PNC TY9605 98-003



# 運転員のメンタルモデル獲得システムの開発（1）

## ―物理レベルのメンタルモデル獲得システムの評価―

（共同研究報告書）

1997年3月



大坂大学産業科学研究所
動力炉・核燃料開発事業団
大洗工学センター

公　開　資　料
PNC TY9605 98－003
1997年　3月

# 運転員のメンタルモデル獲得システムの開発（1）
## -物理レベルのメンタルモデル獲得システムの評価-
（共同研究報告書）

池田　満[1)]、溝口理一郎[1)]
吉川信治[2)]、小澤健二[3)]

## 要旨

本報告書は、原子力プラントの機能や構造に対する運転員の理解状況を計算機によって獲得する手法に関する検討結果を述べる。この手法は、教育課程中の運転員が自らのプラントに対する知識を形成する努力を支援する情報処理技術の中で中核をなすものである。この知識形成支援技術の目的は、予測の困難な事象に遭遇した場合の運転員の対処能力を向上することにある。このような場合には、観測された兆候からその原因を推定したり、推定された異常事象に対して対応操作を決定したりする際に、経験則のみに依存するのではなく、プラント内部の各部の現象に立ち返った考察に裏付けされた意志決定を行う能力を運転員が有していることが望ましい。

本報告書では、このような手法に発展しうる要素技術の候補として、説明に基づく学習と呼ばれる技法を取り上げ、上述のような運転員の学習支援システムへの適用性及び必要な機能拡張について考察し、解決すべき技術課題を整理した。

1)大阪大学産業科学研究所電子機器部門
2)動力炉・核燃料開発事業団大洗工学センター基盤技術開発部先進技術開発室
3)動力炉・核燃料開発事業団大洗工学センター実験炉部

PNC TY9605 98－003
March, 1 9 9 7

# Development of An Operator's Mental Model Acquisition System(1)
## - Estimation of A Physical Mental Model Acquisition System -

Mitsuru Ikeda[1], Riichirou Mizoguchi[1]
Shinji Yoshikawa[2], Kenji Ozawa[3]

## Abstract

This report describes a technical survey of acquisition method of an operator's understanding for functions and structures of his target nuclear plant. This method is to play a key role in the information processing framework to support on-training operators in forming their knowledge of the nuclear plants. This kind of technical framework is aiming at enhancing human operators' ability to cope with anomaly plant situations which are difficult to expect from preceding experiences or engineering surveillance. In these cases, cause identifications and responding operation selections are desired to made not only empirically but also based on thoughts about possible phenomena to take place within the nuclear plant.

This report focuses on a particular element technique, defined as "explanation-based knowledge acquisition", as the candidate technique to potentially be extended to meet the requirement written above, and discusses about applicability to the learning support system and about necessary improvements, to identify future technical developments.

---

1) Institution of Science & Industrial Research, Osaka University
2) Frontier Technology Development Section, Advanced Technology Division, OEC/PNC
3) Experimental Reactor Division, OEC/PNC

# 目 次

# 図目次

# 表目次

# 第1章

# 序論

原子力プラントは複雑で大規模、かつ安全性、信頼性への社会的要求が高いシステムの典型である。従って原子力プラントの運転員が把握しなければならない工学的知識の量、及び対処しなければならない事象の範囲は他の工学系に見られない広さを持つ。現在、運転員にそのような対処能力を獲得させる方法としては、教材による原子力プラントの概念、構造、及びプラントの種々のモードにおける各部の機能や性能、動作条件の教育と、運転訓練用シミュレータを用いた実地訓練が主に用いられている。訓練用のシミュレータには、正常時のプラント挙動のみではなく、実機において想定される不具合をも模擬する機能が備わっているが、不具合のあらゆる規模や組み合わせを訓練中に経験することは現実的には不可能である。さらに、交通システムや小規模の化学プラントと比較すると、原子力プラントに関しては、周辺環境へ重大な影響を及ぼす事故事例で実機において過去に経験されたものは極めて数少なく、しかもそのような事故に至りうる事象の発生に際してはその初期に正確な診断を行い、かつ迅速に適切な対応操作を取る能力が運転員に要求されるという点が大きく異なる。

また、近年の計測制御技術の進歩が運転員の任務に与えた影響を考えると、従来のようにルーチン的な監視、制御を定常的に行っている場合には、プラント内部の機能や挙動に対する関心が必然的に保たれていたのに比べて、近年は定常的な監視、制御が自動化され、運転員にはこのような自動化システムが対処できないような異常事象が発生した場合に必要な高度な判断が要求されている。これは正常時と異常時の負担の比率が増大したことを意味しており、精神的な負荷が急増しても的確な判断を行う能力の重要性は増してきていると言える。

このような要求に応えるためには、観測された兆候からその原因を推定したり、推定された異常事象に対して対応操作を決定したりする際に、プラント内部の各部の現象に立ち返った考察に裏付けされた意志決定を行う能力を運転員が有していることが望ましい。このような能力は、定常業務における手順の習得や、訓練過程において対象事象に設定されている異常事象に対する対処法を習得する際にも、その学習効率を高め、知識の定着をより確かなものにする効果があると期待できる。

このような能力の醸成を支援するソフトウェアシステムの構築技術を開発するための

研究を平成６年度より行っている。平成１０年度末に、訓練過程にある運転員が計算機の前に座って対話的な操作を行うことによってプラントに対する知識を向上できるようなソフトウェアの枠組を構築することを目標としている。このようなソフトウェアは、1)システム使用者のプラントに対するその時点での理解状況を把握する部分（学習者モデルモジュール）、2)教示すべき知識、情報に基づいて質問を生成し、使用者の回答を評価する部分（教材知識モジュール）、3)前述の２つの部分と使用者の対話を司る部分（個人指導モジュール）から成る。

この中で、教材知識モジュールに関しては、原子力施設の運転及び運転員の教育に関する経験がある程度活用できることから、運転訓練用シミュレータを用いて、予測の困難な異常事象に遭遇した場合の診断における運転員の知識活用形態を観察、分析した[1]。また、先行研究の調査に基づいて人間が複雑なプロセスの運転、診断を行う際に用いる知識の基本構造を仮定し、比較的単純な系を計算機上に構築して被験者を用いてこの基本構造の妥当性を確認した[2]。

これらに対して、1)の学習者モデルモジュールは、システムの中核である一方、３者の内最も技術的課題を多く含み、システムの設計に先だって十分な検討を必要とする。従って、原子力プラントを対象とする診断、操作に領域を限定する前に、対象となる系や学習した知識を適用する作業に関する種々の組み合わせについての考察をしておくことが妥当と思われる。

一方、人間の知識を計算機で処理することを目的として構築されてきた技術分野、主に人工知能と呼ばれる分野においても、知識の獲得は主要テーマである。この分野においては当初、人間が知識を用いて新たな情報を導くプロセス、即ち推論を実現するためのアルゴリズムの研究に重点がおかれてきたが、単純な命題の処理を追及するだけでは実際の場で有益な技術とは成りえないことが次第に明かになってきた。人間がそれぞれの作業において扱う知識そのものの研究に重点が移り、またそのような知識を如何に主観的な解釈を加えずに計算機に取り込むかが知識工学の成功の鍵であることが広く認識されるに至ったのである。

これは主に、専門家の知識をｉｆ－ｔｈｅｎ-ルールで記述してこの上に推論アルゴリズムを搭載すれば専門家の意思決定能力を工業製品化できるという発想化から生まれたエキスパートシステムの開発努力の中で経験された「知識獲得のボトルネック」を背景とする。

「知識獲得のボトルネック」とは、エキスパートシステムに記述する知識を専門家へのインタヴューにより抽出し、ｉｆ－ｔｈｅｎ－ルールに変換して計算機へ書き込んで、推論さえすれば専門家と同じ判断ができるような知識ベースを構築することが、ほとんどの場合工学的に成立しない、ということである。工学的に成立しない、とは、計算機に依存する必要があるような規模の対象については、上記の手順の作業量がただ膨大になるということではなく、個々のルール間の整合性の保持、記述の統一性の確保、未抽出知識のチェック等を人間の手で行うコストが、受容できる限度を越えているということである。このことから、知識ありきを前提とする研究ではなく、人間の所有する知識を計算機に記述するプロセスそのものを工学的に実現することの方がはるかに重要な意

味を持つことが認識され、知識獲得の研究が知識工学の主要テーマになった。

本報告書では、一個の人間のある工学系に対する理解度を計算機によって把握する手法の基となる手法として、説明に基づく学習をとりあげ、その理論的内容、知識獲得のための諸方法の特徴、質疑応答の制御への適用性、教育支援システムの中核モジュールとして使用するために望まれる機能拡張について述べる。

# 第 2 章

# 知識獲得

## 2.1 緒言

知識獲得の重要性は知識工学の黎明期から認識されていたが, 知識工学者自身による専門家へのインタビューによる知識獲得に甘んじていたため, 知識表現等の研究の後塵を拝してきた. 最近になってその重要性が再認識され, 多くの研究が行なわれてきた[3].

本章では, 知識獲得の知識工学における位置付け, ならびに知識獲得の問題を考える上で重要と思われる視点を整理するとともに, 代表的なシステムを概観する.

推論中心の初期の人工知能から知識中心の知識工学への人工知能研究の変遷は, 推論中心の研究の失敗から生じた自然な選択であり, 知識の重要性の再認識の歴史であった. 2.2節では, このような背景を持った知識工学における知識獲得の重要性と将来について述べる.

2.3節では, 獲得手法という視点から知識獲得を整理するとともにそれぞれの手法の代表的なシステムを紹介する. 本論文で取り扱う知識獲得の手法の1つである説明に基づく学習については第3章で詳細に述べ, また第4章で用いるインタビューによる獲得についてはその章で詳細に述べる.

## 2.2 知識獲得の位置付け

人工知能 (Artificial Intelligence, 略して AI) とは, 人間のような知的な能力をもつ機械の実現を目指す研究のことである. ここでいう機械とは, 実際にはコンピュータであり, また実現の手段はプログラムであるから, AI とは知的コンピュータあるいは知的プログラムに関する研究であるといってよい.

さて, 人間のように知的に問題を解決するコンピュータを実現したいという研究はコンピュータが実用化され始めた1950年代中頃からすでに開始されている. しかし, 当初は一般の複雑な問題の代わりに, ゲームやパズルを対象

とした研究がなされていた. これは, どの手が最適であるか, 少なくとも負けを最小にするにはどの手を選ぶべきか, というような判断と, いろいろな戦略ルールや判断基準を組み合わせて1つの決定を行なうという推論についての研究が中心であった. これらの研究を通じて, 単純かつ強力な汎用性の高い推論のメカニズムが追求された. 一般性の高い推論機構が発見できれば, 一般の複雑な問題に応用できると考えられたようである. しかし, 初期のAI研究者達の試みは, そのままの形では成功しなかった. なぜなら, 一般性が高く強力と思われる推論機構が発見されても, 一般の複雑な問題に応用するときに, その柔軟性の欠如および単一性に阻まれ十分な結果を生まなかった. そこで知識工学(Knowledge Engineering)と呼ばれる実用指向のAI研究へと引き継がれた. 知識工学では初期のAI研究で重点が置かれた汎用で強力な推論機構ではなく, 実用に十分な可能性がある知識にその重点が置かれた. 後に初期のAI研究は推論中心のAIと呼ばれるようになった.

知識工学のアプローチでは, 扱う知識の表現, 利用および獲得が最も重要な課題とされた. 以下にそれぞれの課題について述べる.

知識の表現(Knowledge Representation)とは, 知識をコンピュータで処理するために一定の形式で表現することをいう. 形式が一定であることは, コンピュータ処理上極めて重要なことである上, この知識表現は人間にとっても理解しやすい形式となる. これによって, 知識をより構造的に扱い, 整理することができる.

知識の利用(Use of Knowledge)とは, 一定形式で蓄積された知識を利用して, 推論を実行し, 結論を得ることをいう. これは現在の状態, 目標の方向, 現在焦点を当てている問題, 推論などの戦略などに基づいて, 適切な知識を取り出して, 運用し, 推論を進めるということである. このため, 知識の探索, 推論の戦略, 知識の解釈, 推論の道筋の説明など重要な問題がある.

知識の獲得(Knowledge Acquisition)とは, 問題領域の専門知識を獲得することである. 問題解決器の能力は, 獲得された知識の質と量に依存するため, この機能は極めて重要である. 特にエキスパートシステムの場合には, 自分自身の知識を明確に表現することが困難な専門家から知識を引き出すこととなるため, 一般にこの作業は専門家と知識工学者(Knowledge Engineer)との時間のかかる忍耐のいる共同作業である.

このように知識工学には, さまざまな取り組むべき課題があるが, 特に問題解決の能力を決定する要因である知識獲得の重要性は無視できない. 上述したように, 知識獲得はエキスパートシステム出現以来, 最も時間と忍耐のいる作業であった. もし, この作業をコンピュータ支援によって半自動化させることが可能であれば, 知識工学者の負担は軽減されるであろう. また, 専門家と知識工学者の人対人の作業では, 不注意による誤り, 記憶違いによる作業の重複などさまざまな問題が考えられるが, これらも克服することが可能であろう. 知識工学の中心である知識をまず最初に取り扱う部分である知識獲得の研究の成功は知識表現や知識利用の研究を活発化させ, 実用に十分耐える問題解

決器(エキスパートシステム)の開発に拍車をかけることが予想される.

## 2.3 知識獲得の形態

知識獲得に関する研究は最近盛んに論じられているが,多くの研究は知識獲得のある1つの視点を強調したもので,知識獲得の全体像が把握し難いのが現状である.知識獲得は広義には知識ベースの構築支援全般を意味するが,そのように考えたとき,その形態は次の5つに分類することができる[1].

- 深い知識からの生成
- 例題からの学習
- インタビューによる獲得
- 専用言語の開発
- 専用ツールによる管理

以下,各項目についてその形態と代表的なシステムについて概観する.

### 2.3.1 深い知識からの生成

専門家は基本的な知識を学習した後,さまざまな経験を通じて経験則を習得していく.このような専門家の成長過程に着目して,深い知識と呼ばれるドメインの基本原理や対象物の構造や部品の機能に関する知識から,タスクに直接使われる浅い知識をコンパイルすることができる[4].深い知識は客観的な知識であり,基本的にはドメインに依存するが,タスクには依存しないものである.タスク知識は深い推論エンジン(コンパイラ)が持っている.この考えの基本には,有効な経験則(浅い知識)の多くは客観的な知識によってその正当性を説明できるという考えがある.理想的には,深い知識は1つのドメインで共通のものが定義され,その上でのタスクに依存した深い推論エンジンがさまざまな浅い知識を生成するという形態が望まれる.すべての浅い知識をコンパイルしておくことは不可能であり,また現実的でない.ある程度,使用頻度の高い浅い知識だけを事前にコンパイルしておき,残りのものは,実際に困難な問題に出会ったときに動的に生成し,有効に用いられたものを蓄積するという方法が妥当であろう.

深い知識を用いたシステムに來村らの故障診断システムKCIII[5]がある.KCIIIは物理法則に基づいて因果の流れに沿って系の挙動を推論し,故障診断を行ない,その結果を故障診断ルールとして生成する.系の挙動の推論には定性推論[6]を用いており,人の定性的な推論に類似した推論を行なうことができる.KCIIIは原子力プラント"常陽"を対象としており,その推論結果は専門家の行なうものと一致したとの報告がある.また,同研究グループでは"常陽"

のモデルの部品の記述に抽象的概念に対応づけられた語彙(ドメインオントロジー)を用いることを試みており[7], KCIIIの故障診断の効率を高め,高品質の故障診断ルールの獲得を目指している.

### 2.3.2 例題からの学習

学習は知識ベース構築の観点から見れば極めて重要な課題である.従来の学習に関する研究は, SBL(Similarity-Based Learning)と呼ばれ,事前の知識はほとんど仮定せず,多数の例題中の類似性に注目し,帰納的に学習を行なうものが中心であった.近年EBL(Explanation-Based Learning)と呼ばれる新しい学習方式が注目されている.EBLではSBLとは対象的にドメインに関する豊富な知識(Domain Theory)を事前に用意しており,多少の例題（通常は1つ）から学習を行う方式である.実際,人間の学習課程を考えると,EBLの方が現実的であるように思われる.しかも,知識獲得の立場からみれば,Domain Theoryは客観的な知識であるため,その獲得は比較的容易であり,それに基づいてEBLによってタスクの解決に有効な知識が得られることは大変望ましいことである.これは弟子が師匠の動作を側で見ていて,そのノウハウを学ぶことに類似しており,知識獲得の重要な形態の1つとなるであろう.

### 2.3.3 インタビューによる獲得

インタビューによる獲得が普通に言われる知識獲得であり,知識工学者が専門家にインタビューすることによって知識を獲得する行為を指す.現状ではこの形態が大部分のエキスパートシステムの構築において採用されており,最も重要な問題となっている.一般に,人間は自分の知っていることを客観的には理解していない.特に,経験則と言われるような半分無意識の世界にあるような知識に関してはいっそうその表明は困難となる.したがって,インタビューによる獲得では,専門家が自分の知識を表明しやすいような状況を作ることが重要な課題となる.言い忘れているところや矛盾の指摘はもちろんのこと,知識を提供する側の心理的負担を軽くする考慮も必要になる.

### 2.3.4 専用言語の開発

特定のエキスパートシステムが成熟するにつれて知識の枠組が固まることを利用して,そのシステムに固有の知識表現言語を開発することができる.この言語を用いて,記述の概念レベルを高めることができる.汎用の記述言語は記述の際の自由度が大きく,問題領域の専門家が直接その言語を利用して知識を記述するのは困難であるが,十分検討された問題専用の言語が設定されればそれが可能となる.

この専門家が直接知識を記述できるような専用言語を利用したシステムの例として, XCON-RIME[8]やOPAL[9]を挙げることができる.例えば,現在10000

以上のルールを持つXCONは，毎年半数のルールを修正し，保守を行なっている．その作業を軽減するために，RIMEと呼ばれる専用の言語を開発している．

### 2.3.5 専用ツールによる管理

知識ベースの管理は，知識獲得が持つイメージとはかなり遠いが，知識ベースの構築には不可欠なものである．知識の作成の動機や目的，適用結果の履歴，類似した知識，共に用いられるべき知識等の関連情報が得られないと，知識の矛盾や冗長性等が生じ，知識ベース構築に悪影響が発生する可能性が高い．しかも大規模な知識ベースを構築する場合には複数の人が携わる場合が多く，その場合は更に深刻な問題となるであろう．

このような知識ベースの管理を行なうシステムの例として北上らの知識ベース管理システム[10]や辻野らのSPREXII[11]がある．例えば，知識ベースの修正の専用言語の概念も組み込みんだSPREXIIでは，システムの性能を落すことなくルール数を70%まで減少させたり，全く新しいルールを試みることや複数の専門家による構築作業が容易になった，との報告がある．

## 2.4 結言

本章では，知識獲得の知識工学における重要性，困難さについて述べた．推論中心の人工知能から知識中心の知識工学への大きな転換とそれによって生じた知識獲得が人工知能研究で占める重要な役割を概観した．重要性が再認識されてきたことによって，さまざまなアプローチによる研究がなされてきた．本章ではそれらのアプローチを形態によって分類することによって整理した．

# 第3章

# 説明に基づく学習

## 3.1 緒言

コンピュータによる学習(機械学習)の歴史は古く,1959年Samuelによるチェッカのプログラム[12]に起源する.そのプログラムは,ゲームの遂行過程のデータに基づいて,評価関数のパラメータを学習する単純なものであったが,チェッカの世界チャンピオンを破り,コンピュータによる学習の応用可能性を最初に示した.以降,学習の研究は人工知能研究の最も活発なテーマとなっている.

学習の研究には,外部教師あるいは環境から得られる事実に基づいて帰納的推論を行ない,知識を獲得する類似に基づく学習(帰納学習とも呼ばれる),事前知識として公理系が与えられ,学習用の概念として示される定理の証明(演繹的推論)を行ない,知識を獲得する説明に基づく学習(演繹学習とも呼ばれる),既知のある世界の関係を未知の現在の世界の関係に写像し,知識を獲得する類推による学習,与えられた世界の知識から,有用な新しい概念なり知識を形成する発見による学習などがある.知識獲得の一つの形態としての学習においては,例の類似性に基づく学習(SBL: Similarity-Based Learning)が最も精力的に研究がなされており,実用面においても十分な成果を上げている.

一方,説明に基づく学習(EBL:Explanation-Based Learning)は1980年代前半から研究が活発化してきた比較的新しい学習パラダイムである.SBLはドメインに関する知識をほとんど持っていないため,SBLのパラダイムによって得られた結果の正当性は,そのような例題が存在していたから,ということ以上の説明をすることはできない.しかも,適切な学習を行なうためには多くの学習サンプルが必要となる.それに対してEBLはドメインに関する豊富な事前知識を前提とし,例題を証明して学習するため,結果の正当性は保証される.しかもこの学習パラダイムは人間の学習に近く,SBLより自然な枠組であると言える.

本章では,本論文で取り上げた説明に基づく学習の枠組を紹介するとともに,EBLの概念を採り入れたシステムの歴史,EBLに関するトピックスを概観する.

3.2節では,さまざまな形式で用いられていた説明に基づく学習の概念を最初に整理,提言したMitchellらのEBG(Explanation-Based Generalization)とそれの不備を指摘し,同時に発展させたDeJongらのEBLについて紹介する.

3.3節では,説明に基づく学習に関する全体的な様相について述べる.文献[2]においてDeJongらによってEBLという用語が使われ始めたが,実際にはそれより以前にEBLの概念は存在していた.これらEBLの概念を採り入れたシステムをまとめる.またEBLに関するいくつかのトピックについて述べる.

## 3.2 EBG/EBL

説明に基づく学習という言葉は,1986年のMitchellらによる論文"Explanation-Based Generalization: A Unifying View"[13]における提案に対する返答という形で同じ論文誌にDeJongらが投稿した"Explanation-Based Learning: An Alternative View"[2]に起源する.それまでは"Explanation-Based Approach"としてさまざまな形態を採っていたが,これら一連の論文によって,それまで帰納学習が主流であった学習分野に説明に基づく学習という演繹学習の新しいパラダイムが統合,形成された.

本節ではこれら演繹学習の着火源となった2つの学習パラダイムを紹介する.

### 3.2.1 EBG: A Unifying View

説明に基づく一般化(EBG)について文献[13]にある例題を用いて紹介する[1].

#### EBG問題

説明に基づく一般化問題を表3.1に示す.また,cupの一般化問題の具体例における入力を表3.2に示す.表3.1と表3.2に示すように,EBGでは以下の4種類の情報を入力とする.

1. 目標概念は獲得すべき概念を示す.例えば,表3.2に示すタスクは,持ち上げられて(liftable),安定していて(stable),上方が開いている(open-vessel)もの(X)を,カップ(cup)として認識できる概念定義を学習することである.ここでの目標概念は述語cupである.注目すべきことは,目標概念を記述する述語と訓練例を記述する述語が異なることである.例えば,cupの例では,目標概念は機能的側面からみたカップの定義として記述されているが,訓練例は,カップの所有者や構造的側面などで記述されている.
2. 訓練例は目標概念の正の例である.例えば,表3.2では,カップであるオブジェクトobj1を前述した構造的側面などで記述している.

[1]文献[13]では表記法として,定項を大文字,変項を小文字として用いているが,本論文では定項を小文字,変項を大文字として統一的に扱う.

表 3.1: 説明に基づく一般化問題

Given:

- 目標概念 (goal concept): 学習すべき概念を記述した概念定義.
- 訓練例 (training example): 目標概念の例.
- 領域知識 (domain theory): 訓練例が目標概念の例となっていることを説明するために用いられるルールとファクトの集合.
- 操作性規範 (operationality criterion): 学習した概念定義を表現するための形式を制限する概念定義上の述語.

Determine:

- 目標概念に対する十分な概念定義となり, かつ, 操作性規範を満足する例題の一般化.

表 3.2: カップ一般化問題

Given:

- 目標概念: 次の cup(X) を満たすオブジェクト X のクラス
  cup(X)⇔liftable(X)∩stable(X)∩open-vessel(X)
- 訓練例:
  owner(obj1,edgar)
  part-of(obj1,concavity-1)
  is(obj1,light)
  ⋮
- 領域知識:
  is(X,light)∩part-of(X,Y)∩isa(Y,handle)→liftable(X)
  part-of(X,Y)∩isa(Y,bottom)∩is(Y,flat)→stable(X)
  part-of(X,Y)∩isa(Y,concavity)∩is(Y,upward-pointing) →open-vessel(X)
  ⋮
- 操作性規範:概念定義は, 例を記述するときに使われる構造的側面で表現されなければならない (例えば, light, handle, flat など).

3. 領域知識は，訓練例がなぜ目標概念のメンバとなっているかを説明するために必要なルールやファクトの集合を含む．表3.2の例では，liftable, stable, open-vesselの定義を含んでいる．例えば，あるオブジェクト(X)が軽くて(light)，その一部(part-of)に取っ手(handle)があれば，それは持ち上げられる(liftable)というように，liftable(X)を定義している．つまり，目標概念の定義で用いられた用語を，対象のよりプリミティブな構造的性質(flat, handleなど)に関係付けている．
4. 操作性規範は，出力の概念定義を表現する用語を規定している．例えば，表3.2では，カップの例を視覚的に認識するために有用な概念(つまり，構造的側面)で表現することを要求している．

説明に基づく一般化は，以上の4種類の入力情報をもとに，目標概念に対して十分な概念定義でかつ操作性規範を満足する訓練例の一般形を決定することである．

ここで操作性規範の重要性を認識する必要がある．もし，操作性規範がないと，目標概念がそのまま出力概念となり，何も学習しないことになる．学習される概念定義は観念的・理論的なものでなく，特定のタスクを処理する者にとって実利的・実用的なものでなければならない．この操作性規範の重要性と方向性については第5章で詳しく議論する．

### EBG手法

説明に基づく一般化の手法は，次の二つのステップからなる．

1. 説明(explanation): 訓練例が目標概念の定義をどのように満足するかを領域知識の表現言語を使って証明した説明木を構築する．この説明木の各枝の終端は操作性規範を満足する用語でなければならない．
2. 一般化(generalization): 操作性規範を満足する用語で記述された説明が成り立つ十分条件の集合を決定する．これは説明木を通して目標概念を回帰する(goal regression)ことで達成する．回帰して得られた概念記述式の連言が学習結果の概念定義となる．

前述した表3.2の例題タスクにおけるカップ例obj1が目標概念cupをどのように満足するかを明示する説明木を図3.1に示す．この説明木は訓練例で本質的な側面(lightであることなど)と無関係な側面(colorなど)とを区別している．次の一般化ステップでは，目標概念を説明木で回帰することで目標概念に関連する用語が一般化される．つまり，最初に，説明木での根(root node)において，目標概念cup(X)が，ルールliftable(obj1) ∩stable(obj1) ∩open-vessel(obj1) →cup(obj1)上で回帰される．この結果，liftable(X) ∩stable(X) ∩open-vessel(X)はcup(X)を推論するための十分条件であると決定される．以下，liftable(X), stable(X), open-vessel(X)に関しても同様の目標回帰が行なわれ，以下に示すcup概念の一般的定義(構造的側面で記述するという操作性規範を満足する概念記述)を得る．

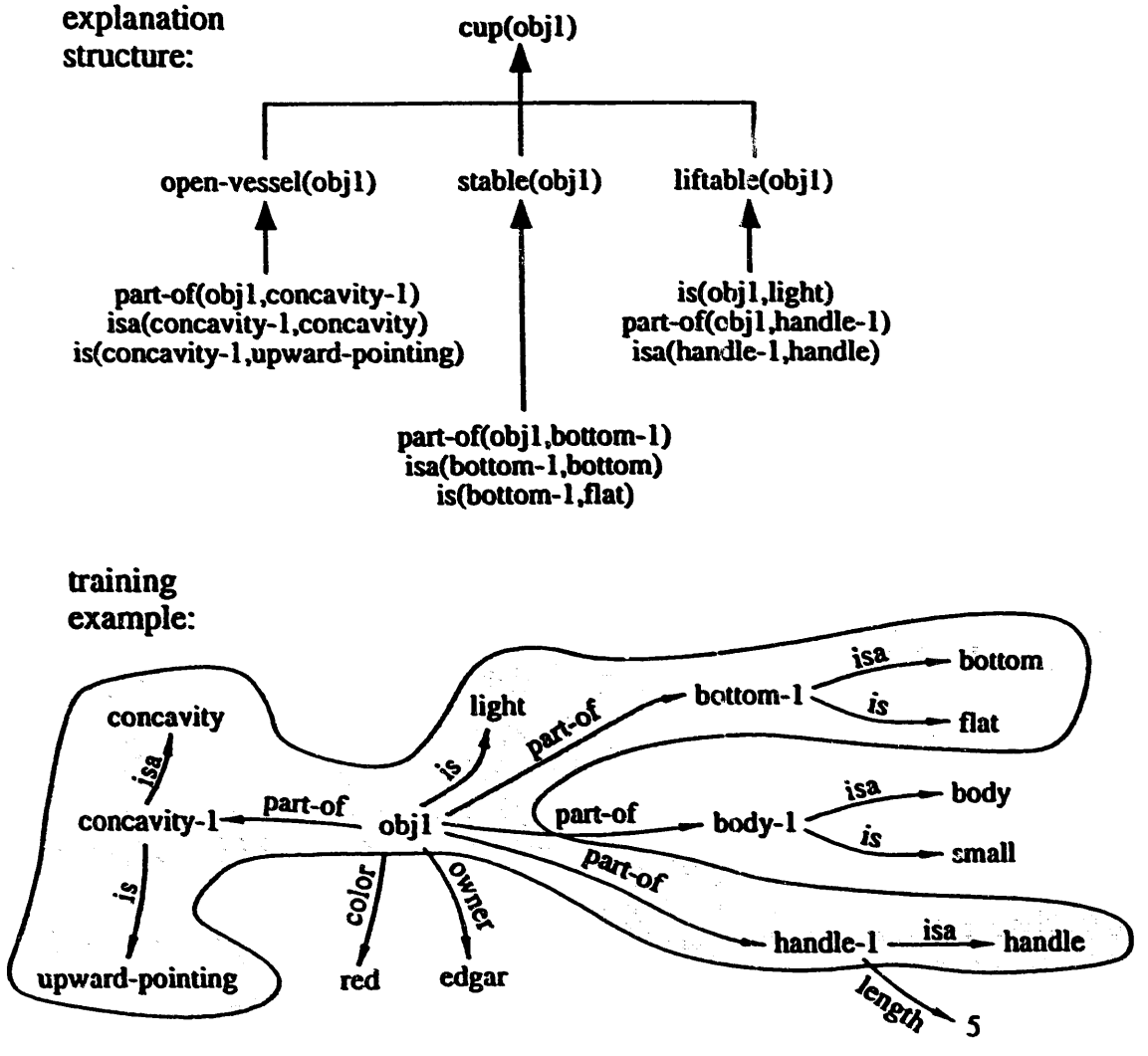
explanation structure:
cup(obj1)
open-vessel(obj1)
stable(obj1)
liftable(obj1)
part-of(obj1,concavity-1)
isa(concavity-1,concavity)
is(concavity-1,upward-pointing)
part-of(obj1,bottom-1)
isa(bottom-1,bottom)
is(bottom-1,flat)
is(obj1,light)
part-of(obj1,handle-1)
isa(handle-1,handle)
training example:
concavity
isa
concavity-1
part-of
obj1
is
light
part-of
bottom-1
isa
bottom
is
flat
part-of
body-1
isa
body
is
small
part-of
handle-1
isa
handle
length
5
is
upward-pointing
color
red
owner
edgar

図 3.1: カップの説明木

part-of(X,XC)
∩ isa(XC,concavity)
∩ is(XC,upward-pointing)
∩ part-of(X,XB)
∩ isa(XB,bottom)
∩ is(XB,flat)
∩ part-of(X,XH)
∩ isa(XH,handle)
∩ is(X,light) →cup(X)

Mitchellらは文献[13]において今後の研究課題として次の3つを挙げている.

1. 不完全問題: 領域知識が不完全(incomplete)または取り扱い難い(intractable)または矛盾のある(inconsistent)ような場合, どのように説明を構築するのか, もしくは, 誤った説明はどのように取り扱うのか, という問題に対処することが望まれる.
2. SBLとの結合: 学習の一方の雄であるSBLと結合することによって, 双方の利点を生かした学習が望まれる.
3. 一般化タスクの定式化:EBG問題の入力がどのようにして与えられるのか, またシステムにおける学習要素の位置付けを明確にすることが望まれる.

### 3.2.2 EBL: An Alternative View

DeJongらは文献[13]で提案された説明ベースのアプローチに対して, いくつかのシステムではそのアプローチ(EBG)は十分であるが, 自分たちのグループが開発しているシステムでは必ずしも十分ではないとして, EBGに代わる一般化手法の枠組を提案した[2].

#### EBGの問題点

文献[2]でDeJongらは, 次のような点においてMitchellらのEBGの問題点を提起した.

1. 説明に基づく学習は説明に基づく一般化よりより広い用語である. 概念の一般化と同様に概念の洗練に説明に基づく学習のアプローチを応用することは可能であり, 望ましいことである. EBGは概念の過剰一般化(over-generalization)を行なう可能性がある. 実際, cupの例では過剰一般化によってバケツまでも認識してしまい, 推論効率が低下してしまった. 過剰一般化を防ぐためには特殊化が必要であり, そのためには説明に基づく学習という例題の概念の一般化と特殊化を包含する広いアプローチが望ましい.

2. MitchellらのEBGの仕様は例題の説明がどのように構成されたかを明記していない．結果となる一般化から最大限の利益を引き出すには，一般化アルゴリズムは説明の派生にまで敏感であるべきである．Mitchellらの EBGの説明の構成には2つの可能性がある．(1) 説明は内部の定理証明器または問題解決器によって外部からの指導なしに構成される．(2) 説明は他の問題解決の振舞いを観察し，解釈することによって構成される．それら2つの可能性は問題解決能力の異なったレベルを要求し，異なった一般化ステップを要求する．なぜなら学習システムの内部計画器によって説明が作られたならば，その説明は最適であると確信しても良いが，他の問題解決器の振舞いを観察することによって説明が構成された場合は，その解や説明は最適であるという保証はないからである．一般化方法は説明の構成が内部計画器によるものか，観察によるものかによって異なるべきである．

3. MitchellらのEBGの一般化手法は貧弱である．EBGの一般化アルゴリズムは縮小目標回帰 (goal regression) を用いている．完全な目標回帰アルゴリズムは非常に複雑で扱い難いため，この選択は極めて正当である．しかし，縮小目標回帰アルゴリズムを用いた一般化では，述語の一般化および構造の一般化ができない．これらの一般化が行なえないことは一種の過少一般化 (under-generalization) であり，これを回避する必要がある．

4. Mitchellらが提案した操作性規範の概念にはいくつかの問題がある．操作性規範は，最終的な操作可能な定義が問題解決器に理解できる語彙で記述されなければならないという意味では非常に重要な概念である．Mitchellらは許容できる語彙に制限をつけることによって回避した．しかし，これには問題がある．まず，操作性規範を記述するプロセス自身が操作可能でないということである．これはどのようにその述語を選ぶのかについてMitchellらが言及していないためである．次に，操作可能な述語は一般には操作可能であるとは限らないことである．述語名からのみの判断では操作可能であっても実際にはその述語の引数によって操作可能でなくなる場合があるからである．

5. 知識のチャンク (chunking) やスキーマ化 (schematizing) は一般化プロセスの中心的な役割として働くべきである．スキーマ間の階層的関係を与えることによって，説明に基づく学習システムは例の代わりとなる説明を効率良く探索することができる．一般化プロセスにおけるスキーマに関するこの推論はMitchellらによって提案された手法の一部ではない．

6. Mitchellらはオブジェクトの一般化において目標回帰 (goal regression) の縮小バージョンを採用した．説明に基づく一般化には完全な目標回帰は不適切であり，それについてはMitchellらは完全に正しい．しかし，目標回帰アルゴリズムでは説明の前件部しか一般化しない．そのため，完全な一般化された概念を得るためには説明中のルールを用いて説明のゴールを一

般化して後件部を作るというパスが必要である. これは効率が悪い. これを克服し, 縮小目標回帰と多数のケースで等しく, さらにより一般化や説明の変更ができる単一化撤回の機能を持ったメカニズムがある.

これらの問題点に対してDeJongらは, スキーマによる一般化, より良い操作性規範, 目標回帰の代案, という3点から回答を提示した. 以下それぞれについて説明する.

スキーマによる一般化

スキーマを学習する説明に基づくシステムにおいては, スキーマの知識は例題を一般化するために使うことができる. このタイプの一般化はMitchellらのEBG手法の部分外であるが, 過少一般化問題を解決することができる. なぜならスキーマはそれ自身一般化された知識のチャンクであるからである.

スキーマに基づく問題解決システムに基礎をなす中心的なアイデアは, システムの知識はあるゴールを果たす関連した知識がお互いにグループ化されるという方法で構造化されているべきである, ということである. このシステム知識のスキーマへの構造化は, 観測された入力の理解を助け, また重要なかつ効率的な一般化を生み出す.

次の物語の例を用いてスキーマによる一般化の有効性を示す. この例から一般的な誘拐概念を学習することを想定する.

> Fredは Maryの父であり, 大富豪である. Johnは Maryに近付いた. 彼女は青いジーンズを着ていた. Johnは彼女に銃をつきつけ, 彼の車に乗るよう言った. 彼は泊まっているホテルに彼女を連れて行き, 部屋に閉じ込めた. Johnは Fredに電話し, Maryを預っていることを告げた. Johnは FredにTrenosで250000ドル払えばMaryを解放すると告げた. Fredは彼にお金を与え, そしてJohnは Maryを解放した.

Johnは Maryをホテルに車で連れていき, 拘禁するために銃をつきつけた. これはJohnにとっては非常に重要なことである. しかしながら, この行為は全ての誘拐で本質的なことではない. Johnが Maryを車に乗せる方法は幾通りも考えられる. しかも車に乗せてホテルに連れていくこと自体も本質的なことではない. なんらかのトリックによってホテルの部屋に入れて捕まえればいいことである. つまり, 一般的な誘拐のスキーマには銃は要求されるべきではない. 銃でつきつけるということは既知のスキーマthreatenによって理解可能である. 更に彼女を捕まえるということは同様に既知のスキーマcaptureによってできる. これらの既知スキーマを用いることによって新しい概念スキーマ, つまり誘拐を学習できる.

### より良い操作性規範

システムが所有するスキーマのためのゴールは操作可能でなければならない．つまり，操作性規範はそのようなゴールである．換言すると，既知のスキーマの具体化による典型的な意味を用いてシステムが簡単に実行できるようなゴールの集合が操作性規範である．これには2つの重要な含みがある．まず，操作性規範はシステムの実行要素で推論できる．それゆえに独立したシステムへの入力であるべきではない．次に，操作性規範は静的ではなく動的である．システムが新しいスキーマを学習したら，そのゴールはそれ以降の説明を構成するために用いることができるように操作可能でなければならない．

### 目標回帰の代案

Mitchellらの目標回帰を用いた一般化にはさまざまな問題があるため，それに代わる一般化アルゴリズムを提供する．その技術は汎用のルールの非具体化されたバージョンを使い，証明木(Mitchellらは説明構造と呼んだ)のバージョンを維持する．そして，SPECIFICとGENERALと呼ぶ2つの独立した代入リストを持つ．SPECIFIC代入を説明構造に適用すると，特定の例のための説明ができ，GENERAL代入を適用すると，一般化された説明が得られる．

特殊化されたステートメントまたは特定の例のゴールと汎用の領域ルール間の単一化はSPECIFIC代入のコンテキストの中で作られる．そして単一化に必要な新しい代入はSPECIFICリストに加えられる．説明構造中の2つの領域ルール間の単一化は，SPECIFIC代入リストを使う時と，GENERAL代入リストを使う時の2回行なわれる．各々の場合，新しく作られた代入が加えられる．このプロセスはGENERALリストが説明構造の正当性を維持するために必要なそれらの代入を持つことを保証する．

この2つのリストを用いて目標回帰アルゴリズムの代案となるアルゴリズムを定義する．$\sigma$をSPECIFIC代入，$\gamma$をGENERAL代入として表すと，一般化手続きは次のようになる．

for 説明構造中に現れる等価な2つの式$e_1$と$e_2$全てについて
  if $e_1$がある領域ルールの前件部である and
    $e_2$が別の領域ルールの後件部である
  then let $\phi = e_1\sigma$と$e_2\sigma$の最汎単一化代入
      let $\sigma = \sigma\phi$ (* SPECIFIC代入の更新 *)
      let $\delta = e_1\gamma$と$e_2\gamma$の最汎単一化代入
      let $\gamma = \gamma\delta$ (* GENERAL代入の更新 *)
  else (* 説明構造の根か葉の場合 *)
      let $\phi = e_1\sigma$と$e_2\sigma$の最汎単一化代入
      let $\sigma = \sigma\phi$ (* SPECIFIC代入の更新 *)

このアルゴリズムがどのように一般化に用いられるかを示すために簡単な例を挙げる．領域知識として，次のようなルールがある．

R1 : hate(A,B) ∩possess(A,C) ∩weapon(C) →kill(A,B)
R2 : depressed(W) →hate(W,W)
R3 : buy(U,V) →possess(U,V)
R4 : gun(Z) →weapon(Z)

そして次のような訓練例が与えられる.

depressed(john)
buy(john,obj1)
gun(obj1)

いま操作性規範は訓練例の各述語が満足しているとし,これから一般的な自殺の概念を学習することを想定する.

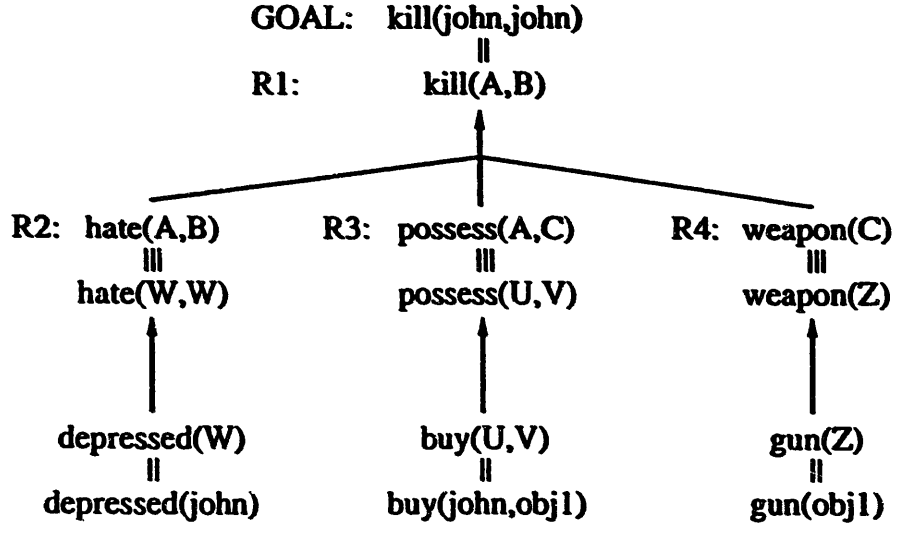


図 3.2: 自殺の例の説明構造と代入

図 3.2にこの例の説明構造とその一般化における代入を示す. この図における最終代入はルールR1からR4へと順番に単一化を行なった結果である. 但し,どのような順番で行なおうとも変数名が異なるだけで同等の代入となる. 表3.3に単一化のトレースおよびどのようにそれらが2つの代入リストに作用するかを示す.

目標回帰の代案アルゴリズムは次のような一般的なルールを生成する.

depressed(W) ∩buy(W,Z) ∩gun(Z) →kill(W,W)

表 3.3: 自殺の例の単一化のトレース

| Unification | SPECIFIC | GENERAL |
|---|---|---|
| R1: | | |
| kill(john,john)=kill(A,B) | john/A, john/B | |
| R2: | | |
| hate(A,B)=hate(W,W) | john/W | W/A, W/B |
| depressed(W)=depressed(john) | | |
| R3: | | |
| possess(A,C)=possess(U,V) | john/U, V/C | U/W, V/C |
| buy(U,V)=buy(john,obj1) | obj1/V | |
| R4: | | |
| weapon(C)=weapon(Z) | obj1/Z | Z/V |
| gun(Z)=gun(obj1) | | |

このようにしてこのアルゴリズムは機能的には目標回帰と同等のルールを生成する. しかしながら, このアルゴリズムには目標回帰より優位な点がある. まず, 一般化プロセスが説明構築のプロセスに統合することができることである. 目標回帰アルゴリズムでは一般化プロセスは説明が構成された後にトレースすることによってなされる独立したプロセスであった. これに対していつでもルールを説明に加えることができ, SPECIFIC と GENERAL 代入リストは更新される. 次に, 説明の完全な一般化ができることである. 目標回帰では, 前件部の一般化に過ぎなかったため, 完全な説明の一般化を得るために説明構造中のルールを一般化された前件部に再適用して構成しなければならなかった. この手法が計算機的な複雑さの点から実際に EBG より効率的であるかどうかは明らかではないが, より簡潔なアプローチであると感じられる.

更に付加的な機能として, GENERAL 代入リストから単一化を撤回することができることが挙げられる. これは操作性規範が証明木のより高いノード, つまり葉でないところにあった場合に説明の枝刈りができるという重要な機能である. 例えば, 上の自殺の例において, 与えられるアイテムが weapon であるかどうかを決定することが容易であるならば, 武器が gun であることを要求せずに自殺の例を一般化することができる. これは一般化された説明から継承されたルール gun(Z) →weapon(Z) を取り去り, GENERAL 代入リストからその単一化を撤回することによって行なわれる. つまり, GENERAL 代入リストから代入 Z/V を撤回する. 新しく一般化されたルールは以下のようになる.

$$\text{depressed(W)} \cap \text{buy(W,V)} \cap \text{weapon(V)} \rightarrow \text{kill(W,W)}$$

### EBGの新しいバージョン

これらの回答に基づいてDeJongらはMitchellらのEBG問題とEBG手法を改良した.

DeJongらは問題解決システムとして,スキーマベースの問題解決システムを前提とし,このコンテキストのもとでEBGのさまざまな問題点を克服することを試みた.特に,観察からの学習の可能性を含み,述語の一般化や説明構造の自由度も考慮した.また明白な入力としての操作性規範を削除した.表3.4にDeJongらによって改良された説明に基づく一般化問題を示す.この新しい

表3.4: DeJongらによる説明に基づく一般化問題

---

Given:

- 領域知識(domain theory): 領域知識は3つの部分から構成される.第1に,世界におけるオブジェクトのタイプ仕様とそれらの特性.第2に,ある特性や関係を他の特性や関係から推論するための推論ルール.第3に,問題解決オペレータのライブラリと既知の汎用スキーマ.それらのスキーマは以前に学習したか,手作業で作られたものである.
- 目標(goal): 目標状態の一般的な仕様.一般に,目標は不完全な世界の状態であり,操作可能な用語で記述されていない.
- 初期状態(initial world state): 世界におけるオブジェクトの仕様とそれらの特性.
- 観測オペレータ/状態系列(observed operator/state sequence): 目標の具体化をなす専門家によって行なわれた低レベルオペレータの観測系列.いくつかの状況では,オペレータが見つからないかもしれない.そのような場合,それらのオペレータは入力で与えられた最終状態から推論されるべきである.

Determine:

- 一般的な方法で目標を成し遂げる新しいスキーマ

---

EBG問題のもとで行なわれる一般化手法を表 3.5に示す.ステップ1では観測された解があればそのオペレータ系列を解釈することによってなぜ目標がそれらのオペレータ系列によって実現されているのかを示し,もし入力がなければシステム自身によって目標が実現される.ステップ2では目標が実現されるに当たって不必要であったオペレータや初期状態を削除する.これらステップ1とステップ2が説明フェーズとなり,以降のステップが一般化フェーズとなる.ステップ3では既知のスキーマの名目だけの具体化を削除する.これは最終的

表 3.5: DeJongらによる一般化手法

- もし観測されたオペレータ入力がなければ,目標を成し遂げる計画を構成するために領域知識を用いる. さもなければ,観測されたオペレータ入力の原因となった全ての解釈を構築するために領域知識を用いる. 各々の場合, GENERALとSPECIFIC代入は維持する.
- 原因として目標をサポートしないオペレータと状態を削除する. 残っている構造は解となる説明である.
- 既知のスキーマの名目だけの具体化を同定し,スキーマの一般的仕様と具体化で使われた特殊なオペレータ間の単一化を撤回することによってそれらを削除する.
- より抽象的な行為または状態を推論することをサポートした行為または状態を取り除く. これはそれらの抽象化の推論をそれらの前件部と一緒に取り除き,説明の残りに対するそれらの単一化の束縛を撤回することによって行なわれる.
- もし説明が観測された解から構成されたなら,より効率の良い方法で成し遂げられる副目標を探す. 説明中の各々の副目標において,観測で使われた副目標より効率的な副目標を成し遂げるためのスキーマをシステムが既に持っていないか検査する. もしそうなら,この副目標を行なう観測された計画を撤回し,より効率的なスキーマを挿入する.
- 残りの説明構造に結果のGENERAL代入を適用することによって最終的な一般化された説明を生成する.

に得られる一般化された概念に重要でない情報を削除する. ステップ4では目標を実現するために必要最小限のスキーマに一般化する. 例えば, gunならばweaponであるように抽象化する. ステップ5では問題解決が外部から与えられた場合にその説明のある副目標を効率の良いシステムが既知の副目標に取り替える. ステップ6は残った説明構造にGENERAL代入を適用し, 最終の一般化された説明を得る.

文献[2]には, これらの新EBG問題と新EBG手法を用いた例題が掲載されているが, ここでは割愛する.

## 3.3 説明に基づく学習の様相

本節では説明に基づく学習に関するさまざまな様相について述べる. ここで, 説明に基づく学習という用語はDeJongらによって再定義されたEBG問題やEBG手法を指すのではなく, DeJongらが提示したEBGの第1の問題点で用いられたEBGよりより広い概念, 概念の一般化のみならず洗練も含む, を指している. 本論文では以降, 断らない限り説明に基づく学習(EBL)をこの意味で用いる.

### 3.3.1 EBLの歴史

文献[2]においてDeJongらによってEBLという用語が使われ始めたことは既に述べた. 説明に基づく学習のルーツは1970年に遡るとされている. 1983年に開催された機械学習に関するワークショップにおいて, Mitchell, Silver, DeJongらによって, お互いの研究がEBLという概念で統一できる同じ方向性を持った研究であることが確認された. 表3.6にEBL研究の流れを示す[14][2].

EBLの枠組については1986, 1987年頃収束し, それ以降は複数例題の取り扱いや, 操作性規範, 効用問題, 再帰的領域知識, 不完全領域知識についてなどが主な研究課題となっている. Mitchellらが示した今後の課題(3.2.1節参照)は不完全問題以外はほとんど注目されていないのが現状である.

### 3.3.2 EBLに関するトピック

この節では広い意味での説明に基づく学習EBLに関するトピックスを簡単に紹介する.

#### Prolog-EBG

Kedar-CabelliらはPrologによってEBGのインプリメンテーションを提供した[15]. そのインプリメンテーションでは, 領域知識はpure Prologのプログラムと

[2]文献[14]では1970年から1986年にかけてのEBL研究の流れをまとめているが, いくつかの研究を割愛し, 更にそれ以降の研究を筆者が付け加えた.

表 3.6: EBL の歴史

| 年 | 研究システム名 | 研究者 | 研究機関 |
|---|---|---|---|
| 1970 | POKER | Waterman | Stanford |
| 1972 | STRIPS/MACROPS | Fikes et al | SRI |
| 1973 | HACKER | Sussman | MIT |
| 1978 | BASEBALL* | Soloway | Univ. of Massachusetts/ Amherst |
| 1981 | KIDNAP* | DeJong | Univ. of Illinois/Urbana |
| 1982 | CRITTER | Kelly & Steinberg | Rutgers |
| | LEX2 | Mitchell | Rutgers |
| 1983 | ANALOGY | Winston | MIT |
| | LP | Silver | Univ. of Edinburgh |
| 1984 | GAMES* | Minton | Carnegie Mellon Univ. |
| | MA | O'Rorke | Univ. of Illinois/Urbana |
| 1985 | ARMS | Segre | Univ. of Illinois/Urbana |
| | GENESIS | Mooney | Univ. of Illinois/Urbana |
| | OCCAM | Pazzani | Univ. of California/ Los Angeles |
| | SHIFT* | Ellman | Columbia |
| 1986 | EBL-SOAR | Rosenbloom & Laird | Xerox PARC/Stanford |
| | EGGS | Mooney & Bennett | Univ. of Illinois/Urbana |
| | MORRIS | Minton | Carnegie Mellon Univ. |
| | UNIMEM | Lebowitz | Columbia |
| 1987 | ROE | Hirsh | Stanford |
| | LT | O'Rorke | Univ. of California/Irvine |
| | MetaLEX | Keller | Rutgers |
| | POLLYANNA | Ellman | Columbia |
| 1988 | PRODIGY | Minton | Carnegie Mellon Univ. |
| | IMEX | Braverman & Russell | Univ. of California/ Berkeley |
| | ADEPT | Cohen | Rutgers |
| 1989 | IOE | FLANN & Dietterich | Oregon State Univ. |
| | BAGGER2 | Shavlik | Univ. of Wisconsin |
| | RINCON | Wogulis & Langley | Univ. of California/Irvine |
| 1990 | RECEBG | Letovsky | Carnegie Mellon Univ. |
| | AL-1 | Laird & Gamble | NASA |
| | FAILSAFE-2 | Bhatnagar & Mostow | Siemens/Rutgers |
| 1991 | EBG+ | Hoppe | GMD |
| | STATIC | Etzioni | Univ. of Washington |

して表現され,推論エンジンはメタインタプリタとして実現されている.訓練例はファクトとして与えられ,目標概念は実行時に与えられる.このメタインタプリタやその類似メタインタプリタはProlog-EBGと呼ばれ,非常にコンパクト(図3.3参照)なためさまざまなEBL研究者に用いられている.本論文の第5章で述べる筆者のシステムにおいてもこのKedar-Cabelliらによる Prolog-EBGを用いている.

```
ebg(A,GenA,[A],GenA):-clause(A,true).
ebg((G1,Gs),(GenG1,GenGs),Proof,(Conds1,Conds2)):-
    ebg(G1,GenG1,Proof1,Conds1),
    ebg(Gs,GenGs,Proof2,Conds2),
    append(Proof1,Proof2,Proof).
ebg(A,GenA,Proof,Conditions):-
    clause(GenA,GenB),
    copy((GenA:-GenB),(A:-B)),
    ebg(B,GenB,BProof,Conditions),
    append([A],[BProof],Proof).
```

図 3.3: Prolog-EBG

**EBG = PE?**

van Harmelen と Bundy は彼らの論文[16]においてEBGアルゴリズムは機能プログラミング,より最近では論理プログラミングの世界においてプログラムの最適化に用いられるテクニックであるPE(Partial Evaluation)アルゴリズムと定義や根本的なアルゴリズムにおいて本質的に同等である,と主張した.実際,EBGとPEのアルゴリズムにはほとんど差がなかった.しかし,EBL研究者らは彼らのその主張をほんの2,3行の言葉で一蹴した.それは,EBLにおける訓練例の重要さを割り引いて考えている,EBLは経験から学習するための手法を意図しているという事実を見逃している,というものであった.

## 3.4 結言

本章では,説明に基づく学習に関して,この研究分野の方向性を最初に整理したEBGとそれを発展させたEBLをその枠組から紹介し,またさまざまな様相について述べた.DeJongらが指摘したようにMitchellらのEBGには不備な点が多いが,しかしDeJongらのEBLも彼らの開発していたシステムの上に説明に基づく学習の枠組を展開しようとしたため,特殊な枠組に陥っている.現在では説明に基づく学習は,概念の一般化と洗練のための枠組と位置付けられている.

# 第4章

# インタビュードライバとしてのEBL

## 4.1 緒言

知識獲得の有効な手法にインタビューがある. しかし, インタビューを用いた従来の知識獲得システムでは, 獲得したい知識全てについて専門家に質問を行なっており, 専門家にとって煩わしい質問, 冗長な質問, 答えるのが困難な質問が往々にしてあった. このため, 獲得した知識の診断や実際に問題解決を行ないながら, インタビューによって獲得するシステムの研究がおこなわれた[17][18]. これらのシステムでは, 問題解決器が不足する知識を検出することを契機として, インタビューモジュールが駆動される. このようなインタビューシステムの駆動源のことをドライバと呼ぶ.

問題解決器をドライバとしたシステムは, 実際に問題を解きながら, そのときに必要な知識やその問題解決の思考過程に沿って知識を獲得しようと試みるため, 専門家にとって答えやすい質問を生成するという意味で, 良い質問を生成することができた. しかし, 設計などの合成型のタスクは一般にタスク自身が複雑なため, 要所要所で質問をしなければならないため, 依然として煩わしい質問や冗長な質問がある問題が残り, この問題を解決することが望まれている.

本章では, すでに存在する例題を学習し, 学習ができなかったときにのみ質問を行なうことによって, 専門家を煩わせる質問や冗長な質問を減少することができるというアイデアのもとで, 学習機構に説明に基づく学習(EBL)の枠組を用い, EBLをドライバとしたインタビューシステムを提案する.

4.2節では, インタビューシステムを概観する. ドライバなしのインタビューシステムから問題解決器をドライバとしたインタビューシステムへの遍歴を示す代表的なシステムを紹介する.

4.3節では, 本章のアイデアを示し, 学習機構をドライバとすることの意義を述べる.

---

関連論文[A-1][B-1][B-2]

4.4節では,本章のアイデアを基に開発したシステムのドメインである油圧回路について概説し,簡単な実際の回路例を示す.

4.5節では,本章のアイデアを基に開発した設計知識獲得システムについて述べる.

## 4.2 インタビューシステム

インタビューを用いた知識獲得システムは,ROGET[19], MORE [20]に代表される知識ベースの雛型からの逐次インタビューによる獲得システムから,MOLE[18], I²S/D[21]に代表される問題解決をドライバとしたインタビューによる獲得システムへとその形態は変化してきた.前者のインタビューシステムは専門家から円滑に知識を獲得するためにさまざまな手法を用いたが,残念ながら専門家から十分に知識を引き出すことができなかった.

このような失敗から,インタビューシステムの研究者らは獲得した知識を用いて実際に問題解決を行ない,その過程で不足した知識を獲得したり,誤っていた知識を修正したり,必要となる知識を獲得した.更にその問題解決は専門家が実際に行なっているタスク過程に準じているため,専門家も実際に問題解決を行なっているような状況になるため,インタビューも円滑に進むことがわかった.

本節ではこれら前者の代表的なシステムであるMOREとその発展形であり,後者の代表的なシステムとなったMOLEについてその枠組と基本思想について概観する.

### 4.2.1 MORE

MORE[20]は,石油発掘に関連して掘削泥水から地層を分析するシステムの考察から開発された診断型問題に限定された知識獲得支援システムである.

MOREは,3段階の洗練化操作を経て,ルールベースを構築していく.まず,専門家との対話を通して知識ベース第1版を構成する.この知識ベースは表4.1に示す5つの実体からなるドメインモデルとして構築される.次に8つの戦略に基づいて,専門家にインタビューを行ない,知識ベース第1版を洗練し,第2版を構成する.この戦略は,ドメインモデルを一般的にどのように洗練できるかを示しており,ドメインに依存しないものである.最後に知識ベース第2版をルールベースに変換し,専門家に確信度を尋ねて,知識ベースの構成を終える.

この3段階のうち,知識ベース第1版は,仮説,兆候,条件およびそれらの関係を次々に専門家に尋ねることで構成される.また最後に行なうドメインモデルのルールベースへの変換は機械的に行なうことができ,確信度の付加は,簡単な質疑応答でできる.したがってインタビューという観点からはドメインモデルの洗練が最も興味が持たれる.

ドメインモデルを洗練する8つの戦略を表 4.2に示す.また表中,例として

表 4.1: MOREのドメインモデルの構成要素

- H(仮説): 診断の結果となる事象.
- S(兆候): 仮説から生じる事象および状態.
- C(条件): 仮説の兆候とは言えないが, 診断に深く関わりあう事象および状態であり, 以下の5種類のものがある.

  FC(frequency-condition): HのSと言うよりは, Hに直接関連する事象および状態.

  T(test): Sを観察するための手続き.

  TC(test condition): Tの環境.

  SA(symptom-attribute) Hに対するSの詳細な記述.

  SC(symptom-condition): Sの発生条件.

- Link: HとSとCを関連付け
- Path: HとSの特別な関連付け

挙げられているドメインモデルを図 4.1に示す. MOREの戦略は, 本質的にはネットワーク表現されたドメインモデルに対して, ノードおよびリンクの付加あるいは修正するためのメタレベル知識と言える. 戦略をドメインモデルに適用することにより, 疑似的に診断を試行することができる.

MOREの特色は, ドメインモデルという中間的な知識ベースを介して, 知識獲得(インタビュー)の戦略を検討した点である, というところに大きな価値がある. MOREの適用事例としては, ディスク故障診断, コンピュータネットワーク故障診断, 回路ボード故障診断などがある.

### 4.2.2 MOLE

MOLE[18] は MORE の欠点を克服するために構築されたシステムである. MOREは, 8つの質問戦略を用いてドメインエキスパートにインタビューを行なった. しかし専門家は多くの場合, MOREの質問に答えることや支持値を割り付けることができなかった. MOLEは専門家に対する質問をできるだけ少なくし, その代わり実際に診断を繰り返しながら専門家と対話することによって知識を引き出すことを試みた. この方法は, 実際に診断を行なう状況に置かれることによって, 専門家は容易に自身の知識を述べられるだろうという考えに基づいたものである.

MOLEのドメインモデルは表4.3に示す5つの実体からなるネットワークで

表 4.2: MOREのドメインモデル洗練のための戦略

- 仮説の区別: 兆候を獲得するための戦略である. 全く同じ兆候を持つ仮説のペアに対して, それらを区別するための兆候を専門家に尋ねる.
- パスの区別: 兆候を獲得するための戦略である. ある兆候Sから, 仮説H1と仮説H2の両方にパスが存在すれば, ひとつの仮説にだけ関連する兆候を専門家に尋ねる.
- パスの分割: 兆候を獲得するための戦略である. ある仮説Hから, ある兆候Sへのパス上に存在する別の兆候を専門家に尋ねる. ただし, HとS間の因果関係が弱いときに限られる.
- 兆候に関連した条件: SCを獲得するための戦略である. 兆候の発生に影響する事象を専門家に尋ねる.
- 兆候の区別: SAを獲得するための戦略である. ある兆候Sが複数の仮説$H_1$~$H_n$と関連しているとき, 各仮説が認別されるようなSの属性を専門家に尋ねる.
- 兆候に関連したテスト: テストを確認するための戦略である. 兆候の存在確率を高くする手続きなどを専門家に尋ねる.
- テストに関連した環境: テストの使用環境を獲得するための戦略である. テストの信頼性に影響する条件を専門家に尋ねる.
- 仮説に直接関連する条件: FCを獲得するための戦略である. 他の兆候とは無関係に仮説に直接関連する条件を専門家に尋ねる.

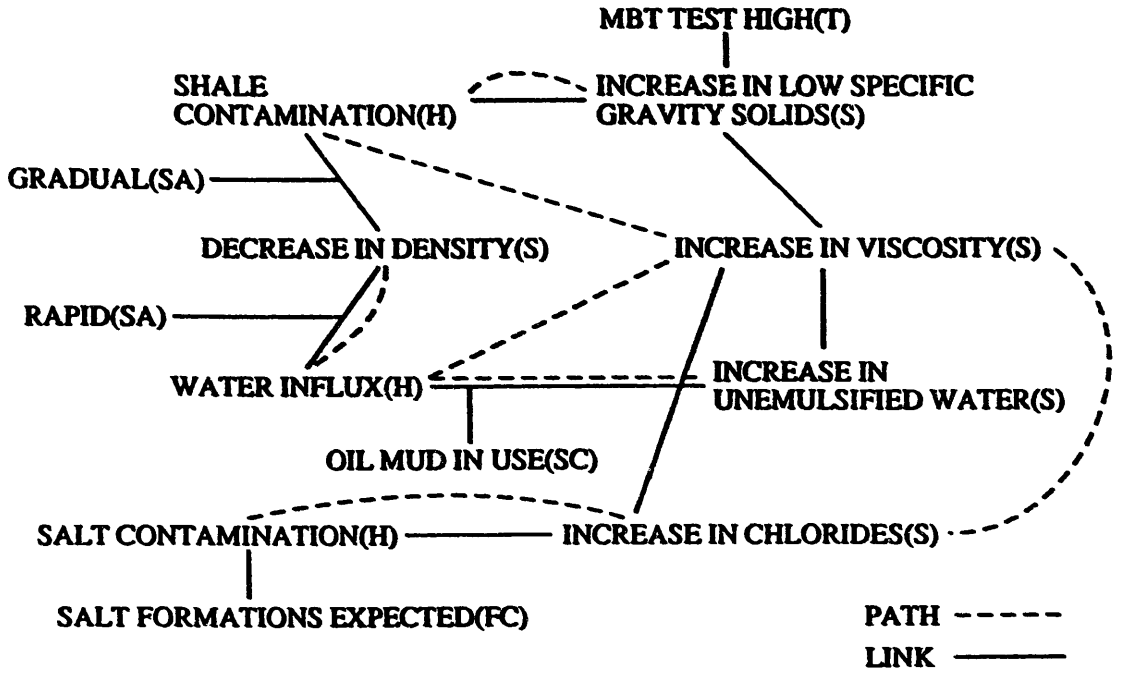


図 4.1: MOREのドメインモデルの1例

ある. このドメインモデルの構築には, 初版ドメインモデルの構築とそのドメインモデルの対話的な洗練のプロセスがある. 初版ドメインモデル構築は仮説とそれに関連するイベントを専門家に尋ねることによって行なわれ, その後にこのドメインモデルの洗練が行なわれる. この洗練には静的解析と動的解析の2つの過程がある. まず, 静的解析により初版ドメインモデルの明確化を行ない, 弱点を発見し解決策を助言する. 次に, 動的解析により, ドメインモデル中の欠落知識や誤り知識を発見し解決策を助言する.

動的解析はMOLEに実際に診断を実行させ, その結果を専門家の診断結果と比較することで行なう. もし, 両者に相違があれば, MOLEは表 4.4に示す洗練戦略に基づいて, 以下のことを支援する.

- 欠落知識の発見
- 支持値の修正
- ドメインモデルの再解釈

MOLEの特色は, 診断システムの診断結果をフィードバックすることにより, 知識ベースを繰り返し洗練できることである. このシステムの適用事例としては, 圧延機の故障診断, Micro-VAXのチューン問題, 発電機の故障診断などがある.

表 4.3: MOLEのドメインモデルの構成要素

- H(仮説): 診断される問題の原因・説明.
- S(兆候): 仮説から生じる事象・状態.
- PC(事前条件): 仮説の生起確率に影響を与える条件.
- QC(指示条件): 仮説に対する兆候や事前条件の指示に影響を与える条件.
- Link: 実態を関係付けるために用い, covering, circumstantial, qualying の 3 種類がある.

表 4.4: MOLEのドメインモデル洗練のための戦略

- 支持値変化させることによって, 専門家の診断に到達できる場合.

  i) 仮説に関する条件(PCの獲得).
  仮説の支持を上げる(下げる)必要があって, その仮説に強いネガティブ(ポジティブ)の事前条件がない場合, その事前条件を尋ねる.

  ii) 背景条件(QCの獲得).
  仮説の支持を上げる必要があって, その仮説に強いネガティブの支持がある場合は, そのネガティブの効果を防げる背景条件を尋ねる.

  iii) 兆候の区別(QCの獲得).
  兆候の支持を仮説間で移動する必要のある場合は, 識別条件を尋ねる.

  iv) 支持値の修正とネットワークの再解釈.
  専門家が何一つ新しい情報を与えられなかった場合は, その信頼性に基づいて, どこを修正または再解釈するか決定する.

- 支持値を変化させても, 専門家の診断に到達できない場合.

  v) 兆候の説明(Hの獲得).
  兆候を説明するのに必要なため, 棄却されるべき仮説が棄却できない場合, その兆候に別の仮説がないか尋ねる. もし, そのような仮説が与えられなければ, その兆候の信頼性を下げる.

  vi) 仮説の区別(Sの獲得).
  棄却されるべきでない仮説が棄却されている場合は, その仮説が説明するが今のネットワーク中にはない兆候がないか尋ねる.

## 4.3 学習とインタビューの融合

一般に,あるタスクドメインに関する知識をインタビューによって専門家から獲得する作業は膨大な時間が費やされるうえ,専門家にとって煩わしい質問や冗長な質問が多くなされる.このため前節のように専門家から円滑に知識を獲得する研究がなされてきた.しかし,問題解決器をドライバとしたMOLEであっても,その適用事例はタスクが比較的簡単な診断が中心であった.もしMOLEを設計などの合成型問題に適用した場合,診断などの分析型問題よりはるかに巨大な空間を探索する必要があり,そのためその探索空間を枝刈りするための知識を頻繁に専門家にインタビューすることになり,専門家を煩わせる結果となるであろう.つまり,診断などの分析型問題を対象とするならば,MOLE等の問題解決器をドライバとしたインタビューシステムで十分であるが,設計などの合成型問題を対象とする際には他の戦略が必要となる.

設計の専門家の経験則が,他の人[1]に理解できる形で表面化しているものに,その専門家の仕事の結果である設計図がある.例えばドメインを油圧回路とするならば,仕事の結果は油圧回路図である.油圧回路の専門家が設計した油圧回路図にはその専門家が用いた専門知識が,機能仕様に対する要素機器の選択と組み合わせの形で盛り込まれており,その油圧回路を構成する要素機器の存在理由を説明することによって専門家の用いた知識を取り出すことが可能である.これはMOLEやI²S/Dのように問題解決を行ないながら専門家にインタビューするのではなく,専門家の実際の仕事の結果を理解,説明しながら専門家にインタビューすることに他ならない.

また,専門家の実際の仕事の結果を理解,説明する機構がそのタスクドメインの基本的な知識を用いているならば,理解,説明に失敗したところは基本的な知識では説明がつかない専門家の長年の経験によって蓄積されたいわば経験則といわれる知識であると考えられる.つまり,この方法は膨大な時間や煩わしい質問や冗長な質問といった問題の解決のみならず,経験則の的確な獲得という利点がある.また理解に失敗した過程や状況を示すことによって専門家に良い刺激を与えることができ,より円滑に知識を獲得することができる.

インタビュアーの理解過程を説明構造により解析するために第 3章で述べたEBLを用いる.EBLは領域知識と呼ばれるドメインに関するルールと事実が与えられ,この領域知識を用いて例題を説明,一般化する枠組である.ここで領域知識について少し考察する.領域知識はドメインに依存し,タスクに依存しない知識である.つまり,あるドメインに関する領域知識を用意すれば,そのドメインの種々のタスクにおいて,その領域知識は用いることができる.しかし,領域知識はタスクに特化していないため,その処理効率は悪い.従って,エキスパートシステムの知識ベースに領域知識を用いることは,推論効率の点で好ましくない.むしろ知識獲得システムが領域知識を用いて,タスクに依存した知識を獲得,用意すべきである.

[1]ある程度そのタスクドメインに対して知識を持っている人

一方, EBLはあるタスクと例題を与えて, 領域知識をそのタスクに依存した知識に再構成する枠組である, と言える(Fig. 4.2参照). このようにEBLを用い

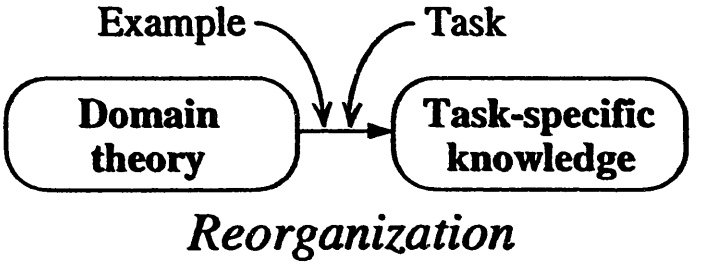


図 4.2: 領域知識からタスク依存の知識への再組織

て領域知識からタスクに依存した知識を獲得することができる. また専門家の経験則は領域知識がChunk[22]化されたり, 特殊化された知識の集合であると考えられる. 従って, あらかじめ領域知識を備えておけば, それらの経験則の理解は原理的には可能である. しかし領域知識の不備や領域知識外の経験則が用いられていた場合などの原因で理解に失敗することが考えられる. この場合は, その理解過程や理解失敗状況を示しながらインタビューによって獲得を行なう. つまり, この枠組は"理解すること"をドライバとしたインタビューシステム, 換言すると, 学習とインタビューを融合したシステム, となっている.

## 4.4 油圧回路

### 4.4.1 油圧システムの概要

油圧技術の原理となっているのは, 17世紀にフランス人パスカルが発見したパスカルの原理である. この原理を物理学的に言い換えると次のようになる.

> 密閉され, しかも静止した液体の一部に加わった圧力は, 液体のすべての部分にそのまま伝わる

このことは, 図4.3において2つのピストンが平衡していることによって簡単に示される.

油圧の特徴は, 液体エネルギーを伝達する液体として潤滑性に富んだ油を使用していることである. このことによって, 油圧の利用価値が向上し, さまざまな分野に利用されるようになって油圧技術が発展した.

油圧技術の発展の背景となっている油圧の長所を以下に列挙する.

1. 小型装置で大出力が可能
2. 無段階で変速可能

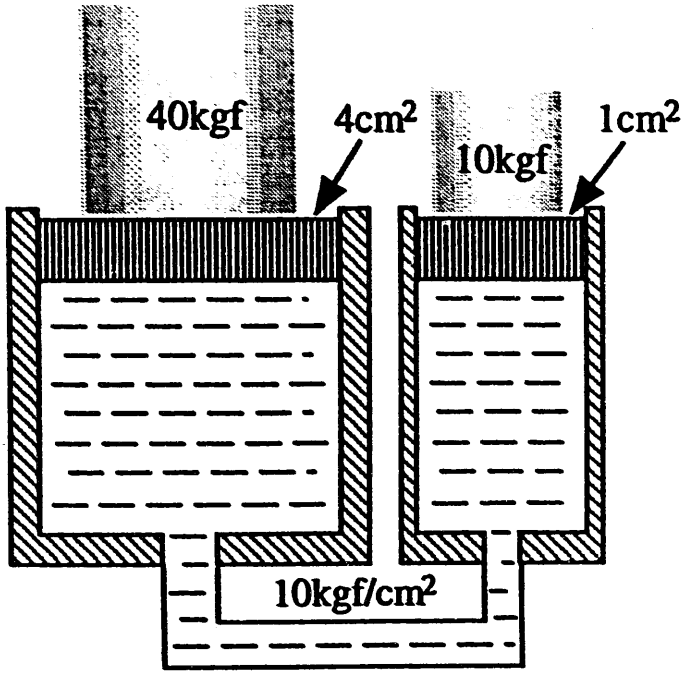


図 4.3: パスカルの原理

3. 力の調整が容易で, 正確
4. 振動が少なく作動が円滑
5. 遠隔操作が可能
6. 過負荷防止が簡単で正確

などがある. 短所としては,

1. 油温変化に伴って速度, 出力が変化する
2. 配管作業が困難
3. 装置の継目から油が洩れやすい
4. 油中に塵や空気が含まれると故障しやすい

などがある. 油圧の代表的応用事例としては,

1. ブルドーザ, フォークリフトなどの建設機械
2. N/C旋盤, 平削盤などの工作機械
3. 自動車における油圧操作のハンドル, ブレーキ
4. 船舶における油圧操作の舵取

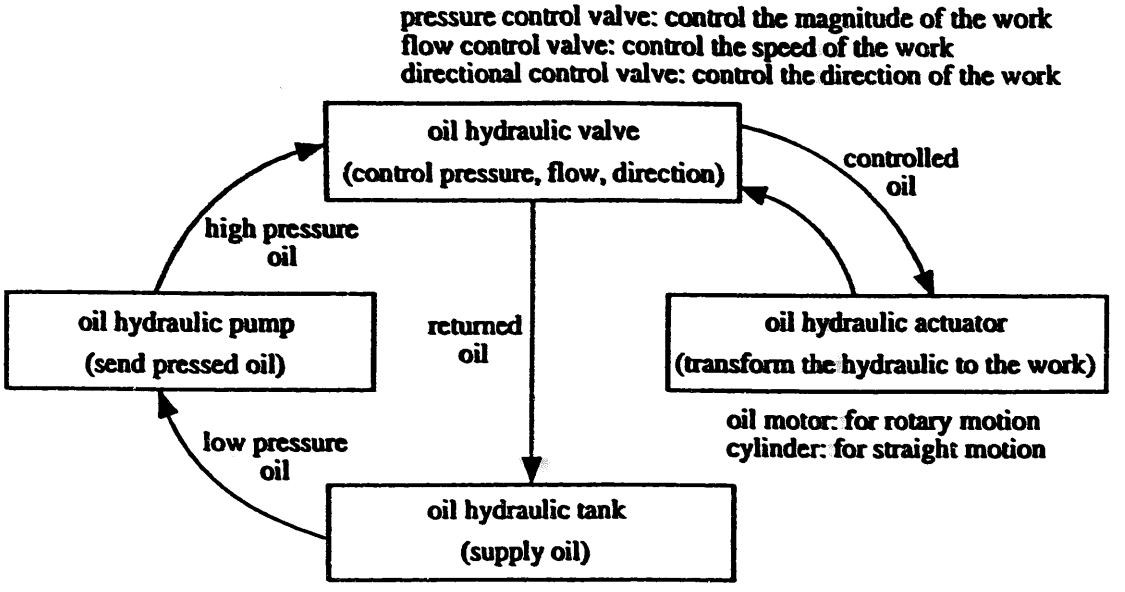


図 4.4: 一般的な油圧機器の構成

などが挙げられる.

図 4.4に一般的な油圧機器の構成 [23] を示し, その動作について述べる. まず, エネルギーの運搬役である油を貯蔵するための油圧タンクから油圧ポンプを使って油を吸い上げる. このようにして, 油圧ポンプは機械のエネルギーを油に与えて油圧のエネルギーに変化させて高圧油を供給する. 油圧エネルギーの持つ圧力の大きさ, 流れの方向, 流量は油圧バルブを使用して制御を行なう. このとき, 仕事の大きさを決めるためには圧力制御弁を用い, 仕事の早さを決めるためには流量制御弁を用い, 仕事の方向を決めるためには方向制御弁を用いる. 制御時生じる余分な圧油は戻り油として油圧タンクに戻される. そして, 制御された油を油圧アクチュエータに流し油圧エネルギーを仕事の速さ, 仕事の大きさに変えてアクチュエータが仕事を行なう.

### 4.4.2 油圧回路の例

ここでは, 簡単な油圧回路の例を示し, 図 4.4の一般的な油圧機器の概要と比較して説明する. 図 4.5に油圧回路の例を示す[2]. このような図面は一般に油圧回路図と呼ばれる.

ここに示した油圧回路図は, 基本回路, 主要要素機器などを JIS 規格に決められた図記号を用いて表し, すべての機器の内部流路を含めた全回路の流路を示すものである. この油圧回路図を用いれば, 油圧システムの制御方法, 機能を簡単に表現することができる. 図 4.5の油圧回路図は, 油圧を用いてテーブ

[2]テーブル駆動回路

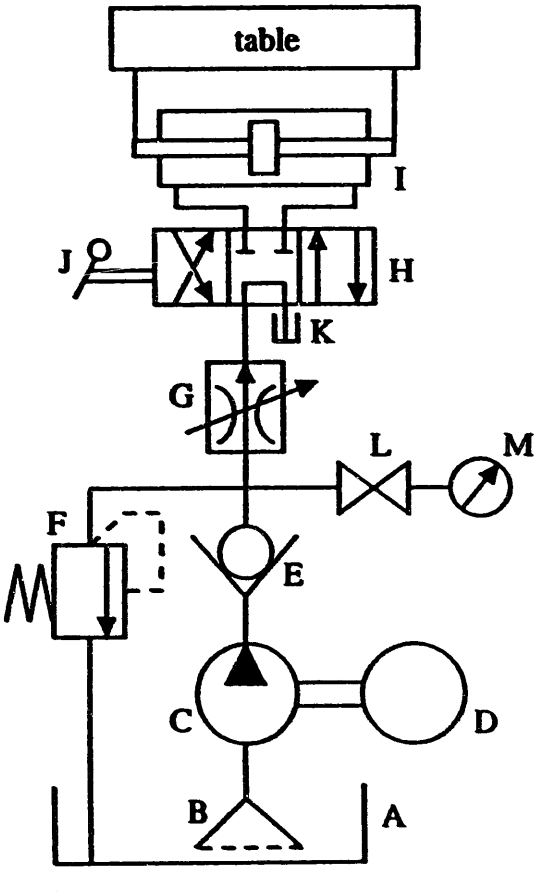


図 4.5: 油圧回路図

ルを往復させるある機械のものである.

図4.4の油圧タンクに相当するものが図4.5のAである. 図4.4の油圧ポンプに相当するものが図4.5のCである. 図4.4の油圧バルブに相当するものが図4.5のE, F, G, Hである. 特に油圧バルブは圧力を制御する油圧バルブ, 流量を制御する油圧バルブ, 方向を制御する油圧バルブからなっている. それぞれ圧力を制御する油圧バルブがFで, 流量を制御する油圧バルブがGで, 方向を制御する油圧バルブがE, Hである. 図4.4の油圧アクチュエータに相当するものが図4.5のIである.

以下, 油の流れを追いながら図4.5の回路図を説明する. まず, 油は油圧タンクAからモータDに駆動される油圧ポンプCによって吐出される. 次に, 油は油圧ポンプCへの逆流を防止するためのチェックバルブEを通り, テーブルの速度を決定するための流量制御弁Gを通ってテーブルの方向を決める方向制御弁Gを通過する. そして, 油はシリンダJに入り, 出口から押し出された油は油圧タンクKに戻る.

このような一連の油の流れによってテーブルが動作することをこの油圧回路図が示している. その他の機器の働きを述べると, 油圧フィルタBは, 油中の塵, 異物を除去する. レバーJはテーブルの方向操作を行なう. リリーフバルブは回路圧を一定に保つ. また, シリンダの出力を変えることもできる. 圧力計Mは圧力を知る. ゲージコックLは圧力計Mの保守を行なう.

## 4.5 ISAK

本節では4.3節で述べたアイデアに基づいた油圧回路設計の専門家の経験則を獲得する知識獲得システムISAK(an Interview System for Acquiring design Knowledge)について述べる.

### 4.5.1 システムの概要

ISAK[24][25]は例題からの学習を行なうモジュールUBL (Understanding-Based Learning)と, インタビューによる獲得を行なうモジュールUBI (Understanding-Based Interview)とからなる. ISAKのシステム構成図を図4.6に示す.

ISAKの入力は専門家の設計した回路例とその油圧回路が実現した場合に満たす機能の仕様書であり, 出力は専門家が入力された油圧回路図を設計したときに用いたと考えられる設計知識である. ISAKシステムが事前に所持しておくべき知識は, 学習を行なう際に必要となる油圧回路に関する基本知識と, 専門家にインタビューを行なう際に必要となる質問戦略に関する知識である. 油圧回路に関する基本知識は専門書などから抽出可能であり, 比較的容易に用意することができる.

ISAKの出力は与えられた回路例を設計したときに専門家が用いたと考えられる設計知識であるが, 実際にISAKによって新たに獲得される知識には次

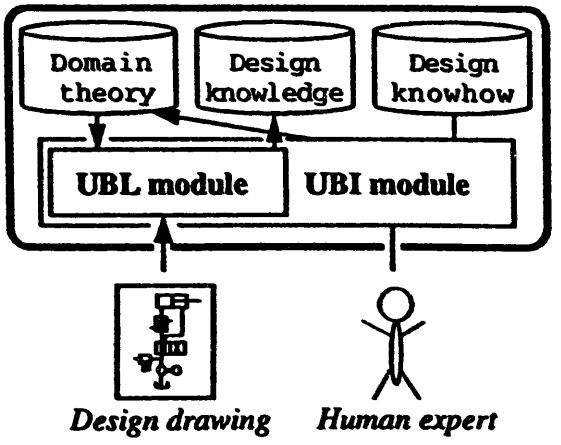


図 4.6: ISAK の構造

の4種類のものがある. (1) 基本知識, (2) 設計ノウハウ, (3) 設計知識, (4) 機能仕様. 基本知識はEBLでいうところの領域知識であり, 本来はシステムがあらかじめ備えているべき知識であるが, 何らかの理由で知識ベースに不備がある時に獲得し蓄積され, 以降の学習とインタビューにおいて用いられる[3]. (2) の設計ノウハウは, 基本知識では説明することが困難な知識であり, 回路例からではなく, インタビューをすることによって獲得される知識である. (3) の設計知識は, 基本知識や設計ノウハウを用いて回路例から間接的に獲得することができる知識である. 基本知識, 設計ノウハウ, 設計知識はともに, "ある機能や目的と, それらを実現する回路のパターンを結びつける知識" である. ISAK はこれら設計ノウハウ, 設計知識を出力し, 基本知識をシステムにフィードバックする. またISAKは専門家にとっては明らかであり仕様書に明記する必要のなかった隠れていた機能仕様も獲得する. 獲得された機能仕様は新たな機能仕様として改めて設計知識の獲得に用いられる.

ISAKは油圧回路例とその回路例の機能仕様が与えられると, 学習モジュールUBLにおいて基本知識を用いて, その回路例が機能仕様の全ての機能を満足していることの理解を試みる. 理解することができた機能については, その機能を満たすために貢献した要素機器を見つけ, 機能と要素機器を結びつける知識を設計知識とする. 理解することができなかった機能や存在する理由を説明することができなかった要素機器があった場合は, それをトリガーとしてインタビューモジュールUBIに制御を移す. UBIはUBLにおける理解失敗の過程や状態と質問戦略知識を用いて専門家から基本知識や設計知識を獲得する. 基本知識がUBIによって獲得されると, 再度UBLで理解を試みる. このように与えられた回路例の全ての機能仕様と要素機器について設計知識を獲得するまで, 上記の処理を続ける.

[3]誤った基本知識からは誤った設計知識が導出される. EBLでいうところの学習記述の正当性を損なうが, ISAKは専門家による指摘を受けることができるため, これを許している.

以下, ISAKシステムの例題からの学習を行なうモジュールUBLとインタビューによる獲得を行なうモジュールUBIについて, そのアプローチを簡単な例を用いて説明する. 現在ISAKシステムはほとんどインプリメントが完了しており, 以下の節ではそのプロトタイプシステムのスナップショットを示しながら説明する. ISAKシステムの推論エンジンにはOPS83を用い, X-window上でC言語を用いてインタフェースを構築し, 推論エンジンと通信している.

### 4.5.2 UBL:例題からの学習

例題からの学習を行なうモジュールUBLにはEBLと同様に次のものが与えられる.

目標概念: 機能仕様に現れるパラメータが満たすべき条件の記述.

訓練例: 油圧回路の設計例.

領域知識: 油圧回路についてのルールと事実, 具体的には要素機器の機能に関する知識が中心となっている. 訓練例が目標概念の例になっていることを説明するために用いられる.

操作性規範: 設計知識の条件部が機能仕様, 実行部が回路の構造と機器, の形式で記述する.

これらを用いて,

ある要素機器または回路の構造の機能と役割の記述

を得る. その方法は図4.7の各ステップよりなる. 与えられた機能仕様をTranslatorが目標概念に変換し, 回路例をSimulatorがシミュレーションを行ない, その結果を用いて目標概念を満足していることをProverが証明し, その証明から目標概念に対して大きな役割を果たしている要素機器または構造をExtractorが取り出す. 取り出した要素機器または構造がその回路に存在する理由を証明結果などを用いてExplainerが説明する. Explainerが構成した説明を基にして設計知識として適切な形式にProducerが再構成する.

UBLの処理を実際の油圧回路を例にとって順に説明する. 図4.8に差動回路を用いた油圧回路図を示す. 差動回路とは, ピストンを往復させるのに, 往路はゆっくりと力をかけ, 復路はあまり力をかけず高速で戻すための回路であり, 図のように油圧アクチュエータであるシリンダの左右両側のポートに同時に圧油を送り, ピストンが両面から受ける力の差で前進することを利用したものである. つまり, ピストンが受ける圧力は左右とも同じであるが, ヘッド側はキャップ側[4]に比べてロッドの断面積だけ小さいため, ピストンを進ませる力も小さくなり, ピストンはキャップ側によって押されてヘッド側に進む. そこでヘッド側の油はピストンに押し戻され, 油圧ポンプから送られてくる圧油と合流

[4]ピストンロッドが出ている側をヘッド, 反対をキャップという

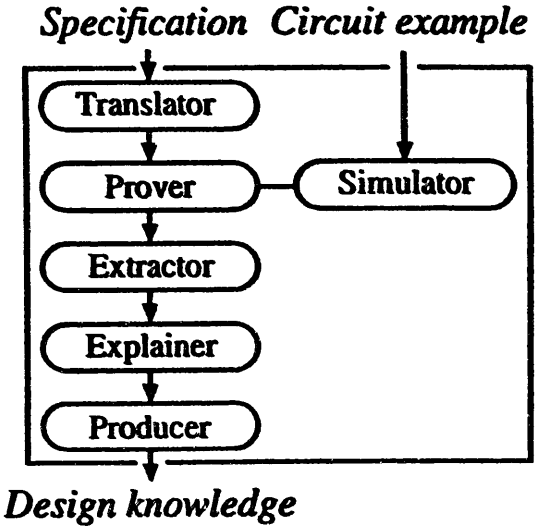


図 4.7: UBL のブロック図

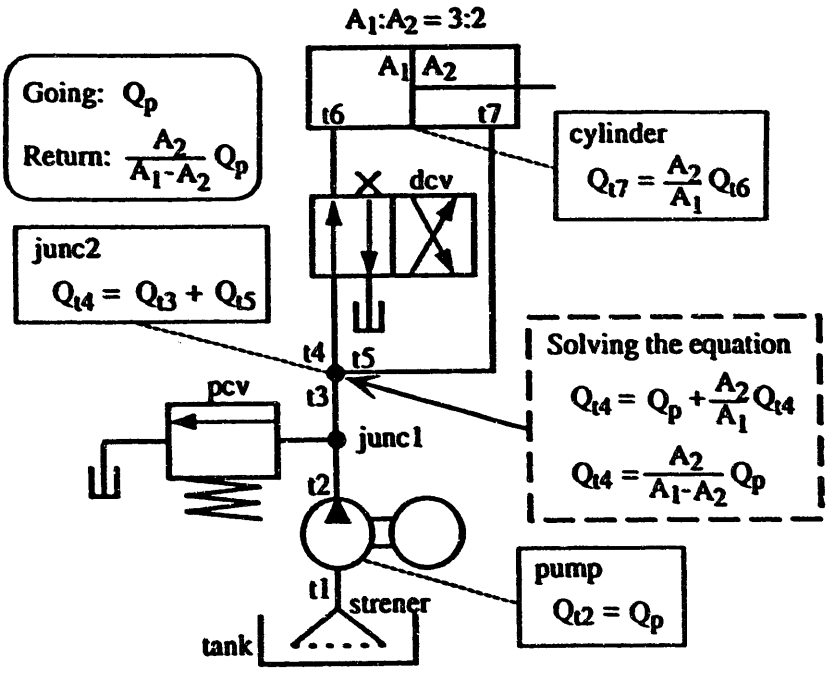


図 4.8: 差動回路の油圧回路図とシミュレーション結果

し，シリンダを更に前進させるため，油圧ポンプからはロッド容積分の油を補充するだけでよいことになる．このようにして，シリンダ送りのスピードは，少ない油で非常に速くすることができる．

油圧回路の機能仕様は一般に油圧回路の非専門家である発注者に理解できる“力”，“速度”といったパラメータによって表現されている．それに対してUBLの内部では“圧力”，“流量”といった油圧回路よりのパラメータを用いている．Translatorはこれらの表現レベルを変換することによって，機能仕様を目標概念の形に変換する．目標概念は以下のような3つの属性を持っている．

**Condition:** 目標となる回路例の外的条件

**Position:** 注目すべき回路内の要素機器

**State:** 注目すべき油圧パラメータとその状態

Translatorは変換ルールを用いて，機能仕様を目標概念の形にする．図4.8で具体例を示す．図中の実線の四角で囲まれた式は要素機器の機能を記述した式であり，破線の四角はシミュレーション中に成立した方程式とその解を示している．角なし四角で囲まれた値はシミュレーション結果であり，行きと戻りの流量をそれぞれ示す．Qは流量パラメータであり，A1とA2は断面積である．また最低限必要なポート(t1やt2)も示してある[5]．図4.8の機能仕様には“戻りの速度が行きの速度の倍”があり，Translatorによって，

**Condition:** dcv left; dcv right

**Position:** cylinder t7

**State:** 戻りの流量が行きの流量の2倍
$(Q_{return}/Q_{going} = 2)$

という目標概念に変換される．Translatorは機能仕様には明記されていない注目すべき回路内の要素機器について，デフォルトのPositionである“cylinder t7”を追加し，速度を流量に変換した．

### Simulator

目標概念が得られると，UBLは与えられた回路例がその目標概念を満足していることを証明するためSimulatorを起動する．Simulatorは，油圧回路を構成する要素機器の機能(基本知識)を参照することによって，要素機器の状態モード[6]を考慮した回路の特性を記述した関係式を導出する．まず回路の有効経路を探索し，その後，圧力と流量を求める．圧力，流量のシミュレーションにはプロダクションシステムを用いている．シミュレーションルールにはキルヒホッフの法則や，パスカルの法則などがある．

---

[5]わかりやすくするために，図中ではポートにユニークな番号を振ってある．

[6]例えば，圧力制御弁がONになっているなど

シミュレーション結果の一般形を示す.

$$f(F_1, F_2, \cdots, F_m) : (C_1, C_2, \cdots, C_n)$$

ここで$F_1, F_2, \cdots, F_m$は要素機器の機能が持つパラメータ(例えば, 圧力, 流量), $f(F_1, F_2, \cdots, F_m)$は$F_i$間に成り立つ関係式であり, $C_1, C_2, \cdots, C_n$は要素機器に固有の制約式である. 図4.8にSimulatorが要素機器の機能を参照しながらシミュレーションする様子を示す. まずpumpの機能を参照し, junc2に達したところでt3の流量は$Q_p$となっている. そしてjunc2の機能から, t4の流量を求めるが, $Q_{t5}$が分からないため, t4の流量を$Q_{t4}$としてシミュレーションを続ける. そしてcylinderの機能を参照し, t7ポートにおいて流量は$\frac{A_2}{A_1}Q_{t4}$となる. t7の流量とt5の流量は等しいため, junc2に達したところで, 再びjunc2の機能を参照し, $Q_{t4} = Q_p + \frac{A_2}{A_1}Q_{t4}$という方程式が立てられる. そしてこの方程式を解くことにより, t4の流量が求まり, 最終的にt7の流量が求められる.

**Prover**

ProverはSimulatorの結果が目標概念を満足していることを証明する. Proverの証明戦略には大きく分けて, 式の評価とマッチングの2つがある. 式の評価とはSimulatorによって得られたシミュレーション結果を評価し, 目標概念のStateを満足しているかどうか調べる証明戦略であり, シミュレーション結果の関係式の部分を使う. マッチングとは, Simulatorによって得られたシミュレーションに要素機器固有の制約が現れ, その制約をシンボリックにマッチングをとる証明戦略であり, シミュレーション結果の制約式の部分を使う. この例では式の評価戦略を採る. 行きの流量と戻りの流量の比をとり, 2倍になっているか調べる.

$$\begin{aligned} \frac{Q_{return}}{Q_{going}} &= \frac{\frac{A_2}{A_1 - A_2}Q_p}{Q_p} \\ &= \frac{A_2}{A_1 - A_2} \\ &= 2 \end{aligned}$$

**Extractor**

与えられた回路例が目標概念を満足していることの証明に成功すると, それを説明するために, 証明に最も貢献した要素機器または構造をExtractorが取り出す. Extractorはシミュレーションプロセス, 証明プロセスをバックトレースすることによってそれらの重要な要素機器を見つける. また, 見つけられた要素機器または構造の回路内における位置情報を付加し, それをCauseとする. Causeは与えられた回路例において機能が満たされている原因であるところの要素機器または構造とその位置情報である. また, Extractorは重要要素機器の取り出しに際して, 必要・不必要なものの選択のヒューリスティックスを用い

ている. このヒューリスティックスの例に,「圧力に関する証明を行なった場合, その証明にほとんど無関係な各要素機器の圧力損失に対して, その圧力損失を呈した要素機器は取り出さない」や「記号処理中に約分されて消えた値を呈した要素機器は取り出さない」などがある. 図4.8におけるバックトレース木を図4.9に示す. Extractorは図のようにバックトレースしてCauseの候補とし

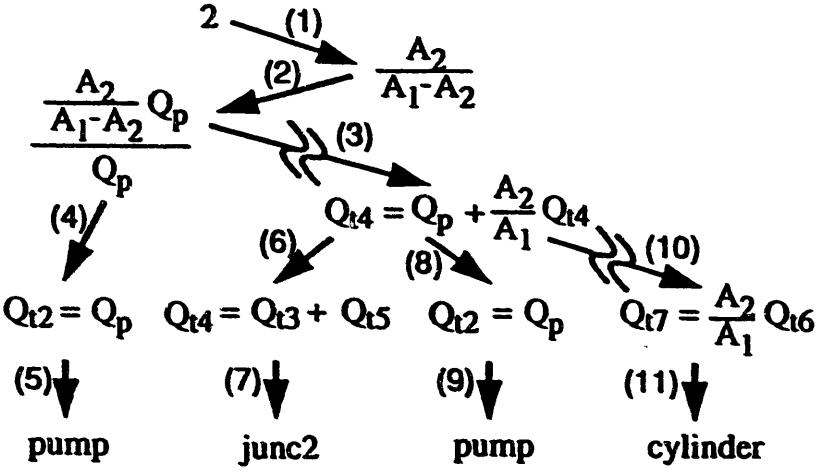


図 4.9: バックトレース木

てjunc2とcylinderとpumpを見つける. しかし, バックトレース木の(2)において$Q_p$が約分されているため, $Q_p$を導出するpumpは2倍という結果に貢献していない. つまり, 上のヒューリスティックス(後者)によって, pumpはCauseとして取り出されない.

### Explainer

Extractorによって得られたCauseと目標概念を用いてExplainerは説明を構成する. 説明の一般形は次のようになる.

Causeによって Conditionのとき Positionにおいて Stateとなる

上記の例では,

Cause: junc2とでループになっている断面積比3:2のcylinder, cylinderとでループになっているjunc2

目標概念: cylinder t7において戻りの流量が行きの流量の2倍

から説明を構成する.

### Producer

設計知識を生成するProducerでは, Explainerによって構成された説明結果を設計知識として適切な形に再構成する. 実際は次のようになる.

> Conditionのとき Positionにおいて Stateとなる回路を作るためには Causeを接続する

つまり,

> cylinder t7において戻りの速度が行きの速度の2倍となる回路を作るためには, junc2とでループになっている断面積比3：2のcylinderと, cylinderとでループになっているjunc2を接続する.

という設計知識が生成される. また, Producerは Translatorと同様の理由から, "圧力", "流量"といったパラメータから"力", "速度"といったパラメータへと変換ルールを用いて表現を変換する. この差動回路の例題の実行結果を図4.10に示す.

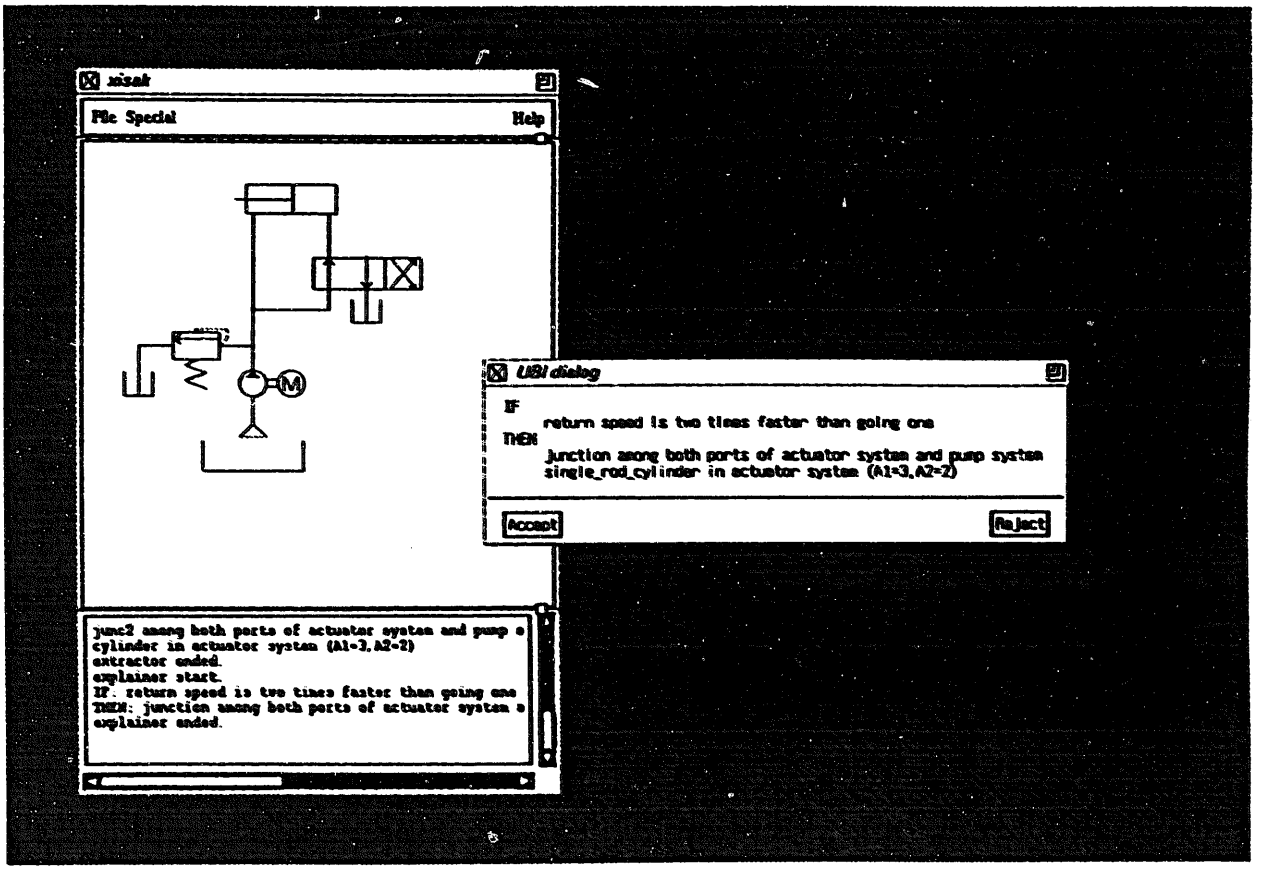


図 4.10: 差動回路の実行結果

### 4.5.3 UBI:インタビューによる獲得

図4.11にUBIの概念図を示す. UBIは図のようにUBLを含んでおり, UBLの

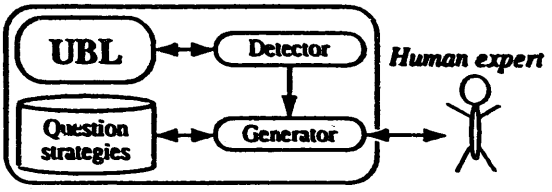


図4.11: UBIのブロック図

理解過程を監視する失敗検出器と専門家とのインタビューを管理するインタビュー生成モジュールとインタビューの質問を生成するための質問戦略が蓄積された知識ベースからなる.

失敗検出

UBIの起動条件は,

1. ある獲得された設計知識が専門家に不適当であると判断された場合
2. ある目標概念に関連した設計知識が獲得されなかった場合
3. ある要素機器に関連した設計知識が獲得されなかった場合

であり, これらが満足されるかどうか監視するため, 失敗検出器は次の3つのことを行なう.

1. UBLで設計知識が生成された場合, それが設計知識として適当か専門家に質問する.
2. UBLの処理が全て終了した後, UBLの全ての目標概念に対して, 設計知識が生成されたか調べる.
3. UBLの処理が全て終了した後, 回路例の要素機器の内で設計知識に関与していない要素機器を調べる.

これら監視タスクを失敗検出器が行なうことによってUBLの学習失敗を検出する.

インタビュー戦略

UBIで獲得する知識は, ISAKで獲得する知識の内, 明示的には記述されていない機能仕様, 不備であった学習モジュールで用いられる基本知識, 基本知識では説明することが困難な設計ノウハウの3種類である. UBIで獲得される機能

仕様は,ISAKの入力である機能仕様と同様にUBLもしくはUBIで設計知識を獲得するために利用される.基本知識はシステムにフィードバックされ,以降のUBLでの学習に利用される.

UBIは起動すると,UBLのどこで失敗が発生したのかを調べ,獲得すべき知識の種類を同定する.その上でその知識に適合した質問戦略を用いてインタビューを行なう.例えば,基本知識の獲得は,TEIRESIAS[26]で用いられた以下のような知識ベースの誤り原因分類に基づいて行なわれる.

1. 実行されたルールの結論に誤りがある.
2. 実行されなかったが,実行されるべきルールが存在した.
3. あるルールが実行されたが,実行されるべきでなかった.
4. ルールベース中に,ルールが不足していた.

獲得する知識はそれぞれ異ったレベルの知識であるため,我々は獲得する知識に応じた質問戦略をデーモンの形で用意した.3種類のデーモンはそれぞれ次のような手順で知識を獲得する.

≪機能仕様獲得デーモン≫

1. 関連する要素機器の確認
2. 一般的な機能との差異の確認
3. 2W1H[7]の獲得
4. 機能仕様への変換

≪基本知識獲得デーモン≫

1. 失敗したモジュールの同定
2. 失敗した原因の同定
3. 基本知識の獲得

≪設計ノウハウ獲得デーモン≫

1. 設計ノウハウの条件部の獲得
2. 設計ノウハウの実行部の獲得
3. 設計ノウハウの合成

UBIによって獲得された知識は次回からのUBLに用いられ,基本的には理解できる回路の範囲が広くなる.得られた知識が機能仕様であれば,例題と同様に類似の回路にその機能仕様が抜けていたならば,その機能仕様で理解することができる.基本知識の場合には,類似の回路において理解に失敗する可

[7]When,What,Howに対応する知識のこと

能性が減少する. 設計ノウハウ知識であれば, 獲得された回路と類似の回路に用いることによって, 基本知識に基づく理解能力を越えた設計知識を獲得することができる.

これらの獲得デーモンのインタビューの進め方を対話例から示す. まず, 図4.8の例で見る. UBLにおいて設計知識が生成されると, UBIの失敗検出器はその知識が設計知識として適当か質問し, 不適当な場合にはその理由を聞く. この例では, 前述の約分に関するヒューリスティックスがないものとし, 吐出量$Q_p$のポンプが取り出されている.

ISAK: "cylinder t7において戻りの速度が行きの速度の2倍となる回路を作るためには, 吐出量$Q_p$のpumpとjunc2とでループになっている断面積比3:2のcylinderと, cylinderとでループになっているjunc2を接続する"が得られましたが, これを設計知識としてよろしいでしょうか?

USER: いいえ.

ISAK: 誤っている理由は, (1)条件部が不適当. (2)実行部が不適当. (3)条件部と実行部の対応が不適当. の内どれですか?(図 4.12参照)

USER: 2.

ここでUBIは基本知識を獲得するため, 基本知識獲得デーモンを起動する. 基本知識獲得デーモンはUBLの実行結果をモジュール単位で専門家に提示することにより, どのモジュールで失敗が発生したか調べ, デバッグを行なう.

ISAK: UBLの取り出し部の失敗のようです. 取り出し部の結果は, (1)吐出量$Q_p$のpump, (2)断面積比3:2のcylinder, (3)junc2, です.

ISAK: この結果が誤っている理由は, (1)取り出す必要がない要素機器を取り出している. (2)取り出す必要がある要素機器を取り出していない. の内どれですか? (図 4.13参照)

USER: 1.

ISAK: 取り出す必要がない要素機器はどれですか?

USER: 1.

ISAK: 1の要素機器を取り出したのはこのようなバックトレース木の結果からです. 1の要素機器を取り出す必要のない理由が木中にありますか? (図 4.14参照)

USER: はい.

ISAK: どのバックトレースですか?

USER: 2.

ISAK: 2ではオペレータ"約分"を使っています. このためですか?

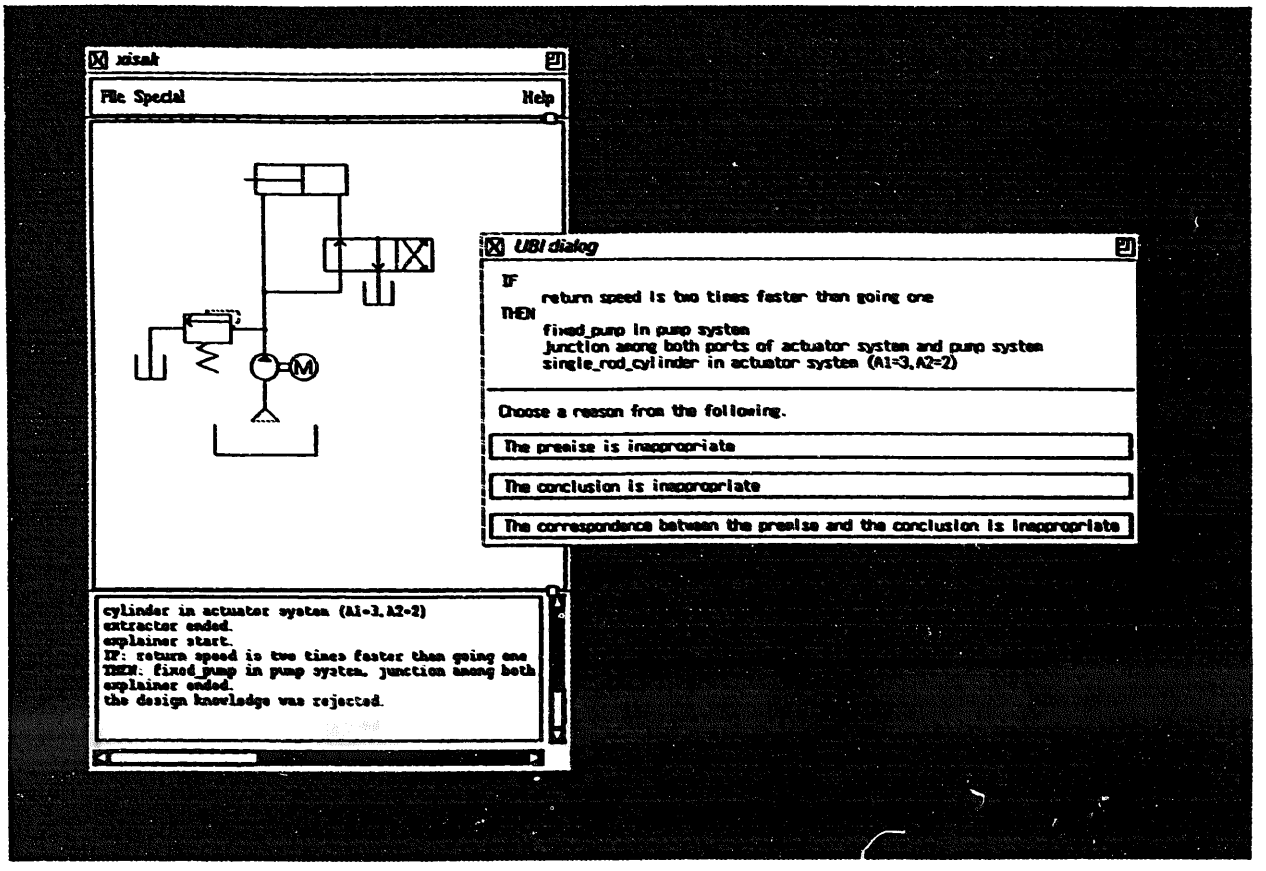


図 4.12: UBIのインタビュー(誤っている理由の質問)

USER: はい.

ISAK: オペレータ"約分"に関する取り出しルールを教えて下さい.

USER: はい.

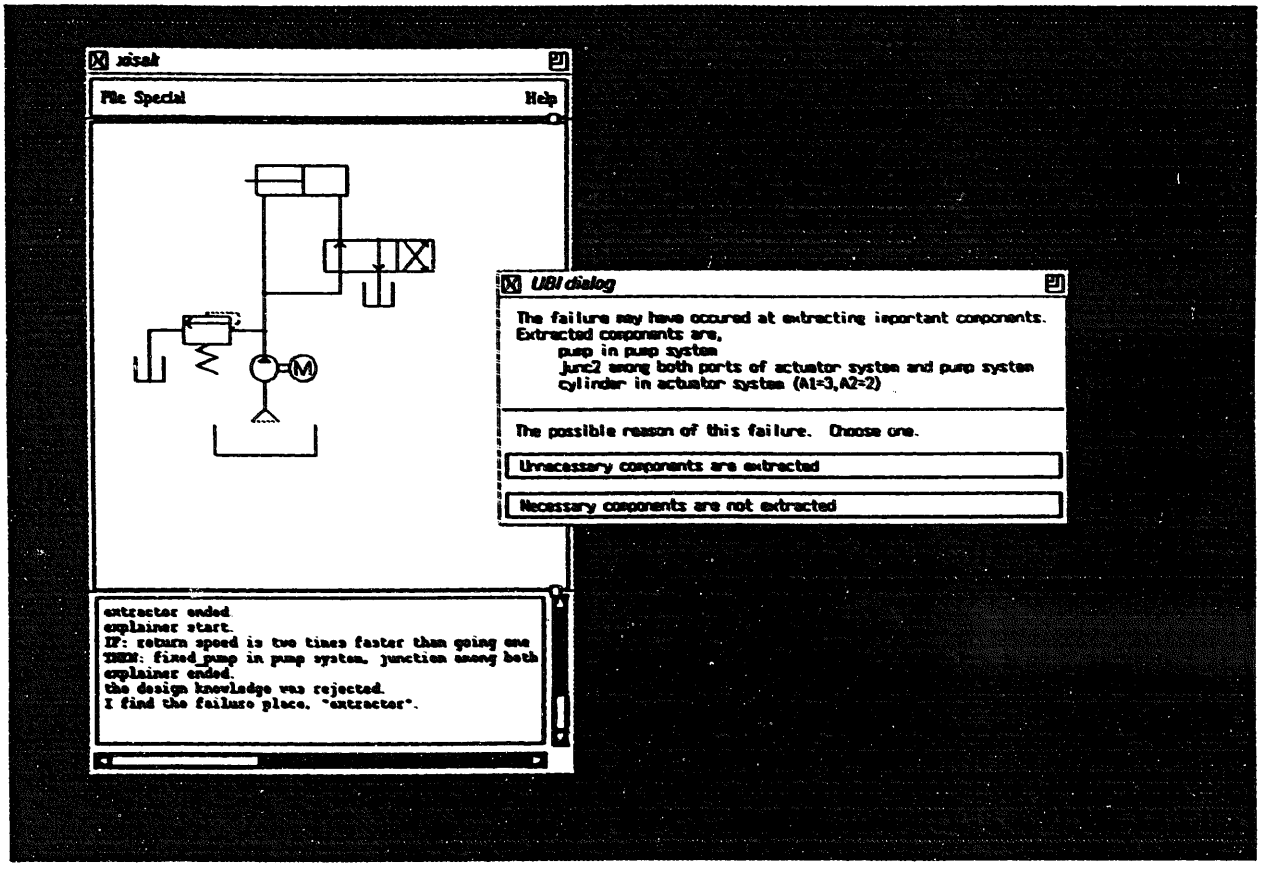


図 4.13: UBIのインタビュー(取り出しが誤っている理由の質問)

このように基本知識獲得デーモンは,不備のルールの獲得やルールの修正を行なう.また,UBLの実行時をシミュレートし,ルールの発火状況,競合ルールの有無,バックトレース木の図などできる限りの情報を提示し,専門家の負担を軽減する.

図 4.15に加硫プレスの油圧回路図を示す.この回路の特徴は,最大加圧の設定を行なうリリーフバルブpcv1以外に,ゴム加硫中に加えられる熱によるシリンダ内の作動油の膨張を逃すリリーフバルブpcv2が接続されていることである.

ISAK: この回路においてリリーフバルブpcv2に関する設計知識が得られませんでした.このリリーフバルブに関する機能仕様は次の内にありますか?
(1)力が一定値まで出せる.(2)上下運動できる.(3)一旦停止できる.

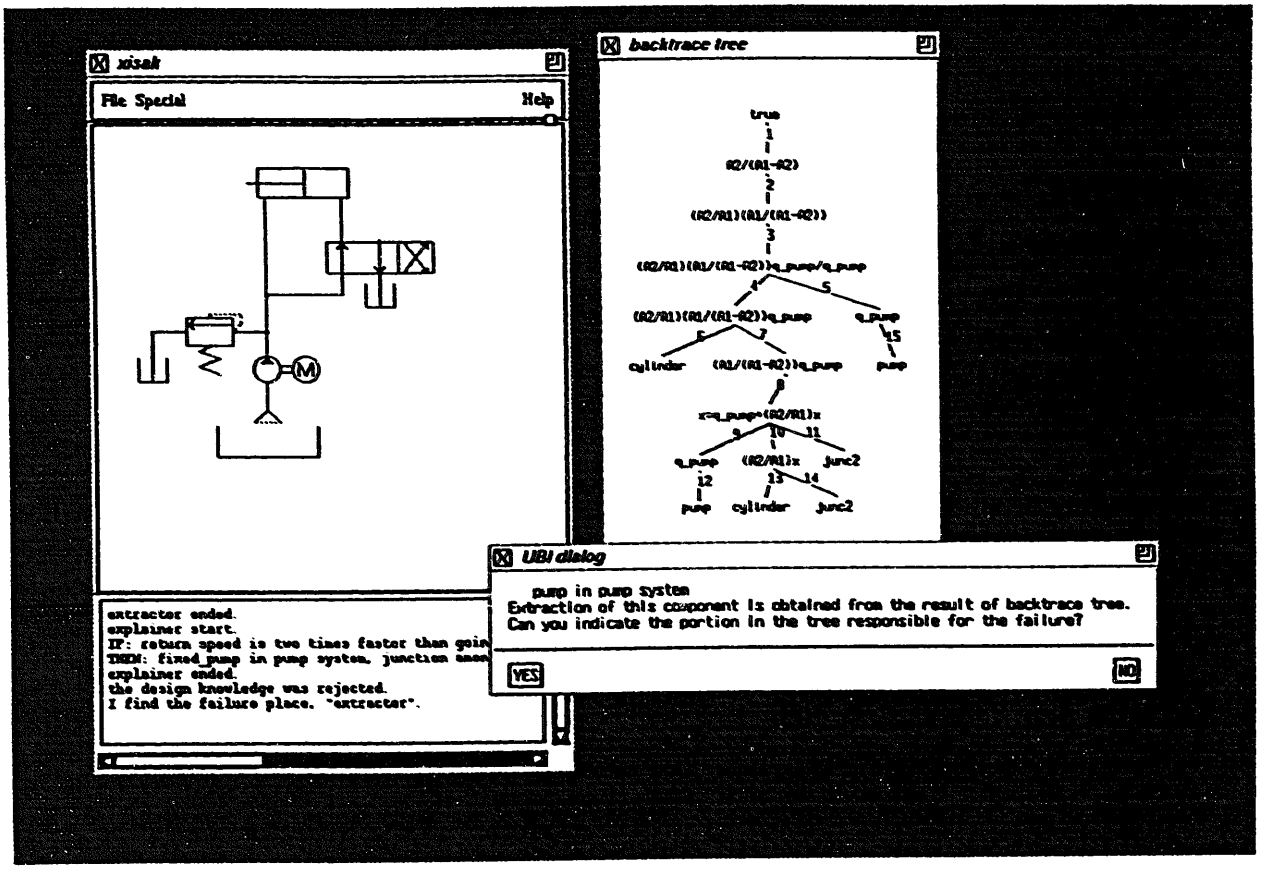


図 4.14: UBIのインタビュー(取り出す必要がない理由の質問)

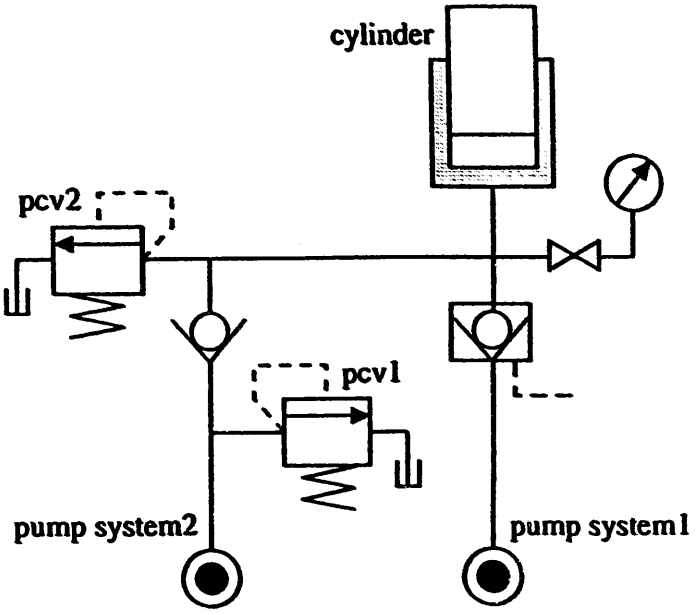


図 4.15: 加硫プレスの油圧回路図

USER: いいえ.

UBLの出力にリリーフバルブpcv2が関与する設計知識がない場合, UBIが起動される. そして, そのリリーフバルブに関する機能仕様がないことがわかると, 隠れた機能仕様があると判断し, 機能仕様獲得デーモンが起動する.

ISAK: リリーフバルブは一般に余分な圧油を逃す要素機器ですが, このリリーフバルブpcv2もそのためですか?

USER: はい.

ISAK: どのような時に余分な圧油を逃さねばならないのですか? (1) 自走. (2) 過負荷. (3) 温度上昇.

USER: 3.

ISAK: 温度上昇による過圧からどのような不都合が生じますか? (1) 自走. (2) 異常加圧. (3) ポンプの破壊.

USER: 2.

ISAK: "温度上昇時の異常加圧を防ぐ" という機能仕様でよろしいでしょうか?

USER: はい.

機能仕様獲得デーモンはリリーフバルブの一般的な機能や目的などを参照し提示することにより, 専門家に刺激を与え, 隠れた機能仕様を獲得する.

ISAKは得られた機能仕様を使い設計知識の獲得を試みるが，UBLのSimulatorは温度というパラメータを扱っていないため[8]失敗する．Simulatorの失敗には，未知のパラメータの使用や非線形現象などがある．このような場合は，ノウハウ的な設計知識として設計ノウハウデーモンを起動することにより獲得する．

### 4.5.4 獲得した知識に関する考察

UBLシステムが生成した設計知識について考察する．プロトタイプシステムは4.5.2で示した差動回路以外に，メータイン回路という例題からも設計知識を獲得することができた．メータイン回路から獲得された設計知識は，速度が可変である油圧回路を設計するためであればいつでも用いることができる．差動回路から獲得された設計知識も同様である．また，獲得された2つの設計知識を組み合わせることもでき，この場合，速度が2倍の範囲内で可変である油圧回路が設計できる．このように，獲得された設計知識は既に知られている知識ではあるが，十分有用であると考えられる．

差動回路の例では，シリンダの断面積比を3:2と固定しているため，差動回路一般としての設計知識としては不十分なものとなっている．しかし，Proverの式の評価の段階で，$\frac{a2}{a1-a2}$という式が現れているため，これを用いて，“シリンダの断面積比が$a1:a2$のとき，速度は$\frac{a2}{a1-a2}$倍となる”と一般化することが可能である．

現在プロトタイプが動作できる回路例は，用いる要素機器の面では，油圧ポンプ，油圧アクチュエータとも数が限られており，また証明戦略の面からも限られており，十分な量の例題を実行することができない．要素機器の数に関しては順次拡張を進めており，証明戦略に関しても，その数え上げを行なっている．

今回用いた例題に関しては，Extractorが取り出す重要要素機器は適切なものとなっている．差動回路の例では，シリンダとループを形成しているジャンクションのみを取り出し，油圧ポンプは取り出さなかった．これは実際に，“速度が2倍”であるという性質に油圧ポンプは全く関係ないためである．このように，取り出しに際して必要な要素機器，不必要な要素機器を選ぶヒューリスティックスがあり，それには様々なものが考えられる．例えば，普通の油圧回路ではパイプによる圧力損失についてはあまり考慮しないが，大量に油が流れる回路ではパイプの圧力損失を防ぐための要素機器が接続される場合があるが，そのような場合には，その要素機器を適切に取り出す必要がある．このようなヒューリスティックスを数え上げることは今後の重要な課題の1つである．

## 4.6 結言

本章では，設計型エキスパートシステムのための設計知識を獲得する設計例からの学習とインタビューによる獲得を融合した新しいアプローチを提案

[8]現在，Simulatorは圧力と流量の2つのパラメータのみを用いている

した. このアイデアに基づいたISAKシステムについて述べた. またプロトタイプシステムの実行結果から, 専門家が用いている設計知識のある程度の部分を自動的に獲得することができることを確認した. 従来のインタビューシステムではUBLが自動的に獲得することができる知識についても問題解決を行ないながら質問をしていたが, 学習機構をドライバとした本インタビューシステムによってこれらの質問を減少させることができ, インタビューのドライバとして学習機構(説明に基づく学習)が適切であることが確認できた. また, 基本知識では説明困難な知識に焦点を当てたインタビューを可能にしたが, このことにより, 専門家のノウハウの獲得が容易になると思われる.

ISAKと同様に設計知識を獲得するインタビューシステムにSALT[27]がある. SALTは問題解決を行ないながら基本的に全て質問を行なうが, ISAKは設計図の理解を行ない, 理解できないところのみ質問を行なう. 専門家への質問の数から見ると, ISAKの方がより少なくなっている.

これまでは学習と知識獲得はそれぞれの分野で独立して研究がなされていたが, 本章で示したように学習と知識獲得を融合した試みは今後の研究において重要な意味を持つものと考えられる.

# 第5章

# EBLの機能拡張

## 5.1 緒言

EBL(Explanation-Based Learning)[2]は訓練例が目標概念のインスタンスであることを領域知識を用いて証明することによって,問題解決システムの実行効率を向上させる学習記述を獲得する枠組である.一般に知識ベースシステムが対象とする種々の問題領域において,利用者がシステムに要求する問題解決に関連するパフォーマンスとして,実行効率の向上に加えて,再利用性や移植性,問題解決結果の性能や信頼性などさまざまなものが考えられる.EBLに関して蓄積されたこれまでの研究成果はこれらのパフォーマンスの一つである問題解決時の実行効率の向上に焦点を合わせており,知識獲得の一形態としては不十分である.EBLはSBL(Similarity-Based Learning)に比べて実世界の問題への適用という点で遅れているが,上述のことがその理由の一つになっているものと思われる.

EBLの振舞いは領域知識をある方向に組織化することと考えることができることから,従来のEBLは領域知識を問題解決時の実行効率向上という方向に組織化していたとみなすことができる.しかしながら,EBLを知識獲得の一形態とすることを考えれば,上述の多様な要求に応じて,問題解決時の実行効率向上という方向以外の方向にも組織化できるような拡張された枠組を作ることが望まれる.領域知識を組織化する方向を比較視点と呼び,この視点のもとで学習記述を修正することによって領域知識を複数方向に組織化することが可能になる.

本章では,EBLを知識の再構成メカニズムと解釈し,EBLに領域知識の組織化方向である比較視点を導入することによってEBLを拡張し,知識獲得の一形態として機能させることを考える.以下5.2節では比較視点導入の動機について述べ,具体例を示す.また,比較視点について考察する.5.3節では比較視点を導入したEBLの定式化を行なう.その際,従来のEBLが比較視点を導入した

---

関連論文[A-2]

EBLの特殊な場合であることを示す. 最後に上述の伝統的なEBLの例に対して比較視点を導入したEBLを適用し, その有効性を示す.

## 5.2 比較視点

本節ではEBLへの比較視点の導入の動機と意義について述べる. その際, ホーン節の属性を利用した例とインスタンスの属性を利用した例の2種類の例を示すことによって比較視点の有用性を示す. 最後に比較視点について考察する.

### 5.2.1 比較視点導入の動機と意義

EBLによって生成される学習記述は訓練例が一般化された知識となっている. 一方, 人が例題から学習を行なう場合は訓練例が一般化された知識以外にも有用な知識を学習する. この有用な知識とは, 例えば類推による異なったドメインの知識であったり, 同じドメインでも訓練例の影響を弱くした知識であったりする. 前者の類推による学習も非常に重要な問題であるが, ここでは後者の学習に注目する. 後者の学習は例題に基づいて学習を行ない知識を生成し, その知識に何らかの修正を加えることによってより有用な知識を生成する. 一般に人は経験からの学習だけではなく, このように経験から想起される学習も行なっている. 従って, 例題からの学習であるEBLをより強力に, より人の学習に近くするためにはこのような学習も行なう必要がある.

例題から学習した知識を修正する際に用いる観点を比較視点と呼び, 1つのドメインに対して複数の比較視点があると考えた. つまり, あるドメインにおいて学習した知識($LD$)は比較視点$A$のもとでは$LD_A$となり, 比較視点$B$のもとでは$LD_B$となる. 後の知識の利用状況を考慮すると, その利用状況に即した知識に修正することが好ましい. つまり, 比較視点は学習された知識を利用する方法や状況によって変化する要求と換言することができる. このような比較視点をEBLに導入することによって, 例題をきっかけとしたある特定の要求を満足するような知識を生成することができるであろう.

以下にEBLに比較視点を導入する例を2種類の方法を用いて示す. 第1の方法はEBLの説明構造の構成要素であるホーン節の属性を利用して修正し, 第2の方法は訓練例の引数インスタンスの属性を利用して修正する.

### 5.2.2 ホーン節の属性を利用した修正

EBLにおいて行なわれるタスクには, SAFE-TO-STACKやCUPや自殺などの概念定義, ロボット計画, 数式処理などの問題が主である. ここではタスクの一例として自殺の概念定義を取り上げる. 以下に文献[2]をより詳細にした領域

知識を示す[1].

```
kill(A,B):-hate(A,B),possess(A,C),weapon(C).
hate(W,W):-depressed(W).
possess(U,V):-buy(U,V).
weapon(Z):-gun(Z).
weapon(Z):-knife(Z).
weapon(Z):-rope(Z).[2]
```

ここで,操作性規範は訓練例として与えられる述語で満足されると仮定する.これらのもとで次のような訓練例を与える.

```
depressed(john), buy(john,obj1), gun(obj1)
```

すると,次のような学習記述を得る.

```
kill(Y,Y):-depressed(Y),buy(Y,C),gun(C).                    (1)
```

学習記述は領域知識から演繹される範囲にあるため,この学習記述の正当性も保証されている.

上記のようにEBLを行ない,得た学習記述を用いて自然言語理解システムを利用することを想定する.今,システムがアメリカや日本で利用されると仮定すると,このシステムにはアメリカだけではなく,日本の社会事情にも適した学習記述の実装が要求される.つまり,得られる学習記述はある特定の国に依存しないものである必要がある.これが移植性(portability)という比較視点である.この移植性を満足するように(1)の学習記述を修正することを考える.まず,学習記述の中で移植性を低下させている述語に注目する.この場合,gun(C)である.この述語gun(C)をknife(C)またはrope(C)と交換する.これは学習記述(1)のweapon(C) :- gun(C)の部分木をweapon(C) :- knife(C)またはweapon(C) :- rope(C)と交換することを意味する.銃はアメリカでは容易に入手できるが,日本ではそうではない.これに対してナイフやロープは特定の国で手に入らないということはないため,移植性に富んだ学習記述となる.knife(C)を選ぶと,(1)の学習記述は(2)のようになり,rope(C)を選ぶと,(3)のようになる.

```
kill(Y,Y):-depressed(Y),buy(Y,C),knife(C).                  (2)
kill(Y,Y):-depressed(Y),buy(Y,C),rope(C).                   (3)
```

他の状況を想定する.同じ領域知識のもとで訓練例{depressed(taro), buy(taro,obj2), knife(obj2)}が与えられ,(2)のような学習記述が得られたとする.システムの利用する学習記述に信頼性(reliability)が比較視点として求められる場合を想定する.ナイフで刺すより銃で撃つ方がより確実に人を殺すことができるため,

[1]以降,prologの書式を用い,定項を小文字,変項を大文字で表す.
[2]少し無理があるがロープを武器とする.

knife(C)に注目し,この述語をgun(C)と交換する.この修正によって,自殺の成功率が高くなる,つまり信頼性に富んだ学習記述(1)が生成される.

また,システムに高速性が求められる場合を想定する.この比較視点を効率性(efficiency)(いわゆるUtilityではない)と呼ぶ.この場合も基となる学習記述のweapon(C)の下の述語を,銃で撃つ,ナイフで刺す,ロープで首を締める,の中から最も苦しみが少なく,速く殺すことができる述語に交換することによって,効率性に富んだ学習記述が生成される.

上記のようにある例題をきっかけとして,ある比較視点を満足するような学習記述を生成することができ,問題解決システムにユーザが求める知識の組織化に柔軟に対処することができる.また,この修正に用いたある比較視点に固有の知識を比較知識と呼び,ユーザによって要求されるであろう比較視点毎に用意しておく.比較知識はある視点のもとで得られた説明構造のあるホーン節と,同じヘッドを持った領域知識にある別のホーン節の属性を調べることによって比較し,その優劣関係を導出する.

比較視点の例として,移植性,信頼性,効率性を挙げたが,比較視点はその他にもさまざまなものがある.例えば,武器の値段である価格性(value),武器の使い勝手である操作性(operationality)などである.これらは自殺の例というドメインによって生じる比較視点であり,他のドメインではそのドメインに固有の比較視点がある.

### 5.2.3 インスタンスの属性を利用した修正

前節ではホーン節の属性を調べることによってそれらホーン節間の比較を行ない,修正することによってより有用な学習記述が生成されることを示した.本節ではインスタンスに構造などのさまざまな属性が含まれている場合についても比較視点を導入することによってそれらを扱い,視点のもとでより有用な学習記述が生成できることを例を用いて示す.

具体例として,UNIXのシェルスクリプトプログラムをドメインとする.簡単のため,領域知識を2つに限定し,以下に示す[3].

```
write_user_name_logined(who | sed X):-
    write_login_info(who), write_first_field(sed X).
write_user_name_logined(who | awk X):-
    write_login_info(who), write_first_field(awk X).
```

write_user_name_logined(X)という述語は現在システムにログインしているユーザのログイン名を表示するという仕様をXが満たすことを表している.このもとで次のような訓練例を与える.

```
write_login_info(who), write_first_field(awk '{print $1}')
```

[3]引数中の"|"はprologのリストの頭部と尾部を分ける記号ではなく,シェルにおけるパイプの記号である.

その結果,説明構造は次のようになる.

```
write_user_name_logined(who | awk '{print $1}'):-
    write_login_info(who),
    write_first_field(awk '{print $1}').                    (4)
```

前節の例とは異なり,ここでは一般化する前の説明構造に注目する.この説明構造は"現在システムにログインしているユーザのユーザ名を表示する"という仕様のシェルスクリプトを表している.今この仕様を変えずにより高速で動作するシェルスクリプトに修正することを考える.つまり比較視点として効率性(efficiency)が要求されたとする.awkは非常に強力な文字列操作コマンドであるが,そのために動作速度が犠牲となっている.それに対して同じく文字列操作コマンドであるsedはawkより動作速度が格段に速い.従って効率性の比較視点のもと(4)の説明構造を向上させるにはawkをsedに修正すれば良い.

また上の仕様のユーザ名の部分をユーザの端末に変更されても簡単にプログラムが修正できるようなシェルスクリプトに修正することを考える.つまり比較視点として再利用性(reusability)が要求されたとする.この場合も同様にawkとsedが比較できる.ただし再利用性を考えた場合,文字列のフィールドという概念があるawkの方が正規表現でしか文字列を扱えないsedより修正が簡単なため,(4)の説明構造はそのままの方が良いという結果がでる.

上記の例はインスタンスの持つ属性によって比較できる顕著な例である.インスタンスの持つ属性による比較は自殺の例でも行なえる.例えば前節においてその他の視点で例として挙げた価格性において,gun(X)という述語自身も値段という属性を持てるが,gun(X)のインスタンスの1つobj1にも値段という属性があり得る.このようにホーン節の属性のみならず,インスタンスの属性にも注目してユーザの要求する比較視点のもとで知識を組織化することができる.

### 5.2.4 比較視点についての考察

EBLがDeJongらによって提案された当初は,EBLは一つの例題からタスクに役立つ学習記述を獲得する枠組として急速に発展した.これは領域知識をチャンクすることによって問題解決のステップ数を減少させ,結果として問題解決速度を向上させるという原理に基づいている.しかし,実際には[28]にも報告されているように,学習を繰り返し行なうと学習記述が単純に増加し,最後には学習なしのシステムより問題解決速度が低下することがある.従って,実行効率性の問題は学習結果が蓄積された知識ベースを用いて実際に問題解決を行なった場合の実行効率を考えなければならない.この問題はutitlity problemと呼ばれ,操作性規範の主な課題として,近年さまざまな研究が行なわれている.しかし,Kellerが文献[29]で定義しているように,操作性規範とは上のutility problemで扱われているutilityとusabilityの2種類の要求である.usabilityとは

問題解決システムがEBLによって得られた学習記述を用いる時の利用の容易さの基準であるが, utilityと比較して十分に研究されているとは言い難い. 従来のEBLでは, 説明構造の木をあらかじめ静的に与えておいた操作性規範を満足するようにより抽象的な概念の述語へと一般化することによってusabilityに対処していた. 本論文で提案する比較視点を導入したEBLは, より抽象的な概念の述語への一般化のみならず, 与えられた比較視点においてより適切な概念の述語へ学習記述を修正することを可能にする. 換言すると, 操作性規範の1つであるusabilityを一般化していると解釈することができる. これを図5.1に示す. 従来のEBLではgunの集合からweaponの集合へのように含意関係に

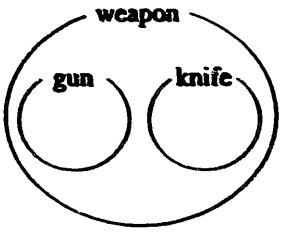


図 5.1: usabilityの一般化

ある集合に学習記述を修正していたが, 提案する方式によってgunの集合からknifeの集合へのように兄弟関係にある集合にも学習記述を修正できる.

比較視点を導入したEBLは, 例えば故障診断の専門家がある徴候から特定の原因を突き止めた経験からヒントを得て, その推論パス中のあるノードやインスタンスから想起されるその例とは異なった故障診断知識を見い出す, というような人の学習過程と類似している. これによって, 従来のEBLより人の知的活動に近い柔軟な学習を行なうことができ, 知識獲得の一形態として十分機能すると考えられる.

## 5.3 比較視点を導入したEBL

本節では比較視点を導入したEBLを定式化する. まず, 比較視点を導入したEBLの定式化のための諸定義を示す. 5.2節で示したホーン節の属性を利用した修正とインスタンスの属性を利用した修正をそれぞれ構造操作と変数操作と呼び, これらの定式化を行なう. 最後に比較視点を導入したEBLの定式化を示す.

### 5.3.1 諸定義

ここでは従来のEBLの説明構造と学習記述の定式化[4], 操作可能性の定式化を示す.

【定義1】 説明構造

$A$と$A'$を同じ述語定項をもつ述語, $A' \leftarrow B_1 \wedge \cdots \wedge B_n$を領域知識のホーン節とするとき, 次のように再帰的に定義される式を説明構造という.

(a) $A : A'$.

(b) $A : (A' \leftarrow \langle B_1 \rangle \wedge \cdots \wedge \langle B_n \rangle)$.

ここで, $\langle B_i \rangle$は述語$B_i$を根とする説明構造, " : " の両辺は変数を共有しないとする. 説明構造の根とは, 説明構造を木構造として見たときの根で, (a),(b)における述語$A$を指す. また, 木構造の末端のうち(a)における述語$A'$を葉という.

$D$を領域知識(ホーン節の集合), $E$を訓練例(変数を含まないファクトの集合), $g$を目標概念とするとき, 以下の式を証明という.

$D \cup E \vdash g\theta$.

ここで, $\theta$は証明結果としての代入を表す. 説明構造$t$がこの証明過程の結果として生成されるとき, この操作を説明と呼ぶ. ■

【定義2】 学習記述

説明構造$t$の根を$A$, 葉を$B_1, \cdots, B_n$とする. このとき, 次のホーン節を説明構造$t$に基づく学習記述という.

$A \leftarrow B_1 \wedge \cdots \wedge B_n$. ■

本研究の目的である比較視点の導入による学習記述のusabilityの向上に議論を集約するために, 操作性規範に起因するutilityに関する議論は, 操作可能な述語の集合が与えられるものと単純化した枠組の上で捉え, 本論文では深く立ち入らないこととする. usabilityとutilityは独立のものとして考えることができるため, このように問題を簡単化してもEBLとしての定式化の一般性を損なわない. Utility問題に対してなされている様々な議論は自然に本論文で提案する枠組の上で展開できる.

【定義3】 学習記述の操作可能性

操作可能な述語の集合を$O$とすると, 述語$B_i \in O (1 \leq i \leq n)$であるとき, 学習記述$A \leftarrow B_1 \wedge \cdots \wedge B_n$は操作可能であるという. ■

説明構造$t$の葉の系列を$B_1, \cdots, B_n$とする. 以下の2節で定義する操作を施された説明構造から得られる学習記述が操作可能であるためには, 葉の述語がすべて操作可能, すなわち$B_i \in O (1 \leq i \leq n)$という条件を満足すれば良い.

### 5.3.2 構造操作の定義

説明構造$t$の学習記述が操作可能であるようにするアルゴリズムを次に与える.

[4]山村ら[30]の説明構造の定義, Hoppe[31]の証明の定義を参考にした.

【定義4】　構造一般化
説明構造$t$中の任意の部分構造$t_p = X : (X' \leftarrow \langle P_1 \rangle \wedge \cdots \wedge \langle P_n \rangle)$について$X$が操作可能で，かつ$t$における$X$から$t$の根に至るパス中に操作可能な述語が存在しないとき，$t_p$を$X : X'$で置き換えたものを$t^{sg}$とする．この操作を構造一般化と呼ぶ．■

構造一般化によって生成された説明構造$t^{sg}$は明らかに操作可能である．この$t^{sg}$に対して次に示すアルゴリズムに従って構造操作を施す．説明構造$t^{sg}$中の部分構造の内$B_i(1 \leq i \leq n)$が全て説明構造中の葉であるような部分構造$A : (A' \leftarrow \langle B_1 \rangle \wedge \cdots \wedge \langle B_n \rangle)$が対象となる．$t^{sg}$中で構造操作の対象となる部分構造の集合を$Ls(t^{sg})$とする．また，$V$は比較視点の集合であり，対象ドメインによって決定される．例えば，5.2.2 節で示した例では，$V = \{portability, reliability, efficiency, value, operationality, \cdots\}$となる．

【定義5】　構造操作
述語$p$をヘッドとする操作可能なホーン節の集合を$Oc(p)$とする．compare_clause(v,c1,c2)は与えられた2つのホーン節を比較視点$v \in V$のもとで比較し，その順序関係を返す関数，$Order$はホーン節間の半順序関係を表すグラフである．このとき，次のアルゴリズムによる説明構造の操作を構造操作と呼び，これによって生成される説明構造を$t^{so}$と表す．

forall $p : (p' \leftarrow \langle u_1 \rangle \wedge \cdots \wedge \langle u_m \rangle)$ in $Ls(t^{sg})$ do begin
  forall $(p'' \leftarrow v_1 \wedge \cdots \wedge v_n)$ in $Oc(p)$ do
    add compare_clause$(v, p' \leftarrow u_1 \wedge \cdots \wedge u_m, p'' \leftarrow v_1 \wedge \cdots \wedge v_n)$
    to $Order$
  $bt$ := find_max($Order$)
  replace $p : (p' \leftarrow \langle u_1 \rangle \wedge \cdots \wedge \langle u_m \rangle)$ by $bt$ in $t^{sg}$
end

ここでfind_max(G)はグラフGから最大元を求める関数である．また，$bt$は説明構造である必要があるため，find_maxは最大元であるホーン節$x \leftarrow y_1 \wedge \cdots \wedge y_n$の$y_i$を$y_i : y_i'$として説明構造$x \leftarrow \langle y_1 \rangle \wedge \cdots \wedge \langle y_n \rangle$に変換する．ここで$y_i'$は$y_i$の構造を保存するが変数を共有しない述語とする．■

関数compare_clauseはホーン節の属性と比較視点$v$によって規定される比較知識$C_v$を用いてそれらのホーン節を比較する．ホーン節の属性はファクトの集合として与えられ，比較知識はそれらのファクトを次のような述語で参照する．

attribute(Clause,Fact)

述語attributeはホーン節Clauseの属性Factを次々と参照することができる．比較知識とホーン節の属性の具体例は5.4節で示す．

### 5.3.3 変数操作の定義

構造操作によって得られた説明構造$t^{so}$に対して次に示すアルゴリズムに従って変数操作を施す.

【定義6】 変数操作

説明構造$t^{so}$の葉の系列を$L_1,\cdots,L_n$とし,その代入を$\theta$とする. make_instance($L_i\theta$)は$L_i$に対する$L_i\theta$以外の具体化$L_i\theta_x$の全可能性を集合として返す関数である[5]. compare_instance(v,p1,p2)は与えられた2つの述語のインスタンスを比較視点$v$のもとで比較し,その順序関係を返す関数,*Order*はインスタンス間の半順序関係を表すグラフである.このとき,次のアルゴリズムによる説明構造の操作を変数操作と呼び,これによって生成される説明構造を$t^{vo}$と表す.

```
for L_i := L_1 to L_n do begin
  forall L_iθ_x in make_instance(L_iθ) do
    add compare_instance(v,L_iθ,L_iθ_x) to Order
  L_iθ* := find_max(Order)
  replace L_iθ by L_iθ* in t
end
```

ここで,find_max(G)はグラフGから最大元を求める関数である. ■

変数操作によって生成された説明構造$t^{vo}$または構造一般化によって生成された説明構造$t^{vg}$に対して次に示すアルゴリズムに従って変数一般化を施す[31].

【定義7】 変数一般化

代入$\theta$を領域知識$D$による代入$\theta_D$と訓練例$E$による代入$\theta_E=\{v_1/g_1,\cdots,v_n/g_n\}$とに分ける.すなわち,$\theta=\theta_D\theta_E$となる.このとき,次のアルゴリズムによる説明構造の操作を変数一般化と呼び,これによって生成される説明構造を$t^{vg}$と表す.

```
forall g_i of v_i/g_i in θ_E do begin
  forall const in (A ← B_1 ∧ ··· ∧ B_n) do
    if const↑ = g_i then replace const↑ by v_i
end
```

ただし,*const*は$(A\leftarrow B_1\wedge\cdots\wedge B_n)$における定数の集合の1要素を指し示すポインタである. ■

この変数一般化アルゴリズムによって,領域知識による束縛は保存し,訓練例による束縛を一般化することができる.

### 5.3.4 比較視点を導入したEBLの定義

以上の定義によって比較視点を導入したEBLの定式化の準備が整った.

[5]一般にはこの集合は有限とは限らない.しかし,実際には比較可能なインスタンスに限定することができる.

【定義8】　比較視点を導入したEBL
比較視点$v \in V$が与えられ，説明によって生成された説明構造$t$に構造一般化，構造操作，変数操作，変数一般化の各操作を順に行なうことを洗練と呼ぶ．このとき，次の一連の操作を比較視点を導入したEBLという．

(1) 説明

(2) 洗練

また，比較視点が与えられなかった場合，洗練操作は構造一般化，変数一般化と行なわれ，特にこの洗練操作を一般化と呼ぶ．■

【定義8】からもわかるように従来のEBLは説明と一般化という2つのプロセスによって学習が行なわれており，これは比較視点を導入したEBLの比較視点がない学習方法となっている．つまり比較視点を導入したEBLは従来のEBLを特殊な形として含む拡張となっている．比較視点を導入したEBLは一般化のプロセスをusabilityを高める洗練のプロセスとしたところに特徴がある．洗練は従来の学習記述の一般化を含み，更に置換可能な述語への修正も行なえるプロセスである．これは従来一方向であったusabilityの操作を多方向にしたという意味でusabilityの一般化となっている．

## 5.4　比較視点を導入したEBLの学習例

本節では比較視点を導入したEBLの学習例を自殺の例とシェルスクリプトの例で示す．自殺の例ではホーン節の属性を用いた修正を行ない，シェルスクリプトの例ではインスタンスの属性を用いた修正を行なう．現在，ホーン節の属性を用いた修正はインプリメントが完了しており，自殺の例ではその実行結果を示す．プロトタイプシステムのEBLエンジンにはC-Prologを用い，X-window上でC言語を用いてインタフェースを構築し，EBLエンジンと通信している．

### 5.4.1　自殺の例

比較視点を導入したEBLによる学習を文献[2]にある自殺の概念定義を拡張した例に当てはめ，実際にどのような学習が行なえるかを検証する．領域知識は以下のようになる(図 reffig:自殺の例の領域知識参照[6])．

```
kill(A,B):-hate(A,B),possess(A,C),weapon(C).
hate(W,W):-depressed(W).
possess(U,V):-buy(U,V).
weapon(Z):-gun(Z).
eapon(Z):-knife(Z).
weapon(Z):-rope(Z).
```

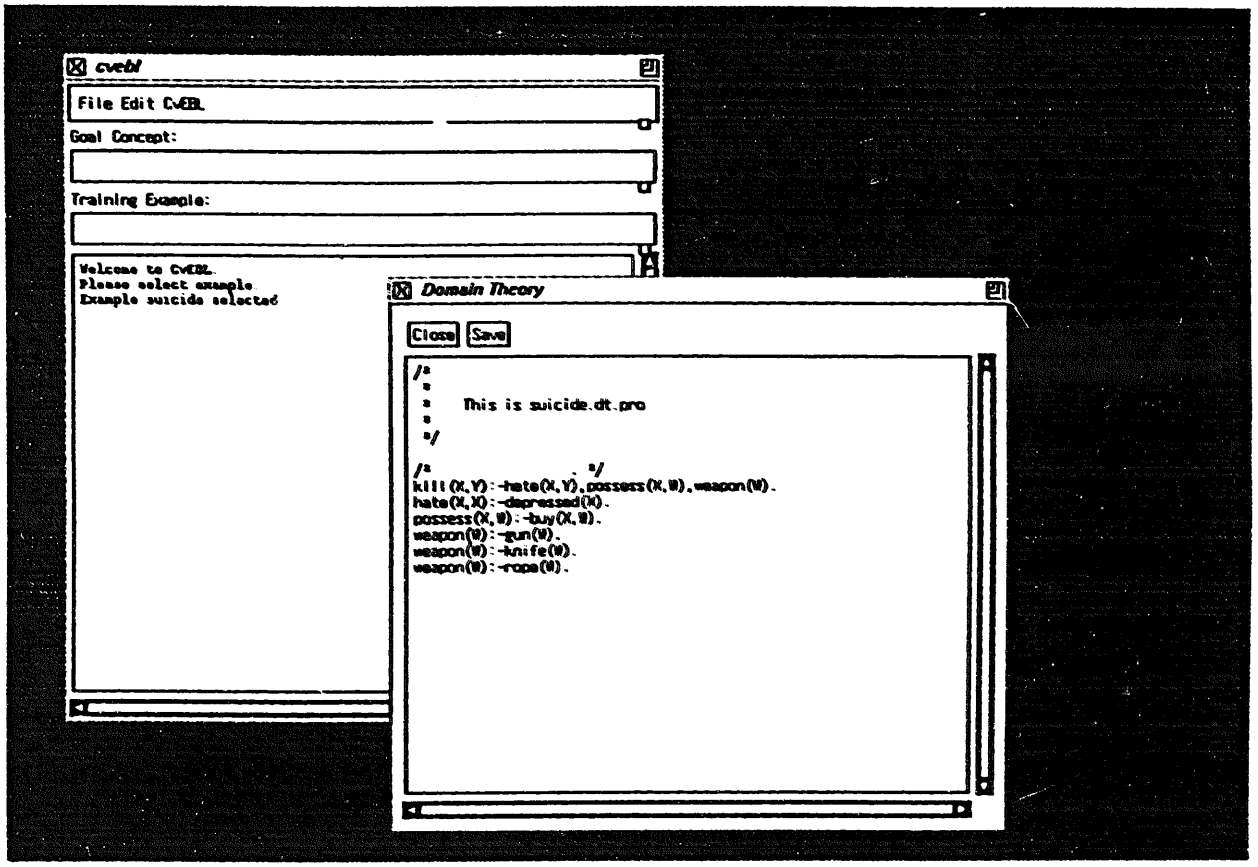


図 5.2: 自殺の例の領域知識

また領域知識に加えてホーン節の属性が与えられる.

```
weapon(Z):-gun(Z)::
        object(weapon).
        object(purchasable).
        exist(legal_control).
        made_by(metal).
        wound_by(viscera_destruction).
        wound_by(loss_of_blood).
        how_to_use(pull_the_trigger).
                ⋮
weapon(Z):-knife(Z)::
        object(weapon).
        object(purchasable).
        not_exist(legal_control).
        made_by(metal).
        wound_by(loss_of_blood).
        how_to_use(stick).
                ⋮
weapon(Z):-rope(Z)::
        object(weapon).
        object(purchasable).
        not_exist(legal_control).
        made_by(thread).
        wound_by(suffocation).
        how_to_use(throttle).
                ⋮
```

操作性規範は訓練例として与えられる述語で満足され,次のようなものとなる.

```
operational([depressed(_),buy(_,_),gun(_),knife(_),rope(_)]).
```

また比較知識として図 5.3と図 5.4の 3 種類の知識を用意しておく. 図 5.3は対象に依存しない比較知識であり,図 5.4は対象に固有の比較知識である.

### 移植性

まずユーザが要求する比較視点として移植性(portability)を想定する(図 5.5参照). いま次のような訓練例$E_1$を与えたとする.

```
depressed(john), buy(john,obj1), gun(obj1)                    E1
```

[6]プロトタイプシステムは用いている全ての知識(領域知識, ホーン節の属性, 比較知識)が実行中に編集できるようにエディタが内蔵されている.

```
Cportability = {  /* 移植性 */
        better(A, B):-widely_apply(A, B).
        widely_apply(A, B):-
                usable_area(A, AA), usable_area(B, AB), wider(AA, AB). }

Creliability = {  /* 信頼性 */
        better(A, B):-certainly_exec(A, B).
        certainly_exec(A, B):-
                achieve_prob(A, PA), achieve_prob(B, PB), higher(PA, PB). }

Cefficiency = {  /* 効率性 */
        better(A, B):-fast_finish(A, B).
        fast_finish(A, B):-
                time_used(A, TA), time_used(B, TB), shorter(TA, TB). }

Creusability = {  /* 再利用性 */
        better(A, B):-widely_use(A, B).
        widely_use(A, B):-
                possible_role(A, RA), possible_role(B, RB), more(RA, RB). }
```

図 5.3: ドメインに依存しない比較知識

```
Cportability = {  /* 移植性 */
        usable_area(X,PX):,-purchasable(X),purchasable_area(X,PX).
        purchasable(0):-attribute(0,object(purchasable)).
        purchasable_area(0,limited):-attribute(0,exist(legal_control)).
        purchasable_area(0,unlimited):-
                attribute(0,not_exist(legal_control)).
        wider(unlimited,limited).}

Creliability = {  /* 信頼性 */
        achieve_prob(X,PX):-is_weapon(X),ease(X,PX).
        is_weapon(W):-attribute(W,object(weapon)).
        ease(W,0.7):-attribute(W,how_to_use(pull_the_trigger)).
        ease(W,0.6):-attribute(W,how_to_use(stick)).
        ease(W,0.5):-attribute(W,how_to_use(throttle)).
        higher(X,Y):-number(X),number(Y),X>Y.}

Cefficiency = {  /* 効率性 */
        time_used(X,TX):is_weapon(X),waiting_time(X,TX).
        is_weapon(W):-attribute(W,object(weapon)).
        waiting_time(W,1):-
                attribute(W,wound_by(viscera_destruction)).
        waiting_time(W,10):-attribute(W,wound_by(loss_of_blood)).
        waiting_time(W,60):-attribute(W,wound_by(suffocation)).
        shorter(X,Y):-number(X),number(Y),X<Y.}
```

図 5.4: 自殺の例の比較知識

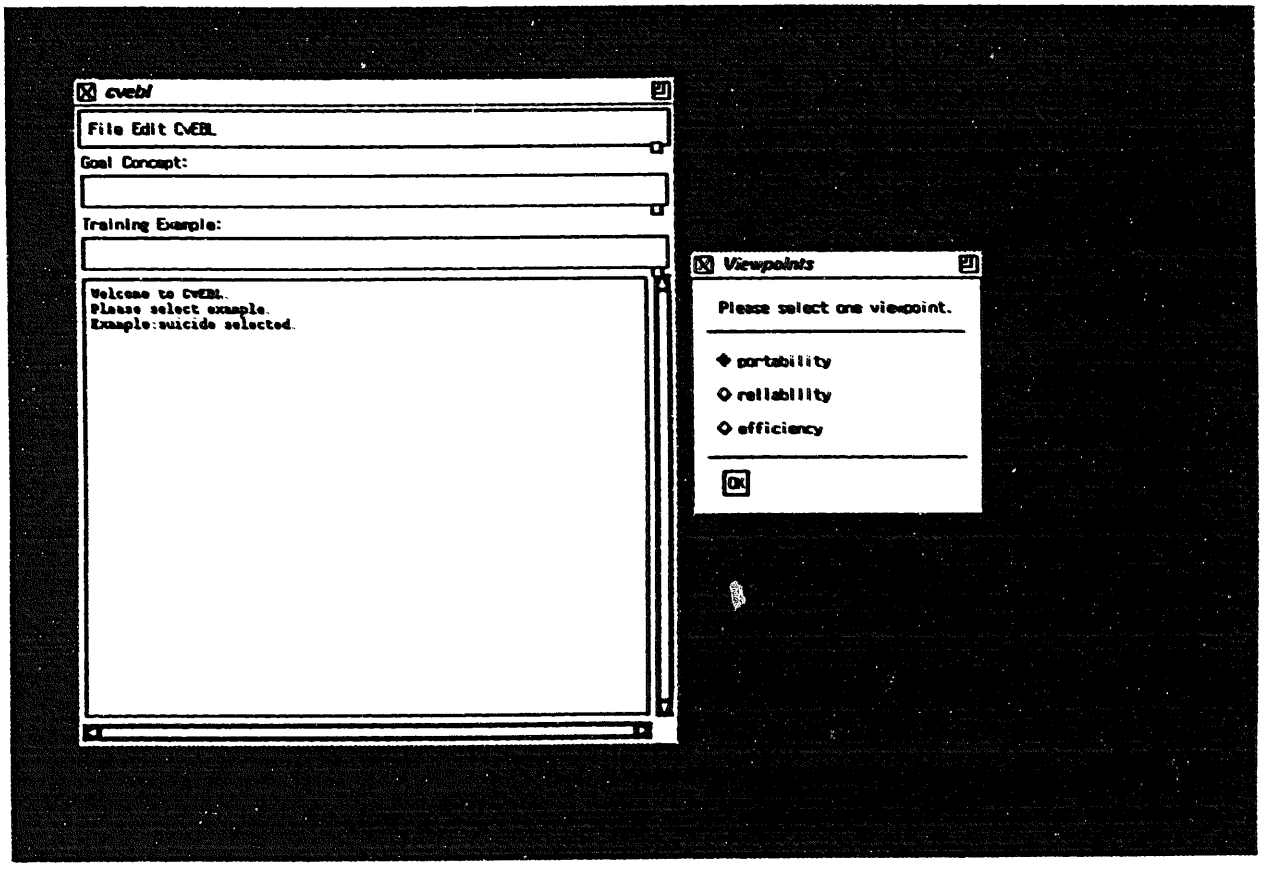
cvebl
File Edit CVEBL
Goal Concept:
Training Example:
Welcome to CVEBL.
Please select example.
Example:suicide selected.
Viewpoints
Please select one viewpoint.
portability
reliability
efficiency
OK

図 5.5: 比較視点の選択

この訓練例から従来のEBLは次の学習記述を生成する.

kill(A,A):-depressed(A),buy(A,B),gun(B). $L_1$

しかし比較視点を導入したEBLは構造操作において説明構造中のweapon(C)以下の部分構造に注目し,その説明構造のホーン節weapon(Z) :- gun(Z)と,このホーン節と置換可能なホーン節weapon(Z) :- knife(Z)とweapon(Z) :- rope(Z)の中から移植性に関する最大元を求める.ホーン節の属性と比較知識$C_{portability}$によると,weapon(Z) :- knife(Z)とweapon(Z) :- rope(Z)がweapon(Z) :- gun(Z)より優れているが,weapon(Z) :- knife(Z)とweapon(Z) :- rope(Z)の優劣関係は決定できない.

```
?- better((weapon(Z):-knife(Z)),(weapon(Z):-gun(Z))).
yes
?- better((weapon(Z):-rope(Z)),(weapon(Z):-gun(Z))).
yes
?- better((weapon(Z):-knife(Z)),(weapon(Z):-rope(Z))).
no
?- better((weapon(Z):-rope(Z)),(weapon(Z):-knife(Z))).
no
```

このような場合は最大元は決定できず,部分構造の修正に用いられるホーン節は2つの極大元からランダムに選択される.weapon(Z) :- knife(Z)が選択されたときの修正された学習記述を$L_2$, weapon(Z) :- rope(Z)が選択されたときの修正された学習記述を$L_3$に示す(図 5.6参照).

kill(A,A):-depressed(A),buy(A,B),knife(B). $L_2$
kill(A,A):-depressed(A),buy(A,B),rope(B). $L_3$

これまで用いたホーン節の属性は日本など銃の購入が困難な国におけるものであった.今,アメリカ等の銃の購入が容易な国における学習を想定する.この場合,weapon(Z) :- gun(Z)のホーン節の属性であったexist(legal_control)はweapon(Z) :- knife(Z)やweapon(Z) :- rope(Z)の属性と同様にnot_ex-ist(legal_control)となる.従って,これら3種類のホーン節は移植性に関して共に極大元となり,優劣関係は決定できなくなる.比較元のホーン節(この場合weapon(Z) :- gun(Z))が極大元の1つである場合は構造操作における修正は行なわない.

信頼性

次にユーザが要求する比較視点として信頼性(reliability)を想定する.いま次のような訓練例$E_2$を与えたとする.

depressed(fred), buy(fred,obj2), knife(obj2) $E_2$

この訓練例から従来のEBLは上の$L_2$の学習記述を生成する.比較視点を導入したEBLは移植性と同様にweapon(Z)をヘッドとする3種類のホーン節をその属性と比較知識$C_{reliability}$によって比較する.ここでは最も信頼性が高いweapon(Z) :- gun(Z)が選ばれ,$L_1$の学習記述が生成される.

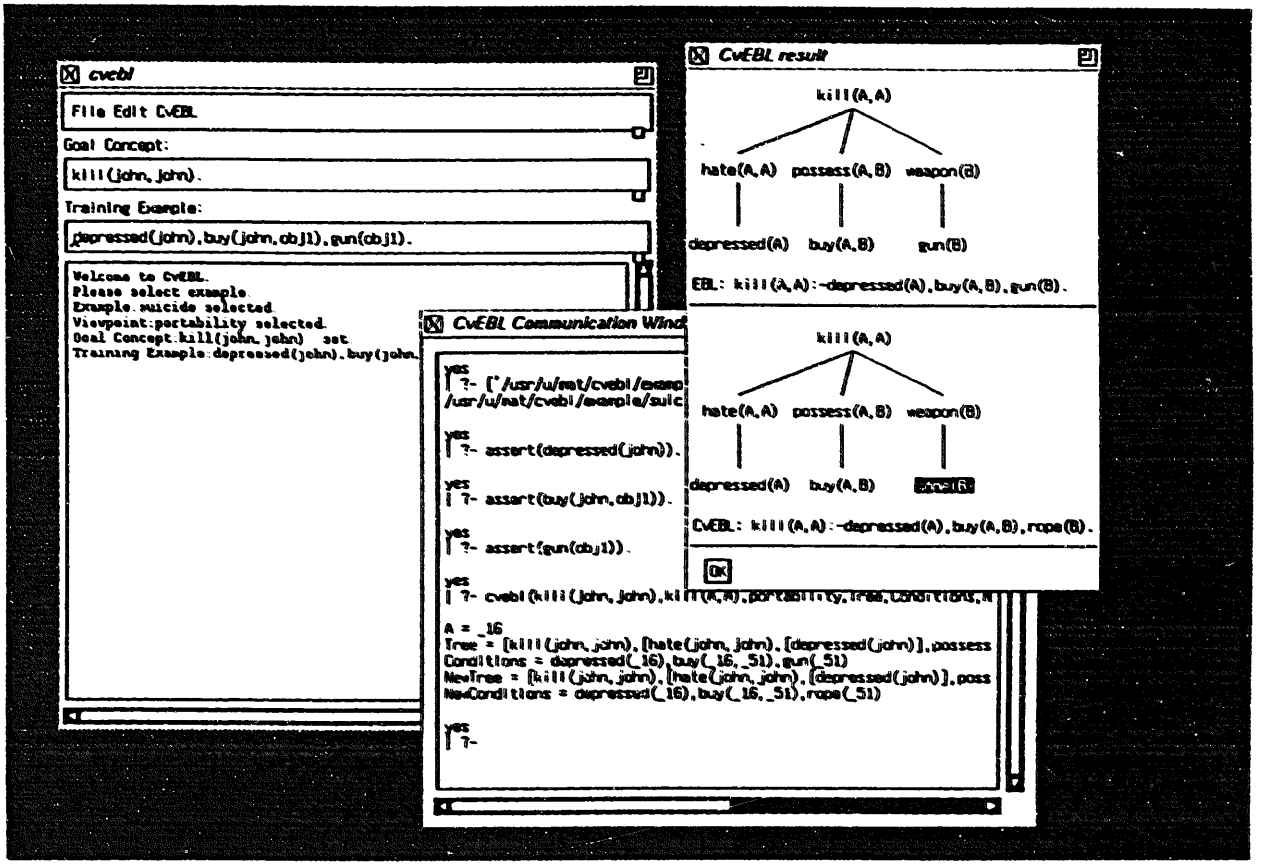


図 5.6: 修正された学習記述

効率性

最後にユーザが要求する比較視点として効率性(efficiency)を想定する．訓練例$E_3$が与えられたとする．

depressed(taro), buy(taro,obj3), rope(obj3)　　$E_3$

従来のEBLでは上の$L_3$が生成されるが，ここでも同様に3種類のホーン節を比較する．ホーン節の属性と比較知識$C_{efficiency}$によって，weapon(Z) :- rope(Z)から最も効率性が高いweapon(Z) :- gun(Z)に学習記述$L_3$は修正される．これによって$L_1$の学習記述が生成される．

表5.1に上記の3種類の比較視点において，与える訓練例によってどのような学習記述が生成されるかを修正の有無も加えて示す．

表5.1: 自殺の例における学習結果

| 比較視点 | 訓練例 | 修正 | 学習記述 |
|---|---|---|---|
| 移植性 | $E_1$ | 有 | $L_2$ or $L_3$ |
| | $E_2$ | 無 | $L_2$ |
| | $E_3$ | 無 | $L_3$ |
| 信頼性 | $E_1$ | 無 | $L_1$ |
| | $E_2$ | 有 | $L_1$ |
| | $E_3$ | 有 | $L_1$ |
| 効率性 | $E_1$ | 無 | $L_1$ |
| | $E_2$ | 有 | $L_1$ |
| | $E_3$ | 有 | $L_1$ |

### 5.4.2　シェルスクリプトの例

ここでは5.2.3節で示したシェルスクリプトの例をより具体的に示し，インスタンスの属性を用いた学習がどのように行なわれるかを検証する．シェルスクリプトプログラミングの領域知識は以下のようになる．

```
write_user_name_logined(who | sed X):-
    write_login_info(who), write_first_field(sed X).
write_user_name_logined(who | awk X):-
    write_login_info(who), write_first_field(awk X).
```

簡単化のため，5.2.3節と同様に領域知識を2つに限定する．またこの領域知識に加えてインスタンスの属性も与える．

```
awk::
        command(unix).
        do(string_operation).
        support_in(bsd).
        support_in(sysV).
        memory_use(big).
        handle(regular_expression).
        handle(field).
              ⋮
sed::
        command(unix).
        do(string_operation).
        support_in(bsd).
        support_in(sysV).
        memory_use(small).
        handle(regular_expression).
              ⋮
```

操作性規範は訓練例として与えられる述語で満足され，次のようになる．

```
operational([write_login_info(_),write_first_field(_)]).
```

また比較知識として図5.3と図5.7の2種類の知識を用意しておく．

効率性

まずユーザが要求する比較視点として効率性(efficiency)を想定する．いま5.2.3節と同じ訓練例を与える．

```
write_login_info(who), write_first_field(awk '{print $1}')
```

この訓練例から次の学習記述がまず生成される．

```
write_user_name_logined(who | awk '{print $1}'):-
    write_login_info(who),
    write_first_field(awk '{print $1}').                    (4)
```

比較視点を導入したEBLは変数操作においてインスタンスのawkに注目し，このインスタンスと置換可能なインスタンス(sed)の中から効率性に関する最大元を求める．インスタンスの属性と比較知識$C_{efficiency}$によると，sedがawkより優れていることが導出される．

```
?- better(sed,awk).
yes
```

```
Cefficiency = {  /* 効率性 */
        time_used(X,TX):-is_command(X),execution_time(X,TX).
        is_command(C):-attribute(C,command(_)).
        execution_time(C,fast):-attribute(C,memory_use(small)).
        execution_time(C,slow):-attribute(C,memory_use(big)).
        shorter(fast,slow).}

Creusability = {  /* 再利用性 */
        possible_role(X,RX):-is_command(X),roles(X,RX).
        is_command(C):-attribute(C,command(_)).
        roles(C,R):-assert(tmp([])),role(C,H),roles2(A),
              assert(tmp([H|A])),fail.
        roles(C,R):-retract(tmp(R)).
        roles2(X):-retract(tmp(X)),!.
        role(C,regular_expression):-
              attribute(C,handle(regular_expression)).
        role(C,field):-attribute(C,handle(field)).
        more(X,Y):-nmember(X,N1),nmember(Y,N2),N1>N2.
        nmember([],0):-!.
        nmember([A|B],N1):-nmember(B,N2),N1 is N2 + 1.}
```

図 5.7: シェルスクリプトの例の比較知識

これによって(4)の学習記述は以下のようになる.

```
write_user_name_logined(who | sed X):-
    write_login_info(who), write_first_field(sed X).          (5)
```

再利用性

次にユーザが要求する比較視点として再利用性(reusability)を想定する. 効率性の例と反対に次のような訓練例を与える.

```
write_login_info(who), write_first_field(sed 's/ .*$//')
```

これによって(5)の学習記述が得られるが, 変数操作においてsedとawkのインスタンスの属性と比較知識$C_{reusability}$によってbetter(awk,sed)という比較結果が導出される. これらのインスタンスを置換することによって(6)の学習記述が生成される.

```
write_user_name_logined(who | awk X):-
    write_login_info(who), write_first_field(awk X).          (6)
```

## 5.5 結言

本章では, 利用方法や利用状況によって変化するユーザの問題解決システムに対するパフォーマンスへの多様な要求に対処するための切口である比較視点をEBLに導入し, EBLを知識獲得の一形態として利用するための定式化を行なった. また, 簡単な例を用いて, EBLに比較視点を導入することによってユーザのさまざまな要求に答えることができ, 知識獲得の一形態として十分機能することができると考えられる.

比較視点を導入したEBLは従来のEBLでは十分に研究されていないusabilityに注目し, 比較知識とホーン節の属性, インスタンスの属性を用いて比較することによって, 構造一般化や変数一般化のみならず, 構造操作, 変数操作を行ない, よりusabilityに富む学習記述を生成することを可能にした. これは従来の述語の一般化というusabilityの概念の一般化となっている. また知識獲得においてはutilityよりusabilityの方がより重要であるため, この拡張はこの意味からも意義のあることと思われる. 本章で提案した構造操作と変数操作を用いた学習方法は従来のEBLより人の学習に近い柔軟な学習方法と言える.

# 第 6 章

# 結論

本論文では,説明に基づく学習を知識獲得の一形態として活用するための方法論の提案とその方法論に基づいたシステムの実現と評価について述べた.

第 2 章と第 3 章では,本論文の主題である知識獲得と説明に基づく学習について概観し,基本となっている思想について述べた.第 2 章では,知識工学における知識獲得の位置付け,知識獲得問題を考える上で重要と思われる視点を整理した.第 3 章では,説明に基づく学習の概念を整理した.

第 4 章では,本論文で提案した方法論の第一のアプローチについて述べた.EBL を専門家の経験則を獲得するためのドライバとして機能させることにより,従来のインタビューシステムの欠点であった過剰な質問を削減し,専門家の負担を軽減することができることを示した.EBL をインタビュードライバとするこのアプローチに基づいた設計知識獲得システム ISAK について述べた.ISAK は ISAK と同様に設計知識を獲得するインタビューシステム SALT より,専門家への質問の量が減少しており,EBL をインタビュードライバとする方法論の有効性が明らかになった.

第 5 章では,本論文で提案した方法論の第二のアプローチについて述べた.利用方法や利用状況によって変化する問題解決システムの利用者のシステムに対するパフォーマンスへの多様な要求に対処することが可能な比較視点の概念を提示した.EBL を知識の再構成メカニズムと解釈することによって,比較視点を導入し,EBL を拡張した.この拡張では,従来の EBL では十分に研究されていない usability に注目することによって,獲得する知識の質を向上させることが可能となった.これによって,比較視点を導入した EBL が知識獲得の一形態として十分機能することを示すことができた.

本論文では,説明に基づく学習を知識獲得手法として活用するための方法論について検討した.専門家の経験則を獲得するためのドライバとして EBL を利用するアプローチと比較視点を導入することによって EBL を拡張するアプローチの二種類の異なったアプローチを提案し,それぞれのアプローチに基づいたプロトタイプシステムを設計開発することによってこれらのアプローチの有効性を示した.知識獲得の知識工学における重要性は極めて大きい.そ

の知識獲得の一つの形態である例題からの学習におけるパラダイムに，従来のSBLに加えてEBLが利用できることの意味は非常に大きい．本論文で提案したEBLを実際に知識獲得に利用可能にする方法論は，今後の知識獲得システムにおける手法の選択範囲を広げることと思われる．また，これまでは学習と知識獲得はそれぞれの分野で独立して研究がなされていたが，本論文で示した知識獲得に説明に基づく学習を用いる試みは，今後の研究において重要な意味を持つものと考えられる．

説明に基づく学習を知識獲得の一形態とするための方法論は本論文で提案した二種類のアプローチだけであるとは限らない．例えば，Mitchellらが今後の研究課題として提示したEBLとSBLの結合がある．SBLによって，EBLの問題点である領域知識の不完全問題を補うことができると考えられる．このEBLとSBLの結合のアプローチなどを定式化することによって説明に基づく学習の知識獲得への応用は更に進むものと考えられ，今後の重要な研究課題となるであろう．

# 参考文献

[1]Yoshikawa, S. Ozawa, K., Koyagoshi, and N., Oodo, T.:Knowledge Dependency Analysis of Plant Operators using Consistent Protocol Formulation, *Proceedings of PSAM3/ESREL'96, (1996)*

[2]Furuhama, Y., Furuta, K. and Kondo, S: A methodology to represent plant operator's mental model for training support system, *Proceedings of CSAPC'95, (1995)*

[3] 渡辺：エキスパートシステムにおける知識獲得, 情報処理学会誌, Vol. 28, No. 2, pp. 167–176 (1987).

[4] 山口, 溝口, 田岡, 小高, 野村, 角所：深い知識に基づく知識コンパイラの基本設計, 人工知能学会誌, Vol. 2, No. 3, pp. 333–340 (1987).

[5] 來村, 笹島, 池田, 吉川, 遠藤, 溝口：原子力プラント「常陽」のための対象モデルと定性推論方式, 人工知能学会研究会資料SIG-KBS-9203-9, pp. 67–76 (1992).

[6] Kleer, J. D. and Brown, J. S.: A Qualitative Physics Based on Confluenes, *Artificial Intelligence*, Vol. 24, No. 1, pp. 7–83 (1984).

[7] 笹島, 來村, 池田, 溝口：機能と振舞いに着目したドメインオントロジーに関する検討, 1993年度人工知能学会全国大会論文集, pp. 631–634 (1993).

[8] Soloway, E., et al.: Assessing the maintenability of XCON-in-RIME : Coping with the problems of a very large rule-base, *Proceedings of AAAI87*, pp. 824–829 (1987).

[9] Musen, M. A., et al.: Use of a domain model to drive an interactive knowledge-editing tool, *J.Man-Machine Studies*, Vol. 26, No. 1, pp. 105–121 (1987).

[10] 北上, 國藤, 宮地, 古川：論理プログラミング言語Prologによる知識ベース管理システム, 情報処理学会誌, Vol. 26, No. 11, pp. 1283–1295 (1985).

[11] 辻野, 櫻井, 野村, 千種, 溝口, 角所：知識ベース構築支援環境を備えた連続音声認識エキスパートシステム：SPREXII, 電子情報通信学会論文誌, Vol. J71-D, No. 3, pp. 531–542 (1988).

[12] Samuel, A. L.: Some studies in machine learning using the game of checkers, *IBM J. Research and Developement*, Vol. 3, pp. 210–229 (1959).

[13] Mitchell, T. M., Keller, R. M. and Kedar-Cabelli, S. T.: Explanation-Based Generalization : A Unifying View, *Machine Learning*, Vol. 1, No. 1, pp. 47–80 (1986).

[14] DeJong, G.: An Introduction to Explanation-based Learning, *An Introduction to Explanation-based Learning*, ed. H. E. Snrobe, chapter 2, pp. 45–81, MORGAN KAUFMANN (1988).

[15] Kedar-Cabelli, S. T. and McCarty, L. T.: Explanation-based generalization as resolution theorem proving, *Proceedings of the 4th International Workshop on Machine Learning*, pp. 383–389 (1987).

[16] Harmelen, van F. and Bundy, A.: Explanation-Based Generalisation = Partial Evaluation, *Artificial Intelligence*, Vol. 36, No. 3, pp. 401–412 (1988).

[17] 川口, 溝口, 角所：インタビューシステムのためのシェル, SIS, 人工知能学会誌, Vol. 4, No. 4, pp. 411–420 (1989).

[18] Eshelman, L. and McDermott, J.: MOLE : A Knowledge Acquisition Tool That Uses Its Head, *Proceedings of AAAI86*, pp. 950–955 (1986).

[19] Bennett, J. S.: ROGET : A Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System, *Journal of Automated Reasoning*, Vol. 1, No. 1, pp. 49–74 (1985).

[20] Kahn, G., Nowlan, S. and McDermott, J.: MORE : An Intelligent Knowledge Acquisition Tool, *Proceedings of IJCAI85*, pp. 581–584 (1985).

[21] 川口, 溝口, 角所, 山田, 野村, 馬場, 中島：設計時の仕様獲得を支援する知的インタビューシステム $I^2S/D$–油圧回路の設計に関して–, 情報処理学会研究会 知識工学と人工知能, pp. AI59–16 (1988).

[22] Laird, J. E., Newell, A. and Rosembloom, P. S.: SOAR : An Architecture for General Intelligence, *Artificial Intelligence*, Vol. 33, No. 1, pp. 1–64 (1987).

[23] 不二越（編）：知りたい油圧基礎編, ジャパンマシニスト社 (1978).

[24] Matsuda, K., Niina, H. and Mizoguchi, R.: An Interview System for Acquiring Design Knowledge based on Understanding of Examples, *Proceedings of PRICAI92*, pp. 854–860 (1992).

[25] 松田, 新名, 溝口：設計例の理解に基づく知識獲得インタビューシステム, 人工知能学会誌, Vol. 7, No. 6, pp. 1038–1048 (1992).

[26] Davis, R. and Lenat, D. B.: *Knowledge-Based Systems in Artificial Intelligence*, MacGrow-Hill (1982).

[27] Marcus, S.: Taking Backtracking with a Grain of SALT, *International Journal of Man-Machine Studies*, Vol. 26, No. 4, pp. 383–398 (1987).

[28] Minton, S.: Selectively Generalizing Plans for Problem-Solving, *Proceedings of IJCAI85*, pp. 596–599 (1985).

[29] Keller, R. M.: Defining Operationality for Explanation-Based Learning, *Proceedings of AAAI87*, pp. 482–487 (1987).

[30] 山村, 小林：EBLの複数例題下への拡張, 人工知能学会誌, Vol. 4, No. 4, pp. 389–397 (1989).

[31] Hoppe, H. U.: An Analysis of EBG and its Relation to Partial Evaluation: Lessons Learned, Technical report, Arbeitspapiere der GMD 572 (1991).