設計と計画の統合的関連性支援
に関する研究

西岡 靖之

Kodak Color Control Patches

Blue Cyan Green Yellow Red Magenta White 3/Color Black

© Kodak, 2007 TM: Kodak

Kodak Gray Scale

A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19

© Kodak, 2007 TM: Kodak

博士学位請求論文

# 設計と計画の統合的問題解決支援に関する研究

西岡 靖之

指導教官　堀 浩一 助教授

東京大学大学院 工学系研究科 先端学際工学専攻

1996 年 3 月

# 論文内容の要旨

## 研究の目的と特徴

製造業におけるモノ作りは、現在大きな岐路にあり、より知識集約型に移行することが求められている。そのためには、モノ作りという行為を根本から再定義した上で、そこでの知識の効率的な管理を追求するとともに、計算機による有効な支援の方法を確立する必要がある。本研究は、製造業におけるモノ作りを１つの問題解決の行為として認識し、この問題解決を計算機によって支援する枠組を開発することを目的とする。特に、製品開発における製品設計と製造計画という２つの異なる問題を統一的な枠組みの中で扱うことに重点を置く。これによって、より柔軟で効果的なモノ作りの実現が可能となることを示し、今後の情報社会におけるモノ作りの１つのあり方として本研究を位置付ける。

本研究では、設計と計画という従来は異なる領域にあった２つの問題解決を、統合的に扱うための知識表現を開発する。設計が空間的な構造を、計画が時間的な構造を主に対象とする点に注目し、問題構造を空間的な構造と時間的な構造の両面から記述できる統一的な表現形式を定めることで、設計と計画の統合的な問題解決のための下地をつくる。さらに、問題解決の枠組みとして、問題そのものの明確化から始め、そこで明らかとなった問題構造を、時間的な構造と空間的な構造を同時に考慮しながら、人間と計算機が協調して解を求める方法を提案する。ここで提案する問題解決の枠組みは、計算機援用システムの開発を通してその有効性を示す。

本研究の特徴は、(1) 関係指向の知識表現を定め、問題解決に利用した点、(2) 形状とプランを同時に計算できるアルゴリズムを開発した点、そして、(3) 現実の製品開発の事例への適用により手法の有効性を検証した点の３点である。まず、問題解決を計算機上で扱うために用いた知識表現形式は、空間的な構造と時間的な構造を扱える関係指向の表現形式としたが、この知識表現は、表現対象の広さ、計算可能性、知識の利用効率などの点で従来の知識表現に勝ってる。また、形状とプランを同時に計算できるアルゴリズムは、従来のプランニングが対象としていた状態遷移と、幾何制約などに関する制約充足とを統合したものであり、本研究が新たに提案するものであ

る。さらに、これらの知識表現およびアルゴリズムを用いた問題解決の枠組は、現実規模の問題によってその有効性を検証しており、従来の特に人工知能の研究がトイプロブレムに終始していることを考えると、このことは本研究の大きな特徴といえる。

## 本論文の内容

本研究では、問題とは人間のもつ要求と現実世界の事実とのギャップであると定義し、問題解決をこのギャップを埋める行為とみなす。本論文ではまず、このような定義に従って、問題解決という行為を形式化し、本研究の対象を明らかにする。ここで、問題解決の対象である問題の表現は、本研究で独自に定めた知識の表現形式を用いて行なう。本研究の知識表現形式では、状況や視点に依存しない最小粒度の表現単位をプリミティブ、状況や視点を伴い蓄積と再利用に適した表現単位をテンプレートとし、この２種類の単位によって問題解決に関係するすべての知識を表現する。プリミティブはさらに、実体を表わすエンティティ、行為を表わすプロセス、そして、静的関係を表わすリレーション、動的関係を表わすオペレータの４つのクラスによって構成される。リレーションとオペレータは静的または動的な関係式を持っており、これらをもとに、計算機は問題を解釈する。なお、テンプレートはプリミティブを要素とするグラフ構造として表現される。

問題解決の対象を計算機上で表現可能とした上で、本論文では、人間と計算機とが協調的に行なう問題解決の枠組みを提案する。提案する問題解決の枠組みは、問題明確化、制約処理、可視化と評価、そしてトップダウン精緻化という各処理単位に分けて説明する。論文では、提案する枠組みによって、空間的構造と時間的構造の統合と、トップダウンな問題構造の精緻化が、人間と計算機とのインタラクションによって可能になった点を紹介する。また、本研究のアプローチは、問題解決の自動化よりはむしろ、問題解決に関する知識の効率的な利用を意識しており、提案する枠組みが、知識の効率的な管理という点でも優れていることを示す。

本研究の問題解決の枠組みの中で、制約処理は、制約充足計算と状態遷移計算の２つから構成されている。この２つの計算モジュールは、前者が静的な関係構造を主に対象とし、後者は動的な関係構造を主に対象とする。ただし、この２つのモジュールは、お互いに補完関係にあり、制約充足のためには部分的な状態遷移を必要とし、状態遷移のためには部分的な制約充足が必要となる。空間的構造と時間的構造の同時決定は、この２つのモジュールを適宜切替えることによって実現される。本論文では、例題によって、この制約処理のしくみを説明するとともに、提案する枠組みに合わせて開発された計算機プログラム PICCSS (Problem Interactive Clarification and Concurrent Solving System) によって、その動作を確認する。



本論文ではさらに、提案する問題解決支援の枠組を、現実の製品開発の事例に適用し、その有用性と実現性を確認した結果を報告する。対象とした製造業は、ポンプの製造を行なう小規模のメーカーであり、典型的な加工組立型の機械産業である。あらかじめ製品開発に必要な知識を整理し、必要なプリミティブを計算機上に準備した上で、それらの知識を用いて、新規開発のシミュレーション実験を行った。PICCSS を用いた実験の結果、設計者が行うトップダウンの製品開発の過程に対し、提案する枠組みが有効に機能することが確認された。論文では、PICCSS が、設計者の問題解決をさまざまな面から支援し、最終的な解を、製品形状に関する情報と工程プランに関する情報として出力したようすを示す。特に、PICCSS は、製造可能性を考慮した問題構造の精緻化の過程を、計算機が柔軟に支援でき、さらにまた新規設計以外にも、改良設計や設計変更において有効な支援ツールとなり得ることを、実験の過程を通して説明する。

## 本研究の工学的成果

本研究の工学的な成果としては、以下の３点が挙げられる。まず、第１の成果は、設計と計画を統合する知識の表現形式を提案したことである。本研究の知識表現は、設計が主に対象とする空間的な構造と、計画が主に対象とする時間的な構造を、統合的に扱うことが可能である。これに対して、従来のさまざまな知識表現は、静的な関係構造と動的な関係構造のどちらか一方に重点がおかれているか、あるいは両者の区別を曖昧にしたまま扱っている。本研究の知識表現はさらに、知識レベルの表現と、記号レベルの表現がきちんと対応づけられいる点、そして、より汎用的な知識をその他の知識と明確に区別することで、知識の効率的な利用を実現している点も、重要な特徴となっている。

工学的な成果の第２は、時間概念を付加した制約処理アルゴリズムを開発したことである。制約充足問題に代表される従来の制約指向のパラダイムに対して、本研究では、時間概念を取り入れた新たなアプローチの有効性を示した。ここで時間概念とは、行為による状態遷移によって成立するものである。このようないわば動的な時間に対して、従来の多くの研究では静的な時間、つまり、パラメータとして定義可能な時間を扱っている。本研究では静的な時間と動的な時間をともに扱っているが、単に制約関係の延長として静的な時間を扱ってきた従来の制約処理の研究を一歩進め、制約充足計算と状態遷移計算とを組あわせたアルゴリズムにより、特に動的な時間に関して、その位置付けを問題解決の枠組の中で明確にした。

第3の工学的成果としては、悪定義問題に対する問題解決方法を提案し、その有効性を示したことが挙げられる。本研究では、あらかじめ問題そのものが定義できない悪定義問題に対する、精緻化によるトップダウンな問題解決の方法を提案した。提案した方法は、従来のプロトタイピング手法などと異なり、問題そのものが明確でない状態から計算機上で処理を行う。つまり、問題構造の明確化の過程を計算機上で行う点に大きな特徴があり、このような、従来は人間の概念世界の中で行なってきた処理を、計算機上で効率的に行なう研究は少ない。特に本研究では、計算機内部に表現された部分的な知識に対して関係するより粒度の細かい知識を必要に応じて追加するというトップダウン精緻化を検討し、これを悪定義問題に対する有効なアプローチの1つとして位置付けることができた。

### 結論と今後の課題

本研究の以上のような工学的な成果は、製造業の現在のモノ作りにおいて、既存知識の有効利用による設計作業の省力化、ラフな製造プランの生成による問題の早期確認、製品開発の上流における計算機能力の有効利用、製造環境の変化に対する製品開発のダイナミックな対応、そして、創造的な製品開発のための環境整備への貢献、といった点で非常に期待できる。提案した方法の実用化を通して、本研究の成果を、製造業における知能情報処理技術として確立することが可能であるとの見通しが得られた。

また、今後の製造業のモノ作りにおいて、生産者と消費者の関係、企業組織と個人の関係などがますます多様化し、従来のモノ作りの枠組みの中ではとらえきれない状況が訪れた場合、モノ作りを1つの問題解決の行為として認識した本研究の枠組みは、製造業の新たな1つの方向性と可能性を示すことだろう。本論文では、モノ作りに対する新たな視点を提供し、具体的な問題解決の枠組みおよび支援システムによって、次世代へ向けての情報技術を積極的に活用した新たな製造業のあり方を示唆することができた。

今後の研究課題としては、要求展開の自動化、テンプレートの事例からの獲得、深い探索への対応、CAD/CAM、CAPP (Computer Aided Process Planning) システムとの連動、グループ問題解決への拡張などが挙げられる。

## 関連する発表論文

### 論文

- 西岡靖之,『状況対応型スケジューリングシステムにおけるペナルティ伝播グラフを用いた計画修正方法』, 経営システム, Vol.5, No.3-4, pp.263–271, 1995 年

### 論文 (レフェリーなし)

- 西岡靖之, 中須賀真一, 堀浩一,『設計問題と計画問題の統合的問題解決支援システム』, 人工知能学会シンポジウム, 人工知能学会研究会資料 SIG-J-9501-11, pp.72–79, 1995 年 12 月, 東京

### 国際会議発表

- Y. Nishioka, T. Terano, "An Approach to Reactive Scheduling Problems Using Constraint Propagation," First Singapore International Conference on Intelligent Systems, pp.231–236, Sep. 1992, Singapore

### 学会口頭発表

- 西岡靖之, 寺野隆雄,『制約伝播を利用した再スケジューリング手法』, 人工知能学会全国大会論文集, pp.421–424, 1992 年 6 月, 東京
- 西岡靖之,『ダイナミックな生産環境における状況対応型スケジューリング問題の解法』, 生産スケジューリングシンポジウム講演論文集, pp.158–163, 1993 年 12 月, 名古屋
- 西岡靖之,『次生産システムにおける情報の意味について — 生産のための組織知能を自己組織性 —』, 経営情報学会秋期全国大会発表要旨, pp.33–38, 1993 年 11 月, 埼玉

- 西岡靖之, 『モノづくりを介した組織の意味創造 － 次世代生産システムのコンセプト －』, 経営情報学会春期全国大会発表要旨, pp.121–124, 1994 年 5 月, 東京
- 西岡靖之, 堀浩一, 大須賀節雄, 『知能のダイナミクルにおけるプロセス知識の利用に関する研究』, 人工知能学会全国大会論文集, pp.11–14, 1994 年 6 月, 東京
- 西岡靖之, 堀浩一, 大須賀節雄, 『製造業に期待される新設計パラダイム』, 設計シンポジウム講演論文集, pp.7–12, 1994 年 7 月, 東京
- 西岡靖之, 堀浩一, 大須賀節雄, 『ボトムアップダウンの生産組織を可能にするプロセス知識可視化ツールの試作』, 経営情報学会春期全国大会発表要旨, pp.213–216, 1994 年 10 月, 京都
- 西岡靖之, 堀浩一, 大須賀節雄, 『コンカレント・エンジニアリングとスケジューリング：一品生産のための設計要求を考慮したスケジューリング方法』, 生産スケジューリングシンポジウム講演論文集, pp.67–72, 1994 年 10 月, 東京
- 西岡靖之, 中須賀真一, 堀浩一, 『コンカレントな問題解決のための製造プロセス知識の獲得と再利用』, 人工知能学会全国大会論文集, pp.479–482, 1995 年 7 月, 東京


### 解説論文


- 西岡靖之、『状況対応型スケジューリング問題へのアプローチ』, 計測と制御, Vol.33, No.7, pp.571-574, 1994 年


## 謝辞


本論文がこうして形となるまでには、非常に多くの方々からの御指導、御援助を頂いた。まず、大須賀節雄名誉教授には、博士課程の最初の 2 年間、指導教官として御指導を賜わった。また、東京大学退官後も、数多くの御助言を頂き、それらのさまざまな御指導のもと本論文の骨格ができあがった。研究内容は勿論のこと、常に現役であり続ける大須賀名誉教授の研究への情熱の中から、ことばにならない非常に貴重な多くのことを学んだ。感謝の辞を申し上げる。

また、堀浩一助教授には、特に心より感謝の辞を申し上げたい。堀助教授には、入学当初から、方向の見えない本研究を辛抱強く見守って頂き、常に大局的な視点からご指導頂いた。研究が行きづまり方向性を見失いかけたときも、的確な御助言とともに終始本研究に理解と強い興味を示して頂いた。このことが本研究を進める上での大きな支えとなった。また、博士課程の最後の 1 年は、指導教官として、最終的な論文の構成と内容について多くの御指導を頂いた。さらに、本論文の内容を細かな点までチェックして頂き、多くの貴重な御意見を賜わった。心より御礼申し上げる。

さらに、中須賀真一助教授には、本論文の具体的な内容について、非常に有意義な御指導と御助言を頂いた。本論文で提案したいくつかのアルゴリズムは、中須賀助教授の御指導をきっかけとして生まれたものが多い。また、中須賀助教授には、本論文の審査委員として、全体の構成から細かな点まで多くの御助言を頂いた。中須賀助教授の熱意ある御指導に深く感謝したい。

木村文彦教授には、特に、生産システム全般に関する研究から個々の研究に至るまで、常に最先端の研究動向を教えて頂いた。また、本論文をまとめるにあたり、要所要所で、的確な御助言と多くの御示唆を頂いた。また、木村教授には、本論文の審査委員としても、御指導、御鞭撻を賜わった。本論文が生産に関する研究分野で少しでも価値のあるものであるとすれば、それは木村教授のお蔭である。深く感謝したい。

中島尚正教授と河内啓二教授には、本論文の内容および基本的な方向性について、非常に本質的な多くの御意見を頂いた。中島教授には、設計問題のあり方について、また、河内教授には、本論文の工学的な位置付けについて、貴重な御示唆を頂いた。

また、両教授には、本論文の審査委員にもなっていただき、多くの御指導を賜わった。感謝の辞を申し上げる。

本論文の背景にある問題認識は、精密機械工学科の冨山哲男助教授、人工物工学専攻の田浦俊春助教授、久保田晃弘助教授、そして桐山孝司助教授のさまざまな御指導によところが大きい。特に田浦助教授には、多くの御助言を頂いたとともに、エンジニアリングにおける「知」に関する研究会を通して、さまざなま工学的問題と直接接する機会を与えて頂いた。研究会では、鯉淵興二教授、都立科学技術大学の福田収一教授、東京工業大学の伊藤公俊助手、そして高武淳夫氏より有益な御助言を頂いた。この場を借りて御礼申し上げる。

本論文がまだ形となる以前の段階では、多くの方々との議論が大変有益であった。特に、経営情報学会の post-MRP ならびに post-JIT 研究部会では、主査である神奈川大学の松浦春樹教授をはじめ、手島歩三氏、内山研一氏には、何度も議論して頂いた。本論文がより現実的な方向でまとめることができたのは、そこでの貴重な議論のお蔭であり、感謝の辞を申し上げる。

また、本論文をまとめるにあたり、社会学的な視点として、HCS ワークショップを通じた多くの方々の御意見が大変参考になった。当時主査であった大東文化大学の林武教授をはじめ、相模女子大学の里深文彦教授、早稲田大学の土方正夫教授、NTT データ通信 (株) の佐藤桂弘氏、NEC の石黒広洲氏、(株) 東芝の谷内宏行氏、その他メンバー各位に感謝したい。

本論文の中で、スケジューリングに関する部分については、筑波大学大学院の修士課程において、鈴木久敏教授、寺野隆雄助教授に御指導頂いた。特に鈴木教授には、数理計画の分野における最適化の手法を基礎から御指導頂き、その内容が本論文の至るところで役に立っている。また、寺野助教授には、修士課程修了以降も、各論文の発表を強く勧めて頂いた上、その内容に至るまで御指導を賜わった。本論文をまとめる最終段階でも、人工知能の研究分野からの貴重な御意見を頂いた。心から感謝の意を表する。

スケジューリングに関する研究と、本論文の中心的内容である設計問題との結び付けは、日本オペレーションズリサーチ学会と日本経営工学会共催の COM のための生産計画・スケジューリング研究部会での多くの議論によって実現した。特に、主査である青山学院大学の黒田充教授には、製造業の全体的最適化を指向した魅力溢れる内容の御指導、御鞭撻を頂いた。また、同じく主査の東海大学の村松健児教授、早稲田大学の森戸晋教授、(株) 東芝の米田清氏に多方面からの御意見を頂いた。この場を借りて感謝する。

さらに、工業技術院電子技術総合研究所の宮下和雄氏には、生産管理に関する人工



知能的なさまざまな手法と、米国での研究動向について御教示を頂いた。また、大阪大学の岩田一明教授、小野里雅彦助教授、機会振興協会の福田好郎氏、そして大阪大学の池田満助手には、本研究の最後の段階で、非常に貴重な御意見を数多く頂き、その結果として本論文をこのように完成させることができた。ここに感謝の辞を申し上げる。

The Pennsylvania Sate Univ. の Prof. Soundar R. T. Kumara には、本研究の方向性を最終的に具体化する最も重要な時期において、半年に渡り、概念的な内容から具体的な方法論までを、懇切丁寧に御指導して頂いた。Prof. Kumara には、製造に関する非常に多方面の研究内容についての議論にも乗って頂き、そこでの議論を通して多くのアイデアを得ることができた。深く感謝し、御礼申し上げる。

また、木村研究室では、鈴木宏正助教授、産能大学の松田三知子助教授、東京電機大学の田中一郎講師、安藤英俊氏、九州工業大学の吉川浩一客員助教授、博士課程の尹泰聖氏より、CAD 分野に関する多くの御教示を頂いた。特に尹氏には、本研究に関して数多く議論して頂き、そこから多くの示唆を得ることができた。また、秘書の筧麻子氏には文献等の調査でお世話になった。感謝の意を表したい。

本論文の実験においては、ダイナフロー (株) の西岡洋一氏、泉谷正雄氏に多大な御協力を頂いた。非常に御多忙な業務の合間に、さまざまな質問に御回答頂いた上、貴重な多くの情報の提供を御快諾頂いたことに、心より感謝し、御礼申し上げる。

知能工学研究室での 3 年間の研究活動を進めるにあたっては、山内平行助手と秘書の二木晶子氏に、非常にお世話になった。山内助手には、不自由ない研究環境を整えて頂いた上、計算機プログラミング上の多くの相談にものって頂いた。また、二木氏には、陰で研究室での活動を支えて頂いた。この場を借りて、心より御礼申し上げる。

研究室の諸先輩、同輩、後輩諸氏には、日頃の研究活動やその他公私の全ての面に渡ってお世話になった。特に同輩の土橋喜氏、渡辺光一氏、春木良且氏は、研究のよきライバルであると同時に、最も信頼できる友人でもある。研究を含めた多くの相談にいつものって頂いた。また、工藤隆司氏、内藤祐介氏、楠房子氏と行なったさまざまな議論も、研究活動をより幅の広いものにしてくれた。また、高田昌之氏（現電機通信大学）、矢野新一郎氏（現川崎重工業）、角康之氏（現 ATR）、杉本雅則氏（現学術情報センター）の諸先輩には、研究者としてのよき手本を示して頂いた。さらに、計算機環境その他の面で、相原健郎氏、吉住英典氏、吉増大氏、田中伸彦氏に非常にお世話になった。この場を借りて、感謝の意を表したい。

最後になったが、3 年間のわかままを暖かい目で見守り、そして陰ながら支えてくれた多くの友人と家族、その中でも特に両親に心より感謝したい。

# もくじ

# 図一覧

# 表一覧

# 第 1 章

# 序論

## 1.1 はじめに

人間の知的活動として“モノ作り”が挙げられる。ホモファベルということばがあるように、人間はモノを作る動物である。モノ作りを通して人間は知能を発達させてきたといってもよい。モノ作りの結果として生まれた人工物は、モノ作りの知能と密接に関わりあっている。人間が今までに作ってきたさまざまな人工物の変遷は、過去から現在に至る人間の生活様式の変遷を物語っているが、それは同時に、モノ作りに関する人間の知能の変遷をも物語っている。

ここでモノを作る知能とは何であるかを考えてみたい。モノ作りは広い意味で人間が行う問題解決の一種である、というのが本研究における一貫した立場である。つまりモノ作りは、人間のモヤモヤとした要求に対し、それを具現化するために人工物を生みだす行為として位置付けられる。従って、モノを作る知能とは、このような行為のための、さまざまな経験と知識とからなる総合的な能力のことであるといえよう。

現実を見渡せば、現代のモノ作りの大半は製造業によって営まれており、そこでは、さまざまな物と情報の習慣的な流れと、そして、その各所で利用される膨大な量の知識が存在する。一方、個人で行なうモノ作りも、さまざまな経験と知識によって各所で行なわれており、一見してまったく形態の異なるこの２つのタイプのモノ作りも、突き詰めて考えてみると何らかの共通点があるようにも思える。

本研究は、モノを作る知能の本質に少しでも近付きたいという動機から出発した。そのためにまず、現代のモノ作りの中心的な役割を担っている製造業をとりあえずターゲットとし、そこで行なわれているさまざまな行為を、知能情報処理的な視点から理解しようと試みた。現代のモノ作りは、ある意味で手段が目的化している状況であるともいえ、この状況を脱するためには、「モノづくりとはなんぞや？」という疑問に、ある程度形式的に答える必要があるのではないかと考えたことも、本研究の大きな動

機となっている。

序論では、まず最初に、本研究の背景にある製造業をとりまく状況認識と、関連する研究の大まかな動向を確認した上で、本研究の主な目的を述べる。続いて、本研究の内容を簡単に要約し、主な特徴を説明する。また、本論文で用いられる用語のなかで、特に重要と思われるものについては、ここであらかじめ説明する。序章では、最後に、本論文の全体の構成を説明する。

## 1.2 研究の背景と目的

製造業におけるモノ作りでは、製品開発の過程において非常に大きな付加価値を生み出す。そこでは、製品の企画から、基本的な仕様の決定、製品の形状の決定、製造方法の決定、そしてさらにその製品の保守や廃棄のためのマニュアル作成など、さまざまな知的活動が行なわれる。このように考えると、製造業はむしろ、製品というモノを介して価値のある情報を生み出す情報処理産業である、という見方もできる。

次世代の生産パラダイムを多くの企業や研究者が模索するなか、これからの製造業は徐々に知識集約型へ移行していくという見方が多い [富山 94]。従来は非常に属人的であった製品開発や製造のさまざまな行為が、処理・伝達が可能な知識情報に徐々に置き換えられようとしている。特に、計算機によって処理可能な形式で知識を表現し、製品開発にかかわるできるだけ多くの行為を計算機上で仮想的に行なうことによって、開発期間と効率を飛躍的に向上させようとする仮想生産 (virtual manufactring) [Kimura94] などが注目されている。

このような知識集約型の生産へ向かうさまざまな流れは、モノそのものに価値があったハードの時代から、モノを生み出す方法にあたる知識が価値を持つソフトの時代への移行の表れであると見てよいだろう。ただし、そのような流れは、知識あるいは知能情報処理の技術的な進歩がともなってはじめて、現実のものとなる。

従来の CAD/CAM 技術に代表されるように、生産の自動化や無人化あるいは省力化は、非常に多くの情報処理技術の研究によって実現され、大きな成果をあげてきた。しかし製造業全体として見た場合、人間の行なう知的な作業が減っているかというと、計算機や機械によって代替可能な部分は非常に限定された一部分であり、特に製品開発の初期段階などは、まったく手つかずの状態であるといってもよい。

本研究では、このような従来は経験やノウハウにたよっていた製品開発の初期段階などに対し、そこで扱われている非常に整理しにくい知識を計算機上で扱うことによって、知識の獲得と再利用を円滑に行なうことができるような仕組みを考える。また、人間の最も基本的な行為である“問題解決”について、製品開発あるいはモノ作りと



いう行為に対しても当てはめられるように今いちどとらえ直し、そこで得られるより本質的な知見をもとに計算機による支援のあり方を検討する。

一般に製品開発では、“何を作るか”といったモノに対応する側面と、“いかに作るか”や“いかに使うか”といった行為や方法に対応する側面について、両者の関連性を維持しながら作業が進められている。ここで、前者つまりモノの側面を強く意識した問題を設計問題、後者つまり行為や方法の側面を強く意識した問題を計画問題と呼ぶことにすると、製品開発を含む非常に多くの問題は、設計問題と計画問題の統合的問題として認識できる。従って、この設計と計画の統合問題に対し、より効果的な問題解決の枠組みとそれに合わせた支援の方法を確立する必要がある。

本研究の目的は、製品設計と製造計画を統合する問題解決の枠組みを提案し、計算機援用システムの開発を通してその有効性を示すとともに、人間と計算機の関わり方を再定義することで、今後の情報化社会におけるモノ作りの１つのあり方を提案することにある。また、人間の行なっている問題解決が、設計という空間を意識した側面と計画という時間を意識した側面の２つの側面を同時に考慮することで、より柔軟なものとなっていることを、モノ作りという問題解決の例により示すことも本研究の目的の１である。

## 1.3 研究の内容と特徴

設計と計画の統合的な問題解決のために本研究で提案する方法を、“コンカレントな問題解決 (Concurrent Problem Solving)”と呼ぶことにする。ここで、コンカレントな問題解決とは、設計という空間的な構造を対象とする問題と、計画という時間的な構造を対象とする問題を、できるだけ分離せずに同時並行的に扱う問題解決方法である。コンカレントな問題解決を実現するために、本研究ではまず、設計と計画という従来は異なる領域にあった２つの問題解決を、統合的に扱うための知識表現を開発する。設計が空間的な構造を、計画が時間的な構造を主に対象とする点に注目し、問題構造を空間的な構造と時間的な構造の両面から記述できる統一的な表現形式を定めることで、この統合的問題解決の下地をつくる。さらに、この統合的問題解決の枠組みとしては、問題そのものの明確化から始め、そこで明らかとなった問題構造を、時間的な構造と空間的な構造を同時に考慮しながら、人間と計算機が協調して解を求める方法を提案する。特に、空間的な構造と時間的な構造との同時決定のために、制約充足計算と状態遷移計算を組み合わせた計算方法を考える。ここで提案する問題解決の枠組みは、計算機援用システム PICCSS (Problem Interactive Clarification and Concurrent Solving System) の開発を通して具体化し、現実の製品開発の事例に適用

することでその有効性を示す。

本研究の特徴としては、以下の3点が挙げられる。

**(1) 関係指向の知識表現を定め、問題解決に利用した**

本研究では、独自の知識表現方法を定めた。本研究で定めた知識表現は、関係指向であるという点に大きな特徴がある。ここで関係指向とは、要素の属性としての関係をとらえる従来の方法に対して、関係を要素から独立して定義し、要素は関係の集まりとしてとらえるという点に特徴がある。また、本研究の知識表現は、空間的な構造と時間的な構造の両方を扱える点、知識表現の内部に計算機が計算可能な記号レベルの表現を持っている点、そして、知識の利用効率を考慮し、オントロジーに相当する部分とそれ以外とを明確に分けている点が特徴である。

**(2) 形状とプランを同時に計算できるアルゴリズムを開発した**

本研究で提案する問題解決の枠組みは、問題の空間的な構造と時間的な構造とを同時並行的に決定する。そのために、問題解決の中の制約処理において、制約充足計算と状態遷移計算を組み合わせたアルゴリズムを開発した。ここで、制約充足計算は、問題構造に含まれる制約式をもとにパラメータの値を計算し、状態遷移計算は、問題構造に含まれる因果関係をもとにパラメータの値を変更する。本研究では、このアルゴリズムにより、製品開発において、製品形状と製造プランを同時に計算することを可能にした。

**(3) 現実の製品開発の事例への適用により手法の有効性を検証した**

従来の人工知能あるいは知識工学の研究では、対象とする問題を単純化した上で、問題解決の枠組みを議論してきたが、これらの研究の多くは、理論的な厳密性がある反面、現実規模の複雑さに対する検討が行なわれていない。本研究の大きな特徴は、提案する知識表現およびそれを用いた問題解決の枠組について、現実規模の問題に適用することを通して、その有効性を検証している点にある。本研究では、計算機プログラムとして PICCSS を開発し、提案する方法が現実の製品開発におけるさまざまな課題を解決する可能性があるとの見通しが得られた。

## 1.4 用語の定義

本論文で用いられている重要な用語について、ここであらかじめその定義を明確にしておく。以下に説明する用語は、本論文で特に独特な意味で用いられているもので



あり、本研究の内容を理解する上でのキーワードとなる。

**知能と知識 (intelligence, knowledge)**

本研究では、知能とは、人間の知的活動を生み出すしくみを指す。知能は、それ自身で目的概念を持ち自律的な存在となりえる。これに対して、知識とは、知能の一部を代替するものであり、何らかの表現形式にもとづいて表現されたものを指す。人間の行為は知能の一部といえるが、行為そのものは知識ではない。ただし、それらの行為を知識として記述することは可能である。表現された知識は、処理・伝達・蓄積が可能であるという性質をもつ。

**問題 (problem)**

問題とは、人間の概念世界にある要求と、現実世界にある事実との間のギャップを指す。もし要求と事実の間にギャップがなければ、その状態は“問題がない”あるいは“問題が解けた”状態であるといえる。問題には潜在的な問題と顕在化した問題があり、要求を分節し事実を認識することによって、潜在的な問題が顕在化する。顕在化した問題は要素とその関係によって表現が可能であり、顕在化した問題の持つこのような構造を問題構造と呼ぶ。

**問題解決 (problem solving)**

問題解決とは、潜在的な問題を顕在化させると同時に要求と事実との間の対応関係をとること、と定義する。ここで、要求と事実との間の対応関係をとるということは、両者の構造的あるいは数量的な差異をなくすことに相当する。この過程では、要求と事実とのインタラクションを通して、要求または事実そのものが変化する場合もある。本研究における問題解決のこのような定義の特徴として、あらかじめ問題そのものが明確となっていない状態から問題解決が始まり、問題解決の進行にともなってその構造が明らかになっていく点が挙げられる。

**解と解候補 (solution)**

問題解決によって設定された問題構造の中で、論理的矛盾を含まないものはすべて、問題の解候補となる。これに対して、問題の解は、これらの解候補の中で、要求と事実とが十分な粒度で表現され、かつ、両者の間のギャップが許容可能な範囲にあるものを指す。ここで、問題構造の粒度やギャップの判断基準は、最終的には問題解決者による主観的なものとならざるを得ないが、解としての必要条件として、その解を現

実世界に実際に適用させる際に、解の内容を実現可能な具体的な行為に直接的または間接的に置き換えることが可能でなければならない。

### 制約関係と因果関係 (constraint, causality)

本研究における制約という概念は、人間の概念世界にある要求と、現実世界にある事実の双方に存在し、問題構造の中の要素の関係を論理的あるいは数量的に規定するものである。従って本研究では、制約関係は数式によって表現される。制約関係が問題構造の静的な関係を表すのに対して、因果関係は、問題構造における各要素の動的な関係を表す。因果関係は、状態遷移を司るものであり、その結果として時間の進行を与える。

### 静的関係と動的関係 (static, dynamic)

静的な関係と動的な関係とは、ひとたび表現された知識としての構造あるいはパラメータが、問題解決の進行にともなって変化するかどうかを識別する際に用いる用語である。具体的には、制約式は静的な関係を表現し、代入式は動的な関係を表現する。問題構造の空間的な関係と時間的な関係の中にも、それぞれ静的な関係と動的な関係が存在する。

### 設計 (design)

設計の定義の例としては、“設計とは、人が概念として想定した要求機能を、それを充足する実体へと変換する行為である [吉川 85]” や “設計は、ひとが頭の中で考えたものを、実際の物の形にするためのすべての情報を作り出すことである [畑村 88]” などがある。一方、英語の Design は、形を強く意識したものであり、そこでは空間的な構造が議論される。本論文では、この形状または空間的な側面を特に強調し、設計とは、問題解決において空間的な構造を主に決定する行為、と定義する。

### 計画 (planning)

計画ということばは、本来非常に広い意味でもちいられており、場合によっては人間の意思決定全般を指すこともある。人工知能の研究における計画（プランニング）の定義の例としては、一般問題解決器（GPS: General Problem Solver [Newell63]）における “与えられた目標と初期状態との差異を減じる手段の中から適当な作用素を発見することにより、状態間の差をしだいに縮めていく問題 [安部 90]” などがある。本論文では、計画が特に行為の時間的な関係に着目していることをふまえ、計画とは、問題解決において時間的な構造を主に決定する行為、と定義する。



### コンカレントな問題解決 (concurrent problem solving)

コンカレントな問題解決ということばは、本研究で独自に用いている用語であり、設計と計画と統合した問題解決のことを指す。本研究では、コンカレントな問題解決とは、人間の持つ要求を明らかにすると同時に現実世界の事実関係を明らかにし、それらを対応づける空間的／時間的な関係構造を定めること、と定義する。このコンカレントな問題解決とその他の問題解決との違いは、空間的構造の決定と時間的構造の決定の同時性にある。

## 1.5 論文の構成

本論文は、図 1.1 に示すように、大きく 3 つのパートに分けることができる。この序論に引続き、まず第 1 のパートでは問題の定義を行なう。ここでは、“問題” とは何かを明らかにするとともに、そこで明らかにした問題を知識工学的に表現する方法を提案する。そのために、第 2 章「対象とする問題」でまず製造業におけるモノ作りを考慮しながら、問題解決の一般的な定義を行ない、特に設計と計画の統合的な問題解決について、その概念を明らかにする。また 3 章の「知識の表現方法」では、対象とする問題を表現するための基本的な表現形式を説明し、それを用いた知識表現の方法を示す。

第 2 のパートでは、問題の解法について述べる。ここでは、問題のより一般的な解法と計算機援用のための枠組みを示し、制約処理の中で設計問題と計画問題を統合するアルゴリズムを解説する。そのために、第 4 章「問題解決の枠組み」では、本研究で提案する問題解決の方法の基本的な枠組みを説明する。また、計算機上の中心となる制約処理のアルゴリズムとして、第 5 章「制約緩和手法」で制約充足計算における制約緩和の手法を紹介した後、第 6 章「設計と計画の統合問題」では、空間的構造を決定する制約関係の処理と、時間的構造を決定する因果関係の処理を統合する手法について説明する。

第 3 のパートでは、本研究の有効性の実証を行なう。ここでは、製品開発に必要な知識を整理し、提案する方法を製品開発におけるトップダウンな問題解決に適用することで、本研究の有効性を実証する。そのために、第 7 章「製品開発過程への適用」では、製品開発過程におけるさまざまな知識を本研究の知識表現形式を用いて整理し、第 8 章「実験と考察」で、実際の製品開発の事例に対して本研究の問題解決の枠組み

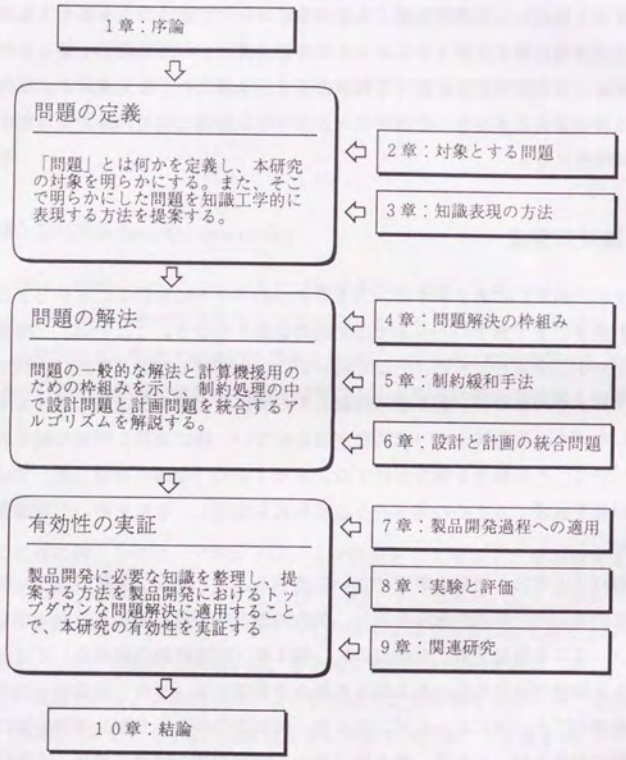



図 1.1: 論文の構成



を適用した結果を報告する。ここでは典型的な加工組立型の生産を必要とする機械製品を取り上げ、開発したツールを用いてトップダウンな製品開発の支援を行ない、提案する方法の有効性を実証する。また、第9章「関連研究」では、関連する研究について実用的な視点からサーベイを行なった結果を報告し、本研究の新規性を主張する。

以上の3つのパートの内容を踏まえて、最後に第10章「結論」で、本論文を総括し結論を述べる。結論では、工学的な視点と実用的な視点から本研究の成果をまとめる。また、研究の今後の課題と、将来のモノ作りのあり方について若干の私見を述べる。

# 第 2 章

# 対象とする問題

## 2.1 はじめに

問題とは、人間のもつ要求と現実世界の事実との間のギャップを指す。本研究では、このような定義のもとで、人間の問題解決を議論する。普段、我々は比較的無意識に“問題”ということばを用いているが、多くの場合、それらの指す対象は、この定義の中で扱うことができるだろう。本研究の対象には、モノ作りあるいは製造業における製品開発のみならず、人間が普段の生活の中で出会うさまざまな問題も含まれる。

本章では、本研究が対象とする問題の特徴について説明する。まず、次節では、対象問題のクラスを、従来の分類方法の中で位置付ける。続いて 3 節で、設計と計画という 2 つの行為の側面から、簡単な例題を用いて対象問題を明らかにする。さらに 4 節では、本研究が対象とする問題を実際の製造業の問題解決の中で位置づける。ここでは、特に製造業におけるモノ作りの中心的な存在である製品開発に対して、提案する方法の現実的な応用をあらかじめ明確にしておく。最後に 5 節で、本研究で提案する問題解決における計算機の役割を示す。

## 2.2 対象問題のクラス

人間が行う問題解決の中で特に難しい問題のクラスとして、悪構造問題（ill-structured problem）と悪定義問題（ill-defined problem）が挙げられる。ここで、悪構造問題とは、問題解決の方法（アルゴリズム）を事前に明確化することが困難な問題をいう。これに対して、悪定義問題とは、問題解決の対象（ユーザの要求）を事前に明確化することが困難な問題をいう [小林 92]。

悪構造問題に対する手法として、例えば組み合わせ最適化問題に属する問題の場合には、局所探索法（local search）、アニーリング法（simulated annealing）、タブー

探索法（tabu search）、遺伝アルゴリズム（genetic algorithm）、そしてニューラルネットワーク（neural network）などのメタヒューリスティックを用いた探索法 [茨木 94] などがある。また、人工知能における事例ベース推論 [Kolodner93]、説明に基づく学習 [Mitchell85] そして仮説推論 [deKleer86] などの研究の枠組みによって、その他の多くの悪構造問題を扱うことが試みられている。

一方、悪定義問題に対するアプローチとしては、ソフトウェア工学におけるプロトタイピング手法 [Tanik89], [Alan93] や要求獲得支援技術 [上原 91]、そしてエキスパートシステム構築方法論におけるインタビューや AHP などを利用したさまざまな手法 [寺野 91]、あるいは組織的な問題発見と学習の方法論としてソフトシステムズ方法論 [Checkland90] などを用いたアプローチなどが試みられている。

悪構造問題あるいは悪定義問題に対するこれらの問題解決のアプローチは、それぞれ部分的には成果を挙げているが、それぞれ問題領域に特化したものが多く、課題は多い。特に悪定義問題に対するアプローチについては、ハウツー的になりやすく、研究領域としてはこれからの部分が多い。

本研究が扱う問題は、以上の分類からすると、悪構造問題でありかつ悪定義問題である。つまり、対象となる問題そのものが複雑であるため明確に把握することができない上に、仮に把握されたとしてもその解法はあらかじめ定まったものがなく、問題解決のたびに個々に考えなければならない。従って、本研究で提案する問題解決の枠組みには、必然的に、問題そのものの明確化の過程が含まれる。つまり、提案する問題解決で重要なのは、問題は何かを明確にすることと、その問題を効率良く解くことの2点となる。

## 2.3 コンカレントな問題解決

問題解決とは、人間の要求と現実世界のさまざまな事実とを対応づける構造とパラメータを決定することである。この問題解決によって設定される構造の大半は、空間的な構造と時間的な構造の2つの構造によって成り立っているということができる。例えば、物の形状や配置などは空間的な構造の表現であり、行為の手順や方法などは時間的な構造の表現である。両者のどちらともいえないもの、例えば色や手触りなども、つきつめれば空間的あるいは時間的な構造によって表現可能であろう。

ただしここで注意すべきことは、空間的な構造あるいは時間的な構造は、対象を表現するという行為の結果得られたいわば便宜的なものであり、表現する対象と表現されたもの（空間的／時間的構造）とはもちろん異なる。本研究では、空間的な構造と時間的な構造を両極とする関係構造によって対象を表現しているが、表現対象そのも



のは両方の側面を同時に持っている。つまり、ある対象を時間の流れの中でとらえればそれは観察者にとって時間的構造として写り、スナップショット的にとらえればそれは空間的な構造として写る。

コンカレントな問題解決では、本来このように空間的な側面と時間的な側面を併せ持った対象（人間の要求や事実）について、部分的にはいったんどちらかの側面を強調した表現として形式化する。問題構造は、これらの異なる2つの要素を組み合わせた構造として表現されることになる。コンカレントな問題解決では、この2つの側面を併せ持った問題構造について、両者の対応関係を常に意識しながら問題解決を進める。このように、問題の明確化とそれ以降の問題解決において、空間的な構造と時間的な構造を同時並行的に扱うことが、コンカレントな問題解決の特徴である。

### 花子さんの問題

ここで、コンカレントな問題解決の例として、よくある生活の一場面を取り上げる。図 2.1 は、「花子さんが棚の上にある花瓶をとる」という場面を表わしている。まず図 2.1 の左側は、当初の問題構造を表わしており、そのままでは花子さんの身長が足りず「花瓶をとる」という要求が満たされない。要求を満たすためには、図 2.1 の右側のような解候補を考える必要がある。ここで、この解候補を設計問題としてとらえると、花子、椅子、床、棚、そして花瓶といった実体とその空間的な位置関係が強調され、計画問題としてとらえると、探す、置く、乗る、伸ばす、そしてつかむといった行為とその時間的な順序関係が強調される。

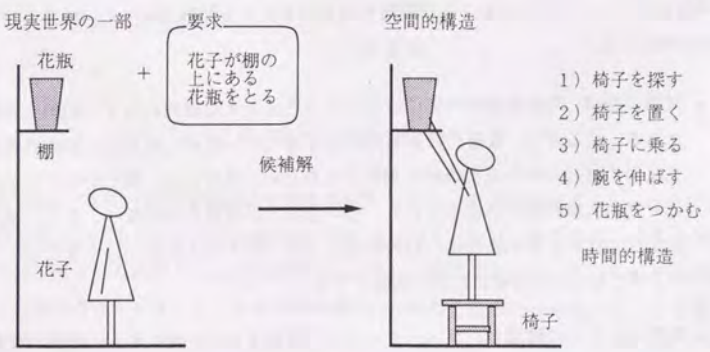



図 2.1: コンカレントな問題解決の例

花子さんの問題解決において、空間的な構造と時間的な構造は、はたして独立して決定可能であろうか。例えば、椅子という実体と花子さんとの関係は、椅子に乗るという行為の存在に裏づけられており、また床と椅子の関係は、その場所に置くという行為によって裏づけられている。行為の裏づけのない実体間の関係は、現実に存在し得ないし人間にとって意味を持たない。一方、行為もまた同様に、実体の裏づけを必要とする。乗るや置くといった行為は、その前提に乗る対象と置く対象、あるいは乗る主体と置く主体の存在がある。このように考えると、空間的な構造と時間的な構造は、非常にプリミティブなレベルで表裏一体の関係にあるということができ、問題解決では、この対応関係をいかに管理するかが重要な要素となる。

もし、花子さんが椅子に乗るという行為が（椅子の強度などの理由で）存在し得ない場合や、椅子を置くという行為が（床の凹凸などの理由で）存在し得ない場合には、例えば「次郎くんにとってもらう」といった解候補や、「棒を使う」「ジャンプする」などの他の解候補が導かれるだろう。おそらく人間が行う高度な問題解決ほど、これらの解候補にある空間的な構造と時間的な構造を、どちらが先というのでもなくほとんど同時並行的に導いていると思われる。なぜなら、両者の対応関係をよりプリミティブなレベルでとるほど、問題を解決する上での探索を効率よく行うことが可能であり、その分だけ、より広い範囲の探索空間の中から解を導くことが可能となるからである。

コンカレントな問題解決では、この例のように、人間の持つ要求と現実世界に存在する事実との対応関係を、特に、空間的な構造と時間的な構造の両方を同時に着目して定める。設計は空間的な構造を主に議論し、計画は時間的な構造を主に議論するため、ここで取り上げる問題解決は設計と計画の統合的な問題解決であるということもできる。

本研究では、このコンカレントな問題解決が対象とする問題に対して、以下の性質を持たせている。

- 問題の解は、問題構造の内部に設定されたさまざまな制約に対する制約充足解となる。ここでは、最適化は直接的には意識していない。ただし、複数の制約充足解の候補の中からどの解を選択するかという部分には、何らかの目的や評価尺度が人間の側に存在するため、広い意味での最適化は行なう。また、制約充足解が存在しない場合は、制約緩和によるペナルティを最小にするため、このような場合は部分的に最適化問題となる。

- 問題そのものは構造とパラメータによって表現される。従って、この形式で表現できない問題は扱うことができない。ここで構造は、要素間の関係によって定義され、取り扱っている関係としては、抽象／具体の関係、全体／部分の関係、原因／結果の関係、そしてさまざまな制約関係（線形多項式によって記述できる関係）がある。



## 2.4 実問題との対応

前節では、生活の中のよくある場面を取り上げてコンカレントな問題解決を説明した。このようなコンカレントな問題解決は、実際の製造業におけるさまざまな場面にも当てはめることができる。図2.2は、製造業におけるさまざまな問題解決を、時間的側面と空間的側面の対比の軸と、構造的側面と数量的側面の対比の軸の２つの軸を用いてプロットしたものである。

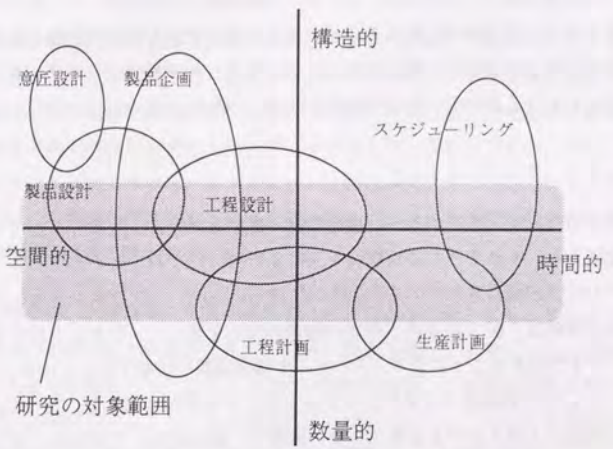



図 2.2: 現実問題との対応

図2.2のように、例えば意匠設計や製品設計などは、空間的な構造を主に決定する問題であり、プランニングや生産計画、そしてスケジューリングなどは、時間的な構造を主に決定する問題ということができる。また、両者の間には、製品企画や工程設計、工程計画などが存在し、それぞれは時間的な構造と空間的な構造のどちらを重視するかの比重がそれぞれ異なる。

また図 2.2 は、問題の内容が、構造、つまり要素と要素の関係を決定することが中心であるのか、それとも、数量、つまり要素そのものの量的な内容を決定することが

中心であるのかについても示しており、製造業におけるそれぞれの問題解決は、この点でも非常にバラエティに富んでいる。

製造業におけるこれらの問題に対する従来の問題解決は、それぞれの問題で必要な知識が異なる上に、このように問題のもつ性格も異なるため、問題解決は、個々の問題に対して独立して行なわざるを得なかった。例えば、製品設計と工程設計あるいは工程計画は、明確に分離され、製品設計が終了した以降に工程設計あるいは工程計画が行なわれる。

本研究の対象範囲は、図 2.2 の斜線の部分であり、コンカレントな問題解決を製造業に適用しようという本研究の試みは、極端にいえば、図 2.2 の斜線にあるさまざまな種類の問題を統一の枠組のもとで扱おうというものである。ただし、本研究ではさしあたり、図 2.2 における製品設計、工程設計、工程計画[1]、そしてスケジューリングを統合した問題を扱うことにする。ここで、工程設計、工程計画、そしてスケジューリングを合わせて製造計画と呼ぶことにすると、製品設計と製造計画は、市場の要求と工場が持つ製造資源などの現実とを対応づけ製品という形でアウトプットするという、製造業における最も中心的な問題解決の流れを構成している。

### コンカレント・エンジニアリング

製品開発の流れを、現在のような製品設計が終了した後に工程設計を行なうという前後の関係から、それぞれの問題解決を同時並行的に行なおうというコンカレントエンジニアリングが注目されている [木村 93], [Parsaei93]。コンカレントエンジニアリングの主な目的は、図 2.3 の各工程間の情報の流れを密にし、できるだけ工程のオーバーラップを持たせることにより、トータルで開発期間を短縮することにある。

これに対して、本研究で提案するコンカレントな問題解決は、図 2.3 に示すように、製品開発の上流工程である製品企画の段階で、製品設計、工程設計、そして工程計画について、事前にある程度ラフな問題解決を行なう方法として位置付けられる。ここでラフとは、問題解決に利用した知識の抽象度が現実の事実情報のレベルよりも高い、あるいは現実の事実情報が確率的なものであるということである。ただし、コンカレントな問題解決では、製品設計、工程設計、そして工程計画などの各問題解決はある程度ラフである反面、それらの各問題の間の相互関係を綿密に考慮することが可能である。従って、このようなコンカレントな問題解決を製品開発に適用することによって、以下のようなメリットが期待できる。

[1]ここで工程設計は、製品を製造するための工法を決定しその工法を実現する装置を設計する問題、工程計画は、その装置と実際の製造現場の資源との対応をとり、実際の注文に合わせて時間軸上に配置する問題として区別している



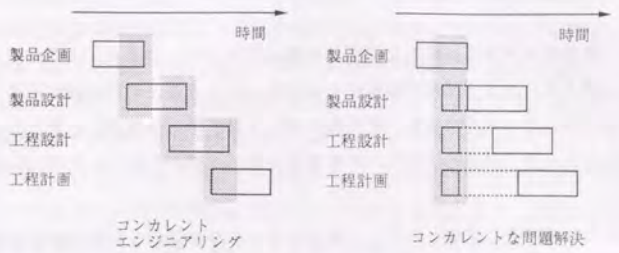



図 2.3: 製造業への適用

まず、製品開発における開発期間が従来のアプローチ以上に短縮されることが予想される。なぜなら、各問題解決の担当者は、あらかじめラフな問題構造およびその解である製品形状の概略や製造プランを知ることができるため、開発期間の延長の最も大きな要因である手戻りのケースが極端に減少するからである。また、特に工程設計や工程計画などの後工程では、コンカレントな問題解決とその後に行なう詳細な問題解決との間に時間的な余裕があるため、その間にさまざまな準備作業が可能となることも、開発期間短縮に寄与するだろう。

また、第 2 に、生産現場の現実の情報を、設計に直接反映させることによって、製品の品質そのものの向上が期待できる。事実、製造工程がまず変わることが製品開発に非常に大きな影響を与えるという“設計のための製造”の重要性が指摘されており [Clark91]、本研究のコンカレントな問題解決の枠組みによってこのような製品開発の方法を支援することで、従来のアプローチでは難しかった詳細な要求や事実への配慮が可能となると予想される。また、このことによって、製造現場が活性化するとともに、より創造的な製品開発が可能となるだろう。

メリットの第 3 として、製造環境のダイナミックな変化や、多様性への対応が可能となる点が挙げられる。近年の製造業を取り巻く環境は、変化が非常に激しく、受注後の仕様変更や供給部品の調達に関するさまざまな変更、あるいは製造現場における装置故障や新たな工法や製造技術の開発など、非常に多くの外的要因によって製品開発あるいは製造は影響を受ける。提案する問題解決の枠組みでは、一度得られた問題構造は再利用可能であり、その問題構造を用いることで、環境の変化や新たな要求の追加といった状況対応型の問題解決においても効果が得られるだろう。

## 2.5 計算機支援の方針

従来の計算機に対する一般的な期待は知能化であった。つまり、人間に代わって、あるいは越えて、さまざまな問題解決を自動的に行なうことを計算機に期待した。しかし、結局のところ知的に振舞う計算機の知的さは、その計算機に知識を設定した人間の知的さに大きく依存しており、計算機が完全に人間から独立した形で知的にはなり得ない。

本研究では基本的な立場として、計算機そのものが自律的な知能を獲得することをめざすのではなく、あくまでも人間の問題解決の支援ツールとして計算機を位置づける。その上で、人間と計算機からなる“システム”がより知的になり、計算機を利用した問題解決が利用しない場合の問題解決に比べてより高いレベルの能力をもつことに興味を向ける。特に、本研究が対象とする問題は悪定義問題であるため、問題そのものをあらかじめ定義することができない。従って、このような問題解決では、計算機の役割をあらかじめ明確にした上で、人間と計算機とのインタラクションの形態をどのように定めるかが重要となる。

このような人間と計算機からなるシステムにおいて、計算機の役割は何かという点を確認しておきたい。計算機とは、明確に表現された知識を蓄積・伝達・処理するための装置である。従って、問題解決において計算機は、過去の問題解決における知識を現在または将来に再利用するための蓄積を行ない、また複数の問題解決者の間での伝達を行なう。つまり、知識が表現された以降、それらの知識を必要なときに必要な場所へ伝達する役割を担う。これが計算機の第1の役割である。なお、知識の伝達、蓄積による再利用とは、単にデータやルールによる断片的な知識のみを指しているのではなく、例えばある人がプログラムを作成し、別の人がそれを利用するという、広い意味の知識（手続き）の再利用も含む。また、プログラム以外でも、手順書やメモといったものも、伝達、蓄積による再利用の対象となる。

計算機の第2の役割として、情報検索および解の探索があげられる。問題解決においては、ある視点のもとでいかに多くの情報を対象とするかという情報の幅が、問題解決の解の品質に大きく影響する。また同様に、ある制約や目的のもとでいかにして解を探索するかという情報の深さもまた、解の品質に大きく影響する。この情報の幅と深さは、計算機を用いることによって飛躍的に拡大させることが可能となるため、この能力によって人間だけでは到底なし得ない高い品質の問題解決を行なうことが可能となる。特にこれは、あらかじめ解法の定まっていない悪構造問題に対する非常に有効な手段となる。

さらに、計算機の第3の役割として、合意形成あるいは知識の体系化の支援が挙げられる。計算機を用いた共同作業では、構成員それぞれのもつ知識を顕在化させ、それらの知識の間の整合性をとり、そしてそれらを体系的に整理された状態にすることが必要となる。この過程では、場合によっては個々の構成員の要求や事実の一部分を変える必要もでる。このような過程は、計算機を用いることにより、より質の高いものとなる。また、具体的な目的を共有しないコミュニティーにおいても、それぞれの構成員が個々に行なう行為において、コミュニティー内部での基本的な背景知識の共有が前提とされている場合が多い。従って、基本的な合意形成と、知識の共有および体系化を計算機によって支援するニーズは極めて高い。



本研究ではこのような計算機の枠割をふまえて、コンカレントな問題解決における計算機の位置付けを以下のように設定した。なお、最初の2つの項目については、本論文の主要な内容となっており、第3の項目は今後の課題としている。

- 計算機は人間の問題解決に関する知識を蓄積、伝達するための媒体として利用し、過去あるいは他人の行なった問題解決をいかに効率良く再利用するかという点に重点をおく。
- 計算機を悪構造問題における解を得るための有効な手段として位置付け、検索機能と探索機能により、情報の幅、つまり対象とする情報の量と、情報の深さ、つまり組合せ探索における探索の密度を拡大させる。
- 計算機を合意形成および知識の体系化のために利用する。知識の蓄積、伝達は、知識の設定者と利用者の2者間のやりとりによって成立するが、計算機の内部でそれらの知識をより情報効率のよい形に再構成する。

人間の知的活動の中で計算機を活用するこれらの方法は、知識の再利用の効率の面以外に、新たな効果が期待できる。例えば、発想支援の研究 [Hori94], [Sugimoto94] では、コンピュータというある種の限定されたメディアを利用することにより、人間のもつ潜在的な概念を分節 (articulate) し、有効な知識として新たな発想につなげることを目的としている。発想された情報は、発想する前には少なくとも形式的には存在していなかった。なお、本研究ではとりあえず発想を阻害する要因を低減することを心がけ、発想に対する効果的な刺激という側面は直接考慮はしていない。

## 2.6 2章のまとめ

本章では、本研究が対象とする問題について説明した。本研究で扱う問題のクラスは、悪定義問題でありかつ悪構造問題に対応する。本章ではまず、問題解決とは何か、

を定義した後、本研究で提案するコンカレントな問題解決の特徴について説明した。コンカレントな問題解決の特徴は、空間的な構造と時間的な構造の同時決定性にあり、これによって設計と計画の統合的な問題解決を可能にしている。本章ではさらに、本研究が想定している現実問題を製造業におけるさまざまな問題の中で具体的に示し、本研究の狙いを明らかにした。そして最後に、計算機による問題解決の支援について本研究の基本的な方針を示した。

# 第 3 章

# 知識表現の方法

## 3.1 はじめに

知能を記号化し、蓄積・伝達・処理を可能にしたことが、現在の知識社会を人間が築いた最も大きな要因であるといえる。知識の表現形式は、そのための共通ルールに相当する。自然言語をはじめ、さまざまな形式言語、記号、信号などによって、現在の知識社会の基盤が構成されている。知識の表現形式とそれを利用した知的活動とは密接に関係しており、特に形式言語の場合は、計算機を用いた情報処理において、その表現形式が能力を決定的に左右する。

本研究では、人間の問題解決そのものを一般的に扱っており、特に、コンカレントな問題解決では、対象問題を空間的な側面と時間的な側面の両方から統合的に扱う必要がある。このような処理に向いた知識表現形式は、既存の研究の中には見当たらない。そこで、本研究では、コンカレントな問題解決のための独自の知識表現方法を設定することとした。

本章では、本研究で設定した知識の表現形式について説明する。まず次節では、知識表現における一般的な問題点を明確にし、その上で3節で本研究で採用した知識表現の特徴を述べる。続く4節では、具体的に知識表現の方法をデータ構造を含めて説明し、5節では簡単な例題によって採用した知識表現の方法を確認する。最後に6節では、知識表現に関する従来のさまざまな研究と本研究の知識表現との違いについて言及する。

## 3.2 問題点と方針

知識表現における問題点として、第1に、知識を表現する側とその知識を利用する側の間での状況と視点の相違による知識の利用効率の低下を、いかにして回避するか

という問題が挙げられる。また第2に、知識の適用が対象世界の状態を変更する場合、そのことによる対象世界の変化をいかに効率よく表現するか、あるい知るかという問題（フレーム問題）が挙げられる。

### 状況および視点の相違

表現された知識は、その知識を表現した人間と利用する人間の2者間を仲介する役割を果たす。知識表現における第1の問題点として、知識を表現する側とその知識を利用する側での“状況”と“視点”の違いによる知識の利用効率の低下、あるいは知識表現行為の限界が挙げられる。

知識というのは人間の持つ知能の断片であるため、人間をとりまく状況から独立であり得ない。これを知識の状況依存性 [中島 90] という。つまり、状況をある程度規定しなければ知識表現はできない。ここで状況とは、主体と環境との関係を指す。また、さまざまな情報を知識として表現するためには、何らかの視点が存在しなければならない。ここで視点とは、状況の解釈の方法といい直してもよい。

ここで問題となるのは、知識の表現者は、あらかじめ知識の利用者の状況と視点を知らないということである。これは、知識の表現者と利用者が空間的または時間的に離れていればいるほど顕著となる。このために、知識の表現者は、あらかじめ利用者の状況と視点を想定して（あるいは自分の状況と視点をそのまま）知識表現を行なう。知識を表現する側の状況と視点が、知識を利用する側とある程度一致していなければ、その知識は利用されないか、利用されても期待した効果は得られない。

このような、知識の表現者と利用者との状況および視点の相違からくる問題点に対して、本研究ではまず、できるだけ知識の表現者と利用者とが同一人物となるように配慮する。古典的な人工知能の研究では、知識ベースに知識を登録者する人とその知識を利用する人とが完全にわけられていたが、本研究では、知識の獲得と再利用をできるだけ区別することがないように工夫する。

そうした上で、さらに残る状況と視点の相違に対する対応策として、本研究では、以下の3つの方針をとっている。まず第1に、両者間の状況および視点の相違がそもそも起こらないように、表現する知識に状況や視点をできる限り入れないような手段をとる。特に知識を構成する要素の中で非常にプリミティブなものについては、状況や視点から中立とする。これは、人工知能の研究でいうオントロジー [元田 93], [溝口 94] と同様のアプローチである。視点の含まれる知識は、視点の含まれない知識をビルディングブロックとして別途構成することにする。

第2の方針として、知識の表現者と利用者の間には状況や視点の差があるという前提で、その相違をいかにして埋めかを、知識を表現する側から工夫する。一般に状況や視点の違いを埋めるためには、知識表現の抽象度を上げることで対応できる。知識表現の抽象度とは、表現する対象をどこまで一般化してとらえるかというレベルのことである。一般に、知識表現の抽象度が高ければ、その知識の利用可能範囲は拡大するが、その分だけ具体性がなくなり、知識利用による効用は減少する。従って、知識を表現する側の立場としては、知識表現の抽象度をどのように設定するかを、さまざまな点を考慮しながら検討する必要がある。



第3の方針として、知識を利用する側が、利用する際に計算機の能力を生かしてダイナミックに知識の構造を再構成し、利用する側の状況や視点に意図的に近付けるという方法を適宜採用する。本研究では、行為とその対象を分離可能な形態となっており、利用する状況に合わせて構造の動的な変更が可能なしくみとなっている。これにより、知識を表現した状況と利用する状況が異なっていても、その共通部分のみに着目し、残りの部分は知識の再構成を計算機によりある程度自動的に行なう方針とする。

### 環境に与える影響の表現 (フレーム問題)

知識表現を考える上でもう1つ重要な問題として、フレーム問題が挙げられる。フレーム問題とは、行為によって変化する世界を記述する際の難しさを指摘したものであり、行為の結果として何が変化し何が変かしなかったかを、いかにして記述するかという問題である [McCarthy69], [Sandewall72]。またフレーム問題を拡張した一般化フレーム問題 [松原 87], [松原 89] では、人間はそもそも、記述の面からいっても処理の面からいっても、必要な情報の一部分しか問題解決に利用することができない、という点を指摘している。

前者のフレーム問題に対して、STRIPS [Fikes71] をはじめとする多くのプランニングの研究では、削除リストと付加リストをあらかじめ明示することで便宜的にこの問題を回避しているが、この方法では、行為の副作用が対象あるいは状況よって異なるような場合にはうまく機能しない。この他にもフレーム公理、閉世界仮説、非単調論理、そして極小限定などのさまざまな形式的な枠組が提案され [McDarmott80], [McCarthy80], [Reiter80]、知識の記述量という面では一見フレーム問題が解決しているかに見えるが、一般化フレーム問題の立場を借りて言えば、これは単に記述量に関する付加を計算量に転嫁したにすぎず、根本的な解決には至っていない [松原 89]。

フレーム問題に対する本研究の立場は以下に示すとおりである。まず、指摘すべきことは、フレーム問題は、表現された知識と環境との間のインタラクションはないという前提にたった問題であるという点である。つまり、フレーム問題は、いったん表

現された知識の内部でいかに環境の変化を予測しえるかという問題なのである。環境とのインタラクションが可能であり、必要な情報の追加が可能であれば、その都度環境変化に対する情報を入力することにより、推論だけでは知り得ない環境変化に関する情報を得ることができる。本研究はこのような人間と計算機のインタラクションにより、事実情報の入力を必要に応じて行なうことによって、フレーム問題に対応している。

ただし、このことは、一般化フレーム問題の解決を意味することではもちろんない。例えば、新たな情報を環境から得るための負荷の問題や、環境のどの情報をどのような視点から要求するかを決定するための推論の負荷については、なんら解決されていない。しかし、一般に人間が行なっている疑似的なフレーム問題への対応と、本研究のアプローチでとっている方法は非常に近く、本研究のアプローチもまた、疑似的なフレーム問題の解決を図っているといってよいだろう。

なお、本研究における知識の非単調性は次のようにまとめることができる。まず知識の絶対的な量は常に単調に増加する。しかし、ここで矛盾に関する整合性の維持を行なった場合に、一時的に対象となり得る有効な知識の範囲が限定されることになる。つまり、状況に応じて存在する知識の中から有効な知識を計算機が取捨選択することにより、見かけ上知識が増減する。このことは、計算機が管理する情報の全体としては矛盾を受け入れるが、ある状況と視点を限定した際に、無矛盾であるような機構を計算機が持つことによって対応する。

## 3.3 知識表現の特徴

知識表現における以上のような問題点と方針をふまえて、本研究では独自の知識の表現方法を設定した。本研究における知識表現方法の主な特徴として、以下に説明するように、関係指向であること、状況や視点から独立した表現をベースとしていること、そして、静的な記述と動的な記述をともに扱えること、の3点が挙げられる。

### (1) 関係指向の知識表現

従来の知識表現の中で、例えばフレーム [Minsky81] や意味ネットワークなどは、実体とその関係によって対象を表現しているが、表現の中心は実体の側にある。つまり、実体があってはじめて実体間の関係が定義可能となる。これに対して、関係指向の知識表現では、結果的には同じように実体とそれらの関係からなる構造で対象を表現するが、表現のベースとしては、関係の側により重きを置いている。

本研究の知識表現では、制約関係や因果関係などの定義によってはじめて要素が定義される。関係概念は、それ自体で独立して定義可能であり、定義された関係を変更することは、実体そのものを変更することに相当する。つまり、実体は、どのような関係から成り立っているかによってその性質が定まる。例えば、4輪車という実体概念から車輪とのつながる関係概念を削除すると、削除した瞬間からその実体は3輪車となる。



### (2) 状況や視点から独立

知識表現のための最小単位であるビルティングブロックは、状況や視点から独立したものとなっている。状況や視点から独立であるためには、個別の知識を表現するよりも前に、それらのビルティングブロックを定義しておく必要がある。これらは、人工知能の研究におけるオントロジーに相当する。これらをあらかじめ定義するためには、対象世界に存在するさまざまな構造を、認識論的な考察をふまえた上でもっとも汎用的でかつ共通の粒度で分節する必要がある。

実際の知識は、これらのビルティングブロックの組合せによって表現する。組み合わせる際には、状況や視点が反映されるため、それらは、限定された経験的な知識となる。このように、本研究の知識表現では、状況や視点から独立であるビルティングブロックの組合せによってさまざまな知識を表現しており、それらの組合せによって表現できる世界が、問題解決の対象となる。

### (3) 静的な記述と動的な記述の混在

静的な記述とは、要素とその関係によって表現されたものであり、例えばフレームは静的な記述といえる。これに対して、ルールベースや手続き的な表現による記述は、動的な記述といえる。動的な記述というのは、知識の構造そのものが問題解決の過程で変化するような記述を指す。

本研究の知識表現では、静的な記述と動的な記述を共に扱うことができる。静的な記述としては、制約関係によって定義できる実体の性質や関係の表現が対応し、動的な記述としては、因果関係に含まれる前提と行動の記述や、状態遷移の前後のパラメータの値の関係などが対応する。これにより、提案する知識表現は、現実世界の空間的な制約関係を意識しながら、同時に時間的な関係を検討することが可能となる。

## 3.4 知識の表現形式

本研究における知識表現方法の骨格は、プリミティブとテンプレートという2つの表現形式によって構成されている。プリミティブは知識を表現するためのビルティン

グブロックであり、状況や視点から独立した知識の最小単位である。これに対して、テンプレートは、知識のある一部分を抽象化したものであり、プリミティブを要素として構成される。人間のもつ経験的な知識のほとんどは、このテンプレートによって表現される。ある表現対象がプリミティブとして表現されるかテンプレートとして表現されるかは、その対象を表現する上で問題解決の状況や視点が含まれているかどうかで判断する。

### 3.4.1 プリミティブ

表現された知識の最も基本的な単位となるプリミティブは、さらに、エンティティ、プロセス、リレーション、そしてオペレータの4つの単位に分類できる。以下に、それぞれのプリミティブについて説明する。

**エンティティ** エンティティは、物理的な実体を表わす基本単位である。人間が視覚的に把握する対象は、ほとんどこのエンティティに対応する。言い替えれば、エンティティとは空間的な広がりを持った対象である。一般に、エンティティは位置情報を持っているため、それらによって3次元空間上にプロット可能である。

**プロセス** プロセスは、行為や方法を表わす基本単位である。人間が把握するさまざまな状態遷移は、このプロセスとして表現される。行為や方法とは、事象[1]を構成要素とする特定のパターンであり、従って、プロセスは、時間的な広がりをもった対象である。プロセスは、それらの事象によって、時間軸上にプロットすることが可能である。

**リレーション** リレーションは、関係や特徴を表わす基本単位である。リレーションはパラメータと制約式によって、静的な関係や特徴を表現する。リレーションには有効な状態とそうでない状態の2通りの状態があり、この状態を切り替えることで問題構造が見かけ上変更される。リレーションは通常2つのプリミティブ間の関係を表すが、1つの場合や3つ以上の場合もある。

**オペレータ** オペレータは、事象を表す基本単位であり、動的な関係である因果関係を表す。オペレータは、一定の条件のもとで、リレーションの持つパラメータを変更することができ、これによって、設定された問題構造は、状態遷移が起こる。オペレータは、時間概念を表す基本単位でもあり、オペレータの適用によって、新たな時間的な制約関係が生成される。

[1]ここでは時間軸上の点に対応するものを事象、区間に対応するものを行為や方法として使い分けている。



エンティティとリレーションによって設計が主に対象とする空間的な構造を表現し、プロセスとリレーションによって計画が主に対象とする時間的な構造を表現する。また、エンティティとプロセスは、人間が要求や事実などの対象世界を表現する上で、直接的に認識している対象レベルの表現であり、これに対してリレーションとオペレータは、関係レベルの表現であるということもできる。リレーションは静的な関係を表し、オペレータは動的な関係を表す。図 3.1 に以上の4種類のプリミティブを整理する。

これらのプリミティブを用いた本研究の知識表現は、関係指向であるということができる。ここで、関係指向とは、関係という概念を実体や行為から独立して表現し、それらの表現から逆に実体や行為を特徴づけるような知識表現のアプローチをいう。つまりこれは、従来のオブジェクト指向やフレーム [Minsky81] が対象の属性として関係を定義しているのと、まったく逆のアプローチである。本研究の知識表現では、関係は実体や行為から独立しているので、関係構造を実体や行為の定義に規定されることなく、自由に変更することが可能である。

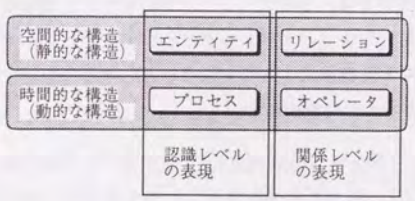



図 3.1: プリミティブの表現対象

#### クラスとインスタンス

プリミティブはそれぞれクラスとして定義される。クラスとは、同じ性質を持つプリミティブをまとめる概念である。例えば、花子さんは “human” というクラスに属する。これに対して、インスタンスとは、具体的な対象に対してクラスを対応づけた結果として生成されたものであり、花子さんは “human” のインスタンス “human_1” として計算機上で認識される。

本知識表現では、エンティティ、プロセス、リレーション、そしてオペレータのクラスは、名称によって識別する。従って、プリミティブ全体としてユニークな名称をつける必要がある。また、インスタンスも、問題構造の中に存在する全てのインスタンスにおいてユニークな名称を設定する必要がある。なお、インスタンスの名称は、計算機が自動設定するため、通常ユーザは意識する必要はない。

インスタンスは、ひとたび問題構造の中に生成されると、そのインスタンスが属していたクラスからは解放される。つまり、生成されたインスタンスは、属していたクラスにないさまざまな性質を、問題構造の中であらたに獲得することが許される。各プリミティブは、他のどのようなプリミティブとコネクションをもつかによってその性質が決定されるために、このようなプリミティブの性質の動的な変更が可能となる。

### 抽象化と継承

プリミティブのクラスは、その抽象レベルに応じたクラス階層を持つ。階層の上位はより抽象レベルが高い。最上位のクラスは“プリミティブ”というクラスであり、その下位に“エンティティ”“プロセス”“リレーション”“オペレータ”というクラスが定義され、そしてその下位にさまざまなクラスが定義される。

それぞれのプリミティブは、そのプリミティブの持つクラス階層によって、性質の継承を行う。ここで性質とは、どのプリミティブのクラスとコネクションを持つかということである。リレーションとオペレータについては、さまざまな制約関係や因果関係も継承の対象となる。継承によって上位クラスの持つ性質は下位クラスによってあえて定義されることなく受け継がれる。

### 制約関係

リレーションは1つまたは複数の制約式を持つことができる。本研究では、扱うことのできる制約式は線形等式または線形不等式に限定している。つまり、各リレーションは、関連するリレーションのパラメータについて、以下の制約を与える。

$$\sum_j a_{i,j} x_j = (\leq)\, b_i \quad i = 1, \cdots, n \tag{3.1}$$

ここで、$x_j$ は、自分自身を含む関連するリレーション $j$ のパラメータであり、係数 $a_{i,j}, b_i$ とともに制約の内容としてリレーションにあらかじめ定義されている。ただし、ここで定義される制約先のリレーション $j$ はクラスに相当し、実際の問題構造において、それが具体的にどのインスタンスが対応するかは、その都度個別に決定される。



制約の中には、緩和可能な制約と、緩和不可能な制約が存在する。問題解決の過程では、いくつかの緩和可能な制約が一時的に違反した状態となるが、緩和不可能な制約は、常に充足した状態となっている。この緩和不可能な制約は、問題解決の過程で論理的な矛盾を避けるために利用される。このような緩和不可能な制約を、本論文では“定義範囲制約”と呼ぶ。なお、本研究では、定義範囲制約は、制約式の項として、その制約が定義されたリレーションのパラメータのみを持つものに限定している。

さらに、制約の中には、そのリレーションの存在を規定する制約が存在する。この種の制約を“有効範囲制約”と呼ぶ。例えば、鋭角というのは角度が90度未満でなければ鋭角とは言えないため、この条件が有効範囲制約となる。有効範囲制約が違反した場合、リレーションそのものが無効となり、そのリレーションは問題構造の中に存在しないものと同等の状態となる。この場合、そのリレーションに含まれるすべての制約は効力を失う。この有効範囲制約は、定義範囲制約と同様に、制約式の項としてそのリレーションのパラメータのみを持つものに限定される。

### リレーションの状態

問題構造として計算機上に生成されたリレーションは、以下に示すようないくつかの状態のいずれかとなっている。問題解決の過程においては以下の状態の中で 2, 3, 4 のいずれかの状態にあり、最終的な解または解候補においては、2 または 4 の状態となる。

1. 定義範囲制約が違反している状態： 定義範囲制約が違反している状態というのは、論理的に矛盾した状態であり、計算機の内部処理においてリレーションがそのような状態に一時的になるが、ユーザの側から見た場合、このような状態は存在しない。

2. 有効範囲制約が違反している状態： 有効範囲制約が違反している場合、そのリレーションが存在していないものと同等とみなされるため、各リレーションの有効範囲制約の違反の状態によって、問題構造そのものが見かけ上、動的に変化する。

3. 制約式のいずれかが違反している状態： リレーションに含まれる制約式が違反している状態は、計算機上に設定された問題構造がまだ解または解候補に至っていない状態である。これは本研究における狭い意味での“問題”に相当する。

4. すべての制約式が充足している状態： 問題構造の中に含まれる（2に該当するリレーションを除いた）すべての制約式が充足されている状態は、解または解候補となる。この状態は、設定された人間の要求と現実世界の事実の間できちんと整合性がとれた形で対応している状態である。

### データ構造

各プリミティブのデータ構造を図 3.2 に示す。図 3.2 のように、エンティティとプロセスは、データ構造がまったく等しい。両者の違いは、エンティティがそのプロパティとして、主にリレーションとのコネクションを持つことで空間的な構造を表わし、プロセスがそのプロパティとして、主にオペレータとのコネクションを持つことで時間的な構造を表す点にある。

また、4つのプリミティブの中で、値を持つことができるのは、リレーションとオペレータのみであり、オペレータのパラメータは時刻に相当する。リレーションのパラメータはクラス定義の中であらかじめ省略値を設定可能であるのに対して、オペレータのを持つ時刻パラメータは、インスタンスのみが持つ。制約関係、先行関係、そして因果関係に含まれる定義式の各項には、リレーションおよびオペレータが持つべきコネクションと数値的な関係が設定される。以下に、各データ項目について説明する。

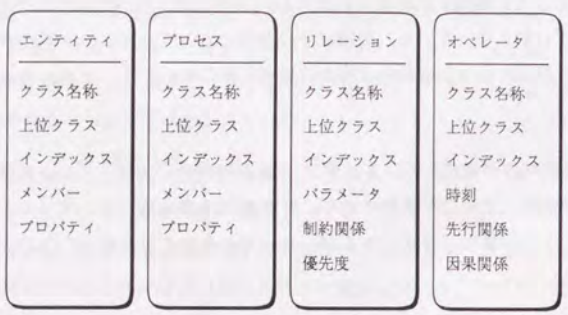



図 3.2: データ構造 1 (プリミティブ)

**クラス名称、上位クラス** クラス名称は、それぞれのプリミティブの識別名であり、すべてのプリミティブに対してユニークな名称を設定する。上位クラスは、クラス階層の上位クラスを1つ設定する。この上位クラスの設定によって、クラス階層が計算機の内部に生成される。



**インデックス** インデックスは、そのプリミティブがユーザ要求あるいは事実のどのようなキーワードに対応するかを表わすものである。プリミティブは計算機の内部表現であるため、この内部表現とユーザの概念世界や現実世界との対応づけをするのがインデックスである。

**メンバー** エンティティ、プロセス、リレーションはそれぞれメンバーを持つ。エンティティのメンバーにはエンティティが、プロセスのメンバーにはプロセスがそれぞれ設定される。ここでメンバーとは、プリミティブ間の has_a 関係を表わすコネクションであり、例えば花子さんに対応する “human” のメンバーは “head” “body” “arm” などとなる。

**プロパティ** エンティティとプロセスは、プロパティとしてリレーションまたはオペレータへのコネクションを持つ。このプロパティは、エンティティまたはプロセスの持つ性質を表わす。つまり、どのようなプロパティとして持つかが、そのプリミティブの性質を決定する。従って、プロパティは、エンティティまたはリレーションの定義そのものともいえる。

**パラメータ** リレーションは、パラメータを持つ。パラメータは、そのリレーションの性質を数値で表わしたものであり、1つのリレーションはパラメータを1つもつことができる。リレーションのクラスに対してあらかじめ設定するパラメータの値は、インスタンスに対するデフォルト値となり、インスタンスのパラメータの値は、実際の問題解決の進行にともない変更される。

**制約関係** 各リレーションには、制約関係として必要に応じて、他のリレーションのパラメータまたはオペレータの時刻との間で、複数の制約式を設定できる。定義範囲制約と有効範囲制約は、制約関係の特殊な形態であり、そのリレーション自身のパラメータの値を規定している。なお、これらの制約関係は、クラス定義の中で設定し、実際の問題構造にあたるインスタンスでは、制約相手のプリミティブのみが変更可能となる。

**優先度** 優先度は、そのリレーションが他のリレーションと比べて問題構造の中でどの程度重要であるかを示す。リレーションには、要求によって設定されたもの、事実によって設定されたもの、そしてそれらを関係づけるために新たに設定されたものなどが存在するが、それらをこの優先度によって識別する。一般に、事実に関するリレーションは、要求に関するリレーションよりも優先度が高く、その制約を緩和あるいは取り消すことが困難である。

**時刻** オペレータはパラメータとして時刻をもつ。時刻は、リレーションのパラメータとほぼ同様に扱われ、制約関係、先行関係の制約式、あるいは因果関係の演算式の項として指定できる。オペレータの時刻からなる構造は、行動プランやスケジュールなどの問題構造の時間的な側面を表現する。なお、時刻は、インスタンスのみがもつものであり、クラス定義の中でデフォルト値を定義することはできない。

**先行関係** 先行関係は、制約関係と似ているが、あらかじめクラスの中で定義することができずに、インスタンスの中で動的に制約式が生成されるという点が異なる。先行関係の制約式は、少なくとも2つのオペレータの時刻を項として持ち、一方のオペレータには自分自身が、他方のオペレータには、計算された状態遷移のグラフの中で関係づけられた直前または直後のオペレータが割り当てられる。

**因果関係** 因果関係には、条件部に、オペレータによる状態遷移を行なう場合の条件となるリレーションを、行動部に、オペレータがパラメータを変更するリレーションを設定する。条件部のリレーションがすべて有効である場合に、状態遷移が行なわれる。行動部のリレーションは1つ、条件部のリレーションは複数設定することができる。行動部のリレーションのパラメータの変更後の値は、そのパラメータの変更前の値と、条件部にあるリレーションのパラメータの値から、あらかじめ設定された演算式によって計算する。

### 3.4.2 テンプレート

テンプレートは、過去の問題解決の事例からいわば帰納的に抽出した知識の単位である。テンプレートはプリミティブと異なり、状況や視点を含んだ知識であるが、知識の粒度と分節は過去の複数の問題解決の事例から知識の再利用性を考慮して決定されており、ある程度の汎用性を失わないように考慮されたものである。テンプレートは、プリミティブのクラスをその構成要素としており、その構造に従って再び該当する対象のインスタンスを生成することによって、知識を再利用することができる。

テンプレートはプリミティブのクラスをノードとするグラフ構造となっている。計算機内部では、それぞれのプリミティブはインスタンスとして問題構造を形成しているのに対し、テンプレートは、この問題構造の一部を切り出し、それぞれのインスタンスをクラスとして抽象化したものである。ただしこの際、問題構造にある複数のプリミティブが、テンプレート上の1つのプリミティブに対応する場合やその逆の場合などがあり得る。



テンプレートの適用

テンプレートは、過去の問題解決で利用された知識を、新たな問題解決において利用するためのものであり、具体的には以下のような場合に利用される。

- 関連するプリミティブの取り込み：問題構造を計算機の内部に生成する過程において、ある程度完成された問題構造をもとに、欠落した部分の知識を過去の類似事例によって補うために利用する。
- 問題構造の精緻化：問題解決の過程で解候補が得られた以降、さらに問題を精緻化する場合に利用する。問題構造の一部分に、より詳細なテンプレートを適用することにより、さらに具体的な問題構造が新たに生成される。
- コネクションの整頓：人間が特定の対象について設定した問題構造は、構造そのものが間違っているかあるいは不完全な場合がある。過去の類似事例にあたるテンプレートと比較することにより、その部分を見つけ、場合によってはそのテンプレートに合わせて修正する。

テンプレートの適用にあたっては、該当事例とテンプレートとの類似度を計算し、類似度がある一定値以上となった場合に適用するという方針をとる。ただし、アルゴリズムの実装上、類似度計算の信頼性がやや劣るため、テンプレートの適用にあたっては、ユーザの確認を必要とする。類似度 $z$ の計算は以下の式に従う。

$$z = \sum_{i \in N^{src}} \sum_{u \in N^{dst}} a^{src}_{i,u} a^{dst}_{j,v} (s_{i,j} + s_{u,v}) \tag{3.2}$$

ここで、$N^{src}$ は、問題構造の中で対象とする部分のグラフ構造のノードの集合、$N^{dst}$ は、テンプレート上のノードの集合である。また、$a^{src}_{i,u}$ は、問題構造内の各ノードに対する隣接行列の要素を、$a^{dst}_{j,v}$ は、テンプレート上の隣接行列の要素をそれぞれ表す。さらに $s_{i,j}$、$s_{u,v}$ は、ノード $i$ とノード $j$、ノード $u$ とノード $v$ の類似度である。

グラフの隣接行列の要素 $a^{src}_{i,u}$ および $a^{dst}_{j,v}$ は、ノード $i$ からノード $j$, またはノード $u$ からノード $v$ へのアークが存在する場合に 1, そうでない場合に 0 の値をとる。なお、この隣接行列の対角要素には 1 が設定される。

類似度には、2つのノードが似ているほど、0 から 1 の間で高い値が設定される。類似度の値は、次の式によって計算される。ここで、$\alpha, \beta$ は、それぞれ類似度係数であり、0 から 1 の間で設定する。さらに、$d$ は、クラス階層ツリー上の距離を表す。

$$s_{i,j} = \begin{cases} \alpha^d, & \text{if } i \text{ is ancestor of } j \\ \beta^d, & \text{if } i \text{ is descendant of } j \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

この定式化では、グラフ間の類似性は、コネクションのレベルまで考慮しいる。この定式化により、2つのグラフ間の柔軟なマッチングが可能となる。つまり、テンプレートの適用は、現実にグラフの構造が完全に等しい必要はなく、構成する要素が個々に類似しており、それらの要素からなる構造もまた、ある程度類似していればよいように工夫されている。

### テンプレートの抽出

テンプレートは、過去のさまざまな問題解決事例において設定された知識をある範囲内でとりだしたものである。このテンプレートの設定は、問題解決が終了したのち、あるいは問題解決の過程の中で行なわれる。テンプレートはあくまでも知識の再利用を目的としているので、事例からの抽出にあたっては、その汎用性を十分考慮しなければならない。

テンプレートを事例から抽出するためには、粒度 (granularity) と分節 (articulation) が重要な意思決定要素となる。ここで、粒度とは、事例におけるプリミティブとテンプレート上のプリミティブの数の関係であり、部分的に事例の複数のプリミティブが1つのテンプレートのプリミティブに対応する。また、分節とは、その場合の事例におけるプリミティブの切れ目をどこにするかに相当する。

実際にテンプレートを得るためには、過去の複数の事例を用意し、その中から部分的に共通な構造を探し、その構造を抽出するアルゴリズムを用いる。ただし、本研究では現在のところ、テンプレートは個別に手作業で作成しており、テンプレート抽出の計算機による自動化には至っていない。この部分については今後の課題としている。

### データ構造

図 3.3 にテンプレートのデータ構造を示す。まず、クラス名称は、テンプレートを識別するためのものであり、すべてのテンプレートの中でユニークな名称を設定する。テンプレートもプリミティブ同様、クラス階層を持ち、上位クラスの指定を通して、テンプレートの階層構造を定義する。インデックスは、人間の持つ要求に含まれるキーワードとの対応をとるものであり、プリミティブと同様にあらかじめ設定する。



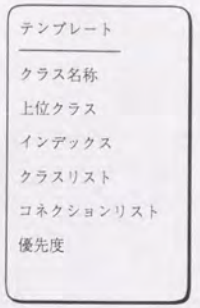



図 3.3: データ構造 2 (テンプレート)

クラスリストの各項には、テンプレートを構成するプリミティブのクラスが設定される。これらは、そのテンプレートの内容そのものとなる。コネクションリストの各項には、クラスリストで設定したプリミティブに対して、それらの構造を定義するために2つのクラスの間でのコネクションの形で設定する。コネクションは有向アークであり、プリミティブのデータ構造におけるメンバーやプロパティなどが対応する。

テンプレートの優先度は、テンプレートを適用する際に、複数の候補が存在する場合の選択の基準となる情報である。この値は、テンプレートを管理する側が設定し、利用する側は優先度の高いテンプレートを優先的に適用する。通常、よく使われるテンプレート、あるいは最近使われたテンプレートなどが優先度が高く設定される。

## 3.5 問題構造の例

ここで、提案する知識表現によって表現された問題構造の例を示しながら、知識表現の具体的な説明を行なう。例としては、2章で取り上げた花子さんの問題と、スケジューリング問題を取り上げる。花子さんの問題は、特に空間的な構造を意識した問題であり、スケジューリング問題は特に時間的な構造を意識した問題であるといえる。

### 花子さんの問題

花子さんの問題とは、「花子さんが棚の上にある花瓶をとる」という要求と、現実とのギャップをうめることであった。要求に対して、花子さんは身長が低いためにそのままでは花瓶に手がとどかず、結果的に椅子に乗るという行為によって要求が満たされる。図 3.4 にこの花子さんの問題の問題構造を示す。

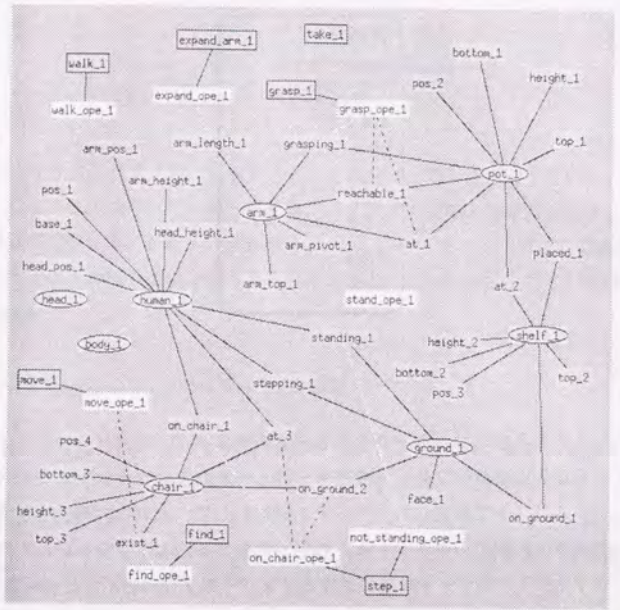



図 3.4: 知識表現例 1 (花子さんの問題)

この問題構造には、エンティティとして花子さん (human_1)、花子さんの腕 (arm_1)、花子さんの頭 (head_1)、花子さんの身体 (body_1)、棚 (shelf_1)、花瓶 (pot_1)、椅子 (chair_1)、そして床 (ground_1) が設定されている。arm_1, head_1, body_1 はそれぞれ human_1 のメンバーとなっている。また、プロセスとして、とる (take_1)、つかむ (grasp_1)、腕を伸ばす (expand_arm_1)、歩く (walk_1)、移動する (move_1)、見つける (find_1)、乗る (step_1) などが設定されている。grasp_1, expand_arm_1 は、take_1 のメンバーである。

ここで設定されたリレーションには、制約式を含むものとそうでないものに分けられる。この例では、制約式を含まないもとして、水平方向の位置を表す pos_1, pos_2, pos_3, pos_4 と、垂直方向の位置を表す base_1, head_pos_1, arm_pos_1, arm_top_1, face_1, bottom_1, top_1, bottom_2, top_2, bottom_3, top_3 がある。これらの位置を表すリレーションは、水平方向については at_1, at_2, at_3 の持つ制約式によって、垂直方向については、height_1, height_2, height_3, arm_height_1, head_height_1, arm_length_1,



および、on_ground_1, on_ground_2, on_chair_1, placed_1, standing_1, stepping_1, arm_pivot_1, reachable_1 がある。なお、arm_pivot_1 は、制約式を持つと同時に垂直方向の位置も表す。上記以外のリレーションとして、grasping_1 と exist_1 があるが、これらはパラメータの値が 0 または 1 であるリレーションであり、有効範囲制約によって、値が 1 の場合に有効となる。

この問題構造では、要求を満たすために grasping_1 というリレーションを有効にする必要があるが、オペレータ grasp_ope_1 を適用して grasping_1 を有効にするためには at_1 と reachable_1 という 2 つのリレーションが有効でなければならない。同様に、腕を伸ばすには expand_ope_1 の適用を、花子さんが移動するには walk_ope_1 の適用を、椅子に乗るためには not_standing_ope_1, on_chair_ope_1 の適用を、椅子を見つけるには find_ope_1 の適用を、そして椅子を移動するには move_ope_1 の適用を検討する必要がある。

### スケジューリング問題

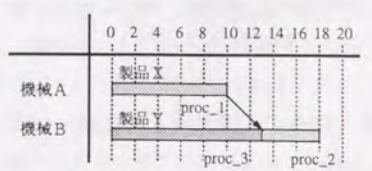
続いて、非常に単純なスケジューリング問題を例として取り上げる。一般にスケジューリング問題は、さまざまな作業に対して、時刻と資源を割り当てる問題ということができる。スケジューリング問題における制約の中で、特に時間的な制約としては、作業順序からくる先行関係や資源の競合などによる排他関係があり、これらの関係を柔軟に扱うことができるかどうかが重要となる。

図 3.5 にスケジューリング問題の例を示す。この問題は、製品 X と製品 Y の 2 つの製品を機械 A, B の 2 つによって製造する例であり、機械の利用順序と所要時間は図 3.5(a) に示すとおりである。スケジュールは図 3.5(b) のように、3 つのプロセスが機械に割り当てられる。

| | 製品X | 製品Y |
|---|---|---|
| 機械A | 1(10) | —— |
| 機械B | 2(5) | 1(13) |

no. (min)

(a) スケジュールデータ




(b) スケジュール結果

図 3.5: スケジューリング問題

このスケジューリング問題は、コンカレントな問題解決の中で特に時間的な構造を

中心的に扱っている例ということができ、図 3.6 のように表すことができる。ここで、エンティティとしては、製品 A に対応する product_1、製品 B に対応する product_2、製品 B を構成する要素である part_1, part_2 がある。また、task_1 は part_1 を対象とし、task_2 は part_2 を対象とし、この 2 つの作業によって product_2 を生産する。同様に、task_3 は product_1 を対象として生産を行なう。task_1 は start_2, end_2 というオペレータと duration_1 というリレーションによって構成され、同様に、task_2 は start_3, end_3, duration_2 から、task_3 は start_1, end_1, duration_3 から構成される。

ここでは、製品または部品の状態を、raw, midium, exist という 3 つのクラスのリレーションによって表現し、raw_1 が作業 task_1 が開始可能な状態、midium_1 が task_1 の処理中、exist_1 が task_1 の終了状態でかつ task_2 の開始可能な状態を表す。同様にして、midium_2 が task_2 の処理中、exist_2 が task_2 の終了、raw_2 が task_3 が開始可能な状態、midium_3 が task_3 の処理中、exist_3 が task_3 の終了状態を表す。raw, midium, exist のクラスに属するそれぞれのリレーションは、定義範囲制約として $0 \leq p \leq 1$、有効範囲制約として $p = 1$ が設定されており、初期状態では、ini の値は 1 で残りは 0 である。

機械 A と機械 B はそれぞれ machine_1, machine_2 によって表され、それぞれが使用可能な状態であるかを level_1, level_2 で表す。これらのリレーションも raw, midium, exist と同様、定義範囲制約として $0 \leq p \leq 1$、有効範囲制約として $p = 1$ が設定されており、初期値は 1 である。この level_1, level_2 の値を変更するオペレータとして、keep_1, free_1 および keep_2, free_2, keep_3, free_3 がある。keep_1, keep_2, keep_3 は値を $-1$ し、free_1, free_2, free_3 は値を $+1$ することにより、level_1, level_2 の値を無効化または有効化する。なお、keep_1, keep_2, keep_3 は start_2, start_1, start_3 と同一のリレーションを条件に持つことにより、同時に適用されるようになっている。同様に、free_1, free_2, free_3 は、end_1, end_2, end_3 と同一のリレーションを条件にもつ。

## 3.6 従来の知識表現との比較

対象世界を表現するための知識表現の方法には、非常に多くのものが提案されており、計算機上で実際に利用されているものだけをとっても、それらをすべて列挙することは不可能に近い。ここでは、本研究で開発した知識表現と比較的その目的や方法が似ているものについて概観し、その上で本研究のアプローチとの違いを明確にする。

まず、ソフトウェア工学の研究領域についてみると、対象とする問題を記述するた



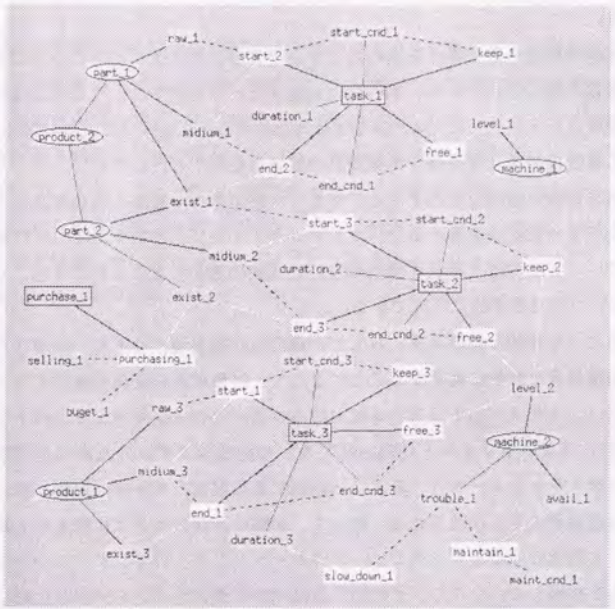



図 3.6: 知識表現例 2 (スケジューリング問題)

めの言語として近年特にオブジェクト指向言語が注目され、実際に数々の成果を挙げている。オブジェクト指向とはあらためて説明するまでもなく、対象世界をオブジェクトという単位で抽象化し記述する方法である。さまざまな手続きもオブジェクトの内部に隠蔽される。本研究の知識表現は、すでに述べたように関係指向であり、このオブジェクト指向の概念と根本的に異なる。関係指向であることによって、オブジェクトそのものの性質を、動的に変更できるばかりか、概念が明確でないオブジェクトを徐々に明確化するといったことも可能となる。

また、ソフトウェア工学における仕様記述のための方法である Entity-Relation モデルや、データフローダイヤグラム、あるいは人工知能の研究における意味ネットワークなど、対象世界を表現するためのさまざまな方法があるが、これらはすべて、対象を厳密に表現することのみに力点がおかれており、ここで表現された内容は、計算機を用いた問題解決で直接的に利用されていない。つまり、計算可能な表現への変換には再び人間の介在が必要である。事実、上記の知識表現が可能であるさまざまな上流

CASE ツールは、プログラムを生成する下流 CASE ツールから、未だに孤立したままである。

人工知能の研究分野で特に実績のあるフレームおよびスクリプトについても本研究の知識表現の優位性は大きい。本研究の知識表現との比較では、フレームは空間的な構造を表すエンティティに、スクリプトは時間的な構造を表すプロセスに相当するが、フレームまたはスクリプトは、対象世界の知識を整理するには便利であるが、そこで記述された知識は静的なものである。つまり、記述された知識と問題解決のメカニズムとはあくまで独立である。また、フレームやスクリプトという単位の中に知識をはめ込んでいく方法では、状況や視点の違いに柔軟に対応することができず、知識獲得のボトルネックは根本的に解消できない。

以上のような静的な知識表現に対して、動的な知識表現、つまり、表現された知識がその表現世界を動的に変更することによって、対象世界の変化を表現するようなものとしては、エキスパートシステムにおけるプロダクションルールや、ペトリネット [椎塚 92] 、ステートチャート [Harel87] などの動的システムのための記述形式がある。プロダクションルールは、非常に汎用的である反面、ルールの定義において知識表現者の意図が大きく反映される。従って、知識のメンテナンスに膨大な工数が必要となる。本研究の知識表現形式では、オペレータとしてこのプロダクションルールに近い形式を採用した。ただし、その使用方法はプロダクションルールと比較して非常に限定されたものとしている。

一方、ペトリネットやステートチャートなどは、対象世界が動的である場合を特に想定して設計された表現形式である。これらの知識表現形式は、対象の特に動的な側面をできるだけ忠実に計算機上で再現することにおいて優れているが、そのそもの目的はモデルの解析にあり、本研究のように問題解決を指向していない。つまり、これらは対象世界の必然的な状態遷移を表現しているのであって、問題解決のような意図的な状態遷移を行なうことには不向きである。

最後に人工知能の研究分野で最近話題となっているオントロジーについて触れる。オントロジーとは、そもそも哲学用語で存在論にあたるが、人工知能の研究分野では、「人工システムを構築する際のビルティングブロックとして用いられる基本概念／語彙の体系」という定義がなされる [溝口 94], [Tijerino93]。オントロジーに関する議論は途上であり、統一した認識は現時点ではまだ出来上がっていないともいえるが、本研究の知識の表現形式は、基本的にはこのオントロジーの目指す方向と一致している。ただし、オントロジーの研究では、タスクオントロジーとドメインオントロジーとにまず大別するが [元田 93]、本研究の知識の表現形式の中には、タスクオントロジーに相当する概念は陽には存在せず、すべてドメインオントロジーである。



## 3.7 3章のまとめ

本章では、対象とする問題を表現するための基本的な要素となる知識の表現形式を説明し、それを用いた知識表現の方法を示した。まず、知識表現の問題点と方針を示した後、知識表現の特徴として、関係指向の知識表現、状況や視点からの独立、そして静的な記述と動的な記述の混在を挙げた。知識表現の形式としては、状況や視点に依存しないプリミティブと、特定の経験的な知識を表すテンプレートに分かれる。それぞれの詳細なデータ構造を解説した後、2つの例題によって、実際に知識を表現する際の方法を説明した。また、従来の多くの知識表現の方法をサーベイし、本研究の知識表現との違いを明らかにした。

# 第 4 章

# 問題解決の枠組み

## 4.1 はじめに

問題解決とは、人間の概念世界にある要求と現実世界の事実との間の対応関係をとることである。ただし、人間のもつ要求と現実世界の事実はともに、問題解決の最初の時点では明確にわかっていない。つまり、問題そのものが隠されている場合が多い。従って、実際の問題解決では、問題構造を明確化することと、その問題構造の中で、要求と事実との対応関係をとることの2つが重要となる。

本章では、設計と計画を統合的に扱うためのコンカレントな問題解決の枠組を示す。人間の要求と現実世界の事実とを対応づける一般的な問題解決の目標に加えて、対象の空間的な構造と時間的な構造をも対応づけるコンカレントな問題解決の目標を達成するための具体的な方法を示す。また、ここで提案する枠組に沿った計算機援用システムである PICCSS (Problem Interactive Clarification and Concurrent Solving System) の概要も紹介する。PICCSS は、本論文で提案する知識表現と推論機構を備えており、人間が行なうコンカレントな問題解決を支援するためのツールである。

まず次節では、問題解決の枠組みの概略をまず説明し、その中で問題明確化について3節で説明する。また、設定された問題に対する制約処理について4節で簡単に説明する。そして、5節では問題構造の可視化と評価について、6節ではトップダウン精緻化について説明する。7節では、具体的な事例を用いて問題解決の流れを確認し、そして最後に8節で、開発した計算機援用システム PICCSS の概要を紹介する。

## 4.2 問題解決の枠組み

図 4.1 に問題解決の流れを示す。図 4.1 において、要求設定と事実設定の2つによって、計算機内部に問題構造が生成される。ここで生成する問題構造には、要求と

事実の構造がともにプリミティブからなるグラフ構造で表現され、この問題構造をもとに制約処理を行い解候補を求める。この一連の流れが何度か繰り返されるうちに、計算機内部の問題構造がトップダウンに精緻化されていく。問題解決の終了は、この精緻化のサイクルの中で、解候補を人間が評価した結果、満足いくと判断した場合とし、その時の解候補が解となる。

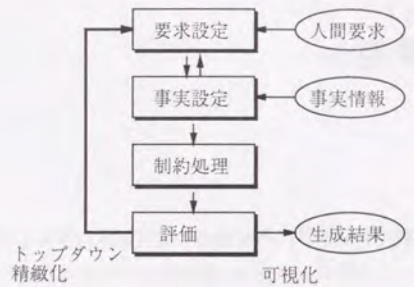



図 4.1: 問題解決の流れ

以下に、簡単に各ステップについて説明する。

**要求設定** 要求設定では、人間の概念世界にある要求を、計算機上に表現する。人間の要求はそもそも、ことばにもならないモヤモヤした状態にあるが、これをことばとして分節し、計算機上の表現形式であるプリミティブの構造に置き換える。要求をことばにするまでは人間が行なうが、それ以降はプリミティブの持つインデックス情報をもとに、人間と計算機が対話的に要求の構造を設定していく。

**事実設定** 事実設定では、現実世界の事実情報の中で、問題解決にとって必要な情報を計算機上に表現する。表現する事実情報は、要求と同様、プリミティブのをノードとするグラフとして要求の構造に追加される形で設定される。ここで設定する事実情報は、すでに設定された要求の構造の断片をもとに、事実情報の内部形式であるファクトの中から検索される。事実情報の設定の結果、計算機内部には、要求と事実が混在した形で問題構造が形成される。

**制約処理** 制約処理では、設定された問題構造について、そこに含まれる制約に対して処理を行なう。ここで行なう制約処理は、大きくは、制約充足計算と状態遷移計算の２つの処理から構成される。前者の制約処理では、すべての制約を充足するようなパラメータや時刻の値を決定し、後者の状態遷移処理では、その結果に従って実際にパラメータや時刻の値を状態遷移によって変更する。



**評価** 制約処理によってすべての制約が充足された状態にある問題構造は、解候補となる。評価では、人間がこの解候補について、計算機上には表現し得ないさまざまな評価基準と照らして、解とするかどうかを判断する。判断にあたって、問題構造は、人間に理解できる形式に可視化される。

以下の各節において、問題解決のこれらの各処理内容について説明を行なう。

## 4.3 問題明確化

本研究において、問題とは要求の構造と現実の構造とが対応していない状態を指す。ここで問題には、潜在的な問題と顕在化した問題がある。潜在的な問題とは、要求や事実が表現されていないため、問題はあるのだが、何が問題であるかが分からない状態であり、顕在化した問題とは、要求や事実を表現することを通して両者の違いを明確にし、問題そのものの構造が明らかになった状態である。

問題解決にあたってまず最初に行なうべきことは、潜在的な問題を顕在化することであり、そのための作業として問題構造の明確化を行なう。図 4.1の要求設定と事実設定はこのための作業であり、この２つのステップを合わせて問題明確化と呼ぶ。なお、要求設定と事実設定は、図では要求設定が先行しているが、実際にはこの２つのどちらが先かは明確ではない。

問題明確化の作業では、要求の構造と事実の構造を、計算機内部にそれぞれプリミティブからなるグラフ構造としてを設定する。要求構造と事実構造はそれぞれ独自に計算機内部に設定するが、それらのを構成するプリミティブはできる限り共通化される。要求と事実におけるそれぞれのパラメータの値の違いや満たされていない制約関係などが存在する場合、それが解決すべき“問題”となる。

### 4.3.1 要求設定

要求設定では、まず人間が計算機に対して、要求を自然言語で入力することから始める。計算機にはあらかじめプリミティブが登録されており、ユーザによって表現されたことばをもとに、関連するプリミティブを検索する。プリミティブの検索には、プリミティブの持つインデックスが利用される。ここで検索されたプリミティブのクラスに対応するインスタンスを計算機内部に設定する。

表 4.1: コネクションの種類

| connection type | source | destination |
|---|---|---|
| entity | entity | entity |
| feature | entity | relation |
| phenomenon | entity | operator |
| process | process | process |
| duration | process | relation |
| event | process | operator |
| function | relation | relation |
| time | relation | operator |
| action | operator | relation |
| condition | operator | relation |

プリミティブがメンバーやプロパティをもつ場合は、対応するプリミティブのクラスのインスタンスと、そのインスタンスへのコネクションを自動的に生成する。また、テンプレートを新規に設定した場合は、そこに定義されている構造がそのままインスタンスとして問題構造の中に設定される。

新たに設定されたプリミティブは、すでに計算機内に設定されているプリミティブと間でできるかぎりコネクションが持たれる。プリミティブ間のコネクションをいかに設定するかは、問題構造を決定するうえで特に重要であり、以下のいずれかの方法によって行なわれる。

- 人間が個々に直接設定する方法：最も原始的な方法は、PICCSS のユーザインタフェースを介して、ユーザが直接 2 つのプリミティブを指定し、そのコネクションの種類を指定することによって、コネクションを設定する方法である。コネクションの種類には、表 4.1 のようなものがある。

- テンプレートによって設定する方法：問題構造が過去の問題と類似しているような場合、テンプレートによる問題構造の補強が有効となる。テンプレートをすでに設定された問題構造の一部に適用することにより、未完成のコネクションが完成され、また場合によって新たなプリミティブが問題構造に追加される。

なお、テンプレートを利用した問題構造の設定では、人間が直接指示したことばには含まれなかった欠落情報や背景の状況に関連するプリミティブを、併せて設定してくれるという利点がある。しかし、反面、意図した要求と異なる問題構造を生成してしまう危険性も持っている。



### 4.3.2 事実設定

事実という語からは、すでに存在するもの、あるいはすでに起こったこと、といった過去の情報という語感を受けるが、本研究の知識表現では、事実とは固定化した実体または方法を指す。事実の設定では、過去のなんらかの情報を基準に人間が決定するため、実際には過去を意識するが、その対象が近未来も（少なくとも次回の問題解決の結果を適用するまでは）普遍的であるといえれば、それらはすべて事実として扱う。

事実設定では、計算機内部に設定された要求構造に対して、関連する事実情報を問題構造として付加する。まず、要求構造に含まれるプリミティブについて、要求の入力の際に使用した名称を手がかりに、該当する現実世界の情報を事実データベースの中から検索する。該当する情報が存在する場合は、要求構造に上書きされる形でプリミティブとそれらのコネクションが設定される。

要求構造におけるプリミティブと事実構造のプリミティブは、通常、計算機上で 1 つのプリミティブを共有することになる。つまり、PICCSS に表現された問題構造のプリミティブは、多くの場合、要求の一部であると同時に事実の一部でもある。このようにして、要求構造と事実構造が計算機の内部に生成されることによって、問題構造が出来上がり、問題明確化の作業が終わる。

事実情報には現実世界の情報すべてが対応すべきであるが、実際には、事実データベースが現実世界に代替し、計算機は事実情報をこの事実データベースから検索する。事実データベースには、事実情報がファクトとして設定されている。ここで、ファクトとは、事実を表わす基本単位である。ファクトは、テンプレートとほぼ同じデータ構造を持つが、テンプレートと異なるのは、構成要素であるプリミティブがインスタンスであるという点であり、個々の事実に対応した具体的なパラメータを持つ。

以上の問題明確化の作業により、プリミティブからなるグラフ構造が生成され、リレーションのパラメータやオペレータの時刻に値が設定される。設定された値は、その設定の方法により以下のように分類できる。

- デフォルトで設定された値：プリミティブのパラメータはクラス定義においてあらかじめデフォルト値が定義されている。計算機に設定されるプリミティブには、当初このデフォルト値がとられる。

- 要求として設定された値：プリミティブの値を要求として設定する場合は、問題構造が生成された後に、PICCSS のユーザインタフェースを介して計算機に直接設定する。

- 事実として設定された値： 事実情報にあたるファクトには、あらかじめプリミティブからなる構造とパラメータの値が設定されている。ファクトに設定されている値は、そのまま計算機内部に複写される。

- 計算機によって設定された値：問題解決のサイクルが実行された以降は、問題解決サイクルの中の制約処理の結果として、計算機によって（オペレータによって）いくつかのパラメータの値が設定される。

## 4.4 制約処理

問題構造を計算機内部に設定することにより、問題が明らかになる。計算機上に設定されたプリミティブは、要求を表わす部分と事実を表わす部分が混在し、表面的には両者は見分けがつかない。ここで問題とは、リレーションに設定された制約式が違反しているということを指す。制約処理では、この制約式を充足することによって、問題を解決する。制約処理は、以下に示すように、主に制約充足計算と状態遷移計算の２つから構成される。

制約充足計算は、設定された制約式について、いわゆる制約充足問題として解を求める処理である。PICCSS では制約式は線形等式または不等式であるため、連立１次不等式の解を求めることに相当する。これにより、要求や事実にあるさまざまな制約が充足され、計算機上の問題構造が、人間の要求と現実世界の事実の両方を同時に反映したものとなる。制約充足計算によって計算されたパラメータの値は、あらたな変更要求となる。

状態遷移計算では、制約充足計算によって求められたパラメータの値を実際に変更するための処理を行う。現実の問題解決では、パラメータの変更を行うことは、場合によっては現実の状況を変化させることに相当するため、変更にあたっては、その変更を行うことができるオペレータ、つまり何らかの行為との対応をとり、そのオペレータの適用によって変更が実現する。これは、対象と行為とのつき合わせを行なうことに相当し、ここで空間的な構造と時間的な構造が個々に対応づけられる。

上記の制約処理の結果、制約の充足が不可能な場合、制約緩和を行う。本研究では、制約緩和の方法を２つに大別している。１つは、制約式のレベルで制約緩和を行なう方法であり、これは各制約式にスラック変数を設定し、この変数の値に応じたペナルティを与え、その合計を最小とするように計算機で管理する。この方法については、５章で詳しく述べる。２つめの方法は、プリミティブのレベルで制約緩和を行なう方法である。このプリミティブのレベルでの制約緩和では、問題構造が変更される。例えば、違反した制約式をもつリレーションを、それと代替する１つまたは複数のリレーションと交換し、交換先のリレーションが持つ制約式に置き換えることにより結果的に制約を緩和する方法などがある。



### 制約充足計算と状態遷移計算の関係

制約処理において、制約充足計算では主に空間的な構造を決定し、状態遷移計算では、時間的な構造を決定する。本研究の問題解決の枠組みでは、マクロに見た場合は、空間的な問題構造の決定と時間的な問題構造を同時並行的に処理するが、ミクロに見ると、両者は個々に独立した処理となっている。ただ、図 4.2 に示すように、両者は密接に関係しあっており、この２つの処理が頻繁に切替わることにより、統合的な問題解決を実現している。

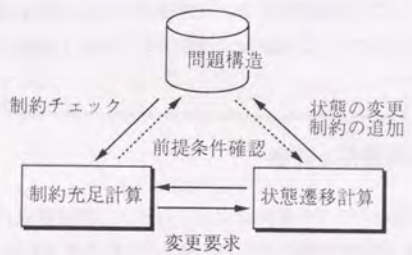



図 4.2: 制約充足計算と状態遷移計算の関係

図 4.2 において、制約充足計算では、与えられたリレーションに含まれる制約式について、制約充足計算を行なうが、結果として制約充足が可能であるか否かと、もし可能である場合には値をどのように変更すればよいかを返す。リレーションのパラメータやオペレータの時刻はここでは変更されない。実際の変更は、状態遷移計算を行なった後に行なう。つまり、制約の充足は状態遷移計算を経てはじめて完了する。制約充足計算で制約充足可能という結果が得られたとしても、その後の状態遷移計算に失敗し、そこで提示した変更方法では制約が充足しない場合もあり得る。制約充足計算は、ユーザにより問題構造の全体あるいは特定の範囲について制約充足処理を指定された場合と、以下の状態遷移計算によってオペレータ適用のための前提条件の真偽を調べる場合に実行される。

状態遷移計算では、ユーザによるパラメータや時刻の変更要求や、制約充足計算の結果提示された変更要求に従って実際に値を変更するが、要求された変更が可能である場合とそうでない場合があり、変更不可能な場合は状態遷移計算は失敗する。値の

変更は、特に任意に変更可能であるとあらかじめ定義されたものを除き、すべてオペレータによって実行される。ただし、オペレータの適用にあたっては、そのオペレータの条件部にあるすべてのリレーションが有効でなければならないため、場合によって制約充足計算を部分的に実行する。また、状態遷移計算を行なった結果、新たに先行関係制約が生成されるが、これらの制約もまた充足される必要があり、そこでも制約充足計算が利用される。つまり、状態遷移計算は制約充足計算をその内部で利用している。

このように、制約充足計算では、実際に制約を充足するために状態遷移計算を実行し、逆に状態遷移計算では、その各所で制約充足計算を実行しており、両者の関係は再帰的なものとなっている。PICCSS では、再帰のレベルが必要以上に深くならないように両者の切替えを基本的にユーザの指示によって行なっている。本研究の制約処理では、このようにして、制約充足計算と状態遷移計算の処理を頻繁に使いわけながら問題解決を進めることにより、本研究の課題である設計と計画の統合的な問題解決を可能としている。

## 4.5 問題構造の可視化と評価

制約処理により有効なすべての制約が充足されると、問題構造の可視化が行われ、解候補としてユーザに提示される。ここで可視化とは、各リレーションの持つパラメータを特定の視点によって取捨選択し、エンティティまたはプロセスごとにチャートに表示することである。このチャートは例えば要素形状の 1 次元表示、あるいは行為のガントチャート[1] などに対応する。

可視化のアルゴリズムは、以下のようになる。

**手順 1：** 可視化方法を $pc \in \{$エンティティ, プロセス$\}$ に設定する。$pc =$ エンティティ の場合は空間的な構造を可視化し、$pc =$ プロセス の場合は時間的な構造を可視化する。

**手順 2：** 可視化の視点を $vc$ に設定する。$vc$ はリレーションのサブクラスの中から選択する。$vc$ は通常、$pc =$ エンティティ の場合には空間上の軸を、$pc =$ プロセス の場合には時間軸に相当するクラスを設定する。

**手順 3：** 集合 $R_v$ に $vc$ を上位クラスとするリレーションのインスタンスを設定する。また、集合 $H_v$ に $R_v$ のいずれかの要素に対してコネクションをもつ $pc$ の下位クラスのインスタンスの集合を設定する。$i = 1$ とする。

[1]プロセスまたは行為を、縦軸を資源、横軸を時刻とするチャート上に表現したもの。



**手順 4：** 集合 $H_v$ から要素を 1 つ選択し $p$ とする。$p$ とコネクションをもつ $H_v$ の要素であるリレーションの集合を $R_p$ とする。集合 $H_v$ が空の場合は終了。

**手順 5：** 集合 $R_p$ のすべての要素を、y 方向を $i$、x 方向をその要素ののパラメータの値としてチャート上にプロットする。$i = i + 1$ として手順 4 へ。

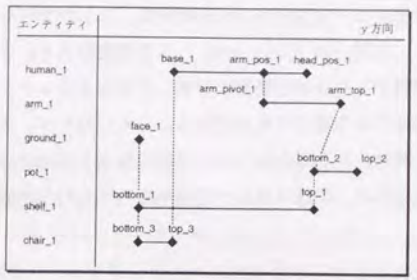



図 4.3: 問題構造の可視化

解候補をこのような方法で可視化し、人間の持つ要求が満足しているか、そして必要な事実がきちんと把握されているかを判断する。解候補が解となるためには、ここでの評価で一定の満足レベルに達していなければならない。ここでの評価は、問題構造として計算機に表現することができなかった部分（この部分には、人間が認識しているかあるいは事実情報として設定されているものに加え、この可視化の結果を見る前には人間すら知らなかった情報も含まれる）を新たに問題構造として取り込んでの評価であるため、必然的に主観的なものとなる。ただし、客観的な評価の基準としては、以下のような項目が考えられる。

- 制約違反：制約違反がある場合は、違反ペナルティ最小とする必要がある。もし、ペナルティをその解よりも減少させる手段があるにもかかわらずそれを行なわない場合は、計算機がその旨をユーザに伝える。
- 問題構造の分割：トップダウンで問題解決を行なう場合は、問題を分割することが有効であるが、問題構造そのものは分割した状態であってはならない。構築された問題構造がそのような状態にある場合、問題の表現に誤りのある可能性があるため、計算機はその旨をユーザに伝える。

## 4.6 トップダウン精緻化

問題解決の流れの中で、問題構造のすべての制約式が満たされている状態は、1つの解候補となる。しかし、ユーザの評価によりその解の粒度が不適切であると判断された場合は、トップダウン精緻化によりさらに粒度の細かいレベルの問題解決を行う。このトップダウン精緻化によって、ユーザの概念世界の隠された要求、あるいは現実世界の隠された事実が計算機上にあらたに展開される。

一般に人間の問題解決はトップダウンに行われる。トップダウンな問題解決の代表的なアプローチとして問題分割 [Ohsuga93] による方法がある。この問題分割による方法では、複雑な問題をより単純な問題に分割して解を求めるが、これは対象とする問題があらかじめ明確である場合に有効である。これに対して、本研究では精緻化による方法をとった。精緻化による方法では、問題の粒度を部分的に精緻化しながら解を求める。この方法は特に、問題の構造や範囲があらかじめ不明確な場合に有効となる。

### 精緻化の方法

トップダウン精緻化は、人間の持つ要求をより詳細に引き出すと同時に、事実をより詳細に設定することであり、計算機による自動化はそもそも不可能である。従って、トップダウン精緻化は、以下のように人間と計算機が協調して行う。トップダウン精緻化は、すでに設定されたプリミティブを展開することによって進められる。

問題構造の中でトップダウン精緻化を行なう範囲は、主に人間が選択する。計算機は人間の意思決定のための情報を提示する。展開するプリミティブとしては、制約式を持たないまたは制約が緩いリレーション（制約緩和した部分）、事実に該当しないリレーション、周囲より抽象度が高いリレーションなどが候補となる。抽象度が高いかどうかは、クラス階層の深さや、コネクション相手のクラスのクラス階層などによって判断する。

一方、選択されたプリミティブの展開の方法は、計算機が主に決定し、人間はそれを確認するという形でおこなう。計算機による展開方法としては、以下の3つの方法があり、これらの方法をユーザが使い分ける。

1. プリミティブのメンバーを展開：対象とする部分のプリミティブにメンバーが定義されている場合、そのメンバーにあたるクラスを計算機内部に展開する。メンバーはそのプリミティブを構成する要素であり、それらのメンバーに関するコネクションを完成させる過程で問題構造が精緻化される。



2. テンプレートの適用：テンプレートの適用では、すでに設定されたいくつかのプリミティブの構造と、あらかじめ登録されたテンプレートの内容とを比較し、両者の類似度が一定値以上の場合、そのテンプレートを適用させる。これにより、過去の知識をもとに関連する情報を問題構造に追加する。

3. 知識ベースの条件付き検索：対象とする部分に個別にプリミティブを追加する場合、知識ベースから要求するプリミティブのクラスを検索する。検索にあたっては、その検索条件として、コネクションの種類や、制約式の緩和条件（制約式の項の内容）などを指定する。

### 問題階層の管理

提案する問題解決の枠組におけるトップダウン精緻化によって、計算機上に生成されたプリミティブは原則として常に増加していく。つまり、問題構造は常に複雑化する方向へ向かう。いわゆるバックトラックに相当する後戻りは、問題明確化の過程の中では行なわず、精緻化された部分とそれに対応する上位の部分は、ともに精緻化後も問題構造上に残る。精緻化した対象を上位構造、精緻化の結果生成した部分を下位構造と呼ぶことにすると、上位構造と下位構造との対応関係は、計算機によって管理される。

この対応関係は、精緻化のサイクルの中で、上位の問題解決サイクルにおける解候補の問題構造と、現在の解候補の問題構造を比較しその差分をとることによって得られる。差分にあたる問題構造のノードであるプリミティブには、問題解決サイクル番号を設定する。一方、新たな問題構造の追加によって、冗長となった部分には同様に問題解決サイクル番号を設定し、追加プリミティブと対にして管理する。冗長性の判断は以下のようにして行なう。

- リレーションの冗長性：リレーションに含まれる制約式が、追加プリミティブの制約式が満たされる場合に常に満たされる場合、そのリレーションは冗長なリレーションといえる。

- オペレータの冗長性：オペレータの条件リストにあるリレーションがすべて真である場合、そのオペレータによらなくても追加プリミティブによって常にターゲットのパラメータが変更可能である場合、そのオペレータは冗長である。

提案する方法では、上位構造と下位構造との対応を、以上のような追加プリミティブと冗長プリミティブという関係によって問題解決サイクルごとに保持し、それらの情報をもとに、問題構造を部分的に簡素化する（追加プリミティブを冗長プリミティ

ブに戻す）。複雑な問題構造の上で実際に解を求める場合には、このような、上位構造と下位構造の関係を利用した問題の部分的な粒度の調整が有効となる。

一般に、同一のプリミティブが複数の形で精緻化されるような場合は、この上位構造と下位構造の対応関係が入り組んだ形になる。このような問題構造を持つ問題に対しては、特に従来の問題分割によるアプローチでは分割が困難であるため対応しきれない。本研究の精緻化のアプローチでは、そのような場合でも、複雑な問題構造の一部分を、対応する上位構造に置き換えることで、一時的に制約処理の対象から切り離しながら解を求めることが可能である。

## 4.7 事例による説明

ここで、花子さんの問題を再び取り上げ、提案する問題解決の枠組みに沿って問題解決を行なうことにする。花子さんの問題では、床と棚の位置関係、棚と花瓶の位置関係、床と花子さんの位置関係、棚の高さなど、図 4.4にあるようなさまざまな制約式が計算機内部に問題構造として設定されており、これらのすべての制約式が充足されていないことが“問題”となる。

この状態は、まずは空間的な構造として、水平方向と垂直方向の 2 つの制約式の集まりに分割され、垂直方向の制約式が充足されていない状態となっている。つまり、床 ground_1 から棚 shelf_1 を経て花瓶 pot_1 へ至る高さに対して、床 ground_1 から花子さん human_1 そして花子さんの腕 arm_1 へ至る高さが小さいことにより reachable_1 というリレーションの制約が満たされない。なお、図 4.4 では、制約式の各項は、問題構造として設定されたインスタンスであり、括弧内は、制約が属するリレーションを表す。

このようにすべての制約式を充足する解が存在しない場合、制約充足計算では各制約式に設定された優先度をもとに制約式のレベルで制約緩和を行なう。ここでは、リレーション standing_1 の制約が緩和され、結果として花子さんの足元 (hanako_1@base_1) が床 (ground_1@face_1) から浮き上がった状態となる。

問題構造のこの状態は、制約違反が含まれており、その違反は許容されるものではないため、続いてプリミティブのレベルの制約緩和を行ない、先に行なった制約式のレベルでの制約緩和を解消させる。そのためには、制約違反となっているリレーション standing_1 について、以下のような手順で代替リレーションを検索し、そのリレーションで standing_1 を代替することによって実質的な制約緩和を行う。まず、図 4.5 のように、緩和制約 standing_1 に含まれている制約を列挙する。続いて、それらの制約に含まれているパラメータを含み、かつ、それらのパラメータに値を代入しても制



human_1@base_1 = ground_1@face_1 (standing_1)
human_1@head_pos_1 - human_1@base_1 = head_height_1 (head_height_1)
human_1@arm_pos_1 - human_1@base_1 = arm_height_1 (arm_height_1)
arm_1@arm_top_1 - arm_1@arm_pivot_1 = arm_length_1 (arm_length_1)
arm_1@arm_pos_1 = arm_pivot_1 (arm_pivot_1)
arm_1@pos_1 = pot_1@pos_3 (at_1)
shelf_1@top_1 - shelf_1@bottom_1 = height_1 (height_1)
shelf_1@bottom_1 = ground_1@face_1 (on_ground_1)
shelf_1@pos_2 = pot_1@pos_3 (at_3)
pot_1@top_2 - pot_1@bottom_2 = height_2 (height_2)
pot_1@bottom_2 = shelf_1@top_1 (placed_1)
pot_1@bottom_2 ≤ arm_1@arm_top_1 (reachable_1)

図 4.4: 制約式の例

約式が違反しないリレーションを登録されたプリミティブの中から検索する。図 4.5では、standing_1 の代替リレーションとして、“stepping”がプリミティブのクラスの中から検索され、それを新たに問題構造に追加した結果、stepping_1 が設定された状態を示している。

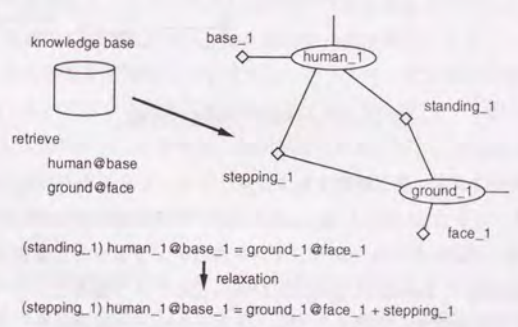



図 4.5: 制約の緩和

これによって、新たなリレーションのもつ制約は、standing_1 の制約に対して stepping_1 という項をもつため、制約違反は起こらない。従って、これは、この時点で設定された問題構造における解候補となる。

ここで得られた解候補は、何かに乗る (stepping) ことによって花瓶をとる、という

ことを意味しており、何に乗るのか、あるいはどうやって乗るのか、といったことは一切表現されておらず、あまり親切な解とはいえない。従って、問題の精緻化を行ない再度問題解決のサイクルを回す。

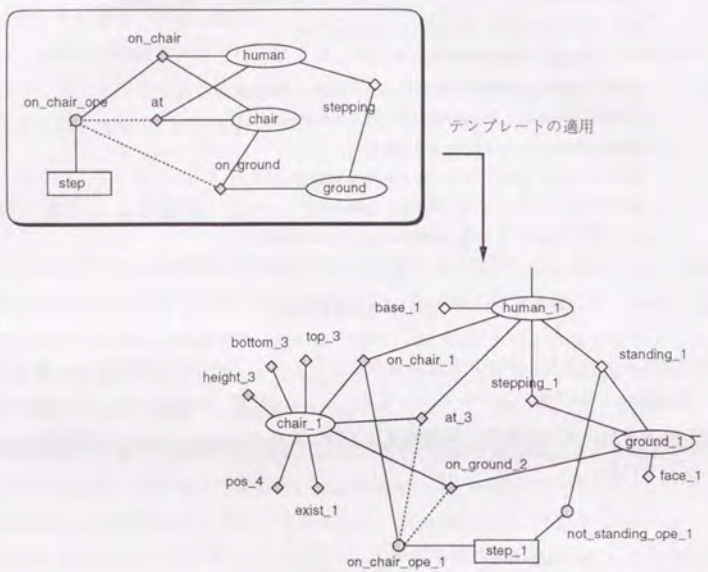



図 4.6: トップダウン精緻化の例

精緻化の処理ではまず、違反制約をもつリレーションである stepping_1 に対し、それとコネクションともつ human_1, ground_1 を手がかりとしてテンプレートを検索する。図 4.6 では、chair というリレーションを含むテンプレートが検索され、そこでの構造は、stepping_1, human_1, ground_1 からなるグラフ構造と、それぞれのクラス要求を満たしているため、このテンプレートがその時点の問題構造に展開される。展開に際して、テンプレートにある human, ground は既存のインスタンスに対応し、対応するインスタンスのない chair, step および on_chair, on_ground, at について、それぞれ chair_1, step_1, on_chair_1, on_ground_1, at_3 が設定され、同時にテンプレートのコネクションの合わせて、それらのインスタンスのコネクションが設定される。

このような精緻化により、問題構造には新たなプリミティブが設定され、この結果、内部の有効な制約式は変更され、新たな制約式について制約充足計算ではすべての制



約式が充足される。つまり、ここで新たに生成された解候補は、「実際に存在する椅子に乗ることにで花瓶をとる」ということを意味している。図 4.3 は、この解候補の高さ方向の空間的構造を表している。

## 4.8 開発システム

筆者らが開発したシステムである PICCSS (Problem Interactive Clarification and Concurrent Solving System) は、ユーザと計算機がインタラクティブに情報を交換することにより、問題構造を明確化し、同時に問題を解決するために利用される。本システムは、C++ と Motif を用いて UNIX ワークステーション上に実装されている。

PICCSS の入力情報は、知識ベース、事例ベース、事実情報、そしてユーザ要求の 4 つである。知識ベースにはプリミティブが、事例ベースにはテンプレートがあらかじめ登録されている。一方、PICCSS の出力情報としては、生成した問題構造モデルと、視点に応じてそれを可視化した出力チャートがある。

図 4.7 に PICCSS の操作画面の例を示す。ユーザと計算機とのインタラクションは、要求定義画面、問題構造表示画面、パラメータ編集画面、制約管理画面、出力チャート画面、知識ベース検索画面、テンプレート管理画面、事実情報管理画面の、合計 8 つの画面によって行われる。本章で述べた問題解決の流れに従い、まず、要求定義画面からユーザの要求をことばで入力し、知識ベース検索画面を利用してプリミティブを生成する。生成されたプリミティブは、問題構造表示画面にグラフィック表示される。続いて、事実情報管理画面より生成されたプリミティブと事実情報との対応をとり事実に該当するパラメータをあらかじめ決定する。続いて、パラメータ管理画面と制約管理画面により制約充足計算あるいは状態遷移計算を行い、その結果は出力チャート画面によって確認する。

以下に、各画面の主な機能を説明する。

**要求定義画面** 要求定義画面では、ユーザが要求の入力を短いセンテンスで行ない、そこに含まれるキーワードによって、知識ベース検索や事例ベースを検索する。検索されたプリミティブまたはテンプレートに従い、計算機上に問題構造を生成する。

**問題構造表示画面** 問題構造表示画面では、計算機上に設定された問題構造をグラフ表示する。また、この画面上でユーザは直接コネクションの設定、プリミティブの生成、プリミティブの削除、プリミティブの統合、分割などの処理を行なうことができる。

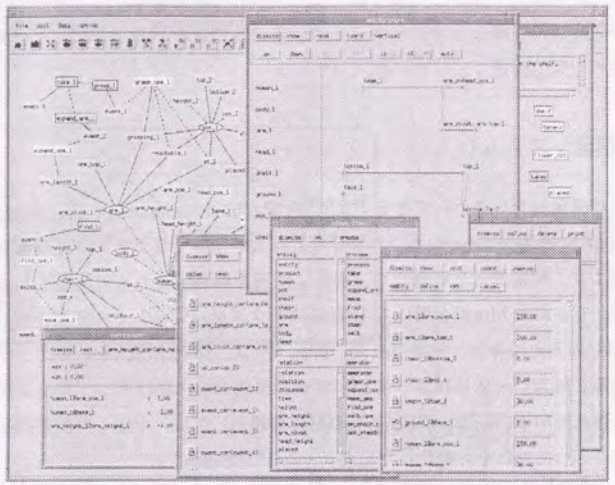


図 4.7: PICCSS 操作画面

**パラメータ編集画面** パラメータ設定画面では、問題構造に含まれるパラメータの値を確認し、場合によってはパラメータの値に関して、パラメータ変更禁止の指定、パラメータの有効化／無効化、パラメータ変更の設定および実行、変更のキャンセルなどが行なえる。

**制約管理画面** 制約管理画面では、制約充足計算の実行、制約の優先度指定、制約緩和禁止指定の他に、制約内容の確認、制約違反の確認、オペレータ先行関係の確認などが行なえる。

**出力チャート画面** 出力チャート画面では、問題構造を可視化し出力する。出力チャートでは、パラメータの大小比較、形状の一次元表示、ガントチャート表示などによって問題構造を確認する他に、視点の変更、制約違反の図的な確認、オペレータの先行関係の確認などが行なえる。

**知識ベース検索画面** 知識ベース検索画面では、登録プリミティブの確認、キーワードによる検索などを行なうことができる。また、新たな知識としてのプリミティブの定義を行なった場合、それらを知識ベースとして登録することもできる。

**テンプレート管理画面** テンプレート管理画面では、登録テンプレートのリスト表示、内容の確認の他に、テンプレートの定義も可能である。また、問題解決の過程におけるテンプレートのマッチング、テンプレートの適用もこの画面で行なう。



**事実情報管理画面** 事実情報管理画面では、事実情報であるファクトのリスト表示、内容の確認、そして、事実情報の新たな定義を行なうことができる。また、問題解決の事実情報設定では、問題構造の一部と事実とを対応づけることで問題構造に情報を追加する。

## 4.9 4章のまとめ

本章では、本研究で提案する問題解決の方法の基本的な枠組みを説明した。まず最初に、問題解決の流れを概観し、そこに含まれるいくつかの基本的な処理単位について簡単に説明した。本研究の問題解決の枠組みには、問題そのものを明確化する過程が含まれており、問題明確化としてその方法について説明した。問題明確化では、要求の設定と事実の設定が中心となる。そこで明らかになった問題構造に対して行なう制約処理について、続いて簡単に説明した。また、制約処理の結果をユーザに提示する際に行なう可視化の方法を示した。さらに、トップダウンに精緻化しながら以上のサイクルを繰り返す方法について述べ、具体的な例題によってその動作を確認した。また、これらの問題解決の枠組みは、実際に計算機上に実装されたプログラムによって実現されており、この計算機プログラムの内容についても簡単に紹介した。

# 第 5 章

# 制約緩和手法

## 5.1 はじめに

前章の問題解決の枠組みの説明では、制約充足計算の方法については深く触れなかった。通常、制約式が線形の等式または不等式の場合には、一般的な連立不等式の解法によって対応できる。しかし、より現実的な問題を考える場合には、制約の緩和が重要な意思決定要素となり、いかにこの制約緩和を現実に則した形で管理できるかが重要となる。本章では、制約緩和を前提としたより現実的な制約充足の手法を提案する。ここで提案するアルゴリズムは、制約違反の状態をそれぞれの違反制約に対する2種類のペナルティの合計によって表現し、すべての制約のペナルティの合計を最小化する最適化問題にいったん置き換えて解を求める。

本章では、提案するアルゴリズムを、スケジューリング問題を例題として、その動作の説明をし有効性の検証を行なう。2節では，本研究が対象とする問題の定式化を行い，続く3節で，ペナルティ伝播グラフを用いた本手法のアルゴリズムを説明する．4節では，現実のデータを利用しての評価実験とその結果を紹介し，それをふまえて5節で，提案するアルゴリズムに関する考察といくつかの論点を述べる．6節では総括的な議論を行なう。

## 5.2 数学的定式化

ここでは，スケジュール対象を，最終的に事象と制約のみによって定式化する．例えば，オペレーションは，開始と終了という2つの事象とそれらを関係づける制約によって表現され，必要に応じて中間的な事象が追加される．また，納期や資材到着などは単独の事象として表現される．なおここでは，スケジュール上の制約はすべて，2つの事象の時間的な関係を定義する形式に統一する．

ここで，事象 $i$ の時刻を $T_i$ とし，制約 $k$ が関係する 2 つの事象を $a_{k,i}$ によって表し，さらにその時の 2 つの事象の間隔（オペレーション時間や運搬時間などに対応）を $D_k$ とすると，制約 $k$ は以下の式で表される．

$$\sum_i a_{k,i} T_i \geq D_k \tag{5.1}$$

なお $a_{k,i}$ には，制約 $k$ にとって事象 $i$ が先行事象の場合に $-1$，後続事象の場合に 1，それ以外の場合に 0 が設定される．

一般に，予期せぬ状況変化があった場合，以前に作成したスケジュール上では，制約のいずれかが違反した状態となっている．スケジュール修正では，このような初期状態における制約違反を柔軟に表現できる枠組が要求される．そこで本研究では，式 5.1 で表される制約はすべて緩和可能とし，違反が発生した場合の緩和量 $\lambda_k$ と違反ペナルティ $\rho_k$ を以下の式で定義する．

$$\lambda_k = max\{D_k - \sum_i a_{k,i} T_i,\ 0\} \tag{5.2}$$

$$\rho_k = \begin{cases} \alpha_k + \beta_k \lambda_k, & \text{if} \lambda_k > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

ここで，$\alpha_k$ は，制約違反が発生した時点で生ずるペナルティを表し，$\beta_k$ は，違反のレベルに比例して生ずるペナルティの割合を表す．この 2 つのペナルティ係数は，制約の重要度を表わすものであり，ともに 0 以上の値をあらかじめ設定しておく．

スケジュールを構成する事象の中には，過去の事象となったものや納期のように変更できないものが存在する．これを固定事象と呼び，その時刻を $B_i$ で表す．式 5.1 が緩和可能な制約であるのに対し，この条件は絶対的な制約として以下の式で表される．

$$T_i = B_i, \quad \forall i \in S^{fix} \tag{5.4}$$

ここで，$S^{fix}$ は，固定事象の集合である．

本研究で対象とする問題は，これらの固定事象を考慮しながら，制約を緩和することによるペナルティ量の合計を最小化する問題である．すなわち，式 5.4 のもとで，式 5.5 を求める最適化問題として定式化することができる．



$$\sum_k \rho_k \rightarrow \min \tag{5.5}$$

## 5.3 制約緩和アルゴリズム

### アルゴリズム

一般にスケジューリング問題は，ＮＰ困難なクラスに属するため，実際的な規模の問題に対しては厳密な最適化計算をあきらめなければならない．本研究では，以下のアルゴリズムに示すように，“ペナルティ伝播グラフ”を用いて初期解を反復的に改善していく方法をとる．なお，以下アルゴリズムは，手順 1-2 および手順 1-3 でさまざまな探索法に対応できるしくみになっている．通常の山登り法を用いる場合，違反制約としては，ペナルティ $\rho_k$ が最大の制約を選択し，スケジュール方向としては，前向きと後向きの両方について一度計算し，後に値のよい方の候補解を選択する．

**手順 1-1：** 初期解（現行スケジュール）にあたる事象と制約を生成し，候補解とする．

**手順 1-2：** 候補解のペナルティ合計を計算し，0 である場合はその候補解を最終解として処理を終了する．また，反復回数や前回の候補解との比較をもとに，探索法に従って処理の終了を判定する．

**手順 1-3：** 探索法に従い，候補解の違反制約の中から，制約を 1 つ選択し $k^*$ とする．また，スケジュール方向を，前向き（先行事象優先）か後向き（後続事象優先）のいずれかとする．

**手順 1-4：** ペナルティ伝播グラフを生成する（次節参照）．

**手順 1-5：** 生成したペナルティ伝播グラフにおいて，最小カット問題（ソースノードと終端ノードの間をアークの重みの合計が最小になるよう 2 つに分割する問題）を解く．求めた最小カット上のアークに対応する制約が，ペナルティの伝播先制約となる．なお，最小カットが存在しない（フローが無限大である）場合は，手順 1-2 へ戻る．

**手順 1-6：** 時刻更新量 $\delta$ を，以下の式により計算する．

$$\delta = \theta \min_k \{\lambda_{k^*},\ \tau_k\} \tag{5.6}$$

ここで $\theta$ は，スケジュール方向であり，前向きが $theta = 1$，後向きが $theta = -1$ となる．また，$\tau_k$ は，制約 $k$ がもつ，時刻更新に対する余裕量であり，以下の式で計算する．

$$\tau_k = \begin{cases} \infty, & \text{if} \quad \sum_i a_{k,i} T_i - D_k \leq 0 \\ & \text{or} \quad \sum_{i \in S^0} a_{k,i} = 0 \\ \sum_i a_{k,i} T_i - D_k, & \text{otherwise} \end{cases} \tag{5.7}$$

ここで $S^0$ は，手順 1-5 で求めた最小カットよりソースノード側にあるノードの集合である．

**手順 1-7：** $S^0$ に含まれるすべてのノードに対応する事象に対して，その時刻に $\delta$ を加算し，その結果を新たな候補解とする．

**手順 1-8：** 対象制約 $k^*$ のペナルティが 0 より大きい場合は手順 1-4 へ戻る．それ以外は手順 1-2 へ戻る．

### ペナルティ伝播グラフの生成

前節のアルゴリズムにおける手順 1-4 の，ペナルティ伝播グラフの生成について説明する．以下に述べるアルゴリズムでは，事象と制約の接続関係をもとに，違反状態にある対象制約 $k^*$ から順に，伝播経路となり得る制約を列挙していく．この結果，事象に対応するノードと制約に対応するアークからなるペナルティ伝播グラフが生成され，アークの重みにはペナルティ増加量が設定される．なお，伝播経路の判定は，表 5.1 に従って行われる．表の中で，○は，事象 $s$ から制約 $k$ を経由して他方の事象に伝播が行われることを表す．

表 5.1: 伝播経路の判定

| 状態＼更新方向 | $a_{k,s}\theta > 0$ | $a_{k,s}\theta < 0$ |
|---|---|---|
| $\sum_i a_{k,i} T_i - D_k = 0$ | × | ○ |
| $\sum_i a_{k,i} T_i - D_k < 0$ | × | ○ |
| $\sum_i a_{k,i} T_i - D_k > 0$ | × | × |

**手順 2-1：** ソースノードと終端ノードをそれぞれ 1 つずつ生成する．



**手順 2-2：** スケジュール方向が前向きの場合は $k^*$ の後続事象，後向きの場合は $k^*$ の先行事象に対応するノードを生成し，スタック $E$ に積む．他方の事象はいったん固定事象とする．また，ソースノードからそのノードへのアークを生成し，無限大の重みを与える．

**手順 2-3：** スタック $E$ が空の場合は，処理を終了する．それ以外の場合，スタック $E$ からノードを 1 つ取り出し $i$ とする．

**手順 2-4：** 事象 $i$ が固定事象である場合，ノード $i$ から終端ノードへのアークを生成し，手順 2-3 へ戻る．アークの重みは無限大とする．

**手順 2-5：** 事象 $i$ に関係するすべての制約の集合をスタック $C$ に積む．

**手順 2-6：** スタック $C$ が空である場合は，手順 2-3 へ戻る．それ以外の場合，スタック $C$ から制約を 1 つ取り出し $k$ とする．

**手順 2-7：** 制約 $k$ に属する $i$ ではない他方の事象を $j$ とする．事象 $j$ に対応するノードがすでに生成されているか，制約 $k$ が $i$ から $j$ の方向への伝播経路とならない場合（表 5.1 参照）は，手順 2-6 へ戻る．

**手順 2-8：** 事象 $j$ に対応するノードと，$i$ から $j$ へのアークを生成する．このアークの重みは，制約 $k$ が違反状態である場合は $\beta_k$，そうでない場合は $\alpha_k + \beta_k$ とする．

**手順 2-9：** スタック $E$ に $j$ を積み，手順 2-6 へ戻る．

### 例題による説明

提案するアルゴリズムの動作を理解するために，図 5.1 のような例題を考える．ここでは，2 つの機械A，Bに，それぞれ 2 つずつのオペレーション op_1, op_2 と op_3, op_4 があらかじめスケジューリングされている．図 5.1 の数字は事象番号，括弧内の数字はその時刻である．表 5.2，表 5.3 および図 2 (a) は，このスケジュールを事象と制約で表現したものである．対象となる 7 つの事象のうち事象 1，4，7 の 3 つは固定事象である．制約としては表 5.3 のような 8 つが存在し，その中には制約 a と制約 b，あるいは制約 f と制約 g ように，通常の等式制約が 2 つの不等式制約に分けられたものも含まれる．

制約のペナルティ係数は，それぞれの制約の重要度をもとに設定されている．例えば，op_1 の処理時間の制約は重要であるが，それ以上に op_1 と op_2 あるいは op_1 と op_4 の先行関係は重要であることが，表 5.3 から読み取ることができる．

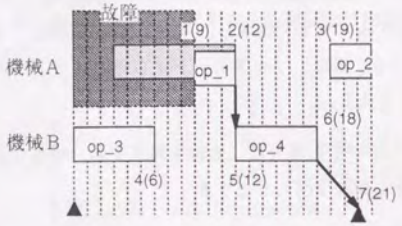



図 5.1: 例題の説明

表 5.2: 事象データ

| 記号 | 時刻 | 区分 | 説明 |
|---|---|---|---|
| 1 | 9 | 固定 | op_1 の開始 |
| 2 | 12 | 可変 | op_1 の終了 |
| 3 | 19 | 可変 | op_2 の開始 |
| 4 | 6 | 固定 | op_3 の終了 |
| 5 | 12 | 可変 | op_4 の開始 |
| 6 | 18 | 可変 | op_4 の終了 |
| 7 | 21 | 固定 | 納期 |

この例題では，予期せぬ状況変化として，機械Aが故障した状況を想定し，op_1 の開始である事象 1 が本来の時刻 3 から時刻 9 に変更されている．従って，この初期解において，制約 a は既に違反した状態となっており，式 5.2, 式 5.3 より計算されるペナルティ合計は 1400 である．

ここで，スケジュール方向を前向きとして，この例題を先のアルゴリズムに適用させてみよう．まず，違反制約 a の後続事象である事象 2 に対応するノードを生成し，同時にそのノードとソースノードを結ぶ重み $\infty$ のアークを生成する（手順 2-1, 2-2）．そしてまず，対象として事象 2 が取り上げられ（手順 2-3），事象 2 に関係する制約 a，b，c，d について，それぞれ伝播経路の判定が行われる（手順 2-5, 2-6, 2-7）．その結果，制約 d が伝播可能と判定され，対応するアークと，伝播先の事象 5 に対応するノードが生成される．制約 d は違反していないので，対応するアークの重みは 900 となる（手順 2-8）．次に，事象 5 が取り上げられる（手順 2-3）．事象 5 と関係する制約 e，f，g について伝播経路の判定が行われ（手順 2-5, 2-6, 2-7），今度は制約 e，制約 f に対応するアークと，伝播先の事象 4，事象 6 に対応するノードが生成される．アークの重みはそれぞれ 110 と 440 である（手順 2-8）．続いて，事



表 5.3: 制約データ

| 記号 | 先行事象 | 後続事象 | 間隔 | 係数 $\alpha_k$ | 係数 $\beta_k$ | 説明 |
|---|---|---|---|---|---|---|
| a | 1 | 2 | 9 | 500 | 150 | op-1 処理時間 |
| b | 2 | 1 | -9 | 500 | 150 | op-1 処理時間 |
| c | 2 | 3 | 0 | 900 | 0 | op-1, 2 の先行関係 |
| d | 2 | 5 | 0 | 900 | 0 | op-1, 4 の先行関係 |
| e | 5 | 4 | -6 | 30 | 80 | op-3, 4 の間隔 |
| f | 5 | 6 | 6 | 400 | 40 | op-4 の処理時間 |
| g | 6 | 5 | -6 | 400 | 40 | op-4 の処理時間 |
| h | 6 | 7 | 0 | 300 | 30 | op-4 の納期 |

象 4 が取り上げられるが，事象 4 は固定事象であるため，終端ノードとアークで結ばれ，アークには重み $\infty$ が設定される（手順 2-4）．また，事象 6 についても，関係する制約についてさらに伝播経路の判定が行われるが，制約 h が伝播経路とならないため，展開が終了する．

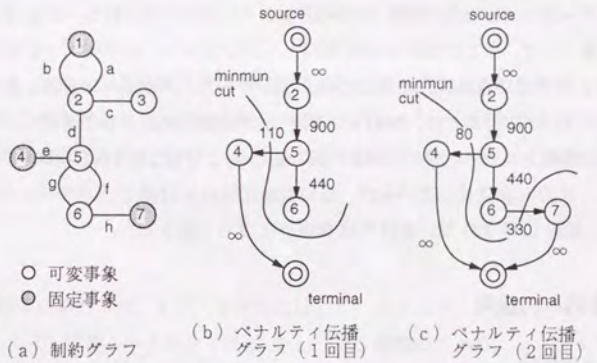
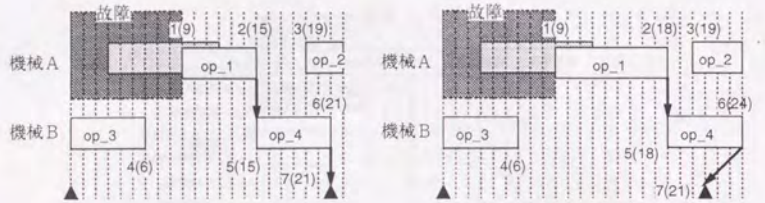



図 5.2: 例題のグラフ表現

図 5.2(b) は，こうして生成されたペナルティ伝播グラフである．各アークに設定された重みをもとに最小カットを計算すると，制約 e が伝播先の制約として選択される（手順 1-5）．時刻更新量 $\delta$ には，制約 h の余裕量 3 があてられ（手順 1-6），最小カットのソースノード側の事象である事象 2, 5, 6 の時刻に 3 が加算される（手順 1-7）．図 5.3(a) に，このペナルティ伝播によるスケジュール更新結果を示す．ここでは，制約 a と，制約 e が違反しているが，ペナルティ合計は 1220 に減少している．




（a）時刻変更1回目　　（b）時刻変更2回目

図 5.3: スケジュール修正過程

2回目の反復では，1回目と同様，事象2に対応するノードおよび，それとソースノードを結ぶアークが生成され，そしてその後，制約 d, e, f に対応するアークと，伝播先の事象 4, 5, 6 に対応するノードが生成される．しかしその後，事象 6 に関係する制約についての伝播経路の判定の結果，今度は制約 h が伝播経路となり，制約 h に対応するアークと，伝播先の事象 7 に対応するノードが生成される．事象 4 と事象 7 は固定事象なので，対応するノードは終端ノードと重み $\infty$ のアークで結ばれる．なお，制約 e はすでに違反状態にあるため，対応するアークの重みは前回と異なる．2回目の最小カットの計算では，制約 e と制約 h が伝播先制約として選択される．以上の結果と，更新されたスケジュールを，図 5.2(c) および図 5.3(b) に示す．ここでは，制約 e と，制約 h が違反しているが，当初の違反制約 a は満足し，ペナルティは制約 e が 510，制約 h が 390 で，合計では 900 へとさらに減少した．

## 5.4 事例への適用

### 実験データの説明

本章では，提案するアルゴリズムの有効性を検証するために数値実験を行う．実験で用いるデータは，現実の化学プラントのバッチプロセスの運転データをベースとし，疑似的にいくつかの予期せぬ状況を発生させたものである．

対象ラインは，図 5.4 に示すように，7つの装置（機械）からなる5つの工程によって構成される．このうち，A工程とB工程はバッチ工程，C工程は連続工程，D工程はバッチ工程であり，最後のE工程は再び連続工程となっている．実験で用いた品目 a，品目 b，品目 c の生産は，各々１８のオペレーションから構成されている．

これらのオペレーションをスケジューリングするには，オペレーション時間や先行



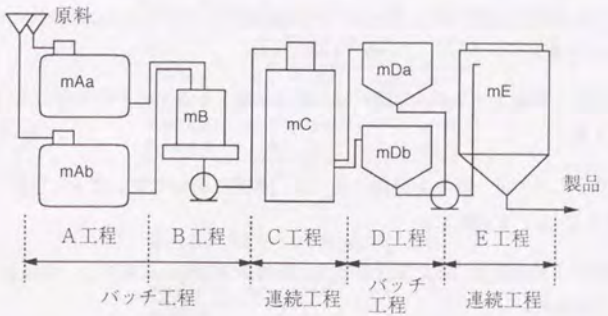



図 5.4: 対象ラインの概略

関係の他に，同時開始や同時終了などの制約を満足させ，そしてその上で連続工程の効率よい運転を考慮しなければならない．従って，バッチ工程に属する各オペレーションは，通常，フォワードやバックワードを組み合せた形でスケジューリングしなければならず，その手順はスケジュール修正の場合さらに複雑となる．

実験では，品目 a，品目 b，品目 c の３種類の生産と，これらの品目の生産の間で生じる段取作業を対象とし，状況変化に対応して再スケジューリングを行う．初期スケジュールは合計 62 のオペレーションによって構成されており，142 の事象と 325 の制約によって表現される．

### 実験方法

実験を行うにあたって，まず，各制約に対して，ペナルティ係数を，ペナルティ係数 $\alpha$ として 10 種類，ペナルティ係数 $\beta$ として５種類の中から，ランダムに選んで設定した．続いて，全事象数の 10 % を固定事象とした．固定事象の選定にあたり，その分布をスケジュール対象範囲に対して以下のように変えることにより，４種類の問題グループを定義した．

1. 前方固定型：前半 30 % に集中
2. 後方固定型：後半 30 % に集中
3. 前後固定型：前半 15 % と後半 15 % に集中
4. 均一固定型：均一に分散

これらの問題グループごとに，制約違反を疑似的に発生させた初期スケジュールをそれぞれ 10 個ずつ生成した．発生させた制約違反は，以下の３つの操作のいずれかによるものである．

1. 操作A：事象をランダムに選択し，固定事象とした上でその時刻に $\pm\Delta t$ を加算する．
2. 操作B：オペレーション時間にあたる（等式）制約をランダムに選択し，間隔値 D に $\Delta t$ を加算する．
3. 操作C：先行関係にあたる（不等式）制約をランダムに選択し，間隔値 D に $\Delta t$ を加算する．

ここで，$\Delta t$ は，平均オペレーション時間の 50 %とした．

実験では，ペナルティ合計の比：(修正後のペナルティ合計 / 修正前のペナルティ合計)×100 およびペナルティ伝播処理の反復回数を比較した．また，修正解の良さを検討するために，以下の２通りの探索法を組み込んだ数値実験も併せて行った．

1. 探索法１：手順1-3 において，選択する制約を違反状態にあるすべてとする．選択制約の候補およびスケジュール方向の候補それぞれに応じて問題を分枝させ，同時に保持可能な解候補の数を４として最良優先探索を行う．
2. 探索法２：手順1-3 では，ペナルティ最大の制約を選択するが，ステップ２で解が改善しない場合でも分枝を続け，以後２回の反復でも改善が見られない場合にはじめて分枝を終端する．探索は，探索法１と同様，同時に保持可能な解候補の数を４として最良優先探索を行う．

#### 実験結果

実験結果を，表5.4，表5.5，表5.6に示す．表5.4 に見られるように，固定事象の分布の違いに応じて，初期ペナルティに対して平均 20.61 %から 47.11 %まで，初期スケジュールを改善することができた，なお，先方固定型や後方固定型に比べて，前後固定型や均一固定型の方が，よりスケジュールの改善が困難であった．

停止までの反復回数は，表5.5 に見られるように，平均２回程度であったが，３回以上のケースが全体の 13 %，５回以上のケースが５%存在した．また，表5.6 に見られるように，対象とする違反制約の選択のしかたや，局所最適解を避けるための手段を付加することにより，結果がさらに改善することが確認できた．本実験のケースの場合，改善幅はそれぞれ，前者が平均 1.41 ポイント，後者が 2.81 ポイントであった．



表 5.4: 修正後のペナルティ合計（%）

| 問題グループ | 操作A | 操作B | 操作C | 平均 |
|---|---|---|---|---|
| 前方固定型 | 29.72 | 4.71 | 27.40 | 20.61 |
| 後方固定型 | 25.88 | 14.86 | 48.12 | 29.62 |
| 前後固定型 | 31.97 | 35.07 | 42.99 | 36.68 |
| 均一固定型 | 41.68 | 42.53 | 57.11 | 47.11 |

表 5.5: 停止までの反復回数（回）

| 問題グループ | 操作A | 操作B | 操作C | 平均 |
|---|---|---|---|---|
| 前方固定型 | 3.7 | 1.4 | 1.4 | 2.2 |
| 後方固定型 | 2.9 | 1.0 | 1.2 | 1.7 |
| 前後固定型 | 1.9 | 1.2 | 0.8 | 1.3 |
| 均一固定型 | 1.4 | 1.2 | 0.6 | 1.1 |

## 5.5 考察と議論

状況に対応してスケジュールを修正する場合，以前に作成した他のスケジュールとの整合性や生産実績への対応が重要となる．この際，対象スケジュールの中に偏在する修正不可能な事象を柔軟に扱うことが要求される．また，すでに行った生産指示の変更容易性などを考慮すると，前回のスケジュールと比較して修正箇所が少ないということは修正スケジュールの評価につながる．これらの点を考えると，本アルゴリズムの反復的な解の改善アプローチは，スケジュールを最初からやり直す従来の方法と比較して，より現実的であるといえる．

数値実験では，典型的な化学プラントの運転データをもとに実験データを作成したが，そこで行った実験は，非常に多様でかつ実際的な状況を広くカバーしている．例えば，実験で発生させた制約違反についていえば，操作Aは，納期変更や資材の到着遅れに対応し，操作Bは，負荷（進捗）変動や設計変更，操作Cは，緊急オーダーや，

表 5.6: 探索による効果（%）

| 問題グループ | 山登り法 | 探索法１ | 探索法２ |
|---|---|---|---|
| 前方固定型 | 20.61 | 20.12 | 16.84 |
| 後方固定型 | 29.62 | 29.62 | 27.16 |
| 前後固定型 | 36.68 | 33.11 | 32.52 |
| 均一固定型 | 47.11 | 45.54 | 46.29 |
| 平均 | 33.51 | 32.10 | 30.70 |