〔非公開〕

ＴＲ－Ｃ－００４３

演繹的学習について

柴田　健次
KENJI SHIBATA

１９９０．２．８

ＡＴＲ通信システム研究所

演繹的学習について

柴田 健次
ATR 通信システム研究所

1990 年 2 月 9 日

もくじ

1 はじめに

近年、知識処理システムであるエキスパートシステムは、実用化システムへと移行しつつあるが、知識獲得に要する労力が多大であるという問題が生じている。ソフトウェアの開発を対象としたエキスパートシステムにおいても例外ではなく、要求の理解からソースコードの作成にわたる広い範囲で自動化のためにはかなりの困難を伴っている。つまり、領域専門家自身が、問題解決に必要な知識を体系的に持っていないことや、明確な知識として確立されていないことが原因となり、知識獲得がボトルネックとなっている。このような状況から、システム自身が問題解決を通し、知識を学習していくことが要求されてきている。

特定の例から一般化された概念を形式化する問題は、長い間機械学習における研究の中心にある。ほとんどの過去の研究は一般化に経験的な手法を取っているが、この一般化は、領域特定の知識を使わずかなりの量の例を用いて行っている。しかし、ここ数年、領域特定の知識を適用し、単一の例から適切な一般化を行う手法が開発されてきている。

本稿では、このような演繹的学習の一つである説明に基づく学習 (EBL:Explanation Based Learning) に注目し、説明に基づく学習とはどのようなものかを実際のアルゴリズム及びそれを使った実際の例を用い解説する。また、説明に基づく学習自身の有する問題についても触れておく。

以下、2章では、説明に基づく学習の概要を説明し、3章で実際のアルゴリズムを紹介する。4章で通信への応用についてとりあげる。なお、2章は、参考文献 [1] を参照して作成したものである。

2 説明に基づく学習の概要

例題からの一般化の能力は、学習システムの主要な能力とみなされている。一般化には、一般的な概念の例題の認識と、これらの例との特徴の同一化が含まれ、これらの共通した特徴に基づく概念定義の形式化が含まれる。一般化のプロセスは、可能な概念定義空間の探索と見なせる。この空間は非常に広大であるので、一般化問題の中心は、この探索を制限するデータの利用にある。

一般化問題のほとんどの研究は、経験的で、データ依存型である。これは、探索を制限するためにかなりの量の例を用いている。これらの方法は総て、ある種の帰納的バイアスを用いている。このバイアスは、一般化機構に知識を供給することによって得られる。この知識は、学習された概念を説明するのに適切と仮定される特徴を持つ例題のものである。様々なアルゴリズムによって、許容される特徴に関して定義される制限された概念空間の探索が可能になり、例題と無矛盾の概念定義を決定することが可能になった。これらの方法は例題と共通の特徴を探索することに基づいているので、類似に基づく一般化手法と呼ぶ。

近年、何人もの研究者が一般化手法を提案している。この手法は、データ依存、類似に基づく手法とはまったく異にしている。正しい一般化のために探索を制限する方策として、沢山の例に依存したり帰納的バイアスによるものよりも、これらの最近の方法は、タスク領域の知識に依存し探索を制限している。この知識に関するひとつの例を解析した後、例題の適切な一般化を生成する。領域の知識及び学習に基づく概念に依存することにより、説明に基づく方法は帰納的あるいは類似に基づく方法に関連する基本的な困難を解消している。既存の方法では、一般化の正当性が立証できない。類似に基づく方法は一般化を導くための帰納的バイアスの形式によらなければならない。一方、説明に基づく方法は、その領域の知識によっており、一般化のより確かな手段を提供している。そして、個々の例から情報を引き出すことが可能である。また、それは、学習者がその領域の知識を持っていることを要求し、学習する概念を持つことを要求している。

2.1 説明に基づく一般化

説明に基づく一般化のキーは、説明能力を有する学習システムに与えられた正の訓練例の正しい一般化を作ることができるということである。特に、システムは、なぜ訓練例が検討中の概念の例になるのかを説明できなければならない。従って、一般化機構は、要求された説明をするための領域知識と共に検討中の概念の定義も持っていると仮定される。この節では、より正確に説明に基づく方法によってカバーされる一般化問題のクラスを定義し、EBG(Explanation Based Generalization) メソッドを定義し、具体例を用いて説明する。

2.1.1 説明に基づく一般化の問題

ここで考慮される一般化の問題を定義するために、いくつかの用語を定義する。

概念 (concept) : インスタンス世界の述語。インスタンスのサブセットを特徴付ける。
インスタンス : 特徴及び値を表すリテラルの集合で記述される。
概念定義 (concept definition) : 概念例の必要十分条件を記述する。
例あるいは正の例 (example or positive example) : 概念定義を満足するインスタンス
負の例 (negative example) : 概念定義を満足しないインスタンス
一般化 (generalization) : 例を含む集合を記述する概念定義
説明 (explanation) : インスタンスが概念の例になっているという説明は、例が概念定義を満足するという証明。
説明構造 (explanation structure) : 証明木。具現化されたルールを関連する一般的なルールに置き換えたもの。

説明に基づく学習の問題定義は、次の四つの情報を特定することである。

1. 目標概念
   学習される概念を定義する。例えば、表 1 に定義された問題において、タスクは、オブジェクトのペアを認識することである。オブジェクトのペア <x, y> は、y の上に x を乗せることができるというものである。ここでの目標概念 SAFE-TO-STACK は、FRAGIL と LIGHTER という述語に関して定義される。

2. 訓練例
   目標概念の正の例である。例えば、表 1 の訓練例は、オブジェクトのペア (box, endtable) を示している。ここで、box は endtable の上に乗せられている。

3. 領域知識
   ルールとファクトの集合を含む。それらは、訓練例が目標概念のメンバであるという説明を可能にする。例えば、この問題の領域知識は、FRAGIL と LIGHTER の定義を含み、DENSITY と VOLUME から WEIGHT のような特徴を推論するルールを含み、ENDTABLE の WEIGHT のようなデフォルト値を示唆するルールを含み、'0.1 は 5 より LESS' のようなファクトを含む。

4. 操作性規範
   学習結果である概念定義の満たすべき基準。学習された概念定義は、あるタスクをするためにある行為者によって使用される。行為者とタスクに対し、実行可能なように定義されなければならない。この問題に対する操作性規範は、最終的な概念定義は、訓練例に記述された述語に関して記述されるか、領域知識から容易に評価される述語で記述されるべきであるということである。

これらの四つが入力され、目標概念の十分な概念定義であり、操作性規範を満足する訓練例の一般化を決める。

2.1.2 EBGメソッド

上記の問題のクラスを解決するために設計されたEBGメソッドは、次のように定義される。

1. 説明：領域知識に関する説明を構成する。訓練例がどの様に目標概念定義を満足するかを証明する。
   説明は、説明構造の各枝が操作性規範を満足する表現になるように構成する。

2. 一般化：十分条件の集合を決定する。
   説明構造に対し、目標概念を逆向きに回帰することによって一般化を行う。

具体的にEBGメソッドがどの様に行われるのかを示すために、ある対象yの上に別のある対象xが置けるという概念SAFE-TO-STACK(x, y)を学習する問題を考える。この例は、OBJ1とOBJ2の二つの物理オブジェクトからなり、OBJ1はOBJ2の上にあり、OWNER, COLOR等によってオブジェクトの特徴が記述されている。表1にSAFE-TO-STACK一般化問題を示す。

表 1: The SAFE-TO-STACK 一般化問題

Given
目標概念:Pairs of objects <x, y> such that SAFE-TO-STACK(x, y),where
SAFE-TO-STACK(x, y)⇔ NOT(FRAGILE(y)) ∨ LIGHTER(x, y)
訓練例
ON(OBJ1, OBJ2)
ISA(OBJ1, BOX)
ISA(OBJ2, ENDTABLE)
COLOR(OBJ1, RED)
COLOR(OBJ2, BLUE)
VOLUME(OBJ1, 1)
DENSITY(OBJ1, .1)
OWNER(OBJ1, BONNIE)
OWNER(OBJ2, CLYDE)
FRAGILE(OBJ2, YES)
領域知識
VOLUME(p1, v1) ∧DENSITY(p1, d1)→WEIGHT(p1, v1*d1)
WEIGHT(p1, w1)∧WEIGHT(p2, w2)∧LESS(w1, w2)→LIGHTER(p1, p2)
ISA(p1, ENDTABLE)→WEIGHT(p1, 5)
LESS(.1, 5)
操作性規範
The concept definition must be expressed in terms of the predicates used to
describe examples or other selected, easily evaluated, predicates from the domain theory
Determine
A generalization of training example that is a sufficient concept definition for
the goal concept and that satisfies the operationality criterion

- EBG メソッドの第一ステップ

  第一ステップでは、訓練例が与えられると、そのうちのどれが目標概念を特徴づけるのに適切か、どれが不適切かを決定する。このために、訓練例がどのようにして目標概念を満足しているかの説明を構成する。図1に例の説明を示す。そこに示されるように、オブジェクトのペア <OBJ1, OBJ2> は、OBJ1 が OBJ2 よりも LIGHTER であるということから、目標概念 SAFE-TO-STACK を満たしている。OBJ1 が OBJ2 より LIGHTER ということは、OBJ1 と OBJ2 の WEIGHT が推論できることからが分かる。OBJ1 について、その WEIGHT は、DENSITY と VOLUME から推論される。OBJ2 について、その WEIGHT は、ENDTABLE のデフォルト値を特定するルールに基づいて推論される。

  このような推論により、説明構造は、OBJ1 と OBJ2 がどのように、目標概念定義を満足するかを説明する。この説明構造は、操作性規範を満たす表現が各枝のリーフになるように構成され、目標概念を満たすのに適した訓練例の特徴を選ぶ。この例では、適切な訓練例の特徴は、OBJ1 の VOLUME と DENSITY、OBJ2 の ISA であり、説明構造のリーフを集めた次の連結と対応する。

  $$\text{VOLUME(OBJ1, 1)} \wedge \text{DENSITY(OBJ1, 0.1)} \wedge \text{ISA(OBJ2, ENDTABLE)}$$

  EBG メソッドの第一ステップは、訓練例の適切な特徴を分離することであり、一般化の制約を決定するものではない。例えば、特徴 VOLUME(OBJ1, 1) は、現在の訓練例が目標概念をどの様に満足するかを説明するのに適切であるが、VOLUME に対して受け入れられる値の一般的な制約はまだ決定されない。

- EBG メソッドの第二ステップ

  EBG メソッドの第二ステップは、第一ステップで選択された特徴値の一般化である。これは、説明構造の各ステップで許容される特徴値の十分条件を決定することによって行う。説明に矛盾を起こさない一般的な十分条件を決定するために、説明構造を一ステップづつ目標概念に後戻りをしてゆく。

  図2に、EBG メソッドの第二ステップを示す。式の各結合は、次に示すように適切な規則により各々回帰される。結合は、規則の右辺と、ユニファイされ、代入セット（特に変数のバインド）が作られる。例と矛盾のない代入は、ルールの左辺に適用され、その結果回帰された式を得る。ルールの右辺とユニファイされない式の結合は、単に回帰された式に追加される。

  1. SAFE-TO-STACK(x, y) は、規則 LIGHTER(p1, p2)→SAFE-TO-STACK(p1, p2) によって、目標回帰され、LIGHTER(x, y) が、SAFE-TO-STACK(x, y) を推論する十分条件であることが決定される。このとき、次の代入が行われる。 [x/p1, y/p2]
  2. LIGHTER(x, y) は、規則 WEIGHT(p1, w1)∧LESS(w1, w2)∧WEIGHT(p2, w2)→LIGHTER(p1, p2) によって、目標回帰され、WEIGHT(x, w1)∧LESS(w1, w2)∧WEIGHT(y, w2) が、LIGHTER(x, y) を推論する十分条件であることが決定される。このとき、次の代入が行われる。 [x/p1, y/p2]
  3. WEIGHT(x, w1) は、規則 VOLUME(p1, v1)∧DENSITY(p1, d1)→WEIGHT(p1, v1*d1) によって、目標回帰され、VOLUME(x, v1)∧DENSITY(x, d1) が、WEIGHT(x, w1) を推論する十分条件であることが決定される。このとき、次の代入が行われる。 [x/p1, v1*d1]
  4. WEIGHT(y, w2) は、規則 ISA(p2, ENDTABLE)→WEIGHT(p2, 5) によって、目標回帰され、ISA(y, ENDTABLE) が、WEIGHT(y, w2) を推論する十分条件であることが決定される。このとき、次の代入が行われる。 [y/p2, 5/w2]
  5. LESS(w1, w2) と、ユニファイする右辺を持つ規則はないので、他の式の回帰によって得られた代入を適用して、回帰された式に追加する。この場合、代入セットは、[x/p1, v1*d1/w1, y/p2, 5/w2] であり、三番目の結合 LESS (v1*d1, 5) を得る。

結局、SAFE-TO-STACK(x, y) の実行可能定義は

VOLUME(x, v1)∧DENSITY(x, d1)∧LESS(v1∗d1, 5)
∧ISA(y, ENDTABLE)→SAFE-TO-STACK(x, y)

となる。

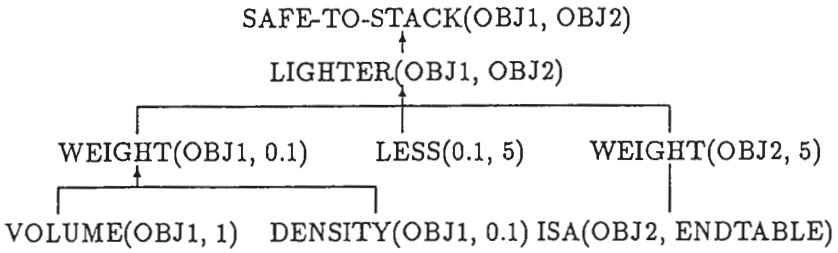

図 1: EBG メソッドの第一ステップの説明

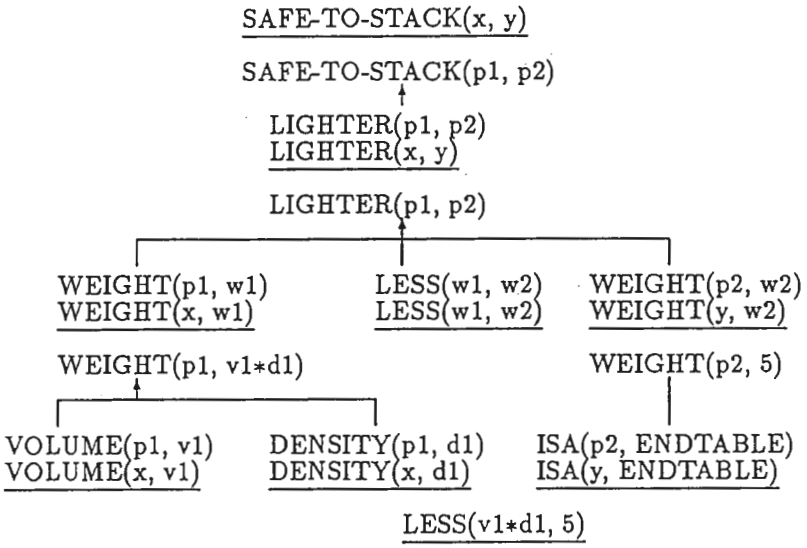

図 2: EBG メソッドの第二ステップの説明

2.2 説明に基づく一般化の特徴

EBG メソッドに関するポイントを示す。

- EBG メソッドは、2 ステップのプロセスで、一つの訓練例から正しい一般化を作り出す。第一ステップは、与えられた訓練例の説明を作る。この説明は、例の中で、適切な特性値とそうでないものとを分ける。第二ステップは、この説明を解析し、特定の制約を決定し、的確な一般化を行った結果を獲得する。このようにして、単一の例題から正しい学習を行う EBG のような説明に基づく方法は、類似に基づく方法の正当な一般化を作れないという限界を打ち破っている。

- 学習者が、そのドメインの有効な知識を持ち、目標概念を持ち、操作性規範を持つと仮定することによって達成される。

- 説明に基づく一般化は、目標概念、訓練例、領域知識、操作性規範を入力とし、目標概念の定義を満たし、操作性規範を満足するような訓練例の一般化された記述を獲得することである。

- EBG によって得られた概念定義は、概念の直接的な再表現というよりも、目標概念の特化である。説明構造が、与えられた訓練例で作られ、目標概念の総ての可能な例を説明しているわけではなく、この説明を解析することで、生成される一般化は、説明される例だけをカバーしていることによる。

- 目標回帰アルゴリズムは、説明が適用できる十分条件だけを計算するので、概念の特化を導く。EBG のこの制限は、今後の研究に興味ある問題のあることを示唆している。

2.3 問題点及び研究課題

2.3.1 不十分な知識の問題

EBG の重要な仮定は、訓練例が目標概念のメンバである事を証明するのに十分な領域知識が存在していると言う事である。つまり、導かれる一般化は、学習者がすでに知っている事から得られると言う事である。

現実世界の学習タスクの大部分において、学習者がそのようなしっかりとした知識をはじめから持っていると仮定する事は現実的ではない。要求される領域知識が複雑な、記述するのがむずかしい、あるいは未知の具体的な例は容易に考えられる。株式状況や、天気を予測する一般的な規則を導くような一般化の問題において、経済理論や気象理論は絶対的な予測規則を作るのに不十分である事は明らかである。このように、説明に基づく一般化の主要な研究は、学習プロセスとしての不完全な知識を改良する方法とともに一般化を導く不完全な領域知識を利用する方法である。不完全な知識を扱う問題は、さまざまな問題のクラスに効果があるであろう。

- 不完全な知識の問題
  株式状況や天気予報の例は、ともに不完全な知識の問題である。ここでの問題は、そのような知識が、訓練例が目標概念のメンバであると言う事を証明するのに十分でないと言う事である。しかしながら、不完全な知識でも訓練例と目標概念とを結び付けるもっともらしい説明を構成する事は可能である。例えば、経済の弱い理論でも会社の'現金手元有高'が目標概念「12 カ月で倍の株式」に関連するであろうと言う事を示唆できる。このように、量的なものよりも質的なものを伴ったもっともらしい因果関係の情報を含んでさえいれば不完全な知識でも一般に重要なガイダンスを与える事ができる。そのような不完全な知識を使用あるいは洗練する方法は、説明に基づく一般化の理解向上の主要ステップとなろう。

- 矛盾のある知識の問題

  三番目のむずかしい点は、矛盾した記述が導かれ得る知識において生ずる。SAFE-TO-STACK 問題における領域知識にそのような知識の一例がある。この知識が、end-table の重さを推論するデフォルトルールを持ち、また、既知の密度と体積から重さを計算する規則をも持っている。この知識は、密度と体積が知られており、これらが重さについてデフォルトの仮定と矛盾している end-table に対して二つの異なった重さを導くであろう。このような場合、一つの訓練例に対し矛盾のある説明を作ってしまう。このため、デフォルトの仮定があらゆる領域の知識において常識であるため、矛盾した知識を扱う事、矛盾した説明の取扱いに関する問題もまた重要な研究課題の一つである。

2.3.2 説明に基づく方法と類似に基づく方法の結合

EBG は、一つの例から概念定義を演繹的に推論するが、類似に基づく方法は、多数の訓練例から概念定義を機能的に推論する。明らかに、概念定義を推論するための領域知識と複数の訓練例を利用できる結合された方法を開発する事が望ましい。一般化に関するこの種の結合のアプローチは、不完全な知識だけの領域においても欠く事ができないであろう。

類似に基づく方法と説明に基づく方法とを結合させるために、以下のような方法が検討されている。

- 経験的な一般化を決定するのに類似に基づく一般化を使用し、説明に基づく一般化によって検証、修正する。このアプローチは、類似に基づく技法が、たくさんの訓練例から可能な一般化候補集合を生成できるという利点を持っている。そのような経験的な一般化が形式化されると、説明に基づく方法はシステムの別の知識を使ってそれらを簡潔に洗練する助けとなる。

- 説明に基づく方法を各訓練例に適用し、類似に基づく一般化の技法を使って、一般化された例を結合する。例えば、多数の訓練例の説明に基づく一般化を結合するのに、バージョン空間方法を使う事を考える。正しい訓練例による説明に基づく一般化は目標概念に対する十分な条件を構成するので、バージョン空間の特定の境界集合を修正する一般化された正しい例として使用できる。同様に、説明に基づく一般化を用いる事によって負の訓練例を一般化する事も想像できる。その結果、一般化された負の訓練例は、バージョン空間の境界集合を修正（特化）することに使用できる。しかし、この結合の方法正当でない一般化を作ってしまう可能性を有している。

- 一般化を結合するために、複数の例の説明をマージする。一般化は、マージされた説明によって正当化される。二つの異なった例に対する説明が与えられると、次のように説明を結合する。二つの説明の共通の系列は結合された説明に不変として残す。結合された説明において選言的な表現になるか、より共通の表現に一般化される系列を分ける。

3 実行アルゴリズム

本章では、前章で紹介した説明に基づく一般化を実現するアルゴリズム [2] について説明する。はじめに、EGGS を簡単に説明し、実行結果を示す。

3.1 EGGS の概要

EGGS は、DEDUCE と呼ばれる演繹的検索機によって得られる証明を、一般化し、証明木からマクロルールを生成する、説明に基づく一般化機構である。

3.1.1 DEDUCE

DEDUCE は、演繹的な検索システムであり、一回に一つの答えを検索し、ジェネレータを使って、証明を生成する。現在は、後向き推論のみをサポートしている。変数形式およびルール形式は、以下のようになっている。ここで、ルールは、述語名で検索される。

変数形式：? を項の先頭につける

ルール形式：(< − CONSEQUENT. ANTECEDENTS)

基本的な関数を以下に示す。

- (index-brules): ルールを入力する
- (index-facts): 事実を入力する
- (clear-database): データベースをクリアする
- (dump-database): データベースを表示する
- (forget <predicate>): 指定される <predicate> の新しく学習されたルールをデータベースから取り除く。
- (retrieve <form><depth-bound>): ルールとファクトの集合が入力されると、要求が検索され、答えの一般化されたリストを返す。検索結果は、(<bindings><proof>) の形式であり、<bindings> には、変数バインディングのリスト、<proof> には、<form> にマッチするデータベースの事実か、ルールに基づく証明が格納される。ルールに基づく証明は、次の形式をしている。
  (rule-proof <consequent> ((<antecedent><proof>)(<antecedent><proof>)···))
  この証明構造は、説明構造を構成し、一般化された説明を生成するのに使用される。

3.1.2 EGGS

EGGS を実行するには、

- clm11:>ebl ディレクトリの deduce.lisp と eggs.lisp の 2 つのファイルをロードする。
- 必要なルール及び事実をロードする。例:clm11:>ebl ディレクトリの toy-data.lisp をロード
- EGGS への入力形式は、(eggs <form>)。例:(eggs '(safe-to-stack obj1 obj2))

始めに、与えられた <form> 、例えば (safe-to-stack obj1 obj2)、とマッチする事実を検索する。答えが見つかれば、その証明構造が新しいルールを作るために一般化され得るかどうかを調べる。もしできれば、それを一般化し、一般化された証明から、マクロルールを作り、既存のルールの先頭に追加する。2 章で説明した例のルール及び事実は、toy-data.lisp に含まれている。

3.2 実行例

図 3に 2章での説明に取り上げた safe-to-stack という概念の学習の例を示す。図の実行例は、 *trace-eggs* 、 *trace-generalizer* 、 *learn* の各変数を t に設定した時の例である。 *trace-eggs* を t にすることにより、 Proof、 Proof Structure を表示する。 *trace-generalizer* を t にすることにより、 Generalizing proof において、 Updated bindings を表示する。 *learn* を t にすることにより、 generalizing proof し、マクロルールを獲得する。

```
Command: (eggs '(safe-to-stack obj1 obj2))

Deduction time: 1.68 sec
Retrieved answer:(SAFE-TO-STACK OBJ1 OBJ2)

Proof:
((SAFE-TO-STACK OBJ1 OBJ2)
 ((LIGHTER OBJ1 OBJ2) ((WEIGHT OBJ1 0.1) (VOLUME OBJ1 1) (DENSITY OBJ1 0.1) (TIMES 1 0.1 0.1))
  ((WEIGHT OBJ2 5) (ISA OBJ2 ENDTABLE)) (LESS 0.1 5)))
Proof Structure:
(RULE-PROOF (SAFE-TO-STACK ?#:G60370 ?#:G60371)
 (((LIGHTER ?#:G60370 ?#:G60371)
   (RULE-PROOF (LIGHTER ?#:G60372 ?#:G60373)
    (((WEIGHT ?#:G60372 ?#:G60374)
      (RULE-PROOF (WEIGHT ?#:G60376 ?#:G60377)
       (((VOLUME ?#:G60376 ?#:G60378) (VOLUME OBJ1 1))
        ((DENSITY ?#:G60376 ?#:G60379) (DENSITY OBJ1 0.1))
        ((TIMES ?#:G60378 ?#:G60379 ?#:G60377) (TIMES 1 ?X ?X)))))
     ((WEIGHT ?#:G60373 ?#:G60375)
      (RULE-PROOF (WEIGHT ?#:G60384 5) (((ISA ?#:G60384 ENDTABLE) (ISA OBJ2 ENDTABLE)))))
     ((LESS ?#:G60374 ?#:G60375) (LESS 0.1 5)))))))

Generalizing proof

(WEIGHT ?#:G60372 ?#:G60374)=(WEIGHT ?#:G60376 ?#:G60377)
Updated bindings: (T (?#:G60374 ?#:G60377) (?#:G60372 ?#:G60376))

(WEIGHT ?#:G60373 ?#:G60375)=(WEIGHT ?#:G60384 5)
Updated bindings: (T (?#:G60375 5) (?#:G60373 ?#:G60384) (?#:G60374 ?#:G60377)
                    (?#:G60372 ?#:G60376))

(LIGHTER ?#:G60370 ?#:G60371)=(LIGHTER ?#:G60372 ?#:G60373)
Updated bindings: (T (?#:G60371 ?#:G60373) (?#:G60370 ?#:G60372) (?#:G60375 5)
                    (?#:G60373 ?#:G60384) (?#:G60374 ?#:G60377) (?#:G60372 ?#:G60376))

General proof:
((SAFE-TO-STACK ?#:G60376 ?#:G60384)
 ((LIGHTER ?#:G60376 ?#:G60384)
  ((WEIGHT ?#:G60376 ?#:G60377) (VOLUME ?#:G60376 ?#:G60378) (DENSITY ?#:G60376 ?#:G60379)
   (TIMES ?#:G60378 ?#:G60379 ?#:G60377))
  ((WEIGHT ?#:G60384 5) (ISA ?#:G60384 ENDTABLE)) (LESS ?#:G60377 5)))

Learned rule: (<- (SAFE-TO-STACK ?#:G60376 ?#:G60384) (VOLUME ?#:G60376 ?#:G60378)
                (DENSITY ?#:G60376 ?#:G60379) (TIMES ?#:G60378 ?#:G60379 ?#:G60377)
                (ISA ?#:G60384 ENDTABLE) (LESS ?#:G60377 5))
(SAFE-TO-STACK OBJ1 OBJ2)
```

図 3: safe-to-stack の実行例

4 通信への応用

本章では、説明に基づく一般化の二番目の例を示す。この例は、モデムと端末が存在し、ある信号をゆっくり送ったり、速く送ったりしたい要求があった場合に、モデムや、端末の設定をどのようにすればよいのかを学習する。つまり、ある要求に対する属性を決定する例である。以下に、モデムの属性決定問題を示す。

- 目標概念
  あるモデムと端末間でのある情報の送り方の概念である。一例として、「ゆっくり送る」ということは、低速で送り、しかも、誤りなしに送るということであると定義している。

  ゆっくり送る (x, y, z)⇔ 低速 (x, y, z)∧ 誤りなし (x, y, z)
  x: モデム、 y: 端末、 z: 情報

- 領域知識
  低速とはどういうことか、誤りなしに送るとはどういうことか、さらに、それらを実現するためにはどのレジスタをどのように操作するか等の知識が含まれ、要求から、具体的な操作を導く規則を有している。

  1. 伝送スピード (x, x1)∧isa(x, modem)∧slow(x1)→ 低速 (x, y, z).
  2. 操作 (x, s50, 1)→ 伝送スピード (x, 300).
  3. 操作 (x, s50, 2)→ 伝送スピード (x, 1200).
  4. 操作 (x, s50, 3)→ 伝送スピード (x, 2400).
  5. インタフェーススピード条件 (x, y)∧ フロー制御条件 (x, y)→ 誤りなし (x, y, z).
  6. modem-dte- スピード (x, x1)∧isa(y, dte)∧dte- スピード (y, x2)∧same(x1, x2)→ インタフェーススピード条件 (x, y).
  7. rd- フロー制御 (x, x1)∧td- フロー制御 (x, x2)∧same(x1, x2)→ フロー制御条件 (x, y).
  8. 操作 (x, s58, 0)→rd- フロー制御 (x, null).
  9. 操作 (x, s58, 1)→rd- フロー制御 (x, half-duplex).
  10. 操作 (x, s58, 2)→rd- フロー制御 (x, full-duplex).
  11. 操作 (x, s58, 3)→rd- フロー制御 (x, xon-xoff).
  12. 操作 (x, s58, 4)→rd- フロー制御 (x, xon-xoff-full-duplex).
  13. 操作 (x, s58, 5)→rd- フロー制御 (x, enq-acq).
  14. 操作 (x, s58, 6)→rd- フロー制御 (x, enq-acq-xon-xoff).
  15. 操作 (x, s68, 0)→td- フロー制御 (x, null).
  16. 操作 (x, s68, 1)→td- フロー制御 (x, half-duplex).
  17. 操作 (x, s68, 2)→td- フロー制御 (x, full-duplex).
  18. 操作 (x, s68, 3)→td- フロー制御 (x, xon-xoff).
  19. 操作 (x, s68, 4)→td- フロー制御 (x, xon-xoff-full-duplex).
  20. 操作 (x, s68, 5)→td- フロー制御 (x, enq-acq).
  21. 操作 (x, s68, 6)→td- フロー制御 (x, enq-acq-xon-xoff).
  22. 操作 (x, s68, 255)→td- フロー制御 (x, equal-rd).

23. セッティング可能(x, s51)∧操作(x, s51, 0)→modem-dte-スピード(x, 300).

24. セッティング可能(x, s51)∧操作(x, s51, 1)→modem-dte-スピード(x, 1200).

25. セッティング可能(x, s51)∧操作(x, s51, 2)→modem-dte-スピード(x, 2400).

26. セッティング可能(x, s51)∧操作(x, s51, 3)→modem-dte-スピード(x, 4800).

27. セッティング可能(x, s51)∧操作(x, s51, 4)→modem-dte-スピード(x, 9600).

28. セッティング可能(x, s51)∧操作(x, s51, 5)→modem-dte-スピード(x, 19200).

29. セッティング可能(x, s51)∧操作(x, s51, 255)→modem-dte-スピード(x, auto).

30. セッティング可能(x, s66)∧操作(x, s66, 1)→セッティング可能(x, s51).

31. セッティング可能(x, s50)∧操作(x, s50, 255)→セッティング可能(x, s51).

32. same(x, x).

- 訓練例

  対象物の設定および具体的な属性値等を記述する。

  伝送スピード(modem1, 2400).
  dte-スピード(dte1, 2400).
  modem-dte-スピード(modem1, 2400).
  rd-フロー制御(modem1, xon-xoff).
  td-フロー制御(modem1, xon-xoff).
  isa(modem1, modem).
  isa(dte1, dte).

- 操作性規範

  いろいろと設定することが可能であるが、例えば、モデム等の設定方法を理解するという視点に立てば、操作という述語を結果から抽出することを要求する。

これらの4項目から、モデムxと端末yとの間で情報zを「ゆっくり送る」ためには、EGGシステム[2]を用いると次のように属性を決定すればよいことが導かれる。

伝送スピード(x, x1)∧isa(x, modem)∧slow(x1)
∧modem-dte-スピード(x, s1)∧isa(y, dte)∧dte-スピード(y, s1)
∧rd-フロー制御(x, c1)∧td-フロー制御(x, c1)→ゆっくり送る(x, y, z).

この一般化式は、以下のことを示している。

- 伝送スピードの値は、slowで定義される値。
- rd-フロー制御とtd-フロー制御が同値。
- dte-スピードとmodem-dte-スピードが同値。

図4は、モデムと端末間で音をゆっくり送るということが、どのようにすることにより低速で、誤りなしになるかということを示す説明構造である。図5, 6にモデムの属性決定の実行例を示す。

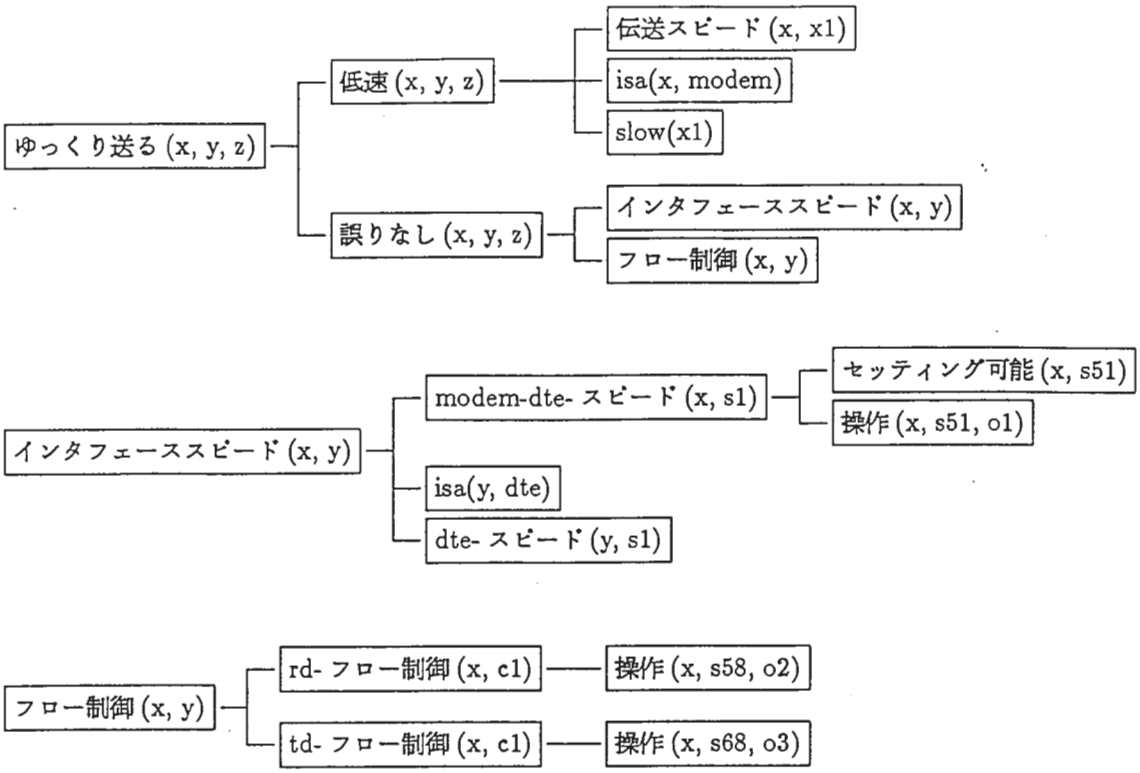

図 4: モデムの属性決定の説明構造

```
(eggs '(ゆっくり送る modem1 dte1 oto1))

Deduction time: 2.19 sec
Retrieved answer:(ゆっくり送る MODEM1 DTE1 OTO1)

Proof:
((ゆっくり送る MODEM1 DTE1 OTO1)
 ((低速 MODEM1 DTE1 OTO1) (伝送スピード MODEM1 2400) (ISA MODEM1 MODEM) (SLOW 2400))
 ((誤りなし MODEM1 DTE1 OTO1)
  ((インタフェーススピード条件 MODEM1 DTE1) (MODEM-DTE- スピード MODEM1 2400) (ISA DTE1 DTE) (DTE- スピー
ド DTE1 2400)
   ((SAME 2400 2400)))
  ((フロー制御条件 MODEM1 DTE1) (RD- フロー制御 MODEM1 NULL) (TD- フロー制御 MODEM1 NULL) ((SAME NULL NULL)))))
Proof Structure:
(RULE-PROOF (ゆっくり送る ?#:G60579 ?#:G60580 ?#:G60581)
 (((低速 ?#:G60579 ?#:G60580 ?#:G60581)
   (RULE-PROOF (低速 ?#:G60582 ?#:G60583 ?#:G60584)
    (((伝送スピード ?#:G60582 ?#:G60585) (伝送スピード MODEM1 2400))
     ((ISA ?#:G60582 MODEM) (ISA MODEM1 MODEM)) ((SLOW ?#:G60585) (SLOW 2400)))))
  ((誤りなし ?#:G60579 ?#:G60580 ?#:G60581)
   (RULE-PROOF (誤りなし ?#:G60586 ?#:G60587 ?#:G60588)
    (((インタフェーススピード条件 ?#:G60586 ?#:G60587)
      (RULE-PROOF (インタフェーススピード条件 ?#:G60589 ?#:G60590)
       (((MODEM-DTE- スピード ?#:G60589 ?#:G60591) (MODEM-DTE- スピード MODEM1 2400))
        ((ISA ?#:G60590 DTE) (ISA DTE1 DTE))
        ((DTE- スピード ?#:G60590 ?#:G60592) (DTE- スピード DTE1 2400))
        ((SAME ?#:G60591 ?#:G60592) (RULE-PROOF (SAME ?#:G60593 ?#:G60593) NIL)))))
     ((フロー制御条件 ?#:G60586 ?#:G60587)
      (RULE-PROOF (フロー制御条件 ?#:G60594 ?#:G60595)
       (((RD- フロー制御 ?#:G60594 ?#:G60596) (RD- フロー制御 MODEM1 NULL))
        ((TD- フロー制御 ?#:G60594 ?#:G60597) (TD- フロー制御 MODEM1 NULL))
        ((SAME ?#:G60596 ?#:G60597) (RULE-PROOF (SAME ?#:G60598 ?#:G60598) NIL)))))))))))
```

図 5: モデムの実行例

```
Generalizing proof

(低速 ?#:G60579 ?#:G60580 ?#:G60581)=(低速 ?#:G60582 ?#:G60583 ?#:G60584)
Updated bindings: (T (?#:G60581 ?#:G60584) (?#:G60580 ?#:G60583) (?#:G60579 ?#:G60582))

(SAME ?#:G60591 ?#:G60592)=(SAME ?#:G60593 ?#:G60593)
Updated bindings: (T (?#:G60592 ?#:G60593) (?#:G60591 ?#:G60593) (?#:G60581 ?#:G60584)
                    (?#:G60580 ?#:G60583) (?#:G60579 ?#:G60582))

(インタフェーススピード条件 ?#:G60586 ?#:G60587)=(インタフェーススピード条件 ?#:G60589 ?#:G60590)
Updated bindings: (T (?#:G60587 ?#:G60590) (?#:G60586 ?#:G60589) (?#:G60592 ?#:G60593)
                    (?#:G60591 ?#:G60593) (?#:G60581 ?#:G60584) (?#:G60580 ?#:G60583)
                    (?#:G60579 ?#:G60582))

(SAME ?#:G60596 ?#:G60597)=(SAME ?#:G60598 ?#:G60598)
Updated bindings: (T (?#:G60597 ?#:G60598) (?#:G60596 ?#:G60598) (?#:G60587 ?#:G60590)
                    (?#:G60586 ?#:G60589) (?#:G60592 ?#:G60593) (?#:G60591 ?#:G60593)
                    (?#:G60581 ?#:G60584) (?#:G60580 ?#:G60583) (?#:G60579 ?#:G60582))

(フロー制御条件 ?#:G60586 ?#:G60587)=(フロー制御条件 ?#:G60594 ?#:G60595)
Updated bindings: (T (?#:G60590 ?#:G60595) (?#:G60589 ?#:G60594) (?#:G60597 ?#:G60598)
                    (?#:G60596 ?#:G60598) (?#:G60587 ?#:G60590) (?#:G60586 ?#:G60589)
                    (?#:G60592 ?#:G60593) (?#:G60591 ?#:G60593) (?#:G60581 ?#:G60584)
                    (?#:G60580 ?#:G60583) (?#:G60579 ?#:G60582))

(誤りなし ?#:G60579 ?#:G60580 ?#:G60581)=(誤りなし ?#:G60586 ?#:G60587 ?#:G60588)
Updated bindings: (T (?#:G60584 ?#:G60588) (?#:G60583 ?#:G60587) (?#:G60582 ?#:G60586)
                    (?#:G60590 ?#:G60595) (?#:G60589 ?#:G60594) (?#:G60597 ?#:G60598)
                    (?#:G60596 ?#:G60598) (?#:G60587 ?#:G60590) (?#:G60586 ?#:G60589)
                    (?#:G60592 ?#:G60593) (?#:G60591 ?#:G60593) (?#:G60581 ?#:G60584)
                    (?#:G60580 ?#:G60583) (?#:G60579 ?#:G60582))
 General proof:
((ゆっくり送る ?#:G60594 ?#:G60595 ?#:G60588)
 ((低速 ?#:G60594 ?#:G60595 ?#:G60588) (伝送スピード ?#:G60594 ?#:G60585) (ISA ?#:G60594 MODEM)
  (SLOW ?#:G60585))
 ((誤りなし ?#:G60594 ?#:G60595 ?#:G60588)
  ((インタフェーススピード条件 ?#:G60594 ?#:G60595) (MODEM-DTE- スピード ?#:G60594 ?#:G60593) (ISA ?#:G60595 DTE)
   (DTE- スピード ?#:G60595 ?#:G60593) ((SAME ?#:G60593 ?#:G60593)))
  ((フロー制御条件 ?#:G60594 ?#:G60595) (RD- フロー制御 ?#:G60594 ?#:G60598) (TD- フロー制御 ?#:G60594 ?#:G60598)
   ((SAME ?#:G60598 ?#:G60598)))))

Learned rule: (<- (ゆっくり送る ?#:G60594 ?#:G60595 ?#:G60588) (伝送スピード ?#:G60594 ?#:G60585)
                (ISA ?#:G60594 MODEM) (SLOW ?#:G60585) (MODEM-DTE- スピード ?#:G60594 ?#:G60593)
                (ISA ?#:G60595 DTE) (DTE- スピード ?#:G60595 ?#:G60593)
                (RD- フロー制御 ?#:G60594 ?#:G60598) (TD- フロー制御 ?#:G60594 ?#:G60598))
(ゆっくり送る MODEM1 DTE1 OTO1)
```

図 6: モデムの実行例 (続き)

5 おわりに

演繹的学習について、説明に基づく一般化を中心に具体例をあげて説明した。また、それを実現する処理系の紹介をした。説明に基づく一般化は、多くの訓練例を必要とし、一般化の正当性の保証の根拠が弱い類似に基づく学習の欠点を克服するべく考えられた、領域知識を用いた学習方法である。2章で指摘したように、説明に基づく一般化に有する問題も存在するが、現在、類似に基づく学習との融合などが研究されつつあり、知識獲得あるいは知識洗練化の手法として研究が行われている。

参考文献

[1] Mitchell, T.M., et al. : Explanation-Based Generalization:A Unifying View, Machine Learning, Vol.1, No.1, pp.47-80(1986)

[2] Mooney, R.J., et al. : A Domain Independent Explanation-based Generalizer, Proc. AAAI-86, Morgan Kaufmann, pp.551-555(1986)

[3] 白井 康之他 : 複数の説明構造の階層化と変数汎化に基づく知識獲得と洗練化, 第9回知識工学シンポジウム (1989)

[4] 小林 重信他 : 知識獲得と学習の展望, 第8回人工知能セミナー (1989)