様々な知識発見システムを統一的に利用可能な環境の構築

A Software Environment for Operating Various Discovery Systems based on the Logic of Discovery

菊池 敏幸

Kikuchi Toshiyuki

北海道大学 工学研究科・知識メディアラボラトリー

Faculty of Technology and Meme Media Laboratory, Hokkaido University

kikuchi@meme.hokudai.ac.jp, <http://ca.meme.hokudai.ac.jp/people/kikuchi/>

山本 章博

Yamamoto Akihiro

北海道大学 工学研究科・知識メディアラボラトリー / 科学技術振興事業団 さきがけ研究 21

Faculty of Technology and Meme Media Laboratory, Hokkaido University / PRESTO, Japan Science and Technology Corporation

yamamoto@meme.hokudai.ac.jp, <http://ca.meme.hokudai.ac.jp/people/yamamoto/>

keywords: knowledge discovery, logic of discovery, inductive logic programming

Summary

The goal of our research is to develop a software environment in which users can select appropriate Knowledge Discovery systems from various candidates, and compare each obtained result with others. This paper reports a prototype of such an environment, named EVLD (Environment for Various Logics of Discovery). The key idea in constructing EVLD is introducing three-level architecture into it. We adopt the logic of discovery proposed by Plotkin for the middle level, and put every Knowledge Discovery system on the lowest level. By designing the transformation of the two levels, users of EVLD obtain outputs of every discovery system in the form of logic programs. They can compare the outputs by comparing the logic programs. Moreover the logic programs are stored in a rule base, and can be used as a background theory for another trial of discovery systems. Though the logic of discovery was invented thirty years ago with philosophical consideration of discovery, it includes Knowledge Discovery from relational databases in first-order logic. Therefore constructing EVLD is connecting recent Knowledge Discovery systems with Plotkin's philosophical consideration.

1. は じ め に

本研究の目標は、機械学習や知識発見の分野においてこれまで開発された様々な知識発見システムを、利用者が適切に選択して利用し、大量に蓄積された情報から有意義な知識を引き出せるようにするためのソフトウェア環境を構築することである。そのプロトタイプとして本稿では、EVLD (Environment for Various Logics of Discovery) を構築し、Plotkin [Plotkin 71] が考案した発見の論理を基盤にすれば、機械学習・知識発見システムを統合することが可能であることを実証する。

計算機技術が急速に発展している今日、個人、特に研究者が収集し、蓄積するデータは急増している。機械学習や知識発見の技術は、人手で探索するにはあまりにも大量のデータから有益な情報を得るために不可欠と認識されている。データベース・システム内に蓄積されたデータから一般に成り立つ有用かつ未知の規則を引き出す行為は、データベースからの知識発見 (Knowledge Discovery in Database, KDD) とよばれ、その手法は近年盛んに研

究されている。

機械学習・知識発見そのものの研究者にとっては、個々のシステムがいかにして適切な (らしい) 仮説を生成するかが興味の対象である。したがって、そのための方式を的確に表現できるような入出力形式を設定すればよい。既に様々なシステムが開発・提案されているが、入力となるデータ形式や出力となる知識 (仮説) の形態は稼動するシステムごとに異なっている。あるものは関係データベース中のテーブルを入力とし、あるものは文字列データを入力とする。出力される仮説の形態も、決定木、相関規則、論理プログラムなど様々である。

しかし、一般の利用者やこれらの分野以外の研究者にとって、機械学習・知識発見によって得られる結果は入力データの一つに対する“見方”を与えたにすぎない。そして、その“見方”こそがシステムの特性である。これはシステムの出力が“仮説”であることから自然な帰結である。通常は、一つのデータから得られる知識は一つだけとは限らないので、様々な“見方”を比較検討できることが望まれる。知識発見システムが前述のようなヘテロ

図 1 統合環境 EVLD の概念構造

な状態では、一般の利用者が様々な知識発見システムを統一的に利用したり、自らの利用すべきシステムを比較検討することは困難である。

本研究では、ある領域に関する知識発見を行って得られた仮説を用いたアプリケーションを開発する人を利用者とし、入力を関係データベースの関係（テーブル）としておきさえすれば、様々な機械学習・知識発見システムから比較可能な形の仮説を得られるようなソフトウェア環境を構築する。一般の利用者（特定の領域に関する様々な仮説が必要な人）はその環境下のアプリケーションを用いることで様々な知識発見システムの利用が可能になる。

一つの機械学習・知識発見システムでも、属性の指定方法によって異なる結果を得ることもあるため、関係データベースと機械学習・知識発見システムとの接続部分も利用者が柔軟かつ容易に設計・変更が可能な形にする必要がある。そこで、学習・発見システムにデータベース理論のような3層スキーマ構造を導入する。そしてその概念スキーマ（もしくは論理構造）として、論理プログラムに基づく発見の論理を採用する。関係データベースの概念スキーマとして普及している関係モデルは事実を記述するためのスキーマであり、仮説を記述するには適していない。

述語論理を用いれば、決定木や相関規則などは論理式という様な形で表現でき、しかも複数の仮説の比較を証明手続きによって定義することが可能である。したがって利用者は、もし必要ならば定理証明システムや論理プログラム処理系を利用して得られた仮説の中でどれが一般的なのかを判定することも可能となる。また、上述の接続部分の構成も述語で表現でき、仮説の可読性が高くなる。

本稿は次のように構成されている。次章では、本稿で提案する環境 EVLD の構成の概要について述べ、構築した

環境を用いて実験した結果を示す。3章では、概念スキーマとしての Plotkin の発見の論理について説明する。4章では、知識発見システムを発見の論理の下位層に置く手法を議論する。5章では、発見の論理の上位層に位置するアプリケーション例について説明する。6章では、EVLD を構築する際に用いた技法について述べる。7章では、まとめと今後の課題について述べる。

2. 統合環境の概要

本研究で提案するソフトウェア環境 EVLD の構成を図 1 に示す。図において矢印はデータの流れを示している。

利用者はまず関係データベースから知識発見の対象となるデータを選択する。そしてデータからの知識発見に必要な補足情報を入力し、仮説を生成させる知識発見システムを選択する。

EVLD は入力された情報を基に SQL 文を生成し、関係データベースからデータを引き出す。引き出されたデータ及び入力された補足情報を用いて選択された各知識発見システムへデータを変換して受け渡す。その際データは各システムに適切な入力フォーマットに変換される。そして知識発見システムを実行し、得られた仮説を論理プロ

ID	class	size	color	animal
a	dangerous	small	black	bear
b	dangerous	medium	black	bear
c	dangerous	large	black	dog
d	safe	small	black	cat
e	safe	medium	black	horse
f	dangerous	large	black	horse
g	dangerous	large	brown	horse

図 2 動物に関する関係 R_{animal}

グラムに変換して出力する。生成された論理プログラムはルールベースに蓄えられ、次の知識発見システムを利用する際に背景知識として利用される。現状で選択可能な知識発見システムは C4.5[Quinlan 93], Progol[Muggleton 95], FOIL[Quinlan 90] である。なお、本稿で扱うデータは離散値だけである。

実際に図 2 に示す関係より複数の知識発見システムを用いて仮説を生成した結果を下に示す。

C4.5 の出力した仮説 $H_{animal}^{C4.5}$

$$\begin{aligned} safe(X) &\leftarrow horse(X), medium(X) \\ safe(X) &\leftarrow horse(X), small(X) \\ safe(X) &\leftarrow cat(X) \\ dangerous(X) &\leftarrow horse(X), large(X) \\ dangerous(X) &\leftarrow dog(X) \\ dangerous(X) &\leftarrow bear(X) \end{aligned}$$

Progol の出力した仮説 H_{animal}^{Progol}

$$\begin{aligned} safe(e) &\leftarrow \\ safe(X) &\leftarrow cat(X) \\ dangerous(X) &\leftarrow large(X) \\ dangerous(X) &\leftarrow bear(X) \end{aligned}$$

FOIL の出力した仮説 H_{animal}^{FOIL}

$$\begin{aligned} safe(X) &\leftarrow cat(X) \\ safe(X) &\leftarrow horse(X), medium(X) \\ dangerous(X) &\leftarrow large(X) \\ dangerous(X) &\leftarrow bear(X) \end{aligned}$$

このように、論理プログラムを用いることで仮説が一樣に表現され、それらの間で比較が可能となる。

3. 論理的基礎

EVLD は、従来個別に開発されていた知識発見システムを、下で述べる発見の論理 [Plotkin 71] の枠組みで捉える環境である。関係データベースの用語を用いれば、3 層スキーマ構造の中間層である概念スキーマ (論理構造) として発見の論理の枠組みを採用し、その下位層 (内部スキーマ, 物理構造) として個々の知識発見システムとその入力データを保持した関係データベースを位置付ける、ということである。発見の論理の上位層には、論理プログラムを利用した様々な知識処理システムが位置する。図 1 においてはアプリケーションやルールベースが上位層に相当する。以下では、オリジナルの発見の論理を若干変更して、論理プログラムの概念を用いて記述する。

仮説生成の対象となる事実を表す基礎アトム集合を $F = \{f_1, f_2, \dots, f_n\}$ とする。発見システムが仮説を生成する際に補助的に用いる基礎アトムは、連言の集合 $V = \{v_1, v_2, \dots, v_n\}$ という形で用いる。集合 V 中の各連言 v_i は F 中のアトム f_i と強い連関があると考えられる証拠 (evidence) を表す。発見システムが保持している論理プログラム B はすべての $f_1, f_2, \dots, f_n$ と $v_1, v_2, \dots, v_n$ に共通の背景知識となる。 F, V, B の間には $B \cup \{v_i\} \vdash f_i$ ($i = 1, 2, \dots, n$) が成立しているものとする。

【定義 1】組 (F, V, B) にして論理プログラム H が $B \cup H \cup \{v_i\} \vdash f_i$ ($i = 1, 2, \dots, n$) を満たすとき、 H は (F, V, B) に対する正しい仮説であるという。また、 H を仮説発見問題 $HFP(F, V, B)$ に対する解ともいう。 $HFP(F, V, B)$ の解全体の集合は有限とは限らない。 $HFP(F, V, B)$ の自明な解 $H_{F,V} = \{f_1 \leftarrow v_1, \dots, f_n \leftarrow v_n\}$ を初期仮説とよぶ。

【定義 2】論理プログラム H_1, H_2, B に対して、 $B \cup H_1 \vdash H_2$ が成立するとき、 H_1 は H_2 より B に関して相対的に一般的であるという。

発見の論理の枠組みにおける知識発見システムとは $HFP(F, V, B)$ に対する解で初期仮説 $H_{F,V}$ より B に関して相対的に一般的な仮説 H を (何らかの基準で選択しながら) 生成するシステムのことである。

知識発見システムを発見の論理の下位層に位置付けるために、EVLD は以下の機能を利用者に提供する。

- (1) データベースからタプル集合を得るための SQL 文と発見の論理における述語記号との対応を与える機能。
- (2) 発見の論理における事実を表す述語記号と証拠を表す述語記号を定義する機能。

また、個々の知識発見システムに対しては、これらの機能によって定義された述語記号および事実と証拠の指定によって、データベースから得られるタプル集合をそのシステムに応じた入力フォーマットに変換する変換系を用意している。さらに、そのシステムが論理プログラム以外の形で仮説を生成する場合には、その仮説を論理プログラムに変換するための変換系も用意している。

4. 属性指向と組指向

関係データベースからの知識発見システムには 2 種類ある。一つは、C4.5 のように入力データの関係を実体 (対象) の属性と属性値による記述とみなし、属性値によって対象の分類を行うものである。このようなシステムを属性指向知識発見システムと呼ぶことにする。一方、Progol や FOIL は入力データの関係を対象間の関連の具体例とみなし、関連を定義する一般的な規則を導出する。このようなシステムと組指向知識発見システムと呼ぶことにする。

4.1 発見の論理における属性指向の定式化

【定義 3】 図 3 に示す関係 R_D において, 属性 ID の値は各タブルを一意に識別する番号 (キー) であるとし, $A_0, A_1, \dots, A_n$ は任意の属性名, a_{ij} は A_j の属性値であるとする. 属性 A_0 の値によって ID の値を分類するとき A_0 を分類属性, 属性 $A_1, \dots, A_n$ を説明属性という.

ID	A_0	A_1	$\dots$	A_n
1	a_{10}	a_{11}	$\dots$	a_{1n}
2	a_{20}	a_{21}	$\dots$	a_{2n}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
m	a_{m0}	a_{m1}	$\dots$	a_{mn}

図 3 説明属性を持つ関係 R_D

関係 R_D の表現方法はいくつか考えられるが, 本研究では以下の 2 通りを定義する.

【定義 4】 関係 R_D において, ID 以外の各属性 A_j ($j = 1, 2, \dots, n$) に対して 2 引数の述語記号 p_j を用意する. A_0 が分類属性であるとき, ID の値が i であるタブルに対して

$$F_{R_D} = \{p_0(i, a_{i0}) \mid 1 \leq i \leq m\},$$

$$V_{R_D} = \{p_1(i, a_{i1}) \wedge \dots \wedge p_n(i, a_{in}) \mid 1 \leq i \leq m\}$$

と定義する. これを属性名に基づく変換という.

【定義 5】 関係 R_D において, ID 以外の各属性の属性値 a に対して 1 引数の述語記号 p_a を用意する. A_0 が分類属性であるとき, ID の値が i であるタブルに対して

$$F_{R_D} = \{p_{a_{i0}}(i) \mid 1 \leq i \leq m\},$$

$$V_{R_D} = \{p_{a_{i1}}(i) \wedge \dots \wedge p_{a_{in}}(i) \mid 1 \leq i \leq m\}$$

と定義する. これを属性値に基づく変換という.

〔例 1〕 ([Plotkin 71, Arimura 00]) 図 2 の関係 R_{animal} から, 危険な動物とそうではない動物に関する仮説を発見することを考える. 属性名による変換を用いると R_{animal} の最初のタブルから

$$f_1 = class(a, dangerous),$$

$$v_1 = size(a, s) \wedge clr(a, black) \wedge anml(a, bear)$$

が生成される. 同様に 2 番目以降のタブルからも $f_2, v_2, \dots, f_7, v_7$ が生成される. スペースの関係で, *small*, *medium*, *large* という定数記号をそれぞれ s, m, l と, また *color* と *animal* という述語記号をそれぞれ $clr, anml$ と略している.

属性指向知識発見システムは, $F_{animal} = \{f_1, f_2, \dots, f_7\}$, $V_{animal} = \{v_1, v_2, \dots, v_7\}$ として $HFP(F_{animal}, V_{animal}, \emptyset)$ の解で $H_{F_{animal}, V_{animal}}$ より一般的なものを生成する. 例えば, 次の 4 つの節からなる仮説はそのよ

うな解の一つである.

```
class(X, dangerous) ←
  size(X, Y), clr(X, black), anml(X, bear)
class(X, dangerous) ←
  size(X, large), clr(X, Y), anml(X, Z)
class(d, safe) ←
  size(d, small), clr(d, black), anml(d, cat)
class(e, safe) ←
  size(e, medium), clr(e, black),
  anml(e, horse)
```

なお, 属性値に基づく変換を用いると,

$$f_1 = dangerous(a),$$

$$v_1 = small(a) \wedge black(a) \wedge bear(a)$$

などが生成される.

ここで, 関係 R_{animal} 中の 1 つのタブルを必ずしも 1 つの基礎アトムで表現していないことに注意する. 一般に発見の論理における“事実”は, 通常の帰納論理プログラミングにおける“正例”とは一致しない.

4.2 決定木の表現

属性指向知識発見システムの出力の典型は決定木である. そこで決定木と論理プログラムの関係を明確にしておく.

まず, 関係 R_D の属性 A_0 に関して ID の値を分類する決定木 T の定式化を与える [田中 00]. 属性値集合を $D_j = \{a_{ij} \mid 1 \leq i \leq m\}$ ($j = 0, 1, \dots, n$) とする.

【定義 6】 関係 R_D の分類属性 A_0 によって ID の値を分類する決定木とは, 以下の条件を満たす木である.

- 葉以外の各ノードには $A_1, \dots, A_n$ のいずれかの属性名がラベル付けされている.
- 木から葉に至る任意の経路 $N_1, N_2, \dots, N_l, L$ に付けられたラベル $A_{j(N_1)}, \dots, A_{j(N_l)}$ は全て異なる. ここで $A_{j(N)}$ はノード N にラベル付けられた属性である.
- 葉以外のノード N に属性 $A_{j(N)}$ がラベル付けされていた場合, N からその子 $M_1, \dots, M_k$ への各エッジ $E_1, \dots, E_k$ には $D_{j(N)}$ の互いに素な部分集合 $S_{M_1}^N, \dots, S_{M_k}^N$ がラベル付けされている. ただし, $S_{M_1}^N \cup \dots \cup S_{M_k}^N = D_{j(N)}$ である.
- 葉 L には属性値集合 D_0 の部分集合 S_L がラベル付けされている.

【定義 7】 決定木 T が関係 R_D を説明しているとは, 関係 R_D 中の任意のタブル $(k, a_{k0}, a_{k1}, \dots, a_{kn})$ に対して, 木の根 N_1 から葉 L に至る経路 $N_1, N_2, \dots, N_l, L$ で $a_{kj(N_h)} \in S_{N_{h+1}}^{N_h}$ ($h = 1, 2, \dots, l$) かつ $a_{k0} \in S_L$ であるようなものがただ一つ存在することをいう.

【定義 8】 決定木 T の根 N_1 から葉 L に至る経路 $N_1, N_2, \dots, N_l, L$ に対して, 論理プログラム H_L を

$$\{p_0(X, a_0) \leftarrow p_{j(N_1)}(X, a_1), \dots, p_{j(N_l)}(X, a_l) \\ | a_0 \in S_L, a_h \in S_{N_{h+1}}, h = 1, 2, \dots, l\}$$

と定義する. さらに木 T のすべての葉が $L_1, \dots, L_n$ であるとき, $H_T = H_{L_1} \cup \dots \cup H_{L_n}$ とする. H_T を T の属性名に基づく表現という.

【定義 9】 決定木 T の根 N_1 から葉 L に至る経路 $N_1, N_2, \dots, N_l, L$ に対して, 論理プログラム H_L を

$$\{p_{a_0}(X) \leftarrow p_{a_1}(X), \dots, p_{a_l}(X) \\ | a_0 \in S_L, a_h \in S_{N_{h+1}}, h = 1, 2, \dots, l\}$$

と定義する. さらに木 T のすべての葉が $L_1, \dots, L_n$ であるとき, $H_T = H_{L_1} \cup \dots \cup H_{L_n}$ とする. H_T を T の属性値に基づく表現という.

命題 1 関係 R_D を説明する決定木 T の論理プログラム表現 H_T は, 属性名, 属性値どちらに基づく変換を用いても仮説発見問題 $HFP(F_{R_D}, V_{R_D}, \emptyset)$ の解であり, 初期仮説より一般的である.

命題 1 の証明は付録で与える.

〔例 2〕 図 2 の関係において $class$ に関して ID の値を分類する決定木 T_{animal} の一つに図 4 に示す木がある. 葉ではない各ノードに属性名 $animal$ と $size$ がラベル付けされている. $animal$ とラベル付けされたノードの子ノードへのエッジには $\{bear, dog, cat, horse\}$ の部分集合がラベル付けされ, $size$ とラベル付けされたノードの子ノードへのエッジには, $\{large, medium, small\}$ の部分集合がラベル付けされている. 葉には $\{dangerous, safe\}$ の部分集合がラベル付けされている. 決定木 T_{animal} の属性名に基づく論理プログラム表現 $H_{T_{animal}}$ は

$$\begin{aligned} class(X, dangerous) &\leftarrow animal(X, bear) \\ class(X, dangerous) &\leftarrow animal(X, dog), size(X, large) \\ class(X, dangerous) &\leftarrow animal(X, cat), size(X, large) \\ class(X, dangerous) &\leftarrow animal(X, horse), size(X, large) \\ class(X, safe) &\leftarrow animal(X, dog), size(X, small) \end{aligned}$$

などの 10 個の節からなる.

属性値に基づく表現を用いた場合, $H_{T_{animal}}$ の節は

$$\begin{aligned} dangerous(X) &\leftarrow bear(X) \\ dangerous(X) &\leftarrow dog(X), large(X) \end{aligned}$$

などになる.

図 4 属性 $class$ に関する決定木 T_{animal}

<i>father</i>	<i>child</i>
george	elizabeth
edward	diana
philip	anne
philip	charles
charles	william

(a) 父と子の関係 R_{father}

<i>name</i>	<i>gender</i>
george	male
edward	male
frances	female
elizabeth	female
philip	male
diana	female
charles	male

(b) 性別に関する関係 R_{gender}

<i>parent</i>	<i>child</i>
frances	diana
elizabeth	charles
elizabeth	anne
diana	william
george	elizabeth
edward	diana
philip	charles
philip	anne
charles	male

(c) 親と子の関係 R_{parent}

図 5 組指向知識発見システムのための関係

4.3 組指向知識発見システム

組指向知識発見システムへの入力は, 関係データベースから得られるタプルに適当な述語記号を与えればよい.

〔例 3〕 図 5 に示す 3 つの関係から, 父親に関する仮説を発見することを考える.

R_{father} に述語名 $father$, R_{gender} に述語名 $gender$, R_{parent} に述語名 $parent$ を与える. 関係 R_{father} , R_{gender} , R_{parent} の最初のタプルからそれぞれ基礎アトム

$$\begin{aligned} p_1 &= father(george, elizabeth), \\ q_1 &= parent(frances, diana), \\ r_1 &= gender(george, male) \end{aligned}$$

が生成される. 同様に 2 番目以降のタプルからも $p_2, \dots, p_5, q_2, \dots, q_7, r_2, \dots, r_9$ が生成される. 事実の集合は $F_{father} = \{p_1, \dots, p_5\}$, 背景知識は $B_{father} = \{q_1, \dots, q_7, r_1, \dots, r_9\}$ となる. 組指向知識発見システムは $HFP(F_{father}, \emptyset, B_{father})$ の解で $H_{F_{father}, \emptyset}$ より一般的なものを生成する. 例えば, 次の節からなる仮説はそのよう

な解の一つである。

$$\begin{aligned} \text{father}(X, Y) &\leftarrow \\ \text{parent}(X, Y), \text{gender}(X, \text{male}) \end{aligned}$$

なお、定義 4 や定義 5 に基づいて述語を定義することで、属性指向知識発見システムで取り扱うタプルを組指向知識発見システムで取り扱うタプルに変換することが可能であるから、組指向知識発見システムは属性指向知識発見システムとして利用することが可能である。実際、2 章で Progol や FOIL を用いて仮説を生成させる際にはこれらのシステムを属性指向知識発見システムとして利用している。

関係データベースから得られるタプルに適当な述語記号を与えることで生成された事実はすべて正の事実であり、負の事実は含まれていない。しかし、Progol や FOIL などのトップダウンに仮説を生成する知識発見システムは、探索範囲を限定するために負の事実を必要とする。そのため、これらの知識発見システムを選択した場合は関係データベースに含まれているデータおよび入力された情報を基に負の事実を生成しなければならない。EVLD ではこの問題に対して、負の事実を直接指定する方法と閉世界仮定を用いる方法の二つの解決方法を用意している。

【定義 10】 全事実 F^a とは事実のとりうるすべての組み合わせの集合を指す。未知の事実 F^u は正の事実 F^+ 、負の事実 F^- のどちらに含まれているか分からない事実を指す。 $F^a = F^+ \cup F^- \cup F^u$ である。

理論的には F^a は無限集合である。しかし、関係データベースにおいては、データの領域を関係データベースに含まれているデータに限ることにより F^a を有限集合と見なしてよい。また、発見システムによっては探索空間を制限するために述語の引数に型を導入することがあり、この場合には F^a はさらに小さくなる。

【定義 11】 閉世界仮定 (Closed World Assumption) とは F^u を空集合と仮定すること、すなわち $F^- = F^a - F^+$ と見なすことである。

F^u を空集合と仮定しない場合、関係データベースからの情報からのみでは F^- を決定することができない。この場合は利用者に F^- の全要素を具体的に入力してもらう必要がある。利用者は $F^a - F^+$ の要素から F^- の要素を個々に選択することになる。

【例 4】 3 の関係 R_{gender} から F^- を生成する。まず 2 引数の述語記号 gdr を用意する。第 1 引数は属性 name 、第 2 引数は属性 gender とそれぞれ対応している。述語 gdr の各引数になりうるデータの領域を $D_{\text{name}}, D_{\text{gender}}$ とすると

$$\begin{aligned} D_{\text{name}} &= \left\{ \begin{array}{l} \text{george, edward, frances,} \\ \text{elizabeth, philip, diana,} \\ \text{charles} \end{array} \right\}, \\ D_{\text{gender}} &= \{ \text{male, female} \} \end{aligned}$$

図 6 EVLD を使用したアプリケーション

となる。述語 gdr の全事実 $F_{R_{\text{gender}}}^a$ はこれらの組み合わせすべてからなるので、

$$F_{R_{\text{gender}}}^a = \left\{ \begin{array}{l} \text{gdr(george, m), gdr(george, f),} \\ \text{gdr(edward, m), gdr(edward, f),} \\ \text{gdr(frances, m), gdr(frances, f),} \\ \text{gdr(elizabeth, m), gdr(elizabeth, f),} \\ \text{gdr(philip, m), gdr(philip, f),} \\ \text{gdr(diana, m), gdr(diana, f),} \\ \text{gdr(charles, m), gdr(charles, f)} \end{array} \right\}$$

である。定数記号 male , female はそれぞれ m , f と略している。さらに述語 gdr に関する正の事実の集合は

$$F_{R_{\text{gender}}}^+ = \left\{ \begin{array}{l} \text{gdr(george, m), gdr(edward, m),} \\ \text{gdr(frances, f), gdr(elizabeth, f),} \\ \text{gdr(philip, m), gdr(diana, f),} \\ \text{gdr(charles, m)} \end{array} \right\}$$

である。もし閉世界仮定を用いるのであれば、負の事実の集合は $F_{R_{\text{gender}}}^- = F_{R_{\text{gender}}}^u - F_{R_{\text{gender}}}^+$ となる。

5. 上位層との接続

EVLD と上位層のアプリケーションとの具体的接続方法を図 6 に示すような Web サービス Animal を例に説明する。Animal は利用者が動物の種類、大きさ、色を入力することで、危険か安全かを判定するサービスである。図 2 に示すように関係データベースの中には動物の種類、大きさ、色、判定が含まれているが、全ての組み合わせは含まれていないと仮定する。Animal は EVLD に仮説を生成させ、仮説を用いてその動物が危険か安全かを演繹によって導出する。

EVLD が F , V , B を構成できるように、関係データベース名と 3 章で述べた EVLD の機能 (1), (2) を行うために必要な事項を生成する。Animal では危険か安全かを判定する仮説を要求するため、EVLD は F'_{animal} と V'_{animal} を生成する。ここで F'_{animal} , V'_{animal} は例 1 の F_{animal} , V_{animal} をそれぞれ属性値に基づいて表現し直したものである。

EVLD は各知識発見システムが生成した仮説を論理プログラムの形に変換したものを Animal に返す. 例えば, 次のような仮説

$$H = \left\{ \begin{array}{l} \text{dangerous}(X) \leftarrow \text{large}(X) \\ \text{dangerous}(X) \leftarrow \text{bear}(X) \\ \text{safe}(X) \leftarrow \text{cat}(X) \end{array} \right\}$$

が EVLD から返される. 次に Animal の利用者に動物の種類, 大きさ, 色を選択させることで単位節の集合を生成する. 例えば利用者が大きくて茶色い猫を選択した場合 (図 6(a)), 次のような単位節の集合

$$P = \left\{ \begin{array}{l} \text{large}(u\text{-selection}) \leftarrow \\ \text{brown}(u\text{-selection}) \leftarrow \\ \text{cat}(u\text{-selection}) \leftarrow \end{array} \right\}$$

を生成する. Animal は節集合 $C = H \cup P$ に対して, $\text{dangerous}(u\text{-selection})$ および $\text{safe}(u\text{-selection})$ のインスタンスで C の帰結となるアトム

$$R = \left\{ \begin{array}{l} \text{dangerous}(u\text{-selection}), \\ \text{safe}(u\text{-selection}) \end{array} \right\}$$

を生成する. R は, 大きい茶色い猫は安全でありかつ危険であることを示している.

なお, H から得られる帰結には, “危険である”, “危険ではない”, “安全である”, “安全ではない” の 4 種類の可能性があるが, この中で “危険ではない” と “安全である” は同じ意味ではないことに注意する. EVLD では現時点で 3 種類の発見システムで H を生成するため, 危険か安全かの判定も 3 種類出力される (図 6(b)).

6. 実現上の技法

6.1 基本構成

EVLD は Java 言語を用いて構築されている. 関係データベースへのアクセスは JDBC を利用している. JDBC とは Java 言語上において, 統一した方法で異なる複数のデータベースサーバーとやり取りすることを可能にする API である. JDBC を利用しているため, 関係データベースは EVLD と同一コンピュータ上に存在する必要がなくなっている. また知識発見システムを実行する部分も EVLD とは独立しており, EVLD とのデータのやりとりはソケット通信を使用して行う. そのため各知識発見システムもまた同一コンピュータ上に存在しなくてもよい. EVLD 中で扱うすべてのデータは XML によって表現されている. 個々の知識発見システムも Java で記述されている必要はなく, 外部プログラムとして実行されてもよい. 実際, C4.5 は WEKA プロジェクト [Witten 00] において Java 言語で記述された J4.8 を使用しているが, Progol および FOIL は外部プログラムとして実行している.

6.2 述語記号の定義と SQL 文との対応

EVLD の利用者はウィザード形式の入力インターフェイスを使用して情報を入力する. ウィザード形式のインターフェイスを用いると, 選択可能なパラメータを絞り込むことによって利用者の負担を軽減する効果があるだけでなく, 後述するようにパラメータ選択の履歴を用いた標準的な述語記号を決定することが可能となる.

利用者はまず属性指向, 組指向のいずれかを選択する. 次に利用者は関係データベースのどの関係のどの属性を用いて述語を定義するのかをメニュー選択形式で決定する.

属性指向を選択した場合, EVLD は定義 5 に基づいて述語を生成する. 組指向を選択した場合は利用者が任意の述語名を入力することで述語を生成する. その述語の定義に基づいて EVLD は SQL の SELECT 文を構成し, 関係データベースからタプルを取得する. 1 つの関係又はビュー R 中の $A_1, \dots, A_n$ の属性名を述語 p で表すときには $p(A_1, \dots, A_n)$ と表記される. EVLD は述語記号を p とする基礎アトムを生成するために SQL 質問文

SELECT $A_1, \dots, A_n$ FROM R

を構成する. 得られたタプルは述語記号を p とする基礎アトムに変換され, それが事実集合 F , 証拠集合 V や背景知識 B 中に用いられる.

なお, 関係名と属性の列 $RA_1A_2 \dots A_n$ はこの列自体が述語記号として利用可能である. この列を標準的な述語記号とよぶ. 標準的な述語記号はウィザードの操作の履歴から容易に構成することができ, 実際にルールベースに仮説を登録する際には標準的な述語記号も利用している. しかし標準的な述語記号は, 意味が理解しづらかったり, 長さが非常に長くなるなど, 利用者や知識発見システムにとって利用しづらいものになる可能性がある. そこで別名として述語記号 p を与えることを利用者に許している.

図 7 に入力インタフェースの実行中の様子を示す. 図は組指向を選択した場合の述語を生成する部分である. Table と書かれている下の選択枠にはデータベース内の関係の一覧が含まれており, 利用者はその中から関係を選択する. 関係を選択すると, その関係に含まれる属性の一覧が Field と書かれている下の枠に表示される. 利用者は属性の一覧から属性を選択し, Predicate Name と書かれている下の入力枠に任意の述語名を入力して Add ボタンを押すことで述語が生成される. Defined Literals と書かれている下の枠にはには生成した述語が表示される. なお, Defined Literals の枠に表示されている $\text{canreach}(\text{src}, \text{dst})$ は基礎アトムではなく, 標準的な述語記号 canreachsrcdst を括弧とコンマを用いて見やすくしたものである. 知識発見に必要な基礎アトムは入力した情報を基にデータベースから取得したタプルによって生成される.

図 7 EVLD の入力インターフェイス

6.3 知識発見システムとの連携

本稿では属性指向の定式化について 2 通り定義し、決定木の表現についても 2 通り定義したが、EVLD ではそれぞれ属性値に基づく変換と表現を用いている。これは今回利用できる知識発見システムのうち FOIL が、属性名に基づく変換をサポートしていないためである。

生成された基礎アトムの集合は事実と証拠、背景知識に分けられ、それぞれの知識発見システムに受け渡す。現時点では C4.5, Progol, FOIL を知識発見システムとして用意しているが、組指向を選択した場合に属性指向知識発見システムである C4.5 は利用できない。一方、属性指向を選択した場合は、前述したように組指向知識発見システムは属性指向知識発見システムとして利用できるため、C4.5, Progol, FOIL が利用可能になる。なお組指向知識発見システムにおいて、Progol, FOIL を利用する場合は負の事実を必要とするため 4.3 節で説明した方法で負の事実を生成する。そして EVLD はそれぞれの知識発見システムから仮説を得るが、C4.5 で得られた仮説は定義 9 に基づいて論理プログラムに変換しルールベースに蓄える。一方、Progol, FOIL で得られた仮説は論理プログラムであるのでそのままルールベースに蓄えられる。

7. まとめと今後の課題

7.1 EVLD の意義

本稿では Plotkin の発見の論理の枠組みを概念スキーマとして採用することにより、様々な知識発見システムから比較可能な形の仮説を得ることが可能なソフトウェア環境 EVLD について記述した。また EVLD で実際に知識発見システムに仮説を生成させ、生成された仮説が比較可能であることを示した。

Plotkin の発見の論理は 30 年前に考案された枠組みである。その論文 [Plotkin 71] 中には、図 2 に示した関係

データベースの属性指向表現が例として与えられている。また、例 3 と類似した組指向表現からの発見も扱われている。すなわち、発見の論理は、関係データベースという用語や概念こそ用いられていないが、関係データベースからの知識発見の原型ということができる。しかも、発見の論理は論理実証主義の形式化という科学哲学的考察から導かれている [有村 00]。したがって EVLD には、科学哲学と現代流の知識発見を述語論理によってつなぐ、という意義がある。

7.2 今後の展開

各知識発見システムが生成した仮説の比較には、定理証明システムを利用することが可能である。実際、2 つの仮説 H_1, H_2 について $H_1 \vdash H_2$ と $H_2 \vdash H_1$ を検証すればよい。

発見の論理の上位層に位置するアプリケーション例として本論文では Web サービス Animal を示した。EVLD を利用するアプリケーションの他の例としては、ページアンネットワーク構築ツール [本村 01] のようなアプリケーションも考えられる。

現状では属性指向発見において EVLD は述語の再帰的定義を含む仮説を生成しない。属性指向発見における仮説は定義 3, 4 から、分類属性を表す述語を頭部に、説明属性を表す述語を本体に持つ確定節からなるが、説明属性と分類属性に共通な属性を認めないため再帰的な定義が許されていない。FOIL や Progol を用いる場合には本来再帰的定義を含む仮説を生成する能力があるが、現状では C4.5 との比較のため、定義 4 という言語バイアスを設定して再帰的定義を含まない仮説を生成するようになっている。もし FOIL や Progol の本来の能力を用いるように設定を変更すれば、再帰的定義を許す仮説発見を行うことが可能である。

また、EVLD では関係データベースから仮説を生成することを前提としているため、帰納論理プログラミングにおける関数記号を含む論理プログラムの仮説を直接生成することは出来ない。ただし、関数記号については、関係データベースシステムが一階述語論理の“項”を文字列として登録することを許すのであればその扱いも可能である。また、平坦化 (flattening) を用いれば、項を関係表現することも可能となる。

EVLD 自体の発展として、EVLD を分散発見システムの基礎にするという方向がある。そのためには仮説の統合、データの分割やサンプリングの機構を導入する必要がある。各知識発見システムが生成する仮説が誤差を含まなければ、それらをすべてマージした論理プログラムはまた一つの仮説となるという性質が発見の論理に基づくシステムの特徴である。したがってこの仮説に汎化などの別の操作を加えればさらに別の仮説を得る。しかし、実際の知識発見システム、特に Apriori アルゴリズム [Agrawal 93] などでは、仮説 H に誤差が含まれているため、得ら

れた結果をそのままマージすることはできないという問題が生じる。また、EVL D では関係データベースがネットワーク上に存在してもよいと、大量のデータを引き出すとネットワークを占有してしまう。また、データの内容には非公開のものも含まれている可能性があるが、現在の EVLD ではデータがネットワーク上をそのまま通過してしまう。このような点を考慮に入れて、知識発見システムとのデータのやりとりを改良する必要がある。

◇ 参 考 文 献 ◇

- [Agrawal 93] R. Agrawal, T. Imielinski, and A. Swami : Mining Association Rule between Sets of Items in Large Database. *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 207–216 (1993).
- [有村 00] 有村博紀, 平田耕一, 山本章博 : 帰納論理プログラミングと証明補完. bit 別冊「発見科学とデータマイニング」, pp. 34–44, 共立出版 (2000).
- [Arimura 00] H. Arimura and A. Yamamoto : Inductive Logic Programming : From Logic of Discovery to Machine Learning. *IEICE Transactions on Information and Systems*, Vol. E83-D, No.1, pp. 10–18 (2000).
- [菊池 01] 菊池敏幸, 山本章博 : 様々な発見システムを比較可能にする環境の構築. 人工知能学会知識ベースシステム研究会資料, SIG-KBS-A101, pp. 49–55 (2001).
- [Lavrac 94] N. Lavrac and S. Džeroski : *Inductive Logic Programming : Techniques and Applications*. Ellis Horwood (1994).
- [本村 01] 本村陽一 : データベースと連携したベイジアンネット構築システム. 2001 年情報論的学習理論ワークショップ予稿集, pp. 223–226 (2001).
- [Muggleton 95] S. Muggleton : Inverse entailment and PRO-GOL. *New Gen. Comput.*, Vol. 13, pp. 245–286 (1995).
- [尾崎 99] 尾崎知伸, 古川康一 : 帰納論理プログラミングと関係データベースの SQL による統合の試み. 1999 年度人工知能学会 (第 13 回) 論文集, pp. 318–321 (1999).
- [Plotkin 71] G. D. Plotkin : A Further Note on Inductive Generalization. *Machine Intelligence 6*, pp. 101–124. Edinburgh University Press (1971).
- [Quinlan 93] J. R. Quinlan : *C4.5: Programs for Machine Learning*. Morgan Kaufmann (1993).
- [Quinlan 90] J. R. Quinlan : Learning Logical Definitions from Relations. *Machine Learning 5*, pp. 239–266 (1990).
- [田中 00] 田中一志, 山本章博 : 飽和法と底法の関係データベースへ適用. 人工知能学会人工知能基礎論研究会資料, SIG-FAI-9904, pp. 97–102 (2000).
- [Ullman 88] J. D. Ullman : *Database and Knowledge-base Systems Vol.1*. Computer Science Press (1988).
- [Witten 00] I. H. Witten and E. Frank : *Data Mining*. Morgan Kaufmann (2000).

〔担当委員：沼尾正行〕

2001 年 8 月 8 日 受理

◇ 付 録 ◇

A. 命題 1 の証明

属性名に基づく表現を用いて証明する。属性値に基づく表現を用いても証明はほぼ同じである。

決定木 T が関係 R_D を説明しているとする。仮説発見問題 $HFP(F_{R_D}, V_{R_D}, \emptyset)$ の初期仮説 H_0 中の節で、関係 R_D 中のタ

ブル $t = (k, a_{k0}, a_{k1}, \dots, a_{kn})$ に対応するものは

$$C_t = p_0(k, a_{k0}) \leftarrow p_1(k, a_{k1}), \dots, p_n(k, a_{kn})$$

である。決定木 T の経路 $\sigma = N_1, N_2, \dots, N_l, L$ で、 $a_{kj(N_h)} \in S_{N_{h+1}}^{N_h}$ ($h = 1, 2, \dots, l$) であるものを取れば、定義から H_L には

$$C_\sigma = p_0(X, a_{k0}) \leftarrow p_{j(N_1)}(X, a_{kj(N_1)}), \dots, p_{j(N_l)}(X, a_{kj(N_l)})$$

という節が存在し $C_\sigma \models C_t$ である。

—— 著 者 紹 介 ——

菊池 敏幸(学生会員)

2001 年 3 月 北海道大学工学部電子工学科卒業。現在、北海道大学大学院工学研究科電子情報工学専攻修士課程在学中。主としてデータベースと知識発見システムの連携に関する研究に従事。情報処理学会学生会員。

山本 章博(正会員)

1985 年 京都大学理学部卒業。1987 年 九州大学大学院総合理工学研究科修士課程修了。1990 年 同博士後期課程終了。同年 北海道大学工学部講師。1994 年 同助教授。1997 年より同大学大学院工学研究科助教授。理学博士。主として論理プログラミングの基礎理論とその帰納推論への応用に関する研究に従事。1996 年 6 月より 1 年間、文部省在外研究員としてドイツ連邦 Darmstadt 工科大学に滞在。1998 年 10 月より 2001 年 9 月まで科学技術振興事業団 さきがけ研究 21「情報と知」領域研究者。情報処理学会、日本ソフトウェア学会、EATCS 各会員。