設計過程表現論

武田 英明

1990 年度博士論文

設計過程表現論

指導教官 吉川弘之教授

東京大学大学院工学系研究科

精密機械工学専攻

博士課程 87071

武田英明

概要

本論文は機械設計の過程がどのようなものであるかを分析し、推論可能な設計過程モデルを提案するものである。

近年、設計の高度化、複雑化、開発時間の短縮に伴い、CAD(Computer-Aided-Design) システムの高度化が望まれている。そのひとつの方向は CAD システムをより知的なシステムとすることである。そのような知的な CAD システム (インテリジェント CAD) は設計の本質的な支援を行なう必要がある。すなわちインテリジェント CAD システム開発の基礎として設計の研究は不可欠である。中でも設計過程の解明はインテリジェント CAD の中核として重要であるにもかかわらず、これまで大きな発展はみられなかった。本研究では設計過程を総合的に分析して、インテリジェント CAD のための基礎理論を提出することを目的とする。

本論文は 9 章からなるが、大きく分けると 2 部からなる。前半では、設計過程を認知的な方法で分析することについて述べる。後半では、設計過程を論理的な枠組みで形式化することについて述べる。

1 章では、本論文の目的と構成を述べたのち、既存の研究を概観して位置付けることにより、本論文が扱う領域について明確化を行なう。ここにおいて、本論文の対象は、設計者の思考としての設計過程であることを示す。

2 章では、認知的方法による設計過程の分析について述べる。ここでは設計を実験的に観察する方法である設計実験を定義し、その実施の方法について論じる。設計実験は主にプロトコル解析で行なわれるので、設計の観察に適合するプロトコル解析の方法を議論する。次にこの議論の結果を用いて、実際に行なった設計実験の手法と結

果について議論を行なう。ここでは、本研究で行なった12実験、総計40時間のプロトコル解析について考察する。

3章では、2章で示した設計実験を用いて、認知的な設計過程モデルを議論する。まず、設計過程の流れを対象の進化として示す。次に設計における問題解決を設計サイクルとして示す。設計サイクルとは「問題提起」、「提案」、「展開」、「評価」、「決定」からなり、これが繰り返されて設計が進行する。次に設計過程における知識を議論する。プロトコルから知識を抽出して、それらを設計サイクルとの関連と知識相互の関連を論じる。さらに、設計で用いられる概念を抽出して設計者の概念構造について議論する。ここではプロトコルから概念を抽出して概念のネットワークをつくり、そのネットワークをコネクショニスト的アプローチにしたがって用いて、設計過程における重要概念の変化や、設計に関わる概念間の関係の強さなどを示す。

4章では、認知的な分析の結果を踏まえ、設計の論理による定式化の基本的な枠組みについて議論する。まず、3章の結果を用いて、設計を知識による問題解決と見た場合の特徴を述べる。設計を問題解決としてみたとき、推論過程の複雑さ、対象記述の不完全性、知識の部分性などの特徴を持つ。これらの特徴を論理的枠組みにおいて表現するには、新たに、前提や結論自体の推論、部分的な状態の記述、多様な推論の実現などが必要となることを示す。

5章では、設計における推論を論理的な枠組みの中で議論して、設計過程の論理的モデルを提案する。まず、設計を論理におけるアブダクションとして定義する。このとき、設計過程とは、知識を用いてアブダクションと演繹を繰り返すことにより、対象の記述を詳細化していく過程である。次に、設計サイクルの推論をこの枠組みで表現して設計過程モデルを提案する。設計サイクルはアブダクションと演繹に加え、サーカムスクリプションとメタ推論の組合せとして表現される。

6章では動的な変化と部分的な状態を表現する論理における意味論を用いて設計過程表現の意味論を議論する。ここでは、部分意味論とその多重世界への拡張であるデータ意味論を導入して、それらを用いて設計過程を表現する。その中では設計の流れは可能世界間の関係として、要求仕様は必然様相で、可能な候補は可能様相で表現される。また、知識も様相を含んだ命題で示される。

7章では、5章と6章で議論した設計過程モデルを推論システムとして実現する。

ここでは、設計シミュレータと呼ばれるシステムを構築して、設計過程モデルに基づき、計算機上で設計過程をシミュレートする。設計実験で得られたプロトコルを元に設計過程のシミュレーションを行ない、システムの動作を示すとともに、設計過程モデルの有効性、妥当性を示す。

8章では、これまで議論した設計過程モデルをもとに、インテリジェントCADシステムの持つべき機能と構成について述べる。ここでは、インテリジェントCADシステムとは概念を鍵に統合化されるシステムであり、さらにその中で知識は設計を行なうにつれ構造化される自己構造化機能を持つものであることが示される。

9章では、本論文の結論と展望を述べる。

目次

第 1 章

序言

本章では本論文の目的と構成を述べる。その上で、既存の設計研究を概観し、本論文の位置づけと研究の基本方針について述べる。

1.1 本研究の目的

本研究は設計過程を人間の思考行為という視点から考察するものである。そして、人間の思考行為の支援として CAD(Computer-Aided-Design) の開発を最終的な目標としている。

人間はいにしえからものを作ってきた。しかし、設計というものが認識され、設計を行なってものをつくるという過程が確立したのはそう古いことではない。単にものをつくるという行為と設計を通じてものをつくるということの間には、その過程が異なるということにとどまらず、その意味において大きな差異がある。ものをつくる行為は、その初めにおいては、自然の模倣であったと思われる。やがてこれを繰り返すことにより、自然には存在しないものもつくることが可能になった。しかし、この時点では生産者はものの働きと存在の関連の原理を知っていたわけではなく、経験的に理解していただけであり、それゆえ要求から作り上げるまでの過程は一体化・未分化の状態であった。しかし、その状態では、その行為は他人に伝えられない。技能として伝承はできるものの、それでは過去の経験の蓄積のみで新規のものへの対処の仕方は伝わらない。ものをつくる行為は単に過去の繰り返しだけでは成り立たない。というのはものに対する要求は変化するからであり、また用いる材料やつくる方法も変化するからである。そして継承を可能にするために、製造のための技術の確立と製造にいたるまでの思考の明示化、すなわち設計の確立がなされたといえよう。正しく継承されるには、まずいかにものを製作したか、ということが明示化されなければならない。単に技能として伝えられるのではなく、未知のものに適用できるようなものでなければならない。そこで製造技術というかたちで体系化・構造化することによってそれを可能にした。

しかし、これだけでは不十分である。いかに製造するかだけではなく、いかにして要求からそこまでたどり着いたかについても説明できなければならない。それが設計という分野の発生の根源であろう。しかし、これは製造の問題とは異なり、物理世界の話でなく、思考の世界の話であるので、まずその対象を明示的に表すこと自体が問題になる。その上でその方法の体系化・構造化がなされなければならない。この意味では設計は人間の思考活動のうち、最も早く認識されたもののひとつであるといえよう。ものをつくるまでにいたる思考の過程を明示化し、さらに体系化すること、そし

てそれを利用して新たなものつくること、それが設計である。このような製造技術の確立と設計という分野の成立により、ものをつくる行為は継承可能になったと思われる。

本研究では設計を以上のようにとらえ、思考過程の明示化・構造化として考察する。思考過程の研究は古くは哲学として、また心理学として、そして近年は人工知能の研究として行なわれきた。特に、近年の人工知能の活発な研究活動は人間の思考過程の考察に多様かつ目新しい視点を提供した。そこで本研究では、人工知能研究の方法・成果を利用することにより、設計の過程を考察していく。

本研究の目標は設計者を知的に支援するCAD、すなわちインテリジェントCADの動作原理を示すことである。すなわち、インテリジェントCADはどうように動作し、どのような方法で設計者を支援するかについての基本原理を示すことである。しかし、本研究では直接それを研究するのでなく、まず設計自身を研究し、その結果を用いてインテリジェントCADを議論する。

1.2 本研究の構成

設計はさまざまな側面を持っている。例えば、社会的な側面としての設計は生産過程の一過程であり、あるいは物理的な側面としての設計は物理的に可能な実体を産みだすことである。そういった多様な側面を持つ設計に対して、本研究では思考としての設計に注目する。このために、ここでは設計の思考としての側面が強く現れる設計の初期の段階を中心に考察していく。また、個々の設計の詳細を議論するのではなく、その共通性質を議論していく。

そこでまず、本章の後半では、これまで行なわれてきた設計研究を概観して、設計研究の方法について議論する。多様な側面をもつ設計の研究においては、研究方法自身に注意を払う必要がある。本研究ではここでの議論をもとに、実験的な方法から情報処理的方法までを含む多面的なアプローチを提案し、そのアプローチに基づいて、研究を行う。

以下に本論文の構成を示す。

第2章「設計実験方法論」では実験的方法による設計研究について述べる。ここでは主にプロトコル解析という心理学的手法を用いる設計実験について検討する。まず2.1節では実験の意味について検討した上で、実験の有効性・データの信頼性について心理学実験の立場と設計研究の立場から考察を加える。そして2.2節では実際に行なった実験の結果を示す。2.3節では他の実験的設計研究について概観する。

第3章「認知的設計過程モデル」では第2章で得られた設計実験の結果から認知的なモデルを導く。3.1節では設計における問題解決過程として設計サイクルというものを示す。3.2節では設計において用いられる知識について分析を行ない、設計者のもつ設計知識のモデルを示す。さらに3.3節では設計者の思考の変化をコネクションニスト的アプローチを用いて分析を行なう。

第4章「論理による定式化」では、設計の定式化の方法を議論する。4.1節では第3章での試みを参考にしながら、設計における論理性について考察を行なう。4.2節では論理を基盤とした定式化の指針とその方法について述べる。

第5章「論理による設計過程モデル」では論理的な枠組みをもちいた設計過程モデ

ルについて述べる。5.1 節では論理推論過程とみたときの設計過程の特徴について述べる。5.2 節では用いる論理的枠組みについて述べる。5.3 節ではこの論理的枠組みを用いた設計過程の計算可能モデルについて述べる。

第6章「設計過程表現意味論」では、第5章で述べた設計過程のモデルを論理で表現する際の意味論について述べる。第5章で議論したモデルは不完全性と動的な変化という点で古典的な意味論では表現できない。そこで 6.1 節では部分意味論を、6.2 節ではデータ意味論というものを導入し、6.3 節でこれらの意味論に立脚したとき設計対象や要求仕様、さらに知識がどう表現されるかについて述べる。

第7章「設計シミュレーション」では計算機によって設計過程を作り出す設計シミュレーションという方法について述べる。7.1 節では第5章で提案したモデルに基づく設計シミュレータの概要について述べる。7.2 節ではこの設計シミュレータを用いた例を示し、結果を考察する。さらに 7.3 節では推論システムとして設計シミュレータを評価する。

第8章「インテリジェント CAD の仕様」では、これまで行なってきた議論をもとに、将来の設計支援のあり方を述べる。8.1 節ではインテリジェント CAD に対する基本的な要求をまとめる。その上で、8.2 節ではインテリジェント CAD システム全体の構成を、8.3 節ではそのなかの設計記述言語の仕様について述べる。8.4 節ではここで述べたようなインテリジェント CAD の実現の可能性について述べる。

第9章「結言」では本研究のまとめと将来への展望について述べる。

1.3 設計の諸相と設計研究の方法論

工学はものをつくることに関する学問であり、それゆえ設計は工学の中心的課題のひとつである。各分野あるいは各対象ごとに莫大な設計に関する知識や経験が積み重なっている。そして、各分野ごとに設計法や設計手法は存在している。しかし、それらの設計方法論は対象や分野の特殊性を強調することにより成り立っている。すなわち、分野や対象の数だけ設計法が必要である。もし、このように設計が分野や対象に依存してのみ存在できないとするならば、日々変化の度合を増す社会において、設計は困難な問題となり、産業の発展の阻害要因にもなりかねない。必要なのは、設計の特殊性だけでなく、設計の一般性である。

ここに設計学の存在意義がある。すなわち、設計の一般的方法論を研究するのが設計学である。設計には多くの要因が絡んでいるので、いろいろな立場からの研究が可能である。実際、いろいろなアプローチから研究がなされてきた。例えば、吉川 [Yoshikawa81c] や冨山 [Tomiyama85a] は、主にその歴史的背景からの分類として、心理学・創造工学、体験主義・経験主義、管理工学、設計工学、古典的設計論、哲学的手法などを挙げている。そこで、第一に設計に関わる要因・要素に注目することで、これまでの行なわれてきた研究を概観するとともに、本研究が範囲とする部分を規定する。次にこれらの研究が何に貢献するかという点について検討して、本研究の目的について述べる。

設計にかかわる要因の主たるものを図 1.1 に示す。これはものを作り出すときの情報の流れに注目した図である。ここにある要因は、設計者 (designer)、設計 (designing)、製造 (manufacturing)、製造者 (manufacturer)、生産環境 (production environment)、使用者 (user)、欲求 (desire)、要求 (requirement)、人工物 (artifact) などである。この他、さらに挙げることは容易であるが、以下の議論ではここに示すもののみを用いる。

ものを作り出すということは、人間の欲求を人工物として実現することである。それはその初期において個人的な行為であった。すなわち、使用者も生産者も同一である。やがて、使用者と生産者が分離され、企業といった生産システム (生産環境) が独立に存在するようになる。また、生産がその物理世界での作業である製造と人間の世

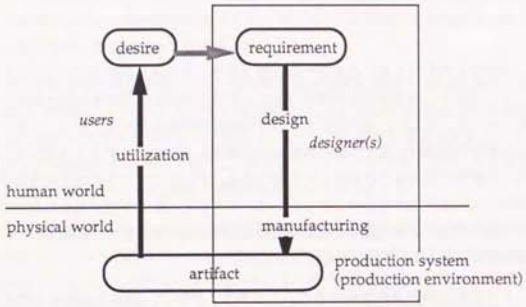


Figure 1.1: 設計に関わる要因

界での作業である設計に分離され、設計者というものが存在するようになる[1]。その結果、図1.1の要素が確立された。この図の何を中心にどの範囲までを対象にするかということで設計研究を位置づけることができる。

物理世界に注目した場合、設計対象が物理世界でどう存在しうるかについての研究である。それはいわば「人工物の科学(science of artifact)」としての設計研究である。無論、人工物を対象にしているので、社会的拘束(生産可能性)や思考過程に依存する抽象化なども考慮する必要がある。ドイツで行なわれてきた古典的設計論では古くからこの点に注目していた。例えば、Hubka[Hubka88][Hubka87][Hubka77]は対象を"technical system"として一般化することにより設計を体系化しようとしている。また、Roth[Roth82]の"design catalog"も設計対象の世界を体系化であるととらえることができる。また、定性推論を利用した研究(例えばFaltings[Faltings87]や車谷[Kurumatani90])は新しいアプローチである。また、制約を利用した対象表現というものも行なわれている(例えば[Arbab90][Suzuki90])。

[1]ここでは生産(production)を設計と製造(manufacturing)を合わせたものとして使っている。

設計を社会の中で行なわれる生産活動の一部と認識するアプローチもある。これは図1.1でいえば、「生産システム・生産環境」に注目したアプローチである。古典的設計論における設計過程とは、現実の工業活動の中で行なわれている設計の体系化であり、その枠のなかで最善の方法とは何かを規定している。あるいは主に企業単位で行なわれている管理工学的手法による設計作業の分析やマニュアル化も同じアプローチである。これらは、よい設計をするにはどうすればよいかという傾向が強く現れているので規範的な方法論であるが、このアプローチが規範的になる必然性はない。現実の設計作業の客観的な分析は少ないが、長澤ら[Nagasawa89]は企業の調査を通じて、設計作業の分析を行なっている。また、コスト評価や標準化といった問題もここでは問題になる。

いわゆる設計の理論は「要求」から「人工物」までの過程(「設計過程」)を、なるべく「設計者」や「生産環境」という要因に依存せずに議論する。一般設計学[Yoshikawa79][Yoshikawa81b][Tomiyama85b]はまさにこの方法である。また、エキスパートシステムに代表される知識工学からの設計研究は同じく「設計者」や「生産環境」を外部変数として設計をとらえるという意味で同じアプローチに立つが、それを設計として一般化・体系化するのではなく、情報処理として体系化しようとしている。設計のためのエキスパートシステムは近年盛んに研究されている(例えば福田[Fukuda89]や中島ら[Nakashima89])。実際的な見地から今後も多く研究され、またその中から設計の体系化に貢献するものでてくると思われる。

認知的なアプローチは「設計者」を中心において設計を考察する。設計過程を思考過程ととらえ、その分析から設計を解明しようというものである。認知心理学や人工知能の分野からの興味はこのアプローチからであり、近年盛んに行なわれている[2]。また、創造工学も関係している。設計は、シンセシス(総合)や創造といった人間の行なう最も複雑な思考を含んでおり、思考過程として興味深い。この他、設計は多くはグループで設計することに着目して、設計を人間の社会的活動としてとらえる研究もある[Bucciarelli88]。ただし、これらのアプローチでは、物理世界や生産環境といった他の要因を考慮しづらいという問題がある。

このようにどの要因に注目するかにより、設計研究の立場をまず分けることがで

[2]その詳細は第2章で述べる。

きる。本研究では、認知的な視点と情報処理的視点から「設計者」を含んだ「設計過程」の解明を行なう。設計過程を設計者の存在を前提にして、複数の視点から形式化を行なうというのが、本研究の基本的な立場である。設計者を含む以上、認知的な考察が必須であるが、それだけでは不十分である。その考察を基礎にして、より形式化を押し進めることが必要である。このために論理的枠組みを利用して、情報処理としての形式化を行なう。したがって、この研究において「生産環境」や「人工物」自身については特には触れないことにする。

次に問題になるのは、研究の目的である。設計の研究の位置付けを考えると、次の3つを挙げることができる。すなわち、

1. 設計自体の究明
2. 計算機化、自動化、計算機支援
3. よりよい設計、設計の最適化、効率化

である。(1) は設計の科学 (design science) の研究であり、設計という人間の行為あるいは社会的現象を科学の対象として論じようとするものである。これは吉川が一般設計学の提唱にあたってまず前提としたものであり [Yoshikawa79]、またそもそも Rodenacker[Rodenacker70] などの設計論や、最近では Dixon[Dixon87] の基本的立場である。(2) は、計算機のハードウエア、ソフトウエアの発達に支えられ、近年急速に興味を増している。これは計算機で処理できることを前提に、研究を行なうというものである。その実現の仕方は多様であり、いわゆる自動設計システムからインテリジェント CAD [ten Hagen87][Yoshikawa89][Yoshikawa90] に代表される CAD の高度化まである。(3) は実際の設計を改善することが目的である。この立場のひとつに例えば Suh[Suh78][Suh90] の公理的設計論がある。この3つは相反するものというより、それぞれ異なる軸で表現されるようなものである。すなわち、どの研究もこれら3点をどの程度含むかにより、位置付けがなされる。

本研究では、基本的には設計過程の理論的究明を目的とする。しかし、そこで提案されるものは単に法則、原理で留まるのではなく、計算機化可能であるものを要求する。この意味では (1) と (2) の両方を目的としている。このような立場をとること

は、(1) および (2) の目的双方にとって有益である。設計の科学は現時点で確立されたものではなく、また人間という要因を内在する以上、一般の自然科学が必要としてきた検証を同様に厳密に行なうことは難しい。このとき、計算機によるシステム開発を通じて妥当性をみることはひとつの有効な方法である。また、計算機支援という目的に対しては、単に個々の設計過程を計算機上で実現するのではなく、より一般性のあるシステムを構築しようとするとき、その背景に理論的裏付けがあることが望まれる。ただし、本研究では (3) は考察の対象としない。

以上をまとめると、本研究では設計過程を認知的視点・情報処理的視点から分析して、その計算可能な形での形式化をすることが目的である。

この目的の実現のための研究の手順は次のようになる。Dixon[Dixon87] や Finger ら [Finger89] は、設計のモデル化をどう行なうかという立場から、認知的モデル (cognitive model)、記述的モデル (descriptive model)、規範的モデル (prescriptive mode)、計算可能モデル (computable model) に分けている。認知的モデルは設計者の行為・振舞いを認知的な意味で説明できるようなモデルであり、記述的モデルとは設計行為とはどういうものであるかについてを説明するモデルである。規範的モデルとは先の目的の (3) に関連するもので、よい設計をするためのどうしたらよいかを記述するモデルである。計算可能モデルは計算機で実行できるような手続き・操作として記述するモデルである。

本研究ではこのうち、認知的モデル、計算可能モデルに注目する。本研究では図1.2に示すよう手順で研究を行なう。すなわち、まず実際の設計過程を観察して、そこから認知的な設計過程のモデルの構築を行なう。このためには設計実験の方法を用いる。次にこのモデルを利用して、情報処理的立場から形式化を行ない、設計過程の計算可能モデルを提案する。このためには論理的枠組みを中心として、人工知能での成果を利用する。そして、このモデルに基づくシステムを構築することにより、モデルの妥当性を確かめると共に、インテリジェント CAD 構築の可能性について議論を行なう。

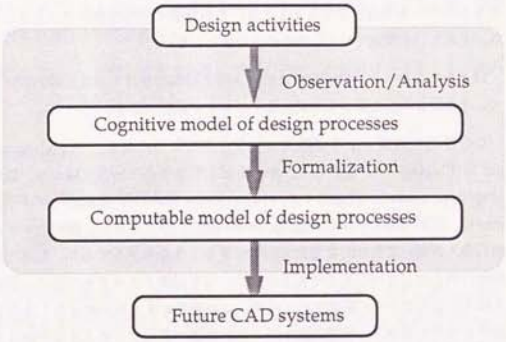


Figure 1.2: 研究の手順

第 2 章

設計実験方法論

本章では実験的設計研究法である設計実験について述べる。まず、設計実験の設計研究における役割を述べる。そのあと、これまで行なわれて来た設計実験を検討する。その上で設計実験の方法そのものに対する検討を行なう。ここでは心理学実験としての検討と設計研究としての検討を行なう。その上で、今回行なった設計実験について述べる。

2.1 設計実験に関する議論

2.1.1 設計の明示化の方法

設計に関する研究を行なうためには、実際にどのような設計が行なわれているかを知る必要がある。設計という行為は、現実には個々の設計者が日々行なっているものであり、それを明示化する必要がある。設計の明示化の方法は大きく分けて、以下の 4 方法があると思われる。

1. 個人的な経験や理解の体系化
   設計者が自らの設計経験や設計の理解を設計方法論として提示する。古典的設計論においてはその設計方法論は自らの設計体験をもとに体系化・一般化している場合が多い。
2. 設計の実例の記録
   設計の実例を記録したもの。企業の内部資料としては数多く存在すると思われるが、刊行されているものは少ない。例えば、[Kumagai76] や [Okamura85] [Okamura89] などがある。
3. 設計の調査
   企業などで実際に行なわれている設計をなんらかの方法で調査するもの。対象は個別の設計あるいはある分野の設計業務全般などである。例えば冨山 [Tomiyama85a] ではある特定の機械設計についての調査をしている。長澤 [Nagasawa89] では機械製図システムの標準化という観点から企業の設計業務を調査している。計算機の利用に注目した調査としては大須賀 [Ohsuga85] に航空機設計に関するものがある。
4. 設計実験
   設計を実験的方法により調べるもの。プロトコル解析などが用いられる。

吉川[Yoshikawa77][Yoshikawa83] や Dixon[Dixon87] が述べているように、設計の研究も単に経験的な方法ではなく、科学的方法に則って行なわねばならない。この場合、(1) の方法ではなく、(2) から (4) の方法に基づいて行なうべきである。

(2) から (4) の方法を比較すると、次のようなことがいえる。

(2) 設計の実例の記録は、

- 具体性が高い
  具体的な対象が常にあるので、極めて現実的な解析・分析が可能である。

といった利点があるもの、

- 一般化が難しい
  特定の分野の例であり、かつ個別例の集積であるので、一般化が難しい。
- 設計過程の情報の欠落
  一般に設計例の記録の場合、設計対象に関する情報は多いものの、それを設計する過程についての記録は少ない。

などの問題点がある。

また、(3) 設計の調査は

- 体系的である
  一般的傾向や特徴の抽出といったことが可能である。
- 大局的な情報が得られる

という特徴をもつが、

- 2 次的な情報である
  個々の設計そのものではなく、既になんらかの形でまとめられていたり、整理された情報であるので、2 次的な情報であるといえる。

といった問題点もある。

(4) の実験的方法は次のような特徴をもつといえる。

- 極めて詳細なデータが得られる
  一つの設計に関するできる限りの詳細なデータを得ることができる。
- 設計者の行為の分析が可能
  設計者の行為、思考を直接解析することができる。

その反面、

- 一般化が難しい
  設計実験では個別例の解析であるので、一般化や特徴の抽出には注意が必要である。
- 現実の設計との差が存在する
  設計実験はあくまで模擬的な設計であるので、必ず現実の設計とは差が存在する。

といった欠点が存在する。

すなわち、それぞれ得られる情報は異なるので、設計の総合的な研究の進展のためにはこれらの方法を適切に組み合わせて行なうべきである。

しかし、ここではこの中では (4) の設計実験に注目する。本研究では、1 章で述べたように設計者の行為という観点から設計過程に着目している。このためには設計者の挙動、行為を直接調べる実験的設計研究法が最適と思われるからである[1]。

2.1.2 設計実験の定義と位置付け

設計実験という名称は中島によってはじめに用いられた [Nakashima71]。その設計研究における位置付けは吉川[Yoshikawa77] に述べられている。吉川はまず、設計の研究法を二つの観点により、4 つに分類した。すなわち、分析的方法かモデル的方法か、あるいは動的構造を対象にするか静的構造を対象にするかという観点である (表 2.1参照)。ここにおいて設計実験とは、次のように示される。

[1] ただし、(3) の方法のひとつである retrospective reporting については 2.1.3 項で考察する。

Table 2.1: 設計研究方法の分類

	分析的方法	モデル的方法
動的	A1 設計実験 (動的分析法)	B1 論理的設計モデル (自動設計型モデル)
静的	A2 機能分析と実体分析 (black box 法)	B2 集合論的モデル (構造的モデル)

> 我々にとって可能な唯一の実験的方法は、実際の設計者による設計行為を調べることである。これを設計実験と呼ぶ。設計実験とは設計者に設計要求を与えたとき、設計者が設計解にいたる思考過程を何らかの方法で観察して、その思考過程を支配する法則性を発見しようとするものである。

本研究も基本的にはこの定義に従う。すなわち設計をなんらかの実験的手法に基づいて観察することを設計実験と総称することにする。この方法が本研究で重要な方法であるのは、この方法が思考過程を直接扱うという点にある。すなわち、他の方法では思考はあくまでその結果 (例えば設計対象物) を通じて間接的にしか扱えない。しかし、この方法に対していくつかの問題点を挙げることができる。その中で重要なものは、思考過程と設計の関係に関する疑問、思考の一般性に対する問題点、思考過程の観察可能性などである。

思考過程と設計の関係

設計は無論設計者の思考の結果であるが、同様なあるいはよりよい設計ができるならば、必ずしも設計者の思考方法に従う必要はないのではないか、と考えることが可能である。例えば構造解析などにおいては、人間とは異なる (人間では不可能な) 数値解析法を用いている。最適化の問題においても同様である。すなわち、この立場に立てば、設計においても人間とは全く異なる方法でもよりよい設計ができればそれでよいのであって、人間の設計過程を調べる必要はないといえる。

設計は本質的に人間の思考行為に依存するという本研究の立場からすると、そもそもその前提 (設計と思考の分離可能性) が問題である。しかし、設計システムの実現といった立場からは、一見有効な方法であるようにみえるが、実際にはシステム構築においてもこのような自動設計的考え方には問題点がある。最も重要な問題点は設計は常に変化発展するものであり、固定的な捉え方では表しきれないという点である。例えば現在、個々の分野では数多くの設計プログラムあるいは自動設計システムが開発されている。しかし、実際に効果があるのはもはや設計自体が変化しない定型的設計においてであり、それ以外ではつくってはみたものの、すぐに設計自身の変化についていけず、利用されなくなっている。このように考えると、設計を設計者の作業・過程と切り離すというやり方のみでは不十分である。

思考の一般性と観察可能性

また、そもそも個々の思考過程から一般的な法則性などを導くことが可能であるかという点も問題になりうる。本研究ではこれは可能であると仮定する。例えば、安西 [Anzai89] はプロトコル解析を利用して、問題解決のメカニズムを探っている。ただ、データから直接導出されるような形で求められることはありえない。というのはデータの整理をすべて客観的に行なうことは難しいからである。しかし、この意味では設計実験だけでは有効な設計研究法にはなりえない。設計実験はあくまで、設計研究に必要な情報を提供する手段である。他の分野でいえば観察方法、測定方法にあたるものといえる。得られたデータをどう用いるかは別の問題である。ここではデータ収集手段として設計実験を位置付ける。無論、できるだけ客観的に行なうのが望ましいので、以降では、整理や処理の方法をいかに客観的に行なうかについて検討を加える。

また、吉川も述べているようにこの方法の正当性、有効性は思考過程の観察可能性に負っている。この問題については、プロトコル解析の場合、プロトコルデータの有効性という形で、Ericsson ら [Ericsson80] が議論している。また、設計実験における設計がどれだけ一般に行なわれる設計に対応しているかという問題もある。これらについても以下で具体的に検討する。

2.1.3 設計実験の分類

設計実験は主に思考過程を調べる方法として心理学実験の手法を利用して実現される。しかし、心理学実験の対象と設計実験の対象ではその複雑さ、規模に大きな差があることに注意しなければならない。

実験的設計研究の方法としては主に 3 つのものが考えられる [Ullman87a]。

1. プロトコル解析 (protocol analysis)
   プロトコル解析とは被験者が考えていることを常に声に出しながら作業を行なう (thinking aloud) というものである。

2. Retrospective reporting
   作業が終った後、実験者が被験者に質問を行ない、その回答を記録する方法。この方法では被験者は何が起こったかを正確に報告するのではなく、自分が認識したことを報告する傾向が強い。

3. Informal reporting (immediate retrospective reporting)
   実験者は、被験者が作業を行っているところを観察して、必要に応じて、質問をするという方法。

この他に、連想実験といった方法も設計実験の一方法といえる。例えば、浅見 [Asami85] では、設計における知識構造の研究として、被験者 (設計経験者と学生) を対象にして、設計にかかわる単語 (部品の名前等) に対する連想を問うという実験を行なっている。また、図形と設計対象に関する連想実験も行なっている [Takeda90]。

Ullman らはさらにプロトコル解析を他の方式と比べて次のように述べている。

- 長所
  発話をすべて記録し、さらに図面やジェスチャーまでも記録するので、データの収集能力が優れている。

- 短所

  1. 被験者は考えていることを口に出しながら、設計を行なうので、設計の速度がおちる。
  2. 被験者が黙考しているときのデータはとれない。
  3. 一人の被験者が問題を解決する過程を研究するには向いているが、実際の設計は何人かがまとまって行なう場合が多い。

また、プロトコル解析と informal reporting を比較して、Waldron らは以下のように述べている [Waldron87]。

- プロトコル解析は量的な制約があり、大規模な設計の解析に使うには問題がある。
- プロトコル解析はグループによってなされたものを把握するのには向かない。
- Informal reporting は設計過程の構造を把握するのに向いている。

したがって、informal reporting はプロトコル解析を助けるものになりうると述べている。

Retrospective reporting においては、細かい情報は落され、記憶に残るような情報、すなわち重要な情報 (あるいは被験者がその時点で重要だと認識した情報) が得られる。プロトコル解析は集められる情報は最大であり、最も詳細なものであるが、情報が詳細過ぎると、かえって分析が難しくなることがある。例えば、発話された文章をそのまま記録にとっておいても、後でその意味がわからなくなる場合がある。そこで、それを補う方法として、retrospective reporting は有効であると思われる。

また、プロトコル解析には多大な労力がかかる。プロトコルの時間に対して、文字化 (transcription) にはその数倍の時間がかかるのが一般である。したがって、あまり長時間にわたるプロトコル解析は現実的ではない。

以上のように考えると、プロトコル解析は比較的短時間の設計の詳細な分析に、retrospective reporting と informal reporting は長期にわたる設計の概略を知るのに適切な方法であるといえる。

以下では主にプロトコル解析に基づく設計実験について議論する。他の方法は補助的に用いる。

2.1.4 プロトコル解析の手順

プロトコル解析の手順はほぼ以下のようにまとめられる。

1. 目的の決定
   過程のどの部分を分析するのかを決める。事前にモデルが存在しているときは、その検証が目的になる。

2. 実験環境の設定
   課題、被験者、被験者が利用できるものなど、実験の環境を設定する。

3. 実験の実行とデータの収集
   被験者に課題を声を出しながら解いてもらい (think aloud)、被験者の発話等をVTR などを用いて記録する。

4. データの文字化 (transcribing)
   音声記録の文字化、画像記録の図面化などを行ない、プロトコルデータとして整理する。

5. データの処理
   プロトコルデータを分節し (segmenting)、記号化する (encoding)。

実験によっては、3の次あるいは4の次に実験者が被験者に被験者が行なったことについての質問を行なう (retrospective reporting)。また、実験者は単に被験者の行為を観察するのではなく、必要時に被験者に質問をする実験もある。

これは心理学実験としてのプロトコル解析の方法であるが、これを設計実験として行なうためには、それぞれ検討が必要である。以下の項では設計過程の研究方法としてのプロトコル解析について検討する。

2.1.5 分析の対象

心理学実験での分析対象は極めて限定され、かつ分離された環境での単純な思考である。従って課題としても、Eight Queen Puzzle やハノイの塔といった単純な問題が多く用いられてきた。この場合、問題解決のかなり細かいレベルが分析対象になりうる。すなわち、あらかじめモデルを用意して、その範囲で解釈することが可能である。このため、前項で挙げた各段階を個別にかつ明確に行なうことができる。総じていえば、良構造問題 (well-structured problem) [Simon73] を扱っているといえる。

これに対して、設計実験では、すくなくても現時点では、心理学実験で求められるような詳細なレベルではなく、設計における問題解決の流れあるいは設計における問題解決の特徴といった、もっと基本的なことを知ることが必要である。

というのは、設計実験の対象となる設計過程は悪構造問題 (ill-structured problem) である。すなわち、良構造問題に対して、

- 規模が大きい。
  短いものでも数時間、長いものなら数か月、数年に渡るものもある。

- 複雑である。
  概念的なものから図形的もの数値的なものまで多種多様なものが複雑に絡み合っている。

- 曖昧である。
  すべてのものが明確に定義できるわけではない。

- open 問題である。
  多くのものから影響を受けており、問題を外部と切り離して考えることが難しい。

ということができる。すなわち、より基本的な問題解決のやり方といった点から分析をしなければならない。

このとき、心理学実験と同様なプロトコルの処理は難しい。これらは設計過程にとって付随的な特徴ではなく、本質的な特徴であるので、心理学実験に近付けるため

に設定をあまりに単純化するのは危険である。というのは、これらの特徴を明らかにするのも設計実験のひとつの目的であるからである。したがって、設計実験においては、心理学実験と同じ設定で行なう必要はなく、それにあった設定・方法で行なうべきであるといえる。

2.1.6 設定方法

前節で示したように、設計実験においては、単に心理学的要素を考慮するだけでは十分でなく、設計という対象に合わせた方法をとる必要がある。そこで、実験の設定方法を心理学実験的視点と設計研究的視点という二つの視点から検討する。

心理学実験的視点

プロトコル解析でえらえるデータは発話報告 (verbal report) である。この発話報告がどれだけ被験者の思考を正しく反映しているかという点については Ericsson ら [Ericsson85][Ericsson80] が議論している。ここではその議論をもとに、どのような設定で実験を行なえば良いかについて考察する。

Ericsson らは記憶構造のモデルから、発話過程 (verbalization process) を分類することにより、発話が記憶の正しい反映であるかを議論している。まず、記憶の構造としては、感覚貯蔵器 (sensory stores)、短期記憶 (short term memory, STM) と長期記憶 (long term memory, LTM) からなり、それが中央処理器 (central processor) によって処理されるというものを仮定している。その上で、発話を次の3つのレベルに分けている。

1. レベル1発話
   発話までに処理のない場合。現在注目されているものが発話される場合である。短期記憶にあるものが直接発話されるときはこのレベルである。

2. レベル2発話
   その時点で注目されている情報であるが、言語表現になっていないものを発話する場合。言語化過程のみが介在する。短期記憶にある非言語的なものが発話されるときはこのレベルである。

3. レベル3発話
   検索や選択過程が介在する場合や、新たに情報を作り出す過程が介在する場合。

そして、think aloud のように自分の考えを発話させること (発話と思考を同時に行なわせること) は発話がレベル1発話であれば、思考過程に全く影響を与えないし、レベル2発話である場合も、思考過程が遅くなるあるいは発話のほうが不完全になるが、思考過程はそのままである。しかし、レベル3発話では、思考過程に影響を与える。

また、発話されたものが不完全なものであることの原因としては、

1. そもそも注目されていなかった。すなわち短期記憶になかった場合、当然発話されない。例えば何度も繰り返された過程は自動的に行なわれるので、短期記憶に記憶されない。

2. すべての短期記憶の内容が発話されるわけではない。

3. 一度長期記憶にはいったのち、再び短期記憶に戻ることができなくなるものがある。

などが挙げられる。

以上のことを考えると、思考を阻害しない、かつなるべく完全な発話を得るための指針として、次のようなことを導くことができる。

1. 被験者に実験者が何を求めているかを知られてはいけない。

2. 被験者がやり慣れている課題を選ばない。

3. 被験者に自然に多く発話させる環境を作る。

(1) はレベル3発話を防止するためのものである。(2) は発話の不完全性の原因の (1) を防ぐためである。(3) は不完全性の原因の (2) と (3) を防ぐためである。

設計研究的視点

設計実験で行なわれる設計過程は、実際の設計過程の代替として行なうものである。したがって、実際の設計過程に近いことが望ましい。さらに、実際の設計と設計実験での設計の違いがどこにあるかを知ることは重要である。

Dixon[Dixon87] は設計に影響を与えるパラメータとして次のものを挙げている。

1. 設計者 (the person or persons)
2. 課題 (the problem)
3. 組織的な環境 (the organizational environment)
4. 設計環境 (the design environment)
   例えば、計算機ツール、解析・製図ツール、情報源など
5. 時間

これらは明確なパラメータというよりは影響を与えるものの分類に近いが、実験においてはこれらの要因がどう設定するかについて、常に意識をする必要がある。すなわち、これらの条件はプロトコル解析を行なう前に確かめておく必要のあるパラメータである。

データ収集方法

一般にはテープレコーダあるいは VTR を用いて行なう。設計におけるプロトコルはその発話の内容が多岐にわたるので、注意が必要である。心理学実験においてはあらかじめ用いられる単語が限定されている。例えば「ハノイの塔」のプロトコルにおいては、全体が 2000 語のプロトコルにおいて異なる 165 語しかふくまれていなかったという [Ericsson85]。これは問題空間が限定されているということによる。このような場合、音声記録から文字化は容易である。しかし設計の場合、先に述べたように open 問題であるので、問題空間が限定されていない。それゆえ、多種多様な単語が出現し、発話内容も多岐にわたる。このため、文字化が難しい場合が生ずる。このような点を補う方法を考えるべきである。

また、図の取り扱いが重要である。心理学実験では図に特に注意を払った研究は少ない (安西 [Anzai89] [Anzai90] は初等物理学の問題解決における図の役割を考察している)。しかし、設計においては、図が大きな意味を持つ。設計における図は、ポンチ絵のような漠然とした図から、部品図、組立図のような極めて正確な図まである。また、図の用い方も多様である。図の利用法としては、例えば次のようなものを挙げることができる。

- 思考の結果、内容の表現
  短期記憶、長期記憶の外在化
- 思考の道具
  幾何的な思考を行なう場として利用
- 他者への伝達
  コミュニケーションの場として利用

このような役割を果たす図をうまく収集する必要がある。

データ処理方法

プロトコルとして得られたデータを利用できる形にする段階である。プロトコルデータは人間の話した言葉であり、このままでは分析することができない。何らかの方法で、分節し (segmenting)、記号化する (encoding) 必要がある。

あらかじめ、プロトコルを解釈するモデルが存在する場合、そのモデルによってプロトコルデータから必要な情報を取り出すことは容易である。例えば、Ericsson ら [Ericsson85] は、次のようなスキーマを挙げている。

1. Intention.
   被験者のゴールなどについての表明。"shall" や "will" といった語で認識できる。

2. Condition.
   現在の状況のどの部分に注意を払っているかについての情報。

3. Planning.
   可能性を見い出すときの情報。"if" 文などで認識できる。

4. Evaluation.
   明示的あるいは暗黙的な比較。"no"、"yes"、"fine" などの語で認識できる。

あるいは Goor[Goor75] は、次のようなスキーマを挙げている。

1. Surveying given information
2. Generating new information of a hypothesis
3. Developing or working on a hypothesis
4. Unsuccessful solution
5. Self-reference or self-criticism
6. silence

これらをもちいると、プロトコルの発話の大部分がこの中に分類されるという。特に、用いられる語彙が限定されている場合、自動あるいは半自動的に記号化を行なうことができるという [Ericsson85]。

しかし、設計の場合、このような方法が行なえるかどうかは疑問であるし、また必ずしも同様に行なう必要はないと思われる。というのはまず、まだ設計においてはこのような一般に認められるモデルが確立していないということ、また 2.1.5 項で述べたように大規模で、複雑、かつ曖昧である設計問題に必ずしも同様の方法を適用する必要がないということが挙げられる。従って、実際の方法としては、

1. できるだけスキーマに基づく分析を行なうこと
   プロトコルを観察して、モデルを構築する。そしてそれに基づき、プロトコルを解析する。

2. スキーマに基づかない分析を行なうこと
   スキーマを利用するのではなく、別の方法で分析する。必ずしも形式的な方法だけではなく、いろいろな方法でプロトコルを利用する。

ということを、目的に応じて使い分けることが必要である。

2.1.7 まとめ

本節ではまず、設計実験をどう設計研究に用いるかという点について検討を行なった。そして次に設計実験の方法論について検討し、プロトコル解析を主にした設計実験は、その目的に合わせて実験方法を設定すれば、心理学的に適切でありかつ設計のデータとしても適切な情報をえることができることを示した。次節ではこれらの議論に基づいて本研究で行なった設計実験の方法を検討する。

2.2 設計実験の実際

本研究では、設計実験を 2 期計 9 回行なった。その内訳を表 2.2に示す。実験 II-A で retrospective reporting を併用している他は、一般のプロトコル解析である。課題の種類は 5 種類、プロトコルデータは総計約 40 時間である。これらの実験の設定方法とプロトコルの実際については、実験 I は [Takeda88][Segawa88]、実験 II は [Takeda89] [Ishihara89]、実験 III は [Takeda90][Hamada90][Matsuki90] を参照されたい。本節では、以下これらの実験の設定について前節の議論に基づいて検討する。データの処理方法は次章で議論する。

Table 2.2: 本研究で行なった設計実験

実験 No.	被験者	時間	課題
I-1	学生 2 名	約 3 時間	自動販売機の搬出部の設計 (1)
I-2	学生 2 名	約 4 時間	
I-3	学生 1 名, 技術者 1 名	約 3 時間	
II-A-1	学生 2 名	3:10	自動販売機の搬出部の設計 (2)
II-A-2	学生 2 名	2:10	
II-B-1	技術者 2 名	5:00	体重計の設計
II-B-2	技術者 2 名	5:20	
II-B-3	学生 2 名	3:35	
III-A-1	技術者 2 名	2:27	首振り機構の設計
III-A-2	技術者 2 名	2:15	
III-B-1	技術者 2 名	2:29	飛び上がるおもちゃの設計
III-B-2	技術者 2 名	2:16	

2.2.1 実験の設定の検討

前提となる条件

前節では設計研究にプロトコル解析をどう適用するかについて検討した。特に実験の設定においては、心理学的側面から導いた守るべき条件と、設計研究的側面から用意される留意すべき要因を挙げた。設計において注意するべき要因とは、

1. 設計者 (the person or persons)
2. 課題 (the problem)
3. 組織的な環境 (the organizational environment)
4. 設計環境 (the design environment)
   例えば、計算機ツール、解析・製図ツール、情報源など
5. 時間

である。また、心理学的側面から導いた条件とは、

1. 被験者に実験者が何を求めているかを知られてはいけない。
2. 被験者にやり慣れている課題を選ばない。
3. 被験者に自然に多く話させる環境を作る。

である。以下、設計の各要因に注目して個別に見ていく。

被験者

被験者は技術者および学生である。技術者は精密機械メーカに勤めており、通常は設計業務に従事している。設計経験は実験 II の被験者は 10 年前後、実験 III では約 6 〜 7 年であった。学生は主に精密機械工学科の大学院の学生であった。これらの被験者 (特に技術者) はこの実験の主旨は知らされていたものの、具体的に発話、行為

のどの部分が分析対象になるかまでは教えられてはいない。これは心理学実験の条件 (1)(被験者に実験者が何を求めているかを知られてはいけない) の条件を満たすためである。しかし、実験者の意図を読もうとする思考も時々みられた。実験者を意識させない、よりよい実験方法が必要と思われる。また、学生実験を行ったのは技術者との差異をみるためであり、これは設計の要因 (1) (設計者) の要因の影響をみるためであった。

課題

用いた課題を図 2.1、図 2.2、図 2.3、図 2.4、図 2.5、に示す。課題の選択においては以下の条件を設けた。

1. 機能が明確である。
2. 機構は未知である。
3. 解が実際に存在しうるものである。
4. 専門的な知識を必要としない。
5. 数時間で設計できるものである。

(1) と (3) は設計が収束するための条件である。すなわち設計が解にたどり着くために必要な条件である。収束しない設計の分析というのも興味深いが、ここでは対象としなかった。また、(2) は概念的な設計を含むための条件である。被験者にとって機構が既知なものであると、被験者は定型的な設計を行ないがちであると思われる。ここでは設計全体、概念設計から詳細設計の設計過程を対象にしたいので、望ましくない。また、心理学実験の条件 (2)(慣れている課題を選ばない) という条件にもそぐわない。

また、条件 (5) は実験の規模を限定している。この条件の理由の一つは実験自体が 1 日で終るためである。1 日以上の実験は長いインターバルをおくため、解析に問題が生じる。もうひとつの理由は解析の労力上、あまり長いプロトコルは困難であるからである。

ここで問題になるのは (4) の条件である。すなわち技術者にもその技術者の通常従事している範囲の課題ではなく、異なる範囲の簡単な課題を与えた。これは心理学実験の条件 (2)(慣れている課題を選ばない) を満たすためである。しかし、これは設計の要因 (2)(課題) を通常の設計と変えて行っているわけであり、注意が必要である。しかし、機械分野一般という意味では共通であるので、その意味での情報は得られたデータに含まれているといえる。また、実際的な課題を与えることは課題設定上困難であると同時にデータ収集の際にも多くの問題を生じると思われる。今後検討すべき問題である。

設計環境

被験者には 2 人一組にして課題を与え、対話によって設計を進めさせている。そして組になった 2 人は現在あるいは過去に一緒に仕事を行なったことのある人間で構成した。これは心理学実験の条件 (3) を満たすため、すなわちより多くの発話が自然にできると期待したからである。より多くの発話が取れるという意味では有効であったが、二人で行うのは不自然であると指摘する被験者もあり、設計環境という意味では検討すべき点であった。

被験者には実験 III を除き、紙と筆記具を使って設計を行わせた。実験 III では CAD と同様の描画システムを併用して、設計を行わせた。この場合、被験者の行動は被験者個人の CAD 経験の有無、CAD 利用分野などに強く作用されることがわかった。また、設計過程自身も描画システムの導入により変化していることが観察された。しかし、その反面、描画操作を限定したことにより、設計者の行う描画行為が明確になり、解析が容易になった。

2.2.2 データ収集方法

発話の規約化と補足

データの収集は基本的には VTR を用いて行なった。発話は VTR および補助的にテープレコーダで記録された。

タバコの自動販売機内の搬出機構部を設計せよ（図１－１参照）。搬出機構部とは、信号に応じて指定のタバコ箱１つを積み上げたコラムのから取り出す機能を果たすものである。簡単のため、タバコは１種類しかないとする。タバコ箱は以下に指定する。またモータ等の制御回路は設計しなくてよい。また収納部も設計せよ（例を図１－２に挙げる）。

できるだけ詳細に（図面まで）設計すること。利用可能なカタログ等は利用してよい。

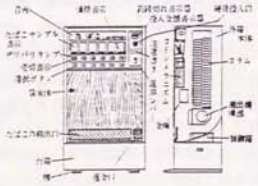


図１－１

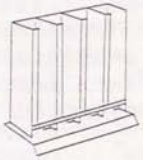
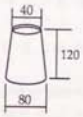


図１－２

Figure 2.1: 実験Ｉの課題

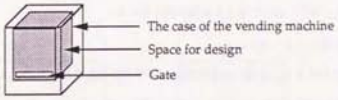
「今、左のような容器に入った飲料の自動販売機を設計したい。しかし、その前に機械系について試作をいくつかつくってみて検討することにした。そこで、その機械系に関する試作機を一つ設計して欲しい。」



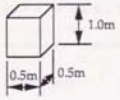
中身を含めた全重量：200g

注)

1. 機械系について設計すればよいのだから、冷却部については考えなくてもよく、収納部および搬出部についてのみ設計すればよい。但し、収納部というのは商品を収納しておく所であり、搬出部とは与えられた信号に応じて収納部から商品を一つ送り出すものである。



2. 設計する機械系は全体として下図の容積内に収まっていればよい。



3. 収納時、商品は横にしないこと。
4. 商品の補充に関しては、特に困難でなければかまわない。
5. 商品は一種類である。
6. モータ等はカタログ参照のこと。
7. 最終的には方眼紙を用いて主要寸法をいれて定規を用いて製図すること。

Figure 2.2: 実験 II–A の課題

以下の条件を満たす体重計の試作機を設計をしてください。

1. 下の範囲内に納まるように設計してください。
   縦300mm、横250mm、厚さ50mm
2. 足を置く場所は指定しても構いません。
3. 電気系はもちいてはいけません。
4. 構成部品の強度について厳密に考える必要はありません。
5. 100kg まで測れるものとします。
6. 100kg の荷重をかけると、体重計の蓋が5mm だけ下がるものとします。
7. 最終的に方眼紙に主要寸法を入れて定規を用いて製図してください。
8. 歯車等はカタログを参照して下さい。
9. 考えたことはできるだけすべて口に出して下さい。また二人で自由に討論しながら設計を進めてください。

Figure 2.3: 実験 II-B の課題

扇風機の首振りのための機構を設計して下さい。（図の支柱より上の部分）

周期：２０秒
首振り角度：９０度
モータ回転数：６００rpm

- 外形寸法は、下図の範囲に収まるものとします。支柱は下図の位置とします。機構とその配置、および機構部分を載せるシャシーの形状や支柱との接続方法などについては（支柱をどこまで伸ばすかも含めて）、特に制約はありません。ケースは設計する必要はありません。

- モータの寸法は、100×100×100[mm]で、内部に自由に配置して構いませんが、支柱と反対側の側面にモータの出力軸を出して下さい（下図参照）。
- １つのモータで羽根の回転と首振りの両方を行ないます。
- 主要な寸法が決まれば十分です。部品の強度などについては特に考える必要はありません。
- 重量についても、特に軽量である必要はありませんが、あまりに重そうなものは避けて下さい（支柱は金属性です）。

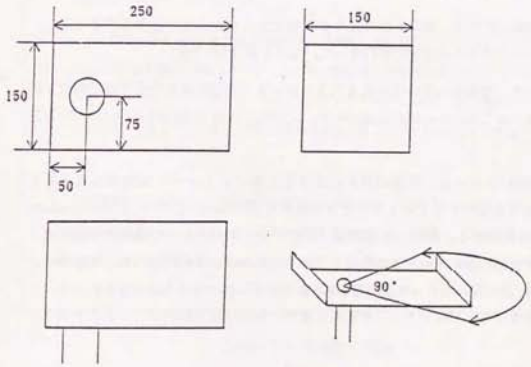


Figure 2.4: 実験 III-A の課題

垂直に飛び上がるオモチャの試作機を設計して下さい。

- 大きさは100mm×100mm×100mm程度に収まること。
- 動力源は単3電池1つ。
- スイッチを入れると、1分以内に飛ぶこと。
- 駆動系には手を触れにくい構造であること。

Figure 2.5: 実験III-Bの課題

2.1節で述べたように、設計のプロトコルでは多岐にわたる発話が行なわれる。このため、発話の内容を正しく文字化するのが困難になることがある。これを防ぐために、実験II-A以外では、記録者は実験に同席して、その過程を観察した。これにより、比較的不明な発話を減らすことができた。

また、実験IIIでは、音声をどのように文字化するについての規約を予め作成し、それに基づいてデータの文字化を行なった。それを図2.6に示す。

これにより、複数の人間が文字化を行なっていも、発話の単位や文字化の精度を同じに保つようにした。この規約に基づいて、文字化した例(実験III-B-2の一部)を図2.7に示す。

また、実験II-Aでは、実験の翌日に文字化したプロトコルを被験者に見せながら、不明な点を質問するということをおこなった[Ishihara89]。これはretrospective reportingに相当する。質問した箇所は、プロトコルデータの中で過程が不連続だと思える場所や長く考えていたところなどである。このような質問法では、被験者のより深い思考(表に現れなかった理由等)はあまりでてこないということがわかった。しかし、被験者が考えてはいたが、声にださなかった部分は比較的よくとることができた[2]。

[2]被験者にVTRを見せながら質問をするというやり方も提案されている[Yamamoto89]。

○設計実験プロトコル・データのテープ起こしに関する注意事項

1. 被験者の二人をA、Bとすると、Aの発言とBの発言は次のように表記するものと決める。
   [14:14:14(A)] カムよりフックの方がええんかいな？
   [14:14:30(B)] フックの方がええやろ。
   ここで、[]の中は[時:分:秒(発言者の略称)]を表し、半角で打ち込む。これを、プロトコル・インデックスと呼ぶ。
2. 各プロトコル・インデックスの後に打ち込む発言の最後には、必ず「。」「、」「...」「？」を下の指示に従って打ち込む。
3. 文の形で完結している場合は、文末に「。」を打つ。
4. 疑問を表す場合は「？」を付ける。
5. 一人の発言が始まって、それが文の形で終わる間に、明らかにブレイクされた時間(発言が全く無い時間)ある場合、次のように決める。短い場合(約5秒以内程度)はブレイクされているところを「...」で表し、一つのプロトコル・インデックスの中に入れる。長い場合は、ブレイクが始まったところを「...」で表し、それ以後の発言は、新しいプロトコル・インデックスの中に入れる。
6. Aの発言中にBが介入してきた場合、次のように決める。Bが介入してきたにもかかわらず、Aが続けて発言している場合は、AはBの影響を受けていないと見なし、Bの発言は無視する。Bが介入してきたことにより、Aが発言を中断し、Bの話を聞いている場合は、Aの発言の最後に「、」を打ち込み、Bのプロトコル・インデックスに移る。

Figure 2.6: 文字化の規約

13:06:17(K)	まずどういう形にするかやね。
13:06:19(M)	そうですね。
13:06:47(K)	まず駆動源何使おうか？
13:06:51(M)	モータ。
13:06:53(K)	まあ、モータやろけど。飛び上がるとき。
13:06:56(M)	板バネですね、バネですね。
13:07:01(K)	だからどうやってね、結局元へ戻すかということが問題やろね。
13:07:07(M)	普通パッと考えるのが、こうモータがグリグリ回ってって、バネがグルグル回ってって、あるとこまでいったらパチンと外れるとかね。
13:07:14(M)	で、バネが解放されてクシャンと飛ぶとかね。
13:07:20(K)	何でグーッと戻して、止めて、止めてと言うか、でパンと解放さす。ま、それには間違いない。
13:07:35(K)	例えば、ゼンマイとかね。
13:07:41(M)	ゼンマイって具体的にあまりよう知らんのです。
13:07:46(K)	まあ、巻きバネ。
13:07:53(M)	そっか、ゼンマイって巻いてって離したら逆に戻るだけなんですかね。
13:07:59(M)	ゼンマイ巻く軸に駆動軸がついてて。
13:08:05(K)	でも離したらけどモータ引きずるよね。
13:08:09(K)	クラッチ一個いれよっか、実際ミニチュアのソレノイドか何かいれてガチンと。
13:08:19(M)	そういうの得意。
13:08:22(K)	でもそうしたらスイッチがいる。
13:08:32(M)	巻いてスイッチ入れたら・・
13:08:58(K)	あーそっか、カム使えばいいんか？
13:09:07(M)	こう考えたら巻いてって巻いてってここにバネがあってあるとこまでいったら・・

Figure 2.7: 発話プロトコルの例

描画プロトコルの自動収集

2.1 節で述べたように、設計において図は重要な役割を持っている。したがって、図に関するデータを体系的に集める必要がある。実験 I と実験 II では、図は適当な時間間隔 (定時および大きく書き加えられた時点) でコピーをとった。また、VTR は常時、図および作図行為を記録している。これらのデータから、記録者が発話プロトコルにおいて参照されている図あるいは図の一部をみつけ、これを新たに書き直すあるいは元図に参照部分を示す印をつけるという形で、図のプロトコルを作成した。

しかし、これでは

- 図の分節が記録者の主観に依存する点が大きいこと。
- 書き直しを行なうと原図のすべての意味を保存できる保証がなくなること。
- 作画の過程を知ることが困難であること。

などの問題点があった。そこで、実験 III では描画システムを用い、描画プロトコルを自動的に収集した。描画システムは 2 次元の CAD システムに似たもので、直線、円、長方形などを書くコマンドを持っている。また、対話型実験のために、一つの図面を 2 人のユーザから操作できるようになっている (図 2.8)。作図はすべてこの上で行なうように設計者は求められた。また、被験者間の図面に関するコミュニケーションもこのシステムにあるポインタを用いて行なうよう求められた。被験者の作業は行なったコマンドの列として、時間とともに記録される。この記録を用いて、被験者の描画過程を再現することもできる。図 2.9を示す。

このようなシステムをもちいることで、

- より客観的で精度の高い情報をえることができる
- より詳細な情報を得ることができる
- 記録を自動的に行なうことができる
- 得られたデータは形式化されているので、その後の処理が機械的にできる

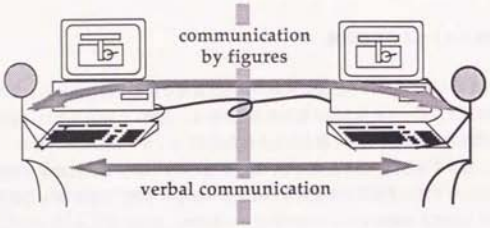


Figure 2.8: 描画システムを用いた設計実験

などの利点がある。その反面、以下のような問題がある。

- システムの利用により、設計過程そのものが変化する。
  例えば手書きで行なった実験 II–B–1 と II–B–2 では独立した図がそれぞれ 33 個と 26 個、紙の枚数で 13 枚と 12 枚であり、紙面で問題を検討していることがわかるが、描画ツールを用いた実験 III–A–1、III–A–2、III–B–1、III–B–2 では独立した図はそれぞれ、1 個、8 個、5 個、3 個と少数であり、既にきまったことを書き込むという形の図が多いことをうかがわせる[3]。すなわち、設計のやりかたが変わっている可能性がある。
- 計算機や CAD の利用経験により、作図の回数や頻度が変化する。
  例えば、操作数 (コマンドを用いた回数) で比較すると、CAD 経験のある被験者 (3 人) の平均が 168 であるのに対して、CAD 経験のない被験者 (1 人) は 96 で、2 倍近い差があった [Hamada89]。

これらは今後解決すべき問題である。

[3]図が独立であるかどうかを判断するのは一意ではないので、図の個数は厳密ではない。また、手書きでは消せないペンを使っており、単に書き直した図というものがあるし、描画ツールの図は細かい図も書けるので一つ一つの図が詳細になる傾向があるので、単純な比較はできない。あくまで参考である。

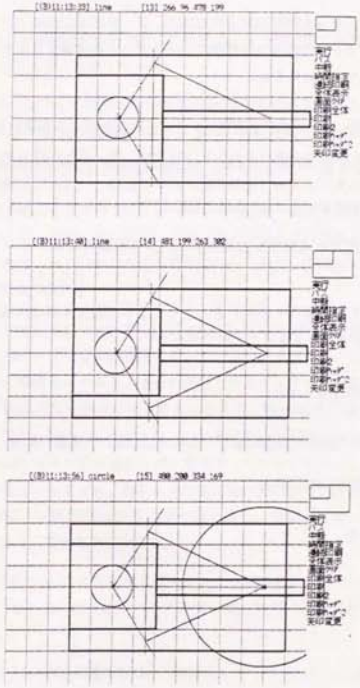


Figure 2.9: 描画プロトコルの例

2.2.3 まとめ

本節では今回行なった設計実験の設定方法、データ収集方法について検討を行なった。ここでは、より目的にあった信頼度の高いデータを収集するために、いくつかの点で改善を図った。まとめると、

- 課題の設定において条件を用意し、その条件に合致する課題を用意した。
- 設計者による違いをみるために、被験者に技術者と学生を用いた。
- 対話型の実験として、より自然に発話を行なわせた。
- 文字化の規約や補助的な質問により、より信頼度の高いプロトコルを作成した。
- 描画システムを開発・利用することで、より信頼度と精度が高く、かつ利用しやすい作図データをえることができた。

である。今後の問題としては、

- より複雑で大規模な課題の場合の実験方法の検討と実験の実施
- データ収集方法の自動化

などが挙げられる。

2.3 関連する実験的設計研究

ここでは主だった実験的設計研究を 2.1 節で示した分析対象、実験設定、分析結果などの各項目に注目して、検討していく。

吉川ら [Yoshikawa81a] の研究は実験的設計研究としては先駆的なものである。彼らは実験的設計法の重要性を強調するとともに、実験を行ない、結果をモデル化している。分析対象は設計過程一般、特に自由度の大きい設計を対象としている。実験の設定は総じて厳密ではない。課題は「全天候型衣服」などで、その意図が不明確である。また、被験者が学生のみである。ただし、対話型で実験を行なっている。データの利用法としては、対象の機能表現、属性表現の分類による思考の推移の分析を行なっている。この他、いくつかのモデル化を行なっており、形式的には未整理であるものの、それぞれ興味深い。

Ullman ら [Ullman87b] [Ullman87a] [Ullman88] [Ullman90] は、機械設計を対象とするプロトコル解析を技術者を被験者に用いて行ない、そのデータを分析している。実験の設定はよく考慮されており、十分に信頼できるプロトコルを収集していると思われる。実験の設定方法は参考にすべきものである。設計時間 10 時間と長く、また計 6 回の実験を行なっている。このプロトコルを元に、DEAM(Design Episode Accumulation Model) という設計過程モデルを提案している。ただし、分析の方向は、設計における思考というよりは、設計における行為の分析、モデル化である。また、図については設定・分析 [Ullman90] は不十分である。全体として、実験方法、分析方法とも明確であり、重要な研究である。

Waldron ら [Waldron87] は大規模なプロジェクト型の設計を retrospective reporting を用いて調べている。これはケーススタディに近いが、大規模な設計の分析の例として興味深い。

また、Goel ら [Goel89] は建築設計、機械設計、手引き (instruction) の設計の 3 種類の設計のプロトコルを分析してしている。異なる設計の比較して共通なモデルを導こうとする方法は興味深く、また議論の方法も緻密である。しかし、モデルとしては GPS 式の問題解決モデルの枠組みからはじめていることが、かえってモデルを複雑にしているように思える。

Adelson ら [Adelson85] はソフトウエア設計において、プロトコル解析を行ない、そのデータから設計者の振舞いなどを論じている。ただ、その分析法は informal であり、結果も定性的な分析である。さらに Adelson[Adelson89] はソフトウエア設計をプロトコル解析を用いて分析している。プログラムの原理の理解の過程を類推によって分類している。教授による理解過程の分析は興味深いが、この研究の分析は 1 例だけであり、またあまり客観的ではない。建築設計では例えば Chan[Chan90] がプロトコル解析に基づいた分析を行なっている。これも実験は 1 例だけである。ソフトウエア設計と建築設計の分野では、プロトコル解析を利用した研究は概して盛んであり、多くの研究がなされているが、領域依存の分析に終始しているものも多い。

Eckersley[Eckersley88] は心理学実験に近い設定、すなわち分離された単純な課題でプロトコル解析を行ない、詳細なデータを示している。しかし、このような設定での思考と設計と関係についての議論が必要であろう。Habraken ら [Habraken88] は数人が同時に参加して、一つの目標を達成する設計を模したゲームを用意して、そのゲームの過程を分析している。むしろこの方法の方が、極端に簡単な課題を用いた実験より、設計研究にとって可能性があると思われる。

以上のように、近年プロトコル解析を用いた設計の研究は盛んに行なわれている。設計のプロトコル解析は、設計の多様性から考えると、さらに多くの研究がなされ、データの蓄積と分析が行なわれることが必要である。しかし、実験の方法やデータの収集法などにはより注意が払われるべきである。

2.4 まとめ

本章では、設計の明示化の一方法としての設計実験の方法と実際について述べた。ここで述べたことをまとめると、以下の通りである。

- 実験的設計研究は、設計研究のなかで一つの重要な方法であること、およびそれが可能であるということを述べた。
- 実験方法を検討して、プロトコル解析を設計に適用する場合の問題点について議論した。
- 実験方法の改善法を示すとともに、実験を行なった。

第 3 章

認知的設計過程モデル

本章ではプロトコル解析に基づく認知的な設計過程モデルについて述べる。最初は問題解決(problem solving)的な視点から観察して、設計における問題解決過程について述べる。次に、設計過程の中で用いられている知識に注目して、知識のモデルを議論する。さらに、設計過程に現れる概念に注目して、概念間のネットワークをつくることにより、設計者の知識構造と設計過程の関係を調べる。

3.1 設計の流れと設計サイクル

設計は人間による問題解決のひとつとしてみることができる。2.1節で述べたように、設計は良構造問題(well-structured problem)でないので、GPS(General Problem Solver)[Newell72]のような狭い意味での問題解決のアプローチでは不十分である。すなわち、大規模、複雑、曖昧である設計における問題解決がどのようなものであるかを知る必要がある。

そこで、まず設計過程の大まかな流れを知るために、設計対象の変化と設計者の視点という二つのものに着目して、設計対象の変化を図式化した。その例を図3.1に示す。これはプロトコルデータから設計対象に関する発話を抜きだし、各時点での対象の記述としてまとめることによってつくったものである。ここで黒で示した矢印はある時点の設計対象を元にさらに設計を進めたことを示す。また、灰色で示した矢印は設計者の視点の移り変わりを示している。ここから次のようなことがわかる。

- 設計者は複数の解を同時並行的に考えており、適宜一つの候補の詳細化を進めたり、注目する候補を切り替えたりする。
- 解決不可能な(あるいは困難な)問題に直面した場合，その解を中断して，同様に中断していた他の解を始める。

このように設計対象は順次詳細化されていく。すなわち、この個々の詳細化がそれぞれ一つの問題解決であるとみることができる。すなわち、この黒矢印のあたる部分は一つの小規模の問題解決を示しているといえる。そこで、ある時点の対象から次に移るまでの間の発話を調べると、ある一定のパターンを抽出することができた。すなわち、以下のような5つの段階が観察することができた。

問題提起 解決すべき問題の発見や指摘をする段階。

提案 問題提起に対して、解決の鍵になる概念を提案する段階。

展開 設計者が有している知識を用いて、提案されたものを具体化する段階。この途上で問題点が見つかると、これが新たな問題提起となる。

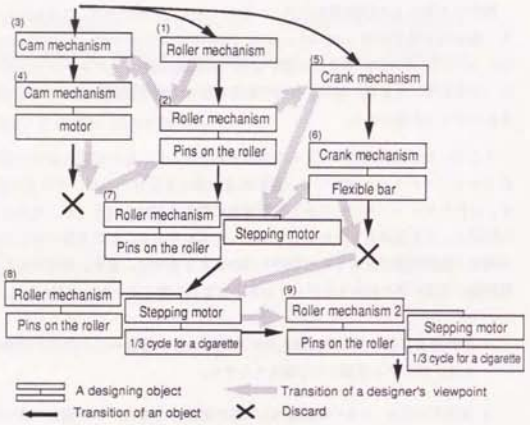


Figure 3.1: 設計対象の変化と設計者の視点

評価 展開された解をある評価基準で、評価する段階。評価で問題が生じれば、また新たな問題提起となる。

決定 評価を元にその解を採用するか、あるいはどの解をとるかを決定する段階。

この 5 つの段階が繰り返されて、ひとつの設計過程ができている (図 3.2参照)。そこでこれを以下、設計サイクルと呼ぶことにする。ただし、順に繰り返されるだけでなく「展開」や「評価」の段階から再帰的に繰り返されることもある (図 3.3参照)。ただし、実際には各段階の明確な区別は難しい。例えば、「展開」と「評価」を厳密に分けることは困難である。

Figure 3.2: 設計サイクル

この設計サイクルによるプロトコルの解釈の例を図 3.4に示す。また、この 5 つの段階による発話の大まかな分類を表 3.1に示す。ただし重複して分類される発話があるので、総発話数と各分類の総和は一致しない。

設計サイクルは基本的には、広い意味での問題解決の枠組みで捉えることができ、2.1 節で述べた Ericsson[Ericsson80] や Goor[Goor75] あるいは三宅 [Miyake85] が示している問題解決と類似性がある。

また、「評価」段階に関しては実際にはさらに二つに分けることができる。すなわち、「展開」された対象がもとの問題に合うかどうかをみる「確認型評価」と、評価

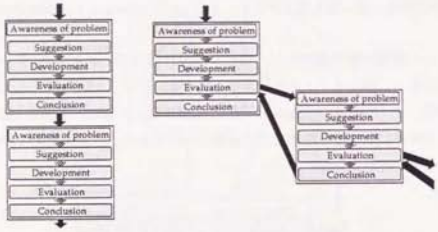


Figure 3.3: 設計サイクルの連鎖

Table 3.1: 発話の分類

実験	問題提起	提案	展開	評価	決定	未分類	総発話数
I-1	23	46	23	71	14	12	153
I-2	26	71	33	84	5	14	187
I-3	28	53	78	49	6	15	154

- 押し出し機構をどうするか？（問題提起）
- カムを用いる。（提案）
- 制御はカムが一回転で止まるようにすればよい。（展開）
- カムがポンと戻った瞬間に止まる。（展開）
- カムでは衝撃がある。（評価）
- 寿命が短いのではないか。（展開・評価）
- この機構は止めよう。（決定）

(a) ひとつのサイクル

- どうやって二段のものを制御するか。（問題提起）
- ...
- 蓋がトグルになっているとよい（図 A）。（提案 1・展開）
- 問題はどうやってトグルを取り付けるかだ。（評価→下位の問題提起）
- ...
- 図 B のようにドアのようなものをつけてはどうか。（提案 2・展開）
- 片側だけで支えるのは心許ない。（評価）
- どういった力でタバコを保持するのか。（評価→下位の問題提起）
- モータの力を使えばよいのでは。（提案）
- モータは回しっぱなしにするのか。（評価）
- トグル式の蓋のほうがいいみたい。（決定）

Figure A

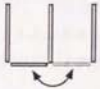


Figure B

(b) 複合的なサイクル

Figure 3.4: 設計サイクルの例

することで新たな問題を見い出そうとする「問題提起型評価」にさらに分けることができる [Segawa88]。例えば、図3.4(a) の文5は「カムを用いる」という提案に対してその挙動を評価している。これに対して、(b) の文8では、「ドアをつける」という提案に対して、単にその良否ではなく、その実現にあたっての問題点を指摘している。

設計の場合、思考と対象とのインタラクションは重要である。すなわち、対象の観察や分析とその操作が設計では重要である。ただし、ここでは設計途上であるので、現実のモノではなく仮想的なモノ、すなわち図面や思考上に存在するモノが対象である。しかし、設計における思考はそれだけではない。設計をどういう方向へ進めるかといった思考も重要である。すなわち、設計には少なくとも2つのレベルの思考が存在すると考えられる。その観点からみると、先の設計サイクルは2つに分離される。すなわち、対象を直接扱い、対象の観察や操作をする部分と、それ以外、すなわちその観察や操作といった行為をどう行なうかについて考えている部分である。前者を「対象レベル」、後者を「行為レベル」と呼ぶことにする。このとき、「問題提起」と「評価」の各段階は両者のレベルにまたがっているとみることができる (図3.5)。このとき、先に挙げた2つの評価は自然に解釈することができる。

すなわち、問題提起の段階では、対象レベルにおいて問題点の発見・認識を行なうと共に、実際に何を解決すべきかということを行為のレベルで決定している。また、評価では対象レベルで「確認型評価」が行なわれ、その上で行為のレベルで「問題提起型」の評価が行なわれる。

ここからわかるように、対象レベルの思考と行為のレベルの思考は独立してあるわけではなく、互いに関連してかつ混在している。

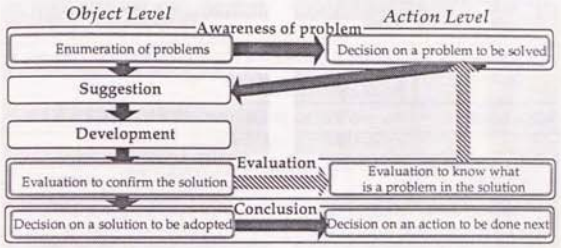


Figure 3.5: 設計サイクルと思考のレベル

3.2 設計知識のモデル

前節では、問題解決の立場から設計過程を分析した。そこでは設計もある範囲では問題解決の枠組みで捉えられることを示していた。問題が決まりさえすれば、ある程度探索空間を限定して思考を進めていると思われる。しかし、問題解決にどのような知識が用いられるのかといった点、あるいはそもそも問題をどう定義するかといった点については触れなかった。ここでは前者、すなわち設計における知識という点に注目する。分析の方法としては、プロトコルデータから知識を抜きだし、設計過程においてその知識がどういう役割を果たしているかを考察する。

3.2.1 設計知識の抽出と分類

ここでは、設計過程に現れる知識を

「設計過程において、設計者の思考に直接影響を与えている記憶情報」

と定義する。記憶のモデルでいえば、長期記憶 (Long Term Memory) の内容に対応するものである。この定義に基づいて実験II-Bのプロトコルから設計知識を抽出した。

その抽出の仕方の例を図 3.6に示す。

プロトコルデータ	「減速が足りないならそこにウォームを使うか。それなら緩まないので固定する必要もない。」
抽出した知識	ウォームを使うと緩まないので固定する必要がなくなくる。

Figure 3.6: プロトコルデータからの知識の抽出

このようにして、実験 II-B-1、II-B-2 から計 133 の知識を抽出した。表 3.2に抽出した知識の例を示す (分類のついては次項で述べる)。

ここで出てきたのはほとんどが対象の記述に直接関係あるものばかりであり、設計方針、ノウハウなどに関する知識はわずかしか (3 個) 抽出することができなかったので、以下の分析では対象に直接関係する知識のみを扱う。

こうして得られたものを観察すると、設計知識はある概念から別の概念を導く一種のルールとみることができる。設計者はその時点での設計対象 (あるいは設計仕様) に含まれている概念を観察して、そこに利用できる知識を適用させて新たな概念を得ていると考えることができる。

さらにこれらの知識で用いられている概念を観察すると、以下の 7 つに大別することができる。

(a) モノ 実体。現実にはその名前で示される。

(b) 機能 注目しているモノが他のモノとの関係において持つ役割。

(c) 属性 注目しているもの自体の、形状、寸法といった性質。

(d) 位相関係 モノとモノとの位相的な接続関係。

(e) 接続方法 モノとモノとの接続方法。

(f) 製作方法 モノの製作方法。

Table 3.2: 抽出した知識の例

知識の例	分類	出現場所
・プーリー径はどんなに細くても 10mm である。	[製造法→属性]	展開
・強度がないといけない所は板金でつくる。	[属性→モノ]	提案
・動滑車ではストロークが 2 倍になる。	[モノ→機能]	展開
・回転運動するものの動きを悪くするにはワッシャなどを使って締めてやればいい。	[機能→モノ]	提案
・部品数は減らした方がいい。	(設計方針)	提案
・くり抜くと強度が落ちる。	[モノ→属性]	評価
・最小プーリー曲率は動的特性である。	[属性→モノ]	展開
・ウォームを使うと緩まないので固定する必要がなくなる。	[機能→モノ]	提案
・ビスの頭を出さないために樹脂をアンダーカットしてひっかけることができる。	[位相→接続]	提案
・普通の体重計の目盛りは実は 360 度も回っていない。	[モノ→属性]	問題提起
・自動組立では上からしか作業ができない。	[製作法→属性]	提案

先に挙げた知識をこれらの概念間の関係とみると、以下のような 8 種類の関係に知識を分類することができる。

1. 機能→モノ 機能からそれを備えた実体(モノ)を導出する知識。
2. モノ→機能 モノからその表す実体の機能を導出する知識。
3. 属性→モノ 属性からそれを備えた実体のモノを導出する知識。
4. モノ→属性 モノからその表す実体の属性を導出する知識。
5. 属性→属性 属性から同じ実体の属性を導出する知識。
6. 位相関係→接続法 モノとモノとの位相的な接続関係からその接続方法を導出する知識。
7. モノ→製作法 モノからその表す実体の製作法を導出する知識。
8. 製造法→属性 製作方法からその属性を導出する知識。

表 3.2の第 2 列はそれぞれの知識がどの分類にあたるかを示している。また、このように分類した知識の関係を模式的に示したのが、図 3.7である。

これらのことから考えられる仮説は、ここで収集した知識は、機能や属性、モノの関係としてチャンク化してあったものが、設計過程の必要な形で発話に現れてきたものであるということである。

3.2.2 設計サイクルで用いられる知識

ここで抽出された知識が設計サイクルのどの段階で用いられているかを調べた。その結果を表 3.3に示す[1]。

この表から分かるように、抽出された知識の半分以上は「提案」段階で用いられている。これは「提案」の段階は、他の段階に比べて、相手に理解させるために、はっ

[1] ここで「決定」段階には知識の利用は見られなかったので省いた。

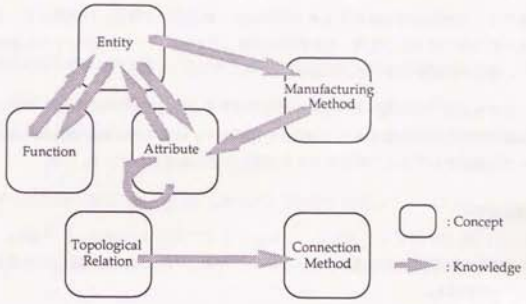


Figure 3.7: 概念間の関係としての知識

Table 3.3: 設計サイクルと知識の関係

知識分類	問題提起	提案	展開	評価	計
1. 機能→モノ		30			30
2. モノ→機能	5	1	1	1	8
3. 属性→モノ		1			1
4. モノ→属性	7	15		16	38
5. 属性→モノ		1	3		4
6. 位相→接続		24			24
7. モノ→製造法		1	2	6	9
8. 製造法→属性		5	2	10	17
計	12	79	8	33	131

きりとした発話が行なわれるためと思われる。例えば、「評価」の段階などでは、お互いに理解しているときは、単に評価の結果(「よい」、「よくない」)のみ発話されて、理由が発話されないことが多い。

次に各段階での知識の利用法について考察する。分類した8つの知識とがどのような場面でどのような働きをしているかを観察する。このとき、設計過程の各段階において知識はそれぞれ以下に示すような共通した使われ方をしている。

問題提起

I 与えられた設計仕様にさらに知っていることを付け加えることによって仕様の詳細化を図る。

知識2: モノ→機能 機能を付け加える。

知識4: モノ→属性 属性を付け加える。

提案

II 知識そのものが提案事項を提示する。

知識1: 機能→モノ 機能が分かっているものの、具体的なイメージがないところに適用される。

知識6: 位相→接続 部品間の組立法を導出する。

III 思考の根拠としての知識を利用する。

知識4: モノ→属性 属性によって提案事項を理由づけする時。

展開

IV 提案されたことが設計対象にどの様な影響を及ぼすかを明らかにする。

知識2: モノ→機能 提案事項がその対象の外部に及ぼす影響を明らかにする。

知識5: 属性→モノ 提案事項がその対象自身に及ぼす影響を明らかにする。

評価

V 提案事項を評価基準と照らし合わせる際に、提案事項の特性を知識から得る。

知識4: モノ→機能 部品などの属性が必要になった場合。

知識8: 製作法→属性 製作コストなどの工程レベルの情報が必要になった場合。

VI 検討の余地もなく一意に決ってしまうことを知識から得る。

知識7: モノ→製作法 提案するまでもなく、一意に決まる工作法(例えば、軸は旋盤でつくるといったもの。)を出して、知識8による評価の準備をする。

ここからわかるように、知識の利用法は極めて多様である。例えば「提案」と「評価」では2通りの利用法があるが、片方(IIとIV)はそこでの推論を進めるために直接使われているのに対して、IIIとVはそれを補助するために用いられている。また、段階により用いられる知識の種類に偏りがあり、また使い方が異なるということは、逆にいえば利用法に合わせて(みかけ上の)知識の形が決められていることを示しており、先に述べた仮説(知識のチャンク化と変形)を裏付けているといえる。

3.2.3 行為レベルの知識

設計を行なうときの推論として、対象レベルと行為レベルと二つがあることは既に述べた(3.1節参照)。

行為のレベルの知識とは、設計をどう進めるかについての知識であり、対象への依存性の度合から分類することができる。

- 一般性のある知識
  設計をどういう順番に行なえばよいかなどを一般的に記述する設計方法論や設計手順、設計時に問題点があればどうすれば良いかを述べる設計のノウハウ的知識は対象に依存しない。これらの知識は対象をなるべく含まずに述べられる。

- 対象に個別な知識
  この対象にはどの分野が関係しているか、といった知識は個々の対象に関して記述される。また、経験からくる、個々の設計で何に着目して設計するかに関する知識も含まれる。さらに、定型的設計のときの知識のように、対象が決まった時点でどの知識を利用して、何を求めればよいかがわかる場合の知識も対象に依存する知識である。

個々の知識の大きさ (granularity) でも区別すると表3.4となる。

Table 3.4: 行為の知識の分類

	知識の大きさ大	知識の大きさ小
一般的知識	設計方法論、設計手順	ノウハウ的知識
対象依存知識	定型的設計の知識	対象知識の管理知識

これまで述べた知識は対象レベルの知識であった。対象レベルの知識は被験者は比較的知識を明示的に用いることが多いので、知識を直接集めることが可能であった。しかし、次に何を行なうかといった行為レベルの知識は明示的に被験者の発話として観察されることは稀である。前述の定義では2例だけしか抽出されていない [Ishihara89]。そこで、設計を、現在設計中の対象を仮説と考え、それを知識 (対象に関する知識) をつかって要求仕様にあうように作っていく過程である (5章参照) とみなしたとき、設計者はそれらを用いて何を行なうかを調べたものを図3.8に挙げる。これは、対象に依存しない知識である。

1. 新たな知識が導入されると、その知識は関連する知識と結び付けられる。
2. 知識が追加されたり変更されたりすると、矛盾が起きないか確認する。
3. 矛盾が起きたら、矛盾解消の作業を行なう。
4. 知識が変更されたら、これまでの仮説から導かれた対象の性質をまた導けるか確かめる。
5. これまで導かれた性質が導かれなくなったら、それが確認できるような問題を提起する。
6. 全ての仮説に対して、それが問題となるかどうか、より詳細化、具体化すべきかどうか、判断する。
7. 問題となる仮説が存在すれば、その仮説を満足するような候補を挙げる。
8. 満足するような候補がない場合、後戻りする。
9. 新しい仮説が提案されたら、それを展開、評価する。
10. 候補の評価をしたとき、矛盾があれば解消する。

Figure 3.8: 行為の知識

3.3 概念ネットワークによる分析

前節では、知識の抽出を行なうことによって、設計における知識構造について考察した。その結果として、

- 知識はいくつかの基本概念の関係としてとらえられること。
- (みかけ上の) 知識の形は利用に仕方に依存していること。
- 知識は用いられる場面に応じて、利用の仕方がかわること。

などを明らかにした。しかし、明示的に発話に現れたもののみを知識としたために、抽出された知識の数は少なく (約 850 の発話に対して、134 の知識)、プロトコルの一部しか利用できなかった。また、知識としては表面的なものを扱っていた。

そこで、本節では設計における概念と概念間の関係を調べることで、設計における知識構造を分析する。まず、発話は設計者の持つ概念や概念間の関係が表現されたものであると仮定する。このとき、発話にある概念や概念間の関係を調べることは設計者のもつ概念構造を調べることになる。発話には多数の概念が含まれているので、プロトコルデータから概念と概念間の関係を抜きだして、概念のネットワークをつくることができる。得られた概念のネットワークの性質や設計の進行に伴う変化などを観察する。設計者はこの概念ネットワークの上で思考を行なっている。すなわち、設計における問題と問題空間はこのネットワークを元に定義されるといえるので、このネットワークの性質の分析により、問題や問題空間の定義を知ることができると思われる。

3.3.1 概念ネットワークの作成

まず、プロトコルデータに現れる概念とその関係を収集する。ここでは、「概念とは名詞あるいは動詞で表されるもの」とする。そして、各文に現れる概念をみつけ、それらの関係を推測して記述する。関係はここでは、基本的には動詞と 2 つの名詞

の 3 項関係とする。複雑な関係もここでは、3 項関係に分解して処理する[2]。また、基本的な関係を示すために isa(具体 - 抽象関係)、inst(具体例、指示語等で特定されるもの) など、いくつかの述語を用意した (表 3.5参照)。また、直接現れていないものの、個々の文を理解するのに必要な関係は背景的知識として、別に記述した。例を図 3.9に示す。

Table 3.5: 準備した基本的な関係の意味

関係	意味
isa	具体 - 抽象関係を示す
%isa	もののひとつの状態を示す
inst	具体例、指示語で示されるものを示す
%inst	具体例のある状態を示す
has	図における全体 - 部分関係を示す

プロトコルデータ	概念間の関係
そやけどバネ縮んだら径変わるでしょ、外径。	(縮む バネ) (持つ 縮んだバネ 縮んだバネの外径) (縮む 縮んだバネ) [背景的知識] (%isa 縮んだバネ バネ) (%isa 縮んだバネの外径 バネの外径) (isa バネの外径 外径) (持つ バネ バネの外径) (isa 外径 直径)

Figure 3.9: プロトコルデータの処理

こうして、求めた概念のネットワークの規模を表 3.6に示す。ここで図を含めた解析とは、図の部分部分を一概念として、図の部分同士の関係も概念間の関係として含めたものである。また、この解析で実際に用いることのできた発話は、実験 III-B-1 で発話数 243(22.3%、、全発話数 1090)、実験 III-B-2 では発話数 276(30.8%、全発

[2] この他にも副詞的意味を示す項や、主観的言明を示す項も用いて処理を行なっているが、以後の分析ではこれらの項は省いて分析を行なっている。

話数 895%) である。このようにしてできた概念のネットワークを用いて、設計過程中における概念の重要性を次のようにして調べた。

Table 3.6: 概念ネットワークの規模

	実験 III-B-1		実験 III-B-2	
	概念数	関係数	概念数	関係数
発話のみの解析	354	619	415	1024
発話の図の解析	417	954	467	1354

3.3.2 重要度の計算

半田ら [Handa88] と Hasida ら [Hasida87] は文章中における概念の重要性を調べる方法として、次のような方法を示した。まず概念の重要性を評価する基準として次のものを挙げた。

- 重要な概念に近接した概念はやはり重要である。
- 他の多くの概念と近接している概念ほど重要である。

このとき概念のネットワークにおいて、各ノードの活性度をコネクショニスト的アプローチによって伝搬させることにより、この2つの重要性の基準が満たすことができる。このとき、飽和状態に達したときの活性度はそのノードが示している概念の重要度と見なすことができる。

ここでいうコネクショニスト的アプローチ (例えば [Rumelhart86]) とは、各ノードに活性値を与え、リンクで近接するノードへ、その活性値からある計算方法で求められる値を伝搬させるというものである。

また、ノードとノードの結合の仕方が線形である場合 (入出力関数が線形である場合)、以下のような行列の計算で活性値を求めることができる。今、ノードが n あるとき、その (i,j) 成分がノード i とノード j の関係を示す $n \times n$ 行列 A、各成分がノードの時点 t での活性値を示す n 次元のベクトル $X(t)$、各成分がノードへの定常入力

を示す n 次元のベクトル C を用いて、

$$X(0) = C \tag{1}$$

$$X(t+1) = AX(t) + C \ (t = 1, 2, 3, ...) \tag{2}$$

と書くことができる。X が収束する場合、収束値 X_{inf} は、

$$X_{inf} = (I - A)^{-1}C \tag{3}$$

となる。しかし、実際には A の成分は 0 が多いので、(3) 式ではなく、(2) 式を繰り返し計算することで近似的に求めることができる。

この方法に従い、設計過程中における重要概念を計算する。ここでは、名詞に相当する概念をノードに、動詞に相当する概念をリンクと見なす[3]。リンク間の結び付きの強さは、関係の回数に比例した値を与える。そして、この概念ネットワークから行列 A の値を求め、(2) 式を繰り返し計算を行なうプログラムを作成して、計算を行なった。

計算は以下の設定で行なった。

設定 I 設計過程全体のデータを用いる。

設定 II isa 関係でつながっているノードを同一と見なして、設計過程全体のデータを用いる。これはより一般的なノードのみで計算をするためである。

設定 III 設計過程を一定間隔で分け、ここの区間で計算する。これは設計の進行による変化をみるためである。

3.3.3 設計過程全体に対する考察

設定 I で計算をして、活性値の高いノードを表 3.7 に示す。ここで、概念の末尾につけた記号はその概念の instance(具体例) であることを示す。実験 III-B-1 では「モータ」、「ラック&ピニオン」に関連する概念が上位に頻出する。また、実験 III-B-2

[3] これ以外に、全ての概念をノードとする設定でも計算を行なっている [Matsuki90]。

Table 3.7: 設定 I の結果

順位	実験 III-B-1		実験 III-B-2	
	概念	活性値	概念	活性値
1	おもちゃ 1	2.110	おもちゃ 1	2.107
2	マブチモータ 4	1.224	スイッチ	1.596
3	バネ	1.218	スイッチ・オン	1.296
4	ギア	1.214	上のスイッチ 21	1.237
5	回転数	1.184	スイッチ 37 オン	1.209
6	モータ 1	1.148	手	1.195
7	モータ	1.122	ギア	1.294
8	直径	1.112	モータ 1	1.845
9	初期状態のおもちゃ	1.112	ゼンマイ 3	1.175
10	フック&ピニオン	1.110	スイッチ 37	1.167
11	フック	1.109	バネ	1.155
12	圧縮バネ	1.107	スイッチオフ	1.149
13	ネジ 1	1.099	上のスイッチ 21 オン	1.146
14	支点	1.088	2 個のスイッチ	1.146
15	板	1.087	マイクロスイッチ	1.130
16	フック&ピニオン 4	1.079	2 個のスイッチ 28	1.121
17	カム 1	1.078	カム	1.113
18	カム 2	1.078	切り欠けギア	1.113
19	20 ストーク 2	1.078	レバー	1.104
20	ワンウェイクラッチ 1	1.078	下のスイッチ 20	1.098

では「スイッチ」に関連する概念が上位に多くある。そこで、設定 II(抽象 – 具体概念の同一化) で計算を行なうと、この傾向はより明確になる。設定 II での結果を表 3.8 に示す。

ここから分かるのは、実験 III-B-1 と実験 III-B-2 で上位 20 にある概念で共通なものとしては、「オモチャ」、「モータ」、「バネ」、「ギア」、「カム」、「支点」、「スイッチ」の 7 個があった。これらは、与えられた課題を解くのに必要な概念であったということが予想される。さらに観察すると、実験 III-B-1 では実験 III-B-2 より「モータ」、「ギア」が上位にあり、またそれらに関連すると思われる「スピード」、「シャフト」も同様である。実験 III-B-2 では実験 III-B-1 より「スイッチ」およびスイッチに関連するいくつかの概念が上位にある。図 3.10 と図 3.11 に 2 実験の設計の終了近くの図面を示す。実験 III-B-1 ではモータの回転数をギアにより減速する部分が詳細に検討されていることがわかる。また、実験 III-B-2 ではテコを用いたスイッチの部分が他の部分に比べて詳細に検討されていることがわかる。すなわち、ノードの活性値を重要度の度合とみるという仮定は妥当であると思われる。

Table 3.8: 設定 II の結果

順位	実験 III-B-1		実験 III-B-2	
	概念	活性値	概念	活性値
1	おもちゃ	2.240	レバー	3.198
2	ギア	2.008	スイッチ	2.761
3	モータ	1.932	おもちゃ	1.869
4	バネ	1.457	ギア	1.768
5	スピード	1.348	スイッチ・オン	1.712
6	ネジ	1.260	バネ	1.557
7	機械全体	1.243	支点	1.481
8	シャフト	1.220	スイッチ板	1.442
9	板	1.216	スイッチ・オフ	1.315
10	圧縮バネ	1.207	押しボタン	1.226
11	直径	1.203	カム	1.217
12	フック	1.181	手	1.198
13	フック&ピニオン	1.163	接点	1.187
14	支点	1.157	シリンダー	1.176
15	ウォーム&ホイール	1.152	モータ	1.154
16	カム	1.131	機構	1.153
17	初期状態のおもちゃ 1	1.114	切り欠けの部分	1.130
18	引っ張りバネ	1.104	押した何か	1.129
19	スイッチ	1.095	一番長い部分	1.111
20	ストローク	1.089	エネルギー	1.090

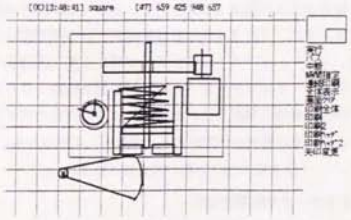


Figure 3.10: 実験 III-B-1 の図面

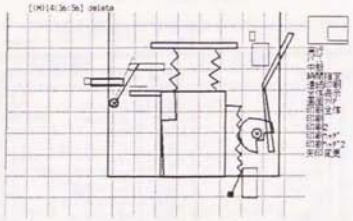


Figure 3.11: 実験 III-B-2 の図面

3.3.4 概念ネットワークの構造

次に主要な概念間の結び付きの強さを計算した。これは今注目する特定のノードのみに定常入力を与えたときの他のノードの活性値の収束値であり、そのノードからのネットワーク上での関連性の度合を示すものである[4]。この中から2実験で共通に上位にあった9個の概念間の結び付きの強さの度合を表3.12に、それを模式的に示したものを図3.13に示す。実験 III-B-1 では「ギア」、「モータ」、「バネ」、「スピード」という概念が互いに関連しており、これらを中心に設計が進められていたことをうかがわせる。また、「スイッチ」は「モータ」との関連性のみ認められる。実験 III-B-2 では、「ギア」と「スイッチ」が他の概念と多く関連性を持っている。この設計にあたっては、「スイッチ」と「カム」が重要な概念であったことを示している。このように同様な概念を利用していながら、2つの実験の間では、概念のネットワークの構造および、その利用の仕方が大きく異なることがわかる。

3.3.5 概念ネットワークの時間的変化

ここでは、設計過程を一定間隔で切り、その区間における各概念の活性値の収束値の変化を観察する。ここでは、30分間のデータを15分ずつずらして、計算してい

[4]計算方法の詳細については松木 [Matsuki00] を参照されたい。

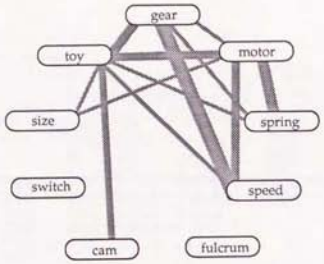
	オモチャ	ギア	モータ	バネ	スピード	支点	カム	スイッチ	サイズ
オモチャ		0.050	0.066	0.026	0.026	0.000	0.057	0.010	0.000
ギア			0.026	0.023	0.083	0.001	0.003	0.001	0.000
モータ				0.073	0.060	0.001	0.004	0.010	0.009
バネ					0.006	0.000	0.001	0.001	0.001
スピード						0.000	0.001	0.001	0.001
支点							0.000	0.000	0.000
カム								0.001	0.000
スイッチ									0.000
サイズ									

(a) 実験 III-B-1

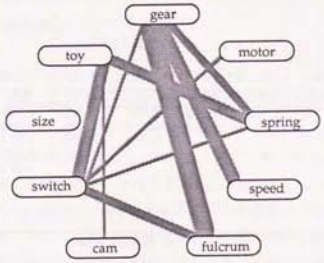
	オモチャ	ギア	モータ	バネ	スピード	支点	カム	スイッチ	サイズ
オモチャ		0.006	0.000	0.057	0.000	0.004	0.030	0.098	0.007
ギア			0.007	0.039	0.006	0.124	0.006	0.019	0.000
モータ				0.002	0.000	0.001	0.001	0.020	0.006
バネ					0.006	0.005	0.002	0.023	0.006
スピード						0.001	0.006	0.000	0.000
支点							0.001	0.046	0.001
カム								0.003	0.000
スイッチ									0.002
サイズ									

(a) 実験 III-B-2

Figure 3.12: 概念間の結び付きの強さ (1)



(a) Experiment III-B-1



(b) Experiment III-B-2

0.01 を越える関係のみ線で表示

Figure 3.13: 概念間の結び付きの強さ (2)

く[5]。主要な概念の活性値の変化を図 3.14に示す。ここでは、1 は定常入力として与えられるものなので、1 を越える分がその概念がどれだけ活性化されたかを示す。

各区間での活性値の収束値はそのときの重要な概念を示しているので、活性値の最大を追っていくと、設計者の注視点の変化を知ることができる。実験 III-B-1 では初期のうちは、「バネ」と「モータ」を中心にいろいろなものが考慮されているが、後期は「ギア」と「モータ」のみに集中していることがわかる。「スイッチ」は初期に現れるだけである。また、実験 III-B-2 では、中期において「スイッチ」に集中しており、その後「レバー」(ここでは、「スイッチ」と「ギア」を結ぶ部品) と「ギア」に移っていることがわかる。「スイッチ」の活性値が高い中期においては、「スイッチ」に関連する概念 (例えば、特定のスイッチやその状態を示す概念) も同様に高い活性値を持っており、この概念のネットワーク上で近傍にある概念を使って、設計を進めていることがわかる。次に「レバー」とその近傍の概念 (例えば「機構」といった概念やレバーの一部を指す図形) が高い活性値を持っている。

これを問題解決の立場から見た場合、基本的な問題領域はその時々に注目している概念のネットワーク上での近傍の概念であるといえる。しかし、全体のネットワークから部分的なネットワークを明確に切り出しているのではない。というのは、ある概念に注目したからといって、他の概念の活性化がまったくなくなるわけではないからである。設計を問題解決とみたとき、問題領域を明確に切り分けられるのでなく、常に連続した概念のネットワークの中で行なわれているといえる。

[5]ただし、背景的知識は常時利用することにする。

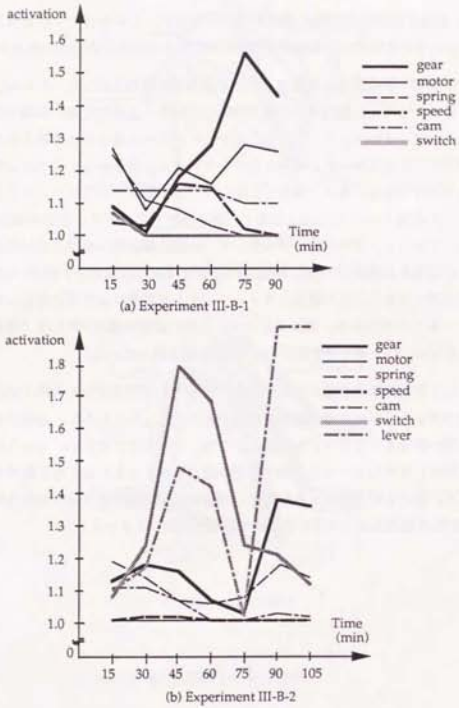


Figure 3.14: 活性値の時間変化

3.4 まとめ

本章では、認知的なアプローチとして、主に 3 つの方法で設計過程を分析した。すなわち、問題解決的アプローチから設計における問題解決の方法の分析と副問題の解決による設計過程全体の流れの分析を行なった。その次にそこで用いられている知識についてその利用方法を含めて考察した。さらにより基本的な問題である、設計者が思考を行なうときの基盤である概念のネットワークの性質を巨視的な観点から分析を行なった。

第 4 章

設計過程の論理による定式化

前章では認知的なアプローチにより、設計過程を分析してきた。プロトコルデータから、いくつかの方法を用いて、設計過程の特徴を明らかにした。本章以降では、設計の形式的な表現法ということについて議論を行なう。ここでいう設計過程の形式的表現とは、単に静的に記述が可能であるというだけでなく、推論などによって利用可能な表現である。

4.1 設計の論理性

本研究の目的は設計過程の形式化であり、これは言葉を変えれば、設計における合理性あるいは設計の論理の研究であるといえる。もちろん、ここでの「論理性」とは形式論理でのそれではなく、一般的な意味においての論理性である。設計には確かに、直感や思い付き、あるいは美的センスのような感覚によって行なわれている場面もある。しかし、そういった一種の神秘性に設計の本質を負わせるのは危険であるし、本研究の目的に反する。第 1 章でも触れたように、設計はその成り立ちから知識と強く関連している。設計は経験の知識化とその知識の利用による行為の生成によって成り立っている。ここでいう経験の知識化とは、経験を分解して、合理的な説明付けをすることである。設計のなかの合理性こそ、設計の本質であり、したがって設計における合理的な推論とはどういうものかを知ることが設計を知ることである。

本研究では形式論理を利用して、この設計の論理性について考察する。ここで形式論理の体系を利用するのは、形式論理は、人間の思考・知識の理論としては最も基本的なものであるからである。近年提唱されているその他の知識表現の理論の多くは、形式的にはこれに還元することができる (例えばフレーム理論については Hayes[Hayes79] 参照)。したがって、形式論理の枠組みを用いて設計過程がどれだけ表現できるか、あるいはそのために何が必要であるか、を考察することは、設計の論理性を知ることに重要な役割を果たすと思われる。

4.2 知識による問題解決としての設計過程

4.2.1 設計と問題解決

設計を問題解決とみた場合、以下のことが問題になる。設計は現象的には、要求仕様が与えられて、設計された対象(の記述)を作り出す行為である。そして、その行為には知識が関与している。したがって、要求仕様や設計解の記述といった対象の表現とそこで用いられる知識が問題になる。また、それらを用いる推論がどのようなものであるかも問題になる。そこで、設計を問題解決とみたときの特徴について前章までの結果を利用して明らかにすることにより、設計過程を論理で形式化するときの準備をする。

3 章では設計過程における問題解決と知識について分析を行なった。3.1 節では、設計サイクルという素過程の組合せとして設計過程を考察した。この設計サイクルでの推論は知識による問題解決とみることができる。設計サイクルでの問題提起で出された問題は目標であり、初期状態はこれまでに設計された対象であり、提案段階から展開段階・評価段階までは知識の探索と適用の過程である。それぞれの過程は、解を提案するのに必要な知識、解を展開・具体化するのに必要な知識、解を評価するのに必要な知識の探索と適用である。3.2 節では、これらの知識の一部を設計過程から抽出して、その特徴について調べた[1]。知識は対象の属性や機能に関する概念を組合せたものであり、設計サイクルの各段階での推論に用いていた。さらに 3.3 節では知識がいつなぜ用いられるかを知るために、設計過程での概念について調べ、概念の重要度という観点から設計過程の流れのなかでの概念について考察した。

例えば、図 4.1は、組み立て性を考慮して詳細化をするという問題提起に対して、組み立て可能な構造を提案し、展開している。これは、「図 A という対象」を「組み立て性を考慮して詳細化をする」という目標に対して、設計者が知識を利用してその解決を行なっているとみることができる。この例で必要とされる知識として例えば図 4.2のようなものが考えられる。すなわち、この過程はある対象(図 A)とこれらの知

[1]ただし、3.2 節でも述べたように、そこで集めた知識は設計者が明確に記憶内容の発話としているもののみを抽出したものであるので、そこで集めた知識だけでは推論はできない。ただ、発話には現れない部分でも、回想的報告で調べたように、設計者はそれなりの知識を利用して推論を行なっている。

○(問題提起 18) この状態(図 A)で組立性 {を考慮した設計} を考えよう。
...
○(サブ問題提起 18-1) まずワイヤリングをどうするか。
○(提案 18-1(1)) ワイヤーはどこから張っても構わない。
○(提案 18-1(2)) できたら固定している方(調整用プーリーの方)から張ってからバネの方に鉄板に切り起こしでもつくっておいて引っかける方が楽だ。ここ(図 A の b)も切り起こしだ。
○(評価 18-1) ここ(図 A の c)のワイヤリングが一番いやらしい。
○(サブ問題提起 18-2) あとバネをいかに固定させるかと、上の板の上下の動きを止めるものをどうするかだ。
...
○(サブ問題 18-3) プーリーにどうやってワイヤーを巻き付けるのか?
○(提案 18-4) 目盛板は一番最後に入れればよい {ので、問題ない}。
○ {(サブ問題提起 18-4) どうやって、目盛を入れるか。}
○(提案 18-4) それこそここにカップリングを切っておいてスプライン・(不明)を切っておいてはめ込むだけでいいようにしておけばいい。

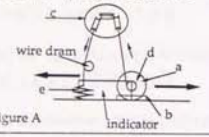


Figure 4.1: 設計過程の例

識を与えたときに、対象の詳細化と組み立て手順を推論する過程である。

このように、適当な知識を仮定すれば、設計サイクルでの推論は知識による推論とみることができる。設計サイクルを含めた全体の設計の流れは複雑であるし、そこで用いられる知識や概念は多岐にわたる上に利用の仕方にも特徴がある。

そこで、設計を知識による推論とみたときの特徴を以下で述べる。大きく分けると、推論の流れという点と、そこで用いられる知識や推論の目標である要求仕様、対象などの表現という点である。

○図 A での組み立て性には、ワイヤリングとバネの固定と上の板の上下の動きをとめることとワイヤーのプーリーへの巻き付け方...を考慮すれば良い。
○ワイヤーはどこから張っても構わない。
○固定している方向から張った方が楽だ。
○バネを鉄板に固定するには、鉄板に切り起こしをつければよい。
○滑車が連続するところにワイヤリングをするのは難しい。
○上から挿入する場合、下のものを先にいれなければならない。
○軸に上から円盤を差し込んで固定するには、カップリングとスプラインを切っておけばよい。

Figure 4.2: 知識の例

4.2.2 設計における推論の流れ

3.1 節では、設計サイクルのつながりとして設計対象の進化の流れを議論した。これを設計における推論の流れとしてみると、以下のような特徴がある。

段階的な変化である (stepwise) 設計者は要求仕様をもとに、設計対象を少しずつ部分的に設計していく。同じような作業の繰り返しにより、設計対象は段階的に変化する。

部分的な変化である (partial processing) 設計者は設計対象の全体を常に扱うのではなく、その部分を対象にして、設計を行なう。

分岐がある (branching) 設計者は場合によっては複数の解の候補を考える。それらを同時に詳細化したり、ひとつずつ検討したりする。

取り消しがある (retractable) 設計者は前に行なった設計を取り消すことがある。

また、各々の設計サイクルでは、提案段階、展開段階、評価段階ではそれぞれ異なる推論を行なっている。

多様な推論 推論は一意ではなく、いくつかの異なる推論の組合せでできている。

4.2.3 設計における知識

設計に用いられている知識は、極めて多岐にわたり、かつ莫大である。3.2 節で挙げた知識をみてもわかるように、多様な視点からの多様な分野の知識が用いられている。3.2 節で示した知識はあくまで、発話に明示されたものだけであり、実際にはそれ以上の多数の知識が用いられているものと思われる。3.3 節では設計で用いられる概念を収集したが、そこでも概念の種類は多数かつ多岐にわたり、それらの概念をもちいて表現される知識の多様さと膨大さをうかがうことができる。

設計過程では、これらの知識がすべて考慮されているわけではなく、推論ではその一部が選択的に用いられている。例えば、3.3 節でみたように設計過程の各時点では、特定の概念に注目して設計を行なっており、その概念に関連する知識が中心的な

役割を果たしていると思われる。多くの場合、用いられた知識は全て予め準備されていたわけではなく、必要なときになってはじめて、考慮されている。

また、設計における知識は互いに矛盾することがある。互いに矛盾する知識を用いた場合、実現不可能な対象を導き、矛盾が判明する。設計過程ではこのような矛盾が多く起こり、かつその矛盾を解決することで設計が進行している。その矛盾の解決とは、知識のどこかを変更する (例えば条件を増やす) などで行なわれている。この知識の非整合性と変更可能性は設計における知識のひとつの特徴である。

すなわち、設計での知識の特徴は挙げると以下のようになる。

部分性： 知識は部分的に用いられる。

非整合性： 知識は設計過程において互いに矛盾することがある。

変更可能性： 知識は設計過程において変更されうる。

4.2.4 要求仕様の表現

要求仕様もまたあいまい性を持つ。一般に設計では、外部から仕様が与えられるが、この仕様は全ての条件を書き尽くしているわけではない。実際には外部の仕様を解釈して、問題解決の目標として適した要求仕様を作っている。例えば、実験 [I]-B-2 では、被験者は設計過程の途中において、与えられた要求仕様に対して、図 4.3 に示すような事項を問題解決への目標としている[2]。すなわち、設計においては、要求仕様をつくることも設計の一部である。

問題解決の目標として要求仕様をみた場合、次のような特徴があるといえる。

不完全性： 仕様は設計開始時に全て決められているとは限らない。

変更可能性： 仕様は設計過程において変更されうる。

---

[2] これは被験者が設計過程中に列挙したものである。

1. 300mm×250mm×50mm 2. 足をおく場を指定してもよい。 3. 電気系は用いない。 4. 構成部品の強度は厳密でなくて良い。 5. 100kg まで測れる。 6. 100kg のとき 5mm 下がる。 7. 製図をすること。 8. 歯車はカタログを参照のこと。	⟹	1. 微調整 2. 精度 3. 傾いても壊れない 4. バネのバランスをどうするか 5. バネの設計 6. 摺動性 7. 乗るところが狭いので表示部と足の位置をどうするか 8. デザイン

Figure 4.3: 仕様の変化

4.2.5 設計対象の表現

3.1 節では設計対象の進化として表現したように、設計対象は順次詳細化され、最終的に設計解となる。すなわち、設計対象は設計過程において常に不完全である。問題解決の解として設計対象を捉えた場合、推論はこのような不完全な解を扱える必要がある。また、先に述べたように、設計解は唯一とは限らず、複数存在することもある。

さらに、設計対象の意味は一意ではなく、状況や利用法によって意味が変化する。特に図面を用いた場合、図面で示された対象の意味は解釈によって大きく異なる。例えば、図 4.1の図 A の各部品は位置を示すための記号であると同時に概略の形状も示しており、どういう意味かは利用法によっている。

したがって問題解決の解として設計対象をみた場合、以下のような特徴を持っている。

不完全性： 解は順次詳細化され、完全になる。

複数解の存在： 唯一の解があるとは限らない。

4.3 一階述語論理とその応用

設計過程の論理による形式化について議論する前に、まず本節では古典的論理とその中での推論について概観する。ここでは一階の述語論理 (first-order logic) を対象とする。一階述語論理理論 (first-order theory) の詳細は他の文献を参照されたい (例えば [Lloyd84]、[Nagao83])。ここでは述語論理の構文論 (syntax) および意味論 (semantics) を後の議論に関係する部分を中心に概略的に紹介する。

4.3.1 一階述語論理

論理式 (well-formed formula, wff) の構文は形式的に定義される。すなわち、変数 (variable)、定数 (constant)、関数 (function)、述語 (predicate)、結合子 (connective; $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$)、限定記号 (quantifier; $\forall, \exists$) などから定義される。また、これらから構成される論理式の集合を一階言語 (first-order language) という。

述語記号、関数記号、定数記号、変数の集合は互いに素な集合である。

定義 4.1 項 *(term)* は変数、定数記号、あるいは次の形をした関数のいずれかである。

$$f(t_1, \ldots, t_m)$$

ここで、f は m 変数の関数記号、$t_1, \ldots, t_m$ は項である。

定義 4.2 素論理式 *(atomic formula, atom)* とはつぎのような形をした表現である。

$$P(t_1, \ldots, t_m)$$

ここで、P は m 変数の述語記号、$t_1, \ldots, t_m$ は項である。

この素論理式は $t_1, \ldots, t_m$ と呼ばれる個体間の P と呼ばれる関係を主張していると解釈される。また、素論理式およびその否定はリテラル (literal) と呼ばれる。

定義 4.3 論理式は、次のいずれかである。

1. 素論理式
2. F と G が論理式のとき、$\neg F, F \wedge G, F \vee G, F \rightarrow G, F \leftrightarrow G$
3. F が論理式、x が変数のとき、$\forall x F, \exists x G$

さらに、論理式にある全ての変数が限定子によって束縛されるとき、その論理式は閉論理式という。

また、意味論は以下のように定義される。今、閉論理式のみを対象にする。

定義 4.4 一階言語の L の一つの解釈 *(interpretation)* は、次のものからなる。

1. 解釈領域 *(空でない集合 D)*
2. L 中の各定数に対する D への割り当て
3. L 中の各 n 引数関数に対する D^n から D への写像割り当て
4. L 中の各 n 引数述語に対する D^n から真理値 $\{t, f\}$ への写像

定義 4.5 I を一階言語 L の領域 D をもつ解釈とするとき、論理式は I に関する真理値 *(true value)* を次のように与える。

1. 素論理式 $p(t_1, \ldots, t_n)$ の $t_1, \ldots, t_n$ に I にしたがった D の要素を割り付けたときの値。
2. 論理式が $\neg F, F \wedge G, F \vee G, F \rightarrow G, F \leftrightarrow G$ の形のとき、その値は下記の表による。

F	G	$\neg F$	$F \wedge G$	$F \vee G$	$F \rightarrow G$	$F \leftrightarrow G$
t	t	f	t	t	t	t
t	f	f	f	t	f	f
f	t	t	f	t	t	f
f	f	t	f	f	t	t

3. 論理式が $\exists x F$ の形ならば、ある $d \in D$ が存在して、x を d で置き換えた式が t ならば、その式は t である。

4. 論理式が $\forall x F$ の形ならば、全ての $d \in D$ において、x を d で置き換えた式が t ならば、その式は t である。

定義 4.6 I を一階の言語 L の解釈とし、S を L の論理式の集合とするとき、I に関して全ての S の要素の真理値が真のとき、I は S のモデルである。

定義 4.7 S を一階言語 L の論理式集合とする。S のモデルとなる L の解釈が存在するとき、S は充足可能 *(satisfiable)* であるという。また、L の全ての解釈が S のモデルであるとき、S が恒真 *(valid)* であるという。また、S にモデルが存在しないとき、S は充足不可能 *(unsatisfiable)* であるという。

定義 4.8 S を一階言語 L の論理式集合とし、F を L の論理式とする。L の全ての解釈 I において、I が S のモデルであれば F のモデルであるとき、F の S の論理的帰結 *(logical consequence)* であるという。

4.3.2 一階述語論理における推論

節形式は論理の一つの記法である。節形式は次のように定義される。

定義 4.9 節 *(clause)* とは、次の形をした表現である。

$$B_1, \ldots, B_m \leftarrow A_1, \ldots, A_n$$

ここで、$B_1, \ldots, B_m, A_1, \ldots, A_n$ は素論理式 *(atomic fomula)* であり、$n \geq 0$ かつ $m \geq 0$ である。素論理式 $A_1, \ldots, A_n$ を積条件 *(joint condition)* といい、$B_1, \ldots, B_m$ を和結論 *(alternative conclusion)* という。節は論理式の記法であり、節に変数 $x_1, \ldots, x_k$ が現れるとき、上記の節は、

$$\forall x_1, \ldots, \forall x_k (\neg A_1 \vee \ldots \vee \neg A_n \vee B_1 \vee \ldots \vee B_m)$$

の同値である。

ただし、関数が含まれるときは、Skolem 化を行なうことによって節形式が得られる。この節形式の論理式における推論は導出原理 (resolution principle) に基づいて行なうことができる。導出原理をもちいた論駁では、求める結論を否定したものと定義された節(前提) が矛盾を示すことを、節と節を組み合わせて新しい節を作ることを続けることで実現する。

いま、二つの節 $c_1 = l \vee F$ と $c_2 = \neg l \vee G$ があるとき、これらからの論理的帰結として $c_3 = F \vee G$ を出すことができる。これが導出である。F と G がないとき、すなわち $c_1' = l$ と $c_2' = \neg l$ のときは、空節 (矛盾) を導く。もし、導出を続けていき、空節を導くことができた場合、元々の節集合は矛盾する。したがって、求めるべき結論は証明される。述語論理式であるときは、二つのリテラルのマッチング (この例では c_1 の中の l と c_2 の中の l のマッチング) の際、引数を調べ同一にすることが可能かどうかを判定しなければならない。これを単一化 (unification) という。

この導出による論駁はトップダウンにもボトムアップにも行なうことができる。トップダウンでは求めるべき結論を含む節を順次導出に用いていき、空節を求める方法であり、実際には節の後件部から前件部へ遡って単位節 (前件部のない節) にいたる方法である。ボトムアップでは適当な節を組み合わせて、最後に求めるべき結論の否定が導出されることを示すやり方で、これは単位節からはじめ、節の前件部から後件部を導いていく方法である。

結論部がひとつである節を Horn 節と呼ぶが、Horn 節に限ったトップダウン推論は Prolog[Lloyd84] によって実現されている。すなわち、Prolog では前提である定義節と求めるべき結論 (これをゴール節) を指定すると、ゴール節にあるリテラルからトップダウンに順次節を適用していき、空節を導くことができれば成功となり、そのときまでにゴール節の変数に代入されたものが返る。

4.4 論理過程としての設計過程

4.4.1 設計過程表現としての論理の問題点

問題解決としての論理に注目する場合、問題解決とは論理で表現された仮定 (前提) の集合を用いて、結論を導くことであると定義することができる [Kowalski79]。すなわち、論理式で表された仮定の集合を S、結論を G とおくと、

$$S \vdash G$$

の関係を調べる (証明する) のが問題解決である。ここで調べるというのは、結論を示すだけでもよいし、結論をもとめるのでもよい。以下、この仮定の集合を theory と呼ぶこともある。そして、その証明過程が問題解決の過程である。その過程は一意ではなく、トップダウン (結論からはじめる) にも、ボトムアップ (仮定からはじめる) にも行なうことができる。

述語論理を問題解決の形式化の枠組みとして用いるときの利点は、

1. 構文法が形式的に定義されている。
2. 構文法と意味が分離されている。
3. 記述に制約が少ない。
4. 推論が可能である。

などが挙げられる。すなわち、述語論理は構文法は実世界で何を示すかは関係なく定義され、かつその定義は形式的になされる。その形式は項、述語、関数、論理記号の組合せからなり、極めて自由な記述が可能である。また、この上で推論を定義することができる。しかし、論理そのものが推論を定義するのではなく、論理の定義を満たすような推論を定義できるという意味である。

しかし、前節でみてきたように、問題解決としての設計は、いくつもの特徴を持っている。このような形で論理を用いた場合、これらの設計の特徴の多くは表現することができない。その理由は、古典的な論理は以下のような特徴をもつからである。

1. 何を前提として何が求めるべき結論かという問題は範囲外である。
   前提や結論が与えられてはじめて推論がはじまる。ところが、設計においては要求仕様や対象のある範囲を決めて、その中で推論を行なっている。すなわち、前提や結論の設定を行なうところから推論をする必要がある。

2. 推論過程の明示的な取り扱いが必要である。
   部分的に推論を行なったり、分岐があるような推論を行なうためには、推論の過程自身も明示的に取り扱う必要がある。

3. 不完全な状態の記述や不完全な知識の利用ができない。
   設計において用いられる知識は不完全であり、また表わされる対象も設計過程においては不完全である。このような不完全な状態や知識が表現できること、さらにはそれらを用いて推論ができることが必要である。

4. 多様な推論が必要である。
   設計における推論は、いくつかの段階に分けられるように多様な問題解決の組み合わせで構成されている。このような多様な問題解決を実現する推論が必要である。

4.4.2 古典的論理の拡張と設計への利用

これらの問題に対して、一階論理を拡張してより問題解決能力あるいは問題記述能力を高めようという試みが行なわれている。

第1および第2の問題の一部は、高階の論理あるいはメタレベル推論で議論されている。メタ言語あるいはメタレベル推論を用いると、仮定の集合や結論、推論の制御などを扱うことができる。この場合、元々の論理の表現系は対象言語と呼ばれる。この両者は反射の原理 (reflection principle)[Weyhrauch80] により関係付けられる。すなわち、対象言語 L_1 、メタ言語 L_2 において、

$$S \vdash_{L_1} G \Longleftrightarrow \vdash_{L_2} demonstrate(S, G)$$

すなわち、対象言語 L_1 において S から G が導けるということはメタ言語 L_2 で素論理式 $demonstrate(S, G)$ が真であるということと同値であるということを示す。メタ言語での推論と対象言語での推論を組み合わせることで、知識の変化する場合の推論を扱うこともできる。

第3の問題と第2の問題の一部は論理の単調性という問題と関連する。古典的な論理はそもそも公理が動的に変化することを想定していない。また、現実の推論では前提 (公理) が増えるときそこから導かれるものが増えるとは限らない。非単調推論あるいは非単調論理では、このように前提が変化し、かつ非単調な推論を行なうことを扱う。これは知識や対象記述の不完全性と動的過程表現に関連する。

また、様相論理も非単調性の表現に関連しているし [McDermott82]、また、時間論理など変化を表す論理にも利用されている。さらに、第3の問題に関連する論理における部分性については部分意味論あるいは閉世界仮説において議論されている。

また、推論については論理における仮説推論 (hypothesis reasoning) や先に挙げた非単調推論、あるいは論理における事例ベース推論や類推などいろいろな推論が研究されている。

5章および6章では、これらの問題点を踏まえ、先に考察したような設計過程に表現するに適した古典的な論理の拡張について考察して、その論理を利用して設計過程を形式化する。

5章では設計過程での推論を議論する。ここでは、設計で行なれていた推論を論理的枠組みの中の推論として構成して、設計過程の動的モデルとして提案する。

6章では、設計過程と設計対象を表現する論理的枠組みについて議論する。そこでは、ここで挙げた不完全性や動的変化を表現する枠組みのなかで設計過程を形式的に表現する。

4.5 まとめ

本章では、設計を形式化する際の基本的な問題点を議論した。まず、知識による問題解決とみたときの設計の特徴を 3 章での結果を利用して議論した。さらに、本研究での形式化の方針である論理的枠組みによる形式化について説明を行ない、その長所、短所などを指摘した。そこで、古典的な枠組みでは設計の形式化には不十分であることを示すとともに、拡張の可能性を議論した。

第 5 章

論理による設計過程モデル

前章では、設計による論理による形式化の基本方針について述べた。論理的枠組みを用いたとき、いくつかの問題点があることが判明した。本章と次章ではこれらの問題点を解決する枠組みを提案する。本章ではまず、設計や設計過程を論理的枠組みの中で定義し、設計における推論を明らかにする。

5.1 基本的な枠組み

設計過程を推論として表現する場合、4.2 節で見てきたように、3 つの大きな要素がある。すなわち、(設計中の) 設計対象、要求仕様、設計知識である。以下の議論では、それぞれが論理式集合で表されると仮定して、順に Ds 、R 、K とおく。Ds には設計対象の記述が素論理式の集合として書かれている。同様に要求仕様を論理式で記述したときの素論理式の集合が R である。また、知識も論理式で記述され、その集合が K であるとする。ここでは知識は節形式で書かれるものとする。

設計は一般に要求仕様から設計解を導く過程であるといわれる (例えば [Yoshikawa79])。しかし、多くの設計では要求仕様が与えられると、すぐに設計の最終的な解がでてくるのではなく、設計過程の間で、設計対象は順次変化していき、詳細化が加えられて最終的な解になる。これを設計対象の進化 (evolution) と呼ぶ。これは一般設計学 [Yoshikawa81b] では解の収束として定式化されており、さらにメタモデルの進化として表現されている [Tomiyama85b]。また、3 章で具体的に見てきたように、設計過程においては設計対象は順次詳細化されるだけでなく、分岐して同時に詳細化されたり、あるいは修正を加えられることもある。4 章で述べたように、設計過程において変化するのは設計対象だけではなく、要求仕様や設計知識も変化する。すなわち、図 5.1のように対象、仕様、知識ともに変化しながら、設計は進行する。このような設計過程を表現することについて考察する。

まず、設計を論理的な枠組みの中で定義する。

ここで設計対象の記述とは、現在考えている対象を記述するのに必要かつ十分な情報とする。すなわち、その記述の内容のどれもそのときにある知識のもとで、他から導き出されることはなく、かつ冗長でないものとする。ここでは対象を記述する「本質的性質」(存在を規定する性質) といったものは考えない[1]。対象を記述する情報はその時にある知識に対して相対的に決められるものであるとする。なお、設計対象の必要十分な記述の意味とその存在についてはメタモデル理論で議論されている [Tomiyama89][Kiriyama91]。したがって、ここでは以下のように形式的に定義するに

[1]設計はものを作り出すに十分な情報を作り出す過程であるので、そのような「本質的な性質」を仮定すると、その途上の対象の記述ができなくなる (6 章参照)。

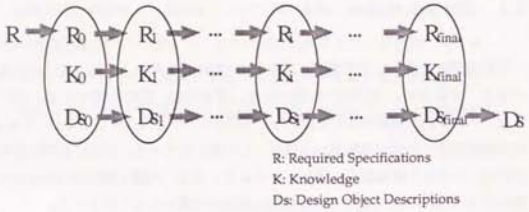


Figure 5.1: 対象、仕様、知識の変化

とどめる。

定義 5.1 設計対象記述 Ds とは、現在の知識 K において、現在考えている対象を記述するのに必要かつ十分な記述である。

ここでは、相対的に構造的 (属性的) か機能的 (性質的) であるかを指摘できる記述があると仮定する[2]。ただし、この判断はある基準を設定したときに可能になる。この基準は記述に用いる言語の要素の一部に (半) 順序関係をつけることによって作ることができる。論理を言語とするときには、述語の部分集合に対する (半) 順序関係である。

設計対象の進化、すなわち設計が進行するときに設計対象の記述の変化は任意にではなく、ある方向性がある。すなわち、その変化は、

1. 機能 (性質) から属性 (構造) を得る方向 (属性化)
2. より精緻な記述になる方向 (精緻化)
3. より具体的な記述になる方向 (具体化)

[2]機能と属性についての議論は例えば [Yoshikawa79]、あるいは [Umeda90] 参照。

の 3 つの方向のどれか、あるいはその組合せである。

一般にものの性質に関する一次的な知識は、(ある状況において) 個々のものがどのような性質をもつかについての知識である[3]。個々の性質をどのようなものがもつかについての知識は、一次的な知識から得られるといえる。このとき、論理的な記述において、ものの記述 (ものの構造や属性の記述) は性質や機能の記述を含意する。すなわち、それぞれの知識において、機能的記述の指し示す外延は属性的記述の指し示す外延を含んでいる (図 5.2(a) 参照)。

また、精緻化[4]されたものも当然、精緻化される前のものがもっていた性質等の記述を満たす必要がある。したがって、論理的記述と考えたとき、精緻化された対象の記述は精緻化される前の対象の記述を論理的に含意する。すなわち、精緻化される前の対象の記述の外延は、精緻化された対象の記述の外延を含む (図 5.2(b) 参照)。

具体化の方向も同様で、具体化された対象の記述は具体化される前の対象の記述を論理的に含意する (図 5.2(c) 参照)。

以上のように先に示した方向性は、論理的には、含意されるものを求める方向である。すなわち、外延においては、与えられた集合に対してその集合に含まれる集合を求めるということである。これは演繹とは逆の方向でありアブダクション (仮説形成、abduction) と呼ばれる [Uchida86][Yonemori81]。設計対象の進化は演繹ではなく、アブダクションによって行なわれる。

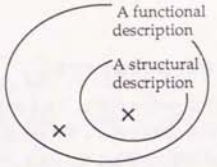
そこで、設計対象に関する知識を次のように定義する。

定義 5.2 設計対象に関する知識は、

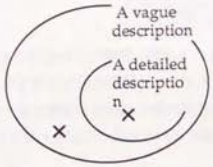
- 属性的記述が機能的な記述を含意する知識
- 精緻な記述が漠然とした記述を含意する知識

[3]これは多分に哲学的な問題である。林 [Hayashi89] は設計知識におけるこの問題について触れている。

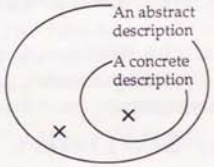
[4]ここでは、設計が進行することを詳細化、より細かく部分部分が決定することを精緻化ということにする。



(a) Functional - Structural



(b) Detailed - Vague



(c) Abstract - Concrete

Figure 5.2: 設計対象の変化と論理的含意

- 具体的な記述が抽象的な記述を含意する知識

の形で書かれる。

また、以下では対象に関する知識 Ko は無矛盾であると仮定する。このとき、設計の進行を次のように定義することができる。

定義 5.3 設計の進行とは、その時点での対象の記述に含意される新しい記述を求めることである。

しかし、無条件にアブダクションを行なうのではなく、設計は要求仕様を満たすことが必要である。そこで、次に要求仕様の問題について考える。

今、性質的記述とは、その対象に関して知りうること一般であるとする。すなわち、いわゆる性質的な記述から挙動の記述、さらには属性的記述まで含む。このとき、ある設計対象において、その対象記述は性質的記述を含意する。すなわち、

定義 5.4 ものに関する知識を Ko とするとき、設計対象記述 Ds_i の性質的記述 P_i とは、

$$P_i = \{p | Ds_i \cup Ko \vdash p\}$$

である。

定理 5.1 設計対象記述を Ds_i 、その性質的記述を P_i とするとき、

$$P_i \supseteq Ds_i$$

証明 定義5.4より自明。 □

設計の要求仕様は多くの場合は、対象の性質・挙動・機能で指定される。しかし、必ずしもそれだけではなく、寸法のように直接属性的な記述を含むこともある。これは先の定義における対象の性質的記述と同じである。すなわち、ある設計対象の性質的記述に要求対象が含まれれば、その設計対象は要求仕様を満たす。反対に、要求仕

様の項目のうち、一つでも性質的記述に含まれないものがあるとき、その対象は要求仕様を満たさない。

定義 5.5 対象記述 Ds_i で示される対象が要求仕様 R を満たすとは、その対象の性質的記述を P_i とするとき、

$$R \subseteq P_i$$

当然、要求仕様が満たされているときは、その要求仕様の全項目は対象記述から導出される。

定理 5.2 ある対象が要求仕様 R を満たすとき、その対象記述 Ds_i と対象に関する知識 Ko において、

$$Ds_i \cup Ko \vdash R$$

である。

証明 定義5.5と定義5.4より自明。 □

定義 5.6 対象記述 Ds_i で示される対象が要求仕様 R を満たさないとは、その性質的記述を P_i とするとき、

$$R \not\subseteq P_i$$

定理 5.3 ある対象が要求仕様 R を満たさないとは、その対象記述 Ds_i と対象に関する知識 Ko において、

$$Ds_i \cup Ko \nvdash r$$

なる $r \in R$ が存在するときである。

証明 定義5.6と定義5.4より自明。 □

である。

以上にように考えると、設計行為とは次のように定義することができる。

定義 5.7 設計とは、対象に関する知識 Ko のもとで、要求仕様 R が示されたとき、

$$Ds \cup Ko \vdash R$$

を満たす設計対象記述 Ds を求めることである。

また、設計の成功とは、求められた対象記述が形状等の設計解と認められる記述だけからなっており、かつものの記述として十分である (実現できる) ことが必要である。実現性というものを無定義で用いると、設計の成功とは次のように定義できる。

定義 5.8 設計が成功するとは、設計記述が設計解記述集合に含まれ、かつ実現可能であるときである。

ここで設計解記述集合とは、言語の部分集合として定義される。実現可能性は、試作等を行なうことによりはじめてわかることがある。

5.2 設計過程の定義

次に設計過程について考察する。先の設計の定義には設計過程は存在しない。しかし、実際の設計では、設計過程の中で、設計対象、要求仕様、知識が変化する。そこで、ある設計対象に関して設計を進めている設計の局面を設計状態と定義する。設計状態は考察の対象となっている設計対象に対して新たな設計が加えられたとき、次の設計状態になる。設計を進めたとき、いくつかの候補ができたときは、複数の設計状態を次に持つ。したがって、設計状態集合は木構造をなす。ある設計状態 t_i から設計が進んで設計状態 t_j になったとき、$t_i < t_j$ と定義すると、設計状態集合は $<$ に関して半順序集合となる。

先の設計の定義は各々の設計状態でも同様に定義される。

定義 5.9 設計状態 t_i において、対象に関する知識 Ko_i のもとで、要求仕様 R_i が示されたとき、その設計状態における設計とは、

$$Ds_i \cup Ko_i \vdash R_i$$

を満たす設計対象記述 Ds_i を求めることである。

先に述べたように設計過程で用いられる知識は変化可能であるが、その変化は単調であると仮定することにする。

定義 5.10 各設計状態で用いられる対象に関する知識は単調である。すなわち、設計状態 $t_i, t_j, (t_i < t_j)$ において、それぞれのときに用いられる知識 Ko_i, Ko_j は

$$Ko_i \subseteq Ko_j$$

を満たす。

また、ここでも用いられる知識は無矛盾であるとする。先に述べたように、設計が進行するとは、現在の対象記述を含意する新しい記述を得ることである。すなわち、設計が段階的に進んだときのそれぞれの時点での設計対象の記述は含意の関係にある。

定義 5.11 ある設計状態 t_i での対象記述 Ds_i とそれ以降のある設計状態 $t_j(t_i < t_j)$ での対象記述 Ds_j は、ものの性質に関する知識 Ko において、

$$Ds_j \cup Ko_j \vdash Ds_i$$

を満たさなければならない。

このように設計対象の進化を定義すると、次のことがいえる。

定理 5.4 ある設計状態での対象の性質的記述は、それ以降の設計状態でも性質的記述に含まれる。

証明 ある設計状態 t_i において、対象記述 Ds_i 、知識 Ko_i 、性質的記述 P_i は、

$$P_i = \{p \mid Ds_i \cup Ko_i \vdash p\} \tag{1}$$

の関係にある (定義 5.4)。また、同様に設計状態 t_i 以降の $t_j(t_i < t_j)$ における対象記述 Ds_i 、知識 Ko_i 、性質的記述 P_i は、

$$P_j = \{p \mid Ds_j \cup Ko_j \vdash p\} \tag{2}$$

の関係である。今、$p_0 \in P_i$ なる p_0 をとる。当然、式 (1), (2) より、

$$Ds_i \cup Ko_i \vdash p_0 \tag{3}$$

定義 5.11 より、

$$Ds_j \cup Ko_j \vdash Ds_i \tag{4}$$

を満たす。また、定義 5.10 より、

$$Ko_j \supseteq Ko_i \tag{5}$$

である。式 (3),(5) より、

$$Ds_i \cup Ko_j \vdash p_0 \tag{6}$$

式 (4),(6) は演繹定理より

$$Ko_j \vdash Ds_j \rightarrow Ds_i \tag{7}$$

$$Ko_j \vdash Ds_i \rightarrow p_0 \tag{8}$$

となるので、

$$Ko_j \vdash Ds_j \rightarrow p_0 \tag{9}$$

これは、

$$Ds_j \cup Ko_j \vdash p_0 \tag{10}$$

であり、したがって、これは式 (2) を満たすので、$p_0 \in P_j$。よって

$$P_j \supseteq P_i \tag{11}$$

が成り立つ。 □

定理 5.5 要求仕様が変化しなければ、ある時点で設計対象が仕様を満たせば、その設計状態以降の設計対象は常に要求仕様を満たす。

証明 ある設計状態 t_i での性質的記述を P_i、それ以降の設計状態 $t_j(t_j > t_i)$ での性質的記述を P_j とする。今、仕様を R すると、設計状態 t_i で要求仕様を満足しているとは、

$$R \subseteq P_i$$

である (定義 5.5)。ところで、定理 5.4 より、

$$P_i \subseteq P_j$$

である。したがって、

$$R \subseteq P_j$$

である。これは設計状態 t_j が要求仕様を満足していることを示している。 □

また次のこともいえる。

定理 5.6 用いるものに関する知識と要求仕様が変化しなければ、対象記述も変化しない。

証明 定義 5.1 の対象記述の必要十分性より、ある設計状態 t_i の対象記述 Ds_i、対象に関する知識 Ko_i において、

$$S \cup Ko_i \vdash Ds_i$$

を満たす S は $S = Ds_i$ である。今、t_i 以降の任意の設計状態 $t_j(t_j > t_i)$ をとると、仮定よりその状態での対象に関する知識は Ko_i であるので、

$$Ds_j \cup Ko_i \vdash Ds_i$$

したがって、

$$Ds_j = Ds_i$$

である。 □

すなわち、このように設計対象の進化を定義すると、ものに関する知識と要求仕様が常に一定であれば対象記述は変化しないことになる。逆にいうと、設計に用いられる知識や要求仕様が変化するために、設計が段階的に行なわれるといえる。

また、設計過程における要求仕様の変化も、対象の変化と同様に方向性がある。すなわち、属性化方向、精緻化方向、具体化方向の 3 つである。いま、この方向性をもつ要求仕様の変化を要求仕様の進化と呼ぶと、設計対象の進化と同様なことがいえ、要求対象の進化を次のように定義することができる。

定義 5.12 ある設計状態 t_i での要求仕様記述 R_i とそれ以降のある設計状態 $t_j(t_i < t_j)$ での要求記述 R_j は、ものの性質に関する知識 Ko_j において、

$$R_j \cup Ko_j \vdash R_i$$

を満たす。

5.3 設計過程における推論

前節では設計と設計過程を形式的に定義した。しかし、これらは設計や設計過程を宣言的に定義したもので、どうやって設計を進めるかについては触れなかった。ここでは、それらを実現する推論を用意して、手続き的な過程として、設計過程を表現する。すなわち、ここでの形式化の中での、設計対象の進化の方法、要求仕様の進化の方法、知識の変化の方法を示すことである。具体的には、3章で述べた設計サイクルを論理による形式化の中で解釈することで、設計における推論を明らかにする。

5.3.1 アブダクション・演繹の利用

まず、アブダクション推論、演繹推論を用いて、モデル化を行なう。アブダクション推論は仮説推論 (hypothesis inference) とみることができるので、仮説推論の方法を用いて実現することができる。ここでは、H、O を素論理式集合、K を節形式論理式集合として、

$$H \cup K \vdash O$$

なる H をもとめることを、

$$H = abduction(O, K)$$

と記する。ただし、具体的な推論については7章で議論する。また、逆に

$$S = \{s | H \cup K \vdash s\}$$

なる S を求める演繹的推論を

$$S = deduction(H, K)$$

と記述することにする。

設計サイクルは、問題提起、提案、展開、評価、決定という5つの副過程からなり、このサイクルが繰り返されて設計対象が順次詳細化される。すなわち、個々の設計サイクルは設計対象の進化をひとつ行なうことであり、ひとつの設計状態が対応する。すなわち、ある設計状態 t_j での設計サイクルは、その前の設計状態 t_i での設計対象記述 Ds_i、要求記述 R_i、対象に関する知識 Ko_i を利用して行なう。そして、5つの副過程をこれらを先の形式化の中で考察すると以下のようになる。

問題提起 この段階では、ここまでに設計した対象を観察して、その一部に注目してその部分を次の問題とする。注目するとは、注目点を決めること、その部分に関連する知識を準備することといえる。注目点を決めるとは、対象記述 Ds_i から注目点 $T(T \subset Ds_i)$ を設定することである。また、その部分に関連する知識をを持ってくるとは、新たな知識 ΔKo_j を付加して、$Ko_j = Ko_i \cup \Delta Ko_j$ とする。展開・評価段階で問題点が見つかったときは、観察して注目点を決めるまでの過程は既に行なわれている。

提案 この段階は、現在の知識をもとに問題提起段階で発見した注目点に関して設計を詳細化する段階である。これは注目点 T を導出する新たな対象記述の候補をアブダクションによって出す段階である。すなわち、

$$T' = abduction(T, Ko_j)$$

$$Ds_j = (Ds_i - T) \cup T'$$

なる Ds_j を新たな対象記述の候補とする。

展開 現在、提案した候補がどのような性質を持つか、どのような挙動をするかを調べる段階。対象記述から間接的に導き出される性質のうち、対象を考える上で重要な性質を導く。すなわち、

$$P_j = deduction(Ds_j, Ko_j)$$

なる P_j を求める段階である。この P_j と R_i を比較観察して、必要な性質が導かれていない場合は、その一部を新たな問題提起とする。また、要求仕様を進化させて R_j を決める。

評価 展開段階と同様に 提案した対象の性質や挙動を導く段階。すなわち、この段階では、対象の評価に必要な性質・挙動などを導く。ただし、評価にしか用いられない知識も用いられる。すなわち、評価用の知識を Ke とすると、

$$P'_j = deduction(Ds_j, Ke)$$

Figure 5.3: 演繹・アブダクションによる設計のモデル

なる P_j' を求める段階である。ここでも同様に P_j' と R_i を比較観察して、必要な性質が導かれていない場合は、その一部を新たな問題提起とする。また、要求仕様を進化させて R_j を決める。

となる。すなわち、基本的には、提案段階でアブダクションを行ない、展開・評価段階で演繹を行っている (図5.3参照)。

そして、

$$Ds_j \cup Ko_j \vdash R_j$$

になったとき、設計は終了する。

このモデル化は、設計過程の基本的な流れを説明するが、このモデル化の問題点としては次の点がある。

1. 問題提起段階での問題発見での推論
   すなわち、形式的には
   $$T = find\_problem(Ds_i)$$
   なる関数がどのようなものであるかは言及されていないということである。

2. 問題提起段階での注目点に関する知識の準備
   すなわち、形式的には
   $$\Delta Ko_j = find\_related\_knowledge(T)$$
   なる関数がどのようなものであるかが言及されていないということである。

3. 展開段階、評価段階での問題点の発見
   性質記述 P_j 、あるいは性質記述 P_j と要求仕様記述 R_i からどのようにして問題点を出すかという点が言及されていない。

4. 決定段階の推論

(4) の決定段階での推論については、最終的な設計者の判断であり、ここでの知識による推論の範囲外であると思われるので、ここでは議論しない。以下では残りの3つの問題を考察する。まず、(3) について考察する。

5.3.2 サーカムスクリプションによる矛盾の解消

展開段階、評価段階で行なっている推論のひとつは、導かれた対象の性質の記述を観察して、そこに問題点がないか、あればどうすればよいか、ということを調べる推論である。論理的形式化における明らかな問題とは、記述の無矛盾性の維持である。すなわち、Ko_i が矛盾する知識を含んでいるとき、得られた性質的記述 P が矛盾している可能性がある。

4章で示したように、設計に用いられている知識は整合的であるとは限らず、お互いに矛盾する知識が含まれることがある。特に異なる分野の知識を同時に用いるときなどに起こりうる。しかし、ここではその矛盾は知識に本質的な誤りがあるために起こったのではなく、知識の利用の仕方にあると考える。すなわち、知識を記述した時点で想定し得ない状況で知識を利用したために起こる。

知識は全ての状況を想定して記述することは不可能である[5]。そこでここでは、全ての知識はその知識が使われるべき状況が省かれて記述されているとする。すなわち、各知識は潜在的に例外事項を持っているが、表面上には記述されない。このとき、知識に記述されていないため、本来、用いてはならない状況で知識を適用することにより矛盾が発生する。このとき、矛盾を解消するとは、本来あるべき例外事項を見つけるという行為になる。この行為をサーカムスクリプション (極小限定、circumscription) によって行なう。

[5]起こりうる全ての状況を記述することは不可能である。これはフレーム問題と呼ばれ、いくつかの議論がなされている [McCarthy69][Matubara87]。

サーカムスクリプションとは McCarthy[McCarthy80] によって提案された非単調推論の一つである。サーカムスクリプションの直感的意味は、「そこに対象としていることしか考えない」という推論であり、指定された述語の外延を極小化することによって実現することができる。

述語 P を含む論理式集合 $A(P)$ における P のサーカムスクリプションは、

$$A(\Phi) \wedge \forall x(\Phi(x) \rightarrow P(x)) \rightarrow \forall x(P(x) \rightarrow \Phi(x))$$

を満たす Φ を求めることである。この意味は Φ は A に P の代わりに Φ を入れたものを満たし (左辺第 1 項)、その Φ は P の部分集合である (左辺第 2 項) ときには、P は Φ の部分集合である (右辺)、すなわち Φ と P は同値であるということである。

例えば今、ブロックの世界でブロック A, B, C があるということを示す、

$$isblock(A) \wedge isblock(B) \wedge isblock(C)$$

という式を A とするとき、述語 $isblock$ のサーカムスクリプション Φ は、

$$\Phi(x) \equiv (x = A \vee x = B \vee x = C)$$

である。これはサーカムスクリプションの定義を満たすことは明らかである。すなわちこの例では「ブロックである」ということは「A, B, C」のどれかであるということに限定されたわけである。サーカムスクリプションはいくつもの述語を同時あるいは優先順序をつけて行なうことができる [Lifschitz85]。先のサーカムスクリプションの定義は 2 階の式であり、一般には解くことが困難であるが、限定された範囲では計算可能であることが示されている[6]。

ここで、サーカムスクライブされる述語に例外を示す述語を使うことで、例外事項の計算にサーカムスクリプションを用いることができる。すなわち、

$$A_1 \wedge \ldots \wedge A_m \rightarrow B$$

という式があるとき、この式の例外を示す述語 ab を用いて、

$$A_1(x) \wedge \ldots \wedge A_m(x) \wedge \neg ab(x) \rightarrow B(x)$$

[6] 計算方法等については 7 章で触れる。

と書く。そして、この式を含む論理式集合 A で述語 ab をサーカムスクライブすると、求められた ab のサーカムスクリプションは A を満たす最小 (極小) な外延を持つ述語である。もし、この式の例外になるような項が A に含まれている他の論理式から導かれるときは、それが ab の外延である。もし、この式の例外になるような項が存在しないとき、ab は常に偽 (f) である。

例えば、

$$bird(x) \wedge \neg ab(x) \rightarrow fly(x)$$

$$ostrich(x) \rightarrow \neg fly(x)$$

$$ostrich(x) \rightarrow bird(x)$$

という例を考える。このとき、ab をサーカムスクライブすると、

$$ab(x) \equiv ostrich(x)$$

となる。元の式に戻すと、

$$bird(x) \wedge \neg ostrich(x) \rightarrow fly(x)$$

となり、例外を含めた式となる。

ここでは、矛盾を起こした場合、関係する知識を記述する論理式を集めて、それらの式全てに例外事項を記述する述語 (アブノーマル述語) の項を付加して、サーカムスクリプションを行なう[7]。例外事項が見つかれば、すなわちその式のアブノーマル述語が偽 (f) でない場合は、その例外事項を付加した式に書き直す。例外が見つからなければ、すなわちその式のアブノーマル述語が偽 (f) であれば、その式は変化がない。

サーカムスクリプションが終った時点で、知識は変更されている。以降はこの変更された知識を用いて、設計過程を進めていくことになる。すなわち、形式的には、

$$Ko' = circumscription(Ko)$$

となる。

[7] ただし複数の述語をサーカムスクライブするときは、サーカムスクライブする順序によって結果が変わってくる (7 章参照)。

このとき、定理5.6からわかるように、再びアブダクションを行ない、さらに演繹を行なう必要がある。すなわち、設計サイクルの最初、すなわち問題提起段階に返る必要がでてくる。このとき新たに例外が見つかった式が注目点を決めるとき重要である。

そのままの対象記述では、これまで導いていた性質的記述の一部が導けなくなることがある。その導けなくなった性質的記述の中に要求仕様が含まれていた場合、設計をやり直す必要がある。すなわち、変更後の知識を Ko_i' とすると、

$$Ds_i \cup Ko_i \vdash Rs \quad Rs \subset R$$

$$Ds_i \cup Ko_i' \vdash Rs' \quad Rs' \subset R$$

$$Rs' \neq Rs$$

となることがある。その原因は変更された式が用いられなくなったことによる[8]。このような場合、その式の結論部が新たな注目点となる。すなわち、この結論部は変更される前までは対象記述に含まれるか、そこから導出されるものである。したがって、結論部の論理式を新たな注目点として、推論を続けることができる。これが展開・評価段階から問題提起段階への移行である。

5.3.3 メタ推論の利用

問題点のうち、(2) と (3) は設計過程をどう進行させていくかに依存している。すなわち、これまで議論してきた設計対象とその性質に関する推論に対して、設計における行為に関する推論である。

この推論は、その時点で行なっている論理的推論の前提や結論を操作する行為である。これは論理的推論に対する操作であり、これを行なう推論はメタ推論となる。ここでは、これまでの論理的推論 (以下、対象レベル推論[9]) に対する操作を設計過程に

[8]逆の可能性はない。今、例外がある可能性があるとしてサーカムスクリプションの対象とした式はなんらかの性質を導くものであるので、必ず用いられた式である。

[9]ここでは「メタ」に対する「対象」、「行為」に対する「対象」という二つの使い方をしている。前者は推論系に関する議論のときに、後者は設計の形式化の議論のときに用いる。ただし、以下の議論では結果的に両者は同一のものを指している。

関する知識を用いたメタ推論として形式化する。ここでのメタ推論は、対象レベルの状態、すなわち設計の対象記述、性質、現在利用可能な対象に関する知識の状態を観察して、設計過程に関する知識を用いて、対象レベルへの操作を推論するものである。

メタ言語からみた場合、メタ言語で demonstrate 述語を用いることは対象言語における証明可能性を調べること、あるいは実行することになる。したがって、メタ言語で demonstrate 述語を含む論理式を書き、その論理式で推論を行なえば、対象言語での推論を含んだ推論を行なうことができる。例えば、対象レベルの結論が指定されたもの (g_0) であれば、指定された論理式集合 (K_{p_0}) を前提に加えるということを、

$$demonstrate(K, g_0) \rightarrow demonstrate((K_{p_0} \cup K), g_0)$$

と書くことができる。すなわち、メタ言語での推論と対象言語での推論を組み合わせることで、利用する知識や結論の変化する場合の推論を扱うこともできる。

ここでのメタレベルでの推論は、観察に基づく次の行為の決定なので、節形式の論理式による演繹推論であると仮定する。すなわち、前提部では対象レベルの状態を観察する述語を用い、結論部には対象レベルへの操作を指示する述語が用いられる。

今、対象レベルで $Ds \cup K \vdash P$ であることを、メタレベルで $demonstrate(Ds, K, P)$ である、すなわち、

$$Ds \cup K \vdash_{L_o} P \Longleftrightarrow \vdash_{L_o} demonstrate(Ds, K, P)$$

とする。このとき、対象レベルの状態の観察とは前提部の *demonstrate* 述語を照合すること、対象レベルへの操作は *demonstrate* 述語を含む論理式の宣言である。

対象レベルの状態とは、各作業領域の整合性やそこに含まれる論理式と推論の履歴である。すなわち、対象記述や性質的記述で矛盾がないかどうか、対象記述や性質的記述に何が含まれているか、あるいは知識としてどの知識、知識ベースが用いられているかなどである。

対象レベルへの操作とは、対象レベルの推論の実行と対象レベルの作業領域への操作である。推論の実行は変項を含む *demonstrate* 述語の素論理式を宣言することで表現される。*demonstrate* 述語の第1項が変数を含むときにはアブダクションを、第

3 項が変数を含むときには、演繹を示す。また、サーカムスクリプションは別の demostrate 述語 *circumscription* を用いて表現する[10]。

対象レベルの作業領域の操作は、メタレベルでの項に対する追加や削除といった操作で表現される。ただし、対象に関する知識については単調性の要請 (定義5.10) より追加のみが可能である。また、知識はいくかの知識が集まった知識ベースとして操作されると仮定する。

したがって、以下の操作が対象レベルへの操作である。

- 演繹推論の実行
- アブダクション推論の実行
- サーカムスクリプションの実行
- 対象記述への操作
- 現在利用している対象に関する知識への操作
- 性質的記述への操作

3.2 節で、知識の利用に関する行為のレベルの知識の分類を行なった。行為に関する知識は対象レベルの状態から次の行為を決める知識であり、これらはここでのメタ推論のための知識として用いることが可能である。ここで、一般性のある行為に関する知識とは、定項を含まない論理式で示されるものであり、対象に依存する知識とは、定項を含む論理式で示される。3.2 節で示した知識は対象レベルの推論をどう進めるかについての一般的知識である。また、設計の一般的手順も行為のレベルの知識として書かれる。設計サイクルの中でどういう順で各推論をしていくかも行為のレベルの知識として記述される。3.2 節で列挙した知識はこのような設計の対象レベルにおける推論をどう進めるかについての知識である。

また、対象とその対象に利用可能な知識の関係は、行為のレベルの知識と解釈することができる。ただし、この知識は個々の対象に依存する知識である。また、個々の

[10]すなわち、サーカムスクリプションは対象推論系でもメタ推論系でもないもうひとつの推論系とみなしている。

設計に関するヒューリスティックスも行為のレベルの知識として表現される。ある対象に着目したとき、関連する知識を導入する過程はこれらの知識の利用による推論でモデル化される。

この行為のレベルの推論は問題提起段階における新しい知識の導入をモデル化している。さらに、設計サイクル内での推論の順序の制御と設計サイクル間の接続についてもモデル化している。すなわち、メタレベルの推論と対象レベルでのアブダクション・演繹・サーカムスクリプションによる推論が交互に繰り返されて設計が進行する (図 5.4参照)。

5.3.4 アブダクション・演繹・サーカムスクリプション・メタ推論によるモデル化

以上をまとめると、次のようになる (図 5.5参照)。すなわち、

問題提起 メタレベル推論による注目点 T の決定と知識の追加。

$$(T = find\_problem(Ds_i),\ K_j = find\_related\_knowledge(T) \cup K_i)$$

提案 アブダクションによる候補の提出。

$$Ds_j = (Ds_i - T) \cup abduction(T, Ko_j)$$

展開・評価 演繹による対象の性質の導出。

$$P_j = deduction(Ds_j, Ko_j)$$

展開・評価から問題提起 サーカムスクリプションによる矛盾の解消。

$$Ko'_j = circumscription(Ko_j)$$

この推論により、設計の最初においては要求仕様で表現されていた対象は、順次詳細化され、最終的な解となる。すなわち、

$$R_0 \cup \{\} \vdash R_0 \Longrightarrow Ds_{final} \cup Ko_{final} \vdash P_{final},\quad R_0 \subset P_{final}$$

メタ推論で記述すると、

$$demonstrate(R_0, nil, nil) \Longrightarrow demonstrate(Ds_{final}, Ko_{final}, P_{final}),\quad R_0 \subset P_{final}$$

となる。

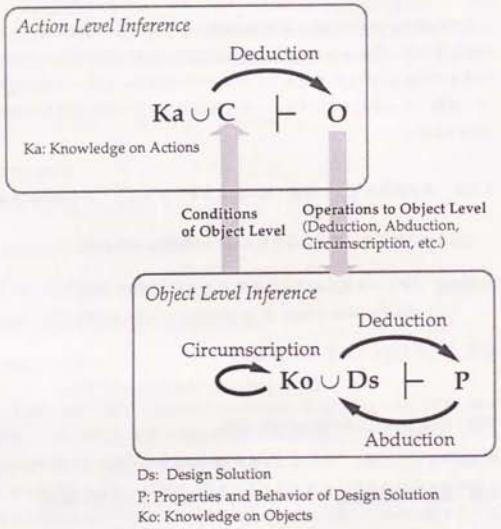


Figure 5.4: 対象レベルと行為レベルによる推論

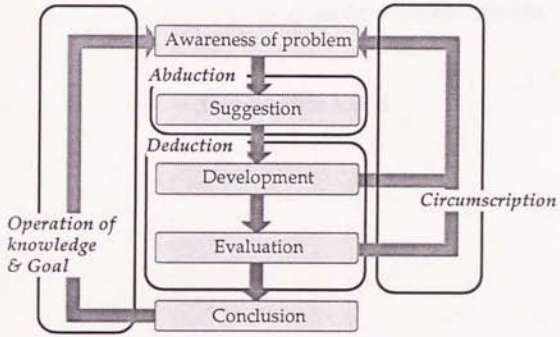


Figure 5.5: 設計サイクルの推論モデル

5.4 まとめ

本章では、設計過程を論理的枠組みの中で形式化を行なった。そこでは設計過程はアブダクションと演繹を繰り返して対象の記述を詳細化していく過程として表現された。さらにサーカムスクリプションとメタ推論を用いて、設計過程の論理的モデルを提案した。このモデルは3章で示した設計サイクルの多くの部分を説明するだけでなく、実際に推論が可能なモデルである。

第 6 章

設計過程表現意味論

本章では論理的な枠組みのなかで、設計対象とくに設計途上の設計対象をどのように表すかについて議論する。これは、論理における意味論 (semantics) であり、論理的な枠組みにおける存在物として、設計対象の表現方法についての議論である。

ここでの焦点は、先に述べた設計対象および知識の表現の性質のうちの「不完全性」と「動的な変化」という 2 点である。まず、この 2 点を満たすような論理的意味論を導入し、その性質を議論するとともに必要な定理等を導出する。次にこれらの意味論を用いた設計過程表現の意味論について議論する。

6.1 部分意味論 (Partial semantics)

古典的な論理のモデル論においては、言語によって規定される全ての対象 (変数のない素論理式に対応する) の真理値は定められる。このようなモデル (完全なモデル) を用意するのが解釈 (interpretation) であり、任意の論理式の真理値はこの解釈により定められる。

ところが、設計は未知のものを既知にする過程である。したがって、設計をはじめる時点で対象については情報がある (真理値が決まっている) というのは不自然である。

また、古典的論理では、情報が増えていく過程を表現できない。というより、古典的な論理においてモデルは静的であり、固定的である。すなわち、モデルが変化することはない[1]。前節まで述べてきた論理的過程とはある論理式集合における証明過程であり、(既にモデル論としては存在している) 与えられた論理式の真理値を手続き的に求める過程であり、モデルが変化しているわけではない。

もちろん、古典的な論理のなかでもこのような不完全性と動的な変化を表現することは不可能ではない。また、デフォールト論理に代表される非単調論理は論理体系として動的な変化を扱うものであり、明示的あるいは暗黙にモデルの不完全性を前提としている。しかし、ここではより自然な表現として、3 値論理を利用した部分論理 (partial logic) あるいは部分意味論 (partial semantics) を導入する。部分論理については Blamey[Blamey86] にまとめられている。また、Turner[Turner84] では、非単調論理との関係で、部分論理を導入して、Kleene の strong matrix が適切であると述べている。また、Treur は応用面から部分論理を議論し、expert system の形式化 [Treur89]、さらには設計の形式化 [Treur90] について述べている。

古典的な論理では真理値は 2 値、すなわち t(真) と f(偽) である。これに対して部分論理では 3 値、すなわち t と f に加えて、u という真理値を取る。部分論理にとっては、この真理値の直感的な意味は未決定 (undecided) である[2]。すなわち、真理値 u は t(真) か f(偽) のどちらかに決められない状態であり、やがては t か f にかわること

[1] もちろん、可能なモデルの集合の変化により、仮想的にモデルの変化をつくることができる。

[2] 3 番目の真理値にどんな直感的な意味を与えるかは任意である。

$\neg A$	
t	f
f	t
u	u

$A \wedge B$	t	f	u
t	t	f	u
f	f	f	f
u	u	f	u

$A \vee B$	t	f	u
t	t	t	t
f	t	f	u
u	t	u	u

$A \rightarrow B$	t	f	u
t	t	f	u
f	t	t	t
u	t	u	u

$A \leftrightarrow B$	t	f	u
t	t	f	u
f	f	t	u
u	u	u	u

Figure 6.1: Kleene の strong 3-valued matrix

もある。しかし、t や f であったものが、u になるということは許されない。これは t や f に関して単調性 (monotonicity) があるということを示している。3 値論理の論理演算は任意に与えることができるが、このうち、表 6.1 に示す Kleene の strong 3-valued matrix はこのような性質を満たす。

形式的な定義は付録を参照されたい。言語 L を第 4 章で定義される述語論理であるとすると[3]、いま、3 値 $\{t, f, u\}$ に対して、$t \sqsupseteq u$、$f \sqsupseteq u$ なる半順序関係 $\sqsupseteq$ を考えると、モデルの半順序関係を定義することができる。この $\sqsupseteq$ は直感的には「決定の度合 (degree-of-definedness)」[Blamey86] を示している。

定義 6.1 言語 L、領域 D のモデル M、M' において、次のような条件を満たすとき、M' は M の *extension*($M' \sqsupseteq M$) であるという。

L の任意の n 項関係 C において、D の任意の $e_0, \ldots, e_{n-1}$ において、

$$C^{M'}(e_0, \ldots, e_{n-1}) \sqsupseteq C^{M}(e_0, \ldots, e_{n-1})$$

である。ただし、$C^M(e_0, \ldots, e_{n-1})$ は $C(e_0, \ldots, e_{n-1})$ のモデル M での値を指す。

これは直感的には、そこから得ることのできる全情報の大小を比較している。

いま、kleene の strong 3-valued matrix の全ての論理演算は単調性をもつので、

[3]一般にはまず、命題論理で定義される。述語論理に拡張するにはいくつかの問題がある。そのひとつは個体の問題である。ここでは単純のため、個体定項の集合は常に同一とする。また、閉論理式のみを対象にする。このとき、述語部分論理は命題部分論理の拡張としてみることができる。

定理 6.1 いま、言語 L の任意の式を A とする。*Kleene の strong 3-valued matrix* においては、$M' \sqsupseteq M$ ならば $[A]^{M'} \sqsupseteq [A]^{M}$ である。

となる (証明は [Turner84] 参照)。これはモデルが単調に変化するとき、そこから得られることも単調に変化するということを示している。これは知識の部分性を表現するのにふさわしい。

いま、素論理式集合 A を全ての $C(e_1, \ldots, e_n)$ からなる集合とし、

定義 6.2 モデル M のダイアグラム $D(M)$ とは、

$$D(M) = \{a | a \in A, [a]^M = t\} \cup \{\neg a | a \in A, [a]^M = f\}$$

である。

と定義する [Treur89]。これはモデルの定義において V を定めていることであるので、ダイアグラムとモデルは一対一対応する。

古典的な 2 値論理とこの部分論理がどのような関係であるかについて知るために、次の 3 つの定理を示す。

定理 6.2 P を閉論理式集合とする。P の古典論理での論理的帰結である素論理式の集合を $th(P)$ とする。また、P を部分論理で満たす部分モデルの集合を Δ とすると、任意の $M \in \Delta$ において、$th(P) \subseteq D(M)$。

定理 6.3 P を閉論理式集合とする。P の古典論理での論理的帰結であるリテラルの集合を $th(P)$ とする。また、P を部分論理で満たす部分モデルの集合を Δ とすると、Δ に最小モデル M_s が存在するとき、$th(P) = D(M_s)$ である。

証明は付録を参照されたい。ここで注意が必要なのは、$th(P) = D(M)$ である部分論理で P を満たすようなモデルが存在するとは限らないということである。例えば、$P = \{A \vee B\}$ としよう。このとき、$th(P) = \{\}$ であるが、P を満たす部分論理モデルは $\{A\}$ と $\{B\}$ と $\{A, B\}$ である。定理 6.3 より、両者が一致するのは、その特別な

場合 (最小モデルが存在すること) であることがわかる。$\sqsubseteq$ は半順序なので、最小モデルが存在することは保証されず、極小モデルの存在のみいえる。P を満たすものの最小モデルが存在せず、極小モデルのみがあるときは、極小モデルの数だけの可能性があるということを示している。例えば、先の例では、$\{A\}$ と $\{B\}$ が極小モデルであり、二つの可能性を示している。

直感的には、部分論理での結果は、古典論理の結果より「強い」とみることができる。これを逆にいうと、古典論理における結果はもとの論理式集合を真とするとは限らないということを示唆している。

定理 6.4 節集合 P において、古典論理での全ての論理的帰結であるリテラル (素論理式またはその否定) の集合を $th(P)$ とする。いま、$D(M) = th(P)$ なる部分論理モデル M、すなわち $th(P)$ をダイアグラムとする部分モデル M を考える。このとき、任意の $\phi \in P$ は $M(\phi) = t$ または $M(\phi) = u$ である。

証明は付録を参照されたい[4]。

以上みてきたように、部分論理を使うことにより、推論の結果、すなわちある前提から得られる情報を部分モデルとして、明示的に取り扱うことができる。これは、状態が順次変化していく過程を示すのには好ましい性質である。そこで次にこの状態の変化自身を明示的に取り扱う方法について考察する。

[4]したがって、古典論理と同様に推論をすることはできない。論理式集合から「推論に用いた」式のみを取り出した部分集合をつくれば、その集合の式は全て t になる。したがって、手続き的な方法によって部分論理での推論を古典論理と同様に定義することもできる [Treur89]。

6.2 データ意味論 (Data semantics)

前節において、部分論理を用いれば部分性をもつ論理が表現できること、そこでのモデル (部分モデル) は古典論理での解釈に対応させることができることを述べた。次に、この部分モデルの単調性をもちいて、情報状態の変化を表せることを示す。

前節で述べたように、部分モデルは単調性をもつ。部分モデルはその extend されたモデルをもつことができるが、ある部分モデルに含まれる情報は、その extension モデルにおいて否定されることはなく、extension モデルは元のモデルにさらに情報が加わることにより構成されたとみることができる。これは不完全な知識のモデルとして妥当なモデルと思われる。一つのモデルに対して、(互いに extension でない)extension モデルを複数持つことができる。これはその情報状態から変化する方向がいくつもあることを示している。次にこの複数の extension を、様相論理の多重世界の枠組みの中で取り扱う。

Veltman と Landman は変化する情報状態での論理として、データ意味論 (data semantics) あるいはデータ論理 (data logic) というものを提案した [Veltman81] [Landman86] [Shirai86] [Kawamori89]。これは様相論理の Kripke モデルに類似したモデルにより、異なる情報状態を表現して、それぞれの情報状態で知りうることを示す。以下で用いる情報状態 (information state) とは、前節で示した部分モデルそのものである。

定義 6.3 情報モデル *(information model)* とは以下の条件を満たす *3* つ組 $(S, \sqsubseteq, V)$ である。

1. S は空集合でない。
2. $\sqsubseteq$ は S の上の半順序関係である。
3. V は S を値域とする関数であり、それぞれの $s \in S$ に対して、Vs は L の素論理式に多くとも *1* か *0* を割り付ける部分関数であり、次の条件を満たす。$s \sqsubseteq s'$ ならば $V_s \sqsubseteq V_{s'}$。

S は可能な情報状態 (possible information state) の集合である。ここで定義される S

における $\sqsubseteq$ が前節で定義された部分モデルにおける $\sqsubseteq$ と同一であるのは、定義より明らかである。

また、情報モデルは閉じていることが条件である。

定義 6.4 情報モデル $\langle S, \sqsubseteq, V\rangle$ が閉じているとは、全ての極大鎖 *(maximal chain)* は $V\,s$ が *total* であるような極大元 *(maximal element)* を持つ。

これは、極大元(極大モデル)とは完全モデルであり、全ての完全モデルを S は含むということを示している。

定義 6.5 M を情報モデル、$s \in S$ をその情報状態とするとき、

1. 素論理式

(a) $s \models^{+}_{M} A \iff V_s = 1$

(b) $s \models^{-}_{M} A \iff V_s = 0$

2. 否定

(a) $s \models^{+}_{M} \neg A \iff s \models^{-}_{M} A$

(b) $s \models^{-}_{M} \neg A \iff s \models^{+}_{M} A$

3. 連言

(a) $s \models^{+}_{M} \phi \wedge \psi \iff s \models^{+}_{M} \phi$ かつ $s \models^{+}_{M} \psi$

(b) $s \models^{-}_{M} \phi \wedge \psi \iff s \models^{-}_{M} \phi$ または $s \models^{-}_{M} \psi$

4. 選言

(a) $s \models^{+}_{M} \phi \vee \psi \iff s \models^{+}_{M} \phi$ または $s \models^{+}_{M} \psi$

(b) $s \models^{-}_{M} \phi \vee \psi \iff s \models^{-}_{M} \phi$ かつ $s \models^{-}_{M} \psi$

5. 可能

(a) $s \models^{+}_{M} \circ\phi \iff$ ある $s' \sqsupseteq s$ において $s' \models^{+}_{M} \phi$ である。

(b) $s \models^{-}_{M} \circ\phi \iff$ いかなる $s' \sqsupseteq s$ においても $s' \models^{+}_{M} \phi$ でない。

6. 必然

(a) $s \models^{+}_{M} \Box\phi \iff$ いかなる $s' \sqsupseteq s$ においても $s' \models^{-}_{M} \phi$ でない。

(b) $s \models^{-}_{M} \Box\phi \iff$ ある $s' \sqsupseteq s$ において $s' \models^{-}_{M} \phi$ である。

7. 条件

(a) $s \models^{+}_{M} \phi \Rightarrow \psi \iff$ いかなる $s' \sqsupseteq s$ においても $s' \models^{+}_{M} \phi$ かつ $s' \models^{-}_{M} \psi$ でない。

(b) $s \models^{-}_{M} \phi \Rightarrow \psi \iff$ ある $s' \sqsupseteq s$ において $s' \models^{+}_{M} \phi$ かつ $s' \models^{-}_{M} \psi$ である。

注意が必要なのはデータ論理では全ての真理値は現在の状態(情報状態)に関して規定される。すなわち、真理値は不変のものではなく、相対的なものである。ここでは $\circ$、$\Box$、$\Rightarrow$ という 3 つの新たな論理記号を導入した。それぞれの直感的な意味は、可能($\circ$)は現在の情報の状態から、それが存在する状態を考えられうる(想像できる)こと、必然($\Box$)は現在の状態から考えうる状態において偽になりえないこと、条件($\Rightarrow$)は現在の状態から考えられうる状態において、前件部が真になるときに、後件部が偽なるようなことがないことである。すなわち、これらのものの真理値はある情報状態そのものだけでなく、その extension を考慮して決められる。すなわち、情報状態の拡張可能性によって決められる真理値である。このため、これらの記号を非記述的(non-descriptive)、それ以外の記号を記述的(descriptive)とよぶ。

これらの論理記号の間には関係がある。以下の定理の証明は付録 B を参照されたい。

定義 6.6 二つの論理式 ϕ, ψ に対して、弱等価 *(weak equivalent)* $\cong$ であるとは、$\phi \Rightarrow \psi$ かつ $\psi \Rightarrow \phi$ であるときである。また、強等価 *(strong equivalent)* $\equiv$ であるとは、$\phi \Rightarrow \psi$ かつ $\psi \Rightarrow \phi$ かつ $\neg\phi \Rightarrow \neg\psi$ かつ $\neg\psi \Rightarrow \neg\phi$ であるときである。

定理 6.5 ϕ, ψ を論理式とする。

1. $\neg\neg\phi \equiv \phi$

2. $\phi \vee \psi \equiv \neg(\neg\phi \wedge \neg\psi)$

3. $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$

4. $\diamond\phi \equiv \neg\Box\neg\phi$

5. $\Box\phi \equiv \neg\diamond\neg\phi$

6. $\Box\neg\phi \equiv \neg\diamond\phi$

7. $\diamond\neg\phi \equiv \neg\Box\phi$

8. $\Box\phi \equiv \neg\phi \Rightarrow \phi$

9. $\Box\Box\phi \equiv \Box\phi$

10. $\diamond\diamond\phi \equiv \diamond\phi$

11. $\Box\diamond\Box\phi \equiv \Box\diamond\phi$

12. $\diamond\Box\diamond\phi \equiv \diamond\Box\phi$

13. $\phi \Rightarrow \psi \equiv \Box(\neg\phi \vee \psi)$

14. $\phi \Rightarrow \psi \equiv \neg\diamond(\phi \wedge \neg\psi)$

15. $\diamond\phi \equiv \neg(\phi \Rightarrow \neg\phi)$

定理 6.6 ϕ を論理式とし、M を様相の列とするとき、$M\phi$ は次のどれかである。

$$\phi,\ \neg\phi,\ \Box\phi,\ \neg\Box\phi,\ \diamond\phi,\ \neg\diamond\phi,\ \diamond\Box\phi,\ \neg\diamond\Box\phi,\ \Box\diamond\phi,\ \neg\Box\diamond\phi$$

また、その強さは次の通りである。

$$\Box\phi \Rightarrow \Box\diamond\phi \Rightarrow \diamond\Box\phi \Rightarrow \diamond\phi$$

$$\phi \Rightarrow \diamond\phi$$

ここで注意する点は、$\phi \Rightarrow \diamond\phi$ は成り立つけれども、一般に様相論理の特徴とされる $\Box\phi \Rightarrow \phi$ は成り立たないということである[5]。$\Box\phi$ が ϕ より "弱い" のが (一般の様相論理に対しての) データ論理の一つの重要な特徴である。

また、ある論理式の真理値は情報状態の変化に伴い、変化しうる。素論理式は当然、単調である (定義より) が、非記述的記号を含む式は必ずしも単調ではない。

定義 6.7 ϕ が単調真 *(T-stable)* であるとは、$\langle S, \sqsubseteq\rangle$ の全ての $s \in S$ において $s \models^+ \phi$ ならば、全ての $s' \sqsupseteq s$ において $s' \models^+ \phi$ であるときである。

定義 6.8 ϕ が単調偽 *(F-stable)* であるとは、$\langle S, \sqsubseteq\rangle$ の全ての $s \in S$ において $s \models^- \phi$ ならば、全ての $s' \sqsupseteq s$ において $s' \models^- \phi$ であるときである。

定義 6.9 ϕ が単調 *(stable)* であるとは、ϕ が単調真かつ単調偽であるときである。

定理 6.7 ϕ が $\neg, \wedge, \vee$ のみからなる式であれば、ϕ は単調である。

定理 6.8 ϕ を単調な論理式であるとき、ϕ のとりうる様相は

$$\phi, \Box\phi, \diamond\phi, \neg\phi, \neg\Box\phi, \neg\diamond\phi$$

のいずれかである。

したがって、基本的にはこれらの様相をもちいればよい。

[5] $\Box\phi \Rightarrow \phi$ は T 様相体系とそれより強い体系で成立する。

6.3 設計対象の動的表現

前節までに説明した部分意味論とデータ意味論を直感的にまとめると次のようになる。部分論理におけるモデルとは、命題の真理値は真であるか、偽であるだけでなく、未知(unknown)であるかの3値を取ることのできるものである。そして、データ意味論ではこの部分モデルを情報状態というあるその時点での情報(知識)の状態に対応付け、その情報状態の集合とその関係で、世界を表現する。情報状態は木構造を構成して、その木構造は根から葉に向かう方向に情報が増えるように関係付けられている(単調性)。

これらの準備のもとに、設計過程における対象記述の意味論を考える。ここでは、データ意味論で導入された情報状態とその関係を対象の記述状態とその変化に対応させる。ここでは、論理記号としては情報状態内のみで定義される $\neg, \wedge, \vee, \rightarrow$ と、情報状態構造において定義される $\Box, \Diamond$ のみを考える(データ論理で導入された $\Rightarrow$ は用いない)。また、以降では様相といった場合、必然様相と可能様相のみを指示し、否定は含まないとする。

6.3.1 過程の表現

設計過程は設計対象に関する必要な情報を増やしていく過程である。ここでいう設計対象に関する情報とは、現在設計している対象に関する記述といった直接的な情報やその設計を進めるのに必要な事項といった間接的な情報を含む。これらを増やしていき、最終的には仕様を満たしかつ存在可能な実体に関する記述を作り出す。ここでは、設計対象に関する情報の変化を用いて、設計過程を表現する。

設計対象に関する情報のそれぞれの状態を、様相論理でのひとつの可能世界と考える。すなわち、各状態にそれに対応する世界が存在して、その世界では、それに対応する設計の状態での設計対象に関する情報を示す命題が真である。

情報の変化は単調的に増加する場合もあるし、取り消しなど非単調的に変化することもある。また、複数の可能性があるときには、二つの状態に分岐することがある(4章参照)。しかし、ここでは世界と世界の関係は単調的であるとする。すなわち、ある状態から次の状態に変化できるときは、次にくる状態が前の状態にある情報を全て持っているときに限るとする。分岐は許すので、非単調的な変化は分岐を用いて表現することにする。ある情報を取り消して新たな情報を付け加えた場合は、その状態より以前の状態から分岐した状態であると見なす。

このとき、世界間の関係は単調性によって関係付けられた木構造であり、6.2節で述べたデータ意味論で定義された関係と同じであり、データ意味論を用いることができる。

設計過程の進行はこの世界の木構造上で次のように解釈される。設計が単調的に進む間は、世界を順に単調増加方向につくっていくことに対応する。設計上で複数の候補を想定する場合は、枝を分岐させ、分岐した枝にある複数の世界を利用することである。また、取り消しや修正といった行為は、枝を遡り、必要なところから新たな枝をはじめることである。

6.3.2 対象の表現

まず、一つの固まりとして認識される「もの」(実体)は定項として表現される。ただし、この定項自身には意味はなく、あくまで識別子として利用される。これには、現実に存在しているものも含まれるし、これから設計するものである仮想的な実体も含まれる。したがって、物理的な世界の存在と論理的世界での存在は必ずしも対応しない[6]。また、ある構造全体も、その部分も論理的世界での実体になりうる。

述語は実体自体の性質や実体間の関係を記述する。識別子としての定項には、述語がつけられて、はじめて意味を持つ。例えば、「あるネジが存在する」とは、定項 $a1$ が存在して、

$$screw(a1)$$

が満たされるときである。そして、「ネジ $a1$ と板 $a2$ が接続している」とは、

$$screw(a1),\ plate(a2),\ connect(a1, a2)$$

[6] 何が論理的世界で存在できるか、どの範囲が存在物であるかについては議論の余地がある(論理の存在論)。仮想的な実体を認める立場もある[Hirst89]。

と記述する。

この記法は、実体に対して事前に規約を設けていないので、実体に対する自由な記述をすることができる。すなわち、識別子をつくることとそれが何を指し示すかは別である。設計は実体を構成していく過程であるので、事前に規約のある項によって実体を記述するのは不自然である。設計過程を進めるにしたがって、実体に性質が付加されていくことが表現できなければならない。

この方針をとった場合、実体の存在のために必須とされる記述とその実体が（付随的に）持つ性質の区別がなくなる。しかし、設計の場合、対象の存在をどう記述するかは一意ではない。例えば、製造可能であるということか、部品のような予め用意されたもので記述されればよいのか、あるいは物理的に存在することであるのか、といくつかの可能性がある。ここでは、ある対象の存在に何が必須であるかは絶対的なものではなく、必要に応じて存在を示す性質を指定すると仮定する。すなわち、実体記述論理式集合 S_{ED} を指定して、任意の論理的実体 $a1$ は $p_i(a1) \in S_{ED}$ なる $p_i(a1)$ が存在するとき、物理的に存在するとする。この集合は、例えば述語の集合 P_{ED} を指定して、論理式集合 $S_{ED} = \{p(x)|p \in P_{ED}, x \in D\}$ ととることができる。

6.3.3 設計対象に関する情報の表現

対象の性質・関係

性質や関係を記述する命題の真理値は、設計過程の状態に依存して真理値が決定される。ある状態で、ある性質・関係が存在するとき、その状態においてその命題が真 (t) である。その性質・関係が否定されるとき、あるいは性質・関係の逆が存在するとき、その命題は偽 (f) である。そのどちらも存在しないときは未知 (u) である。これは、その状態に対応する世界での命題の真理値として表現される。データ意味論では、一度命題は真 (偽) になれば、以降のどの世界でもその命題は真 (偽) である。すなわち、命題を真か偽か決めると、その決定は覆ることはない。例えば、ある時点で「部品 $a1$ と部品 $a2$ を結合する」ことが決定していれば、その時点に対応する世界 (例えば世界 $w1$) で、$connect(a1, a2)$ が成立する。そしてその世界以降の全ての世界でも $connect(a1, a2)$ は成立する (図 6.2参照)。逆に、「部品 $a1$ と部品 $a2$ を結合しない」ということが決定していれば、その時点に対応する世界 (例えば世界 $w2$) で $\neg connect(a1, a2)$ が成立する。$connect(a1, a2)$ も $\neg connect(a1, a2)$ も成立していない世界 (図 6.2での世界 w_0) では、この命題は未知 (u) である。

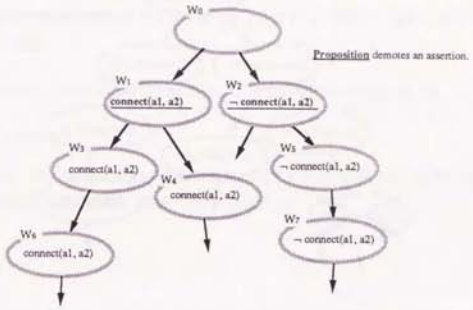


Figure 6.2: 命題の継承

これは、5 章で示した対象の性質的記述の単調性 (定理 5.4) を満たしている。

要求仕様

要求仕様は、設計対象に関する情報の一つであるが、満たすべき条件である点で、他の情報と区別される必要がある。ここでは、要求仕様は必然様相 (□) を用いて表現する。例えば、対象の重量が W であることが仕様であるとき、

$$\Box weight(a1, W)$$

と記述する。データ意味論においては、必然様相は以降の世界で継承される。すなわち、ある世界で必然命題を宣言すると、以降の全ての世界でその必然命題はなければ

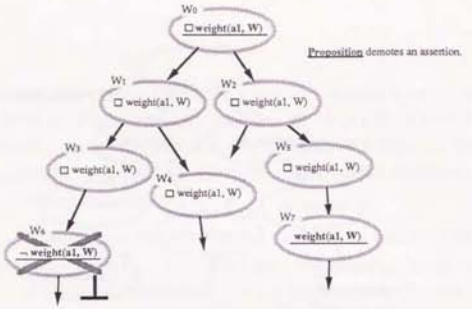


Figure 6.3: 要求仕様の継承

ならない。したがって、要求仕様を必然様相で表現すると、設計過程のいかなる時点でもその要求仕様は必然命題として存在することになる (図 6.3参照)。

また、必然をつけられた命題は命題を含意せず、逆に様相のない命題は必然命題を含意する。先の例でいえば、$\Box weight(a1, W)$ は $weight(a1, W)$ を導かないが、$weight(a1, W)$ は $\Box weight(a1, W)$ を導く。また、否定命題とは矛盾するので、必然様相のついた命題は決して否定されることはない。すなわち、$\Box weight(a1, W)$ と $\neg weight(a1, W)$ は矛盾する (図 6.3参照)。

このとき、設計過程は、必然様相をつけて記述された命題を順に真にしていく過程になる。要求仕様は設計の開始時点では当然満たされていないので未知 (u) であるが、設計を進めるうちに、その各々を真とする設計対象を構成していき、全ての要求仕様の命題を真にしたとき、設計は終了する。そして、設計過程中において要求仕様命題を偽にすることはできない。

また、要求仕様としてとりうる命題に制限はない。すなわち、機能的なことを記述

する命題だけでなく、実体存在命題も書くことができる。例えば、回転を 1/3 にする減速機の仕様は、

$$\Box transmission(a1) \wedge \Box translate(a1, a2, a3) \wedge \Box multiple(a2, 1/3, a3)$$

と書くことができる。

これらは 5 章で議論した要求仕様の満足に関する性質 (定義 5.5、定義 5.6、定理 5.5)、要求仕様の進化の条件 (定義 5.12) に合致する。

設計候補

また、設計過程の次の状態以降で行なわねばならない事項は可能様相を用いて表現される。例えば、ある部品とある部品が結合することがひとつの解として可能なとき、$\diamond connect(a1, a2)$ と表現することができる。しかし、実際にはこのような可能命題の記述は、もともとの命題を真とする新しい世界をつくることで置き換える。これは、可能命題は多重世界の枠組みで解釈するとき、任意性がでるからである。また、設計が後戻りしたときには、そこから進めてえられる設計対象の記述は可能様相で表現される。たとえば、部品 $a1$ と部品 $a2$ を結合する解と、部品 $a1$ と部品 $a3$ と結合する解の二つがあったとき、設計開始時点に対応する世界からは、$\diamond connect(a1, a2)$ と $\diamond connect(a1, a3)$ が見える (図 6.4参照)。

6.3.4 知識の表現

知識の表現は様相 (□ と ◇) を含む論理式とそれ以外の論理式に大別される。

非様相式

様相を含まない論理式は、ある条件や状況で対象がどのような振舞いをするかといった、設計対象の挙動や働きなどを記述する。また、対象の構造や成り立ちも記述する。これらは設計過程に無関係に記述される知識である。例えば、「二つのものが

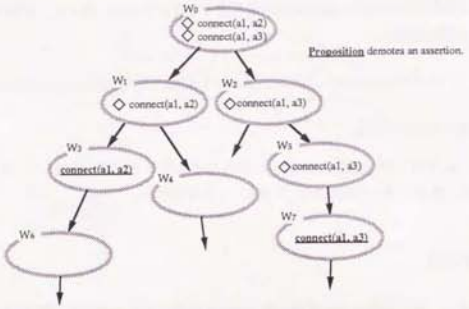


Figure 6.4: 設計候補の表現

上下に結合されていていると、支えになる」というのは、

$$support(x, y) \leftarrow connect(x, y) \wedge upper(y, x)$$

と記述することができる。また、「ギアボックスがギア 2 つとケースからなる」という知識は、ギアボックスであることと 3 つの部分からなる構造が同値であるという形で定義できる。すなわち、

$$gearbox(x) \leftrightarrow has(x, y) \wedge gear(y) \wedge has(x, z) \wedge gear(z) \wedge has(x, u) \wedge case(u)$$

と書ける。

仕様詳細化知識

これは、仕様に関する知識を記述することができる。節形式で書いたとき、後件部が必然命題であるような式は、仕様を詳細化する知識を記述することができる。前件部も必然様相のみならば、仕様自身を設計する知識であり、そうでなければ、そこで

の設計対象の状態を利用して新たな仕様をつけ加える知識である。例えば、「減速機を設計することはギアボックスを設計すればよい」という仕様詳細化知識は、

$$\Box transmission(x) \leftarrow \Box gearbox(x)$$

とかくことができる。また、「モータを使っている場合は、減速機の設計はギアボックスの設計でよい」という知識は、

$$\Box transmission(x) \leftarrow motor(y) \wedge \Box gearbox(x)$$

と書くことができる。ただし、後者は設計過程に依存するアドホックな知識である。

設計候補に関する知識

以降の段階で何をすればよいかということについての知識は可能様相を含んだ式で記述することができる。例えば、「ギアとしては平歯車、はすば歯車が可能である」とき、

$$gear(x) \leftarrow \diamond spurgear(x)$$

$$gear(x) \leftarrow \diamond helicalgear(x)$$

と記述することができる。可能様相をつけた命題を宣言するということは、以降の世界のどこかで真であるということを要請する。したがって、以降の設計過程ですべきことを規定している。先の例では、

$$\diamond spurgear(a1)$$

が宣言された場合、それ以降の世界のどこかで、

$$spurgear(a1)$$

が満たされなければならない。

様相式と非様相式の関係

ここでの様相は設計過程を考えたとき、意味を持つ。すなわち、設計過程でどう使うかに依存して、様相は付けられている。例えば、

$$\Box gear(x) \leftarrow spurgear(x)$$

$$\Box gear(x) \leftarrow \Box spurgear(x)$$

$$gear(x) \leftarrow \Diamond spurgear(x)$$

はどれも、平歯車は歯車であることすなわち、

$$gear(x) \leftarrow spurgear(x)$$

を示している。しかし、ギアや平歯車であるということを仕様と扱うか、あるいは歯車の詳細化はいつ行なうか、といったことを考慮して、異なる式が存在する。

設計における知識とは、単に「もの」や、「もの」と「もの」の関係に関する情報だけでなく、設計でどう用いるのかも含めて知識となっている。したがって、このような様相を使った記述はその一部を表現できるといえる。ただし、全体の無矛盾性を保つためには、様相のある式とそれを取り去った式の両方をうまく管理する方法が必要であろう。

6.4 まとめ

本章では設計過程を論理的枠組みで表現するときの意味論について議論を行なった。設計過程は順次情報が増えていく過程であるので、不完全な情報を扱う意味論が必要であり、そのために部分論理を導入した。さらに、過程を明示的に扱うことが必要であるので、部分論理に多重世界を用いたデータ意味論を導入した。そして、このデータ意味論の枠組みの中で、設計対象の表現、仕様の表現、知識の表現が可能であることを示した。特に、仕様や可能性の表現は、様相を用いることで、体系的に記述可能であった。

第 7 章

設計シミュレーション

前章までに、設計をいくつかの立場からモデル化してきた。1章で述べたように、設計のモデルは記述的 (descriptive) モデル、認知的 (cognitive) モデル、計算可能な (computable) モデルに分けることができるが、本研究では3章で認知的モデル、5章と6章では認知的モデルを基盤に計算可能なモデルを提案した。

知的ＣＡＤのための設計過程のモデルは単に認知科学的に妥当なモデルであるだけでは十分ではなく、計算可能なモデルであることが必要である。前者の意味でのモデルも設計過程の研究を発展させるには重要であるが、それ自身の評価は困難である。これに対し計算可能なモデルは、

1. 数学的な取り扱いが可能である
2. 計算機上に実現することにより、実際の過程との関係を知ることができる。
3. 知的ＣＡＤ構築に直接結び付く。

といった利点を持つ。5章および6章では、(1) の立場から、論理の枠組みの上でモデルを形式的に構築した。本章では (2) の立場、すなわちこのモデルを計算機上に実現することにより、このモデルの有効性・妥当性を検討する。このように計算機によって設計過程を実現することを設計シミュレーションと呼ぶことにする。 本章では、前章まで提案したモデルに基づいた設計シミュレータを構築して、設計実験で得られたデータをもとにした設計シミュレーションを行なう。これにより、モデルの妥当性等を検討する。(3) の問題は8章で検討する。

7.1 システムの構成

設計シミュレータは、5章で述べた設計過程の論理的推論モデルに基づく推論を、6章で述べたような多重世界の上で実現する。その基本構成を図7.1に示す。

基本構成は、対象レベル推論部 (object-level inference system)、行為レベル推論部 (action-level inference system)、対象に関する知識ベース (knowledge base on objects)、行為に関する知識ベース (knowledge base on actions)、多重世界管理部 (multiworld management system)、依存関係管理部 (truth maintenance system) からなる。

対象レベル推論部は3つの作業領域 (workspace) を持つ。すなわち設計対象記述 Ds (assumption 集合) に関する作業領域、性質的記述 P(fact 集合) に関する作業領域、利用可能知識 Ko に関する作業領域を持つ。この作業領域に対して、3つの対象レベルの推論が行なわれる。そのそれぞれは、独立した3つの推論システム、すなわちアブダクション推論部 (abduction subsystem)、演繹推論部 (deduction subsystem)、サーカムスクリプション推論部 (circumscription subsystem) として実現している。

行為レベル推論部は基本的にはルールベースシステムであり、対象レベルの作業領域の状態と推論の実行状態を観察して、行為に関する知識により、対象レベルへの操作を推論する。行為レベルの作業領域には設計過程のコンテクスト (現在行なっている対象レベルの推論と現在いる可能世界など) が含まれる。

多重世界管理部は、設計過程を可能世界の木構造として管理して、設計の後戻りなどの操作を実現する。依存関係管理部は、対象レベルの作業領域に格納される推論結果の依存関係 (dependency) を管理して、推論の取り消しやバックトラックを実現する。

論理式の記述の構文は Prolog/KR[Nakashima83] にしたがっている。

システムは Common Lisp (Allegro Common Lisp) 上で作成して、Sun4 上で実行可能である。また、ユーザ・インタフェース部分は Common Lisp 上の CLX を利用して X11 で記述されている。

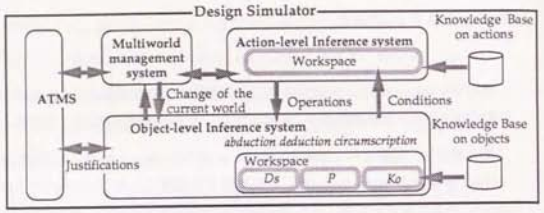


Figure 7.1: 設計シミュレータの基本構成

7.1.1 アブダクション推論部

アブダクション推論に関してはいくつかの研究がある [Finger85][Cox86][Reiter87] が、ここでは Poole[Poole88] の default reasoning の方法を利用する。すなわち、アブダクションとは次のように定義できる。

定義 7.1 可能な仮説集合 Λ と事実集合 F における、ゴール G のアブダクションとは、

$$A \cup F \vdash G$$

かつ $A \subseteq \Lambda$ を満たし、かつ A のどんな真部分集合 A' も

$$A' \cup F \vdash G$$

を満たさないものである。

当然、いくつかの解が可能な場合もある。

可能な仮説集合は任意に取ることができるが、ここでは可能な仮説集合は最大の場合、言語の素論理式集合全体すらとりうるとする。その理由は5章のモデル化において議論したように、仮説として取りうるものは予め一定の集合が与えられるのではなく、その時々において動的に決まるものであるからである。この場合、先の定義に制約をつける必要がある。

定義 7.2 ある仮説集合 Λ と事実集合 F における、ゴール G のアブダクションの二つの仮説において、仮説 A_1 が仮説 A_2 に対して、「より深い」仮説であるとは、

$$A_1 \cup F \vdash A_2$$

を満たすときである。

仮説の深さは反射則と推移則は満たすが、必ずしも反対称則は満たさないので、擬順序 (pseudo-order) である。Horn 節の場合、ループをなしている場合は反対称則は満たさない。以後、ループがないと仮定する。最大な仮説は必ず存在するわけではないが、極大な仮説は存在する。

定義 7.3 事実集合 F に関して極大な仮説とは、それより深い仮説が存在しない仮説である。

定義 7.4 可能な仮説集合 Λ と事実集合 F における、ゴール G の極大なアブダクションとは、

$$A \cup F \vdash G$$

かつ $A \subseteq \Lambda$ を満たす A の中で、極大な仮説である。

以後、極大なアブダクションをアブダクションとする[1]。

定理 7.1 仮説 A が事実集合 F においてゴール G の極大のアブダクションであるならば、任意の $a \in A$ において、a をゴールとするアブダクションは a のみである。

証明 もし、a のアブダクション A_1 が存在するならば、$A_1 \cup F \vdash a$。$a \in A$ なので、演繹定理より、$A_1 - a \cup A \cup F \vdash A$。$A$ は極大の仮説なので、$A_2 \cup F \vdash A$ ならば $A = A_2$。したがって、$A_1 = a$。 □

[1] ただし、この極大性が実際的な意味を持つかどうかはは事実 (知識) の書き方に依存する。ここでは節が5章で述べたように一定の規則にしたがって記述されることが必要である。

この極大なアブダクションは、導出原理 (resolution principle) に基づく Prolog[Lloyd84] の推論を利用して求めることができる。ただし、ここではカット (cut) や失敗による否定 (Negation as Failure) を考えない純粋な Prolog(pure Prolog) を想定する。また、事実集合は Horn 節で表せ、かつループがないものとする。

まず、仮説集合 Λ と事実集合 F が与えられたとき、$\Lambda \cup F$ を Prolog の定義節として、G をゴールとして Prolog による推論を行なう。このとき、推論が成功で終了するときは、$\Lambda \cup F$ は G を論理的に帰結することを意味する。さらに実際にはその部分集合が G を帰結するので、

$$A \cup F' \vdash G$$

ただし、$A \subseteq \Lambda$、$F' \subseteq F$ である A と F' を求めることができる。ここで A と F' は G を帰結する極小の部分集合である。この A と F' は Prolog を実行した場合の利用された節であるので、容易に求めることができる。

ただし、定理7.1より、仮説の極大性を満たすために節の適用に優先順序をつける。すなわち、あるリテラルを他の節のゴールと単一化 (unification) を行なうとき、まず F に含まれる節に対して行なう。F に含まれる節を使った場合が全て失敗したとき、はじめて Λ に含まれる節を試す。こうして得られた仮説は定理7.1より極大の仮説である。

また、可能な仮説集合に変数が含まれる場合とは、存在限量子の意味であるので、(例えば $F(x)$ とは $\exists x F(x)$ である) 任意の定数と単一化が可能である。また、定数と単一化しない場合は、新たに定数を生成して (instantiation)、その定数を含むリテラルを仮説とする。

例えば、

$$\Lambda = \{male(x), female(y)\}$$

と

$$F = \{married(x, y) \leftarrow male(x) \wedge female(y)\}$$

のとき、ゴール

$$G = married(Taro, Hanako)$$

を与えたときは、仮説は

$$A = \{male(Taro), female(Hanako)\}$$

であるが、

$$G = married(Taro, x)$$

であれば、仮説は

$$A = \{male(Taro), female(A1)\}$$

となり、ゴールは

$$G = married(Taro, A1)$$

となる。

設計シミュレータにおいては、アブダクションは全てのリテラルに適用可能ではなく、制限をつけている。すなわち、述語が function カテゴリ、property カテゴリであるもののみアブダクション可能としている (述語カテゴリについては 7.1.4 項を参照)。

7.1.2 演繹推論部

失敗による否定 (Negation as Failure) ではない、陽に否定を意味する *not* を含んだ Horn 節 (すなわち、節) の集合から、可能な事実を導出する。

演繹システムはプログラム節 (前件部がある節) にファクト (アトム節、前件部がない節) を適用することを、新たな導出がなくなるまで繰り返す。

7.1.3 サーカムスクリプション推論部

サーカムスクリプションの計算

サーカムスクリプションは McCarthy[McCarthy80] によって提唱されたものの、その定義に 2 階の述語を用いているので、そのままでは計算が不可能である。しかし、Lifschitz[Lifschitz85] は solitary という形式とその連言形式である separable という形式の論理式に限っては計算ができることを証明した。

今、P を述語定数の組として、Z を P と選言関係で結ばれる関数あるいは述語の組とする。いま、述語の組 P と Z を含む論理式集合 $A(P,Z)$ を考える。P を述語定数 $P_1,\ldots,P_m$ の組とするとき、Z を変数とした $A(P,Z)$ における P のサーカムスクリプション $Circum(A(P,Z);P;Z)$ は

$$Circum(A(P,Z);P;Z) \equiv A(P,Z) \wedge \neg\exists p \exists z(A(p,z) \wedge p < P) \tag{1}$$

である。これは、Z を極小化の過程で変化を許すときに可能な P の外延のうちの、極小な外延 (minimal extension) である。Z が空のときは、$Circum(A(P);P)$ である。ただし、述語の組 $U = U_1 \wedge \ldots \wedge U_m$ と $V = V_1 \wedge \ldots \wedge V_m$ において、$U \leq V$ とは、

$$\forall \bar{x}_1(U_1 \leq V_1) \wedge \ldots \wedge \forall \bar{x}_m(U_m \leq V_m)$$

である。$U = V$、$U \leq V$ も同様に定義される ($\bar{x}_i$ は U_i, V_i に含まれる変数の組を示す)。

論理式 $A(P)$ が P に関して solitary であるとは、$A(P)$ が次の二つの論理式の積として表せるときである。

(i) $P_1,\ldots,P_m$ が正のリテラルとして出現しない論理式、

(ii) $P_1,\ldots,P_m$ を含まない U に対して、$U \leq P_i$、

すなわち、

$$A(P) = N(P) \wedge (U \leq P) \tag{2}$$

ここで、$N(P)$ は $P_1,\ldots,P_m$ が正のリテラルとして出現しない論理式、U は $P_1,\ldots,P_m$ を含まない述語の組である。

このとき、

$$Circum(N(P) \wedge (U \leq P);P) \equiv N(U) \wedge (U = P) \tag{3}$$

である。

さらに、変数がある場合のサーカムスクリプションは、

$$Circum(A(P,Z);P;Z) = A(P,Z) \wedge Circum(\exists z A(P,z);P). \tag{4}$$

であることが証明される。

式 (4) には述語変数が含まれているのでこのままでは計算が困難である。もし、$A(q)$ が q に関して separable である、すなわち、

$$A(q) = \bigvee_i [N_i(q) \wedge (U^i \leq q)] \tag{5}$$

ならば、

$$\exists q A(q) = \bigvee_i N_i[U^i]. \tag{6}$$

であるので、計算することができる。

また、いくつかの述語をサーカムスクライブするときに優先順位をつけることができる (prioritized circumscription)。このとき、

$$Circum(A;P^1 > \ldots > P^k;Z) = \bigwedge_{i=1}^{k} Circum(A;P^i;P^{i+1},\ldots,P^k,Z). \tag{7}$$

すなわち、自分より優先順位の低い述語を変数に加えた場合のサーカムスクリプションの連言となる。

再帰のない節形式の論理式は、solitary である。したがって以上の方法により、prioritized circumscription を計算することが可能である。ここでは中川ら [Nakagawa87] の方法にしたがって計算する。その手順は基本的には上記の式を利用する。

1. prioritized circumscription を式 (7) を利用して parallel circumscription に変形する。
2. circumscription の変数述語を式 (5), (6) を用いて消去する。
3. 各 circumscription に式 (2), (3) を利用して求める。

ただし、not の扱いには注意が必要である。

例えば、以下の式にアブノーマル述語をつけてサーカムスクリプションを行なう。優先順位は、下の論理式ほど高いとする。

not (fly (*x)) ←
fly (*x) ← bird (*x)
fly (*x) ← airplane (*x)

not (fly (*x)) ← ostrich (*x)
not (fly (*x)) ← bird (*x) ∧ dead (*x)

サーカムスクリプションを行い、アブノーマル述語に代入を行なうと下記のようになる。

not (fly (*x)) ← not (not (dead (*x)) ∧ not (ostrich (*x)) ∧ bird (*x))
∧ not (not (ostrich (*x)) ∧ not (dead (*x) ∧ bird (*x)) ∧ airplane (*x))
fly (*x) ← not (dead (*x)) ∧ not (ostrich (*x)) ∧ bird (*x)
fly (*x) ← not (ostrich (*x)) ∧ not ((dead (*x) ∧ bird (*x))) ∧ airplane (*x)
not (fly (*x)) ← ostrich (*x)
not (fly (*x)) ← bird (*x) ∧ dead (*x)

サーカムスクリプションの適用

サーカムスクリプションを実際に利用するにはいくつかの準備と結果の処理が必要である。

サーカムスクリプションは矛盾が発見された時、矛盾を解消されるために用いられる。このときどのルール集合に対してサーカムスクリプションが行なわれるかが問題になる。

まず、ここでは全ての例外を計算するのではなく、発見された矛盾に関係する例外のみを対象とすることにする。したがって、矛盾を導いたファクトを導くのに用いられたルールを対象とする。さらにこのうち、あるファクトを導くのに複数のルールを順次用いた場合、どのルール(あるいは全てのルール)を対象とするのかが問題になる。ここでは順次用いられたルールのうち、最後に用いられたルール、すなわちファクトを直接導いたルールのみを対象とする。これは例外は矛盾を発生したファクトに「より近い」ところにあると仮定しているからである。

そしてサーカムスクリプションの計算においては、先に述べたように複数のアブノーマル述語がある場合、アブノーマル述語に優先順位をつけなければならない。この優先順位の直感的意味は、優先順位が高いほど例外が少ないということである。当然、優先順位の付け方により、サーカムスクリプションの結果は異なってくる。優先

順位はどのルールがより例外を含みうるかを指定するものであるので、本質的にはシステムでは決定することができない。しかし、ここではシステムからの提案として以下の方法によるルールの一般性・特殊性を推定して、順位を提示する。

1. 依存する assumption の包含関係
   そのルールを使う時にどれだけの assumption に依存している(支持されている)かによってルールの特殊性を判定する。支持 assumption の集合が包含関係をなすとき、包含する支持 asssumption 集合をもつルールはより特殊なルールであると認識して、その順位を下げる。

2. ルールの利用設数
   assumption から fact までに用いられているルールの数(ルール利用の設数)によってルールの具体性を推定する。より多くのルールが用いられている方が、その中の個々のルールは一般性をもつと推定する。すなわち、ルール利用の設数が少ないルールの順位を下げる。

サーカムスクリプションを行なって求められたアブノーマル述語はひとつ以上のリテラルの論理積である。したがってその否定は各否定の論理和になるので、節形式に直すと、アブノーマル述語に含められていたリテラル毎のルールが生成される。

先の例の結果の not の後の連言を展開すると、以下のようになる。

not (fly (*x)) ← not (bird (*x)) ∧ ostrich (*x)
not (fly (*x)) ← ostrich (*x)
not (fly (*x)) ← dead (*x) ∧ ostrich (*x)
not (fly (*x)) ← ostrich (*x) ∧ bird (*x) ∧ dead (*x)
not (fly (*x)) ← dead (*x) ∧ bird (*x)
not (fly (*x)) ← not (bird (*x)) ∧ not (airplane (*x))
not (fly (*x)) ← ostrich (*x) ∧ not (airplane (*x))
not (fly (*x)) ← dead (*x) ∧ not (airplane (*x))
fly (*x) ← not (dead (*x)) ∧ not (ostrich (*x)) ∧ bird (*x)
fly (*x) ← not (ostrich (*x)) ∧ not (bird (*x)) ∧ airplane (*x)
fly (*x) ← not (ostrich (*x)) ∧ not (dead (*x)) ∧ airplane (*x)
not (fly (*x)) ← ostrich (*x)
not (fly (*x)) ← bird (*x) ∧ dead (*x)

しかし、このようなルールの分割ができない場合がある。すなわち、アブノーマル述語に中間変数が含まれる場合、分割ができない。この場合は出現する各リテラルの正負で場合わけを行なって、ルールを記述する (例えば3個のリテラルが出現する場合、$2^3 - 1 = 7$ で7通りのルールが生成される)。しかし、実際には fact の性質 (次項参照) より否定の取れないものがあるので、その場合を消去する。否定をとれない述語とはここでは、構造カテゴリと値カテゴリの述語である。

例えば、次の式を考える。

ab = not (indicator(*i)) ∧ not (has (*s *i)) ∧ is-easy-to-see (*i)

このとき、indicator が structure カテゴリである時は、

not(ab)
= not (not (indicator(*i)) ∧ not (has(*s *i)) ∧ is-easy-to-see(*i))
= indicator(*i) ∧ not (has(*s *i)) ∧ is-easy-to-see(*i)
∨ indicator (*i) ∧ not (has (*s *i)) ∧ not (is-easy-to-see (*i))
∨ indicator (*i) ∧ has (*s *i) ∧ is-easy-to-see(*i)

となる。

7.1.4 対象に関する知識ベース部

ここでの知識は対象レベルでの推論で利用される知識であり、節形式の論理式で書かれる。知識は5.1節で述べた定義 (属性化、精緻化、具体化の方向性) にしたがって書かれる。

また、各述語にはその述語の設計における「意味」を示すカテゴリが付けられる。これは6.3.2項で述べたように、性質の記述を区分するために付けられる。現在用意されているのは、structure(構造) カテゴリ、relation(関係) カテゴリ、function (機能) カテゴリ、action(挙動) カテゴリ、property(性質) カテゴリ、quantity (量) カテゴリの6種である。

structure カテゴリは分解不可能なものを示す1引数述語を含むもので、項を「もの」を識別するのに用いられる。relation カテゴリは「もの」と「もの」との関係を

示す2引数以上の述語が含まれる。quantity カテゴリは数値などの定数を認識するための1引数述語が含まれる。function カテゴリ、action カテゴリ、property カテゴリはその他の述語が含まれるが、特に機能的特徴をもつ述語は function カテゴリ、挙動的特徴をもつ述語は action カテゴリ、それ以外に property カテゴリに割り付けられる。

対象に関する知識はいくつか関連するものがまとめられて、知識ベースとして管理される。したがって、知識ベースは節形式の論理式の集合であり、この知識ベースが複数存在している。知識は知識ベースを単位に作業領域に導入され、利用可能な知識の作業領域は知識ベースの集合として表せられる。

サーカムスクリプションを行なって、変更を受けた場合、知識ベースの知識も変更される。すなわち、サーカムスクリプションを行なった結果、ひとつのルールが分割して複数になった場合、それらの全てのルールがもとのルールに置き代わる。

7.1.5 行為レベル推論部

行為レベル推論部は、対象レベルの推論に対して meta-level の推論であるが、ここでは基本的には rule-based 推論として実現している。ルールの前件部は対象レベルの状態の評価であり、後件部は対象レベルに対する操作である。行為レベル推論部は対象レベル状態に変化があると起動され、対象レベルに対する操作を出力する。

対象レベルの状態とは、現在の世界 (current world) の内容、すなわち、

- 対象記述 (assumption) 作業領域の内容
- 性質的記述 (fact) 作業領域の内容
- 利用されている知識ベース

と、これまでの多重世界の内容、

- 過去の操作履歴

であり、ルールの前件部はそれらを評価する関数を選言、連言、否定の結合子を用いて組み合わせて作られる。評価する関数とは、内容の変更の有無、矛盾の有無、特定のカテゴリの述語(述語カテゴリのチェック)の有無などである。具体的には、

- (Can-suggest?)
  現在の assumption にさらにアブダクションが可能な述語(function カテゴリと propery カテゴリの述語)が含まれていれば真。
- (Knowlege-Base-loaded?)
  知識ベースが導入されていれば真。
- (new-KB-loaded?)
  新しい知識ベースが導入されていれば真。
- (new-assumption-add?)
  新しい assumption が足されていれば真。
- (conflict?)
  fact が矛盾をしていれば真。
- (change-rule-his?)
  サーカムスクリプションにより変化したルールがあれば真。
- (Abduced?)
  過去にアブダクションを行なったことがあれば真。

などである。

また、操作としては、

- 演繹推論部の起動
- アブダクション推論部の起動
- サーカムスクリプション推論の起動
- 関連知識導入副システムの起動

- current world の移動
- 多重世界操作モードへの移行

あるいはこれらの列である。ここでは、現在の対象記述に関連する知識を対象知識作業領域にくわえる行為知識は、関連知識導入副システムで行なわれる[2]。この副システムは知識と知識ベースの関連を管理しており、現在の対象記述に関係する知識ベースを作業領域に導入する。

7.1.6 行為に関する知識ベース部

行為レベル推論部で用いられる知識が管理される。3.2 節で述べた行為に関する知識が前項で定義される形式で書かれる。例えば、

「矛盾が起きたら、矛盾解消の作業を行なう。」
`if (conflict?) then (KB-circumscription)`
「新しい仮説が提案されたら、それを展開、評価する。」
`if (new-assumptions-add?) then ((auto-select-KB) (deduction))`

となる。

7.1.7 多重世界管理部

多重世界の意味

設計過程中での設計対象の記述は 6.3 節で示したように木構造を持つ可能世界によって管理される。

可能世界は対象記述の状態の変化に対応して作られる。すなわちアブダクションが行なわれる毎に新しい世界が作られる。その世界はその前の状況を示す世界の次に接続される。複数のアブダクションの候補がある場合は複数の世界を作ることが可能で

[2] 5 章で議論したように、本来は他の行為に関する知識と同様に書かれるべきである。

ある。ただし、6.3 節と同様、接続される可能世界は常に上位の世界を論理的に含意するときのみ許される。すなわち、演繹によって導きだされる fact の集合は常に含意関係にある。

したがって、対象記述の取り消しや訂正 (asusmption の取り消し) はそのまま接続することはできない。可能世界の木構造を遡り、適切な世界 (すなわち含意関係を満たす世界) の下位に接続する世界として生成される。また、サーカムスクリプションによりアブダクションに用いられたルールが変更された場合も assumption の変更が必要になる。この場合は世界の生成情報をもとに適切な世界を選択して、そこへ新しい世界を接続する。

各可能世界は、(その前の世界に対して) その世界で新しく付加された assumption、利用可能な知識ベース集合、世界の生成情報、および世界の利用情報を持つ。世界の生成情報とはどのような理由でその世界が生成されたかを示すもので、アブダクションによって生成された場合は、アブダクションに用いられたルールが記述される。世界の利用情報とは、子孫の世界から矛盾などの原因でその世界に戻ってきた場合、その戻ってきた理由が記述される。これらの情報は current world の移動の際などに利用される。

多重世界の利用方法

多重世界管理部は常に現在の世界 (current world) をひとつ持ち、その世界の内容は対象ワークスペースの内容である。

Current world を変化させる場合、ワークスペースの内容に変更があるときはそれを世界の記述したあと、次の世界の内容をワークスペースに示す。そのとき、多重世界の木構造に対応する世界がない場合 (current world が新しい世界の場合)、新たな世界をつくり、多重世界の木構造に接続する。

アブダクションを繰り返すことで設計が進行する場合は、新しい世界の生成と current world の変化はアブダクションに応じて、自動的に行なわれる。また、assumption の変更やサーカムスクリプションによる知識変更による新しい世界の生成も自動的に行なわれる。また、ユーザによる current world の移動も行なうことができる。

また、多重世界全体に対する検索も可能である。多重世界の検索は assumption、fact の両方に対して可能様相を含む条件式を用いることができる。すなわち、条件式は次のように解釈される。

1. p
   p がリテラルであればそのリテラルを含む世界集合
2. M(p)
   p がリテラルであればそのリテラルの否定を自分の子孫世界に含まない世界集合
3. (and p q)
   p を満たす世界集合と q を満たす世界集合の積集合
4. (or p q)
   p を満たす世界集合と q を満たす世界集合の和集合
5. M(and p q)
   (and p q)) の否定をを自分の子孫世界に含まない世界の集合[3]
6. M(or p q)
   M(p) を満たす世界集合と M(q) を満たす世界集合の和集合[4]

ユーザはこのような多重世界に対する直接操作や複数の世界間の比較を行なうことができる。このようなユーザによる多重世界管理システムの利用は行為レベルの操作のひとつ (多重世界操作モード) として実現される。

7.1.8 依存関係管理部

依存関係管理部は、対象レベルの作業領域にはいる推論結果の依存関係 (dependency) を管理して、推論の取り消しやバックトラックを実現する。ATMS (assumption-based truth maintenance、仮定に基づく真理保持機構) [de Kleer86] に多重世界管理

[3] 付録 B より、$s \models^{+} \circ(\phi \wedge \psi)$ ならば $s \models^{+} \circ\phi \wedge \circ\psi$ であるので、$(andM(p)M(q))$ にはならない。

[4] 付録 B より $\circ\phi \vee \circ\psi \equiv \circ(\phi \vee \psi)$ なので、$(orM(p)M(q))$ である

を含めた機構 [Morris86] を利用して実現している。この依存関係管理部において、推論の分岐は多重世界として管理され、別の枝への移動などが実行される。

このシステムにおいては ATMS の assumption として、可能世界を示す assumption(world assumption) と、アブダクションによる仮説の導入を示す assumption、知識の導入を示す assumption を用意する[5]。知識と対象記述から導かれた性質的記述 (fact) は derived node であり、他の node からの依存関係が記録される。ATMS はこの各 node の依存関係を ATMS assumption 集合 (environment) の選言として管理する。このため、ある性質的記述がどの知識や対象記述に依存しているかどうかを容易に知ることができる。すなわち、世界間の関係、仮説の設定、知識ベースの導入、fact の導出の際のルールの利用が ATMS で管理される。

本システムでは可能世界は ATMS assumption として表せられるので、ある指定された世界で成り立つ性質的記述を知ることは、必要とする environment を変化させることで容易に知ることができる。また、ある特定の性質を成り立つ可能世界を探すこともできる。また、可能世界から可能世界への移動は environment の変化で行なわれる。

ここでは ATMS システムは de Kleer の作成したシステムを利用した。

7.1.9 ユーザ・インタラクションとシステムの外観

現在、いくつかの部分をユーザの選択や入力によって行なうようにしている。

- 要求仕様の入力
- 行為のレベルでの推論結果からの選択
  行為レベルでの推論結果は複数であるので、表示してユーザに選択させる。
- アブダクションの対象となる仮説の選択
- サーカムスクリプションの優先順位の指定

[5] ここでは、混乱を避けるために、ATMS での仮定を ATMS assumption と書くことにして、アブダクションによる仮説と区別する。

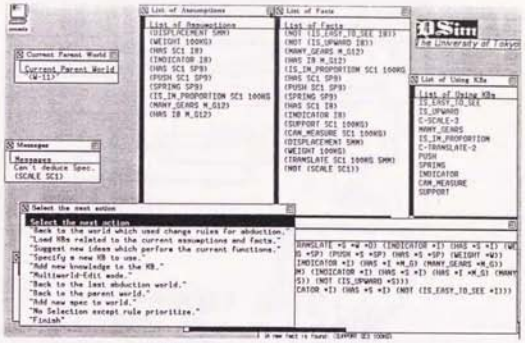


Figure 7.2: システムの表示例

- 導入する知識ベースの指定
- current world の移動
- 複数世界の比較

などである。これらは適宜表示される。

システムの表示例を図 7.2に示す。これは行為のレベルでの推論結果からの選択を行なっている画面であり、そこに表示されている内容の説明を図 7.3に示す。この他にもサーカムスクリプションのときなど、必要な情報が適宜表示される。

7.1.10 推論の流れ

推論は図 7.4のように行なう。すなわち、最初に要求仕様を入力した後は、行為レベルの推論と対象レベルの推論を交互に行ない、もはや何も新しいことができなくなるまで推論を続ける (図 7.4の A の流れ)。ユーザは行為レベルから対象レベルへ変わるときに、次の操作の選択というかたちで関与する。また、適宜多重世界操作モードに移ることにより、これまで行なってきた設計過程全体を概観して、複数の候補を比

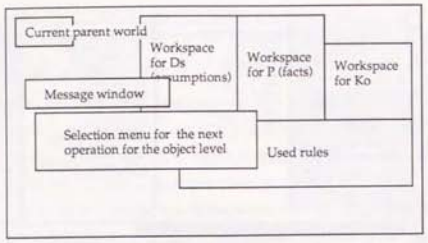


Figure 7.3: システムの表示例の説明

較したり、指定した世界からの対象レベルでの推論を継続することができる (図 7.4で B の流れ)。

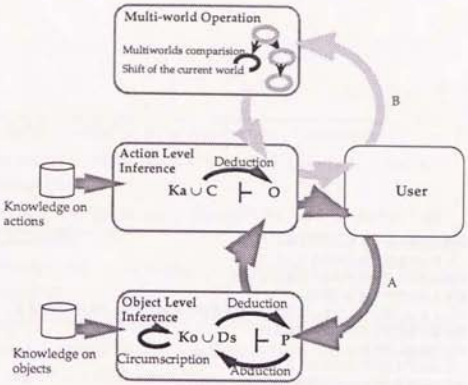


Figure 7.4: 推論の流れ

7.2 設計シミュレーションの実際と考察

前節で説明した設計シミュレータを用いて、設計実験で得られたプロトコルをもとにしたデータで実行して、実際の設計過程との比較を行なう。

7.2.1 対象となるプロトコルと知識

対象となるプロトコルは実験 III-B-3 の最初の部分を抜きだしたものである (図 7.5)。

このプロトコルを元に図 7.6に示すような論理式を用意し、これが対象に関する知識であるとした。これらの論理式はここでの設計者が持っていたと思われる知識、あるいは設計者のその場での思考を表現したものである。各論理式の前につけた番号は発話の番号に対応するものである。

1 普通の体重計はどうなっているのか？
2 こうやって変位を測るのだろう (図1)。
3 ばねは下から押すのではなくて逆でもいい。
4 押すとすればこうなる (図2)。
5 ラックとピニオンでこうすれば (図3)、重さと変位が比例するのを利用して測れる。
6 他に重さをささえられるものはあるか。
7 やはりばねしかない。
8 とりあえずその前に表示を考えよう。
9 要するに変位 (図1の x) が測れればなんでも良い。
10 見易い方がいいだろう。
11 5mm の変位を 100kg にするのだから 1kg 当りの変位は 0.05mm だ。
12 こんなのはこれ (図3) では無理だ。
13 精度はともかくとしてギアを何段もかませればできる。
14 しかし売っているものは表示が上から見えるようになっている。
15 するとこれでははすば歯車を使ってやるしかないが、売っているものはもっと単純な機構に違いない。

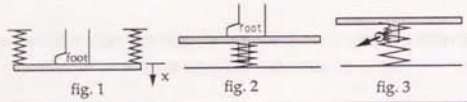


Figure 7.5: 例題のプロトコル

1	scale(*s) ← weight(*w) ∧ can-measure(*s *w) ∧ support(*s *w)
2-1	can-measure(*s *w) ← translate(*s *w *d) ∧ indicator(*i) ∧ has(*s *i) ∧ weight(*w)
2-2	translate(*s *w *d) ← is-in-proportion(*s *w *d) ∧ weight(*w) ∧ displacement(*d)
3	support(*s *w) ← spring(*sp) ∧ push(*s *sp) ∧ has(*s *sp) ∧ weight(*w)
4	support(*s *w) ← spring(*sp) ∧ pull(*s *sp) ∧ has(*s *sp) ∧ weight(*w)
5	is-in-proportion (*s *w *d)← ∧ rack-and-pinion(*r-a-p) ∧ has(*s *r-a-p) ∧ weight(*w) ∧ displacement(*d)
10-1	¬ is-easy-to-see(*s) ← ¬ is-upward(*s)
10-2	is-easy-to-see (*s) ← is-upward(*s)
12	¬ translate(*s 100kg 5mm) ← indicator(*i) ∧ has(*s *i) ∧ is-in-proportion(*s 100kg 5mm) ∧ weight(100kg) ∧ displacement(5mm)
13	translate(*s 100kg 5mm) ← ∧ indicator(*i) ∧ has(*s *i) ∧ has(*i *m-g) ∧ many-gears(*m-g) ∧ is-in-proportion(*s 100kg 5mm) ∧ weight(100kg) ∧ displacement(5mm)
14-1	¬ is-upward(*i) ← indicator(*i) ∧ has(*i *m-g) ∧ many-gears(*m-g)
14-2	¬ scale(*s) ← indicator(*i) ∧ has(*s *i) ∧ ¬ is-easy-to-see(*i)
15-1	is-upward (*i)← indicator(*i) ∧ helical-gear(*h-g) ∧ has(*i *m-g) ∧ many-gears(*m-g) ∧ has(*i *h-g)
15-2	¬ is-easy-mechanism(*s) ← helical-gear(*h-g) ∧ has(*s *h-g)

Figure 7.6: 準備した知識

7.2.2 プロトコルに対応した設計シミュレータの動作

以下では、論理式p(a, b)を(p a b)と表記する。

最初に

```
((scale sc1)(translate sc1 100kg 5mm)(weight 100kg)(displacement 5mm))
```

を要求仕様として与える。このときの設計記述と性質的記述の内容は図7.7である。

List of Assumptions	List of Facts
(DISPLACEMENT 5MM)	(DISPLACEMENT 5MM)
(WEIGHT 100KG)	(WEIGHT 100KG)
(TRANSLATE SC1 100KG 5MM)	(TRANSLATE SC1 100KG 5MM)
(SCALE SC1)	(SCALE SC1)

Figure 7.7: 設計記述、性質的記述の変化(世界1)

アブダクションと演繹を3回ずつ繰り返すと、設計記述は図7.8のようになる。このとき使われた知識は式(1),(2-1), (3)である。この時点で、設計対象はバネと表示部をもち、100kgを5mmに変換する機能があるというものになっている。これは発話の1から4に対応する。

さらに式(12)まで知識を導入して、アブダクションを行なうと、対象記述(assumption)は図7.9のようになるが、この対象記述をもとに演繹を行なうと、

```
(translate sc1 100kg 5mm)
```


List of Assumptions	List of Facts
(DISPLACEMENT 5MM)	(HAS SC1 SP9)
(WEIGHT 100KG)	(PUSH SC1 SP9)
(TRANSLATE SC1 100KG 5MM)	(SPRING SP9)
(HAS SC1 I8)	(HAS SC1 I8)
(INDICATOR I8)	(INDICATOR I8)
(HAS SC1 SP9)	(SUPPORT SC1 100KG)
(PUSH SC1 SP9)	(CAN_MEASURE SC1 100KG)
(SPRING SP9)	(DISPLACEMENT 5MM)
	(WEIGHT 100KG)
	(TRANSLATE SC1 100KG 5MM)
	(SCALE SC1)

Figure 7.8: 設計記述、性質的記述の変化(世界5)

と

```
(not (translate sc1 100kg 5mm))
```

が両方が導かれ、性質的記述において矛盾する。これは発話12での問題点発見に対応する。

このとき、式(2-2)と式(12)を対象としてサーカムスクリプションを行い、矛盾を解消する。このときの式の変化を図7.10に示す。ここでは荷重100kgあるいは変位5mmのときは比例するだけでは荷重の変位への変換ができないというルールに変化する。この状態では図7.9の設計対象記述(assumption)は仕様を満たしていない。これは発話12の時点ではこれまでの設計が否定されていることに相当する。

次に式(13)がはいり、アブダクションを行なうと多段歯車列を使うことになる(図7.11)。しかし、このためには先の設計対象記述では発話12での仕様を満さないとした知識を変更する必要があり、再びサーカムスクリプションを行っている。

式(14-1)から(14-2)が入ると、また矛盾を起こす。これは発話14で「見やすくなければ体重計ではない」という知識がこれまでの設計対象記述(assumption)と矛盾するからである。このときはサーカムスクリプションの結果、式(1)が変更され、

List of Assumptions
(DISPLACEMENT 5MM)
(WEIGHT 100KG)
(HAS SC1 I8)
(INDICATOR I8)
(HAS SC1 SP9)
(PUSH SC1 SP9)
(SPRING SP9)
(IS_IN_PROPORTION SC1 100KG

Figure 7.9: 設計記述、性質的記述の変化 (世界6)

(12) not(translate(*s 100kg 5mm))← indicator(*i) ∧ has(*s *i)
∧ is_in_proportion (*s 100kg 5mm)
∧ weight(100kg) ∧ displacement(5mm) ∧ **not(ab1)**
(2-2) translate(*s 100kg 5mm)← is_in_proportion (*s *w *d)
∧ weight(*w) ∧ displacement(*d) ∧ **not(ab2)**

Circumscription

ab1 = *false*
ab2 = (*w==100kg ∧ *d==5mm)

(12) not(translate(*s 100kg 5mm))← indicator(*i) ∧ has(*s *i)
∧ is_in_proportion (*s 100kg 5mm)
∧ weight(100kg) ∧ displacement(5mm)
(2-2-1) translate(*s 100kg 5mm)← is_in_proportion (*s *w *d)
∧ weight(*w) ∧ displacement(*d) ∧ **not(*w==100kg)**
(2-2-2) translate(*s 100kg 5mm)← is_in_proportion (*s *w *d)
∧ weight(*w) ∧ displacement(*d) ∧ **not (*d==5mm)**

Figure 7.10: サーカムスクリプションの計算例

List of Assumptions	List of Facts
(DISPLACEMENT 5MM)	(NOT (IS_UPWARD I8))
(WEIGHT 100KG)	(MANY_GEARS M_G12)
(HAS SC1 I8)	(HAS I8 M_G12)
(INDICATOR I8)	(IS_IN_PROPORTION SC1 100KG
(HAS SC1 SP9)	(HAS SC1 SP9)
(PUSH SC1 SP9)	(PUSH SC1 SP9)
(SPRING SP9)	(SPRING SP9)
(IS_IN_PROPORTION SC1 100KG	(HAS SC1 I8)
(MANY_GEARS M_G12)	(INDICATOR I8)
(HAS I8 M_G12)	(SUPPORT SC1 100KG)
	(CAN_MEASURE SC1 100KG)
	(DISPLACEMENT 5MM)
	(WEIGHT 100KG)
	(TRANSLATE SC1 100KG 5MM)
	(SCALE SC1)

Figure 7.11: 設計記述、性質的記述の変化 (世界9)

「体重計は見やすくなければならない」ということが付加される。

さらに式(15-1)から式(15-2)が入ると、再び矛盾を起こし、サーカムスクリプションを行なう。その結果を用いて、アブダクションを行なうと、はすば歯車も用いることになる。

全ての式を入れ終った時点では、図7.12のようになる。これはここで示したプロトコルでの設計した内容に対応するものである。

これまでの設計過程の流れを可能世界の木構造として示すと図7.13のようになる。このうち、世界1(w1)の内容は図7.7、世界5(w5)の内容は図7.8、世界6(w6)の内容は図7.9、世界9(w9)の内容は図7.11、世界18(w18)の内容は図7.12、に示したものである。そして、この図の⊥のある世界は矛盾を起こした世界を示し、その中でも灰色の世界はそこでサーカムスクリプションが行なわれたことを示している。また、×は仕様を満たさない世界である。

設計者の視点の変化は世界につけた番号の順が示している。すなわち、まず、世界5まで設計を進めるものの、いくつかの問題点をみつけ何度か解決を模索して、ひとつ解をみつける(世界9)。しかし、その解を展開するのは問題点を発見して、一度戻り、今度はまた新たに違う方法で設計を試み、最後に到達している。

List of Assumptions	List of Facts
(DISPLACEMENT 5MM)	(HAS SC1 R_A_P20)
(WEIGHT 100KG)	(RACK_AND_PINION R_A_P20)
(INDICATOR I13)	(HAS SC1 SP18)
(HAS SC1 I13)	(PUSH SC1 SP18)
(HAS I13 H_G16)	(SPRING SP18)
(MANY_GEARS I17)	(IS_UPWARD I13)
(HAS I13 I17)	(NOT (IS_EASY_MECHANISM I13)
(HELICAL_GEAR H_G16)	(HAS I13 H_G16)
(HAS SC1 SP18)	(MANY_GEARS I17)
(PUSH SC1 SP18)	(HAS I13 I17)
(SPRING SP18)	(HELICAL_GEAR H_G16)
(HAS SC1 R_A_P20)	(INDICATOR I13)
(RACK_AND_PINION R_A_P20)	(HAS SC1 I13)
	(IS_EASY_TO_SEE I13)
	(IS_IN_PROPORTION SC1 100KG
	(SUPPORT SC1 100KG)
	(CAN_MEASURE SC1 100KG)
	(DISPLACEMENT 5MM)
	(WEIGHT 100KG)
	(TRANSLATE SC1 100KG 5MM)
	(SCALE SC1)

Figure 7.12: 設計記述、性質的記述の変化 (世界18)

全体(15の発話)では、8個の知識ベースを用意して、17回の演繹、 12回のアブダクション、4回のサーカムスクリプションを行ない、18個の可能世界を生成した。そしてこれらの推論の結果として、設計者の行なった過程にほぼ相当する過程が行なわれた。例えば、この間設計者は4回自分の設計に用いた知識を訂正しているが、それは設計シミュレータの方ではサーカムスクリプションを行なっていることに相当している。また、ふたつの可能性を出しているが、それは矛盾を起こしていない枝が2本あることに相当する。

7.2.3 行為レベルの推論

次に行為レベルがどのように推論したかを、先の例で発話12に対応する部分を例に示す。

1. まず、その前の推論で新しい仮説

   `(is-in-proportion sc1 100kg 5mm)(weight 100kg)(displacement 5mm)`

   が出された(対象レベルの状態)ので、その仮説に関係する知識ベース(ここでは「比例」に関する知識ベース)が新たに対象知識の作業領域に追加され、さら

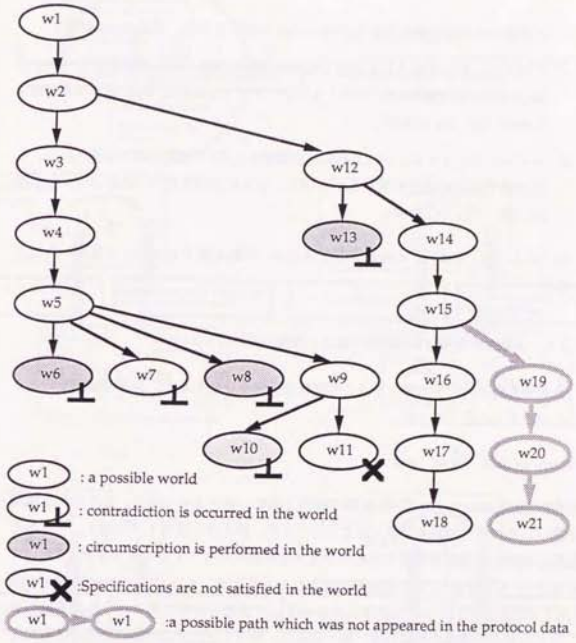


Figure 7.13: 生成された可能世界

に対象レベルの演繹を行なう (対象レベルに対する操作) (図 7.14(a) 参照)。

2. このとき、新たに導入した知識を使った結果、対象レベルで矛盾が見つかった (対象レベルの状態) ので、サーカムスクリプションを行なう (対象レベルに対する操作) (図 7.14(b) 参照)。

3. サーカムスクリプションにより知識が変更された (対象レベルの状態) ので、その知識で現在の仮説が導けるかを試し、さらに演繹を行う (対象レベルへの操作) (図 7.14(c)(d) 参照)。

以上のように、発話 12 に対応する部分では、行為に関する知識は 3 回用いられている。

7.2.4 多重世界を用いた他の解の導出と知識の再利用

多重世界操作モードにおいてはこの複数の世界を比較したり、世界の移動を行なうことができる。図 7.15では、

```
(support sc1 100kg 5mm)
```

を設計記述 (assumption 集合) に持つ世界を選択、表示させている。この中から世界を選び、そこから推論を継続することができる。例えば、世界 1 5 に戻り、そこから推論を継続して最終的に図 7.16に示すような状態にいきつくことができる (図 7.13では世界 19 から世界 21 までの枝が生成されたことを示している)。これは世界 18 (図 7.12) とばねを「引く」か「押す」かという点で異なる結果である。この結果はプロトコルの方では考慮されていない解であるが、持っていた知識から求めることのできた解のひとつである。

また、設計シミュレーションを行なうと、サーカムスクリプションにより知識が変更される。この例では 4 回のサーカムスクリプションにより、4 つのルールが変更され、8 個のルールに変化している (ルールの増加については 7.1 節のサーカムスクリプションの方法参照)。この変更された知識を用いて、再びはじめから推論を行なうことができる。この場合、サーカムスクリプションが行なわず、アブダクション 7 回、演繹 8 回、可能世界 8 個で同じ解 (図 7.12) に到達することができる。

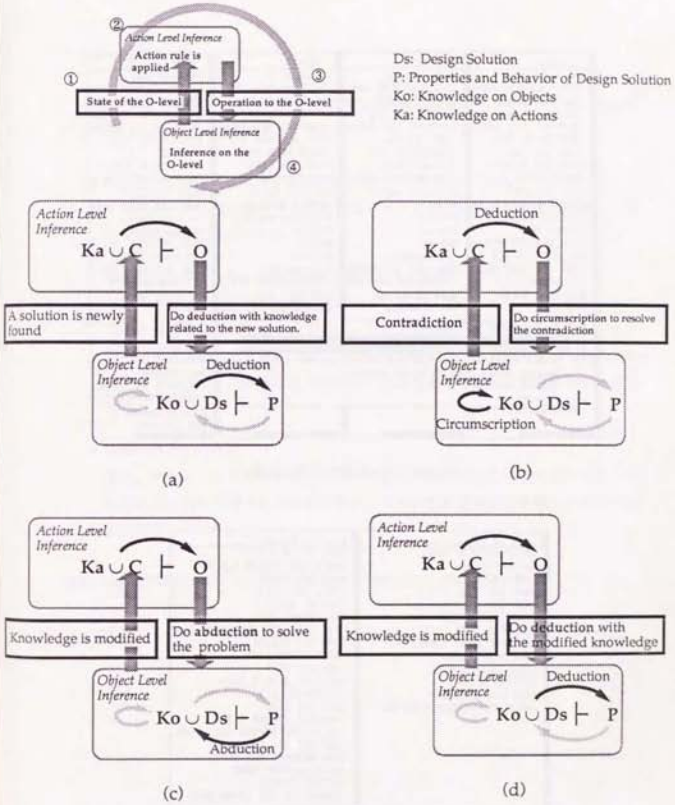


Figure 7.14: 行為レベルの推論の例

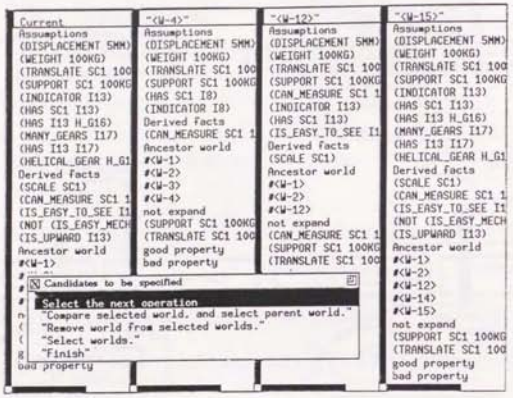


Figure 7.15: 複数の世界の比較

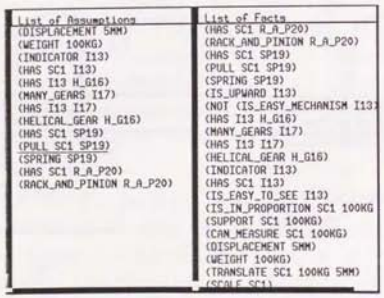


Figure 7.16: 設計記述、性質的記述の変化 (世界2 1)

7.2.5 結果の考察

以上のように、設計シミュレータによる設計実験の過程のシミュレーションから次のようなことをいうことができる。

- 設計者の設計過程に対応する設計過程をつくることができる。
  すなわち、本研究で提案した設計過程のモデルは、実際の設計過程の多くの部分を説明することができる。

- 設計者の行なわなかった設計過程も実現することができる。
  モデルからは多くの可能な過程が導出されるが、設計シミュレータはそれらの可能な過程を実現することができる。したがって、同じ知識を用いてもいくつかの過程を実現することができ、その一つが設計者の行なった過程である。すなわち、可能であるが設計者が実際に行なわなかった過程も実現することができる。

- 知識の構造化の実現
  また、サーカムスクリプションによって修正された知識を知識を用いると、同じ解により速く到達することができる。これは設計者による経験の利用の一側面を実現している。

これらは本設計過程モデルの妥当性および有効性を示している。

7.3 推論システムとしての設計シミュレータ

本節では推論システムとしての設計シミュレータを考察する。

7.3.1 動的な論理的推論

設計シミュレータは前提(公理)と結論を動的に変化させていく論理的推論である。導き出される仮説に応じて新たな前提が付け加えられ、その新たな前提をもとにさらに推論が進んでいく。したがって、推論が分岐する場合は、複数の前提をもとに推論が行なわれる。

人間の推論においては、知識が平等に用いられる分けではない。関連する知識のみを利用して推論を行なっている。特に設計の場合、設計者がもつ知識は膨大であり、それらをすべて用いて推論を行なうというは、実際的でないばかりでなく、認知的な理解や直感的な理解に反する。設計は常に(全体からみたとき)部分的な推論を行なっている。本システムはこの部分的な推論を実現している。

7.3.2 多層知識ベース推論

設計シミュレータは対象に関する知識、行為に関する知識という二つの種類の知識を利用する。対象に関する知識だけでも、推論を行なうことはできる。しかし、行為に関する知識を利用することで、知識が限定された推論や目的を限定した推論(例えば、解の評価の度合の制御など)を行なうことができる。設計の手順は、用いる知識の指定、用いる仮説の指定という形の行為に関する知識として記述される(8章参照)。

設計では、設計段階や設計分野の違いで、2種類の知識の利用の度合が異なる推論が行なわれている。すなわち、概念設計や新規設計では、対象に関する知識に多く依存した推論がおこなれるが、詳細設計や定型的設計では対象に関する知識より、行為に関する知識に基づいた推論が行なわれている。設計シミュレータはこのような多様な推論に柔軟に対応することができる。

7.3.3 知識の自己構造化

設計シミュレータは、結果は終了時点での設計記述(仮説)だけではない。仮説を導いた知識やその過程で用いられた知識ベース集合も残される。

再び同じ、あるいはパラメータのみ違う設計を行なうときは、この使われた知識の集合のみを用いるだけで良い。この意味ではこの知識を指定することはルーチン設計あるいはパラメトリック設計を実現することである。また、類似した設計を行なう場合、先に使われた知識ベースを予め指定することで、より直接的に解を得ることができる。

設計を一度行なうことで得られる、設計と知識集合あるいは知識ベース集合のこのような関係は、新たな知識ベースの生成や、新しい行為に関する知識を作ることで、元々の知識にフィードバックをかけることができる。設計を繰り返すことで、設計知識はそこでの設計に合わせて変化する、すなわち「設計知識の自己構造化」が可能になる。これは設計における経験をモデル化しているといえる。

7.4 まとめ

本章では、5章および6章の論理的枠組みによる設計過程モデルを設計シミュレータとして計算機上に実現した。このシステムは演繹推論、アブダクション推論、サーカムスクリプション推論、メタ推論を行ない、計算機上で設計過程をシミュレートする。本章ではまずシステムの概要を示すとともに、各推論の実現方法について述べた。さらに設計実験のプロトコルを利用して、実際にシステムを実行させ、その結果を実際の設計過程と比較した。さらには、このシステムの推論システムとしての可能性を述べた。

第8章

インテリジェントCADの仕様

本研究ではこれまで、設計過程の形式化について議論を行なってきた。実験的なアプローチから設計過程における推論や知識の特徴を明らかにした上で、論理的な枠組みを用いて設計過程のモデルを提案した。本章ではまず、インテリジェントCADの満たすべき条件について議論する。その上でインテリジェントCAD構築に本研究が提案した設計過程のモデル化がどのような役割を果たすかについて議論を行なう。

8.1 インテリジェントCADへの要求機能

インテリジェントCADという概念は近年、多くの研究者によって、定義あるいは性格づけがなされている(例えばAkmanら[Akman90]、大須賀[Ohsuga89])。そこでの議論はそれぞれ異なるが、総じていえることは、これまでのCADに対して、以下に示すような機能を拡大することである。

- 設計支援機能の拡大
  設計者が設計を行なうとき、より便利な環境をつくることである。これは設計者の道具として機能と、設計者の能力の代替能力の二つに分けることができるが、この区別は厳密でない。具体的に求められることとしては、
  - より多様な形での対象表現ができること
  - 対象だけでなく設計過程も情報として管理できること
  - 推論を行なえること
  - 設計のための各種のツールが使えること。
  - より多くの分野の設計を扱えること。
  - 設計に関連する各種の情報の管理もできること

  などが挙げられる。

- より設計者に対して近付くこと。
  これはより人間に使いやすいシステムになることである。CADは自動設計ではなく、設計者がシステムを利用することにより設計を進めていくのであるからこれはCADの本質的機能である。これはひとつには狭い意味でのユーザインタフェース機能の拡充であり、もうひとつは設計者の設計に合わせて、柔軟に変化することができるシステムの実現である。
  - 柔軟かつ多様なman-machine interactionができること
  - 設計者の設計を理解して、その場面場面に合わせたinteractionができること

しかし、これら機能を実現するものを個々に付加していくことは全体としての機能を損う可能性がある。これらの機能を体系的に実現する構造を提供しなければならない。

8.2 インテリジェントCADの仕様

インテリジェントCADに必要な条件をここで次のようにまとめる。最初の2つは設計という行為を取り扱う上での条件であり、残りの3つはそれを計算機システムとして実現する上での条件である。

1. 動的かつ統合的な設計対象のモデルをもつこと
2. 統合的な設計過程モデルをもつこと
3. 個別の専門領域だけでなく、それらの関係を含めた全体を管理すること

   設計に極めて多数の異なる作業を組合せて行なわれる。設計はある程度まで進行した後は、確立した分野の中での作業になる。したがって、個別の専門分野での支援は必要である。しかし、そこまでの過程はそれぞれの分野での知識を利用しながら、分野と分野の中間領域で行なわれる。したがって、分野を含めた全体の関係を管理できることが必要である。

4. システムの構成に対する柔軟性をもつこと

   また、システムが利用できるツールも順次変化するので、システム構成がツールの進化や付加に耐えるシステムでなければならない。

5. ユーザに対する柔軟性をもつこと。

   設計者の設計はその時々、個々の設計者によって変化するので、このような変化を許容するシステムでなければならない。

ここでは本研究で提案した論理的枠組みによる設計過程モデルを核とするCADシステムを検討する。論理的枠組みによる形式化は以下の特徴を持つ。

- 構文論と意味論が分離されている。
- 概念操作の形式化である。
- 推論が可能である。

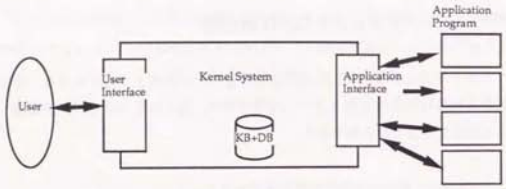


Figure 8.1: インテリジェントCADの構成

さらにここで提案したモデルは、

- 動的な変化を扱うことができる。
- 過程の管理が可能である。
- 分析、総合両方向の推論を含んでいる。
- 知識の管理のモデルである。

本モデルは要求仕様の(1)、(2)を満たすための設計過程に関するモデルである。したがって以下では、このモデルが(3)、(4)、(5)という問題に対してどう対応するかを議論する。

8.2.1 「概念」をキーとする統合化システム

インテリジェントCADの構成を図8.1に示す。中心に核システムがあり、それに各種のサブシステム(例えば、個別領域用の問題解決システム、モデラ、各種解析ツール、データベース)とユーザインタフェースがある。ここでの問題はこの核システムである。

ここでは「概念」とは設計において実体や現象、行為に対する共通性をあるレベルで抽象化したものである。したがってこれは一般設計学[Yoshikawa79]でいう概念と

同一である。ただし、概念はそれ自身では意味を持たず、解釈されてはじめて意味をもつものとする。すなわち、ここでいう概念は「記号」である。また、概念間には関係をつけたり、操作をすることができる。そして、概念間の関係を知識と呼ぶことにする。また、知識を利用して概念を操作することを推論と呼ぶことにする。そして、ここでの概念は論理的枠組みでいうところの項と述語によって表現される。5章で示したように実体はそれを指し示すキーが定項として表現される。そしてその実体のもつ性質が述語として表現される。そして、知識は論理式で、推論は論理的推論で表現される。さらに第6章で示した過程や知識自身を扱うために導入したメタレベルのために、過程や知識といったものも項となりうる。

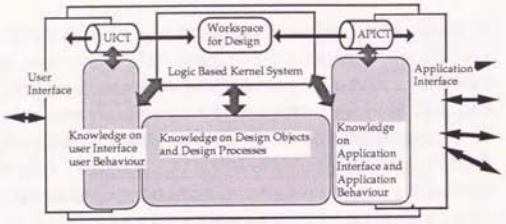
まず、本モデルを利用したインテリジェントCADはものの概念や現象の概念といった「概念」によって結合・統合化されたシステムである[1]。図8.1の核システムは論理的枠組みによるシステムであり、各種のサブシステム(アプリケーション・プログラム)、ユーザインタフェースとは概念レベルのインタフェース、すなわち論理式の形式によって接続されている。また、サブシステムは核システムからみた場合概念によって記述されているので、共通の概念をもつサブシステムはその概念に関する情報を交換することができる。同様に、ユーザはある概念を指定することにより、その概念に関する情報をシステムから得ることができる。

全体の構成を図8.2に示す。ここで核システムはいわば動的概念記述言語(Dynamic Concept Description Language)によって記述されるシステムである。

8.2.2 核システムの機能

核システムは設計対象と設計状況、設計履歴を論理的形式で記述する。また、概念レベルでの設計対象に関する知識、設計過程に関する知識(設計手順、設計方法など)も論理式の形式で管理する。核システム自身は現在の設計対象の状態を含む現在の設計状態とこれまでの履歴を多重世界機構とATMSによって管理する。これによりユーザが行なう設計を整合的に管理して、任意の場面へのバックトラックや複数候補間の比較などを行なう。またそこで管理される設計過程は、知識の構造化(次節参照)

[1] もちろん、最低限の共通の解釈を仮定しなければこれは不可能である。したがって正確にいうならば、最低限の共通の解釈によって関係付けられるということである。



UICT: User Interface Concept Translator
APICT: Application Interface Concept Translator

Figure 8.2: 論理的枠組みによるインテリジェントCADの構成

に用いられる。核システムは演繹、アブダクション、サーカムスクリプション、メタレベル推論などの各種の推論を論理的枠組みの中で実行することにより、概念レベルでの設計対象および設計過程に関する推論を行なうことができる。また、サブシステムの利用方法も知識として書かれているので、知識の利用という形で、サブシステムの利用を行なう。

8.2.3 サブシステムの統合

各サブシステムは論理的枠組みによって記述されるわけではない。しかし、その核システムとのインタフェースは論理式である。核システム側からみたとき、各サブシステムは論理式を入力されると、適当な論理式の真偽を決定して出力するシステムである。すなわち、論理的世界において、各サブシステムは概念に対してそれぞれ独自の解釈をもっており、それにしたがった推論を行なっている。

このようにサブシステムを位置付けると、サブシステムの挙動を論理式で記述することも可能になる。例えば、サブシステムの入出力の記述は論理式でなされるので、それを知識として再利用することも可能になる。また、場合によってサブシステムの

挙動一般を論理式で近似的に記述することが可能になる。このとき、核システムではその論理式を用いてサブシステムをシミュレートすることが可能になる[2]。

個々のサブシステムはいくつかの概念を利用する。サブシステムが用いる概念間の関係を記述することで、サブシステム間の関係を記述することができる。また、核システムとサブシステムが用いる概念が異なる場合も、論理式を用いて概念間に関係をつける。すなわち、サブシステム間、およびサブシステムと核システム間の構造を論理式を用いて記述する。

このようにサブシステムを核システムの中で記述することで、条件の(3)、(4)を満たすことができる。すなわち、システムの構造を知識化することでシステムはシステムの要素を容易に付け加えることもできるし、システム要素間の関係を動的に変えることができる。また、サブシステム間の関係やさらにはサブシステムの挙動を知識として記述することにより、核システムは個別の専門領域だけでなく、それらの関係を含めた全体を管理できることができる。

8.2.4 ユーザインタフェースの統合

また、ユーザも概念を通じて情報の伝達を行なう。ユーザインタフェースはユーザからの各種の方法によって入力された情報を概念として核システムに渡す。また、核システムの概念を変換してユーザに提示する。すなわち、ここでのインタフェースはデータの受渡しではなく、概念の受渡しを行なう概念インタフェースであることが必要である。インタフェースで用いられる概念が核システムの設計対象／過程記述用の概念と異なる場合、その変換は論理式で記述される。また、ユーザの持つ概念間の関係もはやり論理式によって表現される。これにより、ユーザのもつ概念構造がシステムに反映され、ユーザに適合して柔軟に変化するシステムを構成することができる。また、ユーザの概念構造に対する知識が増えたとき、それはユーザの反応をシミュレートする働きをする。核システムから見た場合、ユーザインタフェースは、ユーザの意図、知識による概念の解釈を持つといえる。

[2]これは meta-level inference の一種と捉えることができる。しかし、ここでの意味は meta-level ではなく、むしろ multi-interpretation inference である。

8.3 設計記述言語と推論

前節では、概念をキーとした統合化CADの構成について述べた。その核システムは概念の記述を基本とする設計過程・設計対象記述言語で記述されるシステムである。本節では、これまで議論してきた設計過程の論理的なモデルを元にした場合、このようなCADのための言語に必要とされる機能はどのようなものであるかについて議論する。

8.3.1 インテリジェントCAD核システムの構成

本論文で議論してきた設計過程の形式化では設計を設計対象自身に関する推論(対象レベルの推論)とその推論の進行手順などに関する推論(行為レベルの推論)に分けて、その相互作用として設計をモデル化した。そのモデルに基づくCAD核システムの構成を図8.3に示す。システムは基本的には二つのワークスペースに対して動作する。ひとつは対象ワークスペースであり、設計対象の記述と利用されている対象の性質に関する知識がここに保持される。このワークスペースはメタモデル機構が管理して整合性を保つ。二つ目は過程ワークスペースであり、ここには設計の状態が保持される。設計の状態とは、現在どのような対象レベルの推論を行なっているか、あるいはその履歴、設計段階の記述などである。これらはコンテクストとして管理される。このワークスペースは過程マネージャが管理する。この二つのワークスペースにある情報に対して各種の推論が行なわれる。対象ワークスペースに対しては、対象の性質に関する知識を用いて、対象の挙動の推論や詳細化のための推論などが行なわれる。これはシステムとしては、演繹推論やアブダクション、帰納推論などにより実現される。過程ワークスペースに対しては、設計過程に関する知識を利用して、次の対象レベルへの操作のための推論などが行なわれる。

設計対象の記述は実際の設計においてはそれ自身問題である。すなわち、対象の記述というのは多様かつ多視点的である。すなわち、対象のもつ性質や属性は対象によって多種多様であると同時にその利用の方法によっても変化する。したがって一様な方法をもって各対象を記述することは困難である。そこで、多様な視点からの記述を許容しかつそれら間の整合性を保つ機構が必要である。この問題

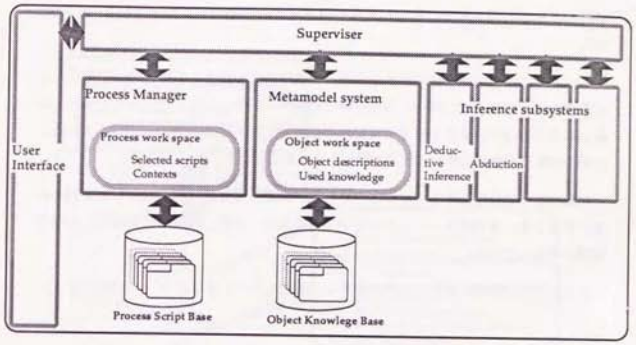


Figure 8.3: インテリジェントCAD核システムの構成

に対してメタモデルという統合的モデルを用いることで解決をすることができる[Kiriyama91][Tomiyama89]。すなわち、対象記述の記述はメタモデル機構によって管理維持される。

8.3.2 対象記述スキーマ

対象は「存在」と存在がもちうる「性質」のふたつで記述される。対象はそれ自身は存在を示す指示子(identifier)であり、それ以外の意味を持たない。これに対して、「性質」は名前付けを含む全ての対象の属性やいわゆる性質を記述する。性質と存在の組によって対象は記述され、それが外部から見た時の概念となる。

8.3.3 対象性質知識ユニット

対象性質知識とは対象の性質や性質間に成立する関係である。この対象性質知識には、物理法則などや工学的な法則などの原理が記述される。また部品や機械の挙動や

機能などもが記述される。さらに対象の構造もこの知識として記述される (図8.4参照)。

ここでは対象自身は存在を示すのみで、一切の性質・属性をもたない。したがって対象の性質の記述とは、同一の存在が持つ性質間の関係として記述される。また、対象の構造も部分や全体などを示す複数の存在が持つ性質間の関係として記述される。原理、法則は異なる対象の性質間の関係として記述される。

対象性質知識は関連するものがまとめられ、対象性質知識ユニットとして管理される。すなわち、各知識ユニットには、ひとつの観点、分野、領域などを指定した時の知識が含まれている。

この知識は設計者に対しての自由度から、3層に分けられる。

1. コア層
   コア層の知識は原理的知識であり、設計者にとっては変更できない不変の知識である。ここには物理法則などの一般的知識、あるいは各分野ごとの領域依存の法則などが含まれる。この層の知識はシステム設計時あるいはシステムを特定の設計分野用に特化するときに準備される。

2. アプリケーション層
   アプリケーション層の知識は設計者の経験的知識などであり、設計を繰り返すことにより、変更されうるものである。ここには個々の設計対象の構造や性質の記述が含まれる。部品などのデータベースもこの層の知識としてアクセスされる。概念的な対象の構造や性質は物理フィーチャー [Tomiyama89] などを利用して記述される。

3. データ層
   データ層の知識は設計を行なう過程で、新たに仮定したり、用意した性質間の関係、あるいは結果としてできた対象の構造などが記述される。この層の知識は設計を行なったときの対象ワークスペースから導入される。このレベルの知識は一般化されることにより、アプリケーション層の知識になる。

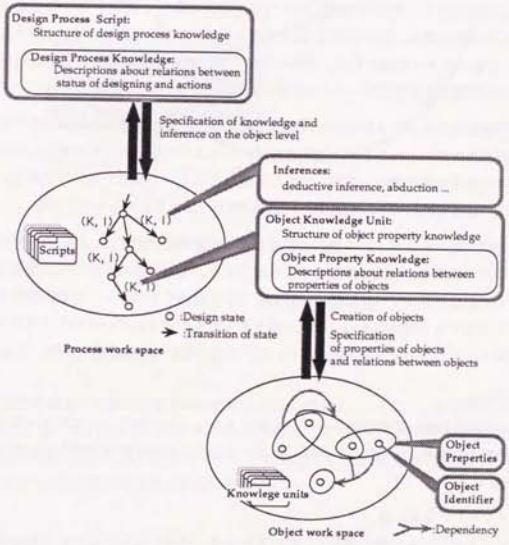


Figure 8.4: 二つのレベルの知識と推論

8.3.4 設計過程知識スクリプト

設計過程知識とは、設計の状態と次に行なう動作の関係を記述するものである。個々の設計過程知識は、設計の状態(設計対象の記述、設計の履歴)に対して行なうべき動作(対象レベルの知識の設定、推論の実行、設計対象の直接操作および過程知識スクリプトの選択などの過程レベルでの動作)を記述する(図8.4参照)。

設計過程知識スクリプトはひとつの目的に対して設計過程知識をまとめたものであり、手続き的に用いられるものと宣言的に用いられるもののどちらかである。手続き型のものはある対象の操作、対象レベルの推論を手順として記述するものである。宣言型のものはある目的に対して関連する設計過程の知識をまとめたものである。

設計過程知識スクリプトは一般的に利用できる問題解決手順から、特定の分野の設計のための手順まで、一般性の異なるものが含まれ、また特定の状況での問題解決の手がかりの記述や利用すべき知識の指摘のような小規模なものから、定型的設計のスクリプトのような大規模なもの、規模の異なるものまで含まれる。このスクリプトも設計者にとっての自由度という観点から、以下のようなレベルに分けられる。

1. コア層
   コア層の知識は設計者にとっては変更できないものであり、ここには一般的問題解決手順や、一般的設計手順、あるいは確立している評価手順などが含まれる。
2. アプリケーション層
   アプリケーション層の知識は、設計を行なう、あるいは繰り返すことにより変更されうるものである。この層の知識としては分野や会社などの標準的な設計手順や設計者が経験的に得た設計方法や設計手順、過去の設計例などが含まれる。例えば、既にある設計手順などは実際に設計にそのまま適用可能なことは少なく、修正をしながら利用されるが、その修正はもとの手順の変更となる。
3. データ層
   データ層の知識とは、実際に行なった設計の履歴であり、過程ワークスペースから導入される。このレベルの知識は一般化、構造化を行なうことにより、アプリケーション層の知識となる。

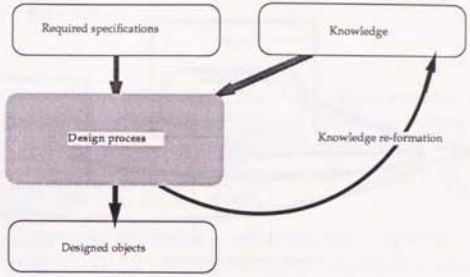


Figure 8.5: 設計の二つの役割

8.3.5 システムの基本動作

設計は新たな対象の創成と同時に知識の自己構造化という二つの作用がある(5章参照)。すなわち、設計はこれまで得ている知識をもとに設計を行なうが、設計はまた知識自体を変化させ、より設計に向くように構造化を行なう(図8.5参照)。システムはこの二つの作用を同時または別個に支援する。

対象創成作用

これは対象ワークスペースに設定された要求仕様を実現する対象記述をえる過程とし実現される。すなわち、対象性質知識によって要求仕様である性質を導出できるような対象をみつけることである。

その過程の実現のひとつの方法は過程知識スクリプトを利用して手続き的に構成することである。すなわち、適当なスクリプトを利用することで、どの対象知識を用いてどのような推論を行なえばよいかの指示、あるいは対象の直接操作が示される。利用されるスクリプトはひとつとは限らず、その場合いくつかのスクリプトが順にあ

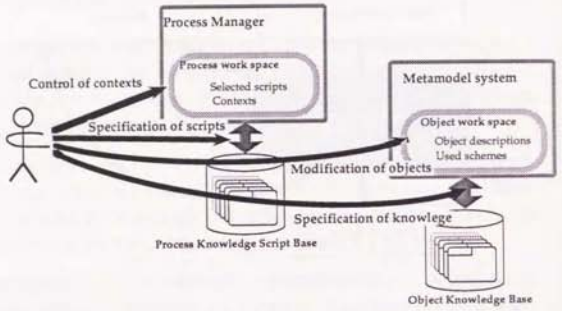


Figure 8.6: システムと設計者の関係

るいは階層的に呼び出されることになる。適用するスクリプトはスクリプトに記述している設計の状態からシステムが探してくる。適用可能性があるスクリプトが見つけられなかった場合は、ユーザ(設計者)が指定して、スクリプトを実行させる。また、ユーザは適宜、コンテクトを変更する。

システムあるいはユーザが指定したスクリプトは対象レベルでもちいるべき知識、推論あるいは対象の直接操作を指示するが、それだけでは設計が完了するとは限らない。その場合、ユーザによる対象レベルへの指示が行なわれる。ユーザは観点や注目する対象などを指定することにより、対象ワークスペースに新たな知識を導入する。そして、用意された推論を適当に行なうことで必要な情報を得る。また、設計対象を直接操作して、対象を詳細化する。

この過程における設計者の介在を図8.6にまとめる。

知識の自己構造化作用

設計に用いられる知識は設計を行なう、あるいは繰り返すことにより、より設計に適切な構造に変化する。この作用は対象の創成作用と同時あるいは別の動作として行なわれる。すなわち、設計時に、新しい対象の記述をつくると同時に、そこで用いられた知識を再構成するやり方と、設計とは別に知識を編集・再構成する作業を行なうことで、知識を変更するやりかたである。

設計過程においては、既存の知識だけでは設計を完了することはできない。既存の知識の訂正や拡張をすることで設計は進められている。この行為はシステムあるいはユーザによって行なわれる。例えば circumscription は知識を例外を含んだ形に再構成を行なう。システムはこのような知識自身を拡張する推論を提供する必要がある。また、ユーザによる知識の拡張や修正も設計中に行なわれる。これらの修正・拡張は過程知識、対象知識の各アプリケーション層に対して行なわれる。

また、設計を終えると、単に設計対象の記述だけではなく、その設計にいたるまでの情報が残される。これらを利用することで新しい知識や既存の知識の拡張を行なうことができる。設計終了時に残される情報は、対象レベルでは各性質を導出するに必要な知識(依存関係)、利用した対象知識ユニットであり、過程レベルでは利用した過程知識スクリプトや対象レベルへの操作や推論の履歴である。これらはデータ層の知識として保持されるが、一般化や構造化をすることでアプリケーション層の知識となる。

例えば、ある対象を設計するのに必要な対象知識や対象知識ユニットを集め、その対象設計のために特化した新しいユニットをつくることができる。あるいは、設計履歴を新たなスクリプトとすることで、その対象に関する手順を示すスクリプトをつくることができる。

より一般性のある知識を獲得するには、対象や履歴の例を抽象化や構造化をする必要がある。このためには、いくつかの例から一般化を行なう帰納推論、履歴から分節したり、特徴を発見するような知識獲得推論を用意する必要がある。

また、この作業はユーザによる知識編集としても行なわれる。

8.3.6 基本動作による多様な設計方法の実現

以上の基本動作と知識を用いて多様な設計過程を実現することができる。

1. 再設計
   これは前にデータ層にある設計した対象に関する情報を利用して行なう。設計の履歴や性質の依存関係などが残されているので、その設計過程を再現して必要なところを修正して、新たな対象を得る。

2. パラメータ設計
   この型の設計では設計手順は過程スクリプトとして記述されているので、スクリプトを実行することで設計の手順が生成される。設計者は欠けている情報を入力するだけで、対象記述が生成される。

3. 定型的設計
   この型の設計はいくつかの過程スクリプトの組合せで実現される。しかし、全ての手順が生成されるわけではないので、設計者が適宜スクリプトを選んだり、対象の直接操作や適当な推論を指示することで、設計を進める。

4. 編集設計
   既にある部品や部分を利用しながら新しい機能をもつ対象をつくる設計では、可変層の対象知識ベースの知識を主に利用して対象レベルで設計を進める。すなわち、各部分の構造や性質が記述されている知識ユニットを組み合わせて全体を構成し、その全体に対して対象レベルの推論を行なうことで、詳細化を進める。既存部品の設計に関しては過程スクリプトを利用して設計を行なう。

5. 新規設計
   まず、原理原則的対象知識(コア層の対象知識)をもとに挙動の推論を行ない、それを実現する構造を順次新たに作っていく。既に実現されている機能や挙動をみい出されればそれを実現する部分の記述を利用する。

8.4 インテリジェントCADの実現可能性

ここでは、前節で述べたインテリジェントCADシステムの実現の可能性について議論する。

8.4.1 インテリジェントCADのプロトタイプとしての設計シミュレータ

前節ではインテリジェントCADシステムの核言語および推論システムの仕様について述べた。本論文で示した設計シミュレータ、設計過程を推論システムでシミュレートするために開発したものであり、小規模で限定された知識を前提に疑似的な設計過程をつくるしたシステムであり、これ自身は新しい設計ができるわけではないので、本来の意味でのCADシステムではない。しかし、設計支援のための推論を実現しているものであるので、設計シミュレータをインテリジェントCADシステムのプロトタイプとしてみることができる。このとき、設計シミュレータはインテリジェントCADシステムの持つべき機能のいくつかを実現している。すなわち、以下のことがいえる。

- 対象レベルと行為レベル(過程レベル)という二つのレベルによる推論を統一的枠組みで実現した。
- アブダクション、演繹などの基本推論で設計が進行させることができることを示した。
- 依存関係管理と多重世界機構により、設計過程のコンテクストおよび履歴を管理する方法を示した。
- サーカムスクリプションにより知識の自己構造化の一方法を示した。

以上のように設計シミュレータはインテリジェントCADシステムの基本的な枠組みを提示しており、インテリジェントCADシステムのひとつの方向性を提示する有用なプロトタイプであるといえる。

8.4.2 インテリジェントCADのための要素技術の動向

前項で示したように設計シミュレータはインテリジェントCADが持つべき機能の一部を既に実現している。しかし、インテリジェントCADは前節で示した機能を統合的に提供するものでなければならない。そこで、このために必要なこの他の要素技術についても考察することが必要である。

- 統合的対象表現
  設計対象の記述を統合的に管理する問題については [Tomiyama89][Kiriyama91] によるメタモデル理論において議論され、メタモデルシステムという形でそのプロトタイプを実現している。現在は、概念的なレベルでの記述であるので、実際の設計の支援のためには、より詳細なレベルでの記述などに対応することが期待される。

- フィーチャー・ベース
  設計の知識は多種多様であるので、知識の記述方式を決めるのは困難な問題である。その記述方式のひとつの方法としてフィーチャーによる記述が考えられる。フィーチャーについては現在、多くの研究があり (例えば [Libardi86]、[Finger90])、今後の進展が期待される。

- 大規模知識ベース
  設計に関係する知識は極めて莫大であるので、大規模知識ベースが必要となる。大規模知識ベースは現在、いくつかのところで研究が行なわれており、実際に知識ベースの構築が図られている (例えば [Guha90])。

- 知識獲得・知識の自己構造化推論
  設計時に動的に知識を獲得するあるいは構造化する推論が必要である。帰納推論 (例えば [Angluin83] など) や類推 (例えば [Arikawa87][Haraguchi86])、事例ベース推論 (例えば [Kolodner88]) などの推論がこのために利用可能であると思われる。

このように必要な要素技術は研究中あるいは実現可能状態になりつつある。したがって今後はこれらの各分野の研究成果を統合することにより、インテリジェント

CADの開発が可能になると思われる。

8.5 まとめ

本章では、本研究で行なってきた設計過程の形式化の議論をもとに、インテリジェントCADのあるべき姿について述べた。この中で、インテリジェントCADは概念をキーとする統合化システムであり、システム構造の知識化により、柔軟かつ知的な設計者支援が行なえることを示した。さらに、そこで用いられる知識は設計を通じて構造化を行なっていくものであることを示した。

第9章

結言

本章では、本論文をまとめ、結論および今後の課題と展望について述べる。

9.1 結論

インテリジェントCADシステム開発の基礎として設計の研究は不可欠である。その中でも設計過程の解明はインテリジェントCADの中核として重要であるにもかかわらず、これまで大きな発展はみられていなかった。本研究では設計過程を総合的に分析して、インテリジェントCADのための基礎理論を提出することが目的であった。

この目的に対して、以下の研究を行なった。本研究ではまず認知的方法による分析から設計過程の特徴・性質を明らかにした。その結果を踏まえ、論理的枠組みによる設計過程の形式化を行ない、設計過程の論理的モデルを提案した。さらに、このモデル化に基づくCADの機能・構成について議論し、この設計過程モデルがインテリジェントCADシステム構築の基礎となりうることを示した。

具体的には、まず設計過程の分析手法としての設計実験法を確立した。設計観察のためのプロトコル解析の方法を具体的に示すとともに、図面の描画過程を計算機を利用して自動的に収集するシステムを実現した。また、プロトコル解析を利用して、設計過程の認知的モデルを提案した。設計は設計サイクルと呼ばれる問題解決過程の繰り返しとしてモデル化され、さらにそこで用いられる知識を収集・分析した。また、概念ネットワークを抽出して、設計者の概念構造とその変化を示した。

次に設計過程を論理的枠組みを用いて形式化して、さらに計算可能なモデルとして提案した。設計は論理におけるアブダクションとして定式化され、設計過程はアブダクションと演繹を繰り返すことにより設計記述を詳細化していく過程として示した。さらにアブダクションと演繹にサーカムスクリプションとメタ推論を加えた設計過程モデルを示した。また、多重世界における部分意味論によって、対象の記述、要求仕様の記述、知識の記述が統合的にできることを示した。さらに、この論理による設計過程モデルは計算可能であることを示し、設計シミュレータと呼ばれるシステムとして計算機上に実現した。

9.2 課題と展望

本研究では、設計の一側面である設計過程のみを研究対象とした。それだけでも、十分極めて困難な課題ではあるが、設計は設計対象あってこそ成り立つものである。本研究での成果は設計対象の研究と合わせて、ひとつの理論となるとき、真の意味での設計の理論になりうるものである。したがって、対象理論との統合が望まれる。

また、認知的方法による分析はより多く、より体系的に行なわれることが望まれる。ここで議論したような精緻な実験を行なうには、多大な時間と手間、そして技術者の協力が不可欠であり、簡単なことではない。しかし、設計の科学という意味では今後、より一層の積み重ねが必要であると思われる。

設計の論理による形式化は、本論文では始まったばかりであり、完結したとはいえない。形式化の問題では、アブダクションの実現可能性という創造に本質な問題には触れていない。また、対象記述に対する考察、行為のレベルに関する考察などが必要である。推論としては大規模な知識を扱う推論、事例に用いた推論など、さらに設計に現れる多様な推論を採り入れて行かなければならない。

また、知識の自己構造化は知的システム構築に重要な問題であり、今後理論的な側面、システム的な側面の両方から取り組んでいく必要がある。

本研究で示した設計シミュレータは単に設計を模倣するシステムということだけでなく、推論システムとして多くの可能性があることがわかった。今後はインテリジェントCADのプロトタイプとして、改良・発展させていく必要がある。

謝辞

指導教官である吉川弘之教授には、卒業論文のとき以来、6年にわたって指導を賜わりました。本研究は、吉川教授の設計研究に対する熱意と慧眼に動機付けられたものであり、多くの点で教授からの適切な助言により研究を発展させることができました。また、単に研究上の問題にとどまらず、研究生活のやり方から物事の考え方まで多くの面で、深く感化されるものがありました。ありがとうございました。

冨山哲男助教授からは、常に適切な助言を頂いたことに感謝します。冨山助教授からの助言と議論により本論文のテーマを深めることができ、多くの成果を出すことができました。また、計算機環境を始めとする研究室の環境、国際学会を含む多くの学会参加など、この上のない研究環境を作って頂いたことにも深く感謝します。

本論文の査読し有益なコメントをしていただいた大須賀節雄教授（東京大学先端科学技術研究センター）、中島尚正教授（東京大学工学部機械工学科）、木村文彦教授（東京大学工学部精密機械工学科）、新井民夫教授（同）には感謝いたします。

オランダCWIのPaul Veerkamp氏には、共同の論文の書く上で議論を行ない、本論文をまとめる上での貴重な意見となりました。また、University of EdinburghのAart Bijl教授の論理に関する見識は、本論文での論理の利用を考える上で大きな参考となりました。ここに感謝の意を表します。

三田工業株式会社の皆様には設計実験に協力して頂きました。特に研究の主旨を理解して、このような機会を作って頂いた三田順啓社長には深く感謝の意を表します。また、下村芳樹氏には、吉川研究室研究員当時から多くの議論を行ない研究上の参考になっただけでなく、設計実験にあたっては折衝等多くの場面で骨を折って頂きました。また、設計実験に参加された、重村豊氏、入江洋一郎氏、谷口正美氏氏、山本治

男氏、小牧進氏、水谷尚樹氏、明石正勝氏、狩野篤氏には感謝いたします。

吉川冨山研究室の先輩、同輩、後輩の方々には、研究活動を始めとする公私共々でお世話になりました。中でも設計過程グループとして一緒に研究を行なったメンバーには深く感謝します。勢川浩之君、石原啓君、松木淳一君、志田篤君は設計実験の実施とその結果の分析という多大な労力を必要とする研究で貢献してくれました。また、濱田進君の対話式設計実験ツールの開発は設計実験の新しい方法論を考える上で大きな参考になりました。馬場重郎君、吉岡真治君は設計シミュレータの開発の点で多大な貢献がありました。また、林千登君、河合浩之君との論理的形式化の問題についての議論は本論文を書く上で大きな刺激となりました。

また、桐山孝司君、薛徳意君との議論はインテリジェントCADの構成に関しての考察に大きな参考となりました。ここに感謝の意を表します。

最後に、研究室での活動を支えてくれた、技官の小野里雅美（旧姓熊沢）さん、碇山みち子さん、秘書の平山雅子さん、荻野まどか（旧姓鈴木）さん、下村貴子（旧姓奈良）さん、荒武由香さん、岡庭美樹さん、高井正子さんに感謝します。

参考文献

[Adelson85] B. Adelson and E. Soloway: The role of domain experience in software design, *IEEE Transactions of Software Engineering*, Vol. SE-11, No. 11, pp.1351–1360 (1985).

[Adelson89] B. Adelson: Cognitive research: uncovering how designers design; cognitive modeling: explaining and prediction how designers design, *Research in Engineering Design*, Vol. 1, pp.35–42 (1989).

[Akman90] V. Akman, P. ten Hagen, and T. Tomiyama: A fundamental and theoretical framework for an intelligent CAD system, *computer-aided design*, Vol. 22, No. 6, pp.352–367 (1990).

[Angluin83] D. Angluin and C. Smith: Inductive Inference; Theory and methods, *Computing Surveys*, Vol. 15, pp.237–269 (1983) （大谷木重夫訳：帰納的推論 — 理論と方法 —, bit 別冊, コンピュータ・サイエンス, pp. 107-135, 共立出版 (1985)）.

[Anzai89] 安西祐一郎: 初等物理学の問題解決における図表現の学習について, 日本認知科学会学習と対話分科会, Vol. SIGLAL89-2, pp.8–15 (1989).

[Anzai90] Y. Anzai: Learning and use of representations for physics, In K. A. Ericsson and J. Smith(eds.): *Studies in Expertise: Prospects and Limits* (1990).

[Arbab90] F. Arbab and B. Wang: Reasoning about geometric constraints, In H. Yoshikawa and T. Holden(eds.): *Intelligent CAD, II*, pp. 93–108, North-Holland, Amsterdam (1990).

[Arikawa87] 有川節夫: 帰納推論と類推によるプログラムの合成, 人工知能学会誌, Vol. 2, No. 3, pp.43–49 (1987).

[Asami85] 浅見直樹: 設計における知識構造, 東京大学工学部精密機械工学科卒業論文 (1985).

[Blamey86] S. Blamey: Partial logic, In D. Gabbay and F. Guenthner(eds.): *Handbook of Philosophical Logic*, volume III, chapter III.1, pp. 1–70, D. Reidel Publishing Company (1986).

[Bucciarelli88] L. Bucciarelli: An ethnographic perspective on engineering design, *Design Studies*, Vol. 9, No. 3, pp.159–168 (1988).

[Chan90] C. Chan: Cognitive processes in architectural design problem solving, *Design Studies*, Vol. 11, No. 2, pp.60–114 (1990).

[Cox86] P. Cox and T. Pietrzykowski: Causes for events: their computation and applications, In *Lecture Notes in Computer Science 230*, pp. 608–621, Springer-Verlag, Berlin (1986).

[de Kleer86] J. de Kleer: An assumption-based TMS, *Artificial Intelligence*, Vol. 28, pp.127–162 (1986).

[Dixon87] J. Dixon: On research methodology towards a scientific theory of engineering design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, Vol. 1, No. 3, pp.145–157 (1987).

[Eckersley88] M. Eckersley: The form of design processes: a protocol analysis study, *Design Studies*, Vol. 9, No. 2, pp.86–94 (1988).

[Ericsson80] K. Ericsson and H. Simon: Verbal reports as data, *Psychological Review*, Vol. 87, No. 3, pp.215–251 (1980).

[Ericsson85] K. Ericsson and H. Simon: *Protocol Analysis: Verbal Reports as Data*, The MIT Press, Cambridge, MA (1985).

[Faltings87] B. Faltings: Qualitative Kinematics in Mechanisms, In *Proceedings of Eleventh International Joint Conference on Artificial Intelligence (IJCAI-87)*, pp. 436–442 (1987).

[Finger85] J. Finger and M. Genesereth: RESIDUE: A deductive approach to design synthesis, Technical report stan-cs-85-1035, Stanford University (1985).

[Finger89] S. Finger and J. Dixon: A review of research in mechanical engineering design. part I: descriptive, prescriptive, and computer-based models of design processes, *Research in Engineering Design*, Vol. 1, No. 1, pp.51–67 (1989).

[Finger90] S. Finger and S. Safier: Representing and Recognizing Features in Mechanical Designs, In J. Rinderle(ed.): *Design Theory and Methodology (DTM '90)*, pp. 19–26, ASME (1990).

[Fukuda89] 福田収一: 溶接構造設計エキスパートシステム, コンピュートロール, Vol. 25, pp.112–117 (1989).

[Goel89] V. Goel and P. Pirolli: Motivating the notion of generic design within information-processing theory: the design problem space, *AI magazine*, Vol. 10, No. 1, pp.18–47 (1989).

[Goor75] A. Goor and R. Sommerfeld: A comparison of problem-solving processes of creative students and non-creative students, *Journal of Educational Psychology*, Vol. 67, pp.495–505 (1975).

[Guha90] R. Guha and D. Lenat: Cyc: a midterm report, *AI Magazine*, Vol. 11, No. 3, pp.32–59 (1990).

[Habraken88] H. Habraken and M. Gross: Concept design games, *Design Studies*, Vol. 9, No. 3, pp.150–158 (1988).

[Hamada89] 濱田進, 武田英明, 冨山哲男, 吉川弘之: 機械設計における図形情報に着目した設計行為の分析, 第5回ヒューマン・インタフェース・シンポジウム, pp. 197–200, 計測自動制御学会 (1989).

[Hamada90] 濱田進: 図形情報を用いた設計行為の分析, 東京大学大学院工学系研究科（精密機械工学専攻）修士論文 (1990).

[Handa88] 半田剛一, 石崎俊: 概念階層構造とコネクショニストアプローチによる重要概念の抽出, 情報処理学会第36回全国大会, pp. 1685–1686 (1988).

[Haraguchi86] 原口誠, 有川節夫: 類推の定式化とその実現, 人工知能学会誌, Vol. 1, No. 1, pp.132–139 (1986).

[Hasida87] K. Hasida, S. Ishizaki, and H. Isahara: A connectionist approach to the generation of abstracts, In *Natural Language Generation*, pp. 149–156, Martinus Nijhoff Publishers (1987).

[Hayashi89] 林千登: 非標準論理を用いた設計過程モデルの研究, 東京大学大学院工学系研究科（精密機械工学専攻）修士論文 (1989).

[Hayes79] P. Hayes: The logic of frames, In D. Metzing(ed.): *Frame Conceptions and Text Understanding*, pp. 46–61, Walter de Gruyter, Berlin, New York (1979).

[Hirst89] G. Hirst: Ontological assumptions in knowledge representation, In R. Brachman, H. Levesque, and R. Reiter(eds.): *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR89)*, pp. 157–169, Morgan Kaufmann Publishers, Inc., Toronto, Ontario, Canada (1989).

[Hubka77] V. Hubka: *Theorie der Konstruktionsprozesse*, Springer-Verlag, Berlin (1977).

[Hubka87] V. Hubka: *Principles of Engineering Design*, Springer-Verlag, Berlin (1987).

[Hubka88] V. Hubka and W. Eder: *Theory of Technical Systems*, Springer-Verlag, Berlin (1988).

[Ishihara89] 石原啓: 設計実験による設計知識の分析, 東京大学工学部精密機械工学科卒業論文 (1989).

[Kawamori89] 川森雅仁: 可能世界から情報へ, 1989年度論理文法研究会チュートリアル (1989).

[Kiriyama91] 桐山孝司: 定性物理を用いた設計対象表現論, 東京大学工学部博士論文 (1991)（予定）.

[Kolodner88] J. Kolodner(ed.): *Proceedings of Case-Based Reasoning Workshop*, Florida, DARPA (1988).

[Kowalski79] R. Kowalski: *Logic for Problem Solving*, North Holland, Amsterdam (1979).

[Kumagai76] 熊谷卓: 自動化機構３００選, 日刊工業新聞社 (1976).

[Kurumatani90] 車谷浩一, 冨山哲男, 吉川弘之: 多変数定性空間を用いた物体の平面運動の推論, 人工知能学会誌, Vol. 5, No. 4, pp.449–461 (1990).

[Landman86] F. Landman: *Towards a Theory of Information: the Status of Partial Objects in Semantics*, Foris, Dordrecht (1986).

[Libardi86] E. Libardi, J. Dixon, and M. Simmons: Designing with features: Design and analysis of extrusions as an example, In *ASME Spring National Design Engineering Conference*, pp. 24–27, ASME (1986).

[Lifschitz85] V. Lifschitz: Computing circumscription, In *Proceedings of Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pp. 121–127, Los Angles, CA (1985).

[Lloyd84] J. Lloyd: *Foundations of Logic Programming*, Springer-Verlag, Berlin (1984).

[Matsuki90] 松木淳一: 設計における概念構造の分析, 東京大学工学部精密機械工学科卒業論文 (1990).

[Matubara87] 松原仁, 山本和彦: フレーム問題について, 人工知能学会誌, Vol. 2, No. 3, pp.266–272 (1987).

[McCarthy69] J. McCarthy and P. Hayes: Some philosophical problems from the standpoint of artificial intelligence, *Machine Intelligence*, Vol. 4, pp.463–502 (1969).

[McCarthy80] J. McCarthy: Circumscription — a form of non-monotonic reasoning, *Artificial Intelligence*, Vol. 13, pp.27–39 (1980).

[McDermott82] D. McDermott: Non-monotonic logic II, *Journal of ACM*, Vol. 29, No. 1, pp.33–57 (1982).

[Miyake85] 三宅なほみ: 理解におけるインターラクションとは何か, 佐伯胖(編): 理解とは何か, 東大出版会 (1985).

[Morris86] P. Morris and R. Nado: Representing actions with an assumption-based truth maintenance system, In *Proceedings of Fifth National Conference on Artificial Intelligence (AAAI-86)*, pp. 13–17, The American Association for Artificial Intelligence, Philadelphia (1986).

[Nagao83] 長尾真, 淵一博: 論理と意味, 岩波書店 (1983).

[Nagasawa89] 長澤勲, 伊藤公俊: インテリジェントCADと設計, コンピュートロール, Vol. 25, pp.9–18 (1989).

[Nakagawa87] 中川裕志, 森辰則: 論理型言語における Circumscription, 情報処理論文誌, Vol. 28, No. 4, pp.330–338 (1987).

[Nakashima71] 中島尚正: 自動設計の基本モデル, 日科技連 (1971).

[Nakashima83] 中島秀之: *Prolog*, 産業図書 (1983).

[Nakashima89] 中島裕生, 馬場富男: 油圧回路設計支援エキスパートシステム OHCS, コンピュートロール, Vol. 25, pp.106–111 (1989).

[Newell72] A. Newell and H. Simon: *Human Problem Solving*, Prentice-Hall (1972).

[Ohsuga85] 大須賀節雄: 次世代 CAD/CAM のための知識処理の応用, マグロウヒルブック (1985).

[Ohsuga89] S. Ohsuga: Toward intelligent CAD systems, *computer-aided design*, Vol. 21, No. 5, pp.315–337 (1989).

[Okamura85] 岡村貴句男: 絵でみるメカトロニクスのアイデア１００選（第１集）— 無段変速機活用編 —, オーム社 (1985).

[Okamura89] 岡村貴句男: 絵でみるメカトロニクスのアイデア１００選（第２集）— センサ・計測器活用編 —, オーム社 (1989).

[Poole88] D. Poole: A logical framework for default reasoning, *Artificial Intelligence*, Vol. 36, pp.27–47 (1988).

[Reiter87] R. Reiter and J. de Kleer: Foundations of assumption-based truth maintenance systems: Preliminary report, In *Proceedings of Sixth National Conference on Artificial Intelligence (AAAI-87)*, pp. 183–188, The American Association for Artificial Intelligence, Seattle, Washington (1987).

[Rodenacker70] W. Rodenacker: *Methodisches Konstruieren*, Springer, Berlin (1970).

[Roth82] K. Roth: *Konstruieren mit Konstruktionskatalogen*, Springer-Verlag, Berlin (1982).

[Rumelhart86] D. Rumelhart, J. McClelland, and P. R. Group: *Parallel Distributed Processing*, The MIT Press, Cambridge, MA (1986).

[Segawa88] 勢川博之: 設計実験の分析による設計過程の分析, 東京大学工学部精密機械工学科卒業論文 (1988).

[Shirai86] 白井賢一郎: 意味と推論 — データ意味論への招待 —, 数理科学, No. 279, pp.63–77 (1986).

[Simon73] H. Simon: The structure of ill structureed problems, *Artificial Intelligence*, Vol. 4, pp.181–201 (1973).

[Suh78] N. Suh, A. Bell, and D. Gossard: On an axiomatic approach to manufacturing and manufacturing system, *Journal of Engineering for Industry*, Vol. 100, pp.127–130 (1978).

[Suh90] N. Suh: *The Principles of Design*, Oxford University Press, New York, Oxford (1990).

[Suzuki90] H. Suzuki, H. Ando, and F. Kimura: Synthesizing product shapes with geometric design constraints, In H. Yoshikawa and T. Holden(eds.): *Intelligent CAD, II*, pp. 309–324, North-Holland, Amsterdam (1990).

[Takeda88] 武田英明: 1987 年度設計実験報告, (吉川冨山研究室内部資料) (1988).

[Takeda89] 武田英明, 石原啓: 1988 年度設計実験報告, (吉川冨山研究室内部資料) (1989).

[Takeda90] 武田英明, 濱田進, 松木淳一: 1989 年度設計実験報告, (吉川冨山研究室内部資料) (1990).

[ten Hagen87] P. ten Hagen and T. Tomiyama(eds.): *Intelligent CAD systems I: Theoretical and Methodological Aspects*, Springer-Verlag, Berlin (1987).

[Tomiyama85a] 冨山哲男: CAD 構成論, 東京大学工学部博士論文 (1985).

[Tomiyama85b] 冨山哲男, 吉川弘之: 一般設計学の展開 (第 1 報) — 概念空間のコンパクト化 —, 精密機械, Vol. 51, No. 4, pp.809–815 (1985).

[Tomiyama89] T. Tomiyama, T. Kiriyama, H. Takeda, and D. Xue: Metamodel: a key to intelligent CAD systems, *Research in Engineering Design*, Vol. 1, pp.19–34 (1989).

[Treur89] J. Treur and Y. Tan: A logical description of a bi-modular system for nonmonotonic reasoning with dynamic assumptions, Technical Report PE8907, Department of Mathematics and Computer Science, University of Amsterdam, Amsterdam (1989).

[Treur90] J. Treur: A logical framework for design processes, In P. ten Hagen and P. Veerkamp(eds.): *Intelligent CAD Systems III — Practical Experience and Evaluation*, Springer-Verlag, Berlin (1990) Forthcoming.

[Turner84] R. Turner: *Logics for Artificial Intelligence*, Ellis Horwood Limited, West Sussex, England (1984).

[Uchida86] 内田種臣 (編): パース著作集 2: 記号学, 勁草書房 (1986).

[Ullman87a] D. Ullman, L. Stauffer, and T. Dietterich: Preliminary results on an experimental study of mechanical design, In *Results from NSF Workshop on the Design Process*, pp. 145–188 (1987).

[Ullman87b] D. Ullman, L. Stauffer, and T. Dietterich: Toward expert CAD, *Computers in Mechanical Engineering*, Vol. 6, No. 3, pp.56–70 (1987).

[Ullman88] D. Ullman, T. Dietterich, and L. Stauffer: A model of the mechanical design process based on empirical data: a summary, In J. Gero(ed.): *Artificial Intelligence in Engineering Design*, pp. 193–215, Elsevier, Amsterdam (1988).

[Ullman90] D. Ullman, S. Wood, and C. Craig: The importance of drawing in the mechanical design process, *Computers and Graphics*, Vol. 14, No. 2, pp.263–274 (1990).

[Umeda90] Y. Umeda, H. Takeda, T. Tomiyama, and Y. Yoshikawa: Function, behaviour, and structure, In J. Gero(ed.): *Applications of Artificial Intelligence in Engineering V*, volume 1, pp. 177–194, Springer-Verlag, Berlin (1990).

[Veltman81] F. Veltman: Data Semantics, In J.A.G.Groenendijk, T. Janssen, and M.B.J.Stokhof(eds.): *Formal methods in the study of language*, pp. 541–565, Mathematisch Centrum, Amsterdam (1981).

[Waldron87] K. Waldron and M. Waldron: A retrospective study of a complex mechanical system design, In M. Waldron(ed.): *Results from NSF Workshop on the Design Process*, pp. 109–141 (1987).

[Weyhrauch80] R. Weyhrauch: Prolegomena to a theory of mechanized formal reasoning, *Artificial Intelligence*, Vol. 13, pp.133–170 (1980).

[Yamamoto89] 山本博樹, 高橋秀明, 天沼聡: プロトコル法の効果的な利用のための基礎的な考察, 日本認知科学会第6回大会発表論文集, pp. 60–61 (1989).

[Yonemori81] 米盛裕二: パースの記号学, 勁草書房 (1981).

[Yoshikawa77] 吉川弘之: 設計学研究, 精密機械, Vol. 43, No. 1, pp.21–26 (1977).

[Yoshikawa79] 吉川弘之: 一般設計学序説 — 一般設計学のための公理的方法 —, 精密機械, Vol. 45, No. 8, pp.906–912 (1979).

[Yoshikawa81a] H. Yoshikawa: General design theory and a CAD system, In T. Sata and E. Warman(eds.): *Man-Machine Communication in CAD/CAM, Proceedings of the IFIP Working Group 5.2 Working Conference 1980 (Tokyo)*, pp. 35–58, North-Holland, Amsterdam (1981).

[Yoshikawa81b] 吉川弘之: 一般設計過程, 精密機械, Vol. 47, No. 4, pp.405–410 (1981).

[Yoshikawa81c] 吉川弘之: 設計論の最近の動向, 精密機械, Vol. 47, No. 8, pp.907–912 (1981).

[Yoshikawa83] H. Yoshikawa: CAD framework guided by general design theory, In K. Bϕ and E. Lillehagen(eds.): *CAD Systems Framework, Proceedings of IFIP Working Group 5.2*, pp. 241–253, North-Holland, Amsterdam (1983).

[Yoshikawa89] H. Yoshikawa and D. Gossard(eds.): *Intelligent CAD, I*, North-Holland, Amsterdam (1989).

[Yoshikawa90] H. Yoshikawa and T. Holden(eds.): *Intelligent CAD, II*, North-Holland, Amsterdam (1990).

発表論文

投稿論文他（査読付）

[1] 武田英明, 冨山哲男, 吉川弘之. 知的 CAD のための設計過程の分析と論理による形式化. 精密工学会誌. (投稿中).

[2] H. Takeda, T. Tomiyama, and H. Yoshikawa. A logical formalization of design processes for intelligent CAD systems. In H. Yoshikawa and T. Holden, editors, *Intelligent CAD, II*, pp. 325–336. North-Holland, Amsterdam, 1990.

[3] H. Takeda, S. Hamada, T. Tomiyama, and H. Yoshikawa. A cognitive approach of the analysis of design processes. In *Design Theory and Methodology (DTM '90)*, pp. 153–160. The American Society of Mechanical Engineers (ASME), 1990.

[4] H. Takeda, P. Veerkamp, T. Tomiyama, and H. Yoshikawa. Modeling design processes. *AI Magazine*, Vol. 11, No. 4, pp. 37–48, 1990.

[5] T. Tomiyama, T. Kiriyama, H. Takeda, and D. Xue. Metamodel: A key to intelligent CAD systems. *Research in Engineering Design*, Vol. 1, pp. 19–34, 1989.

[6] Y. Umeda, H. Takeda, T. Tomiyama, and Y. Yoshikawa. Function, behaviour, and structure. In J.S. Gero, editor, *Applications of Artificial Intelligence in Engineering V*, volume 1, pp. 177–194, Berlin, 1990. Springer-Verlag.

講演論文他

[1] H. Takeda, T. Tomiyama, H. Yoshikawa, and P.J. Veerkamp. Modeling design processes. Technical Report CS-R9059, Centre for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands, October 1990.

[2] 武田英明, 勢川博之, 冨山哲男, 吉川弘之. 設計過程の分析と論理による形式化（第1報）－設計実験による分析－. 63年度精密工学会春季大会講演論文集, pp. 131–133, 明治大、神奈川, 1988.

[3] 武田英明, 冨山哲男, 吉川弘之. インテリジェントCADのための設計過程の論理による形式化. 第6回設計自動化工学講演会講演論文集, pp. 13–15, 東工大、東京, 1988. 日本機械学会、精密工学会.

[4] 武田英明, 冨山哲男, 吉川弘之. 知的CAD開発のための設計過程の論理による定式化. 昭和63年度人工知能学会全国大会（第2回）講演論文集, pp. 161–164, 学習院大、東京, 1988.

[5] 武田英明, 石原啓, 林千登, 冨山哲男, 吉川弘之. 設計過程の分析と論理による形式化（第2報）－設計知識の分析－. 1989年度精密工学会春季大会講演論文集, pp. 5–6, 千葉工大、千葉, 1989.

[6] 林千登, 武田英明, 冨山哲男, 吉川弘之. 設計過程の分析と論理による形式化（第3報）－サーカムスクリプションとアブダクションによるモデル化－. 1989年度精密工学会春季大会講演論文集, pp. 7–8, 千葉工大、千葉, 1989.

[7] 河合浩之, 武田英明, 林千登, 冨山哲男, 吉川弘之. 設計過程の分析と論理による形式化（第4報）－様相論理を利用したＴＭＳ－. 1989年度精密工学会春季大会講演論文集, pp. 9–10, 千葉工大、千葉, 1989.

[8] 武田英明, 林千登, 薛徳意, 冨山哲男, 吉川弘之. 知的CADのための設計シミュレーション. 第7回設計シンポジウム, pp. 34–36, 東大、東京, 1989. 日本機械学会、精密工学会. 1989年7月7日発表.

[9] 武田英明, 河合浩之, 冨山哲男, 吉川弘之. 知的CADのための様相論理と依存関係を利用した推論情報の管理. 1989年度人工知能学会大会（第3回）論文集, pp. 259–262, 学習院大、東京, 1989. 1989年7月24日発表.

[10] 濱田進, 武田英明, 冨山哲男, 吉川弘之. 機械設計における図面情報に着目した設計行為の分析. 第5回ヒューマン・インターフェース・シンポジウム, pp. 197–200, 京都, 1989. 計測自動制御学会. 1989年10月25–27日.

[11] 武田英明, 濱田進, 冨山哲男, 吉川弘之. 設計過程の分析と論理による形式化（第5報）－設計実験方法論－. 1990年度精密工学会春季大会講演論文集, pp. 275–276, 東工大、東京, 1990.

[12] 濱田進, 武田英明, 冨山哲男, 吉川弘之. 設計過程の分析と論理による形式化（第6報）－図形情報を中心とする解析－. 1990年度精密工学会春季大会講演論文集, pp. 277–278, 東工大、東京, 1990.

[13] 松木淳一, 武田英明, 冨山哲男, 吉川弘之. 設計過程の分析と論理による形式化（第7報）－設計概念の構造の分析－. 1990年度精密工学会春季大会講演論文集, pp. 279–280, 東工大、東京, 1990.

[14] 馬場重郎, 河合浩之, 武田英明, 冨山哲男, 吉川弘之. インテリジェントCADのための論理的推論と設計過程のシミュレーション. 1990年度精密工学会春季大会講演論文集, pp. 289–290, 東工大、東京, 1990. 1990年3月28日発表.

[15] 梅田靖, 武田英明, 冨山哲男, 吉川弘之. 機能、挙動、および、構造の関係について. 1990年度精密工学会春季大会講演論文集, pp. 307–308, 東工大、東京, 1990. 1990年3月28日発表.

[16] 武田英明, 冨山哲男, 吉川弘之. プロトコル解析を用いた設計過程の分析の方法について. 日本認知科学会第7回大会発表論文集, pp. 76–77, 九州工大、福岡, 1990. 1990年7月5日発表.

[17] 武田英明, 河合浩之, 冨山哲男, 吉川弘之. 設計過程の計算可能モデルに基づく設計シミュレーション. 1990年度人工知能学会（第5回）論文集, pp. 583–586, 学習院大、東京, 1990. 1990年7月26日発表.

[18] 河合浩之, 武田英明, 冨山哲男, 吉川弘之. CAD における知識表現の枠組み. 1990 年度精密工学会秋季大会講演論文集, pp. 1247–1248, 北海道大、北海道, 1990. 1990 年 9 月 30 日発表.

[19] 吉岡真治, 河合浩之, 武田英明, 冨山哲男, 吉川弘之. 多重世界を用いた設計過程のシミュレーション. 1991 年度精密工学会春季大会講演論文集, 早稲田大学、東京, 1991. 1991 年 3 月 26 日発表 (予定).

付録 A

部分意味論の定義・定理

定義 A.1 部分論理モデル M は $< D, F, V >$ からなり、D は空でない集合、F は各 n 引数関数 F に D^n から D への関数、V は各 n 項述語に対する D^n から $\{1, 0\}$ への部分関数である。

定義 A.2 部分論理モデル M における真理条件は以下の通りである。

1. 素論理式
   - (a) $M \models^+ A \Longleftrightarrow V(A) = 1$
   - (b) $M \models^- A \Longleftrightarrow V(A) = 0$
2. 否定
   - (a) $M \models^+ \neg A \Longleftrightarrow M \models^- A$
   - (b) $M \models^- \neg A \Longleftrightarrow M \models^+ A$
3. 連言
   - (a) $M \models^+ \phi \wedge \psi \Longleftrightarrow M \models^+ \phi$ かつ $M \models^+ \psi$
   - (b) $M \models^- \phi \wedge \psi \Longleftrightarrow M \models^- \phi$ または $M \models^- \psi$
4. 選言
   - (a) $M \models^+ \phi \vee \psi \Longleftrightarrow M \models^+ \phi$ または $M \models^+ \psi$
   - (b) $M \models_M^- \phi \leftrightarrow \psi \Longleftrightarrow M \models^- \phi$ かつ $M \models^- \psi$
5. 含意
   - (a) $M \models^+ \phi \rightarrow \psi \Longleftrightarrow M \models^- \phi$ または $M \models^+ \psi$
   - (b) $M \models^- \phi \rightarrow \psi \Longleftrightarrow M \models^+ \phi$ かつ $M \models^- \psi$
6. 同値

(a) $M \models^+ \phi \leftrightarrow \psi \iff M \models^+ \phi$ かつ $M \models^+ \psi$ であるか、 $M \models^- \phi$ かつ $M \models^- \psi$

(b) $M \models^- \phi \leftrightarrow \psi \iff M \models^+ \phi$ かつ $M \models^- \psi$ であるか、 $M \models^+ \phi$ かつ $M \models^- \psi$

7. 全称

(a) $M \models^+ \forall A(x) \iff$ 全ての $d \in D$ において $M \models^+ A(d)$

(b) $M \models^- \forall A(x) \iff$ ある $d \in D$ において $M \models^- A(d)$

8. 存在

(a) $M \models^+ \exists A(x) \iff$ ある $d \in D$ において $M \models^+ A(d)$

(b) $M \models^- \exists A(x) \iff$ 全ての $d \in D$ において $M \models^- A(d)$

定義 A.3 言語 L、領域 D のモデル M、M' において、次のような条件を満たすとき、M' は M の *extension* （$M' \sqsupseteq M$）であるという。L の任意の n 項関係 C において、D の任意の $e_0, \ldots, e_{n-1}$ において、

$$C^{M'}(e_0, \ldots, e_{n-1}) \sqsupseteq C^M(e_0, \ldots, e_{n-1})$$

であるとき。ただし、$C^M(e_0, \ldots, e_{n-1})$ は $C(e_0, \ldots, e_{n-1})$ のモデル M での値を指す（定義 *6.1*）。

定義 A.4 完全モデル $M_c =< D, F, V >$ とは、V が部分関数でないモデルである。

定理 A.1 いま、言語 L の任意の式を A とする。$M' \sqsupseteq M$ ならば $[A]^{M'} \sqsupseteq [A]^M$ である（定理 *6.1*）。

証明　[Turner84] pp–40 参照。 □

定義 A.5 モデル M のダイアグラム $D(M)$ とは、

$$D(M) = \{a | a \in A, [a]^M = t\} \cup \{\neg a | a \in A, [a]^M = f\}$$

である *[Treur89]*。

命題 A.1 P を閉論理式集合とするとき、P の古典論理モデルの集合を Γ とする。また、モデル I の成り立つ全てのリテラルを $A(I)$ とする。さらに、リテラルで P の古典論理での論理的帰結であるものの集合を $th(P)$ とする。このとき、

$$th(P) = \bigcap_{I \in \Gamma} A(I)$$

が成り立つ。

証明　論理的帰結の定義より、明らか。 □

定義 A.6 部分モデル M が論理式集合 P を満たすとは、任意の $\phi \in P$ において、$[\phi]_M = t$ のときである。

命題 A.2 P を閉論理式集合とする。P の古典論理でのモデルにおいて、そのモデルで成り立つ全てのリテラルをダイアグラムとする部分論理のモデルを考える。このとき、この部分論理のモデルにおいて、P を満たす。また、P を満たす全ての完全モデルは古典論理において P を満たす。

証明　定義より、古典論理のモデルに対応する部分論理のモデルは完全モデルである。このとき、全ての論理演算は古典論理と部分論理において同一である。したがって、古典論理において P のモデルであるものは、対応する部分論理モデルにおいて P を満たす。また部分論理の完全モデルにおいても P を満たすモデルは、古典論理の P を満たすモデルに対応する。 □

定義 A.7 ある部分論理モデル集合 Δ において、極小モデル M とは、$M' \in \Delta$ で、$M' \sqsubset M$ なる M' が存在しないものである。

定義 A.8 ある部分論理モデル集合 Δ において、最小モデル M_s とは、全ての $M' \in \Delta$ に対して、$M_s \sqsubset M$ であるものである。

定義 A.9 部分論理モデル M において、M の拡張であり、かつ完全モデルをあるものを拡張完全モデルと呼ぶ。

命題 A.3 部分モデル M の拡張完全モデル集合を Δ とすると、

$$D(M) = \bigcap_{M_c \in \Delta} D(M_c)$$

である。

証明　任意の $D(M_c)$ で $D(M) \subseteq D(M_c)$ なので、

$$D(M) \subseteq \bigcap_{M_c \in \Delta} D(M_c)$$

である。いま、

$$A \in \bigcap_{M_c \in \Delta} D(M_c)$$

かつ $A \notin D(M)$ なる A を仮定する。M は A を含まないので、A を含む完全拡張モデルと $\neg A$ を含む拡張完全モデルの両方を持つ。これは最初の条件に反する。したがって、

$$D(M) = \bigcap_{M_c \in \Delta} D(M_c)$$

である。 □

定理 A.2 P を閉論理式集合とする。P の古典論理での論理的帰結であるリテラルの集合を $th(P)$ とする。また、P を部分論理で満たす部分モデルの集合を Δ とすると、任意の $M \in \Delta$ において、$th(P) \subseteq D(M)$

証明　Δ の任意の要素 M において、P を満たすので、部分論理の単調性（定理 A.1）より、M の全ての拡張完全モデルも P を満たす。いま、全ての P を満たす完全モデルの集合を Δ とすると、A.3より、任意の P を満たすモデル M は、ある $\Delta' \subseteq \Delta$ が存在して、

$$D(M) = \bigcap_{M_c \in \Delta'} D(M_c)$$

また、$D(M_0) = th(P)$ なる M_0 を考えると、A.2と A.1より、

$$D(M_0) = \bigcap_{M_c \in \Delta} D(M_c)$$

が成り立つ。したがって、任意の $M \in \Delta$ において、$th(P) \subseteq D(M)$。 □

定理 A.3 P を閉論理式集合とする。P の古典論理での論理的帰結であるリテラルの集合を $th(P)$ とする。また、P を部分論理で満たす部分モデルの集合を Δ とすると、Δ に最小モデル M_s が存在するとき、$th(P) = D(M_s)$ である（定理 6.3）。

証明　いま、P を満たす完全モデルの集合を Δ とする。M_s の拡張完全モデルの集合を Δ' とすると、

$$D(M_s) = \bigcap_{M_c \in \Delta'} D(M_c)$$

また全ての $M_c \in \Delta'$ は P を満たす。M_s は P を満たす最小モデルなので、全ての P を満たすモデル M は $M_s \sqsubseteq M$。したがって、全ての P を満たす完全モデルは、M_s の拡張完全モデルである。したがって、$\Delta = \Delta'$。ここで、$D(M_0) = th(P)$ なる M_0 を考えると、A.2と A.1より、

$$D(M_0) = \bigcap_{M_c \in \Delta} D(M_c)$$

が成り立つ。したがって、$D(M_0) = D(M_s)$。よって、$th(P) = D(M_s)$。 □

定理 A.4 節集合 P において、古典論理での全ての論理的帰結であるリテラル（素論理式またはその否定）の集合を $th(P)$ とする。いま、$D(M) = th(P)$ なる部分論理モデル M、すなわち $th(P)$ をダイアグラムとする部分モデル M を考える。このとき、任意の $\phi \in P$ は $M(\phi) = t$ または $M(\phi) = u$ である。

証明　$\phi \in P$ が素論理式であれば、$\phi \in th(P)$。したがって、$\phi \in D(M)$。いま、ϕ を $B_1 \vee \ldots B_m \vee \neg A_1 \vee \ldots \vee \neg A_n$ とする。$M(\phi) = f$ と仮定すると、部分論理の真理値の定義より、$M(A_i) = t$ $(i = 1, \ldots, n)$ かつ $M(B_j) = f$ $(j = 1, \ldots, m)$。したがって、$A_i \in D(M)$ $(i = 1, \ldots, n)$。$\neg B_j \in D(M)$ $(j = 1, \ldots, m)$。したがって $A_i \in th(P)$ $(i = 1, \ldots, n)$ かつ $\neg B_j \in th(P)$ $(j = 1, \ldots, m)$。しかしこれは、P の節である $B_1 \vee \ldots B_m \vee \neg A_1 \vee \ldots \vee \neg A_n$ を満たさない。これは $M(\phi) = f$ に反する。したがって、任意の $\phi \in P$ なる ϕ で、$M(\phi) = f$ でない。 □

付録 B

データ意味論の定義・定理

同値の定義

定義 B.1 二つの論理式 ϕ, ψ に対して、弱等価（*weak equivalent*）$\cong$ であるとは、$\phi \Rightarrow \psi$ かつ $\psi \Rightarrow \phi$ であるときである。また、強等価（*strong equivalent*）$\equiv$ であるとは、$\phi \Rightarrow \psi$ かつ $\psi \Rightarrow \phi$ かつ $\neg\phi \Rightarrow \neg\psi$ かつ $\neg\psi \Rightarrow \neg\phi$ であるときである。

定理 B.1 任意の式 ϕ, ψ において、$\phi \equiv \psi$ であることと、任意の情報状態 s で $s \models^+ \phi$ と $s \models^+ \phi$ が同値であり、かつ $s \models^- \phi$ と $s \models^- \phi$ が同値であることに等しい。

証明　（左辺）$\Longrightarrow$（右辺）
任意の情報状態 s で $s \models^+ \phi \Rightarrow \psi$ かつ $s \models^+ \phi \Rightarrow \psi$ である。したがって、$s \models^+ \phi$ ならば $s \models^- \psi$ でない、かつ $s \models^+ \psi$ ならば $s \models^- \phi$ でない。すなわち $s \models^+ \phi \Longleftrightarrow \models^+ \psi$ である。同様にして $s \models^- \phi \Longleftrightarrow \models^- \psi$ もいえる。
（右辺）$\Longrightarrow$（左辺）
情報状態 s で $s \models^+ \phi$ と $s \models^+ \phi$ が同値であり、かつ $s \models^- \phi$ と $s \models^- \phi$ が同値であれば、$s \models^+ \phi$ かつ $s \models^- \phi$ とういうことはない。したがって任意の状態 s_0 において $s_0 \models^+ \phi \Rightarrow \psi$ である。さらに $s \models^- \neg\phi$ かつ $s \models^+ \neg\phi$ とういうことはないので、任意の状態 s_0 において $s_0 \models^+ \neg\psi \Rightarrow \neg\phi$ である。同様にして $s_0 \models^+ \psi \Rightarrow \phi$ と $s_0 \models^+ \neg\phi \Rightarrow \neg\psi$ もいえる。したがって定義より、$\phi \equiv \psi$ である。 □

以降、この定理を用いて、同値関係の式を証明する。

真理値の単調性

定義 B.2 ϕ が単調真（*T-stable*）であるとは、$(S, \sqsubseteq)$ の全ての $s \in S$ において $s \models^+ \phi$ ならば、全ての $s' \sqsupseteq s$ において $s' \models^+ \phi$ であるときである。

定義 B.3 ϕ が単調偽（*F-stable*）であるとは、$(S, \sqsubseteq)$ の全ての $s \in S$ において $s \models^- \phi$ ならば、全ての $s' \sqsupseteq s$ において $s' \models^- \phi$ であるときである。

定義 B.4 ϕ が単調 (*stable*) であるとは、ϕ が単調真かつ単調偽であるときである。

定理 B.2 ϕ が $\neg, \wedge, \vee$ のみからなる式であれば、ϕ は単調である。

証明 帰納法により証明される。

1. 式が原子式である時、ϕ は単調。
2. 式が $\neg\phi$ であるとき、ϕ が単調であれば、$\neg$ の定義より単調。
3. 式が $\phi \wedge \psi$ であるとき、ϕ と ψ が共に単調であれば、$\wedge$ の定義より単調。
4. 式が $\phi \vee \psi$ であるとき、ϕ と ψ が共に単調であれば、$\vee$ の定義より単調。

□

極大情報状態の性質

定義 B.5 ϕ が s において決定しているとは、$s \models^+ \phi$ または $s \models^- \phi$ であるときである。

定理 B.3 s が極大情報状態であれば、任意の式は決定している。

証明 同時帰納法により証明される。

1. 式が原子式であれば、決定している。
2. 式が $\neg\phi$ である場合、ϕ が決定していれば決定している。ϕ が決定しているならば、$s \models^+ \phi$ または $s \models^- \phi$ であり、$s \models^+ \phi \Longleftrightarrow s \models^- \neg\phi$、$s \models^- \phi \Longleftrightarrow s \models^+ \neg\phi$ である。すなわち、$\neg\phi$ も決定している。
3. 式が $\phi \wedge \psi$ である場合、同様に ϕ と ψ が決定していれば決定している。
4. 式が $\phi \vee \psi$ である場合、同様に ϕ と ψ が決定していれば決定している。
5. 式が $\Box\phi$ である場合。
   今、極大情報状態 s_m を考える時、$s \sqsupseteq s_m$ なる s は s_m のみである。したがって、
   $s_m \models^+ \Box\phi \Longleftrightarrow s_m \models^- \phi$ でない。
   $s_m \models^- \Box\phi \Longleftrightarrow s_m \models^- \phi$。
   今、ϕ が決定している場合、$s_m \models^+ \phi$ または $s_m \models^- \phi$ であるので、
   $s_m \models^+ \Box\phi \Longleftrightarrow s_m \models^+ \phi$
   したがって、ϕ が決定していれば、$\Box\phi$ も決定している。

6. 式が $\Diamond\phi$ である場合。
   今、極大情報状態 s_m を考える時、$s \sqsupseteq s_m$ なる s は s_m のみである。したがって、
   $s_m \models^+ \Diamond\phi \Longleftrightarrow s_m \models^+ \phi$
   $s_m \models^- \Diamond\phi \Longleftrightarrow s_m \models^+ \phi$ でない。
   今、ϕ が決定している場合、$s_m \models^+ \phi$ または $s_m \models^- \phi$ であるので、
   $s_m \models^- \Diamond\phi \Longleftrightarrow s_m \models^- \phi$
   したがって、ϕ が決定していれば、$\Diamond\phi$ も決定している。
7. 式が $\phi \Rightarrow \psi$ である場合。
   今、極大情報状態 s_m を考える時、$s \sqsupseteq s_m$ なる s は s_m のみである。したがって、
   $s_m \models^+ \phi \Rightarrow \psi \Longleftrightarrow s_m \models^+ \phi$ かつ $s_m \models^- \psi$ でない。
   $s_m \models^- \phi \Rightarrow \psi \Longleftrightarrow s_m \models^+ \phi$ かつ $s_m \models^- \psi$ である。
   今、ϕ と ψ が決定している場合、（$s_m \models^+ \phi$ かつ $s_m \models^- \psi$ でない）または（$s_m \models^+ \phi$ かつ $s_m \models^- \psi$）である。したがって、ϕ と ψ が決定していれば、$\phi \Rightarrow \psi$ も決定している。

□

定理 B.4 s が極大情報状態であれば、次のことは成り立つ。

1. $s \models^+ \Diamond\phi \Longleftrightarrow s \models^+ \phi$
2. $s \models^- \Diamond\phi \Longleftrightarrow s \models^- \phi$
3. $s \models^+ \Box\phi \Longleftrightarrow s \models^+ \phi$
4. $s \models^- \Box\phi \Longleftrightarrow s \models^- \phi$

証明 定理 B.3の証明において証明済み。 □

様相の同値関係

定理 B.5 $\neg\neg\phi \equiv \phi$

証明 $s \models^+ \neg\neg\phi \Longleftrightarrow s \models^- \neg\phi \Longleftrightarrow s \models^+ \phi$
$s \models^- \neg\neg\phi \Longleftrightarrow s \models^+ \neg\phi \Longleftrightarrow s \models^- \phi$
よって、$\neg\neg\phi \equiv \phi$ □

定理 B.6 $\Diamond\phi \equiv \neg\Box\neg\phi$

証明 $s \models^+ \neg\Box\neg\phi$
$\Longleftrightarrow s \models^- \Box\neg\phi$
$\Longleftrightarrow \exists s' \sqsupseteq s$ において $s' \models^- \neg\phi$
$\Longleftrightarrow \exists s' \sqsupseteq s$ において $s' \models^+ \phi$
$\Longleftrightarrow s \models^+ \Diamond\phi$
$s \models^- \neg\Box\neg\phi \Longleftrightarrow$
$\Longleftrightarrow s \models^+ \Box\neg\phi$
$\Longleftrightarrow \forall s' \sqsupseteq s$ において $s' \models^+ \neg\phi$ でない
$\Longleftrightarrow \forall s' \sqsupseteq s$ において $s' \models^- \phi$ でない
$\Longleftrightarrow s \models^- \Diamond\phi$ □

定理 B.7 $\Box\phi \equiv \neg\Diamond\neg\phi$

証明 $\neg\Diamond\neg\phi \equiv \neg(\neg\Box\neg)\neg\phi \equiv (\neg\neg)\Box(\neg\neg)\phi \equiv \Box\phi$ □

定理 B.8 $\neg\Diamond\phi \equiv \Box\neg\phi$

証明 $\Box\neg\phi \equiv \neg\Diamond\neg\neg\phi \equiv \neg\Diamond\phi$ □

定理 B.9 $\Box\neg\phi \equiv \neg\Diamond\phi$

証明 $\Box\neg\phi \equiv \neg\Diamond\neg\neg\phi \equiv \neg\Diamond\phi$ □

定理 B.10 $\Diamond\neg\phi \equiv \neg\Box\phi$

証明 $\Diamond\neg\phi \equiv \neg\Box\neg\neg\phi \equiv \neg\Box\phi$ □

定理 B.11 $\Box\Diamond\neg\phi \equiv \neg\Diamond\Box\phi$

証明 $\neg\Diamond\Box\phi \equiv \neg(\neg\Box\neg)(\neg\Diamond\neg)\phi \equiv (\neg\neg)\Box(\neg\neg)\Diamond\neg\phi \equiv \Box\Diamond\neg\phi$ □

定理 B.12 $\Diamond\Box\neg\phi \equiv \neg\Box\Diamond\phi$

証明 $\neg\Box\Diamond\phi \equiv \neg(\neg\Diamond\neg)(\neg\Box\neg)\phi \equiv (\neg\neg)\Diamond(\neg\neg)\Box\neg\phi \equiv \Diamond\Box\neg\phi$ □

定理 B.13 $\Diamond\Box\Diamond\neg\phi \equiv \neg\Box\Diamond\Box\phi$

証明 同様。 □

定理 B.14 $\Box\Diamond\Box\neg\phi \equiv \neg\Diamond\Box\Diamond\phi$

証明 同様。 □

定理 B.15 $\Box\Box\phi \equiv \Box\phi$

証明 $s \models^+ \Box\Box\phi$
$\Longleftrightarrow \forall s' \sqsupseteq s$ で $s' \models^- \Box\phi$ でない。
$\Longleftrightarrow \forall s' \sqsupseteq s$ で ($\exists s'' \sqsupseteq s'$ で $s'' \models^- \phi$) でない。
$\Longleftrightarrow \forall s' \sqsupseteq s$ で $\forall s'' \sqsupseteq s'$ で $s'' \models^- \phi$ でない。
$\Longleftrightarrow \forall s' \sqsupseteq s$ で $s' \models^- \phi$ でない。
$\Longleftrightarrow s \models^+ \Box\phi$
$s \models^- \Box\Box\phi$
$\Longleftrightarrow \exists s' \sqsupseteq s$ で $s' \models^- \Box\phi$
$\Longleftrightarrow \exists s' \sqsupseteq s$ で $\exists s'' \sqsupseteq s'$ で $s'' \models^- \phi$
$\Longleftrightarrow \exists s' \sqsupseteq s$ で $s' \models^- \phi \Longleftrightarrow s \models^- \Box\phi$
したがって、$\Box\Box\phi \equiv \Box\phi$ □

定理 B.16 $\Diamond\Diamond\phi \equiv \Diamond\phi$

証明 $\Diamond\Diamond\phi \equiv \neg\Box\neg\neg\Box\neg\phi \equiv \neg\Box\Box\neg\phi \equiv \neg\Box\neg\phi \equiv \Diamond\phi$ □

定理 B.17 $s_0 \models^+ \Box\Diamond\phi \Longleftrightarrow \langle S, \sqsubseteq\rangle$ （ただし $S = \{s | s \sqsupseteq s_0\}$ ）における全ての極大情報状態 s で、$s \models^+ \phi$

証明 （左辺） $\Longrightarrow$ （右辺）
ある極大情報状態 s_m で $s_m \models^- \phi$ と仮定する。このとき、定理 B.4 より $s_m \models^- \Diamond\phi$ である。$s_m \sqsupseteq s_0$ なので、$s_0 \models^+ \Box\Diamond\phi$ は成り立たず、左辺と矛盾する。したがって、（左辺） $\Longrightarrow$ （右辺）が成り立つ。
（右辺） $\Longrightarrow$ （左辺）
いま、任意の極大情報状態 s_m で $s_m \models^+ \phi$ が成り立つとする。任意の情報状態 s_i は必ず $s_m \sqsupseteq s_i$ なる極大情報状態 s_m をもつので、$s_i \models^+ \Diamond\phi$ が成り立つ。したがって、$s_0 \models^+ \Box\Diamond\phi$ が成り立つ。 □

定理 B.18 $s_0 \models^+ \Diamond\Box\phi \Longleftrightarrow \langle S, \sqsubseteq\rangle$ （ただし $S = \{s | s \sqsupseteq s_0\}$ ）におけるある極大情報状態 s_m で、$s_m \models^+ \phi$

証明 （左辺） $\Longrightarrow$ （右辺）
すべての極大情報状態 s_m で $s_m \models^- \phi$ と仮定する。任意の情報状態 s_i は必ず $s_m \sqsupseteq s_i$ なる極大情報状態を持つので、$s_i \models^- \Box\phi$ である。すなわち、$s_0 \models^+ \Diamond\Box\phi$ に反する。したがって、（左辺） $\Longrightarrow$ （右辺）が成り立つ。
（右辺） $\Longrightarrow$ （左辺）
いま、ある極大情報状態 s_m で $s_m^+ \models \phi$ が成り立つとする。このとき、定理 B.4 より、$s_m \models^+ \Box\phi$。$s_m \sqsupseteq s_0$ なので、$s_0 \models^+ \Diamond\Box\phi$ が成り立つ。 □

定理 B.19 $s_0 \models^+ \Box\Diamond\Box\phi \Longleftrightarrow \langle S, \sqsubseteq\rangle$ （ただし $S = \{s | s \sqsupseteq s_0\}$ ）における全ての極大情報状態 s で、$s \models^+ \phi$

証明 （左辺）$\Longrightarrow$（右辺）
ある極大情報状態 s_m で $s_m \models^+ \phi$ でないと仮定する。このとき、定理 B.3より、$s_m \models^- \phi$ であり、さらに定理 B.4より、$s_m \models^- \Box\phi$ である。さらに $s_m \models^- \Diamond\Box\phi$ である。したがって、$s_m \sqsupseteq s_0$ なので、$s_0 \models^+ \Box\Diamond\Box\phi$ は成り立たない。したがって、（左辺）$\Longrightarrow$（右辺）が成り立つ。
（右辺）$\Longrightarrow$（左辺）
いま、任意の極大情報状態 s_m で $s_m \models^+ \phi$ が成り立つとする。このとき、定理 B.4より、任意の極大情報状態 s_m で $s_m \models^+ \Box\phi$ である。任意の情報状態 s_i は必ず $s_m \sqsupseteq s_i$ なる極大情報状態 s_m をもつので、$s_i \models^+ \Diamond\Box\phi$ が成り立つ。したがって、$s_0 \models^+ \Box\Diamond\Box\phi$ が成り立つ。 □

定理 B.20 $s_0 \models^+ \Diamond\Box\Diamond\phi \Longleftrightarrow \langle S, \sqsubseteq\rangle$ （ただし $S = \{s | s \sqsupseteq s_0\}$ ）におけるある極大情報状態 s_m で、$s_m \models^+ \phi$

証明 （左辺）$\Longrightarrow$（右辺）
すべての極大情報状態 s_m で $s_m \models^+ \phi$ でないと仮定する。このとき、定理 B.3より、すべての極大情報状態 s_m で $s_m \models^- \phi$ である。このとき、定理 B.4より、任意の極大情報状態 s_m で $s_m \models^- \Diamond\phi$ が成り立つ。任意の情報状態 s_i は必ず $s_m \sqsupseteq s_i$ なる極大情報状態を持つので、$s_i \models^- \Box\Diamond\phi$ である。すなわち、これは $s_0 \models^+ \Diamond\Box\Diamond\phi$ に反する。したがって、（左辺）ならば（右辺）が成り立つ。
（右辺）$\Longrightarrow$（左辺）
いま、ある極大情報状態 s_m で $s_m \models^+ \phi$ が成り立つとする。このとき、定理 B.4を 2 回利用することにより、$s_m \models \Box\Diamond\phi$。$s_m \sqsupseteq s_0$ なので、$s_0 \models^+ \Diamond\Box\Diamond\phi$ が成り立つ。□

定理 B.21 $\Box\Diamond\phi \equiv \Box\Diamond\Box\phi$

証明 定理 B.17と定理 B.19より、$\Box\Diamond\phi \cong \Box\Diamond\Box\phi$。定理 B.18と定理 B.20において、$\phi$ を $\neg\phi$ で置き換えることにより、$\Diamond\Box(\neg\phi) \cong \Diamond\Box\Diamond(\neg\phi)$。定理 B.12と定理 B.13より、$\neg\Box\Diamond\phi \cong \neg\Box\Diamond\Box\phi$。したがって、$\Box\Diamond\phi \equiv \Box\Diamond\Box\phi$ が成り立つ。 □

定理 B.22 $\Diamond\Box\phi \equiv \Diamond\Box\Diamond\phi$

証明 定理 B.18と定理 B.20より、$\Diamond\Box\phi \cong \Diamond\Box\Diamond\phi$。定理 B.17と定理 B.19において、$\phi$ を $\neg\phi$ で置き換えることにより、$\Box\Diamond(\neg\phi) \cong \Box\Diamond\Box(\neg\phi)$。定理 B.11と定理 B.14より、$\neg\Diamond\Box\phi \cong \neg\Diamond\Box\Diamond\phi$。したがって、$\Diamond\Box\phi \equiv \Diamond\Box\Diamond\phi$ が成り立つ。 □

和・積・含意を含む同値関係

定理 B.23 $\phi \vee \psi \equiv \neg(\neg\phi \wedge \neg\psi)$

証明 $s \models^+ \neg(\neg\phi \wedge \neg\psi)$
$\Longleftrightarrow s \models^- \neg\phi \wedge \neg\psi$
$\Longleftrightarrow s \models^- \neg\phi$ または $s \models^- \neg\psi$
$\Longleftrightarrow s \models^+ \phi$ または $s \models^+ \psi$
$\Longleftrightarrow s \models^+ \phi \vee \psi$
$s \models^- \neg(\neg\phi \wedge \neg\psi)$
$\Longleftrightarrow s \models^+ \neg\phi \wedge \neg\psi$
$\Longleftrightarrow s \models^+ \neg\phi$ かつ $s \models^+ \neg\psi$
$\Longleftrightarrow s \models^- \phi$ かつ $s \models^- \psi$
$\Longleftrightarrow s \models^- \phi \vee \psi$ □

定理 B.24 $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$

証明 $s \models^+ \neg(\neg\phi \vee \neg\psi)$
$\Longleftrightarrow s \models^- \neg\phi \vee \neg\psi$
$\Longleftrightarrow s \models^- \neg\phi$ かつ $s \models^- \neg\psi$
$\Longleftrightarrow s \models^+ \phi$ かつ $s \models^+ \psi$
$\Longleftrightarrow s \models^+ \phi \wedge \psi$
$s \models^- \neg(\neg\phi \vee \neg\psi)$
$\Longleftrightarrow s \models^+ \neg\phi \vee \neg\psi$
$\Longleftrightarrow s \models^+ \neg\phi$ または $s \models^+ \neg\psi$
$\Longleftrightarrow s \models^- \phi$ または $s \models^- \psi$
$\Longleftrightarrow s \models^- \phi \wedge \psi$ □

定理 B.25 $\phi \Rightarrow \psi \equiv \Box(\neg\phi \vee \psi)$

証明 $s_0 \models^+ \phi \Rightarrow \psi \Longleftrightarrow \forall s \sqsupseteq s_0$ において $s \models^+ \phi$ かつ $s \models^- \psi$ でない
$\Longleftrightarrow \forall s \sqsupseteq s_0$ において（$s \models^- \neg\phi$ かつ $s \models^- \psi$）ではない
$\Longleftrightarrow \forall s \sqsupseteq s_0$ において（$s \models^- \neg\phi \vee \psi$）ではない
$\Longleftrightarrow s \models^+ \Box(\neg\phi \vee \psi)$
$s_0 \models^- \phi \Rightarrow \psi \Longleftrightarrow \exists s \sqsupseteq s_0$ において $s \models^+ \phi$ かつ $s \models^- \psi$ である
$\Longleftrightarrow \exists s \sqsupseteq s_0$ において $s \models^- \neg\phi$ かつ $s \models^- \psi$ である
$\Longleftrightarrow \exists s \sqsupseteq s_0$ において $s \models^- \neg\phi \vee \psi$ である
$\Longleftrightarrow s \models^- \Box(\neg\phi \vee \psi)$
□

定理 B.26 $\phi \Rightarrow \psi \equiv \neg\Diamond(\phi \wedge \neg\psi)$

証明 $\phi \Rightarrow \psi \equiv \Box(\neg\phi \vee \psi)$ （定理 B.25）
$\equiv (\neg\Diamond\neg)(\neg\phi \vee \psi)$ （定理 B.7）
$\equiv \neg\Diamond(\phi \wedge \neg\psi)$ （定理 B.24）
□

様相と和積の関係

定理 B.27 $\Box\phi \wedge \Box\psi \equiv \Box(\phi \wedge \psi)$

証明 $s \models^+ \Box\phi \wedge \Box\psi$
$\Longleftrightarrow \forall s' \sqsupseteq s$ において $s' \models^- \phi$ でないかつ $s' \models^- \psi$ でない
$\Longleftrightarrow \forall s' \sqsupseteq s$ において（$s' \models^- \phi$ または $s' \models^- \psi$）でない
$\Longleftrightarrow \forall s' \sqsupseteq s$ において（$s' \models^- \phi \wedge \psi$）でない $\Longleftrightarrow s^+\Box(\phi \wedge \psi)$
$s \models^- \Box\phi \wedge \Box\psi$
$\Longleftrightarrow \exists s' \sqsupseteq s$ において $s' \models^- \phi$ かつ $\exists s'' \sqsupseteq s$ において $s'' \models^- \psi$
$\Longleftrightarrow \exists s' \sqsupseteq s$ において $s' \models^- \phi$ あるいは $s' \models^- \psi$
$\Longleftrightarrow \exists s' \sqsupseteq s$ において $s' \models^- (\phi \wedge \psi)$
$\Longleftrightarrow s \models^- \Box(\phi \wedge \psi)$
□

定理 B.28 $\Diamond\phi \vee \Diamond\psi \equiv \Diamond(\phi \vee \psi)$

証明 $\Diamond\phi \vee \Diamond\psi \equiv \neg\Box\neg\phi \vee \neg\Box\neg\psi \equiv \neg(\Box(\neg\phi) \wedge \Box(\neg\psi)) \equiv \neg\Box((\neg\phi) \wedge (\neg\psi)) \equiv \neg\Box\neg(\neg(\neg\phi) \wedge (\neg\psi)) \equiv \neg\Box\neg(\neg\neg\phi \vee \neg\neg\psi) \equiv \Diamond(\phi \vee \psi)$ □

定理 B.29 $s \models^+ \Diamond(\phi \wedge \psi)$ ならば $s \models^+ \Diamond\phi \wedge \Diamond\psi$（逆は不可）。

証明 $s \models^+ \Diamond(\phi \wedge \psi) \Longleftrightarrow \exists s' \sqsupseteq s$ において（$s' \models^+ \phi$ かつ $s' \models^+ \psi$）$\Longrightarrow \exists s' \sqsupseteq s$ において $s' \models^+ \phi$ かつ $\exists s'' \sqsupseteq s$ において $s'' \models^+ \psi \Longrightarrow s \models^+ \Diamond\phi \wedge \Diamond\psi$ □

様相の縮約

補助定理 B.1 任意の様相列は □ と ◇ の組合せかその先頭に ¬ をつけたものと同値である。

証明 いま、M を $\neg, \Box, \Diamond$ の任意の組合せからなる様相とする。また、N は $\neg, \Diamond, \Box$ を 0 個以上含む様相、L は $\Box, \Diamond$ のみを 0 個以上含む様相とする。また、$\emptyset$ を様相がないことを示すとする（以降同様）。このとき、$M = N_1\neg L_1$ と表すことができる（$\neg$ が M に含まれていなければ題意は自明）。すなわち、この ¬ は一番内側にある ¬ である。このとき、定理 B.5、定理 B.9、定理 B.10を順次適用すると、$M = \neg L$ あるいは $M = L$ にすることができる。すなわち、

1. $N_1 = \emptyset$ の場合、$M = \neg L$ であり、題意を満たす。
2. $N_1 = N_2\neg$ である場合、
   $M = N_2\neg\neg L_1 = N_2 L_1$ である（定理 B.5）。したがって、$N_2 = N_3\neg L_2$ をとると、$M = N_3\neg L_2 L_1$ であり、かつ N_3 の長さは N_1 より短い。もし、N_2 に ¬ が含まれなければ、M は $\Box, \Diamond$ から構成される。

3. $N_1 = N_2\Box$ である場合、
   $M = N_2\Box\neg L_1 = N_2\neg\Diamond L_1$ である（定理 B.9より）。したがって、$L_2 = \Diamond L_1$ とすることにより、$M = N_2\neg L_2$ であり、かつ N_2 の長さは N_1 より短い。
4. $N_1 = N_2\Diamond$ である場合、
   $M = N_2\Diamond\neg L_1 = N_2\neg\Box L_1$ である（定理 B.10より）。したがって、$L_2 = \Box L_1$ とすることにより、$M = N_2\neg L_2$ であり、かつ N_2 の長さは N_1 より短い。

□

補助定理 B.2 任意の □◇ で始まる様相列は □◇ に等しい。

証明 帰納法により求められる。

- $M = \Box\Diamond$ ならば、自明。
- $M = \Box\Diamond\Box M'$ ならば、$M \equiv (\Box\Diamond\Box)M' \equiv \Box\Diamond M'$（定理 B.21より）。
- $M = \Box\Diamond\Diamond M'$ ならば、$M \equiv \Box(\Diamond\Diamond)M' \equiv \Box\Diamond M'$（定理 B.16より）。

□

補助定理 B.3 任意の ◇□ ではじまる様相列は ◇□ に等しい。

証明 帰納法により同様にして求められる。 □

定理 B.30 任意の様相は次の様相かその否定と同値である。

$$\emptyset, \Box, \Diamond, \Box\Diamond, \Diamond\Box$$

証明 今、M は □ と ◇ からなる任意の様相列とする。
今、M が □□ を含む場合、すなわち $M_0 = M_1\Box\Box M_2$ のときは、定理 B.15により、その □ をひとつ取り除く、すなわち $M_0 = M_1\Box M_2$ とすることができる。M が ◇◇ を含む場合、同様にその ◇ をひとつ取り除くことができる。したがって、任意の様相列は □ と ◇ が交互に並んだ様相列に縮約できる。
このとき、M は □◇ ではじまる様相列、◇□ で始まる様相列、□、◇、$\emptyset$ のいずれかである。□◇ ではじまる様相列は補助定理 B.2により □◇ に、◇□ ではじまる様相列は補助定理 B.3により ◇□ に縮約される。
したがって、□ と ◇ からなる任意の様相列は $\emptyset, \Box, \Diamond, \Box\Diamond, \Diamond\Box$ に縮約される。
また、定理 B.1により、任意の様相列は □ と ◇ からなる様相列かに それに否定をつけたものと同値であるので、任意の様相列は $\emptyset, \neg, \Box, \neg\Box, \Diamond, \neg\Diamond, \Box\Diamond, \neg\Box\Diamond, \Diamond\Box, \neg\Diamond\Box$ に縮約される。 □

様相の強さ

定理 B.31 $s_0 \models^+ \Box\phi$ ならば $(S, \sqsubseteq)$ （ただし $S = \{s | s \sqsupseteq s_0\}$ ）における全ての極大情報状態 s で、$s \models^+ \phi$。

証明 $s_0 \models^+ \Box\phi \Longleftrightarrow \forall s' \sqsupseteq s_0$ において $s' \models^- \phi$ でない。
したがって、当然全ての極大情報状態 s_m においても、$s_m \models^- \phi$ でない。定理 B.3 より、$s_m \models^+ \phi$ □

定理 B.32 $(S, \sqsubseteq)$ （ただし $S = \{s | s \sqsupseteq s_0\}$ ）におけるある極大情報状態 s_m で、$s_m \models^+ \phi$ であるならば、$s_0 \models^+ \Diamond\phi$ である。

証明 $s_m \sqsupseteq s$ であるので、当然、$s \models^+ \Diamond\phi$ □

定理 B.33 $\phi \Rightarrow \Diamond\phi$

証明 任意の情報状態 s_0 において、$s_0 \models^+ \phi$ ならば $s \sqsupseteq s_0$ を満たす s_0 で $s_0 \models^+ \phi$ であるので、$s_0 \models^+ \Diamond\psi$。 □

定理 B.34 $\Box\phi \Rightarrow \Box\Diamond\phi$

証明 ある s_0 において、$s_0 \models^+ \Box\phi$ と仮定する。
$s \sqsupseteq s_0$ なる s を考える時、$\{s' | s' \sqsupseteq s\} \subseteq \{s'_0 | s'_0 \sqsupseteq s_0\}$ であり、かつ $s'_0 \models^- \phi$ ではない。したがって任意 $s' \sqsupseteq s$ において $s' \models^- \phi$ でない。よって $s \sqsupseteq s_0$ であれば $s \models^+ \Box\phi$ である。
また、このとき定理 B.31 より $s_m \sqsupseteq s_0$ なる全ての極大情報状態で $s_m \models^+ \phi$ である。したがって、任意の $s \sqsupseteq s_0$ をとると、$\{s_m | s_m \sqsupseteq s$ かつ s_m は極大 $\} \subseteq \{s_m | s_m \sqsupseteq s_0$ かつ s_m は極大 $\}$ なので、$s'_m \sqsupseteq s$ なる極大情報状態のすべてにおいて $s'_m \models^+ \phi$ である。すなわち B.31 より $s \models^+ \Box\Diamond\phi$ である。したがっていかなる $s \sqsupseteq s_0$ において $s \models^+ \Box\phi$ かつ $s \models^- \Box\Diamond\phi$ ではない。よって、$\Box\phi \Rightarrow \Box\Diamond\phi$ が成り立つ。 □

定理 B.35 $\Box\Diamond\phi \Rightarrow \Diamond\Box\phi$

証明 ある s_0 において、$s_0 \models^+ \Box\Diamond\phi$ と仮定する。
このとき定理 B.17 より $s_m \sqsupseteq s_0$ なる全ての極大情報状態で $s_m \models^+ \phi$ である。したがって、任意の $s \sqsupseteq s_0$ をとると、$\{s_m | s_m \sqsupseteq s$ かつ s_m は極大 $\} \subseteq \{s_m | s_m \sqsupseteq s_0$ かつ s_m は極大 $\}$ なので、再び定理 B.17 より $s \models^+ \Box\Diamond\phi$ である。
また、任意の $s \sqsupseteq s_0$ で $s'_m \models^+ \phi$ かつ $s'_m \sqsupseteq s$ なる極大情報状態が存在する。したがって定理 B.18 より $s \models^+ \Diamond\Box\phi$。したがっていかなる $s \sqsupseteq s_0$ において $s \models^+ \Box\Diamond\phi$ かつ $s \models^- \Diamond\Box\phi$ ではない。よって $\Box\Diamond\phi \Rightarrow \Diamond\Box\phi$ が成り立つ。 □

定理 B.36 $\Diamond\Box\phi \Rightarrow \Diamond\phi$

証明 ある s_0 において、$s_0 \models^+ \Diamond\Box\phi$ と仮定する。
このとき定理 B.18 より $s_m \sqsupseteq s_0$ である極大情報状態で $s_m \models^+ \phi$ である。したがって、$s_0 \models^+ \Diamond\phi$ である。よって s_0 において $s_0 \models^+ \Diamond\Box\phi$ かつ $s_0 \models^+ \Diamond\phi$ である。
$s \sqsupseteq s_0$ においては $s \models^+ \Diamond\Box\phi$ または $\models^- \Diamond\Box\phi$ であり、$s \models^+ \Diamond\Box\phi$ ならば同様にして $s \models^+ \Diamond\phi$ である。したがって任意の $s \sqsupseteq s_0$ で $s \models^+ \Diamond\Box\phi$ かつ $s \models^- \Diamond\phi$ でない。 □

定理 B.37 $\Box\phi \Rightarrow \Box\Diamond\phi \Rightarrow \Diamond\Box\phi \Rightarrow \Diamond\phi$

証明 定理 B.34 と定理 B.35 と定理 B.36 より自明。 □

単調式の様相

定理 B.38 単調式 ϕ において、$s_0 \models^+ \Box\phi \Longleftrightarrow (S, \sqsubseteq)$ （ただし $S = \{s | s \sqsupseteq s_0\}$ ）における全ての極大情報状態 s で、$s \models^+ \phi$。

証明 定理 B.31 より（左辺）$\Longrightarrow$（右辺）。今、全ての極大情報状態 s_m で $s_m \models^+ \phi$ とすると、全ての情報状態 s は $s_m \sqsupseteq s$ なる極大情報状態を持ち、かつ ϕ は単調であるので、$s \models^- \phi$ はありえない。したがって、$s \models^+ \Box\phi$。 □

定理 B.39 単調式 ϕ において、$s_0 \models^+ \Diamond\phi \Longleftrightarrow (S, \sqsubseteq)$ （ただし $S = \{s | s \sqsupseteq s_0\}$ ）におけるある極大情報状態 s で、$s \models^+ \phi$。

証明 定理 B.32 より（右辺）$\Longrightarrow$（左辺）。今、$s_0 \models^+ \Diamond\phi$ であるとすると、ある情報状態 $s \sqsupseteq s_0$ で $s \models^+ \phi$。このとき $s' \sqsupseteq s$ なる全ての s' で $s' \models^+ \phi$。s' には少なくもひとつの極大情報状態を含むので、ある極大情報状態 s_m で $s_m \models^+ \phi$ である。 □

定理 B.40 ϕ が単調であれば、$\Box\phi \equiv \Box\Diamond\phi$。

証明 定理 B.17 と定理 B.38 より、$s \models^+ \Box\phi \Longleftrightarrow s \models^+ \Box\Diamond\phi$。定理 B.18 と定理 B.39 を利用して、$\phi$ を $\neg\phi$ に置き換えることにより、$s \models^- \Box\phi \Longleftrightarrow s \models^- \Box\Diamond\phi$。 □

定理 B.41 ϕ が単調であれば、$\Diamond\phi \equiv \Diamond\Box\phi$。

証明 定理 B.18 と定理 B.39 より、$s \models^+ \Diamond\phi \Longleftrightarrow s \models^+ \Diamond\Box\phi$。定理 B.17 と定理 B.38 を利用して、$\phi$ を $\neg\phi$ に置き換えることにより、$s \models^- \Diamond\phi \Longleftrightarrow s \models^- \Diamond\Box\phi$。
□

定理 B.42 ϕ が単調であれば、可能な様相は、以下に示すものかその否定である。

$$\emptyset, \Box, \Diamond$$

証明　定理 B.40 と定理 B.41 より自明。 □

定理 B.43 ϕ が真単調であれば、$\phi \Rightarrow \Box\phi$

証明　ϕ が真単調であるので、$s_0 \models^+ \phi$ ならば $s \sqsupseteq s_0$ なる全ての s で $s \models^+ \phi$ であり、当然 $s \models^- \phi$ でない。よって、$s_0 \models^+ \Box\phi$ □

定理 B.44 ϕ が単調であれば、$\phi \Rightarrow \Box\phi \Rightarrow \Diamond\phi$。

証明　定理 B.37、定理 B.40、定理 B.41、定理 B.43 より自明。 □