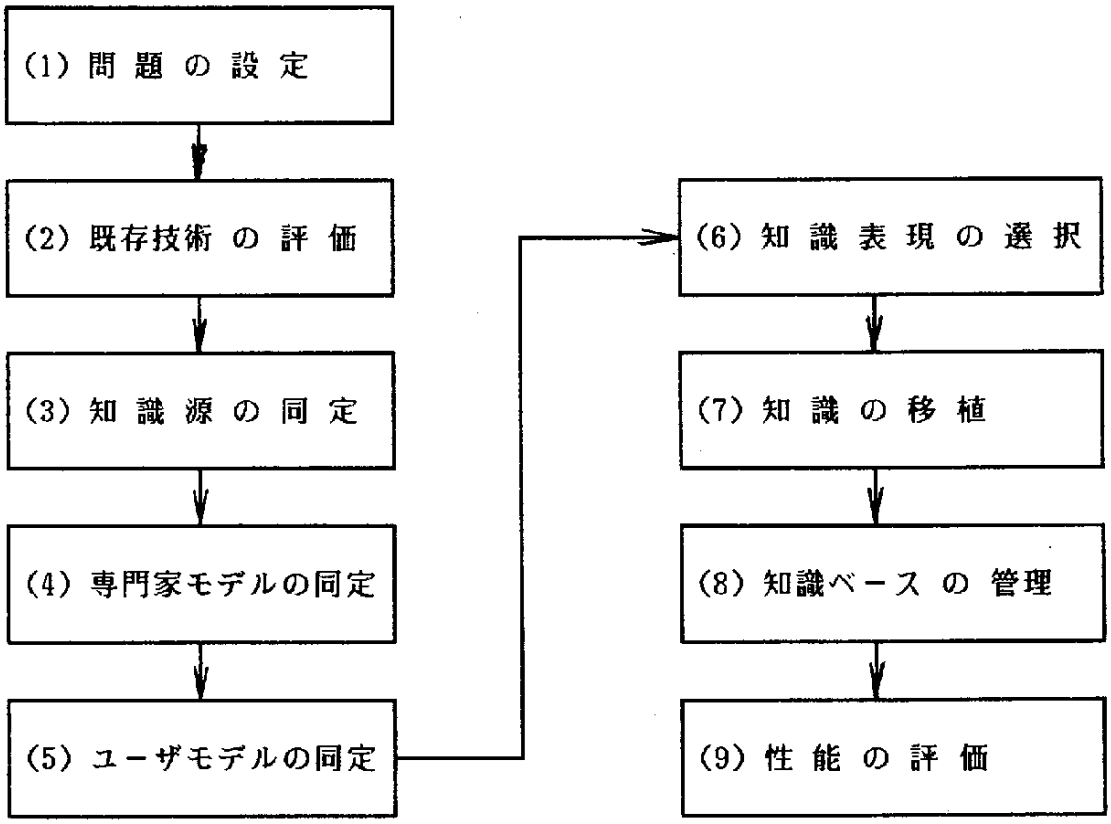


図2.1-1 知識システムの開発手順

以下に，開発手順の各段階について説明を加える．

(1)問題の設定

知識システムによる問題解決の対象として問題をどのように選択し，切出すかはシステム開発の成否にとってクリチカルである．問題の選択は知識システム必要条件を満たしていなければならない．このことについては，2.1.3 節で詳しく論じる．

(2)既存技術の評価

知識工学の技術は，従来のシステム技術およびソフトウエア技術と競合する場合もあれば，互いに補完的な役割を果たす場合もある．問題のシステム分析および技術間のトレードオフ分析に基づいて知識システムの導入の必要性を明らかにする必要がある．このことについては，2.1.4 節で詳しく論じる．

(3)知識源の同定

問題解決に必要とされる知識の所在を明らかにする．特に，エンジニアリングシステムの場合，知識源は専門家やオペレータなどの人間および設計仕様書や操作記録などきわめて多様である．ここでは，各知識源に存在する知識の形態，信頼性および利用可能性を分析，評価する．このことについては，2.1.5 節で詳しく論じる．

(4)専門家モデルの同定

主要な知識源が人間の専門家の場合，そのモデルを同定しておく必要がある．すなわち専門家がもっている知識の守備範囲，知識の質および量の分析，基本的な問題解決戦略の抽出および推論制御の方法，システムの評価基準などを明らかにする．

(5)ユーザモデルの同定

対話型システムの場合，ユーザのモデルを同定しておく必要がある．すなわちユーザがシステムを利用する範囲，ユーザの専門的な知識レベル（専門家，半専門家，素人），推論制御の主導権（システム主導，ユーザ主導，または相互主導）の決定，ユーザインタフェース，ユーザによるシステムの保守および拡張可能性などを明らかにする．

(6)知識表現の選択

(3)～(5)の分析結果に基づき，対象に関する知識や専門家の知識を表現するために，適切な知識表現形式の選択または知識表現形式の組み合せを行う．同時に，問題解決戦略の選択および推論制御方式の選択を行う．

(7)知識の移植

(6)で選択された知識表現の枠組のもとで，各種知識源に存在する知識を実際に抽出し，

これを利用可能な知識表現形式に変換して，知識ベースに移植する．専門家からの知識獲得は一般にはインタビューやプロトコル解析によって行われる．ある種の解析型問題（分類や診断）では知識獲得支援システムの利用も可能である．

(8)知識ベースの管理

知識ベースが対象とする問題領域をすべてカバーしているかどうか，知識間に整合性（矛盾や冗長性の除去）が確保されているかどうか，知識間の整合性がとれていない場合それが推論によってカバーできるか性質のものかどうか，など知識ベースの管理を行う．

(9)性能の評価

テストデータが利用可能な場合，知識システムが妥当な挙動を示すかどうか，実際に知識システムを動かすことによって評価を行う．性能の評価には，領域専門家およびエンドユーザも関与する．領域専門家は知識システムが妥当な挙動を示すかどうかに関心を示す．エンドユーザは応答性や対話性に関心を示す．

以上の(1)～(9)の各段階は逐次的に行われるものの，実際の作業はフィードバックループを構成する．すなわち，ある段階における決定はつぎの段階の決定を制約するものの，上の段階での最適な決定は下の段階における決定に依存するために，必然的に後戻りを必要とする．

システム工学の用語に従えば，(1)～(6)の段階はシステム分析，(7)～(9)の段階はモデリングと呼ばれる．知識システムが対象とする悪構造・悪定義問題では，モデリングよりもシステム分析が非常に重要なウェートを占めることを指摘しておく．

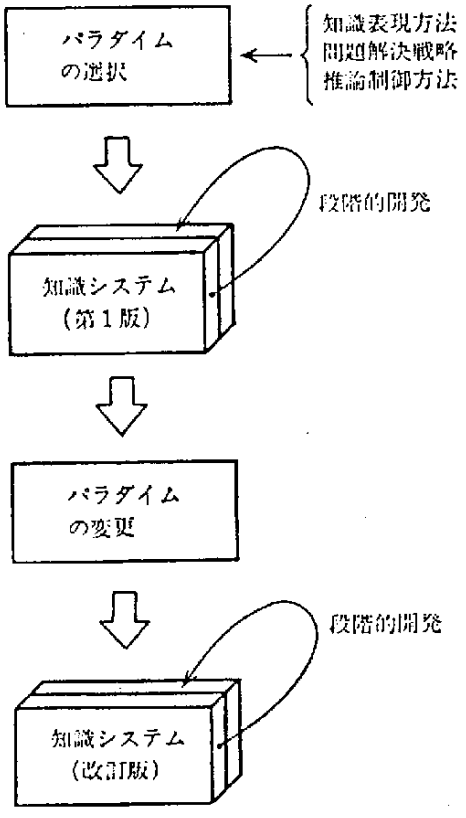


図2.1-2 段階的システム開発過程

2.1.2 ラピッドプロトタイピング

前節で指摘したように，知識システムの開発過程は一般にフィードバックループを形成する．この様子を図2.1-2 に示す．従来のソフトウエアづくりでは初めに厳密な要求仕様書が与えられることを前提としているので，ウォータフォール型システム開発手順が取られているのに対し，知識システムでは事前に明確な要求仕様が与えられるのは稀で，システ

ムの開発を通じてシステムの仕様が次第に明確になるという側面があり，知識システムの開発過程は総じて段階的，発展的であることを特徴とする．すなわち，最初に対象とする問題に対しもっとも適切と思われるパラダイムが仮に選択される．すなわちシステム分析の結果に基づいて，知識表現形式，問題解決戦略，推論制御方式などが仮決定される．

このパラダイムのもとに知識システムの第1版が構築される．これは対象とする問題領域の典型的と思われる問題解決過程をシミュレートするようなデモンストレーション的なシステムである．第1版の開発はできるだけ早期に，数か月以内であることが望ましい．第1版を徹底的に分析することにより，専門家の問題解決過程やエンドユーザの利用形態が明らかになる．この分析結果を反映させて，パラダイムの変更がなされ，改訂版のシステム開発へと移行する．このようなプロセスは数回繰り返されるのが普通である．

このため，知識システムの開発にはラピッドプロトタイピングが可能なプログラミング環境が適しているといえる．ただし，ラピッドプロトタイピングを極端に推し進めると，場当たり的なシステムづくりに走る危険もあることを承知しておかねばならない．

2.1.3 知識システム開発の条件

知識システムを開発するうえで，つぎのような必要条件を考慮すべきである．

1)記号処理を中心とした問題であること：

対象とする問題が記号処理を基本とするタスクであるかどうかを見極めることが必要である．数値処理が基本とするタスクであれば，従来のシステム技術で十分対処できるはずである．イメージのように記号化されていない処理が基本となるタスクも適切な対象とはいえない．人間の高度の感覚器官によって把握されるような感覚データの取扱については未だ技術的に成熟していないからである．いいかえれば，人間の専門家が問題解決の過程を言葉すなわち記号によって表現することが可能な問題領域がシステム化しやすい問題ともいえる．

2)専門家の経験的知識または判断が利用可能な問題であること：

問題の解決に対して，従来の数理工学的あるいはシステム技術的な接近が困難であり，しかも経験的な解法が有用であることが実証されている問題領域を選択すべきである．専門家が経験によってその有用性を確認したヒューリスチックスを有し，それを形式化することが可能でなければならない．専門家の知識が利用可能ではない問題につては，システム分析者自身が問題解決に必要な知識を獲得できるような環境になければならない．たと

えば，シミュレータの利用による知識獲得などはその例である．

3)十分なニーズがある問題であること：

退職などにより専門家の知識や操業のノウハウが失われていくような状況は，システム開発の十分な根拠を与える．専門性が高くかつその分野の専門家が少ないことは開発の動機を与える．逆に，専門家が大勢いるとか，専門性の水準が低く，だれでもすぐその分野の専門家になれるような問題は適していない．

多くの場所あるいは非人間的な環境で専門知識を必要とされる問題はシステム開発の動機を与える．海底や地底などの極限環境下で作業とするロボットに知的な機能を付加することなどはこの例である．

4)明確な価値を生じる問題であること：

知識システムの開発には多くの期間および人手を必要とするために，明確な目標をもつものでないと，中途半端なシステム開発に終る可能性が高い．デモンストレーションまではうまくいきそうだが，実用化するには数多くの困難が予想される問題は最初から取上げるべきではない．システム化が高い価値を生み出すかどうかを評価するために，費用／便益分析または費用／効果分析を行うべきである．100%完全でないと意味がないシステムと90%しかできてなくとも十分意味のあるシステムでは，自ずとシステム開発への取り組み方が異なるはずである．知識システムの導入により，何人分の省力化が可能であるといった指標は明確である．

5)保守および拡張が容易な問題であること：

システム開発に大量の人間を投入して，大規模な実用システムを構築しても，それを維持するのに苦労するようなシステムであってはならない．エンドユーザ自身が保守や拡張を行える知識システムが一番望ましい．そのことを前提として，知識システムの構成を考慮する必要がある．そのためには，知識ベース，推論エンジン，ワーキングメモリなどの知識システムの基本構成要素だけではなく，知識獲得支援システム，説明機能，ユーザインタフェースなどの付加的構成要素に十分配慮する必要がある．

2.1.4 従来技術との関連

知識工学の技術は従来のシステム技術と比べて，ある面では競合的であり，また別の面では相補的である．システム技術はこれまで主として数値的なパラメータで記述される問題を扱ってきたが，悪構造・悪定義問題に対する接近という意味では知識工学と同じ観点

にたってきたといえる．システム工学方法論の方が現在の知識工学よりもはるかに概念も豊富であり，技術的蓄積も豊富である．従来のシステム技術で十分達成できる問題に対して知識工学的接近を試みることは無意味である．両者のトレードオフを考慮することが要請されよう．

例として，VLSIの実装設計システムを考えよう．この分野では，知識システムが導入される以前は整数計画を初めとする数理計画的手法に全面的に依存して，CADシステムが構築され，数理的手法では対処できない未結線部分のみ専門家が介入していた．しかしVLSIの規模が大きくるにつれて，専門家の負荷が大きくなり，このことが知識システムの導入につながっている．既存システムは従来のシステム技術をベースにして，膨大なソフトウエアの蓄積があり，これを知識システムで置き換えることは事実上不可能である．大規模化に対処するために，既存システムを拡張することよりも，この上に知識システムを構築する方が効率的と判断される．

このように，従来のシステム技術で実現されているシステムの上に知識システムを被せて利用するシステムが現在実用化の最短距離にあるといえる．システム技術のもとに蓄積されているソフトウエアは膨大なものであり，これらをユーザの新しいニーズに対応させるうえで，従来技術と知識システムとの接続は有用である．

2.1.5 知識源の同定

知識源は問題領域の専門家には限らない．人工システムの場合，設計仕様書，操作手引書，保守記録書などは有力な知識源である．図2.1-3 は，知識源と知識の関係を示す．対象に関する深いモデルは領域専門家や設計仕様書から得られる．経験則のような知識は領域専門家や保守記録書または操作記録書などから得られる．システムの因果関係に関するモデルは操作手引書や保守記録書などの分析結果から得られる．運用規則は設計仕様書や操作手引書から得られる．

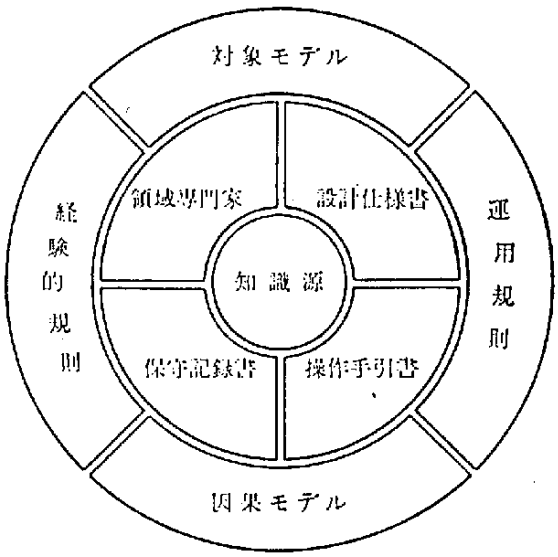


図2.1-3 知識源と知識の関係

各知識源に散在する知識はそれぞれ形態が異なるし，またその適用範囲も異なるが，重

複する部分も少なくないはずであり，これらの知識を統合化することはシステム分析者の重要な役割である．

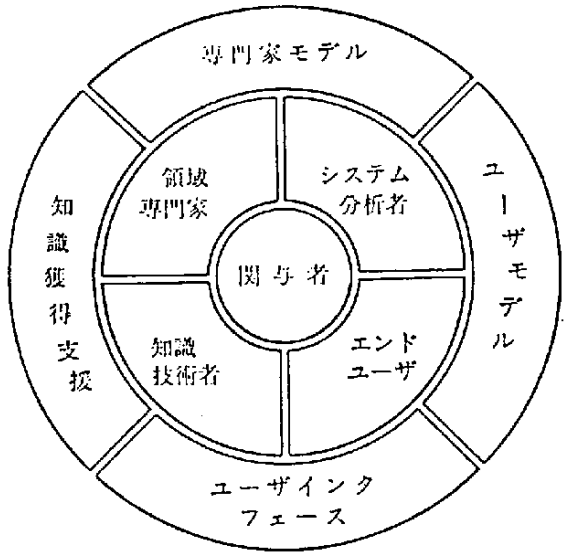


図2.1-4 システム関与者の役割

2.1.6 システム関与者の役割

知識システムの構築と利用には，領域専門家システム分析者，知識技術者，エンドユーザが関与する．領域専門家は知識源としての知識の提供者であり，そして知識システムの性能の評価者でもある．エンドユーザは知識システムの利用者であり，また知識システム設計の際の要求仕様の提案者でもある．知識システムの構築と維持にはシステム分析者と知識技術者が関与する．図2.1-4 はシステム関与者の役割を示す．図2.1-5 はシステム関与者と知識システムとの関係を示す．

システム分析者は領域専門家との対話を通じて，専門家モデルを同定することが要求される．同様に，システム分析者はエンドユーザとの対話を通じて，ユーザモデルを同定することが要求される．

一方，知識技術者は領域専門家に対し，知識の獲得を支援する．また知識技術者はエンドユーザのためにユーザインタフェースを開発する．

システム分析者および知識技術者はそれぞれシステム工学方法論および知識プログラミング方法論に精通していることが望ましい．問題領域の専門家は知識の獲得に貢献するだ

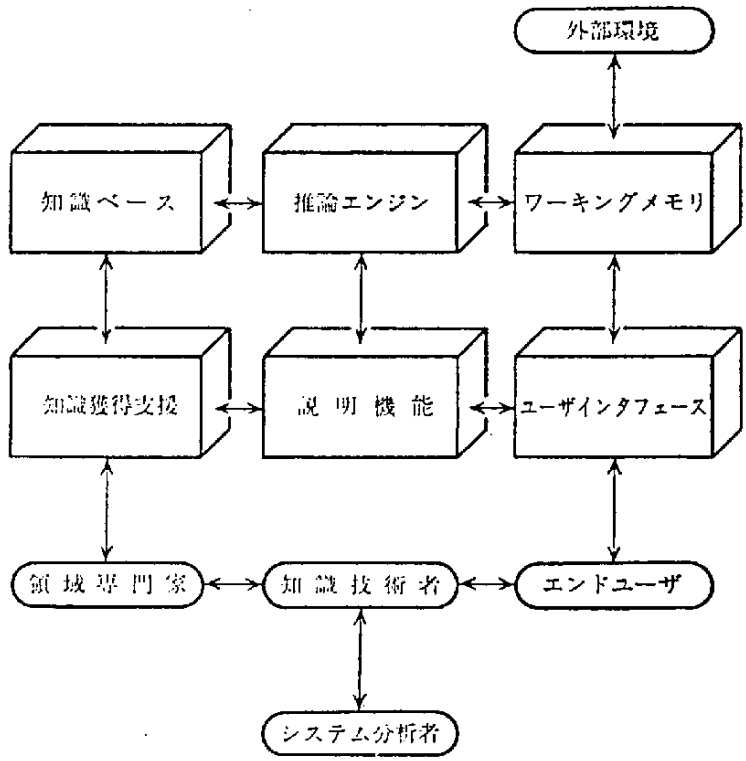


図2.1-5 システム関与者と知識システムの関係

けでなく，知識システムの性能評価にも重要な役割を果たす．

ユーザモデルの考え方はこれから重要になると思われる．第1に，コンサルテーション型システムでは認知科学的側面からユーザの挙動を分析して，システム設計に反映させる必要がある．特に，ユーザが主導権をもって推論を進めるシステムにおいて，システムが知的にこれをサポートするような形態が望まれる．第2に，エンドユーザ自身の手でシステムの拡張が容易に行える手離れの良いシステムが望まれる．そのためには，インタフェースをより人間側に近づける努力が必要である．

2.1.7 システム構築ツールの役割

知識システムを実際に構築するには，言語やツールを必要とする．前に議論したように，通常のプログラム開発とは異なり，知識システムの開発は試行錯誤を繰り返す進化論的なプロセスとして特徴づけられるため，言語やツールはこのようなプロセスを効率よくサポートするものでなければならない．

知識システムを実現するには，与えられた計算機環境のもとで，システム構築ツールを整備することが不可欠である．

知識システムを構築するためのツールがもつべき条件としては，

1)機能性：知識の表現，推論の制御および問題解決におけるさまざまなパラダイムが利用可能であること，

2)性能性：実行時の速度や応答性，実時間処理，，利用可能な資源の許容量などが要求される水準を十分満たしていること，

3)対話性：プログラミングおよびデバッギングを支援するための優れた対話的環境が提供されていること，

4)接続性：他の言語やデータベースへの接続，センサや計測器からのデータの取込み，他のマシンへの接続が可能であること，

5)領域適合性：データ解釈，診断，制御，計画，設計など応用領域の問題解決に必要な機能を陽に提供していること，

が挙げられる．

システム構築ツールが市場に出回るようになってまだ5年程度しか経過しておらず，ツールの機能や性能はまだ成熟段階に到達していない．現在のシステム構築ツールは，知識表現が主体であり，問題解決という観点からすれば，まだ極めて低水準に留っており，マ

シンに実装するための手段ではあっても，モデリングを支援する手段とはいえない．したがってシステム構築ツールは知識システム開発の後半において利用すべきものであり，最初にシステム構築ツールありきという考えをもってはならない．

2.1.8 インタフェース

図2.1-6 は入出力インタフェースとユーザモデルおよび環境モデルとの関係を示す．
現在の知識システムは記号データの入力を前提としているが，これからは感覚データおよび実システムに組み込まれた測定器から実時間で送られてくる計測データの取り扱いが重要視されるであろう．

出力については，コンサルテーション型のシステムでは，ユーザモデルに親和性の高いインタフェースが要請される．組み込み型のシステムでは，制御指令が実システムに受理できるように既存のシステムとの接続が要請される．

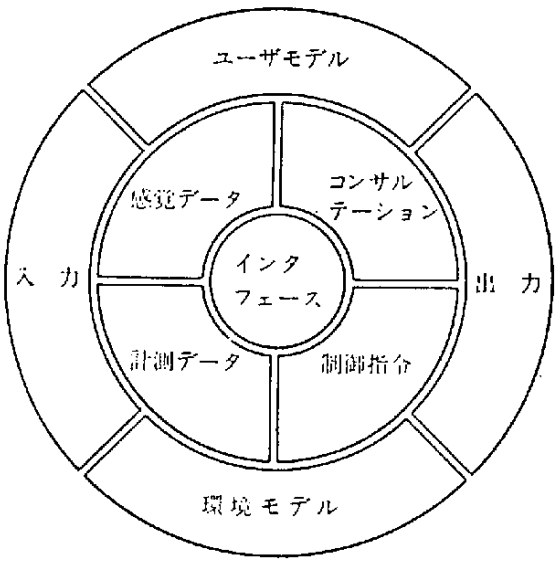


図2.1-6 は入出力インタフェース

2.1.9 まとめ

これまで，知識システムの開発手順およびこれに関係する事項を論じてきた．最後に，知識システムの構築における諸問題を今後の課題としてまとめておく．

1)ライフサイクルの確立の必要性：

これまでに開発されてきた知識システムの構築過程は余りにも経験的過ぎる．システム構築ツールによるラピッドプロトタイピングが強調され過ぎて，場当たり的，試行錯誤的システムづくりが多かったといえよう．この方法では，知識技術者の勘と経験が大きくものをいい，方法論として蓄積されていかないことになる．知識システムの開発過程については，ソフトウエア工学から学ぶことが多いはずである．ソフトウエア工学的観点から知識システム開発のライフサイクルを確立すべきである．

2)システム分析方法論の有用性：

知識獲得以前の問題として，問題領域のシステム分析が極めて重要である．にもかかわ

らず，現実には，システム分析の方法論はほとんど使われていない．システム分析については，現在の知識工学は何もサポートしていないことを承知すべきである．システム工学は大規模かつ複雑な対象を分析するうえ有用な方法論を提供している．システム工学方法論に精通した知識技術者を育てる必要がある．

3)専門家への依存からの脱却：

エキスパートシステムという名のもとに，専門家からの知識獲得が強調され過ぎた傾向がある．確かに，問題よっては専門家の知識に全面的に頼らなければ何もできない問題も少なくない．しかし，エンジニアリングシステムに限っていえば，専門家以外の知識源から利用可能な知識が多数存在する．対象から得られる知識と専門家から得られる知識を明確に区別し，前者の知識の高次利用を考え，後者の知識への依存を極小化すべきである．対象知識を高次利用するためには，柔軟な知識表現や対象システムに関する深い知識の利用が要請される．

4)高次推論による知識獲得の緩和：

知識獲得ボトルネックという問題を緩和するためには，あいまいで，不完全で，相矛盾するような知識の存在を前提として、柔軟な推論によってこれらを補完すべきである．第１章で議論した不確実性推論、デフォルト推論、仮説的推論、分散協調型推論，定性的推論，類推など高次推論技術を積極的に活用していくことが重要である．

現在の知識システムはシステム構築ツールの影響を受け過ぎている．“初めにＡＩツールありき”ではなく、“初めに問題のシステム分析ありき”という考え方が大事である．人工知能、システム科学、認知科学という広い枠組のもとで対象とする問題を分析し、適切なモデル化の枠組を設定することが基本的に重要なことである．

（小 林 重 信）

［参考文献］

［小林 86］小林重信：知識工学，昭晃堂，1986．

［小林 87］小林重信：知識システム方法論，システムと制御，Vol．31，No．4，1987．

２．２．１　富士通におけるシステム開発手順

富士通では既にＡＩ用言語・ツール・アプリケーション・開発方法論等を統合した，「ＫＳＡ知識情報システム」を確立しており，エキスパートシステムの開発手順はこの知識情報システムの一環として位置付けられる．

２．２．１．１　ＫＳＡ知識情報システム

ＫＳＡ（Knowledge Systems Architecture and Applications）知識情報システムは，既存システムとの融合を前提とした「ＫＳＡハードウェア・ソフトウェア製品」，これの利用を支援する「ＫＳＡサービス」，およびＫＳＡ製品をより高度化するための「ＫＳＡ研究開発」から構成される．ＫＳＡの製品・サービス体系を以下に示す．

［ＫＳＡ製品・サービス］

①知識システム構築支援製品

ハードウェア：　ホストマシン，汎用ワークステーション，パソコン

ソフトウェア：　言語（LISP,GCLISP,PROLOG），構造エディタ（EMACS/X），
汎用エキスパートシステム構築ツール（ESHELL/X,ESHELL/PM），
ドメイン向けエキスパートシステム構築ツール（ESHELL/SB）

②知識システム製品

計算機運用支援，プログラム開発支援，エンドユーザ支援，エンジニアリング等

③自然言語処理製品

英日自動翻訳システム，日英自動翻訳システム

④ＫＳＡサービス

エキスパートシステム構築技法（ES/SDEM）

ＫＳＡテクニカルセンタ

各種教育コース，自習用教材

２．２．１．２　エキスパートシステム構築技法ＥＳ／ＳＤＥＭ（Software Development and Engineering Methodology for Expert Systems）

(1) 概要

エキスパートシステム構築ツールであるＥＳＨＥＬＬファミリの応用システムは，社内外で質量ともに確実に発展している．ＥＳ／ＳＤＥＭは，ＥＳＨＥＬＬの豊富な応用事例を基盤として，それらの構築経験から共通な性質を抽出・一般化し，エキスパートシステム構築のノウハウを方法論として集大成したものである．ＥＳ／ＳＤＥＭの全体は，ユーザおよびＳＥ向けに教育コースとして提供されており，本報告書では基本部分について述べる．

一般に，エキスパートシステムを初めとする「解法が不明確な問題」をシステム化する場合，確定した設計仕様を前提とする従来のシステム開発アプローチは適用が難しい．ＥＳ／ＳＤＥＭでは，試作を通して徐々に解法を明確化していくアプローチ（プロトタイピング）を基本とする構築手順と，知識獲得のポイントやプロトタイプ育成のノウハウ等を提示するものである．

(2) 特長

① 専門家モデルの導入

エキスパートシステムの一般的な枠組みを設定し，作業の見通しを良くする．

② 育成型開発形態の採用

大まかなモデル設計でのシステムから徐々に成長させていく手法によって，不明瞭な問題の明確化を計る．

③ ３段階の開発ステージの設定

プロトタイプシステムの目的に応じて開発ステージを設定することにより，作業内容を明確にする．

④ 各ステージにおける構築手順の明示

⑤ 各フェーズにおける技術上のノウハウの提示

⑥ 各種ワークシートによるドキュメントの体系化

⑦ プロジェクト推進の指針／留意事項の提示

⑧ 特定のツールに依存しない方法論

ＥＳ／ＳＤＥＭはツールのプログラミングテクニックを述べたものではなく，人間の思考プロセスのモデル化を目的とする．

２．２．１．３　ＥＳ／ＳＤＥＭのコンセプト

(1)　基本思想の確立

エキスパートシステムとは専門家の問題解決をシミュレートするシステムであり，その構築は問題解決システムの作成である．したがって専門家の思考プロセスを把握することがエキスパートシステム構築のキーポイントとなる．ＥＳ／ＳＤＥＭの基本思想は，思考プロセスのモデル化を行なうための方法論という点にある．

(2)　構築手法の明確化

①　ラピッドプロトタイピングの採用と，プロトタイプの性格付け

ラピッドプロトタイピングとは，その時点で理解した範囲の知識に基づいてシステムの一部を迅速に作成し，追加／修正を繰返して全体システムを構築する手法を意味する．

②　専門家モデルの導入と実現手段の提示

専門家モデルとは，専門家を問題解決システムと見なしたときの，システムの行動様式と問題の構造とをモデル化したものである．

③　ＷＨＡＴ　→　ＨＯＷ　→　ＷＨＹ　に基づく段階的知識獲得による，

概念化　→　構造化　→　詳細化　の実現

段階的知識獲得とは，プロトタイプ作成過程における，概念化・構造化・詳細化の各フェーズ毎の目的に応じた知識獲得方法である．

ラピッドプロトタイピングにより作成するものが専門家モデルであり，ＥＳ／ＳＤＥＭではデモシステム，検証システム，実証システムの３段階の性格付けにより専門家モデルを段階的に成長させる方法をとる．

エキスパートシステムの構築においては，専門家から効率良く，かつシステマティックに知識（問題解法）を抽出しインプリメントすることが重要であるが，このとき次の２点がボトルネックとなる．

①　知識を体系的に説明できない（専門家側）

②　抽出された知識を整理・統合できない（開発側）

これらの原因は，知識獲得において人間の記憶構造の特殊性（サーチに強いがダンプに

弱く，連想によって記憶が想起される）を考慮していないことと，無方針に知識獲得を行うことにある．

ラピッドプロトタイピングと専門家モデルの設定とは，これらの隘路を広げるための手法である．すなわち，専門家モデルという枠組みを設定し，現在のモデルの状況に応じて収集すべき知識の範囲・性質（不足部分，修正部分等）を決定し，モデル（＝システム）の実行結果を新たな知識獲得のキーとして専門家に提示する．これらの関連性を図２．２－１に要約して示す．

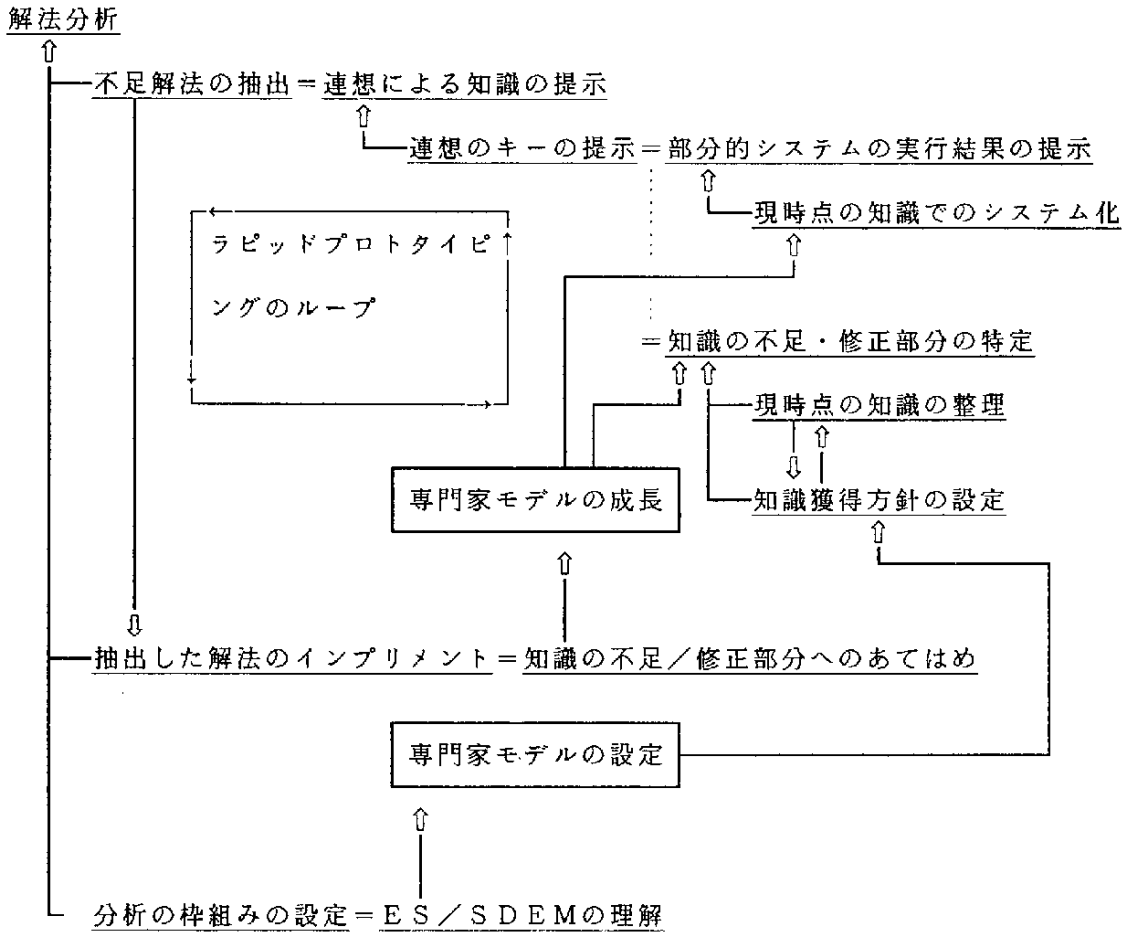


図２．２－１　専門家モデルの設定と成長

ラピッドプロトタイピングは，解法の抽出と表現とに渡るループである．解法抽出は，連想による知識抽出によって行われる．そのためには部分的システムの実行結果や，知識の不足／修正部分を提示しなければならない．専門家モデルの設定・成長はこれらを行うために役立つ．また，抽出した解法を，専門家モデルの知識不足／修正部分へインプリメントすることにより，モデルは成長する．これが新たな解法抽出のためのキーを生み出すことになる．

2．2．1．4　構築手順一覧

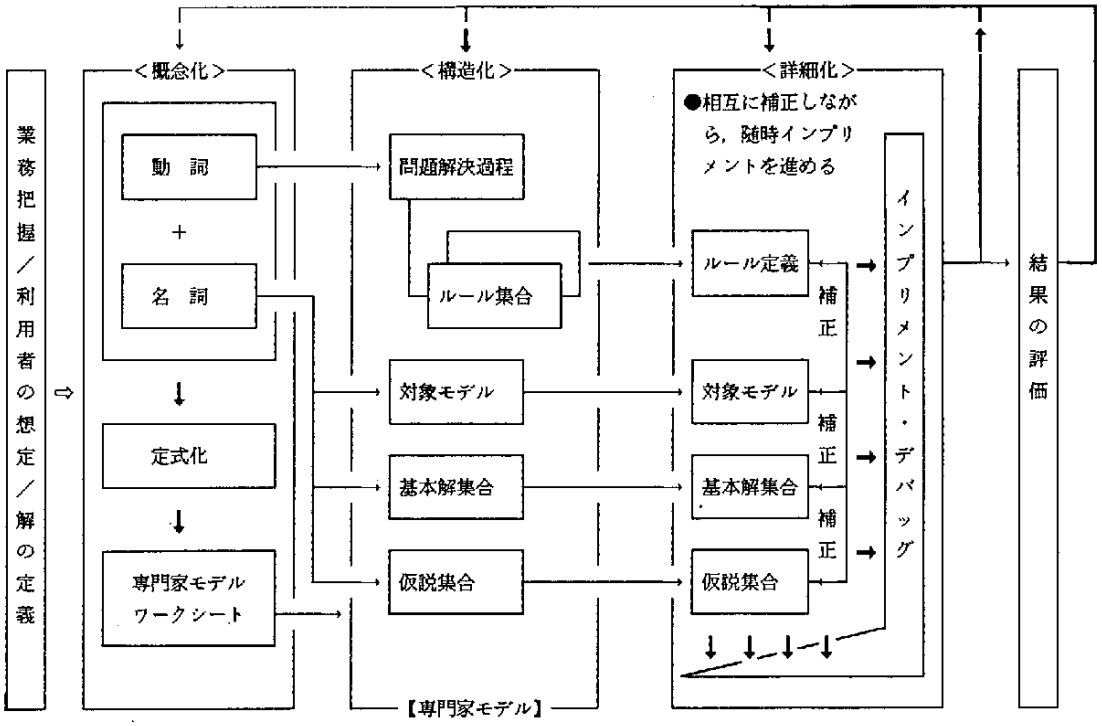
実証用	詳　細　化		検　証
主作業	●問題解決能力の向上		●実現可能性

検証用	（構　造　化）	詳　細　化	検　証
主作業	●問題範囲の拡張に伴う構造の修正	●問題解決能力の向上 ●MMIの強化	●プロジェクト継続の可否

↑ステージ　→フェーズ

確認用	概　念　化	構　造　化	詳　細　化	検　証
主作業項目	●業務内容の詳細把握 ・インタビュ／文献研究 ・現場観察／ケーススタディ ・動詞，名詞の抽出・分類 ・専門家モデルの設定	●動詞概念・名詞概念の構造化 ・問題解決過程の構造化 ・対象モデルの構造化 ・基本解集合の設定 ・仮説集合の設定	●知識の詳細化 ・ルール抽出 ・知識の整理 ・知識の一般化	●専門家モデルの妥当性
知識獲得のポイント	【WHY，WHAT中心の知識獲得】 ［文献・技術資料］［専門家］ ↓　↓ 動詞概念・名詞概念の抽出 ↓ 問題の定式化	【HOW中心の知識獲得】 ［動詞］　［名詞］ ↓　↓ 部分問題化　対象モデル ↓ ルール集合	【HOW中心の知識獲得】 ［専門家］ ↓ ルール収集 ↓ 整理・一般化	【WHY中心の知識獲得】 ［専門家］ ↓ 評　価

各モデル確定の推移



関連ドキュメント	・インタビュメモ　・専門家モデルWS　・各種構造設計WS　・ルール抽出WS　・ソース　・評価報告書 ・観察記録　・用語集　・アンケート用紙　・関連図 ・ケーススタディ記録　※WS…ワークシート　・関数仕様定義WS　・管理情報

Copyright © 富士通㈱，1987.

ＥＳ／ＳＤＥＭでは，プロトタイプに３種類の性格（確認システム，検証システム，実証システム）を与え，これをステージと呼ぶ．これは，従来のプロトタイピングが何の指針も持たずに行なわれ，単にシステムの実行結果をフィードバックするという枠組みしか設定できなかった欠点を解消するためである．

プロトタイプは上記のステージの順に成長させる．必ずしも作り直しを意味しない．

１種類のプロトタイプの作成は，概念化・構造化・詳細化・検証を通じて行なわれる．これをフェーズと呼ぶ．ただし各ステージにおいて全てのフェーズが必要ではなく，各ステージの性格に応じて必要なフェーズのみを行なう．逆に言えば，各ステージの性格を明確にし，各ステージにおいて中心になるフェーズを設定することが，方法論の目的となるのである．

各ステージ，各フェーズには固有の知識獲得のポイントがある．これは，エキスパートシステム開発過程において，現在のステージ，フェーズに最も必要な知識を重点的・系統的に収集するためである．

プロトタイプの開発・成長過程を以下に示す．開発過程が螺旋を形成するので，ラピッドプロトタイピングはスパイラル型開発形態とも呼ばれる（図２．２－２）．

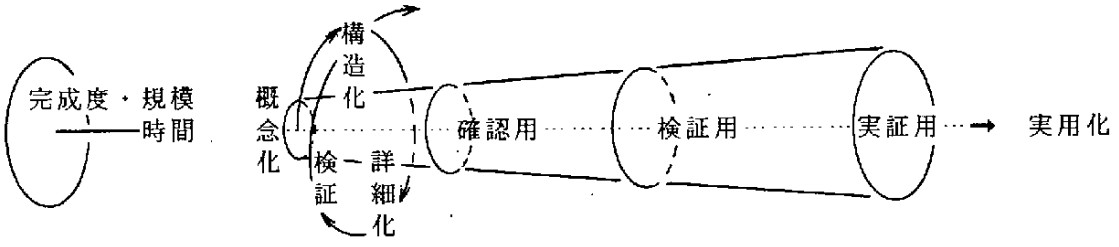


図２．２－２　スパイラル型開発形態

2．2．1．5　専門家モデルの構成

専門家モデルとは，専門家が行なう問題解決の環境を表現したモデルである．これは，問題そのものと，問題に対する行動とに大別される．

問題解決の環境 ―― 専門家が扱う問題範囲と対象（実体）
　　　　　　　 ―― 対象への働きかけの方法（行動様式）

一般に，専門家による問題解決は以下のプロセスをとる．

①対象物（例えば人体の疾患モデルや設計対象）の状況をある判定基準によって把握する．

②判定結果に基づいて仮説（例えば病名・着目部位や対象物の修正・操作方法）を立てる．

③仮説を判定基準によって検証し，解（仮説や判定基準のひとつ，あるいは対象象物の状況）に至るまで①から繰返す．

以上の過程は，ある戦略によって制御される．これらは図2．2－3で表現される

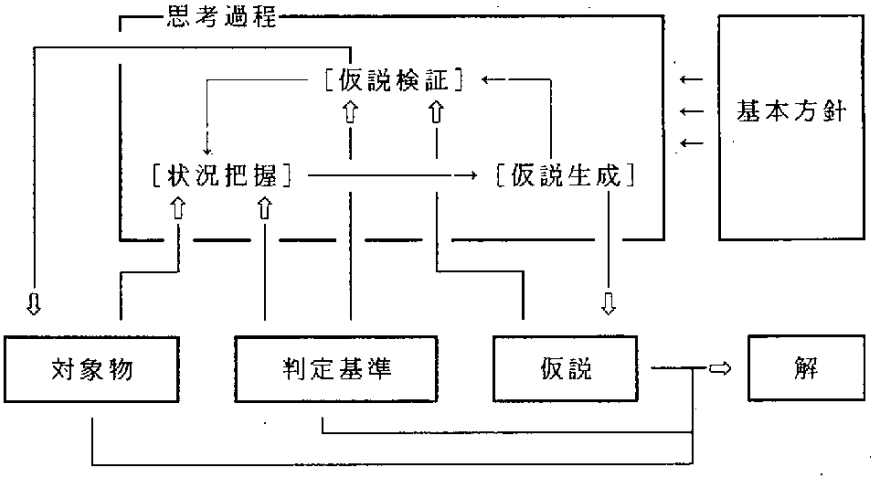


図2．2－3　専門家モデル

このような，専門家の行動様式と実体とを表現したモデルを専門家モデルと呼ぶ．ラピッドプロトタイピングによるＥＳ構築は，専門家モデルの作成／成長を行うことに他ならない．

ＥＳ／ＳＤＥＭでは次の用語を採用している．

思考過程 …→ 問題解決過程

対象物 ……→ 対象モデル

判定基準 …→ 基本解集合

仮説 ……→ 仮説集合

基本方針 …→ 戦略

２．２．１．６ 専門家モデルの構成要素と定式化

(1) 専門家モデルの構成要素

各構成要素は次のように定義される．

① 対象モデル …… 専門家が考察する問題分野・実体を表現し，判断のための基礎情報を提供する，インテリジェンスを持ったデータ構造

対象モデルの例：

・医療診断における「患者の症状」

・操業計画における「操業対象プラントの状態」

・個人の財産相談における「家族構成や，各人に関する将来の出来事」

② 基本解集合 …… 予め用意される解候補の集合（計画型や設計型には存在しない場合もある）

基本解集合の例：

・機器故障診断における「予め想定される故障箇所」の集合
（対象モデルと重なる場合も多い）

・計算機構成設計支援における「初期条件で定まる典型的構成例」の集合

・個人の財産相談における「金融商品データベース」

③ 仮説集合 …… 実行過程で動的に生成される中間結果あるいは解候補の集合

仮説集合の例：

・機器故障診断における「疑わしい部分」の集合

・スケジューリング問題における「計画案」の集合

・個人の財産相談における「有利と思われる金融商品の組合せ」の集合

④ 問題解決方式 … 問題解決の大きな流れ，基本的な動作，専門家の戦略など

問題解決過程＋専門家の戦略

ＩＦ～ＴＨＥＮ… ルールの集合あるいは手続きで実現

⑵ 定式化

エキスパートシステムを，上記の専門家モデルの構成要素から解への写像ととらえ，これを「定式化」と呼ぶ．

定式化： ＥＳ：｛対象モデル，基本解集合，仮説集合，戦略，問題解決過程｝

→｛解｝

定式化の例を以下に示す．

① 医療診断ＥＳ：

｛患者の症状，病名集合，着目部位，診断戦略，種々の診断規則｝

→｛その患者の病名｝

※医師は，

・患者の症状を把握し，

・疑わしい部位や，病名を仮定しながら，

・診断戦略（問診から計測器による診察に移る　など）に沿って，

・様々な診断規則を駆使して診察を行ない，

・予め用意された病名のいずれかを患者の病名として特定する．

② プラント操業計画ＥＳ：

｛プラントの状態，（　），計画案集合，立案戦略，

（制約条件　計画案作成　計画案評価）｝

→｛ある月の操業計画｝

※スケジューラは，

・プラントの状況を把握し，

・立案戦略（重要度の高い部分から割り当てて行く　など）に沿って，

・制約条件にかからないように計画案を作成し，その妥当性を確認しながら，

・ある月の操業計画を作成する．

2．2．1．7　各ステージにおけるプロトタイプの性格付け

(1)　確認用：

①　問題領域の特定

現在のモデル設定で，ある問題が解決可能であることを確認する．

②　全体システムのイメージ固め

扱える問題の範囲は狭くても良いが，問題の主要部あるいは典型例について，一通り動作する（それらしい解が得られる）ことが重要．

③　専門家との意識合わせ

専門家の興味をひくためにはＭＭＩもある程度凝る必要がある．

確認段階の基本目的は，問題構造を把握し，専門家モデルの各要素への対応付けを実現することであり，現在のモデル設定で問題が解決可能であることを専門家とともに確認する段階である．

(2)　検証用：

①　問題解法の確定

扱える問題の範囲を拡げ「使えそうだ」という認識を専門家に持たせる．

②　実用化の可能性の検証

問題解決能力から実用性を見極め，プロジェクトの継続可否を決断する．

検証段階はシステムが物理的に最も成長する段階である．詳細な知識を多数インプリメントし，解決可能なケースを増加させる必要がある．一方，知識ベースが煩雑化し易いため，適切な分類・階層化等による知識管理を考慮し，全体の整合性を維持しなければならない．

(3)　実証用：

①　エンドユーザへの解放の可否を見極める

ＭＭＩの向上，エンドユーザに対する説得力ある説明機能，知識追加・変更用のツ

ール整備やメンテナンス体制なども重要となる．

実証システムは，問題解決システムとしてほぼ完成品であり，エンドユーザ（非専門家の場合もある）が使用可能になる段階である．したがって，知識ベースの内容を凍結するとともに，使い勝手を向上させて実用化に耐えられるシステムにする必要がある．

2．2．1．8　フェーズドインタビュー

知識獲得の一環として専門家へのインタビューを行なう場合，3種類の基本方針（WHAT，HOW，WHY）をもとに，各フェーズに応じた最適なインタビューの目的を設定する．これをフェーズドインタビューと呼ぶ．これにより，明確な方針に基づいて系統的なインタビューを行なうことができる．

(1) 基本方針

① WHAT：　目的の把握（問題の認識）

全体的・原則的な知識を広く浅く抽出し，問題の特性・構造・範囲を明確化する．

② HOW　：　手段の把握（知識の詳細化）

具体的・個別的・例外的な知識を数多く抽出し，問題全体をカバーする．

③ WHY　：　理由の把握（深層知識の発見）

知識を一般化・抽象化し，予期しない状況にも柔軟に対応できる能力を追求する．

(2) フェーズドインタビュー

① 概念化フェーズ：

・専門家が，何をもとに何をするかを中心に探る（WHAT型）

・KEはインタビュー後，自分の理解した内容を専門家モデルの形式で黒板などに描きそれを専門家に説明して確認をとる．

この時点で，一つのモデルに絞る必要はなく，考え得るモデルを複数紹介してもよい．

KEは専門家モデルを意識しながらインタビューを行い，それを図示する訓練をしておかなければならない．

② 構造化フェーズ：

・専門家がどんな手順で問題解決を実現しているかを探る（HOW型）

動詞概念に注目し，

・問題解決過程の連続性や独立性の把握

・各過程の抽象度による階層化 ➡ 部分問題化

・機械的機構へのマッピング

などを意識しながらインタビューを実施する.

・使用される知識はどのような関係を持っているかを探る（ＨＯＷ型）

名詞概念（クラス，インスタンス）に注目し，それらの意味関係を把握する

③ 詳細化フェーズ：

・各プロセスで使用される判断ルールを探る（ＨＯＷ＆ＷＨＹ型）

この段階では全体の構造があきらかにされているので，目的を持ってルールを収集することができる.

・収集したルールの冗長な部分を削除し，不明確な部分を明らかにする.

２．２．１．９ 各フェーズの作業

(1) 概念化

概念化は，問題の内容と解法との関連性を把握するために行う．すなわち，問題全体を俯瞰することが目的であり，ＷＨＡＴを中心とした知識得に基づき，関連概念と思考過程を把握する.

専門家は，問題解決のために

①何に関する知識を使って ➡ 関連する諸概念 （名詞概念）

②何をするのか ➡ 各段階における目的（動詞概念）

を把握する（インタビューメモなどを活用する）.

(2) 構造化

構造化は，専門家モデルの骨組を作成することを目的とする．すなわち，専門家の問題解決方法に沿って実際に動作可能なモデルを構成するために，ＨＯＷを中心とした知識獲得に基づき，専門家モデル実現のための構造を明確化する.

専門家は，問題解決のために

① 対象をどのように表現し ➡ 対象モデルの構造，解・仮説の表現形式

② どのように使っているか ➡ 部分問題への分割，ルール集合の設定

この段階においてＥＳ／ＳＤＥＭでは，問題解決のための各種の手法が用意されている．それらは次のようなものである.

① 問題解決過程の部分問題への分割方法

② 分析／合成の類別方法

③ 生成検査法，得点法，焦点調整法

④ 探索手法

⑤ 対象モデル構築手法

⑥ ルール整理・翻訳・抽象化

(3) 詳細化

構造化がある程度できた後，徐々に詳細レベルの知識（主にルール）を埋めていく．

① 文献などから定石的な知識を洗い出す

② ケーススタディやインタビューなどから，専門家が

・個々の判断場面（ルール集合名）で，

・何（対象モデルや仮説のどの部分）を見て，

・それがどのような状態の時（ルールの条件部），

・どう判断したか（ルールの帰結部）

をワークシートを利用して収集し，整理する．

作業が軌道に乗ったら，専門家に直接書いてもらうことが望ましい．

③ 整理が進んだ後，抽象化を試みる．

2．2．1．10 ドキュメント体系

構築作業を支援するために，３０種類に及ぶワークシート類が用意されている．これらはプロジェクトの進行状況に沿って体系化されている．

次ページに，ドキュメント体系を示す．

担当：中島（富士通）

ＥＳ／ＳＤＥＭ　ドキュメント体系

◎必須：Essential
○選択：Optional

- 調査：Survey
  - ◎問題認識　Recognition
- 計画：Planning
  - ◎開発体制　Organization
  - ◎開発環境　Environment
  - ◎システム構成　Configuration
  - ◎スケジュール　Schedule
  - ◎確認事項　Confirmation
- 概念主導設計：Concept Oriented Design …　ＣＯＤ
  - ◆概念化：Conceptualization
    - ○インタビュメモ　Interview Memo
    - ○ケーススタディメモ　Case Study Memo
    - ○用語定義　Vocabulary
    - ◎専門家モデル　Expert Model (COD-CHART)
  - ◆構造化：Structuring
    - ◎問題解決過程　Problem Solving Process
    - ◎対象モデル　Object Model
    - ○基本解集合　Built-in Solution Set
    - ○仮説集合　Hypothesis Set
    - ○分析部　Analytic Part
    - ○合成部　Synthetic Part
    - ○生成検査法　Generating & Testing Method
    - ○得点法　Scoring Method
    - ○焦点調整法　Focusing & Adjusting Method
  - ◆詳細化：Detailing
    - ◎ルール抽出　Rule Extraction
    - ◎関数定義　Funciton Specification
- システム化設計：System Design
  - ◎入力部　Input Specification
  - ◎出力部　Output Specification
  - ○外部連携　External Modules
- 進捗管理：Management
  - ◎作業記録　Log
- 検証：Verification
  - ◎検証実施報告　Verification Report

2．2．2　三菱総合研究所におけるシステム開発手順

2．1節では、知識システム開発の手順を一般論として言わばトップダウン的に展開したが、実際に特定のエキスパートシステム開発ツールを用いてシステムを開発する際にはそうしたツールに特有の機能の使い方なども合わせて考慮する必要があり、よりインプリメントのレベルに近い開発手順も明らかにしなければならない。これら2種類の開発手順は相補的なもので、第一のものがツールに依存しない開発手順であるのに対し、第二の手順はツールごとに異なるものになる。本項では特定のツールを前提にしたエキスパートシステム開発手順の例として、㈱三菱総合研究所が昭和60年に開発したルールベースのエキスパートシステム開発ツール　ＺＥＵＳを対象にしたシステム開発手順を示す。

2．2．2．1　ＺＥＵＳの概要

はじめに、対象としているエキスパートシステム開発ツールＺＥＵＳの機能を簡単にまとめておく。

ＺＥＵＳは典型的なルールベース型エキスパートシステム開発ツールで、言わゆる第一世代のシステムである。したがって、以下で説明する開発手順も最先端のＡＩ技術、エキスパートシステム技術を前提にしたものでは必ずしも無い。

(1)　知識表現機構

ＺＥＵＳの知識表現機構は2種類に大別できる。一つは対象問題についての様々な仮説を表現するための機構で、（文脈、属性、値）の三つ組と確信因子とを用い各仮説とその確からしさを表現する。文脈と属性によって対象問題の構造が表現できると共に推論の効率化も実現できる。第二の知識表現機構は仮説生成・検証のための仕組みである。すなわち、ある仮説から別の仮説を生成する、あるいはある仮説が成立するか否か検証するための知識を表現する機構であるが、ＺＥＵＳでは、プロダクション・ルールとアルゴリズム的知識の二つの形式でこれを実現している。アルゴリズム的知識はプロダクション・ルールの形式に適さない知識を表現するための枠組みで、ＦＯＲＴＲＡＮのサブルーチンなどを用いて記述される。

以上のような知識は一括して知識ベースと呼ばれるが、この知識ベースに関する知識を制御用ルールとして表現することもできる。制御用ルールによって各コンサルテーションに最適な知識ベースを選択することが可能である。

(2) 推論機構

コンサルテーション部の中心となる機能である。(1)で述べた知識表現機構のうち、仮説生成・検証のための仕組みを用い、コンサルテーションの目的となる問題解決を行なう。推論は前向き推論、後ろ向き推論ともに可能であるが、基本的にはＺＥＵＳは後ろ向き推論用のシステムである。推論の意味を考えると、前向き推論は仮説生成のため、後ろ向き推論は仮説検証のための手続きということができる。この他、補助的な推論機構としてプロダクション・ルールやアルゴリズム的知識に依らず、あらかじめ定義してある属性値間の階層関係に基く推論も可能である。

また、属性の性質によっては推論に依らず、ユーザへの問い合わせや外部ファイルへのアクセスによってその値を得る仕組みもある。

(3) 説明機能

コンサルテーションの詳細な過程を説明する機能である。多くのコマンドが用意されており、ユーザはこれらを場合に応じて使い分けることが出来る。また、知識ベース改良のために各コンサルテーションの際の知識ベース内の知識の使用状況を統計処理して表示する機能も持つ。

２．２．２．２　システム開発手順の概要

ＺＥＵＳによるシステム開発手順を図２．２－４に示す。

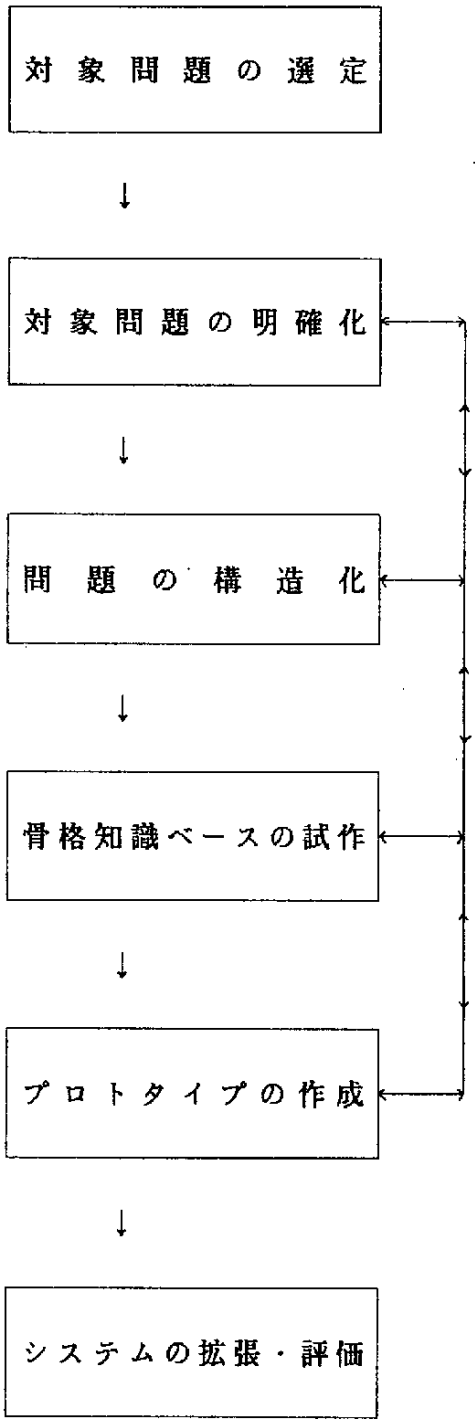


図2．2—4 ZEUSによるシステム開発手順

各フェーズごとの詳細な作業内容は事項以降に示すが、ここでは全フェーズを通じての留意点を挙げておく。

(1) 全期間に亘る知識の抽出

専門家からの知識抽出は、知識ベース構築の一段階というよりは、構築と拡張の全期間を通して行われる作業である。具体的には図2．2—4に示す知識ベース構築の各段階に

おける前半の作業として位置づけられ、各段階毎にそれまでに抽出した知識の整理および不足する知識の抽出が行われる。

(2) 専門家と知識エンジニアとの協力体制

専門家から知識を抽出する場合には、開発の初期段階において、専門家と知識エンジニアの両者が充分に時間をかけて、互いの持つ知識を吸収、理解するのが理想である。知識エンジニアが対象問題について充分理解するのはもちろんだが、可能であれば専門家にも使用するエキスパート・システム構築用ツールについて勉強してもらうのがよい。

専門家がエキスパート・システムの行う「推論」がどのようなものであるか（人間の推論に比較していかにプリミティブであるか）を認識しているのといないのでは、知識抽出の効率は大きく異なる。また、問題の構造化の段階では、何故そのような知識ベースの構造になったかを専門家が納得していた方が、その後の協力を得る上でも望ましい。

2．2．2．3 対象問題の選定

一般にエキスパート・システムに適している問題は次のようなものと言われている。

① 解決すべき問題について専門家がいるか、解くためのマニュアルが存在する問題

② 解決するために専門的な知識を必要とする問題

③ 専門家にとって難しすぎず、また易しすぎもしない問題

④ それ程専門的でなくとも、複雑で思い違いやミスが起こり易い問題

⑤ 理論的に解法が確立されたものでなく、経験的な判断知識が重要視される問題

⑥ 常識を必要としない狭い範囲の知識で解ける問題

以上はエキスパート・システム一般に言える事柄であるが、この他、使用するエキスパート・システム構築用ツールの性格によって問題の適、不適がある。

ＺＥＵＳの場合は、

① 目標指向の分析型の問題

② 対話型で使用するコンサルテーション型の問題

③ 個別のコンサルテーションの推論の流れの構造を抽象的な階層構造として表現できる問題

を最も得意としており、また次のような問題を取り扱うこともできる。

① あいまいな判断知識や入力データを含む問題

② 推論やユーザへの質問によって何も得られない時、あらかじめ決められたデフォルト値を使用する問題

③ 経験的知識だけでなく、アルゴリズムや数値計算により解ける副問題を含んでいる問題

④ 複数の知識体系（知識ベース）を場合によって使い分けるような問題

また、ＺＥＵＳは、次のような問題を解くのは一般に不得意である。

① 図形や手続き的なプログラムを入力し、それを解析するような問題

② 時間の概念が重要な役割を果たし、時系列データなどを解析する問題

③ リアルタイムでデータを処理する問題

④ 合成型でしかも複雑な問題

しかし、これらの問題の場合も、解析へのアプローチや問題の構造化を工夫したり、アルゴリズム的知識を応用することにより、ある程度カバーすることも可能である。

また、当然のことではあるが、エキスパート・システムが専門的知識を多用した難易度の高い問題を解くのに適しているとは言え、それは従来型のプログラムと比較しての話であり、開発には多くの困難があることを付け加えておく。

２．２．２．４　対象問題の明確化

対象とする問題が選定されると、その問題をどのように捕らえ、何をどのように解いていくかを決めるための対象の明確化の作業を始める。しかし、この作業は、前節の対象問題の選定とそれ程明確に区別される訳ではない。先に挙げた選定のための各ポイントは、問題の扱いがある程度決まらないと明らかにならないことがほとんどだからである。

対象問題の明確化における主要な検討項目は次の３つである。

① 取り扱う問題の範囲の限定

② 問題の特徴把握と解決へのアプローチ

③ 専門家の問題解決手順の分析

現実には、これらも互いに密接な関わりを持っているが、以下、それぞれについて行うべき検討内容を示そう。

(1) 取り扱う問題の範囲の限定

まず対象とする問題の範囲をどの程度とするかについて明らかにしなくてはならない。エキスパート・システムの開発の目的にもよるが、一般には短期、中期、長期の３段階について考える必要がある。これはそれぞれ、プロトタイプ・システム、ポスト・プロトタイプ・システム、実用システムに対応する。エキスパート・システム開発の特徴の１つはその拡張性を生かした段階的な開発であるため、始めに、こうした実用化へのシナリオを

作っておくのがよい。

ここでは、開発の第一歩としてのプロトタイプ・システムのみについて記述することにするが、この段階では問題のフィージビリティの確認を目的としており、以下のことに注意して問題の範囲を限定する。

① 対象とする問題のうち典型的でしかも難易度があまり低くないケースを解けるものであること。

② 社会科学分野の問題で複雑度の高いものである場合には、入力情報を、結果に大きな影響を与えるいくつかの主要因に絞ること。

③ 3～10ヵ月程度で完成できる範囲であること。

④ 100～500程度のルール数でカバーできる範囲であること。

⑤ プロトタイプの完成時、そのフィージビリティを評価するためのテストケースを設定できるようなものであること。

(2) 問題の特徴把握と解決へのアプローチ

前述のように、問題の特徴や解決へのアプローチについての検討は、対象問題の選定においてある程度行われているはずであるが、ここでは入出力情報を含め、さらに深く検討する。

一般にある問題の解決にはいくつかの可能なアプローチが存在し、どれを採用するかによって、問題の特徴や性格が大きく変わることも多い。アプローチを決定する際、問題の特徴に影響を与えると思われる事項としては、以下のものが挙げられる。

① エンド・ユーザはその問題についてある程度知っている人物か、あるいは全くの素人か。

② 出力は意思決定の材料としての状況・情勢の判断なのか、あるいは意思決定を含めた判断をも下すのか。

③ 出力される解は最善、または可能性の高いものだけに絞るか、あるいは可能性のあるものすべてを出力したいのか。

④ 事実の因果関係を忠実にシミュレートして判断するのか、あるいはもっと高次レベルの相関関係や経験に基づいて判断するのか。

⑤ 公的機関の発表やプラントの検出器などからの生データを基に判断するのか、あるいは人間のフィルタを通したデータや、数値化できないデータを基に判断するのか。

知識エンジニアはこれらの問題を充分に踏まえ、対象問題の解決とエキスパート・システム構築用ツール（ＺＥＵＳ）の性格の両面から最も適したアプローチを決定しなくてはならない。

(3)　専門家の問題解決手順の分析

取り扱う問題の範囲限定、特徴把握および解決へのアプローチが固まってきたら、専門家が実際にその問題をどのように解決するかの分析を行う。

この作業は協力を依頼した専門家のタイプによって全く異なったものとなり、結果としてでき上がるエキスパート・システムの推論プロセスにも大きな影響を与える。また(2)のアプローチそのものも、協力する専門家の影響は通常避けられない。

専門家のタイプとは、個人に依るものも大きいであろうが、一般には、現場の第一線で直接問題に携わっている者か、背後でその問題の分析や検討を行っている者かの違いである。例えば前者はプラントのベテラン・オペレータや為替のディーラーであり、後者はプラントの設計者や管理者、経済全体の立場から為替を分析する者である。、一概にどちらのタイプの専門家が良いということはできないが、前者では問題解決の手順分析が難しい一方、でき上がる知識ベースはコンパクトでエンドユーザに分かり易いものとなり、後者はその逆であるとは言えるだろう。

この段階での作業は、推論の手順を具体的に逐一取り出すと言うよりは、むしろ問題解決に必要な概念を取り出し、それらを関連づけて整理することに重点を置く。専門家の頭の中にある問題の構造を抽出すると言ってもよい。抽出した各概念については以下のことを明らかにする必要がある。

①　各概念の定義は何であるか。

②　各概念はどのような属性により特徴づけられるか。

③　各概念間の因果関係はどのようになっているか。

これらについての分析が終わり、専門家の行う問題解決の構造が明らかになったら、これを簡単な問題解決概念図としてまとめておくのがよいであろう。

２．２．２．５　問題の構造化

エキスパート・システムの取り扱う問題についての明確化が終了すると、いよいよ物としての知識ベースの構築へ向けての作業が開始される。一般のシステム開発との対応で考えると、前節までが問題の分析と要求仕様の決定で、この段階が基本設計と言えるかもしれない。

ここで行う作業は、専門家とのインタビューやマニュアルから抽出した各種の概念およびそれらの関係を、いかに知識ベースとして実現するかを決定することである。換言すると専門家の頭の中にある問題の推論構造をエキスパート・システム用の推論構造にブレイク・ダウンすることでもある。

エキスパート・システムの持つ構造には次の２つのレベルのものが考えられる。

① エキスパート・システムが行う推論プロセスの構造

② エキスパート・システム構築用ツールが持つ知識表現形式に合わせた知識ベース構造

前者は、推論の流れの中に陰に埋め込まれた概念レベルでの構造で、必ずしもツールの持つ知識表現形式に依存するものではない。これは全く平板な物理的構造を持つ知識ベース上にも存在でき、知識エンジニアが専門家の推論構造をルールによって模擬するために作り出すものである。これに対し、後者は文脈クラス樹や黒板、ルールのグループ化など前者の推論構造の実現を助けるための物理的仕組みといえる。

もちろん専門家の推論構造とエキスパート・システムの推論構造が全く同じであり、知識ベースの構造はそれらを素直に表現できるものが理想であるが、それは一般に不可能に近い。現実の問題の構造は千差万別であり、それらすべてを素直に表現するためには、問題毎に個別のツールを作らざるを得ない。通常は与えられた知識表現の範囲内で、より自然な表現となるように知識ベースの構造を考えることになる。

以下、ここでは対象とするツールをＺＥＵＳに限定し、推論プロセスの構造化および知識ベースの構造化をどのように行えばよいかについて述べる。

(1) 推論プロセスの構造化

ＺＥＵＳは後ろ向き推論と前向き推論の両方を扱うことができるが、推論プロセスの構造化はそれぞれ異なる方法で行わねばならない。ＺＥＵＳを含めて一般にルール・ベースシステムでは、簡単な問題を除き、後ろ向き推論の知識ベースと前向き推論用の知識ベースはかなり異なったものとならざるを得ないからである。

① 後ろ向き推論を主体とする場合

後ろ向き推論を主体とする知識ベースの場合には、まず推論の目標となるpurpose 属性を決定する。ＺＥＵＳが行う推論の結果は最終的にこの属性の値として集約されるので、purpose 属性の値がコンサルテーションで得られる答となるように知識ベースを設計するのが最も自然である。

しかし、すべての問題をこのように扱うことはできない。複数の一連の属性の値を求めるような問題や、解が多くの要素と構造を持っている問題も数多くある。このような場合には、purpose 属性をコンサルテーションを開始させるためのダミー属性として考え、それを追跡することをきっかけとして必要な複数の属性の値や構造が求まるように推論のプロセスを構造化することになる。

後ろ向き推論は、まず目標（purpose 属性）を設定し、その解を求めるためにいくつかの副目標を作り、さらにそれらを求めるためにより細かい副目標を作っていく推論形式である。つまり、推論プロセスの主要構造は、ほとんど上位レベルの推論ルールによって決められていると言ってよい。従って、後ろ向き推論の場合の推論プロセスの構造化は、purpose 属性に近い上位レベルのイメージ・ルールの（言葉で表現するレベルのルール）を作ることに相当している。なお、イメージ・ルールの数は問題にもよるが、大体１０～２０程度が適当であろう。

② 前向き推論を主体とする場合

前向き推論を主体とする知識ベースの場合には、まず与えられた初期データからコンサルテーションの目標に至る処理手順を決定する。これは、前向き推論がデータ駆動でルールを起動する、言わばデータまかせの推論を行うために、コンサルテーションと関係のない無駄な推論を行わないよう、ユーザが目標へ向かう道筋をつける必要があるからである。またこのことは、前向き推論ではユーザが自ら推論の制御を行わねばならないことを意味し、後ろ向きの場合に較べてかなり手続き的なルールを作ることになる。

処理手順を決定した後は、この手順に従って推論が行われるようにルールの起動を制御する仕組みを考える。これには問題によって様々な方法があろうし、また実際のルール作成時に場当たり的に考えることもできるが、一般には、処理の各段階をいくつかのモードまたはタスクに分割し、それらを順に処理していくと目標に到達するように考えるのがよい。

(2) 知識ベースの構造化

ＺＥＵＳは文脈クラス樹によって、知識ベースを明示的に構造化する手段を提供している。文脈クラス樹による構造化は推論の流れ全体を概念レベルで構造化するものである。また、問題の適性にも依るが、(1)における推論プロセスの構造を素直に反映させることが可能であり、同じ問題を扱うにしても解決へのアプローチによって全く異なる文脈クラス樹を作成することができる。

文脈クラス樹は基本的に、下位の文脈クラスの表す概念から上位の文脈クラスの概念が特徴づけられ、しかも各概念にはいくつかの具体例（文脈インスタンス）が存在する、という問題に適している。各概念の特徴は、複数の属性によって記述することができる。

このような適性から見ても、文脈クラス樹はどのような問題をも構造化できる程、汎用性は高くない。従って取り扱う問題や　解決へのアプローチに向いてないと思われる時、およびＺＥＵＳを初めて使用する時には、無理に文脈クラス樹を使用しない方が無難である。また、実際に知識をインプリメントしている時、文脈クラス樹の構造に無理を感じた場合はなるべく早期にその構造を再検討し、適切でないと思われた時は知識ベースの構造化からやり直すべきである。

２．２．２．６　骨格知識ベースの試作

問題の構造化が終了すると次はルールのコーディングに入る。しかし、通常のプログラム開発のようにコーディングのすべてが終了してからデバックを開始するのではなく、ルール・コーディング、インプリメント、テストを繰り返し、少しずつ大きくしていくのがよい。エキスパート・システムの最大の長所の一つは段階的にシステムを開発できるところにある。しかし、エキスパート・システムが従来のシステムに較べていくら拡張性があるとはいえ、推論の基本的枠組みの変更は知識ベースの再構築を意味するから、問題の構造化で手抜きをせず、充分に検討した後に始めるべきである。また推論のプロセスが不自然であったり、今後の拡張が難しそうであったら、この試作の段階で再構築を行うか、このままプロトタイプまで拡張を行うかの検討を行うことが必要である。プリ・プロトタイプともいえる骨格知識ベースの規模は、問題にもよるが、ルール数で数十から百程度と考えればよい。

以下ではこの段階で中心となるルールの作成に関して、その手順と視点の重要性について述べる。

(1)　プロダクション・ルール作成の手順

基本的にルールベースのシステムを構築する場合は、その推論の流れに沿って、順にルールを作成していくのが最も自然であり、無駄も少ない。後ろ向き推論を主体とするのであれば、purpose 属性を結論するルールから、前向き推論であれば与えられた初期データを前提部に持つルールから作り始めることになる。以下、それぞれの場合についてもう少し詳しく述べよう。

① 後ろ向き推論を主体とする場合

後ろ向き推論を主体とするシステムを構築する場合は、まず、目的となる属性（purpose 属性）について結論するルールをすべて作成し、それらの前提部に現れる属性の値を得るルールを次に作る。すなわち、ルール作成はトップダウンに行うべきである。このルール作成の手順には次のようなメリットがある。

（a） システムの行う推論の連鎖を順にたどるような結果になるので、全体が見通せる上、知識の欠落が少なくなる。

（b） 必要なルールのすべてができなくても、インプリメントすればすぐに動かすことができ、システム全体の動きの観点からルールの正当性をチェックできる。従って、ルールの作成とインプリメントおよびチェックを平行して、インクリメンタルに行える。

（c） 後ろ向き推論では、トップレベルに近いルールが推論の枠組みを構成しており、上から下へとルールを作成することは、外部仕様から内部仕様へと少しずつブレイク・ダウンしていくことに相当する。

② 前向き推論を主体とする場合

前向き推論の場合は、データ駆動で推論されるから、与えられた初期データを前提部に持つものから順に考えていくのがよい。しかし、後ろ向きの場合とは異なり、前向きの場合は推論のゴールへ向かうルールだけが起動されるとは限らないため、システムが行う推論の連鎖をたどってルールを作成することは難しい。従って、実際には、推論全体をいくつかの処理モード（あるいは小タスク）に分割し、それらのモードが実行される順に沿って、モード毎にルールを構築していくのがよい。

(2) プロダクション・ルールの作成の視点

プロダクション・ルールの表現するものは人間の判断知識であり、論理学における含意(implication)とは全く別物である。推論エンジンの行う処理は、形式的にはルールとその前提部が表現する事実から結論を導く演えき推論であるが、ルールそのものは原因から結果を導くもの、結果としての現象から原因を推定するもの、専門家の勘など千差万別である。

このような判断知識の場合、ルールはその視点によって正しくも誤りにもなる。異なる視点から作成された２つのルールは、それぞれ単独では正しくとも、２つ合わせると互いに矛盾となる場合がある。if～then～という判断の形式は人間にとって理解し易く表現力

も豊かであるが、一方ではこのような矛盾を知らぬうちに作り出してしまう危険性もある。知識ベースを作成する時には常に、どの視点に立ち、何を推論しようとしているのかを意識しておく必要がある。

2．2．2．7　プロトタイプ作成と拡張・評価

図2．2－4ではプロトタイプの作成と拡張を分けて書いてあるが、実際には、プロトタイプも骨格知識ベース（プリ・プロトタイプ）からの段階的拡張によって構築される。プロトタイプの作成および拡張において注意すべき事項を以下に示す。

① ルールは、なるべく相互の依存関係のないように作る。

② 属性名は、長くてもよいから見てわかり易いものとする。

③ ユーザへの質問のプロンプトもわかり易い充実したものとする。

④ ルール作成時、未使用ルール番号を充分に用意しておく。

(1) 相互依存しないルール

ＺＥＵＳの知識ベースは断片的な知識であるプロダクション・ルールから作られておりコンサルテーション部がそれらを組み合わせて推論する。またルールの適用はすべてＺＥＵＳが決め、ユーザは陽にルールの適用順番を指定できない。従って、例えば、あるルールが先に適用できるかどうかチェックされ、それが失敗した時に次のルールが適用されるということを意識し、そのような順で適用された時のみ正しく動くようなルールは作るべきではない。当面は正しく動いても、ルールのｃｆ値を変更したり、そのルールが参照している属性を使用する他のルールが修正・追加されたりすると、適用の順番が変わる恐れがあるからである。

相互依存しないようなルールは、各ルール毎にそれが実行される条件を厳密に記述したもので、知識ベース全体からみると冗長ではあるが、各ルールのモジュール性および宣言的知識の色彩を強くし、拡張性が非常によくなる。

(2) 属性名の付け方

ＺＥＵＳのルールは推論を行うだけでなく、その過程をユーザに説明するためにも使用される。そのため、ＺＥＵＳはルールを表示する時に、内部のLisp表現を英語風に翻訳する。しかし、ユーザにとってそのルールがわかり易いか否かは、使用されている属性の名前がその属性の意味を適切に表現しているかどうかに依るといってよい。

知識エンジニアは、想定するエンド・ユーザにとって理解し易い属性名となるように注意してつけるべきである。

(3) プロンプトの充実

コンサルテーション時に出力するユーザへのプロンプトも軽視してはならない。プロトタイプが作成され、その評価を行う時、専門家はわかり易さを非常に重視する。対象領域が工学の分野である場合は、専門家は機械や計算機に慣れており、多少の使い勝手の悪さは気にしないことが多いが、社会科学や医療であるとそうはいかない。プロトタイプだからとプロンプトの充実を怠ると、わかりにくく、使いたくないシステムだと評価されがちである。

(4) ルール番号の付け方

ルール番号はルール作成順に付けていくのが自然であるが、後からのルール追加を考え空き番号を充分にとっておくのがよい。ＺＥＵＳは知識生成支援部でルール番号の変更機能は持っているものの、途中にルールを挿入するためにそれ以降の番号をもつルールを１つずつ変更するのは現実的ではない。

この様にして、逐次的にシステムの拡張を行いながら、一方でシステムの評価を実施していく必要がある。システムの評価項目はシステムの目的にもよるが、大ざっぱには以下の様に考えることができる。

評価項目 / レベル	1 助言、決定の正当性	2 推論の正確さ、知識表現の適切さ	3 マン・マシン インターフェイス	4 性能	5 コスト・パフォーマンス
1 (プロトタイプ)	評価の対象 →				
2 (ポスト・プロトタイプ)			評価の対象 →		
3 (実用システム)				評価の対象 →	

図　2．2—5エキスパート・システムのレベルと評価項目の関連

なお以上に示したシステム開発手順は、特定のツールを前提にしたものとしてはかなり概念的な段階のものである。実際のシステムのインプリメントに際してはＺＥＵＳの機能・特徴を強く意識したノウハウ、テクニック等を盛り込む必要があり、こうした手法も開発手順の中に含まれているが、ここではそれらについての説明は割愛している。

（担当　三菱総合研究所　小林）

参考文献

〔Hayes-Roth　83〕F.Hayes-Roth　et　al.,Building　Expert　Systems,
Addison-Wesley　Publishing　Company Inc., 1983

〔三菱総研　85〕三菱総合研究所，人工知能革命フェーズ２事例研究報告書，1985

3. 応用分野の特徴とアプローチ

3．応用分野の特徴とアプローチ

3.1 問題のクラスと基本的な接近法

本節では，知識システムの対象を解析型問題と分析型問題に分類し，各問題の基本的な特徴について考察を加え，各問題に要請される基本的なタスク，これを実現するための問題解決機能を論じる[小林 87-1]，[小林 87-2]．

3.1.1 解析型問題と合成型問題

知識システムが対象とする問題は，大別すると，解析型問題と合成型問題に分けられる．解析型問題とは，図3.1-1 に示すように，システムの構造とサブシステムの特性が与えられたとき，システムの特性を推測するような問題である．一方，合成型問題とは，図3.1-2 に示すように，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ような問題である．

一般に，サブシステムの特性とシステムの構造が与えられたとき，システムの特性が一意に定まるのに対し，システムの特性が与えられたとき，これを実現するサブシステムの特性とシステム構造は無限に存在する．ここに合成型問題の基本的な難しさが存在する．

解析型問題でも，比較的単純な分類問題や診断問題を例外として，少し複雑な問題にな

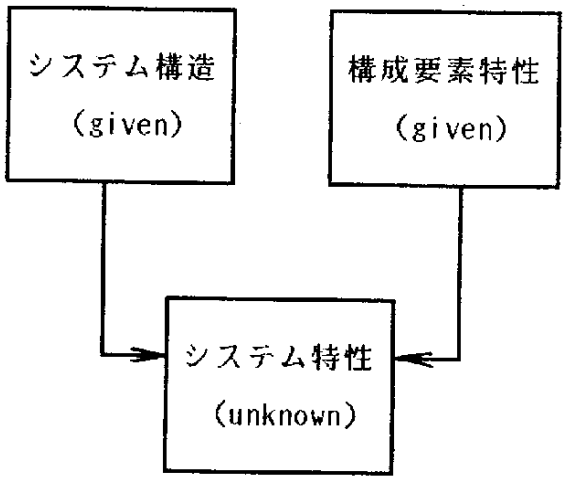


図3.1-1 解析型問題

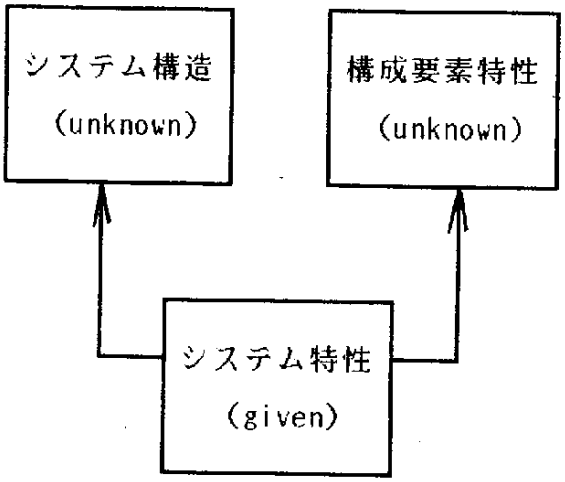


図3.1-2 合成型問題(クラス１)

ると，いくつかの制約条件を満たすことが要請され，組合せ的爆発の問題をどう克服するかが主要な問題とされる．

一方，合成型問題はもともと組合せ的な性質を有する．合成型問題に対するもっとも一般的な手法として，“生成検査法”(generate and test)がある．これはシステムの構造を仮に生成し，解析を行ってシステムの特性を求めその評価を行うことを繰り返して，最適なシステムを探索する方法である．この方法は“解析による合成”(synthesis by analysis)とも呼ばれる．この意味で，合成問題は解析問題を内包しているといえる．

3.1.2 個別問題の定義，特徴，基本タスクおよび問題解決機能

解析型問題および合成問題は，さらにつぎのように分けられる[小林 86]．

[解析問題]

(1) 解釈問題

計測器あるいはセンサを通して観測されたデータを分析して，システムの状態を推定し，これに物理的な意味づけを与えること．

(2) 診断問題

システムに異常または故障が生じたとき，観測されたデータおよびシステムに関する因果関係または対象モデルの知識を利用して，原因を同定すること．

(3) 制御問題

システムの状態を監視していて，予め定められた計画どおりにシステムの状態が遷移するようにシステムを制御すること．

[合成問題]

(1) 計画問題

与えられた目標を実現するために，利用可能な行動を系列化すること．

(2) 設計問題

システムの入力と出力の要求仕様が与えられたとき，これを実現するために，構成要素を組み合わせ，かつ各構成要素の内部仕様を決定すること．

表3.1-1 および表3.1-2 に各問題の特徴，基本タスク，問題解決機能を示す．3.2～3.5の各節において，事例を含めて，これらを詳しく論じる．

表3.1-1 および表3.1-2 からも明らかなように，解析型問題と合成型問題では非常に異なった型のタスクおよびこれに関連する問題解決機能が要請されることが理解されよう．また解析型問題に限っても，解釈問題と制御問題では異なった接近法が必要とされることが分かる．

これまで，知識システムを構築するために，数多くのシステム構築ツールが開発され，利用されるようになってきている．しかしこれらは汎用性を追求するために，知識の表現と利用において，問題領域の構造に依存しない普遍的なメカニズムを提供している．しかしそのために，非常に低いレベルでの知識表現および問題解決能力しか提供しないことになり，知識技術者の負担が非常に重いものになっている事実を見逃すわけにはいかない．

このようなギャップを埋めるためには，各問題領域ごとに抽出された基本タスクおよび問題解決機能を汎用的に利用できるパッケージのライブラリの形にまとめ，必要に応じてそれらを組み合わせて利用する考えが要請される．

3.1.3 類型的タスク

[Chandrasekaran 86] は，従来の知識システムがルールやフレームのような実装レベルでの知識表現を中心に組織化されたことによる問題点を指摘したうえで，問題解決を中心に組織化することの必要性を指摘し，類型的タスク（Generic Task）という考えを提唱している．この考えは，知識処理の対象とされる問題の解決をいくつかの類型的タスクの組み合わせによって実現しようとするものである．

彼らはこれまでにつぎの６つの型の類型的タスクを抽出し，モデル化している．

(1)階層的分類 (Hierachical Classification)

分析の対象について観測された事実に照合する仮説を，分類型の知識階層木の中より見いだす類型的タスクをいい，診断型エキスパートシステムにおいて中心的な役割を果たす．このタスクでは，階層構造をトップダウン的に調べることにより，適合する仮説が検索される．このタスクを実現するために，確証／精密化（establish／refine）と呼ばれる知識表現と推論方法が提案されている．

この方法では，各仮説は階層木のノードに対応づけられる．各ノードにはスペシャリストと呼ばれる推論モジュールが付加される．このモジュールは確証部と精密化部からなる．確証部では観測事実がそのノードが表す仮説に照合するかどうかを判断する．精密化部は受理された仮説をより下位のノードに送るための判断を行う．これらはスペシャリスト間

のメッセージパッシング機構によって実現されている．

(2)仮説照合・評価 (Hypothesis Matching or Assesment)

階層的分類において，確証メッセージを受けた スペシャリストは，不確実性が伴うデータと仮説の間でどの程度照合するかを決定しなければならない．

仮説照合タスクでは，一般に仮説の評価は，単一データからではなく，複数の異なったデータから総合的に判断する必要がある．この場合に生じるあいまいさを処理するための知識および推論方法がこのタスクを構成する．

(3)知識指向的情報処理 (Knowledge-directed Information Processing)

観測された事実やデータが不完全な場合には，これを補完するための処理が必要とされる．フレームシステムにおけるデフォルト値の利用や性質の継承などがこれに相当する．もっとも一般的な言い方をすれば，知識指向的情報処理タスクとは，デフォルト推論および知識ベースを利用して，不完全なデータを完全なものにすることである．

(4)発想論的仮説の組み立て (Abductive Assembly)

解釈や診断問題では，いくつかの要因が複雑に絡み合った複合的な事象や状況を診断しなければならない場合が少なくない．(1)～(3)の類型的タスクによって，部分的にデータを説明する要素仮説の集合が得られたとして，これらから観測データをもっともよく説明するような複合的仮説を生成しなければならない．

発想論的仮説の組み立てとは，要素的仮説集合が得られたとき，仮説の関連性に関する知識を利用して，発想論的に複合仮説を合成するタスクをいう．このタスクは要素仮説を逐次加えて複合仮説を生成し，評価を行い，妥当性に欠ける仮説を排除しながら，より好ましい複合仮説を探索することからなる．

(5)計画選択・精密化による階層的設計 (Hierachial design by Plan Selection and Refinement)

設計問題は，システムの特性が仕様として与えられたとき，これを実現するシステムの構造および構成要素の特性を決定することである．一般に，設計問題では可能なシステム構造と構成要素の特性は無限に存在するために，設計活動は非定型的であり，類型化は困難である．しかしシステムの構造が所与で，構成要素の特性の決定方法がある程度確定している場合には，定型的となり，類型化が可能である．このような定型的設計を支援する類型的タスクとして，計画選択／詳細化による階層的設計が提案されている．このタスクは診断問題における確証／精密化プロセスとアナロジーの関係にあり，同じようなメカニ

ズムによって実現されている．すなわち与えられたシステム構造に対応して階層木を作成し，各ノードに付加されたスペシャリストによって計画の選択および下位ノードへの精密化が行われる．この作業はスペシャリスト間のメッセージパッシングによって進められ，必要に応じて上位階層への後戻りが行われる．

(6)状態抽象化（State Abstraction）

定型的設計では，システム構造が所与であり，構成要素の選択が中心的な役割を果たす．しかし構成要素の特性を決定したとしても，それがシステムの特性にどのような影響を与えるかは他の構成要素の特性が定まらない限り，不確実である．階層的設計において，ある階層における構成要素の特性の決定が上位階層にどのような影響を与えるのかを定性的に予測するのが状態抽象化と呼ばれる類型的タスクである．このタスクを実現するためにconsolidation と呼ばれる定性推論の１種が使われている．

以上の類型的タスクのうち，階層的分類および仮説の照合と評価についての類型的タスクについては，ＣＳＲＬと呼ばれるシステム構築ツールにおいて実現され，主として診断問題向けに適用されている．設計型問題に対しては，ルーチン的設計問題に限って，ＤＳＰＬと呼ばれるシステム構築ツールが開発されている．

Chandrasekaran らによって提案されている類型型タスクは，まだ典型的な診断問題および典型的なルーチン的設計問題に限られており，表3.1-1 および表3.1-2 に示される基本タスクをカバーするまでには至っていない．しかし，この類型的タスクという考えを発展させることにより，次世代の知識システム，特に，計画や設計問題をモデル化していくうえで重要な手がかりになることが期待される．

3.1.4 合成型問題のクラス

図3.1-2 に示されるもっとも一般的な合成型問題を“クラス１の問題”と呼ぶことにする．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．

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

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

より特殊化された合成型問題として，＜システムの構造〉または〈構成要素の特性〉のいずれか一方が所与の問題が考えられる．

〈構成要素の特性〉が所与の合成型問題をクラス２の合成型問題と呼ぶことにする．

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

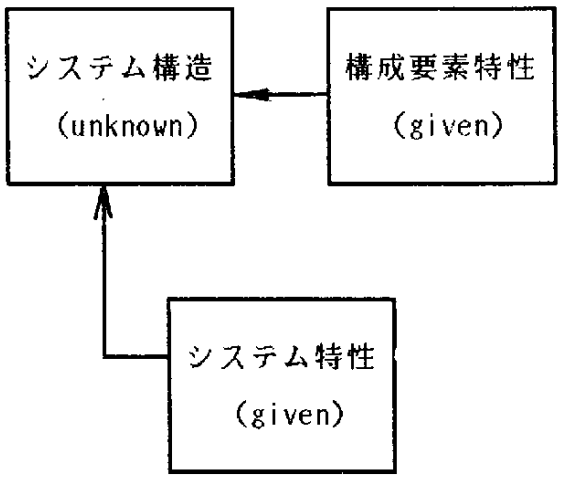


図3.1-3 合成型問題(クラス２)

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〉が所与の合成型問題をクラス３の合成型問題と呼ぶことにする．

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

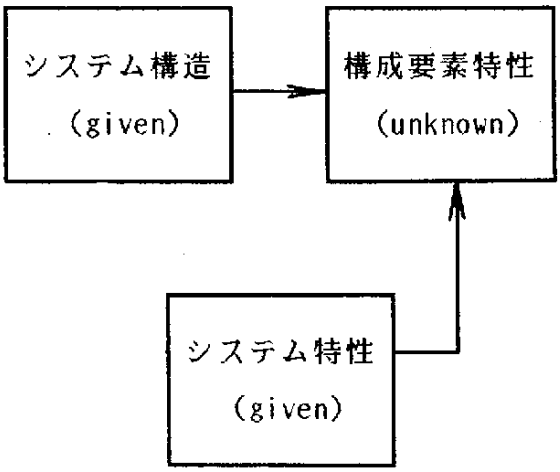


図3.1-4 合成型問題(クラス４)

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

図3.1-3 および図3.1-4 に，クラス２とクラス３の合成型問題における＜システムの特性＞，＜システムの構造＞および＜構成要素の特性＞の関係を示す．

3.1.5 合成型問題における類型的タスク

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

[1] クラス３の合成型問題における類型的タスク

クラス３ではシステム構造は所与であるので，これを階層木に対応させれば，階層的生成検査法に基づく接近が可能である．Chandrasekaran らは，これを計画選択・精密化による階層的設計と呼ばれる類型的タスクによって実現している．

階層的設計において構成要素の特性を決定するためには，それがシステム全体の特性にどのような影響を与えるかを推定する必要がある．ある階層における構成要素の特性の上位階層への定性的な影響を予測，評価する必要がある．Chandrasekaran らは，これを状態抽象化と呼ばれる類型的タスクによって実現している．

以上２つの類型的タスクは，階層的生成検査法の１つの具体化を与えるものである．

[2] クラス２の合成型問題における類型的タスク

クラス２の合成型問題では，構成要素は所与であるが，システム構造は未知である．構成要素のつながりをすべて同時に考慮することは組合せ的複雑さのために困難である．普通に取られる戦略は，構成要素群の中で，空間的および時間的に重要と考えられるものをいくつか取り出し，他は考慮しないで，まずそれらの間の部分的システム構造を決定する．ついでこの部分的システム構造を制約条件として，つぎに重要と思われる構成要素群を考慮の中に入れ，それらの間のつながりを決定するとともに，すでにつくられた部分的システム構造の中に埋め込むことを行う．このような過程はすべての構成要素が考慮されるまで再帰的に繰り返される．この戦略はトップダウン精密化（topdown refinement)と呼ばれる．

上で述べたように，クラス２の合成型問題に対する基本的な問題解決戦略はトップダウ

ン精密化である．トップダウン精密化による設計を可能とするためには，

第１に，構成要素の集合を，重要度または影響度に応じて，いくつかの部分集合に分割できなければならない．そのためには，与えられた構成要素の集合をある評価基準のもとで最適に分割するためのタスクを必要とする．

第２に，部分集合内では部分的システム構造の決定ができなければならない．そのためには，手続きまたは最適化によって，最適な部分的システム構造を見出すためのタスクを必要とする．

第３に，上位階層での部分的システム構造の決定は，それより下位の問題に対して，制約条件として機能しなければならない．そのためには，制約条件の伝播が適切に行うタスクを必要とする．

第４に，上位レベルでの決定が下位レベルでの実行可能性を侵害してはならない．そのためには，抽象的レベルでの実行可能性を適切に評価するタスクを必要とする．

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

[3] クラス１の合成型問題における類型的タスク

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

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

以上３つのクラスの合成型問題における類型的タスクを考察したが，各問題に対する基

本的な接近法および類型的タスクをまとめると，次表のようになる．

表3.1-3 合成型問題における類型的タスク

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

3.1.6 まとめ

本節では，知識システムへの基本的な接近を応用領域の特徴分析に基づいて考察した．従来のシステム構築ツールが汎用性を追求するために，非常に低いレベルでの知識表現および問題解決能力しか提供しないことによる問題点を指摘したうえで，このギャップを埋めるために，各問題領域ごとの特徴分析を行い，これに基づく基本タスクおよび問題解決機能を明らかにすることの必要性を指摘した．この方向は類型的タスクという考え方につながるが，現時点では類型的タスクを直ちにツール化することは，試験的には有用であっても，実用的観点からは，時期尚早と思われる．

合成型問題におけるタスクの類型化についてトップダウン的な考察を行った．現実の合成型問題は，表3.1-1 および表3.1-2 に示されるようにもっと多様であり，問題のクラスをより詳細に分類し，事例との対応づけを行いながら，合成型問題の類型化を進めていくことが必要である．そうすることにより，合成型問題に対する知識システム構築の方法が体系化されてくるものと期待できる．

（小 林 重 信）

[参考文献]

[小林 86]　小林重信：知識工学，昭晃堂，1986.

[小林 87-1] 小林重信：知識システム方法論，システムと制御，Vol. 31，No.4，1987.

[小林 87-2] 小林重信：問題解決と類型的タスク，ICOT KSS/KRP WG 資料，1987.

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

3.2 解釈型知識システム

計測の目的は，対象とするシステムが自律的または他律的に発生する連続的／離散的なデータを計測器やセンサによって検出し，これに何らかの処理を施して信号に変換するとともに，信号に含まれる特徴を抽出し，対象の理解に基づいて，その物理的な意味を解釈することにある．この前半の部分は信号処理，後半の部分は信号理解またはデータ解釈と呼ばれる．

信号処理では，計測器やセンサから得られたアナログ信号をディジタル信号に変換すること，時系列データであるディジタル信号に対して，フィルタリング，フーリエ変換，スペクトル分析などの信号処理技術を施して，何らかの意味で正規化された信号を抽出することが主目的とされる．信号処理の結果だけから，対象に関する十分な理解が得られる場合も少なくない．しかし対象の構造および挙動が複雑になればなるほど，信号処理の結果だけからは，物理的に意味のある帰結を得ることは困難になる．

ここに，信号理解のプロセスが要請され，解釈型知識システムのニーズが存在する．たとえば，ソナーから得た音響データの解釈，人間の音声データの解釈，画像データの解釈，センサデータに基づくプロセスの診断，X線回折データの解釈などが解釈型知識システムの処理対象とされる．

3.2.1 解釈型問題の特徴

解釈型問題の特徴は，つぎのように要約される．

(1) 時空間データの高次相関性

大規模・複雑な対象では，計測データは多次元に及び，それらが時間的にも空間的にも高次の相関を有するために，従来の信号処理の結果だけからは，物理的に意味のある帰結を得ることが困難である．

(2) 不規則な雑音の混入

計測データの中に，従来のフィルタリング技術だけでは除去できない不規則な雑音が存在する場合，そのような雑音の発生源および原因を追求することにより，これを処理しなければならない．

(3) 誤りの存在

データ中に誤りが存在する可能性がつねにある．したがって，誤りが存在することを前

提として，推論を行い，もし矛盾が生じた場合には，後戻りして，誤りの箇所を同定して，これを訂正しなければならない．

(4) 必要なデータの欠落

解釈に必要なデータが完全にそろっているのはむしろ例外的なことであり，必要なデータが一部欠落していることがふつうであり，そのような状況のもとでも解釈がおこなえるような頑健性が要求される．

(5) 信号処理技術との接続

計測器から取得されたデータが直ちに知識処理の対象とされるのは例外であり，通常はデータの前処理において従来の信号処理技術の利用が必要である．したがって，信号処理技術と知識処理技術の間での役割分担およびその接続が要請される．

(6) 対象に関するモデルの利用

解釈型知識システムでは，対象の理解に基づいての処理，すなわち，対象のシステム構造，機能および挙動に関するモデルを構築して，これを陽に利用して問題解決を行うことが基本である．

(7) 感覚データの取扱い

視覚，聴覚，臭覚など人間が得意とする感覚データに基づく判断が大きなウェートを占めている問題では，これを知識処理だけで解決することは極めて困難であり，いくつかの研究課題を残している．

3.2.2 解釈型問題における基本タスク

解釈型問題に対する知識処理における基本的タスクとしては，つぎのものが考えられる．

(1) 特徴抽出

特徴抽出のタスクは，基本的には信号処理技術によるところが大きいが，人間の専門家に頼っている場合には，知識獲得が主たる作業となる．抽出された特徴は分類階層概念に基づくクラス分けの対象とされる．

(2) モデルとの不完全な照合

特徴抽出によって構造化された情報と予めシステムに登録されている構成要素のモデルの間では局所的な照合が行われる．モデルの検索にはパターン照合のタスクが要請される．構成要素レベルでの対応づけにおいても不完全性が存在する．

(3) システム構造の同定

解釈データよりシステム構造を同定することが目的の場合，(2)における構成要素レベルでのモデルとデータとの対応づけをベースにして，システム全体として整合性のある構造を構成的に同定するタスクが要請される．

(4) システム状態の同定

解釈データよりシステム状態を同定することが目的の場合，予め知られているシステム構造と構成要素特性をベースとして，観測データを矛盾なく説明するための診断的タスクが要請される．

(5) あいまいさの処理

(2)，(3)および(4)の過程には，照合の不完全性に起因するあいまい性および多義性が必ず伴う．あいまいさや多義性を回避するためには，種々のソースから取得した情報を加重，評価，統合するタスクが要請される．

(6) 不完全性の処理

情報の欠落に伴う不完全性のために処理が不可能な部分については，デフォルト推論などによるデフォルト値の利用あるいはこれに代る方法を導入して，情報の欠落をカバーするタスクが要請される．

(7) 解釈の多様性

システムの構造や状態の推定において，解釈が一意に定まるのは，むしろ例外的であり，最終段階においても多様な解釈が存在するのが普通である．多様な解釈の中から最適な解釈を選択するためのタスクが要請される．

3.2.3 解釈型問題における問題解決機能

3.2.1 および 3.2.2 の議論に従えば，解釈型知識システムがもつべき問題解決機能はつぎのようにまとめられる．

(1) 分類階層によるクラス分け

解釈型問題は，対象から得られた特徴データを分析して，対象がどのクラスに属するかを同定することが基本である．そのためには，対象の特徴に基づく分類階層の構成と分類階層に基づくクラス分けをモデル化できる機能が要請される．

(2) モデル検索機能

解釈の対象の構成要素およびシステムレベルでの構造的および機能的モデルをライブラリーとして登録し，検索要求に対して効果的かつ効率的にパターン照合を行い，もっとも

可能性の高い候補を選択的に返す機能が要請される．

(3) 部分構造の評価機能

システムの任意の部分的構造が与えられ，他の部分が未知である状況において，この部分的構造がシステム全体の特性，挙動にどのような影響を与えるかの予測を与える評価機能が要請される．

(4) 階層的生成検査法

核となる部分的構造を出発点として，構造を階層的に表現し，部分構造を段階的に生成し，その妥当性を評価しながら，最適なシステム構造を見出すために，階層的生成検査法の機能が要請される．この戦略の有用性は，上位レベルでの部分構造に対する高精度の評価に基づく強力な枝刈りがシステムの性能に依存する．

(5) 不確実性推論

データに不規則なノイズが存在したり，誤りが存在する状況のもとでは，データのソースを多重化して，冗長性をもたせることを必要とする．すなわち，複数の根拠に基づいて結果の信頼性を高める方法がとられる．これらの情報を統合化するうえで，不確実性推論の機能が要請される．

(6) 仮説推論

データの欠落に対処するには，デフォルト的な値を利用するためのデフォルト推論の利用が不可欠である．デフォルト推論よりもっと一般的な枠組みとして仮説推論の機能が要請される．仮説推論においては，仮説およびこれに基づく推論の依存関係を系統的に管理し，必要に応じて，盲目的ではなく，知的後戻りの機能が提供される．

(7) 協調型推論

解釈における多様性を克服するためには，知識システムに複数の異なった視点をもたせた自律的な知識源を多数配し，各知識源に自律的，局所的な問題解決を行わせながら，必要に応じて協調をとり，全体として信頼性の高い問題解決に結びつける協調型推論の機能が要請される．

3.2.4 解釈型問題へのアプローチ

(1) 知識工学的接近の枠組み

解釈型問題に対する知識工学的接近の基本は，深いモデルに基づく推論と浅いモデルに基づく推論の統合にあるといえる．深いモデルとは，対象とするシステムの構造と機能の

理解に基づいてつくられる知識ベースであり，浅いモデルは経験的知識に基づいてつくられる知識ベースである．深いモデルは，その完全性ゆえに，高い信頼性を保証するが，必ずしも効率的とはいえない．一方，浅いモデルは経験に基づくだけに，しばしば効率的であり得るが，未知状況に対応できない欠点をもつ．したがって，工学的な観点からは，深いモデルと浅いモデルを結合して，信頼性と効率性のバランスを図ることが重要である．

解釈問題が対象とするアナログデータには，不規則なノイズの混入やデータの欠落がある場合が多く，3.2.3 節で述べたように，不確実性推論やデフォルト推論が要請される．

解釈型問題に対するより一般的な接近法としては，複数の異なった視点をもつ知識源が自律的に局所的な問題解決を行いながら，必要に応じて協調をとり，全体として信頼性の高い問題解決に結びつける協調型推論の枠組みが有用である．協調型推論では，知識源間の通信負荷を増大させることなく，問題解決の大局的一貫性を確保することが重要である．

(2) 解釈型知識システムの事例

解釈型問題の先駆的システムとして，有機化合物構造同定システム DENDRAL [Buchanan 78] が知られている．DENDRAL は質量分析器から得られたスペクトルデータを解析して，構造の同定を行うもので，階層的生成検査法を基本的問題解決戦略として採用している．

地質探査の支援システムとして，Dipmeter Advisor [Smith 83] が知られている．dipmeter による地層傾斜データのパターンを解釈するために，専門家の経験則が黒板モデルとして組織化されているが，解析はボトムアップ式に前向き推論によって行われる．

音声理解システム HEARSAY-II [Erman 80] は協調型問題解決のモデルである黒板モデルを提案したことで有名である．HEARSAY-II の知識源は12個存在し，ボトムアップ的な解析とトップダウン的な解析を同時に行うことにより，協調的な問題解決を実現している．

画像理解システム ACRONYM [Brooks 81] は，飛行場の航空写真を対象として，画像から抽出されたリボンと呼ばれる軸対称四角形と一般化円筒モデルとの間での照合および画像の撮影条件に関する情報を利用して，整合性のチェックを行い，画像の大局的な解釈を与えている．

自律移動ロボットNAVLAB [Harmon 86] は，相互接続された４台のオンボードコンピュータをもち，それぞれ距離検出器のデータ解釈，ＴＶカメラのデータ解釈，ソナーのデータ解釈と操縦および地図の作成と計画の作成を担当し，協調型問題解決を実現している．

(3) 知識工学的接近の課題

信号理解への知識工学的接近の課題をまとめておく．

第１は，知識表現の問題である．現在までに確立されている知識表現はルール形式とフレーム形式であり，いずれも記号レベルでの表現を前提としいる．しかし解釈型知識システムが取り扱う情報は，多次元のアナログデータであり，これをいかに推論において利用可能なパターン情報として形式化するかが知識獲得における主たるタスクとなる．領域専門家といえども，パターン情報を感覚的にとらえているため，必ずしも明確な概念を形成しているとは限らず，知識技術者がこれらを形式化する必要がある．

第２は，認知情報処理の問題である．現在の音声理解システムや画像理解システムは，人間の聴覚プロセスや視覚プロセスを陽にモデル化するものではない．対象の構造や機能に関する解析的な知識を利用するだけでは高次の理解システムを構築することはできない．人間が生得的にもつ認知情報処理を再現するためには，現在の知識処理技術では限界がある．したがって対象をうまく限定しないと，実用システムの構築は困難であることを承知しておく必要がある．

認知情報処理を自然に実現する技術として，最近，コネクショニストモデルの研究が注目されており，工学的な応用への期待が高まっている．知識システムが人為的なルールベースによる推論(Rule-Based Reasoning)に基づいているのに対し，コネクショニストモデルは事例ベースによる推論(Case-Based Reasoning)に基づいており，より人間に近い自然な方法と思われる．

人間の認知過程が主要な役割を果たすデータ解釈については，知識工学的接近には自ずと限界がある．音声理解はその典型と思われる．事実，コネクショニストモデルに基づく音声理解システムが試作され，知識システムと同等以上の性能を示したとの報告もある．

コネクショニストモデルに基づく解釈システムを構築する研究はこれから進展すると思われるが，学習のための適切な学習アルゴリズムの開発および適切な訓練データの与えかたが重要な課題とされよう．

（小 林 重 信）

［参考文献］

[小林 87] 小林重信：信号理解における知識処理，システムと制御，Vol.31，No.5，1987.

[Buchanan 78] Buchanan, B.G. and Feigenbaum, E.A.: DENDRAL and Meta-DENDRAL: Their Applications dimension, Artificial Intelligence, vol.11, pp.5, 1978.

[Smith 83] Smith, R.G and Baker, J.D.: The Dipmeter Advisor System, IJCAI-83, pp. 122, 1983.

[Erman 80] Erman, L.D. et al: The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty, Computing Surveys, vol.12, pp.213, 1980.

[Brooks 81] Brooks ,R.A.: Model-Based Computer Vision, UMI Research Press, 1981.

[Harmon 86] Harmon, S.Y. et al.: Sensor Data Fusion through a Distributed Blackboard, IEEE Int. Conf. on Robotics and Automation, pp. 1449, 1986.

3．3　診断型知識システム

診断システムの機能は、対象とするシステムに異常または故障が生じたとき、観測されたデータおよびシステムに関する知識を利用して、原因の箇所を同定することである。診断は、ＭＹＣＩＮに見られるようにエキスパートシステムの初期の時代から知識システムの主要な適用分野の一つである。また、診断型知識システムの基本タスクは制御問題にも関連を持っている。

3．3．1　診断型問題の特徴

(1) 設計レベルの知識の利用

プラントや機械設備などのエンジニアリングシステムでは、設計レベルでの知識を利用することが可能である。

浅いモデルに基づく診断型システムにおいては、専門家から得られる経験的知識を利用するが、その診断知識の多くは設計仕様書などを基に整理・抽出される。即ち、設計レベルの知識が経験的知識の中に埋めこまれていると言えよう。

また最近では、構成要素、接続関係、機能といった因果関係に関する深いモデルの知識を推論機構が直接利用する研究が精力的に進められている[Davis 84]。このような診断型システムは、診断の完全性、未知の状況への対応、適用範囲の限界の判断など、従来の浅いモデルに基づくシステムにはない特長を持っている。

実際に深いモデルを利用したシステムとしては、電子回路の診断[Pipitone 84]などいくつか報告されている。また、診断専用シェルのひとつであるＩＮ－ＡＴＥは、各要素間の接続関係と統計的な故障頻度および計測に要するコストなどから、最適な計測手順を選択する機能を持っている。

(2) 経験的知識の利用

診断問題の専門家は、それぞれの故障原因についての特有な症状、統計的な故障頻度、適切な計測の順序などといった経験的知識を持っている。生体や環境など自然システムを対象とする場合や、エンジニアリングシステムでもその挙動が解析的に不明な場合には、人間が経験によって獲得した表層的な因果関係知識に頼らざるを得ない。

また、設計レベルでの知識が利用可能である場合でも、経験的知識により推論過程を省略して、異常や故障が生じた箇所近傍へ効率的にアクセスできることが期待できる。ただし、理論的根拠が無いために、未知の状況に対しては無力化する恐れがある。

(3) 経験的知識の不確実性

経験的知識は対象システムの構造から導かれる因果関係に基づくものではないため、それから得られる結論には不確実さが伴う。ＭＹＣＩＮにおける不確実性の取扱いを契機にして、種々の理論や方法が開発され応用されている。

診断型問題では、原因を効率良く絞りこむことや、専門家の経験的知識の整理を容易にすること等に不確実性の導入が大きな効果をもたらす。ただし、専門家の知識に適合した確実度の割当て方や、妥当な結論が得られるような合成の仕方については充分検討する必要がある。特に知識ベースの規模が大きくなった場合には、局所的に定めた確実度が最終的な結論にどのように影響するかを見通すことが困難になり、保守性の低下をまねく恐れがある。

(4) 操業データの利用

プラントや機械設備などの故障診断では、操業データが次のように利用される。

- 過去の故障履歴から故障の傾向などに関する知識を得る
- 実際に故障が発生した場合に診断データの一部として利用する
- 操業データをモニタすることにより故障の予知を行なう

(5) 計測のための時間とコスト

故障現象から原因の箇所を同定する過程では、それまでに得られている情報をもとにシステムを計測して、より詳細なデータを収集する。しかし計測を行なうためには、時間とコストがかる。このため、診断型システムでは計測によって得られる情報とのトレードオフを考慮して質問（計測点指示）応答の制御を行なう必要がある。

(6) 知識の抽象化の必要性

設計レベルの知識を扱う際に、システムの詳細な状態を全て対象とすると知識が膨大なものとなる。これを回避するためには、適切なレベルで機能の抽象化を行ないシステムの状態を階層化することが必要となる。

(7) 対話的診断の推論制御

診断型システムの開発にあたっては、利用者とシステムとの対話性が実用上大きな影響を与えることを考慮すべきである。このため、利用範囲、専門的レベル等に関するユーザモデルを明確に定めて、適切な推論制御を行なう必要がある。特に、冗長な質問を抑止して効率良く絞りこみを行なう機能や、必要に応じて利用者主導で探索の方向を決定する機能などが重要である。

3．3．2　診断型問題における基本タスク

(1) 異常事象の分類階層的表現

多くの診断型問題において、異常事象あるいは故障現象を分類階層的に表現することができる。各ノードは抽象的な故障階層に対応する。推論時にはその階層を辿ることにより故障原因を絞りこむ。

この表現は、領域専門家にとって理解しやすく、また、構造を持たない表現に比べて知識ベースの大規模化への対応が容易である。

さらに保守性の高いアーキテクチャとして、推論制御の情報も各ノードに分散させた、モジュール性の高いモデルが提案されている[Bylander 86]。

(2) システム構造の階層的表現

診断を行なう際に、対象としているシステムの構造は重要な情報である。深いモデルによる推論を行なう場合はこの表現を直接利用して推論が行なわれる。大規模なシステムでは因果関係に関する知識が膨大になるため、抽象化による階層的表現が要請される。

また、浅いモデルによる推論の場合でも、対象システムの構造を利用することにより、知識表現の柔軟性を高め記述量を減らすことが可能である。

(3) 解釈問題のタスクを内包

診断型システムには、入力された故障状況のデータを推論に適した形式に解釈する機能が要求される。特にオンラインでデータを取り込む場合には、データの特徴抽出、モデルとの照合、システム状態の推定、あいまいさの処理、不完全性の処理といった解釈問題と共通の基本タスクを考慮する必要がある。

(4) 計測点の選択的決定

故障原因を追及する過程では、その時点で得られている情報から、絞りこみのための効率のよい計測点を順次選択的に決定できることが望ましい。すなわち、対象システムの構造に関する知識を利用し、計測結果から得られる絞りこみの度合と、計測に要する時間やコストとを考慮して最適な計測戦略を動的に立てる機能が必要である。

(5) 異常原因の同定

故障現象から推論をすすめて原因を同定する。複数の故障が同時に起こりうるとするかどうか、また、対象システムのシミュレーションが可能かどうかなどにより、アプローチが異なる。異常原因の同定は、より広く捉えれば異常現象の最も確からしい説明を推論することとも考えられる。

(6) 浅いモデルによる効率性

大規模な診断型システムでは、推論の効率性を高めることが重要である。このため経験的な知識（浅いモデル）を利用する。診断型システムの基本タスクとして浅いモデルを考える場合には、対象問題の特性および他の基本タスクとの統合を考慮して適切な知識構造と推論機構を構築する必要がある。

(7) 深いモデルによる完全性

対象システムの設計レベルの知識（深いモデル）を利用すれば、因果関係から完全な診断を行なうことが出来る。ただし、現状では電子回路など限られた領域でいくつかのシステムが開発されているに過ぎない。基本タスクとして確立されるためには、定性的推論などの分野における課題の解決が必要である。

3．3．3　診断型問題における問題解決機能

(1) 事象駆動型推論

診断問題においては入力された故障事象に対応して故障原因を推論するために事象駆動型推論が用いられる。操業監視システムが、オンラインで操業データを受け取り異常を監視することなどが典型的な事象駆動型推論といえよう。通常の対話的な診断型システムでは、ユーザ主導で非同期に入力可能とする機能や、複数の故障が同時に起こっている場合にも対応する機能などの実現に事象駆動推論が有用である。

(2) 目標駆動型推論

想定される原因から、それを検証するための推論を行なう方式であり、関連する項目のみを質問するため効率がよい。実例としてはＭＹＣＩＮが著名である。(1)の事象駆動型と組み合わせることにより、初期入力から事象駆動で原因の候補を絞り、それから目標駆動で故障原因を同定するといった推論方法も実現されている。

(3) 不確実性推論

故障事象と故障原因との結び付きが確定的であることはむしろ例外であり、大部分の診断問題では不確実な知識を扱う必要がある。最も広く用いられているのはＭＹＣＩＮで導入された確信度の考え方である。そのほかDempster-Shaferの方法やファジイ理論に基づく方法などが利用されている。これらの方法はそれぞれ特徴を持っているが、導かれる結論の妥当性については注意が必要である。特に大規模なシステムの場合には、単純な戦略を機械的に適用すると思わぬ振舞を示すことがある。このため、不確実性推論を局所的に

適用し、その部分的な結論をさらに上の階層で統合化するといったきめ細かな制御が必要である。

(4) 仮説推論

仮説推論は非単調推論の一種で、"生成－検証"パラダイムに基づく探索に用いられる。診断型問題では、例えば対象システムの要素が持つ様々な状態を仮説とし、因果関係を利用して各仮説間の依存関係を定めることにより、仮説推論の手法を応用することが可能になる。診断の過程では、仮説に基づいて探索をすすめ、原因を見出せなかった場合には後戻を行なう。このときどのような仮定をしたかを依存関係を利用して記録しておくことにより、後戻を知的に制御することが可能となる。

(5) 協調型推論

対象システムが大規模な場合、単一の知識ベースで記述すると知識が膨大かつ複雑になり、保守性・拡張性に問題が生ずる。このため、適切な機能の単位を定めて、それに対応した、独立の制御機構とデータ構造を持つ知識源による、協調型推論のモデルが要請される。

協調型推論には、入力のノイズに対する頑健性など様々な利点が期待されるが、統合的な制御の方式には解決すべき課題が残されている。

(6) 効率性と完全性の調和

浅いモデルに基づく診断は、少ない入力情報から効率良く結論にいたることができる半面、知識を整理する際に想定しなかった事態に対しては無力化してしまう危険性が高い。一方、深いモデルに基づく診断はモデルで表現可能な全ての場合をカバーすることができるが推論に多くの負荷がかかる。

実用性の高いシステムにするためにはこのふたつを統合化したバランスのよい推論機構が必要である。

(7) 費用／効果分析

診断を進めるためには、計測を行なって、より詳しい故障状況の情報を入力することが必要である。診断型システムの重要な機能のひとつは、計測に要する費用と、その計測により故障原因を絞りこめる度合を考慮することにより、最適な計測点を決定することである。このためには、対象システムの構造や統計的な故障頻度などの知識を用いた費用／効果分析の機能が必要である。

3．3．4　診断型問題へのアプローチ

診断に対しては、ＦＴＡ（故障木解析）やＦＭＥＡ（故障モード影響解析）などのアプローチがある。しかし、これらは対象システムの解析を主な目的とした手法であり、実際の故障発生時の対処に適したものではない。

知識システムの手法は柔軟性や保守性に優れているため、初期の段階から診断問題への応用が精力的に行なわれてきた。ここではそれらのシステムの例をあげて、アプローチを検討する。

専門家の診断の知識を利用した診断型システムの例としては次のようなものがある。

ＭＹＣＩＮ

伝染性疾患の治療法について医師に助言をするシステム。診断型システムの基本的な考え方の中で次のようなものを実現した。

(a)知識ベースの手法

(b)プロダクションルールによる知識表現

(c)ｃｆ値の利用による不確実性の扱い

(d)説明機能

(e)知識獲得支援の試み(ＴＥＩＲＥＳＩＡＳ)

ＣＳＲＬ(Conceptual Structures Representation Language)[Bylander 86]

分類問題を基本タスクとして捉え、問題のモデル化に適した知識表現形式と問題解決機能をもつ言語。主な特徴は次のとおり。

(a)異常事象の分類階層による知識表現

(b)分散化されたアーキテクチャ

(c)メッセージパスによる問題解決

また、システム設計レベルでの知識（深いモデル）に基づく診断型システムの例としては、電子回路などを対象とする次のシェルがある。

ＩＮ－ＡＴＥ(Intelligent Automatic Test Equipment)[Cantone 87]

システムの接続関係を利用する診断専用シェルで次のような機能を持っている。

(a)対象システムの接続関係を階層的に表現し推論を行なう

(b)故障頻度と計測コストから最適な計測手順を指示する

(c)経験的なルールも混在可能

知識システム開発にあたっては、問題の性質に応じた適切なアーキテクチャを選択することが重要である。診断型の場合のポイントとしては

・深いモデルと浅いモデルの組み合わせ

・不確実性の取扱い

・大規模知識ベースの保守

・説明機能

といったことがあげられる。

3．3 の参考文献

（山崎　知彦）

[Davis 84] R. Davis, Diagnostic Reasoning Based on Structure and Behavior, Artificial Intelligence 24, 347-410(1984).

[Pipitone 84] F. Pipitone, An Expert System for Electronics Troubleshooting Based on Function and Connectivity, 1st Conference on Artificial Intelligence Applications, 133-138(1984).

[Bylander 86] T. Bylander & S. Mittal, CSRL: A Language for Classificatory Problem Solving and Uncertainty Handling, AI Magazine 7(3), 66-77 (1986)

[Cantone 87] R. Cantone, Diagnostic Reasoning With IN-ATE™, Proc. A.I. '87 Conf., (1987).

３．４　制御型知識システム

３．４．１　制御型問題の特徴

一般に「制御（ｃｏｎｔｒｏｌ）」と言っても、マイナーなフィードバック制御からシーケンス制御、プロセスの管理・監視まで多岐に渡る。ここでは、制御問題を次のように広義にとらえて議論を進める。

制御問題の定義・・・システムの状態を監視していて、あらかじめ予定されたとおりにシステムの状態が遷移するように操作を加えることである。

制御問題の解決に知識システムを適用する場合も一般の制御系設計と同様に次の条件を満たす事が要求される。

①制御対象の時定数に応じた制御の応答性の確保。

②必要な制御精度の確保。

③系の安定性の確保。

こうした条件の実現のためにはまず数学的に証明された制御理論の適用を検討すべきであり、その適用が難しい制御対象に知識システムの適用が求められる。従って知識システムのアプローチを行おうとする場合には従来技術によるアプローチの問題点とその限界を十分洗い出して、知識システム適用の目的を明確にすることが重要であり、システムの成否を決める主要な鍵となる。

以下に知識システム適用が適している制御問題が持つ特徴、及び、制御問題が本質的に持つ特徴を列挙し、コメントを加える。

①時間遅れによる履歴依存性

②非線形性による局所性

③安定化の応答性

④制御精度の実現

⑤実プロセスとの接続

⑥実時間性と信頼性の確保

⑦離散系における問題解析

時間遅れが大きいシステムでは一般にフィードバック制御が困難である。このようなシステムでは現代制御理論に基づくフィードフォワード制御が可能であるが、数学的モデルの同定が必要条件であり、モデルの同定が困難な非線形システムに対しては無力である。このようなシステムに対して知識システムのアプローチが適している。

制御問題の場合、当項の冒頭に述べたとおり③，④は特徴というよりむしろ本質的な目的であり、どのようなアプローチを行うにしろ必要な「応答時間」と「制御精度」の確保がなされなければ制御の意味を果たさない。

制御問題以外の知識システムは、閉じた系又は他のデータベースとの若干のやりとりを伴う程度で成り立つ場合が多いが、制御問題の場合はプロセス状態を把握するためのセンサやプロセスをコントロールするためのアクチュエータとの接続が前提となる。実プロセスとの接続を行ったシステム全体での実時間性や信頼性の確保は、制御の目的を果たす上で重要である。現状技術レベルにおいては、知識システムの制御周期をミリ秒オーダーまで要求するのは難しい。また、知識システムによる制御においていかに信頼性を保障するかは、今後の大きな技術的課題である。

このように制御問題の場合見逃せない点は、既設システムとのインタフェイスが多いということである。知識工学の適用により制御規則部分のソフトウエア生産性が向上しても既設システムとのデータのやりとり、ソフトウェア作成により多くのマンパワーが必要なこともあるので、このような観点からの検討も必要である。

離散システムに対しては、従来シーケンス制御による接近がなされてきたが、制御理論を手続き的言語に変換して実行させる形態がとられてきたために、生産性、特に保守性が大きな課題とされていた。これには、ルールベース言語の導入により、保守性の向上が期待できる。

３．４．２　制御型問題における基本タスク

制御問題における知識システム適用の基本タスクを列挙し、コメントを加える。

①システム構造の表現

②システム動特性の表現

③診断問題の基本タスクを内包

④モデルによる状態の予測

⑤安定化動作の優先的実行

⑥安定操業下での省エネ化

⑦オペレータガイダンス

知識システムによる接近を行おうとする場合にも何らかのシステム構造及び、システムの動特性の表現を行う必要が出てくる。但し、システム構造を陽に定義するか暗黙の前提と

するかは、問題解決の方針・知識獲得の可能範囲・知識表現の方針等により異なる。

制御問題の基本タスクには「診断問題」の基本タスクが内包されている。

診断問題・・・①異常事象の分類層的表現　②システム構造の階層的表現　③解釈問題のタスクを内包　④計測点の選択的決定　⑤異常原因の同定　⑥浅いモデルによる効率性　⑦深いモデルによる完全性

このうち、②の「システム構造の階層的表現」は、制御問題でもよく行われる有効な手段である。また、⑥の「浅いモデルによる効率性」と⑦の「深いモデルによる完全性」は制御問題においても互いにトレードオフの関係にあるため、問題解決の方針によってそれぞれの兼合いを十分検討すべきである。例えば、システム構造を物理・化学的原理まで立ち入って厳密なモデリングを行えば、制御の完全性は増してくるが、モデリングは一般に容易ではない。逆に経験に基づいた制御則を列挙すれば知識表現は容易となるが、制御の完全性は証明しにくい。

また、診断問題の基本タスクには「解釈問題」の基本タスクが内包されている。

解釈問題・・・①特徴の抽出　②モデルとの不完全な照合　③システム構造の同定　④システム状態の推定　⑤あいまいさの処理　⑥不完全性の処理　⑦解釈の多様性

このうち、⑤の「あいまいさの処理」は、システム状態を把握するためのセンサ情報の把握・制御方針の決定の過程においてしばしば発生する問題であり、必ずしも数学的に証明されていなくても対象問題に適した簡易でわかりやすい方法が求められるケースが多い。

制御問題において、安定化操作の優先実行についてフェイルセーフの観点から考慮しておかなければならない。例えば、「安定化の戦略」と「やや不安定要素はあるが効率を重んずる戦略」とをグレード分けし、メタ知識で安定化の戦略を優先的に実行するという方法もある。

閉ループ的な制御方針をとる場合でも、グローバルな学習機能を人間に頼らざらるをえない現状技術レベルにおいては、オペレータに対するガイダンス機能は重要である。そのためにも、知識システムの機能の１つとしてあげられる「説明機能」を充実させ、制御対象に応じたリアルタイム性をもって実現することが必要である。

３．４．３　制御型問題における問題解決機能

制御型問題における問題解決機能には次のような項目があげられる。但し、制御問題に

おける知識システムは事例が少ないこともあって、有効な問題解決機能はまだ十分に洗練されていない。以下に考えうる問題解決機能を列挙する。

①状態の診断機能

②シミュレータによる予測

③定性推論による予測

④制御則の多目的評価

⑤アクションガイダンス

⑥デッドロックの解析

⑦インターロックの回避

大規模な人工システムでは、系の挙動を知るには大規模なシュミレーションを実行する必要がある。ここに定性推論の工学的ニーズが存在する。定性推論では、対象システムを記述するためのパラメータは定性変数として記述され、変数間の関係は定性的微分方程式として記述される。この定性的微分方程式に基づいてシステムの局所的動特性を与える手続きをルール形式で記述する。これらの知識を用いて、システムの大局的な動特性をうまく表わせば、定性推論の有効活用が図れると考えられる。この具体的な利用方法は今後の課題の一つである。

３．４．４　制御型問題へのアプローチ

(1) 対象問題の選定

３．４．１項でも若干述べたとおり、このフェーズでは従来技術によるアプローチの限界と問題点を十分洗い出して、知識システムの適用の目的を明確化することが重要である例えば、制御対象の挙動を明確に数学的モデルで記述できるのに、知識システムの適用を図ろうとするのは得策ではない。

また、制御の目的が明示的な制御精度の確保や安定性の確保である場合、知識システムでそれを実現する事は現状技術水準では難しい。それに対し、目標が不明確な中でのグローバルな目的の達成といったいわゆる悪構造問題では知識システムによるアプローチが期待されている。このため、必要に応じて知識システムと制御理論とを使い分け、場合によっては階層的融合を行うのが良いと思われる。例えば、高度の信頼性を要求されるマイナーな制御部分には従来の制御理論を適用し、大局的な方針決定部分には知識システムを適用するというのも一つの方法である。

(2) 知識獲得

制御問題へのアプローチにおける”知識獲得”は言いかえれば制御規則の整理であり、グローバルには仕事の手順・考え方の整理である。具体的な知識獲得の手法は次の２つの型に分けられる。

①領域専門家のモデリング

模範となりうる領域専門家が存在し知識獲得のための十分な協力と時間の確保が可能であるならば、インタビューやディスカッションにより知識獲得を行ってゆくのが良いこの知識獲得の過程おいては、作業標準書、業務日誌、事故報告書等のドキュメントも大いに有効活用すべきである。制御対象が比較的大規模で有効なプロセスモデルの存在しない監視・制御型問題には特にこの方法が有効である。但し、領域専門家と円滑なインタビューやディスカッションを行ってゆくためには、知識工学者側にも領域問題の前提知識と深い理解能力が要求される。

②シミュレーションによる制御規則の獲得

制御対象が比較的小規模で、ある程度の精度をもったプロセスシミュレーションモデルが利用可能な場合は、シミュレーションにより制御知識を獲得してゆく手法が有効であろう。挙動の比較的明確な制御問題にはこうしたアプローチが良い。

この知識獲得の段階で有効なＡＩツールの開発が今後期待されている。ＡＩツールが制御問題の実現に有効となるためには、ＡＩツール自体に本項にあげたような基本タスク・問題解決機能を兼備えていなければならない。

(3) システム環境の選択（ＡＩツールの選定または構築），プログラミング

ツールの選定条件としては他分野と同様、①獲得した知識をスムーズに表現できること②システムのユーザーのレベルに応じたマンマシンインターフェイスが整っていることをあげることができる。

制御問題へのアプローチでは前述のとおり既存プロセスのとの入出力インタフェイスを考慮しておかなければならない。特に既存システムが存在し、エキスパートシステムをこれに付加するような場合には、全体システムとしてのバランスを考慮してシステム環境の選択・設計を行う。

プロトタイピングレベルの場合は、これらの条件をそれほど厳密に考える必要はないが実用化時の見通しは十分考えておく必要がある。

(4) 制御問題へのアプローチの若干の事例

高炉のように無駄時間が大きく非線形な系においては、知識システムによるアプローチが有効と考えられ、いくつかの実用事例も発表されている。規模が大きく複雑で、非線形な系であっても、大局的な制御則を加えればあとは制御対象自体の自律性にある程度期待できるので、知識獲得の労力は緩和されている。

離散系のシステムにおいては従来手続き言語を用いたシーケンス制御によるアプローチが行われてきたが、ソフトウェアの生産性・保守性が課題とされてきた。これに対しルールベースの言語を導入し、生産性・保守性を向上させた事例が、鉄鋼業におけるビレット精製ヤードへの適用が発表されている。

ＦＡシステムは、離散システムの代表例であるが、一般に非同期同時進行システムとしてモデル化される。単純なルールベースシステムはこれらを表現するのに自然な枠組みとは言いがたく、ペトリネットや並列オブジェクト指向モデルといった枠組みが期待を集めている。こうした枠組み適用を検討した事例としてはコイルヤードにおいて錯そうする複数のクレーンの制御やごみ焼却炉の制御が発表されている。

（担当　新日本製鐵㈱　湯井勝彦）

（参考文献）

[小林　８７] 小林，”多目的意思決定 ― 理論と応用 ― Ⅳ ― 知識工学方法論 ― ” システムと制御　３１巻４号，２７５－２８５（１９８７）

[船橋　８７] 船橋、増位，”制御分野におけるエキスパートシステム” 情報処理　２８巻２号，１９７－２０５（１９８７）

[石塚　８３] 石塚，”不確かな知識の取扱い” 計測と制御，２２巻９号，７７４－７７９（１９８３）

[de Kleer.J. 86] de Kleer.J. ,"An Assumption-based TMS" Artificial Intelligence Vol.２８，１２７－１６２（１９８６）

[山崎　８４] 山崎、菅野，”ファジィ制御” システムと制御，２８巻７号，１８－２２（１９８４）

[湯井　８６] 湯井、貝塚，”鉄鋼プロセスにおける制御理論応用と新しい制御システムへの展開” システムと制御，３０巻１号，７－　（１９８６）

[湯井　８７] 湯井、森谷，”鉄鋼業におけるエキスパートシステムの応用” 電気学会論文集Ｃ，１０７巻２号，１１５－１２０（１９８７）

[諏訪　８６] 諏訪、ほか，”エキスパートシステム開発事例にみる知識獲得の諸相” 計測と制御，２５巻９号，８０１－８０９（１９８６）

[湯井　８７] 湯井、ほか,"高炉プロセスの操業監視支援における知識システムの適用" 計測と制御，２６巻８号，７１２－７１９（１９８７）

[岩本　８６] 岩本、ほか，”エキスパートシステムの高炉操業への応用” 第２５回計測自動制御学会学術講演会予稿集，２１３－２１４（１９８６）

[都島　８５] 都島、ほか，”流れ作業ライン制御へのルール型制御方式の適用 ― 製鉄所のビレット精製ライン制御への適用 ― ” 計測自動制御学会論文誌，２１巻１０号，１１１３－１１２０（１９８５）

[小野　８６] 小野、ほか，”並列オブジェクト指向モデルによるクレーン群の動的スケジューリング” 計測自動制御学会第４回知識工学研究会資料，　９－１４（１９８６）

[阿瀬　８７] 阿瀬、ほか，”ごみ焼却炉運転支援エキスパートシステム” 人工知能学会全国大会（第１回）論文集，３３９－３４２（１９８７）

３．５　計画型知識システム

３．５．１　計画型問題の特徴

生産工程計画，旅客機運行計画，列車運行計画，資金運用計画，電力系統運用計画，プラント操業計画，機器の割り当て計画，実験計画等，××計画と名の付く計画型問題は種々ある．これらは，基本的に与えられた目標を達成するために，利用可能な資源（人，金，もの）の割り当てを時系列として決定する問題として捉えられ，計画型問題の固有の性質として次の特徴が挙げられている．[小林 87]

(1) 探索空間が非常に広い．

計画型問題は，基本的には，組み合わせ問題であり，組み合わせ的爆発が起こり易い．例えば，列車運行問題で，ダイヤを発着時刻の組み合わせと捉えると，ダイヤの作成過程における試行錯誤は，組み合わせ的爆発を生じる．

(2) 評価属性が多様である．

計画型問題は，資源（人，金，もの）の割り当てと，時に関する問題であり，経済的評価，時間的評価他，種々の属性で評価される．例えば，電力系統運用計画では，経済性，系の信頼性，安定性，余裕などの評価属性が用いられる．

(3) 環境予測が不確実である．

計画は，未来の手順決定に関する問題である．環境を予測し，その下で適した解を求めようとするが，環境予測自体不確実要素を含み，結果的に解の修正が必要となったり，複数解が必要となる．例えば，資金運用計画における金利の設定は，仮定であり，種々のケースを想定した複数解が必要となる．

(4) 部分計画間の相互作用．

(5) 既存計画の再利用．

計画の中には，全く新しい計画を立案しなくても，既存計画をそのまま，一部利用，一部修正で対応できるものも多い．

(6) 費用と便益のトレードオフ．

(7) 対話型計画作成支援の要求．

全ての制約条件や，評価基準を知識ベース化することは難しく，計画作成のノウハ

ウも利用した計画作成の要求がある．電力系統運用計画においても，計画作成は対話的支援の方式が採られている．

３．５．２　計画型問題の基本タスク

計画型問題では，その特徴を考慮して，問題分析を行う必要があるが，その基本的な観点は，基本タスクとしてまとめられている．[小林 87]

(1) 計画過程の階層化

計画型問題は，基本的に組み合わせ問題であり，組み合わせ的爆発が生じる．組み合わせ的爆発の問題を回避することが重要である．回避のために，問題を抽象レベルから具体レベルへと階層的分解をする．実験計画において，戦略空間，設計空間，計画空間と３つに階層化されている例もある．

(2) 組み合わせ的検索

(3) 制約条件の相互作用

計画型問題においては，制約条件は通常一つだけであることは少ない．複数の制約条件は独立でなく相互作用を及ぼす．制約条件の敏感度，相関度が，解の探索順序に影響する．電力系統運用計画では，経済性，安定性，信頼性の順で解の探索を進めている例がある．

(4) 環境の予測

(5) 上位レベルへの知的後戻り

探索制御において，探索失敗の時，網羅的探索であれば，自動的にバックトラックし，次の探索へと進むが，この時，探索失敗の情報を用いてバックトラックを制御することにより探索の効率化を図る．

(6) 計画事例の検索と利用

(7) 計画評価の多様性

３．５．３　計画型問題における問題解決機能

問題点を解決するための基本的な戦略として，次の問題解決機能が提示されている．[小林 87]

(1) トップダウン精密化戦略

問題を階層的に分割し，まず詳細な部分を無視し，抽象的なレベルで計画し，この決定を制約条件として，順次詳細レベルの計画を作成していく戦略である．例えば実験計画の戦略空間，設計空間，計画空間への分割はこの戦略であり，ロボットの計画作成システムABSTRIPS [Sacerdoti 76] は，このトップダウン精密化の考え方に基づいている．

(2) 拘束最小化戦略（least commitment）

トップダウン精密化がうまく働かない場合，上位レベルの強い決定が下位レベルの実行を阻害しないように上位レベルの決定をできる限り緩くする戦略を採る．

(3) 網羅的検索機能

網羅的検索機能は，検索時間は掛かるが，制御は自動的であるので，状態空間が複雑でない問題には有効な手段である．

(4) 仮説推論による知的検索

計画型問題は，基本的に生成－検査のパラダイムに基づく探索を必要とするので，仮説推論は，有効な手法である．

(5) 類推による計画事例の利用

(6) 計画の多目的評価

(7) リスク下での意志決定

３．５．４　計画型問題へのアプローチ

(1) 状態空間表現に適した計画型問題

８－パズルや農夫のジレンマ問題は，初期状態から目標状態までの手順を求める計画型問題である．これらの状態空間（一つの問題について到達可能なすべての状態からなる集合）は，作用素（ある状態から他の状態へ変化させる手段）が限られているため，複雑な

構造はしていない．例えば，農夫のジレンマ問題は，農夫（H）が，オオカミ（W），キツネ（F），キャベツ（C）を船で川の向こう岸へ運ぶ手順を求める問題である．言い換えると，Xは何もいないことを表すとすると，状態（H，W，F，C；X）から状態（X；H，W，F，C）への遷移手順を計画する問題である．今，途中の状態（C；H，W，F）を取り上げてみる（これは，川の手前岸にはキャベツが残り，向こう岸には船と，農夫，オオカミ，キツネがいる状態である）と，これから遷移の可能性は３通りである．

（i） 農夫のみが手前岸に戻る． （C；H，W，F）→（H，C；W，F）

（ii） 農夫がオオカミを連れて手前岸に戻る． （C；H，W，F）→（H，W，C；F）

（iii） 農夫がキツネを連れて手前岸に戻る． （C；H，W，F）→（H，F，C；W）

この遷移状態を調べると（i)の（C；H，W，F）はオオカミがキツネを食べる状況になり，制約条件に抵触し，棄却される．又，(ii)，(iii）のいずれかは元の（C；H，W，F）の一代前の状態と一致するので，遷移状態としては棄却される．この結果，(i)，(iii)のいずれか一方の状態が残り，次の探索へと進む．

この様に，このタイプの計画型問題は，網羅的検索や，生成検査法のアプローチが適用できる．

(2) 問題分割表現に適した計画型問題

問題分割手法は，問題記述が与えられると，問題を順次分割し，最終的にすぐ解ける副問題の集合に変換していく．ハノイの塔パズルは問題分割表現がうまくあてはまる例として有名である．これは，簡単な例で示すと，柱１，２，３と円板A，B，C（直径の小さい順）があり，柱１の上からA，B，Cの順にある円板を，柱３に上からA，B，Cの順になる様に移す手順を求める問題である．この問題は，次の３つの問題に置き換え，

① 高さｎ－１の山を柱ｉから柱ｊに移す

② 高さ１の山を柱ｉから柱ｋに移す

③ 高さｎ－１の山を柱ｊから柱ｋに移す

これを順次副問題として展開し，解を求める．[Cohen 82] 問題分割表現は，状態空間表現と密接な関係にあるが，問題により表現の適・不適がある．このタイプの問題は，与えられた問題を分割可能な問題に置き換えられるかどうかがポイントとなる．

(3) 数理計画タイプ

このタイプは数理モデルがあり，従来，その最適解（準最適解）を線型計画法等で求めていた問題である．しかし，対象とする系が大規模になるにつれて，解を求めるのに長時間の計算が必要になってきたため，計画作業の効率化が要求されるようになった．この要求に答えるべく，エキスパートシステムの導入が図られ，次のような利点をもたらしている．

（i） 解を求めるプロセスが人間の思考過程に近い．

（ii） 知識を知識データベースとして，独立に扱える－－保守性，理解性がよい．

（iii）解の妥当性がわかり易い．

このタイプの代表的な例の一つとして，電力系統運用計画問題がある．このタイプの一つの問題解決法として，電力系統計画・解析支援システム［藤原 86］（4．5参照）のフローを示す．

① 経済的制約条件による解の探索

② 安定性条件を考慮して解の再探索

③ 信頼性条件を考慮して解の再々探索

(4) その他

配送計画問題：出発点からいくつかのステーションを廻って終着点に行く．時間的制約，処理量的制約，経済的制約等を満たすには，どのルートを選択すればよいかを決定する問題である．この場合，ルート選択の自由性は大きく，探索空間は，組み合わせ的爆発が起こる．更に，時間的制約においては，道路事情の予測不確実要素が入る．

列車運転計画問題：この問題は，特急，急行，各駅停車等種類の違う列車の，始発駅，通過駅，停車駅，待避駅，終着駅など種々の駅での発着時刻を計画する問題である．この問題は，ダイヤを発着時刻の組み合わせと捉えると，組み合わせ的爆発が起こり，探索空間が膨大なものとなってしまう．又，ダイヤを発着順序の組み合わせと捉えると，組み合わせ数は大幅に減少するが，発着時刻の決定に問題が残る問題である．これらのタイプの問題解決手法は，通常トップダウン精密化により階層分割を行う．列車運行計画（DRAFTS）［福森 86］では，発着時刻にある程度曖昧さを含ませて判断を進め，他列車との関係を考慮しながら曖昧さを除去して行く方法を，時刻幅という手法で実現している．

実験計画：分子遺伝学の実験計画（MOLGEN）［Stefik 80］では，実験計画を，戦略空間

（戦略を決定する），設計空間（制約条件から計画設計を行う），計画空間（実際の操作に具体化する）に階層化し，詳細化はその計画が後で捨てられないことを判断されるまで延期される最小拘束化方式を採っている．

（福島正俊）

参考文献

[小林 87] 小林，多目的意志決定－理論と応用－VI －知識工学方法論－，システムと制御 Vol.31 No.4, 275-285(1987)

[Sacerdoti 76] E. D. Sacerdoti, Planning in a Hierarchy of Abstraction:MYCIN (American Elsevier, 1976).

[Cohen 82] P. R. Cohen, E. A. Feigenbaum 編，田中，淵 監訳，人工知能ハンドブックⅠ～Ⅲ(共立出版，1984)

[藤原 86] 藤原，河野，電力系統計画・解析支援エキスパートシステム，システムと制御 Vol.30 No.6, 351-357(1986)

[福森 86] 福森，佐野，長谷川，坂井，人工知能の手法を導入した列車ダイヤ作成のための基本アルゴリズム，電子通信学会論文誌(D) Vol.J69-D No.4,569-579(1986)

[Stefik 80] M. J. Stefik, Planning with Constraints, Computer Science Dept. Rep. No.80-784(1980).

３．６　設計型知識システム

３．６．１　設計型問題の特徴

設計問題は次のように定義できる。

> ”設計問題とは，システムの入出力の要求仕様が与えられたとき，これを実現するために，構成要素を組み合わせ，かつ各構成要素の内部仕様を決定することである。”

設計問題を解析型問題と比較すると，解析型問題では，システムの構造とサブシステムの特性を推測することである。これに対し，設計問題等の合成型問題では，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ことが要求される。システムの構造とサブシステムの特性が与えられたとき，システムの特性は一意に定まるのに対し，システムの特性が与えられたとき，これを実現するようなシステムの構造とサブシステムの特性は一般に無限に存在する。このように，合成型問題はもともと組合せ的性質をもつことから解析型問題に比べてより難しいといえる。このためにシステム構造の解空間が非常に大きくなる。設計問題では，組み合わせ的爆発の問題を回避するために，問題解決の基本制御ループの設計が重要である。問題の分解統合および抽象化は必要不可欠なことである。しかしこれは，一般に極めて創造的な行為であり，このプロセスを支援するような方法論の開発が望まれている。

設計型問題の特徴をまとめると，次のようになる。

1．システム構造の解空間が大
2．解析による合成が基本的
3．構成要素の最適化の必要性
4．検証の完全性および効率性
5．対話形式での設計支援問題
6．設計段階に応じた支援形態

7．対話型設計支援の要求

設計型問題では，前もって最終的な設計結果の用意をしておくことはできない。設計型問題においては可能な構成要素あるいは部品の組合せは膨大な数になる。現実的には無限であると考えても見なしてもよい場合が多い。設計型問題では，このように、無限と言えるほど可能性が存在しているので，問題空間が非常に大きくなる。ただし，設計型問題はシステムをサブシステムに分解してゆくと、多くのものが選択問題として定義できるので、拘束の大きな問題の場合には解析型問題の積み重ねとして取り扱うことができる。すなわち，解析的な手法による合成というアプローチが可能である。この際，部分的な構成要素に大して，局所的な意味での最適性を定義し、局所最適解を求めつつ設計を進める。

設計問題においてシステム構造の決定の仕方については設計者の専門的知識あるいは経験的知識が利用される。システム構造の決定、サブシステムへの分解については、種々のやり方が可能であるが、設計結果としてのシステムはサブシステム分解のやり方によらず客観的に評価できる必要がある。例えば，機械やＶＬＳＩなどの設計においては，設計の設計結果のシステムは解析的に完全に検証できることがのぞまれ，またその効率性も要求される。

設計に用いる知識のすべてが明確に定式化されたものであり，与えられた条件から設計結果が明確な手続きを経て導かれるものであるならば問題は容易である。しかし，設計作業はそう簡単ではない。多くの設計で使われる知識は，定式化されたものもあれば，断片知識や，不完全な経験データも多く，非常にあいまいなものである。このような知識に基づいての判断はベテラン設計者に任せるしかない。そして，設計には，たとえ理論的にわかっていても実際にその理論による正確な計算や判断はできないため，経験知識による判断という便宜的方法をとるケースもよくある。このような場合も，専門家のノウハウによる判断に任せるしかない。このように，設計作業では，人間の直接介入が不可避である。したがって，その時点時点の設計段階に応じた対話形式での設計支援が必要である。

3．6．2　設計型問題における基本タスク

設計問題は，システムの特性が仕様として与えられたとき，これを実現するシステムの

構造および構成要素の特性を決定することである。一般に，設計問題では可能なシステム構造と構成要素の特性は無限に存在するために，設計活動は否定型的であり，類型化は難しいが，あえてあげるとすれば次のようなものである。

1．構成要素とその関連の表現
2．設計問題の階層的表現
3．代替案の自動生成
4．部分システムの評価
5．上位レベルへの知的後戻り
6．設計事例の検索と利用
7．並列的問題解決

設計問題の多くは，物理的な対象を扱うために，構成要素相互間の距離や形状など空間的な情報の取り扱いが必要であり，これを適切に表現しかつ操作し得る機械要素とその関連の表現方法の開発が望まれる。

対象となる設計システムは一般に大規模であり，設計に必要な知識は膨大なものとなる。これらの知識全体を同時に取り扱うことは事実上不可能となる。従って設計対象を抽象化し階層的表現が要求される。

設計の問題は探索問題として定義できる問題であり，基本的接近法は生成検査法のパラダイムに求めることができる。この方法を首尾よく成功させるために，可能部分システムの代替案を自動生成し，部分システムの評価をできるだけ高精度で行ない，かつ無駄な代替案をできるだけ早期に排除できるような評価ヒューリスチックスを規定する必要がある。

与えられた設計問題が＜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＞を見出すようなクラスの問題であるとすると，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。＜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。このようなクラスの設計型問題では，必然的に後戻りを必要とする試行錯誤的探索が不可避である。従って上位レベルへの“知的後戻り”といった処理が必要となる。

実際に設計が行なわれている現場の状況をを考えると，全く新規の開発設計の要求は少なく，日常業務の大半はすでに設計されている結果の修正的ないわば類似設計が非常に多い。そのために設計事例の検索と利用が重要である。このような類似設計のためのツールは実用上強い要望がある。

設計事例利用の重要な意味がもう一つある。それは設計事例が”実績”のある実際にうまく行った例であるということである。現実の設計を考えると，問題をモデル化し，その範囲での解を求めている訳であるが，実際には近似でしかあり得ない。例えば，電子回路をモデル化し，いかに綿密に回路解析を行なったとしても，実際の回路を忠実にシミュレートしているのではなく，あくまで近似である。現実の物理構造物の設計問題では多かれ少なかれ，未だ解明されていない未知の部分を含んでいる。そのために，既存の設計事例をもとに，多少の変更を加えて，所望の特性を得ようとするアプローチは有効である。

設計のための目的と満たすべき制約条件が与えられたとき，システム構造を決定し、サブシステム分解を行なう。この部分システム分解に見合った部分システムを生成する。一般にこの候補となる代替案は複数ある。１つの代替案を選択するために，それらを評価するが，代替案の生成・評価は並列的に行なうことが可能である。このような部分解の生成には並列的問題解決が要求される。

３．６．３　設計型問題における問題解決機能

設計型問題では，問題解決の基本戦略の設計が最も重要であるが，現在のＡＩツールは問題解決能力については何もサポートしているとはいえず，次世代ツールでは問題解決能力の付与が１つのポイントになる。下のような戦略あるいは機能が考えられる。

１．トップダウン精密化戦略
２．拘束最小化戦略
３．最適化機能
４．検証機能
５．仮説推論による知的探索
６．類推による設計事例の利用

7．協調型推論

構成要素は所与であるが，システム構造は未知である，というような設計問題を考える。構成要素のつながりをすべて同時に考慮することは組合せ的複雑さのために困難である。普通に取られる戦略は，構成要素の中で，空間的および時間的に重要と考えられるものをいくつか取り出し，他は考慮しないで，まずそれらの間の部分的システム構造を決定する。ついでこの部分的システム構造を制約条件として，次に重要と思われる構成要素群を考慮の中に入れ，それらの間のつながりを決定するとともに，すでにつくられた部分的システム構造の中に埋め込むことを行なう。このような過程はすべての構成要素が考慮されるまで再帰的に繰り返される。この戦略はトップダウン精密化（topdown refinement）と呼ばれる。

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

1)構成要素の集合をいくつかの部分集合に分割すること，

2)部分集合内での部分的システム構造の決定が可能であること，

3)上位階層での部分的システム構造の決定がそれより下位の問題に対し，制約条件として機能すること，

4)上位レベルでの決定が下位レベルでの実行可能性を侵害しないこと，

などが要請される。

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

このような設計型問題に対する問題解決戦略としてトップダウン精密化を採用すると，これを実現するための問題解決機能として，構成要素の階層化および部分構造の最適化機能（制約の伝播と実行可能性の評価を含む）などが必要とされる。

システム構造も構成要素特性も未知であるような設計問題を考える。このような問題では層的生成検査法やトップダウン精密化によるアプローチは困難である。これらの問題に

対してよくとられる方法は，過去の設計事例を参考にして，これに手を加えて要求仕様を満たすように改良することである。このプロセスをシステム化するためには，設計事例を知識ベース化することとともに，現在の設計作業に最も参考になる類似の設計事例を効率よく検索し，利用するための類比及び類推の機能をシステムにもたせる必要がある。

複雑な設計型問題では，目標が必ずしも明確ではなく，単純な最適化問題に分解できないのが通常である。このため異なった視点をもつ複数の知識源が互いに情報交換しながら，全体として整合性のある設計解へと収束させるようなメカニズムをもった分散協調型推論の枠組みが基本的となり，協調型問題解決機能が必須である。

３．６．４　設計型問題へのアプローチ

設計型問題には，ＬＳＩ設計，電子回路設計などの電気系設計，機械設計、プラント設計，構造物設計などの機械系設計をはじめ薬品設計など様々な様相の問題が含まれる。またスケジューリングなどの計画問題にも設計的な色彩がある。

設計型問題あるいは合成型問題に対するもっとも基本的な接近法として，生成検査法(Generate and Test)と呼ばれる手法がある。これはシステムの構造を仮に生成し，解析を行なってシステムの特性を求め，その評価を行なうことを繰り返して，最適なシステムを探索することを基本とする。従って生成検査法は解析による合成(Synthesis by Analysis)とも呼ばれる。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。

目的関数および制約条件に対する要求仕様の厳しさの違いによって，接近法は大きく異なる。制約条件が緩く，実行可能領域が広く，しかも目的関数に対する最適性がそれほど要求されないシステムでは，トップダウン精密化による後戻りなしの接近でも十分通用する。しかし反対に制約条件が厳しく，実行可能解を見出すことが非常に難しい場合には，最小拘束化のような堅実な接近法を採用せざるを得ない。

設計型問題へのアプローチは，上に述べた解析による合成、トップダウン精密化の比重の掛け方によって大きく２種類に分類することができる。一番目のアプローチは，合成に基づく設計に比重があるもので、ある一定の拘束条件を生成し、この拘束条件の下で，与えられた要求仕様を満足するようなシステムあるいは部分システムを合成するというアプローチである。例えば、与えられた要求仕様を満足するようなシステムを，あらかじめ用

意された利用可能な構成要素（部品）をうまく組み合わせて目的のシステムを合成していくというやり方である。プログラム部品を前提とした、ソフトウェア設計はこのようなアプローチの典型的な例である。またスタンダードセル方式とよばれるＶＬＳＩ設計では、あらかじめ用意された機能ブロック（スタンダードセル）を組み合わせて所与の仕様を実現する。

二番目のアプローチは，トップダウン精密化に比重があるもので、マクロな表現から出発し、詳細な表現へと展開してゆくことによって目的のシステム設計を行なうというやり方である。要求仕様を厳密な形で記述した仕様書が与えられて，一定の拘束条件を生成しながら具体的システム設計に展開し、詳細化していくという設計のやり方がこのアプローチである。ＶＬＳＩシステムの設計として，与えられた外部動作仕様記述から，内部動作記述、演算モジュールレベルの記述，論理回路レベル、トランジスタ回路レベル、レイアウトレベルへと，詳細に展開を行ないながら設計をするというアプローチは，この典型的な例である。

（森　啓）

[小林 87] 小林、“多目的意志決定－理論と応用－Ⅳ　－知識工学方法論－”
システムと制御、Vol.31,No.4,41-51(1987)

[千吉良 87] 千吉良、小林、“ソフトウェア再利用技術の動向”人工知能学会誌、
Vol.2,No. 3, 60-67(1987)

[長澤 87] 長澤、“設計エキスパートシステム”情報処理
Vol.28,No.2,187-191(1987)

4. 知識システムの開発事例

４．１　診断型システムの事例：自動車の故障診断システム－ＡＴＲＥＸ－

自動車は数万点の部品からなる複雑なシステムであり、その整備点検には整備士の持つ知識や経験が重要な役割を果たしている。熟練した整備士は、経験から得られた知識を利用して効率良くしかも適切な診断を行なうことができる。このような熟練者の技能を集約し標準化することは重要な課題である。

また故障診断技術にも高度化・自動化をはかる必要性が高まりつつある。これに対応するものとしては、整備工場に設置したコンピュータを利用して計測の自動化を行なう点検診断装置や、車両に自己診断機能を持たせるオンボード診断装置などが研究開発されている。しかし現在のところ利用条件やカバーする範囲などに制約が多い。

一方、車種の多様化やエレクトロニクス部品の導入に伴い、不具合を解決するためには、診断や設計レベルでの多くの知識をタイムリイに利用できることがますます重要となっている。

知識システムの手法は、故障木解析等の方法や、従来のコンピュータ利用技術に比べ柔軟性が高く、自動車故障診断の分野に有効であることが期待されている[岸 86]。ここでは、その一例として、自動車の故障診断システムのプロトタイプＡＴＲＥＸ(Automobile TRouble-shooting EXpert system)について紹介する[Takahashi 86]。

４．１．１　システムの目的と機能

本システムはサービス部門において、診断についての問合せに応えることを目的として開発されている。対象が自動車であることから、多様な車種や年式に対応できることが重要である。また、運用を含めて利用部門への技術移転を可能とするため、知識ベースの保守が利用者主体で行なえる保守性の高いシステムを目指している。

本プロトタイプでは、次のような機能を実現している。

（１）故障診断機能

(a)不具合状況から故障原因を絞りこむための点検指示

初期情報（車種、年式、型式、故障状況、点検した部品等）を入力すると、原因を絞りこむための点検手順を指示する。

(b)点検結果から原因を同定

点検結果(事象)から、その原因と考えられる部品あるいは部品系を推定する。

確信度の利用により、可能性のある複数の原因を扱うことができる。

(c)故障原因の検証

ある故障が生じているとき、それがどのような現象を引き起こすかを表示することにより、推定された故障原因の妥当性を確かめることができる。特に、二種類の故障原因から引き起こされる現象の中で共通でないものを表示する機能があり、原因の切分けに有効である。

(d)適切な修理手順のガイダンス

（２）マンマシン・インタフェース機能

(a)タッチパネルによる入力

全ての操作はタッチパネルにより行なわれるため、キーボードに不慣れなユーザでも容易に使うことができる。

(b)動画による表示

レーザディスクを利用して、ヘルプ画面や、点検・修理ガイダンスなどを動画で表示する。

（３）知識入力支援機能

知識整理の段階で、原因－結果を表の形でまとられる部分がある。この表から知識ベースのルール形式への変換を行なう。

以下に、本プロトタイプのセッション例を示す。

（１）故障状況を確信度を付けて入力

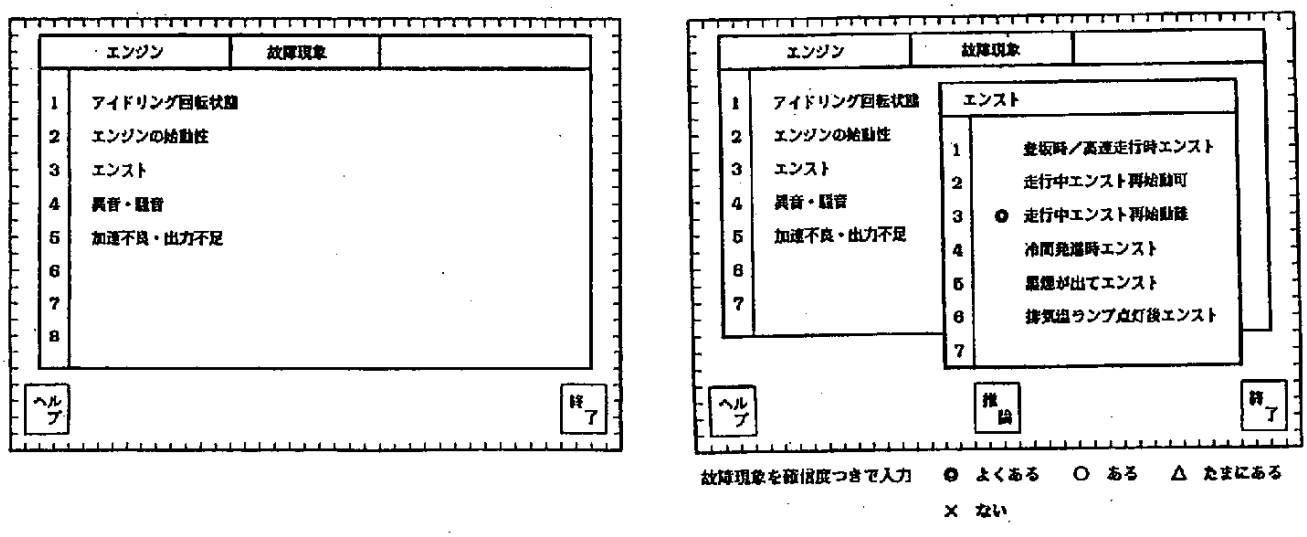
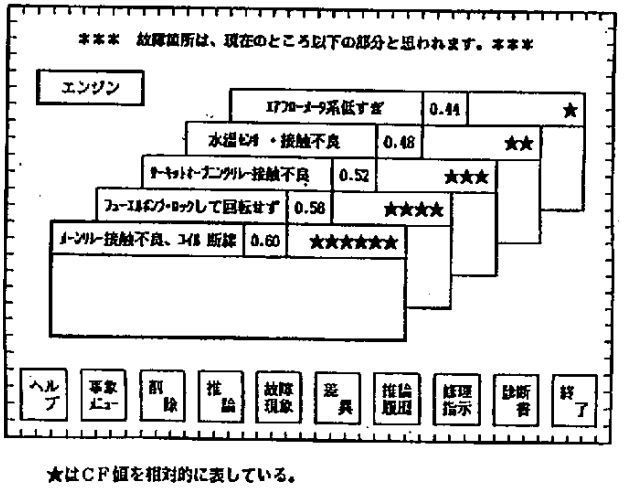


図４．１－１　セッション例１－故障状況入力

（2）推論結果の表示

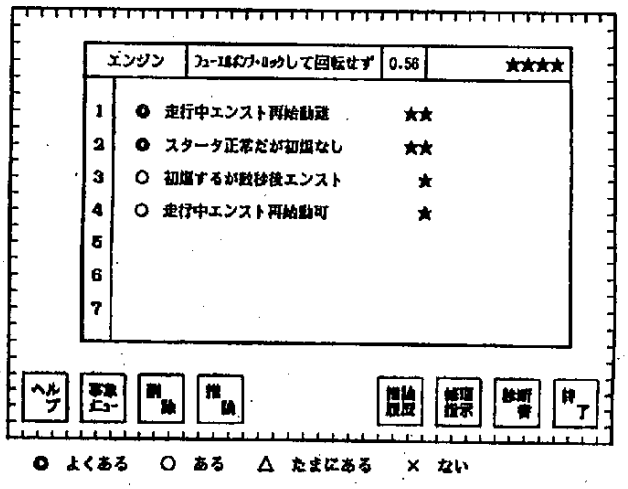


★はCF値を相対的に表している。

図４．１－２　セッション例２－推論結果の表示

（3）故障原因の検証

原因候補の中から一つを選択すると、その故障が起きた場合に引き起こされる故障現象を表示する。この時点で入力を修正追加し推論を再実行することも可能である。



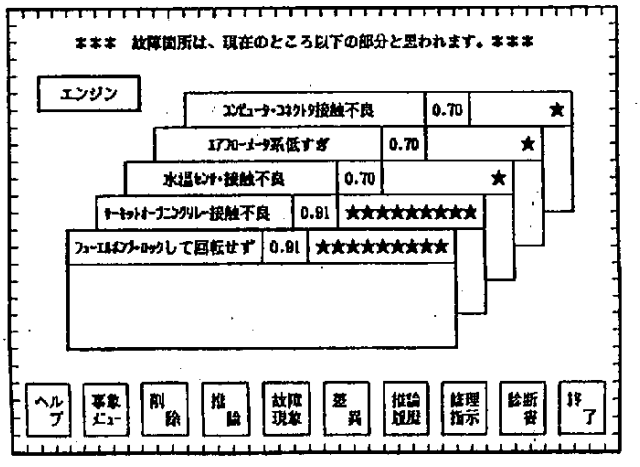


図４．１－３　セッション例３－故障原因の検証

４．１．２　プロトタイプシステムの構成と実現環境

プロトタイプの構成を図４．１－４に示す。推論を行なうホストコンピュータはＡＣＯＳ－８５０、端末は知識ベース開発用としてＰＣ９８００，コンサルテーション実行用としてＮ５２００／０７を使用した。動画の表示には端末から制御されたレーザディスク装置を利用している。メッセージおよび動画は、光学式タッチパネルを備えたマルチスキャ

ン・モニタテレビに表示される。

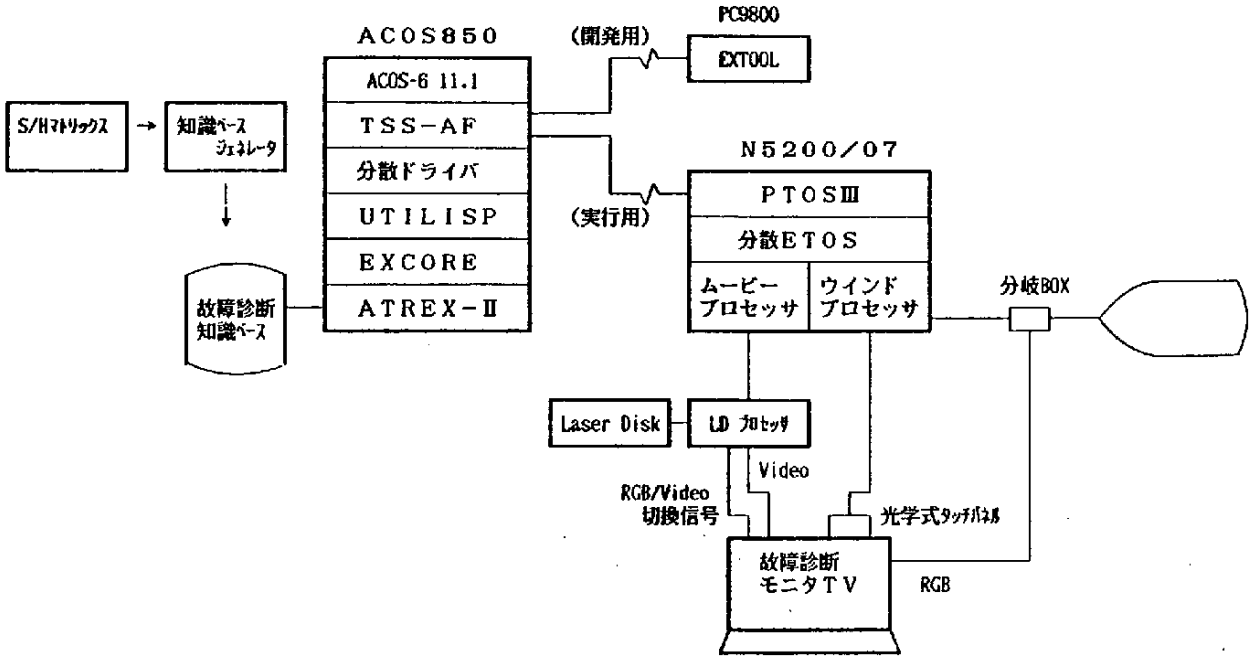


図４．１－４　プロトタイプシステムの構成

４．１．３　システム開発の手順

本システムの開発手順について、２章と対比させながら述べる。

（１）　問題の設定

知識システムの開発が成功するか否かは、問題の選択の適否に大きく依存する。自動車の故障診断については次のように考えられる。

(a)記号処理を中心とした問題であること．．．○

点検結果等で一部数値を扱うが、本質的には記号で表すことが可能である。

(b)専門家の経験的知識が利用可能．．．．．．△

サービス部門の熟練担当者から知識を得ることができる。ただし、新規機能については充分な経験の蓄積がない場合があり、設計部門と協力して診断知識を新たに作成することも必要である。

(c)充分なニーズがある．．．．．．．．．．．．○

診断の標準化、診断情報の活用などに有効。

(d)明確な価値を生ずる．．．．．．．．．．．．．○

診断の工数低減および品質向上をはかる。

(e)保守が容易．．．．．．．．．．．．．．．．．○

知識ベースの手法により保守性向上が期待出来る問題である。

（２）既存技術の評価

故障木解析や従来型のプログラムでは、自動車の故障を扱うには保守性、柔軟性の点で非現実的である。専門家自身がシステムを理解できるような構成にすることが是非必要である。

（３）知識源の同定

サービス部門の専門家の知識、整備マニュアルおよび設計的知識が利用できる。

（４）専門家モデルの同定

専門家は経験的知識と設計レベルの知識の両方を持っている。問題解決には事象駆動推論と目標駆動推論とを組み合わせて用いている。

（５）ユーザモデルの同定

サービス部門のスタッフが利用する。推論制御はシステム主導であるが、ユーザ主導の動作も可能とする。コンサルテーション時の操作性に優れていることの他に、ユーザによる知識ベースの保守容易性が要請される。

（６）知識表現の選択

プロトタイプではルール表現。推論制御は双方向可能。

（７）知識の移植

知識入力支援ユーティリティを用いて知識ベースを作成。

（８）知識ベースの管理

自動化は困難なため、当面は知識の階層化を行なうとともに、知識ベースエディタなどの支援機能の充実をはかる。

（９）性能の評価

実際の問合せデータを入力し結果を評価する。

４．１．４　タスクと問題解決機能のまとめ

自動車故障診断システムーＡＴＲＥＸーを、３．２で述べた特徴・基本タスク・問題解決機能に従って分析した結果を表４．１－１に示す。

表４．１－１　自動車故障診断システム－ＡＴＲＥＸ－の分析

	項　　目	関連性	実現度	コ　メ　ン　ト
特徴	1.設計レベルの知識の利用	○	△	浅い知識にコンパイルされている
	2.経験的知識の利用	○	○	
	3.経験的知識の不確実性	○	△	ＣＦ値を利用(入力にも指定可)
	4.操業データの利用	△	○	オンボード・ダイアグのデータを利用
	5.計測のための時間とコスト	○	△	点検手順で考慮されている
	6.知識の抽象化の必要性	○	×	因果関係を陽に扱っていない
	7.対話的診断の推論制御	△	△	冗長な質問の抑止等
基本タスク	1.異常現象の分類階層的表現	○	○	診断システムのモデルとして自然
	2.システム構造の階層的表現	○	△	車種による構造の違いを表現
	3.解釈問題のタスクを内包			
	システム状態の推定	○	○	
	あいまいさの処理	○	△	症状の表現(排気の色など)にあいまい性有り
	4.計測点の選択的決定	○	△	点検指示機能
	5.異常原因の同定	○	○	
	6.浅いモデルによる効率性	○	○	
	7.深いモデルによる完全性	○	×	
問題解決機能	1.事象駆動型推論	○	○	
	2.目標駆動型推論	○	△	故障原因から、それが及ぼす影響を推論する
	3.不確実性推論	○	△	現状はＣＦ値の足しこみであるが、今後D-S理論などの適用をはかりたい
	4.仮説推論	○	×	深いモデルに関連
	5.協調型推論	○	△	保守性向上に重要
	6.効率性と完全性の調和	○	×	
	7.費用/効果分析	○	△	点検コストと絞りこみの兼ねあい

４．１の参考文献

（山崎　知彦）

[岸 86]　岸 則政,自動車産業におけるAI応用-故障診断-,日本機械学会誌89-815,1189-1193(1986).

[Takahashi 86] R. Takahashi et.al., An Artificially Intelligent Car Mechanic, FISITA XXIth Intnl. Automobile Technical Congress, 865100(1986).

4．2　診断型システム、設計型システムの事例：ユーティリティプログラム使用法ガイドシステム

4．2．1　システムの目的と機能

本システムは㈱三菱総合研究所が昭和５９年から６０年にかけて実施したマルチクライアントプロジェクト「人工知能革命」の一環として開発したエキスパートシステムである。コンピュータシステムに付属している各種のユーティリティプログラムの使用法をユーザにアドバイスするシステムでプロトタイプのレベルのものである。なお、本システムが用いているエキスパートシステム開発用ツールＺＥＵＳも同じ様にこのプロジェクトの中で開発されたものである。

(1)　システムの目的

本システムの開発のねらいは大きく２つある。

①　専門家ではなく、マニュアルからの知識獲得

②　合成問題へのＺＥＵＳの応用

エキスパート・システムは専門家の知識を移植したコンピュータシステムである。その知識は、通常、知識エンジニアが１人または複数の専門家とのインタビューを通して引き出し、ルール化することによって得られる。しかし、これはエキスパート・システム構築の過程の中で最も困難な作業といわれている。専門家は知識エンジニアが自分のどの知識を引き出そうとしているか判らず、知識エンジニアはどう言えば専門家がシステムに合う形の知識を出してくれるかに苦労する。本システムではエキスパート・システムへの期待を知識獲得という観点からは一段下げ、既にマニュアル化された知識から構築した知識ベースでも充分に知的で、有用なものとなり得る、ということを確認することを１つのねらいとしている。

２番目のねらいは合成問題へのＺＥＵＳの応用である。エキスパート・システムが解決すべき問題は一般に分析型（Analysis)と合成型（Synthesis)に分類できる。　分析型の問題は対象（システム）における情報の流れや因果関係を詳細に調べ（分析し）、問題点、原因、解決策などを抽出するタイプの問題である。これに対し合成型とは設計やプログラムの作成など、新たな物やシステムを作り出す問題である。

合成型の問題は、分析型に比較して通常はるかに困難で、これまでに開発されたエキスパート・システムの多くは分析型の問題を扱っている。ＺＥＵＳもタイプとしては典型的な分析型向けのエキスパート・システム構築用ツールであるが、この事例研究ではＪＣＬ

（ジョブ制御言語）の生成という合成型の問題を、複数レベルの分析型問題に分割し、それらの結果をアルゴリズム的知識でまとめ上げることにより解決している。

(2) システムの機能

一般に、コンピュータ・システムにはファイルの複写・転送をはじめとして多くの定形的な処理を行うためのユーティリティ・プログラム群がある。しかし、通常それらは非常に多くの種類・機能を持っており、一般のユーザが適切にそれらを使いこなすことは必ずしも容易ではない。そこで、本システムではファイルの複写に問題を限定した上で

（i） 用途にあったユーティリティプログラムの選択

（ii） 選択されたユーティリティプログラムの使用方法の提示

を行うものである。

① ユーティリティプログラムの選択

この問題は典型的な分析型（診断型）の問題と言える。すなわち、解の候補としてのユーティリティプログラムをあらかじめ定義しておき（ここではＩＢＭの汎用大型機上に用意されているファイル複写ユーティリティプログラムＩＥＢＣＯＰＹ，ＩＥＢＧＥＮＥＲ，ＩＥＢＰＴＰＣＨの３種類）、一連のユーザとの対話によってデータセットのタイプと数、入出力デバイスなどを判断し最適なユーティリティプログラムを選択するものである。

② 選択されたユーティリティプログラムの使用方法の提示

①で選択されたユーティリティプログラムの具体的な実行方法をＪＣＬ（ジョブ制御言語）としてユーザに提示するものである。

ここで扱うＪＣＬ生成の問題には次のような特徴がある。

（i）ＪＣＬ生成は、コンピュータにある仕事をさせるために多くの制御ステートメント、ＤＤ文、ＥＸＥＣ文の組合せを作成するという意味で、合成型問題といえる。

（ii）制御ステートメントは処理の手続きを表す手続き的なもので、その順序に意味がある。

しかし、ＺＥＵＳは典型的な分析型のシステムであり、また、その帰結ルール（後ろ向き推論用のルール）は適用の順番にほとんど依存せず、宣言的知識の色彩が強い。従って状態を分析して、ある属性の値について推論するのは得意であるが、組合せや手続きを作り出すには必ずしも向いていない。

一般に合成型の問題は解を１つの属性の値として表現することができず、またその解も多数あるいは無限にある。そのため分析型よりはるかに難しい問題とされており、ＺＥＵＳに限らず他のエキスパートシステム、一般のコンピュータ・プログラムでもあまり取扱っていない。

とはいえ、ＪＣＬの生成は一般的な設計ほど複雑な問題ではなく、そのフォーマットはかなり限定されている。このように限定された合成問題の場合は、複数の階層からなる分析型の副問題に分割、整理することが可能である。そこで本知識ベースでは、後ろ向き推論の連鎖が作り出すツリー状の推論構造によって問題を多階層の副問題に分割し、末端における各副問題がすべて解決された後に、アルゴリズム的知識でそれらをまとめてＪＣＬを生成する、というアプローチをとっている。

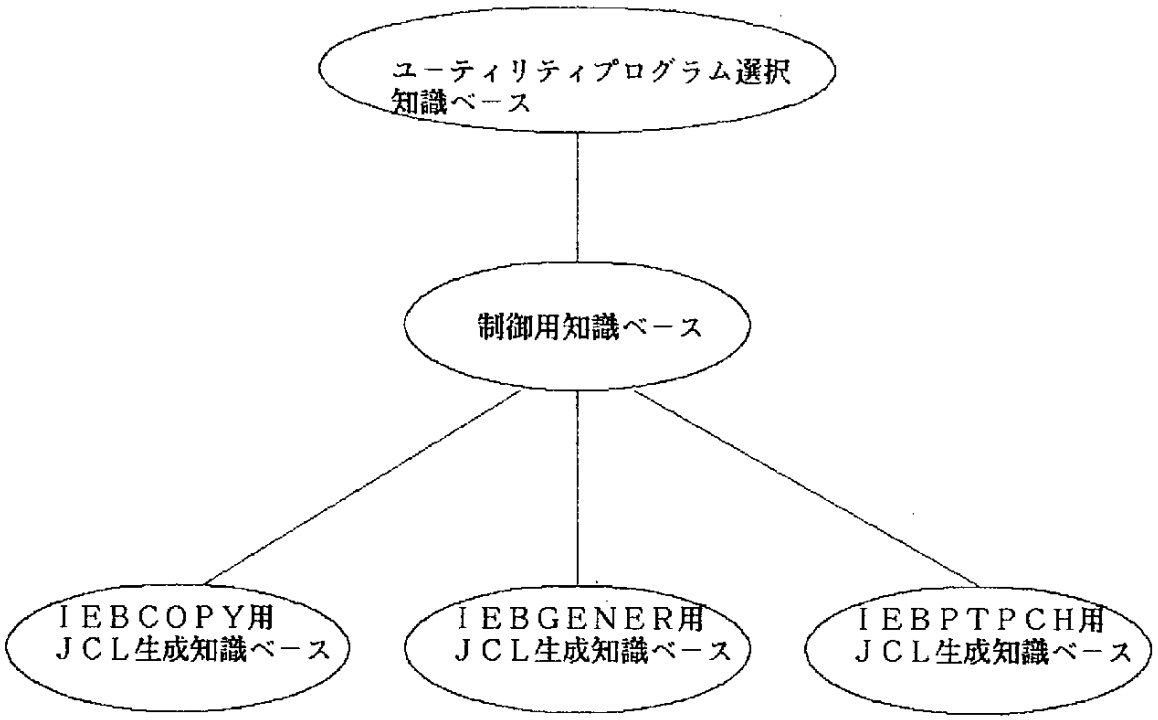
また文脈クラス樹（文脈インスタンス樹）は合成されるＪＣＬを表現するためのモデルとして使用されている。

なお、①と②の処理はそれぞれ独立な知識ベースの中で行われるもので、実際にはそれらの知識ベースを制御するための知識ベース（制御用知識ベースと呼ぶ）が用いられている。

4．2．2　システムの構成

(1)　システムの構成

知識ベースの構成を以下に示す。



(2)　システムの実現環境

ハードウエア	ＳＵＮ－２
ソフトウエア	ＵＮＩＸ　4.2ＢＳＤ
ツール	ＺＥＵＳ

(3)　システムの規模

①　ユーティリティプログラム選択知識ベース

前向き推論用ルール	１０５個
後向き推論用ルール	０個
アルゴリズム知識	０個

②　ＩＥＢＣＯＰＹ用ＪＣＬ生成知識ベース

前向き推論用ルール	１７個
後向き推論用ルール	１１８個
アルゴリズム知識	８個

③　制御用知識ベース

ルール	６個

4．2．3　システム開発の手順

(1)　問題の設定

①　記号処理を中心とした問題 …… ○

②　専門家の経験的知識が利用可能 …… △

マニュアルとして既に知識が体系化されている

③　充分なニーズがある …… ○

使い易いシステムとしてユーザインタフェースの向上必要

④　明確な価値を生ずる …… ○

プログラム相談などの手間が省力化でき、ユーティリティプログラムの使用法の教育にもなる。

⑤　保守が容易 …… ○

(2)　既存技術の評価

従来型手法では生産性悪く、保守性、拡張性にも難がある。

(3)　知識源の同定

マニュアル

(4)　専門家モデルの同定

無し

(5)　ユーザモデルの同定

コンピュータシステムについて一通りの知識を持った技術者を想定

(6)　知識表現の選択

プロダクションルールによるユーティリティプログラム選択規則の表現、文脈樹による解の合成

⑦　知識の移植

⑧　知識ベースの管理

知識ベースの矛盾や冗長性などをツールであるＺＥＵＳがある程度まで指摘

⑨　性能の評価

ユーザインタフェースへの工夫や扱える問題の範囲（すなわちユーティリティプログラムの種類）の拡張が必要

4．2．4　タスクと問題解決機能のまとめ

(1)　ユーティリティプログラム選択知識ベース

	診　断　問　題	関連度／実現度	コ　メ　ン　ト
定義	診断問題とは、システムに異常または故障が生じたとき、観測されたデータおよびシステムに関する知識を利用して、原因の箇所を同定することである。		厳密な診断問題とは言えないが、診断問題を選択型問題と考えると、この範囲に入れることができる。
特徴	1．設計レベルの知識の利用	×／×	
	2．経験的知識の利用	△／×	マニュアル化された知識を用いる
	3．経験的知識の不確実性	×／×	
	4．操業データの利用	×／×	
	5．計測のための時間とコスト	×／×	
	6．知識の抽象化の必要性	×／×	
	7．対話的診断の推論制御	○／○	
基本タスク	1．異常現象の分類階層的表現	×／×	
	2．システム構造の階層的表現	○／○	
	3．解釈問題のタスクを内包 システム状態の推定 あいまいさの処理	 ×／× ×／×	
	4．計測点の選択的決定	×／×	
	5．異常原因の同定	○／○	目的のユーティリティプログラムを異常原因と考えることができる。
	6．浅いモデルによる効率性	○／○	ユーティリティプログラム選択の知識をプロダクションルールとして表現
	7．深いモデルによる完全性	×／×	
問題解決機能	1．事象駆動型推論	○／○	
	2．目標駆動型推論	○／×	目標駆動型推論でも実現できるが、用いていない。
	3．不確実性推論	×／×	
	4．仮説推論	×／×	
	5．協調型推論	×／×	
	6．効率性と完全性の調和	○／△	効率はあまり意識していない
	7．費用／効果分析	○／×	プロトタイプでもあり費用／効果分析はしていない。

(2) ＩＥＢＣＯＰＹ用ＪＣＬ生成知識ベース

	設計問題	関連度／実現度	事例との関連
特徴	1．システム構造の解空間が大	△／△	
	2．解析による合成が基本的	○／○	
	3．構成要素の最適化の必要性	△／△	プロトタイプでもあり、最適化は意識していない
	4．検証の完全性および効率性	○／△	解の完全性は必須であるが、効率性については3と同様意識していない。
	5．対話形式での設計支援問題	○／○	
	6．設計段階に応じた支援形態	○／×	
基本タスク	1．構成要素とその関連の表現	○／○	文脈樹として表現
	2．設計問題の階層的表現	○／○	
	3．代替案の自動生成	△／×	
	4．部分システムの評価	○／○	文脈樹中の各文脈ごとに解を生成
	5．上位レベルへの知的後戻り	△／×	
	6．設計事例の検索と利用	△／×	
	7．並列的問題解決	×／×	
問題解決機能	1．トップダウン精密化戦略	○／○	文脈インスタンス樹の生成という形をとる。
	2．拘束最小化戦略	×／×	
	3．最適化機能	△／×	
	4．検証機能		
	5．仮説推論による知的探索	×／×	
	6．類推による設計事例の利用	×／×	
	7．協調型推論	×／×	

参考文献

（担当　三菱総合研究所　小林）

〔三菱総研　85〕　三菱総合研究所、人工知能革命フェーズ２事例研究報告書、1985

〔日本アイ・ビー・エム〕　日本アイ・ビー・エム株式会社、ＯＳ／ＶＳ２　ＭＶＳユーティリティ、リリース3.7

４．３　制御型システムの事例：高炉操業監視支援システム

制御領域での知識システムの実用的応用事例はほとんどないため、制御問題のtypicalな例ではないが、新日本製鉄㈱君津製鉄所において開発された「高炉操業監視支援システム」について紹介する。

４．３．１　システムの目的と機能

４．３．１．１　制御対象プロセスの特徴

鉄鉱石や石炭といった原料をもとに鋼板やパイプといった製品を製造する一貫製鉄工程において、高炉プロセスは最も上工程に位置して溶銑（溶けた銑鉄）を生産している。この高炉プロセスは、全工程で消費するエネルギーの７０～８０％を消費すると共に生産量を決定しているきわめて重要なプロセスである。

「高炉」は鉱石及びコークスを原料として溶銑を得るプロセスであり、本体内容積は5000m^3、出銑量は日産10000tonにも達する。高さ約40mの炉体上部から鉱石及びコークスを層状に装入し、炉体下部からは約1000℃以上の熱風が高圧で送りこまれる。炉内では、これらの固体・気体及び溶融した液体が入交じり、酸化・還元が高温環境下で大規模かつ連続的に行われる。

炉の周辺に取付けられた数百の熱電対をはじめとする炉内センサの情報を連続的に収集しており、操業オペレータはこれらのセンサ情報、生産された溶銑の状態等の目視情報及び部分的なプロセスモデルによる推定結果などを総合的に考えて炉内状況を判断し、操業上のアクションを行う。このアクションの目的は、①炉の不安定化傾向をいち早く予想し、”棚吊り”“冷え込み”といった悪い状態になることを回避する、②安定状態にある時より安い生産コストになるように燃料比を下げる、である。特に、①の不安定回避に失敗し、冷え込み状態にまで陥ると、百億円以上の生産損失になるため、日々の操業判断は非常に重要である。このアクションの種類には、炉体上部での原料分布（ＭＡノッチまたはＰＷノッチによって制御可）、鉱石とコークスの割合（ＦＲ：Fuel Rate ）、吹込む熱風の量（風量）、等がある。

制御の対象としての高炉の特徴をまとめてみると、次のようになる。

①高炉は時定数の大きな系であり（制御周期は最短１０分程度）、システムのレスポンスにそれほどリアルタイム性を要求されない。

②今回の知識システムの適用は、オペレーターに対するアクションガイドの方式をとっており、一般の閉ループ制御システムに求められるような厳密な意味での制御の信頼性・安

定性の保障は要求されない。

４．３．１．２　既存技術とその問題点

高炉の操業管理は、従来主に次の２つの方法で行われてきた。

①プロセスモデリングによる演繹的問題解決

②経験的操業判断基準（作業標準）のシステム化

①の例として、装入物分布推定モデル，軟化融着帯推定モデル，炉熱推定モデル等の部分的プロセスモデルが実用化され、操業オペレータの判断のための１つの情報として活用されている。しかしながら、高炉プロセスは炉内現象が極めて複雑なため、日常操業監視に使える精密なプロセスモデルの作成には限界があり、この点においてはモデルをベースとした演繹的問題解決は実現されていない。

②の例として操業判断基準をプログラム化した「操業管理システム」がいくつか実用化され、日常操業判断のばらつき軽減等の成果をあげている。一方、操業前提・方針の変更に伴った判断ロジックの変更をシステム分析者の手を経ずに操業オペレータでできるよう、より柔軟で保守性に富むシステムが要求されている。

４．３．１．３　知識システム導入の狙いとシステムの全体像

高炉プロセスの監視・管理の問題に対し次の効果を期待して知識システムを導入した。

(1) 優秀な操業オペレータの知識の有効活用

操業エキスパートが持つ経験的知識を獲得し利用可能とすると共に、従来技術（プロセスの部分的モデル）の利用ノウハウも使用できるようにする。

(2) 操業前提・方針の変更等に対するシステムの柔軟性の確保

モデルを簡略化することなくロジックの可読性を確保し、従来システム分析者が対応していたロジック変更に対しても操業オペレータが自ら対応できるようにする。

(3) 操業オペレータとのコミニュケーションが良好なシステムの実現

システムから示されるガイドを操業オペレータが十分理解して判断できるようなロジックの可視化手段や対話機能をはじめとする各種マンマシーンインターフェイスを充実させる。これらを実現するシステムの全体像を図４．３－１に示す。

このシステムは次の３つのフェーズで運用される。

①フェーズ１・・システムの構築

高炉の操業に関する知識を操業エキスパートから抽出し「知識ベース」を作成する。

②フェーズ２・・システムの実行

時々刻々オンラインで送られてくるセンサのデータ（プロセスコンピュータでの１次前処理情報を含む），部分的プロセスモデルの計算結果（プロセスコンピュータ内で計算），操業オペレータの目視情報の入力結果、及び作成済みの「知識ベース」の内容に基づいて「推論機構」が働き、操業オペレータに対しアクションの指示を行う。

③フェーズ３・・システムの修正

システム運用結果は逐次記録される。これに基づいてより良い操業条件を検討し、知識ベースの内容を追加・修正する。

システムは上記フェーズ２，３を繰り返すことにより精密化が図られる。

４．３．１．４　操業知識の獲得とその知識表現

操業オペレータの監視・判断に関して、知識システムでサポートが必要な知識は、①日常操業の知識、②非定常操業の知識、③休風ガイドの知識、④設備トラブル対処の知識、の４つに大別される。実機システムではこれら全てについて知識ベースの作成を行ったが、ここでは①の「日常操業の知識」の一部である「炉の不安定回避の知識」の獲得とその表現について述べる。

知識の獲得は、長年の操業と解析の経験を持ち最も信頼されている操業オペレータ経験者を選任し、知識システムのとりまとめを行ういわゆるＫＥ（ナレッジエンジニア）との徹底的なディスカッションを行うことによって行われた。高炉の炉内状況判断からコントロールアクションを決定する枠組みは、情報として手に入る検出端のデータ（履歴を含む），既存モデルのアウトプット，目視データ等を基に、物理的意味付けをもった中間的判断（中間仮説）を行い、その中間仮説に基づいて炉の悪化傾向の種類及び度合いを判定しそれに応じたコントロールアクションを決定する。入力情報から中間仮説に至る判断、及び、中間仮説から悪化傾向の予想に至る過程には不確実性を伴うため、ＨＧ（Heuristic Grade ）を考案した。このＨＧは前提部に重み付けを行い、結論部にしきい値をおいて、前提部の重みの和との比較で結論部の成立可否を決める方法である。

４．３．２システムの構成と実現機能（君津第４高炉の実機システム”ＡＬＩＳ”）

前述の検討結果に基づいて設計・製作された君津第４高炉の実機システム”ＡＬＩＳ”（Artificial and logical Intelligence System for blast furnace operation）についてシステム構成を図４．３－２に示し特徴を列挙する。

(1) 操業オペレータでも理解・作成可能なプロダクションシステム型ツールの採用。

(2) 操業オペレータに対する推論結果説明機能の充実（適用ルールの表示、各中間仮説群におけるＨＧテーブルの表示、推論結果に応じた説明グラフィックス群の連動表示、等の機能）。

(3) プロセスコンピュータ内の既存技術の最大活用を考慮したシステムリンケージ。

(4) テスト機能の充実（現在オンライン状態にある知識とは別の知識ベースを用いて、①逐次のオンラインデータを用いたパラランによる比較検証、②過去のデータを用いた知識ベースの妥当性検証及びデバッグ）。

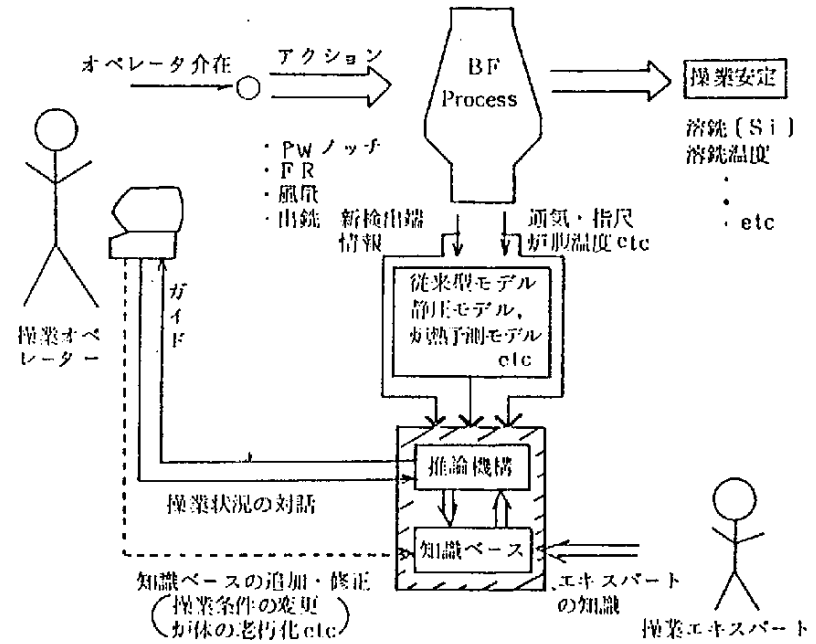


図４．３－１　知識システムの高炉への応用イメージ

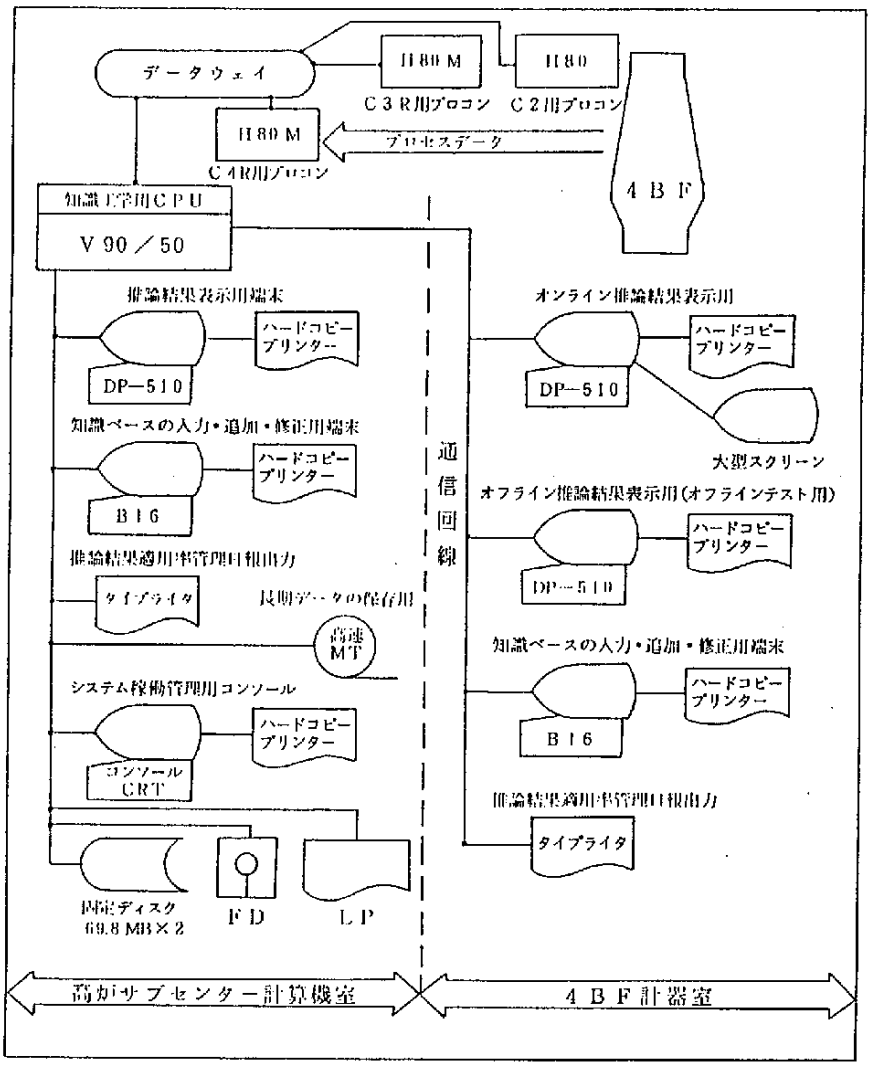


図４．３－２　システム構成と機器配置

４．３．３　システム開発の手順

ここでは、制御問題のtypical な例ではないが当社が手掛けた「高炉操業監視支援システム」についてその開発手順を以下のフェーズ毎に考察したい。

(1) 対象問題の選定

(2) 知識獲得

(3) システム環境選択（ＡＩツールの選択または構築），プログラミング

(4) 不確実性等に関する知識表現

(5) チューニング・効果検証

４．３．３．１　対象問題の選定

(1) 既存技術の適用の問題点

①プロセスモデリングの限界

高炉は固体・液体・気体の三相間で酸化・還元が高温環境下で大規模に行われる極めて複雑な反応炉である。上部から装入されるコークス・鉱石が数時間かけて炉内を降下しながら徐々に加熱され、反応変質しつつ液化あるいは気化してゆくわけであるが、この状況を十分な精度で定量化することは極めて困難である。

②経験的知識のモデリングの限界

操業における判断基準をモデル化し、「操業管理システム」としてまとめあげる手法はかなり導入され一応の成果をあげてきた。しかしながらここで扱われた判断ロジックは陽に定義されておらず、操業者に判断過程が示されない、又は、わかり易くしようとするあまりモデルの簡素化を図りすぎ十分なモデリング精度を得られていない、等の問題があった。

(2) 知識工学適用の範囲

上記のような問題点のあった高炉プロセスの監視・制御に対し、知識システムの適用を図り効果の確認を行った。本システムは一般の制御システムと比して次のような特徴を有している。

①知識工学適用の目的は主にアクションガイドシステムの実現であり、一般の制御システムに求められるような厳密な意味での制御の信頼性・安定性の保障は要求されない。

②高炉プロセスは時定数の大きな系であり（短くても１０分単位）、システムのレスポンスにそれほどリアルタイム性は要求されない。

４．３．３．２　知識獲得

高炉への応用の場合、領域専門家のモデリングにより知識獲得を行い、良い結果を得た。これは、次のような前提があったためであると思われる。

①高炉操業担当者の中に極めて優れた人が一人いたこと。

②知識獲得段階にこの人を専従させることができたこと。

③知識を引出す知識工学者の立場の人が高炉の操業に精通しており、円滑かつ深いインタビュー，ディスカッションを行なえたこと。

４．３．３．３　システム環境の選択（ＡＩツールの選定または構築）

(1) ツールの選定

プロトタイプでのツールに対する要求は、プロセスデータの取扱いを前提とした表層表現能力が主であった。

選択した日立ＥＵＲＥＫＡは、プロセスデータをフレーム的に表現し、規則はプロダクションルールで表現する形式をとっており、獲得した知識をコーディングするのは比較的容易であった。

(2) 既存システムとの整合の考慮

高炉にはプロセス制御用コンピュータがすでに存在しており、プロセスデータのロギングやマイナーな制御系を形成している。今回のエキスパートシステムはガイダンスシステムであるため、①既存システムへの様々な悪影響をなくすアイソレーションを施すこと、②既存システムが複数の高炉に対して共通のデータウェイで接続されており今後も複数の高炉に対する共通の拡張性を確保したいこと、等を考えて、バックエンドプロセッサ型のシステム設計とした。

(3) マンマシーンインタフェイス

高炉でのシステムは最終ユーザーを操業者とし、知識のメンテナンスも彼等に解放するという思想に基づいているため、日本語表現はもとより専用知識エディタ，推論結果説明用簡易グラフィクス等を製作した。

なお、実用システムにおいては、既存システムとのデータ取合いのためのプログラム製作に要したマンパワーは大きかった。

４．３．３．４　あいまいさの表現

不確実性等の表現方法については、次のような事柄を考慮しておく必要がある。

①各データの大きさの認識（区分）に起因する望ましくないバラツキを排除して論理が進

むような構成とすること。・・・例えばｆｕｚｚｙ等を有効に活用する。

②場合分けの数が膨大になることを避けるため、表現の適度な一般化を行うこと。

高炉の場合、「プロセスデータを表す各要素」と「高炉内での現象を表す各中間仮説」の各々に重み付けを行い、適度なグルーピング、一般化をした知識表現としている。

４．３．３．５　チューニング・効果検証

制御分野へのアプローチにおける大きな問題点は目的の達成度，制御の最適性の把握が困難なケースが多く、最適化のための方法が不明という点である。現状では知識のチューニングや効果検証はシミュレーション＆チェックという手段に頼らざるを得ない。シミュレーションを網羅的に行うことは事実上不可能であり、シミュレーションでの効果的検証方法が今後の課題である。

チューニングおよび効果検証では、テスト用プロセスデータを用いたシミュレーションにより判断を行った。テストに用いるデータは操業上想定される様々なパターンを十分反映したものを注意深く選んだ。ちなみにテストケースは１００近い。

4．3．4　タスクと問題解決機能のまとめ

前述の「個別問題領域における特徴・基本タスク・問題解決機能」と本項で取上げた「高炉操業監視支援システム」での関連度・実現度について、表の形式でまとめる。

制御問題のタスクと問題解決機能		高炉プロセスの操業監視支援問題との関連度及び実現度		
	制　御　問　題　一　般	関連度	実現度	コ　メ　ン　ト
特徴	1．時間遅れによる履歴依存性	○	○	・時定数、時間遅れ要素大
	2．非線形性による局所性	○	○	・非線形性大
	3．安定化の応答性	◎	○	・グローバルな安定化をめざす
	4．制御精度の実現	△	△	・厳密な精度は要求されない（評価難）
	5．実プロセスとの接続	◎	◎	・既存システムとの接続・協調が重要
	6．実時間性と信頼性の確保	△	△	・分オーダーのリアルタイム性でよい
	7．離散系における解析問題	×	×	・特に関連なし
基本タスク	1．システム構造の表現	○	○	・領域専門家の思考を表現
	2．システム動特性の表現	△	○	・陽には記述していない（内包している）
	3．診断問題のタスクを内包＊	△＊	△	
	4．モデルによる状態の予測	○	○	・部分的プロセスモデルの利用方法も含む
	5．安定化動作の優先的実行	◎	◎	・炉の不安定回避を優先的に実行
	6．安定操業下での省エネ化	◎	◎	・「攻撃的操業知識」として表現
	7．オペレータガイダンス	◎	◎	・コミニュケーション手段
問題解決機能	1．状態の診断機能	◎	◎	・＝操業状態の判断
	2．シミュレータによる予測	○	×	・問題に適したシミュレータはない
	3．定性推論による予測	△	×	・実施していない
	4．制御則の多目的評価	×	×	・　〃
	5．アクションガイダンス	◎	◎	・コミュニケーション手段
	6．デッドロックの解析	×	×	・実施していない
	7．インターロックの回避	×	×	・　〃

凡例	関連度	実現度
◎	非常に大きい	かなり実現
○	大きい	ほぼ実現
△	やや有り	やや実現
×	全く無し	全く実施せず

	＊　＊＊	関連度	実現度	コ　メ　ン　ト
＊診断問題の基本タスク	1．異常事象の分類層的表現	△	△	
	2．システム構造の階層的表現	○	○	
	3．解釈問題のタスクを内包**	△**	△	
	4．計測点の選択的決定	△	△	
	5．異常原因の同定	○	○	
	6．浅いモデルによる効率性	○	○	
	7．深いモデルによる完全性	△	△	
＊＊解釈問題の基本タスク	1．特徴の抽出	×	×	
	2．モデルとの不完全な照合	△	△	
	3．システム構造の同定	△	△	
	4．システム状態の推定	△	△	
	5．あいまいさの処理	◎	◎	・重要要素。HGとして表現
	6．不完全性の処理	○	○	・関連あり
	7．解釈の多様性	△	△	

（担当　新日本製鐵㈱　湯井勝彦）

（参考文献）

［湯井　８７］湯井、ほか，”高炉プロセスの操業監視支援における知識システムの適用”計測と制御，２６巻8号，７１２－７１９（１９８７）

［岩本　８６］岩本、ほか，”エキスパートシステムの高炉操業への応用”第２５回計測自動制御学会学術講演会予稿集，２１３－２１４（１９８６）

［奥野　８６］奥野、ほか，”高炉内層頂部におけるコークス層崩れ現象の装入物分布に及ぼす影響”鉄と鋼，７２巻7号，７８３－７９０（１９８６）

［加瀬　８０］加瀬、ほか，”融着帯形状推定モデルによる高炉操業の解析”鉄と鋼，６６巻１３号，１９２８－１９３６（１９８０）

［杉山　８３］杉山，”高炉の流動・伝熱・反応解析”第９４回西山記念技術講座資料，１３３－１７２（１９８３）

［新日本製鉄　８４］新日本製鉄（株）名古屋，”高炉操業管理システム（ＡＧＯＳ）について”鉄協共研第６４回製銑部会資料，鉄６４－１７自（１９８４）

［岡部　７９］岡部、ほか，”ＧＯ－ＳＴＯＰシステムによる高炉の安定操業”川鉄技報，１１巻１号，３４－４３（１９７９）

［岩村　８５］岩村，”プロセス状況の分類 ― 高炉炉況診断を例として ―”計測と制御，２４巻１１号，１０２５－１０３１（１９８５）

[資料　８５]資料，”鉄鋼業における高炉炉況診断システム”計測と制御，２４巻１１号，１０３２－１０３３（１９８５）
[増位　８６]増位、ほか，”システム制御用知識処理ソフトウェアＥＵＲＥＫＡの開発”計測自動制御学会第２回「知識工学研究会」資料，（１９８６）
[石塚　８３]石塚，”不確かな知識の取扱い”計測と制御，２２巻９号，７７４－７７９（１９８３）
[Barnett,J.A.　８１]Barnett,J.A.,"Computational Method for a Mathematical Theory of Evidenc-e”７－ｔｈ　ＩＪＣＡＩ，８６８－８７５（１９８１）
[Dempster,A.P　６８]Dempster,A.P.,”A Generalization of Baysian Inference"J.of Royal Stati－stical Society，Ｂ－３０，２０５－２４７（１９６８）
[特集　８３]特集，”あいまいを考える”計測と制御，２２巻１号，（１９８３）
[諏訪　８６]諏訪、ほか，”エキスパートシステム開発事例にみる知識獲得の諸相”計測と制御，２５巻９号，８０１－８０９（１９８６）
[特集　８５]特集，知識工学，情報処理，２６巻１２号，（１９８５）
[小林　８６]小林重信，「知識工学」，昭晃堂（１９８６）

4．4　計画型システムの事例：集配スケジュール作成システム

4．4．1　システムの目的と機能

(1)　システムの目的

本システムは，集配スケジュールを作成している専門家の業務を支援することが目的であるが，非専門家もユーザとして想定している．システムが扱う問題内容は以下の通りである．

ネットワークで表現された地域において，資源を集配する問題を考える．ネットワークの各ノードは単位１の資源を生産するノード，単位Ｕの資源を消費するノード，及びその他のノードに分類されている．ここでＵは全ノード数より大きい．この地域で，容量Ｎの集配車を決められた台数用いて生産ノードから資源を受け取り，消費ノードへ資源を供給しなければならない．各集配車は，いずれかの生産ノードまたは消費ノードから出発し，自分の容量の範囲内で資源を集め，消費ノードへ運ぶ．各消費ノードは到着した集配車の持つ資源を全て消費する（到着したときに集配車が空であってもよい）．(図４．４－１)

集配車は，ある生産ノードまたは消費ノードから他の生産ノードまたは消費ノードへ到着するまで一定の時間がかかる．ここでは仮に１日とする．すなわちその日の朝，生産または消費ノードに駐車していた集配車は，その日の夕方，他の生産または消費ノードに到着して１日が終わる．ここで，全ての生産または消費ノードには，一般に複数の集配車が駐車しているが，朝と夕方とで駐車台数は等しくなければならない（ただし集配車実体は異なっていてよい）．この状態で次の日の集配が続く．（図４．４－２）

初期値として，ｎ台の空の集配車を各ノードに与えたとき，各集配車のその日の行き先を上の条件に抵触しないように決定することが目的である．

この他にも条件があるが，それらを含めて問題内容は次のように要約される：

「集配車が自分の容量を越えずに資源を集め，消費ノードへ供給する．１台の集配車がある消費ノードから出発し，全ての消費ノードと生産ノードとを１回づつ通過して最初の消費ノードに戻れるような集配スケジュールを作成せよ」

これをＡＲＰ（Automobil Routing Problem）と呼ぶ（図４．４－３）．

(2)　問題の性質

ＡＲＰをグラフにおける閉路問題として定式化する．

定式化：「ある有向グラフ G(V,E) に対し，「特定頂点」Ti (i=1, …,n) の集合

{Ti ; Ti ∈ V}，　V＝{ v1,……,vv }

及び整数N≧0が与えられるものとする．このときTp からTq (p,q≦n ； p≠q) への径路に始点と終点とを除いてN個以下の頂点を含み，かつ全ての頂点を1回づつ通過する閉路が Gに存在するか．」

T1, T2 ；N＝2の例を図4．4－4に示す．

このように定式化するとＡＲＰには本質的に，効率的な解法アルゴリズム（問題サイズの多項式オーダで計算できるアルゴリズム）が存在せず，いわゆるＮＰ完全であることが証明できる．したがって単純な木探索は適用できず，実システムではヒューリスティクスおよび問題の分割・階層化による探索を応用している．

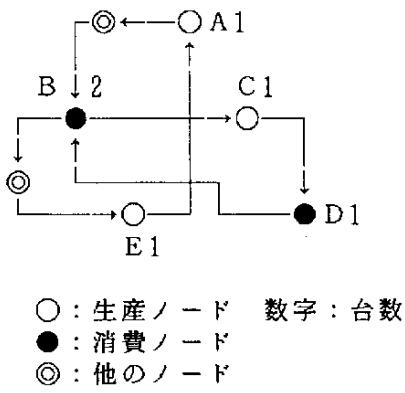


図4．4－1　ネットワーク

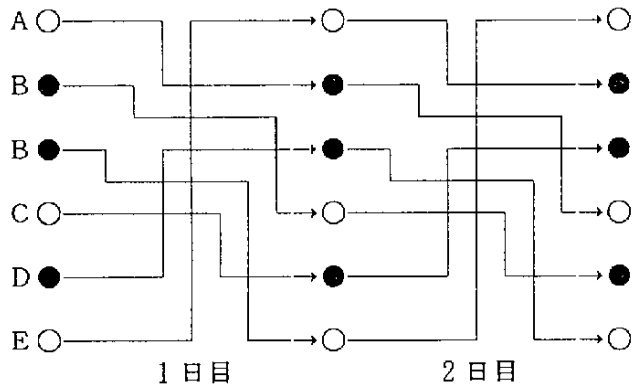


図4．4－2　スケジュールの進行

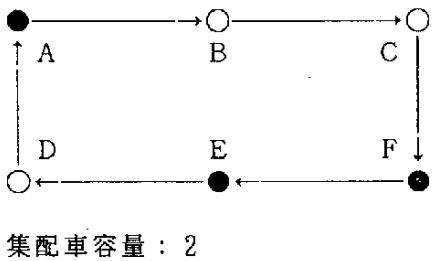


図4．4－3　ＡＲＰ

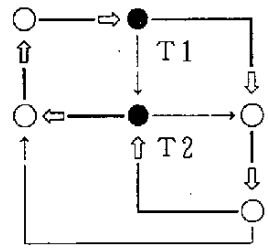


図4．4－4　適合閉路

(3)　システムの機能

本システムは与件データ（ネットワーク，集配車配置等）を入力すると，自動的にＡＲＰを解くように動作する．この間，人間は介在せず，表示される実行過程をモニタするのみである．最終結果はＬＩＳＰからＣＡＤシステムにデータを渡してグラフィック表示す

る．結果の妥当性はこれを見れば判断できるので，実行履歴の保存は特に行なってはいない．

問題解決機能としては，メモリ容量の範囲でネットワークや集配車数に制限はない．与件データに矛盾がない限り，パラメータ変化には影響されないように知識ベースを作成した．

４．４．２　システムの構成と実現環境

システム構成を図４．４－５に示す．システムはメインフレーム上のＴＳＳ環境で走行する．後で述べるように，システム全体では４種類の知識ベースが存在し，ＡＲＰのための知識ベースはその１つである．

言語はＬＩＳＰ（ＵＴＩＬＩＳＰ），ツールはＥＳＨＥＬＬ，およびCAD 用にＩＣＡＤを使用した．メモリはユーザ領域７ＭＢである．

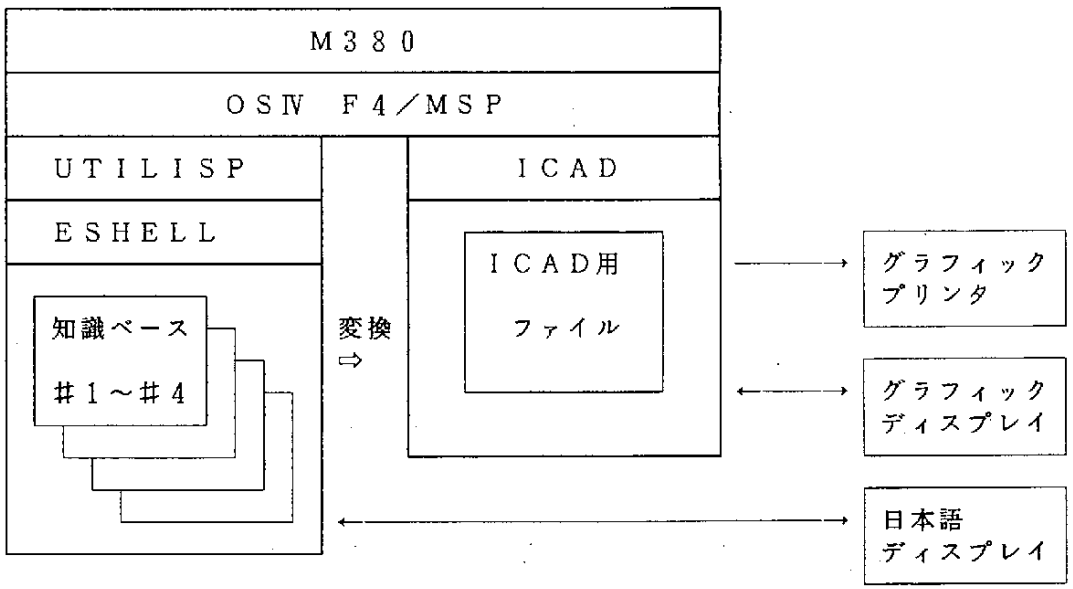


図４．４－５　システム構成

４．４．３　システム開発の手順

システム開発の内容は，大部分が知識獲得（インタビュー）やプロトタイピングによる問題分析過程であったといえる．問題分析の過程を整理すると次のようにまとめられる．

(1)　問題領域の理解と知識獲得

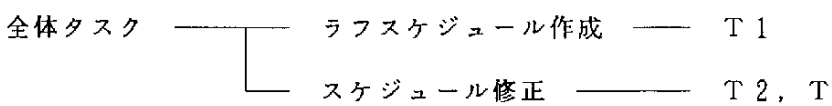
専門家が扱っている問題の概念を理解すると同時に，システムの運用形態をユーザと協

議した．これらは独立ではなく，問題の特性が運用形態を定め，その運用形態によって知識ベースの表現形式が制約を受け，さらに知識獲得の方針にまで波及することとなった．すなわち，問題の特性としてネットワークの構成や集配車の数・種類等のパラメータの変更が多発するため，ユーザとしてはパラメータ変化の都度知識ベースのメインテナンスが発生しては困る訳である．したがって知識ベースはメインテナンスフリーとする必要がある．一方，専門家による知識提示はパラメータの具体値を含んだ形で行なわれるため，そのままルール表現にすることができず，パラメータ値に依存しない表現に変換しなければならない．専門家の知識は，スケジュールを作成するための手段を表わしたものであるから，提示された手段（表層知識）に対し，なぜその方法をとるかという理由（深層知識）を把握する方針を採った．このため知識獲得の負担はかなり重くなったが，パラメータ変化に依存しない知識を得ることができた．

(2) タスクの抽出と分割

よほど簡単な問題でない限り，問題全体をいくつかの副問題に分割することは必須である．本例ではスケジューリングにかかわる各種の制約条件の優先度という観点から問題を分割した．

専門家の作業（タスク）を検討すると，２種類のタスクに大別することができる．すなわち「まず与件からラフなスケジュールを作成する」，「次に各種の制約条件を満足するように修正を施す」というタスクである．後者のタスクは更に細分化され，優先度の高い制約条件から順次修正するという戦略になる．すなわち対象モデルとしてのスケジュールが各タスクによって徐々に改善されるプロセスになる．したがって制約条件がいくつあるかが問題であるが，最終的に４種類の条件となった．その中の１つが，実際には最初に述べたＡＲＰである．これらの各々が独立なタスクとして確定され，ラフスケジュール作成を含めて合計５個のタスクとなった．



タスクの流れはＴ１からＴ５へ至るが，協議の結果Ｔ５はシステム化の対象範囲外とすることにした．また，各タスクはそれだけで１つのシステムに匹敵する規模であため，各々を独立した知識ベースとして作成した．

タスクを分割するにつれ，独立なタスクとは逆に，各タスクに共通なタスクも浮び上が

ってきた．これは，スケジュールを作成／修正するための操作手段あるいはテクニックといった性格を持つタスクである．したがってＴ１からＴ４のタスクを実現するシステムでは，必要に応じてこのタスク（操作手段）を用いている．

(3) 概念の統合・抽象化

パラメータ値に依存しない知識表現が必要なため，具体的な知識を抽象的な知識に変換する必要があった．また，制約条件やスケジュール構成要素の属性も，当初は雑多な概念が入り組んでいた状態から，共通化・言い換え・新概念などによって極力概念の数を減らすようにした．この点は思想の問題で，意見が分れるところかもしれない．

(4) 階層化

ＡＲＰを解く場合，スケジュール上で修正タスクを適用すべき要素を探索する必要があった．しかも修正タスクは１回ではなく何回か適用しなければならない．実際の問題規模は，平均するとノード数３０，ノード間の径路数８０，集配車数１０という規模である．この中から修正タスクが適用可能であり，しかも適用して効果があがる要素（時刻，集配車，径路に依存して決るノードを通過する集配車の組合せ）を選択しなければならない．これを単純な木探索で行なうわけにはいかず，スケジュールの構成要素を階層化して表現し，上位階層から順に生成検査を適用することとした．

(5) インプリメント

各タスクのインプリメントは同時進行であったので，システム間のインターフェースすなわちスケジュールのデータ構造を最初に決定しておく必要があった．これは，あるタスクの結果が次のタスクの入力となるためである．特定のタスクが省略されることも考え，インターフェースはすべて共通にした．

(6) 評価

評価は，各タスクが担当する制約条件が，各種のケース（実データ）にたいしてどの程度満足されているかという観点から行なわれた．レスポンス・メモリ容量・ＭＭＩ等は努力目標という程度であり，問題解決能力が最重要視された．

4．4．4　タスクと問題解決機能のまとめ

表4．4－1　問題領域別基本タスクとの比較

	計画問題	関連度	実現度	事例との関連
特徴	1. 探索空間が非常に広い	◎	－	・探索空間に占める解の比が僅少
	2. 評価属性が多様である	○	－	・多様であるが各タスク毎に定義可能
	3. 環境予測が不確実である	◎	－	・パラメータの変更は業務計画と直結している
	4. 部分計画間の相互作用	○	－	・上位タスクの結果が下位タスクの実行を拘束する
	5. 既存計画の再利用	△	×	・専門家の経験として折り込まれる
	6. 費用と便益のトレードオフ	△	－	・便益優先
	7. 対話型計画作成支援の要求	○	○	・非専門家がルーチンワークで使うにはバッチ的な方が良い
基本タスク	1. 計画過程の階層化	◎	◎	・問題分析を通して階層を見出す
	2. 組合せ的探索	◎	◎	・階層化生成検査
	3. 制約条件の相互作用	◎	◎	・相互作用が片方向になるようにタスクの順序を決定
	4. 環境の予測	◎	◎	・環境に適応するよりも，環境変化への耐性が重要
	5. 上位レベルへの知的後戻り	○	×	
	6. 計画事例の検索と利用	△	×	
	7. 計画評価の多様性	○	○	・各評価基準を各タスクに分担
問題解決機能	1. トップダウン精密化戦略	◎	◎	・下位タスクは上位タスクの条件を崩さない
	2. 拘束最小化戦略	◎	◎	・上位タスクは下位タスクの条件を考慮しない
	3. 網羅的探索機能	○	○	・1, 2 と併用して役立つ
	4. 仮説推論による知的探索	△	△	・生成検査の枠組み内での仮設生成
	5. 類推による計画事例の利用	△	×	
	6. 計画の多目的評価	○	○	・各タスク毎に評価基準が異なる
	7. リスク下での意思決定	×	×	

担当：中島（富士通）

参考文献

［中島　87］　中島，集配スケジュール作成ＥＳにおけるグラフ閉路問題の性質と解法，第１回人工知能学会全国大会論文集，275-278，(1987)．

４．５　計画型システムの事例：電力系統計画・解析支援エキスパートシステム

４．５．１　システムの目的と機能

４．５．１．１　システムの目的

電力系統は，情報化社会を支えるエネルギー基盤として，システムの高い信頼性と経済性が要求されている．この要求を満たすには，綿密な系統解析に基づく系統建設計画や設備運用計画が不可欠である．これらの計画業務を省力化し，適切な計画案を能率良く作成できる環境を構築することが本システム開発の目的である．

４．５．１．２　機　能

電力系統の計画には，数年～２０年先の電力需要予測に基づく発電所や送変電設備の建設計画と，翌日～数年先にわたる発電機や調相設備の運用計画がある．いずれの計画においても，システム解析を通して得られる系統の安定性や信頼性，建設費用や運用費用から見た経済性などで計画の善し悪しが評価される．

例えば，運用計画では，夏のピーク時や深夜の軽負荷時などの特徴的な電力需要パターンを想定して，その時に稼働させるべき発電機とその出力，および調相容量の投入量などを上記の評価指標を考慮しながら決める．極く簡単に言えば，次の様な手順で計画を立てる．

(1) 系統条件を無視して総発電費用が最小となる様に稼働すべき発電機と発電機出力を決める．

(2) 上記の条件で系統内の定常的な電力潮流と母線電圧を計算し，その結果，送電線の過負荷や母線電圧の異常などがあれば，発電機出力や調相容量などを調整して異常を解消する．

(3) 以上で得られた運用状態が，主要な系統事故に対しても過渡的に安定であるか数値シミュレーションによりチェックする．

この計画策定の過程で，系統計画者はそれぞれのステップについて

①検討すべき解析条件を設定し，

②解析結果から問題点を発見して，

③対策案を立て，

④結果を評価する．

という事を繰り返しながら，高い評価指標を持つ計画案を探索する．本システムでは，現在，これらの計画過程のうち，(２)の過負荷や電圧異常の解消を知識ベースシステムの支援により，自動的に実行する機能を有する．

４．５．２　システムの構成と実現環境

４．５．２．１　ソフトウェアシステムの構成

本システムは，全体を統括するディレクタシステム，問題解決を支援する知識ベースシステム，解析プログラムのプログラムベース，そして系統データのデータベースから成る図 4.5-1の構成をしている．プログラミング言語としては，知識ベースシステムではC-Prologを利用している．他の解析プログラムおよびマンマシンインタフェースはFortran で記述している．実行時には，ディレクタシステム，知識ベースシステム，解析プログラムは各々マルチウィンド環境下の並列プロセスとして起動され，互いにメイル通信を交わしながら協調して計画，解析作業を支援する．

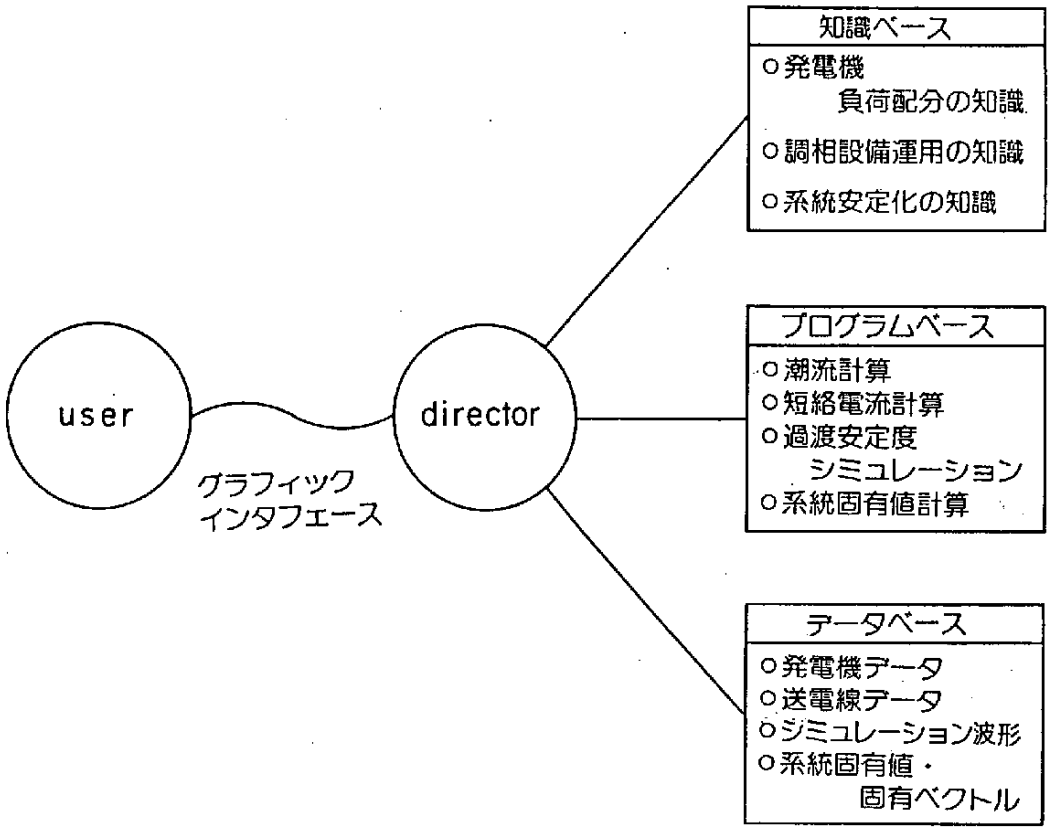


図 4.5-1　ソフトウェア構成

４．５．２．２　ハードウェアシステムの構成

電力系統図やアイコンメニューを中心とするマンマシンインタフェース，Prologで記述された知識ベース，系統の定常的な電力潮流や母線電圧を計算する潮流計算プログラムおよび系統のデータベースなど主要部分はワークステーション(Apollo)上で動く．大規模計算を必要とする系統事故時の過渡安定度シミュレーションと固有値解析計算については，系統データと共にホスト計算機であるVAX11/780 に自動転送され，計算結果だけがワークステーション上に表示される．このようにワークステーションを中心として補助的にスーパーミニコンピュータをリンクした図 4.5-2のハードウェア構成をしている．

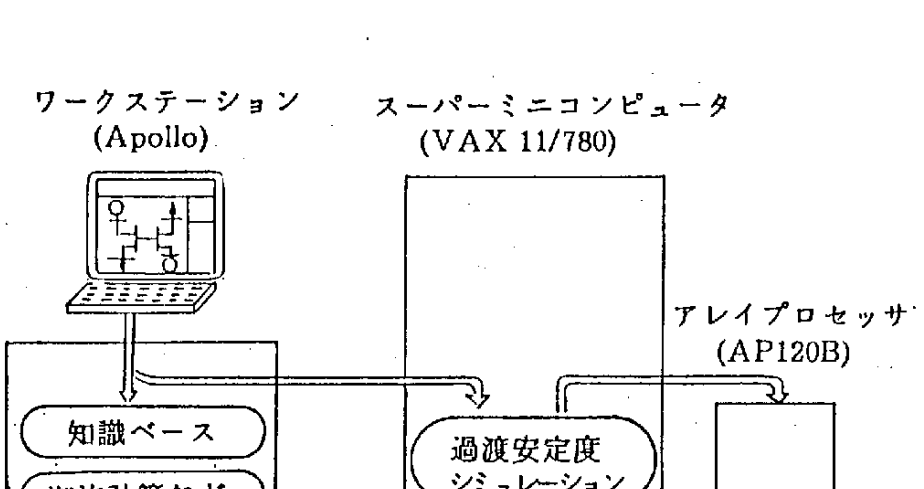


図 4.5-2　ハードウェア構成

４．５．２．３　知識ベースの規模

知識ベースシステムは前述のようにPrologで記述しているが，バックトラックとパタンマッチングの機能を利用して，後向きプロダクションシステムに近い構造をとっている．知識ベースのサイズは約１５０ルールで，マンマシンインタフェースや解析プログラムなどのFortran 部分を含めた全システムのサイズは約５万行である．

4．5．3　システム開発の手順

4．5．3．1　問題の設定

(1) 記号処理を中心とした問題であること

数値計算によって得られた電力潮流や母線電圧の状況を判断し，電力系統の定性的な因果関係に関する経験知識を利用して問題解決を行うものである．

(2) 専門家の経験知識が利用可能な問題であること

社内および電力会社の専門家は，日常業務としてこれらの問題解決を行っているので，この経験知識が利用できる．

(3) 十分なニーズがある問題であること

日常的な計画業務の省力化と高信頼化，および社会情勢の変化に即応した効率の良い計画立案体制の確立．

(4) 明確な価値を生じる問題であること

エンジニアリング業務の省力化，自動化．

(5) 保守が容易な問題であること

このプロトタイプシステムでは，Prologで直接コーディングしているため，必ずしも保守は容易ではないが，実用システムでは適切な知識表現言語を用いてユーザ自身による知識ベースの保守，拡張を可能にする．

4．5．3．2　既存技術の評価

問題を送電線の過負荷解消と母線電圧異常の解消に限定すれば，線形計画法や非線形計画法を用いて解くことも可能であるが，その場合には計画案作成の過程がブラックボックス化してしまうため，計画案の修正や変更に際して計画者の知識を生かして柔軟に対応することが難しくなる．また，計画の評価の段階では上記の条件の他に，系統の経済性，信頼性，安定性など多様な評価基準を考慮する必要があり，既存の手法で実現することは困難である．

4．5．3．3　知識源の同定

系統計画，運用計画の実用的手法に関する教科書やマニュアル類は全く無いので，複数の専門家の経験知識を利用した．

４．５．３．４　専門家モデルの同定

システム開発者自身が領域専門家でもあるので，開発者の知識を核にして，その上に他の専門家の知識を付加する形で知識ベース化した．

４．５．３．５　ユーザモデルの同定

専門家のルーチンワークを支援する．

４．５．３．６　知識表現の選択

既開発の対話型支援システムに知識ベースシステムを付加する形でシステムを構築したが，当初，ワークステーション(Apollo)上で動く知識処理言語が無かったので移植の容易さを考慮してC-Prologを採用した．また，知識ベースシステムがFortran でかかれた解析プログラムやマンマシンインタフェースプログラムとインタラクションする必要があるためメイル通信機能を持つ様に拡張した．

ただし，実用システムではエキスパートのノウハウに基づく動的な推論が中心となることから前向きプロダクションシステムを用い，手続き的な知識は出来る限りFortran で記述して高速化している．

４．５．３．７　知識の移植

ディスカッション，インタビュー，バーバルプロトコル分析などにより専門知識を抽出して知識ベース化した．

４．５．３．８　知識ベースの管理

説明機能(How) を用いて推論過程を検証することにより知識ベースの管理を支援している．

４．５．３．９　性能の評価

推論結果および推論過程の専門家による評価，バーバルプロトコルデータとの比較，数理計画法による解との比較などにより性能を評価している．

4．5．4　タスクと問題解決機能のまとめ

表 4.5-1　事例との関連表

	計　画　問　題	関連度	実現度	事　例　と　の　関　連
定義	計画問題とは，与えられた目標を達成するために利用可能な資源（人・金・もの）の割り当てを時系列として決定することである．	○		いくつかの代表的な時間断面について，複数の評価基準を満足するように制御量を配分する．時系列的なスケジューリングは含んでいない．
特徴	1.探索空間が非常に広い．	○	○	選択枝が多く，その制御量も決める必要がある
	2.評価属性が多様である．	○	△	経済性，信頼性，安定性，余裕など多数の評価属性があり，これらの重みは計画目的に応じて変わる．制約条件も状況に応じて変わり，例外もある．
	3.環境予測が不確実である．	×	×	いくつかの代表点について検討することにして不確実性は除いている．
	4.部分計画間の相互作用．	○	○	系統を地域的に分割して調整を進めるが，地域の境界が曖昧であり，地域間の相互作用もある
	5.既存計画の再利用．	○	○	既存計画をもとに，設定条件に応じた修正を加えて計画案を得る．
	6.費用と便益のトレードオフ．	○	△	制御設備の建設費用と信頼性その他との間のトレードオフ．
	7.対話型計画作成支援の要求．	○	○	全ての制約条件や評価基準をあらかじめ知識ベース化しておくことは難しく，計画者との対話が必要．

	計画問題	関連度	実現度	事例との関連
基本タスク	1.計画過程の階層化.	○	○	相互作用の有無による部分計画の分離とグローバルな調整からローカルな調整へという階層化が経験的に行われている.
	2.組合せ的探索.	○	○	地理的距離や感度などにより探索対象を絞り込む.
	3.制約条件の相互作用	×	×	
	4.環境の予測.	×	×	別システムで予測を行い，本システムでは，ある断面についてのみ計画する.
	5.上位レベルへの知的後戻り.	○	△	ローカルな調整で済まなければ再度グローバルな調整に戻る.
	6.計画，事例の検索を利用.	○	×	知識ベースとしての支援機能は無いが，計画例はデータベースに保存されている.
	7.計画評価の多様性.	○	○	
問題解決機能	1.トップダウン精密化戦略.	○	○	経験的に組み込まれている.
	2.拘束最小化戦略.	×	×	
	3.網羅的探索機能.	×	×	
	4.仮説推論による知的探索.	○	○	評価基準が多様であるため最適解が一意に定まらず，複数の候補を生成する.
	5.類推による計画事例の利用.	○	×	過去の事例から予め解の範囲を予測する.
	6.計画の多目的評価.	○	○	探索過程で複数の評価に効果のあるものを優先する．基本案生成後，更に評価の高い案を探索する.
	7.リスク下での意思決定.	×	×	

（福島正俊，協力：三菱電機㈱　藤原良一）

4．6　設計型システムの事例：ＬＳＩ配線設計システム　ＷＩＲＥＸ

4．6．1　システムの目的と機能

ＬＳＩの配線設計問題とは，

(1)　回路接続条件：論理的接続関係
(2)　幾何学的条件：製造プロセスにより決まるもの（配線幅，コンタクトの大きさ，配線同士の間隔，コンタクト同士の間隔，配線とコンタクトとの間隔）
(3)　電気的条件：遅延時間，雑音

上述の３つの条件を満たしながら，チップ基盤上に回路の配線パターンを決めることである。

この問題は難しい組合せ問題（ネットワーク理論における整数型多種フロー問題）に属しており，問題の規模（結線要求数）が大きくなれば，解くアルゴリズムの計算手数（計算時間と記憶容量）は指数関数的に大きくなる。このため，実用的規模の問題（数百以上の結線要求数）に対しては，アルゴリズム（自動配線プログラム）だけを使って，完全な配線パターンを求めることはあきらめねばならない。

したがって，現在のＣＡＤシステムは，自動配線プログラムと対話型の手動配線システムの両者を備えている。まず自動配線プログラムでできるだけ配線パターンを見つける（通常は98～99％の結線率）。結線できなかった信号線に対しては，専門設計者がディスプレイ上の配線パターン図を見ながら，パターンの修正と追加の作業を繰り返して，完全な配線パターンを実現する。

自動配線に要する時間は，小規模なＬＳＩでは数分であり、大規模なＬＳＩでも数時間以内ですむ。しかし，対話設計に要する時間は，高機能の対話システムを使っても，自動配線時間の10倍以上の時間を必要とする。全体の設計時間を短縮しようとすれば，いかにして，対話型設計に要する時間を少なくするかが，重要になってくる。

ＷＩＲＥＸシステムは，ＬＳＩ配線用に対話設計の効率化を目的として，開発したエキ

スパートシステムである。

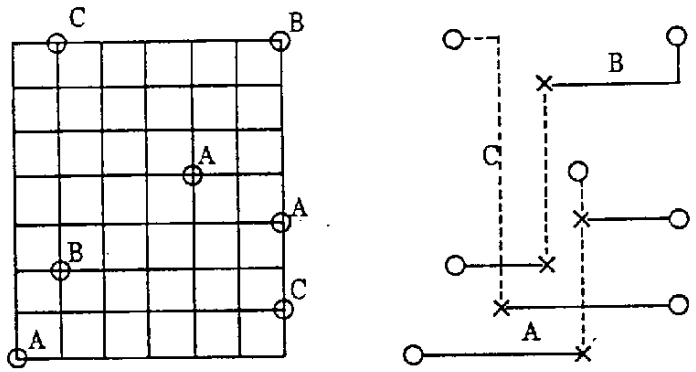


図４．６－１　ＬＳＩ配線問題の例

本システムは以下のような機能を持っており，実際の設計作業に使用されている。

(1) 専門家のノウハウを知識ベースに蓄え，非専門家が利用できるとしたために，非専門家でも専門家並みの高度な設計が可能となった。

本システムは，知識ベースの中に，各局面に応じた問題解決のための戦略をルールの形であらかじめ用意している。このため，配線状況に応じて，最も適当なルールをシステムが設計者に提案できる。設計者はシステムが提案するルール（配線戦略）の妥当性をチェックし，承認を与える，あるいは別の戦略を要求する，といったかたちで設計を進める。ディスプレイ上には，設計者が判断に必要とする情報が適切に表示されるために，非専門家でも専門家並みの高度な設計ができるようになった。

(2) ソフトウェア開発の専門家でないＬＳＩ設計者が，自分自身で設計ノウハウを知識ベースに格納できる。このためＬＳＩ設計者自身が設計対象に最適なＣＡＤシステムを容易に作り上げることが可能となった。

設計の対象となるＬＳＩのデバイス技術（ＭＯＳ，ＣＭＬなど）や構造（ゲートアレイ，ビルディング・ブロックなど），規模（1000～10万ゲート）などは，ユーザごとに異なっている。そのような相違点に応じて，配線設計の戦略が異なってくる。従来のＣＡＤシステムはこれに対し，あらかじめ用意した制御パラメータ変更程度しかできず，融通のきかないかたいシステムであった。また，従来のＣＡＤシステムに新しい機能を追加する場合には，新しい機能のプログラムはもとより，本体のプログラムの書換えに加え，コンパ入るとリンクという過程が常に必要であった。設計ノウハウの更新が頻繁に起こるシステムに対して，従来のプログラミング手法を用いることは，プログラム開発と管理の面から極めて効率が悪い。

本システムでは，プログラム作成に経験のないＬＳＩ設計者でも，設計ノウハウをルールの形で表現し，知識ベースに加えられる。設計者は設計を進めながら，ＣＡＤシステムを自分の問題に合ったものに変えられる。

４．６．２　システムの構成と実現環境

図４．６－２にハードウェア構成のブロック図を示す。ホストコンピュータは主記憶容量１６Ｍバイト，処理速度約４ＭＩＰＳのミニコン（日本電気ＭＳ１９０）である。チップ基盤の配線状況は２１インチのグラフィック・ディスプレイ上に表示し，ルールの説明のテキスト画面はＰＣ９８００パーソナル・コンピュータ上に表示する。

図４．６－３にソフトウェア構成のブロック図を示す。本システムは，推論システム，知識ベース，ＣＡＤシステム，および，ＣＡＤデータベースからなっている。知識ベースには，配線設計過程において設計者が用いる経験則を格納してある。具体的には，未結線端子付近の配線状況の把握，未結線となった原因の究明，未結線端子対の経路を発見するための操作などである。

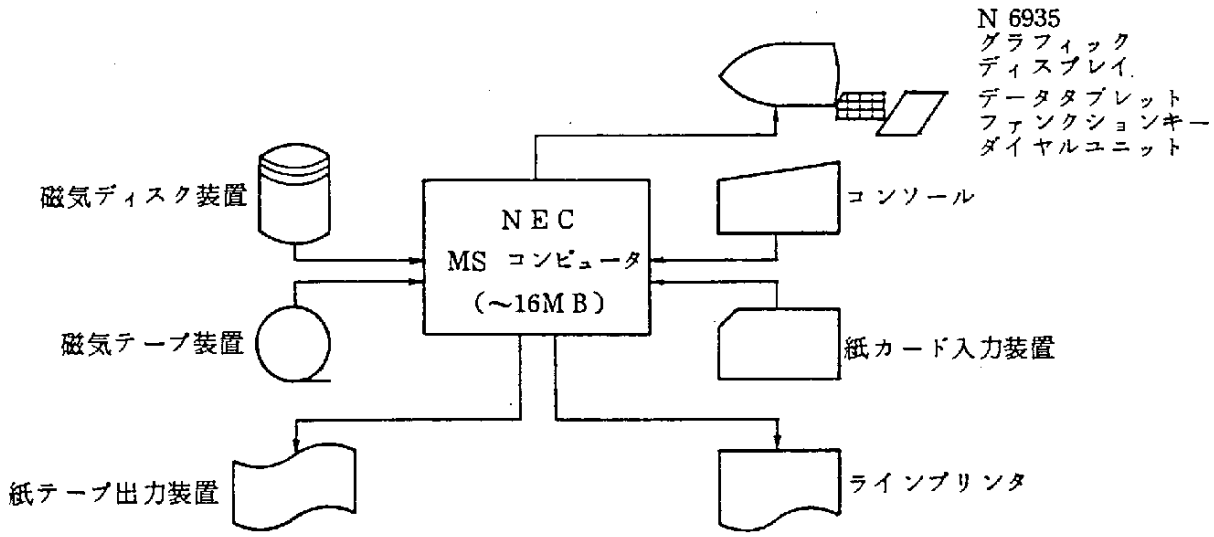


図4．6－2　ＷＩＲＥＸシステムのハードウェア構成

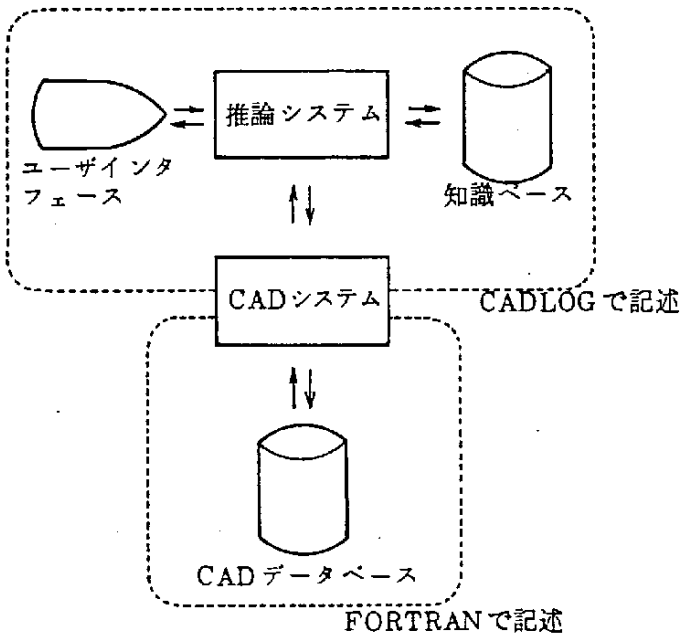


図4．6－3　ＷＩＲＥＸシステムのソフトウェア構成

一方，ＣＡＤデータベースは，従来のＣＡＤデータベースと同様のもので，配線設計用のデータ，たとえば，端子対を構成している端子の集合，これらの端子の座標，すでに結線した線分データなどを格納している。

ＣＡＤシステムは既存のＣＡＤシステムに用いているＦＯＲＴＲＡＮサブルーチン群からなる。これらのサブルーチン群はＣＡＤデータベースへアクセスするルーチン群，経路探索などの手続き処理を実行するルーチン群，およびグラフィックスを制御するルーチン群の３つに大別できる。これらのＦＯＲＴＲＡＮサブルーチンは，推論システム（ＣＡＤＬＯＧ）の外部ファッション述語に対応するサブルーチンとして，インタプリタに対応関係を登録してある。

推論システムは，Prologインタプリタ：ＣＡＤＬＯＧを用いている。このインタプリタは，既存のＣＡＤシステムと結合するために，PrologプログラムからＦＯＲＴＲＡＮサブルーチンを呼び出す機能をもっている。

４．６．３　システム開発の手順

”専門家の問題解決法を分析し，整理し，表現する。”という基本方針で，下図のような流れにそって開発を進めた。

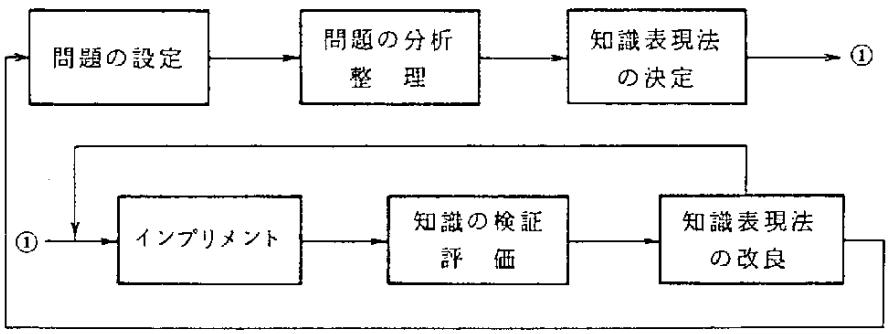


図４．６－４　開発の流れ

開発のステップを整理すると次のようになる。

（ステップ１）
- エキスパートシステム，ＣＡＤの専門家が実験的にいくつかのルールを作成する。

（ステップ２）
- ＣＡＤの運用の専門家が，従来の経験をもとにして基本的に必要な，かつルールとして表現可能な問題解決の手続き的知識を作成する。
- 同時に，状況判断の知識と情報表示のための知識をＣＡＤ専門家，エキスパートシステム専門家が作成する。

（ステップ３）
- ステップ２のルールの評価を行ない，改良，追加を行なう。

（ステップ４）
- エキスパートシステムの専門家から，システムは手が離れ，ＣＡＤ運用専門家が独自に知識を入れ，システムを運用する。

（森　啓）

4．6．4　タスクと問題解決機能のまとめ

	設計問題	関連性	実現性	コメント
定義	設計問題とは，システムの入出力の要求仕様が与えられたとき，これを実現するために，構成要素を組み合わせ，かつ各構成要素の内部仕様を決定することである。	○		配線設の場合，構成要素は配線であり，各配線の構造（幾何学的な経路）を決定すること。
特徴	1.システム構造の解空間が大	○	○	大規模組み合わせ問題である。
	2.解析による合成が基本的	×	×	アルゴリズムによる部分解の生成。
	3.構成要素の最適化の必要性	○	△	配線経路は短い方が望ましい。
	4.検証の完全性および効率性	×	×	
	5.対話形式での設計支援問題	○	○	システムの提案した方針の可否をオペレータが判断する。
	6.設計段階に応じた支援形態	×	×	設計の一つの段階を対象としている。
	7.対話型設計支援の要求	○	○	

	設計問題	関連性	実現性	コメント
基本タスク	1.構成要素とその関連の表現	○	○	いかにして要素間の関連を表現するかが重要である。
	2.設計問題の階層的表現	×	×	一階層のみを対象としている。
	3.代替案の自動生成	○	○	経路の候補，操作対象の候補の生成。
	4.部分システムの評価	○	○	部分解の評価。
	5.上位レベルへの知的後戻り	×	×	
	6.設計事例の検索と利用	×	×	
	7.並列的問題解決	○	×	複数経路を同時に考慮することが必要
問題解決機能	1.トップダウン精密化戦略	△	△	概略経路から詳細経路という意味でトップダウン
	2.拘束最小化戦略	×	×	
	3.最適化機能	○	△	最短経路発見アルゴリズムを利用
	4.検証機能	×	×	
	5.仮説推論による知的探索	×	×	
	6.類推による設計事例の利用	×	×	
	7.協調型推論	×	×	

４．７．設計型システムの事例：計算機室機器レイアウトＣＡＤ

４．７．１．システムの目的と機能

電子計算機システムのシステムエンジニアの業務の一部に、計算機システムを構成する機器の部屋内へのレイアウト作業がある。部屋の形状と配置すべき機器が与えられたとき、機器の部屋内での位置や機器相互間の位置関係などに関する制約を満足する適切なレイアウト図を作成する必要がある。適切なレイアウト図とは、数値的に与えられる条件というよりも、むしろ次のような意味的な条件を満足していることを指す。

(1) 機器同士は重なってはならない。

(2) 機器の保守エリアが確保されている。

(3) 機器前面がコンソールディスプレイ側から目視できる。

(4) 入出力機器は部屋の出入口の近くに配置し、ディスクなどは部屋の奥に配置する。

(5) 機器前面線がそろっている。

などである。

本エキスパートシステムは、このような意味的制約をもつ配置問題の計算機による処理を目的とするものであるが、そのためには次のような機能が要求される。

(1) 計算機に熟練した業務担当者の作業知識が備わっている。

(2) 配置状況や問題点を認識・理解することができる。

(3) 状況理解に基づいて適切な知識を利用し、レイアウト作業を進めることができる。

(4) 作業知識のうちには事前に定義できないものも多いため、ユーザがシステムを利用しながら知識を拡張することが可能である。

これらの諸機能を手続き形プログラミング技術によって計算機化することは困難であったため、レイアウト作業のうち本質的な部分は従来、人手に頼らざるを得なかった。

４．７．２．システムの構成と実現環境

本システムは、知識ベースと推論機構で構成している。知識ベースには配置候補地決定定石などを記述したルール形知識、機器属性などを記述したフレーム形知識、制約満足状況チェックや物体移動などを実施する手続き形知識を格納している。推論機構は、機器配置状況や制約満足状況を記憶するワーキングメモリと、ルールの選択・意味解釈・実行を行なう知識選択制御機構から成っている。システムに対して部屋形状、配置機器名称及び配置状況を強制的に指定したい機器があれば、その配置情報を入力すれば残りの機器については、システムがワーキングメモリ内の状況を理解しながら適切な知識を呼び出して利用することを繰り返して自動配置を行なう。なお、一部の知識、特に機器配置ルールについては、簡単な図形入力によってルールを自動生成する機能を保有している。

知識ベース

知識の大部分はＬＩＳＰのＳ式で記述している。

(1) ルール形知識

(2) フレーム形知識

(3) 手続き形知識

などから成る。

推論機構

(1) ワーキングメモリ

ワーキングメモリでも、フレームを利用している。レイアウト問題を解かせるとき、最初に部屋形状などを入力する。

レイアウト問題を解く過程では、計画状況を示すフレームが作成される。問題の解は、個々の機器に対する配置決定の連鎖で表わしている。各々の決定内容もまた、フレームとして記述されてゆく。与えられた一連の制約の満足状況評価結果なども記載される。

2) 知識の選択制御機構

ルール選択機構は、ワーキングメモリ内に記憶される配置状況を認識・理解し、その状況で次に実行すべきルールを知識ベースから選択する。ルール左辺のif部がワーキングメモリ内の状況に対して真となるものが選択の対象となる。状況判断のために、例えばＣＲがＣＤの右にあることを認識・理解するための手続き形知識が利用される。

ルールの意味解釈・実行機構は選択したルールの右辺のthen部を実行処理する。処理の多くは手続き形知識の利用を伴う。機器配置位置決定ルールの右辺の実行によって、「右、左、近く……」などの自然語的記述の意味解釈に基づく配置可能範囲の生成、配置決定、制約満足状況の分析、結果のワーキングメモリへの記入などが実施される。配置不可能な場合には、他の可能範囲の探索、既配置物のスライドなどを実施するルールが呼び出される。一つのルールの実行の結果、ワーキングメモリの内容は変化し、新しい状況に応じたルールが探索され、実行される。これによって、ダイナミックに変化する状況に応じて作業を進めるという、人手作業にみられる決定的過程を計算機化している。

4.7.3. システム開発の手順

項番	項　　目	コ　メ　ン　ト	備　考
1	問題の設定	計算機室内の適切なレイアウト図を作ること。 適切なレイアウト図とは、以下の意味的条件を満たすこと。 (イ)機器同士は重なってはならないこと。 (ロ)機器の保守エリアが確保されていること。 (ハ)機器前面がコンソールディスプレイ側から目視できること。 (ニ)入出力機器は部屋の出入口近くに配置すること。 (ホ)機器前面線がそろっていること。 ……など	
2	既存技術の評価	これまでのレイアウト図作成。 (イ)計算機に熟練した業務担当者が実施。 部屋の形状、機器配置の優先度、将来の拡張など、エンドユーザの希望を考慮しながら、”レイアウトの定石”を適用。 結果的に従来の手続き型プログラミング技術でレイアウトＣＡＤ化は困難だった。 (ロ)ＣＡＤ化への試行。 (ｉ)レイアウト問題に記号処理を導入。 (ii)レイアウト対象の属性の記述にフレームを導入。 (iii)熟練者知識を導入 —— 無意味な解の探索を防止。	
3	知識源の同定	知識の集合。 (イ)配置候補地決定定石知識。 (ロ)機器属性。 (ハ)制約満足状況チェック、及び物体移動実行知識。 ……など	

項番	項　　目	コ　メ　ン　ト	備　考
4	専門家モデルの同定	(1)レイアウト図作成手順（大工程)。 (イ)問題の定義。 (i)部屋の定義：形状、出入口、柱などの位置、大きさ。 (ii)配置物の定義：機器名称、数量など。 (ロ)配置物の位置の決定。 (i)配置物の位置：位置、オペレーション上の隣接関係。 (ii)配置物の方向：設置方位。 (2)配置物の位置の決定に関する作業手順。 (イ)現時点での配置状況を整理・確認し、次の作業(配置物)を決定する。 (ロ)配置候補地(範囲)を設定する。 (ハ)候補地で制約を満たす配置位置と方向を決める。 (ニ)配置不可能な場合、配置物をずらし、スペースを作り、配置を試みる。 ((イ)～(ニ)の繰り返しととらえることができる)	
5	ユーザモデルの同定	レイアウト図作成者をユーザと考えている。 従って、専門家モデルと同一と考える。 (エンドユーザは計算機オペレータと考えることもできる。レイアウト作成では、これらオペレータの動作を熟知した上で、作図しているとのこと)	
6	知識表現の選択	(1)配置候補地に関する定石的知識、作業手順 プロダクション・ルール (2)配置状況や機器の属性 フレーム (3)重なり具合などのチェック、面積の計算、物体移動など プロダクション・ルール	

項番	項　　目	コ　メ　ン　ト	備　考
7	知識の移植	知識表現の枠組みに従って知識を記述し、支援ツールで蓄積。 ただし、配置に関する定石的知識をすべて事前に決定しておくことは困難。このため、エキスパートがレイアウト作成することにより、 ルールの生成と知識ベースへの追加を自動的に実行する機能を具備。	
8	知識ベースの管理	知識ベースの無矛盾性のチェック、知識の世代管理等は、エキスパートによる人手である。	
9	性能の評価	従来人手により５～８時間かかって作成されていたレイアウト図が、３０分程度で作成可能。 （本ツールのレスポンスについては未評価）	

4.7.4. タスクと問題解決機能のまとめ

項目	設計問題	関連度	実現度	コメント
特徴	1)システム構造の解空間が大	◎	○	解を得るための戦略知識を用いる。
	2)解析による合成が基本的	○	○	
	3)構成要素の最適化の必要性			
	4)検証の完全性および効率性	×	×	
	5)対話形式での設計支援問題	◎	○	
	6)設計段階に応じた支援形態	○	◎	一つの実現解を得るための対話を実施。
	7)対話型設計支援の要求	◎	○	
基本タスク	1)構成要素とその関連の表現	◎	◎	
	2)設計問題の階層的表現	◎	○	解を得るための戦略知識を基本とする。
	3)代替案の自動生成	○	○	実行可能解を得るため、制約条件の強弱を考慮する必要がある。
	4)部分システムの評価	○	○	
	5)上位レベルへの知的後戻り	○	○	再配置を繰り返しながら'後戻り'を実施。
	6)設計事例の検索と利用	○	×	
	7)並列的問題解決	○	×	
問題解決機能	1)トップダウン精密化戦略	◎	○	「配置候補地から配置決定」を繰り返す。
	2)拘束最小化戦略	◎	○	解不能ケースを減少させるため、重視。
	3)最適化機能	○	×	
	4)検証機能	×	×	
	5)仮説推論による知的探索	○	×	今後、強化すべきと考えられる。
	6)類推による設計事例の利用	○	×	
	7)協調型推論	○	×	今後、強化すべきと考えられる。

(千吉良英毅　記)

参考文献

1)小林重信;多目的意志決定-理論と応用-VI,システムと制御,vol31No4,pp275-285,1987

2)渡辺俊典 他;知識工学の計算機室機器とレイアウトCADへの応用,日立評論,vol67No12,1985

－禁無断転載－

昭和６２年９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

６２－Ａ００１