論　文

Technical Papers

デフォルト規則を含む拡張論理プログラムの学習

Learning Default Rules in Extended Logic Programs

井上　克巳[*1]　工藤　嘉晃[*2]　羽根田　博正[*1]
Katsumi Inoue　Yoshimitsu Kudoh　Hiromasa Haneda

＊1　神戸大学工学部電気電子工学科
Department of Electrical and Electronics Engineering, Kobe University, Kobe 657-8501, Japan.
＊2　北海道大学大学院工学研究科電子情報工学専攻
Division of Elecronics and Information Engineering, Hokkaido University, Sapporo 060-8628, Japan.

1997年7月25日　受理

Keywords: machine learning, inductive logic programming, nonmonotonic reasoning.

Summary

This paper presents a method to learn nonmonotonic rules with exceptions from positive/negative examples and background knowledge in Inductive Logic Programming. We adopt extended logic programs as the form of programs to be learned, in which two kinds of negation, negation as failure and classical negation, are effectively used in the presence of incomplete information. We implemented the learning system LELP. LELP first generates candidate rules from positive examples and background knowledge using ordinary ILP techniques. Default rules with exceptions are then generated as rules with negation as failure using the OWS algorithm. Exceptions are identified from negative examples and are then generalized to default cancellation rules. In LELP, default rules can be produced for both positive and negative consequents. Then, if some instances are possibly classified as both positive and negative, nondeterministic rules are constructed to resolve the inconsistency by getting multiple answer sets. Moreover, hierarchical defaults can also be learned by recursively calling the exception identification algorithm.

1. はじめに

帰納的論理プログラミング (inductive logic programming; ILP) [Lavrač 94, Muggleton 92a] は，論理プログラミングの枠組において関係記述の帰納学習理論と実用的なアルゴリズムを研究する分野である．これまでのILP研究では，学習する論理プログラムの形式として，主に確定ホーンプログラムまたは節形式プログラムを学習することが考えられていた．しかしながら，AIにおける知識表現研究の成果によれば，これらの単調なクラスの論理に基づくプログラムでは，常識則や分類学的階層のような知識を表現するには不十分であることがわかっている．それゆえ，デフォルト規則や分類学的階層知識を学習するためには，非単調推論を扱える帰納推論の枠組が必要である．

一方，論理プログラミングと非単調推論の最近の理論において，**失敗による否定** (negation as failure; **NAF**) を含む論理プログラムが知識表現用言語として利用できることが認識されている [Baral 94]．一般論理プログラム (normal logic program; **NLP**) は，規則のボディにNAFが自由に現れることを許すプログラムである．NLPはデフォルト規則や例外を含む規則を表現するだけでなく，多くの場合に確定プログラムよりも簡潔でわかりやすいプログラムを記述できる [Bergadano 95]．NLPのクラスの中でも，層状プログラム (stratified program) の学習は [Bain 92b, Bergadano 95, Srinivasan 92] で考えられている．

NLPの学習はより優れた学習方法への重要なステップではあるが，不完全な情報を直接扱えないという制限がNLPにはある [Gelfond 91]．NLPは自動的にすべての述語に**閉世界仮説 (CWA)** を適用する．帰納的概念学習において，CWAの自動的な適用は，特に正例と負例を両方用いる場合には適当ではない．正例は目

標概念の例を表すが，他の例はCWAにより概念の例でないとされてしまうため，負例の役割が明確ではなく，CWAはすべての対象の完全な分類を与えてしまう．これはDe RaedtとBruynooghe [De Raedt 90]によって指摘されたパラドックスを引き起こす．すなわち，もしすべてがわかっているならば，われわれはもはや何も学習する必要はないということである．実世界においては，正例であるか負例であるかがわからない例はいくらでも存在するが，そのような不完全な情報をNLPによって表すことができない．

NLPの上述の問題を克服するために，本稿では**拡張論理プログラム** (extended logic program; **ELP**) の形式を用いて，不完全な情報を扱える新しい学習方法を提案する．GelfondとLifschitz [Gelfond 91]により紹介されたELPは，**論理否定** (classical negation) を含めることでNLPのクラスを拡張している．ELPの意味論は，安定モデル意味論を三値に拡張した**解集合意味論** (answer set semantics) により与えられる．プログラムの各解集合において，ある例について正負いずれのリテラルも含まれないならば，その例の真偽値は *unknown* であると解釈されるため，CWAの自動的適用が防がれて負例にはならない．

本研究においては，正例，負例，背景知識を入力として，ELPの形式で，例外を含むデフォルト規則を学習するILPシステム **LELP** (learning extended logic program) を開発した．LELPは**NML** [井上 96]を整理拡張したものである．LELPでは，はじめに従来のILP手法を用いて，正例 (または負例) と背景知識から一般規則を生成する．例外は，負例 (または正例) の中で，求めた一般規則と背景知識から導出される個体として認識される．NAFを含むデフォルト規則は，**開世界特殊化** (open world specialization; **OWS**) を用いて一般規則を特殊化することで生成される．OWSはBainとMuggletonのCWS [Bain 92b]と密接な関連があるが，三値の意味論においてより望ましい動作をする．さらに，デフォルト打ち消し規則が，例外をカバーするように生成される．

現実的には，一般的なデフォルト規則が，正の結論を持つべきか負の結論を持つべきかを決定することは容易ではない．どちらが一般的であるかを判断することが困難な場合は，非決定的規則または並行するデフォルト規則を生成する．ここで非決定的規則を導入することにより，層状プログラムではないプログラムが得られる．さらに，OWSとデフォルト打ち消し規則の生成の部分を再帰的に処理することで，階層的なデフォルト規則を生成することも可能である．

本論文は [Inoue 97] を一部詳細化しており，学習アルゴリズムの改訂とその正当性の証明を含んでいる．以下の章では，**2章**で本研究のシステムLELPについて説明し，**3章**で複雑な例外構造を持つ規則を生成できるようにシステムを拡張し，**4章**で関連研究について述べ，**5章**でまとめる．

2. デフォルト規則の学習

本章では，例外を含むデフォルト規則を学習するためのアイデアを明確にするために，一般的な規則とそれに対する例外があるだけの簡単なモデルを考え，LELPの動きを順に説明する．例外の例外を含むような複雑なモデルは**次章**で考察する．

LELP全体のアルゴリズムの概略は次の通りである．

アルゴリズム **2・1** $LELP1(E^+, E^-, BG, H)$

入力: 正例 E^+，負例 E^-，背景知識 BG

出力: 規則 $H := T_1 \cup T_2 \cup T_3$

(1) E^+ と E^- の情報を基に，正負いずれの規則を一般則として求めるかを決定する．
if 負の規則を学習 **then** E^+ と E^- を交換．

(2) $GenRules(E^+, BG, T_0)$.
BG の下で E^+ をカバーする規則 T_0 を求める．

(3) $OWS(T_0, E^+, E^-, BG, AB, T_1)$.
OWSを用いて T_0 を特殊化し，E^- と BG に対して無矛盾となるようにデフォルト規則 T_1 を求め，同時に例外集合 AB も求める．

(4) $Counter(E^-, AB, T_1, T_2)$.
AB の下で E^- をカバーする規則 T_2 を，T_1 と対応させた形で求める．

(5) $Cancel(AB, BG, T_3)$.
BG の下で AB を一般化し，デフォルト打ち消し規則 T_3 を求める．

2・1 拡張論理プログラム

LELPでは，学習する論理プログラムの形式として，**拡張論理プログラム** (ELP) を採用する．ELPは次の形式の規則の集合として定義される:

$$L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$$

ここで，$L_i\ (0 \leq i \leq n; n \geq m)$ はリテラルであり，L_0 をヘッド，$\leftarrow$ の右側を**ボディ**という．ELPではNAFを表す *not* と論理否定の $\neg$ の2種類の否定が現れる．直観的に上の規則は，もし $L_1, \ldots, L_m$ が信じられており，$L_{m+1}, \ldots, L_n$ が信じられていないならば，L_0 は信じられていることを表す．ELPの意味論は，基礎リテラルの集合である**解集合**によって定義される．ELP

のクラスは，**デフォルト論理** (default logic) [Reiter 80] の部分クラスと同一視できる [Gelfond 91]．すなわち，ELP における上記の形式の各規則を，デフォルト

$$\frac{L_1 \wedge \ldots \wedge L_m : \overline{L_{m+1}}, \ldots, \overline{L_n}}{L_0}$$

と同一視する．ここで，$\overline{L}$ はリテラル L の補リテラルを表す．このとき，各解集合はデフォルト理論の各拡張に含まれるリテラル集合に一致する．もしリテラル L が ELP P のすべての解集合に含まれるならば，L は P によって**帰結される**といい，$P \vdash L$ と表す．

LELP の入力として与える**正例**は正リテラルで表し，**負例**は負リテラルで表す．**背景知識**は，ELP 形式の規則の集合である．ヘッドが正リテラルである規則を**正の規則**と呼び，ヘッドが負リテラルである規則を**負の規則**と呼ぶ．以下，プログラム中においては，論理否定 $\neg$ を $-$ で，NAF の *not* を \+ で，含意記号 $\leftarrow$ を :- で表す．

概念学習の完全性と無矛盾性 [Lavrač 94] は，三値の意味論において次のように定式化する．BG を背景知識とし，E を正例と負例の集合，R を仮説とする．すべての $e \in E$ に対して，$BG \cup R \vdash e$ (R は e を**カバーする**) ならば，R は BG と E に関して**完全**である．もし任意の $e \in E$ に対して，$BG \cup R \nvdash \bar{e}$ (R は $\bar{e}$ を**カバーしない**) ならば，R は BG と E に関して**無矛盾**である．ここで，正例と負例には同じ優先度が与えられていることに注意せよ．すなわち，背景知識と例に対して無矛盾であるように，正例と負例の両方が学習された規則によってカバーされなければならない．

LELP によって正の規則と負の規則が生成されるが，ここでは目標概念に対するデフォルト規則は，正あるいは負のどちらかに決める場合を考える．この決定においては，例えば，正例および負例の個体全体に対する比率などを考慮することが考えられる．

2・2 一般規則の生成

アルゴリズム 2·1 においては $\underline{GenRules(E, BG, T)}$ で表される一般規則の生成が必要となる．これは，一般規則 T が次の条件を満たすような極小の規則集合として生成され，従来の ILP 手法を用いて実現される．

(1) T は BG と E に関して完全かつ無矛盾である．

(2) どんな例 $e \in E$ に対しても，T 中に e と単一化可能なリテラルをヘッドに持つ規則が存在する．

ここで第 2 の条件は，E を直接カバーするような概念の定義が T として学習されることを意味する．また第 1 の条件により，ある例が一般化されない場合には T にそのまま加えられることになる．特別な場合として，$T = E$ であれば条件を満たすため，*GenRules* の出力の存在は保証される．

ここでは，LELP の実現において具体的に用いたアルゴリズムを例として示す．

〔例 **2・1**〕(***GenRules*(*E*, *BG*, *T*)** の例)

入力: 正例 (または負例) E，背景知識 BG

出力: 一般規則 T

(1) 背景知識 BG から帰結されるすべての基礎リテラル集合 M を生成する．

(2) E と M から RLGG を用いて規則の候補 R を生成する．

(3) リテラルの連鎖を用いて R に含まれる余分なリテラルを除去し R' を得る．

(4) R' の BG による unfolding により T を得る．

上のアルゴリズム例を用いる場合は，帰結される基礎リテラル集合を求めるため，BG に関数記号が現れないという制限が必要となる．RLGG は GOLEM [Muggleton 92b] におけるボトムアップ手法に従って計算できるが，RLGG で求めた規則のボディには多くの冗長なリテラルが含まれるので，リテラルの連鎖を用いてそれらを除去する．これは規則のリテラルの引数の連鎖を基に，有用なリテラルを抜き出す操作であり，Helft [Helft 89] により帰納推論に導入されている．

〔例 **2・2**〕(***GenRules*(E^+, *BG*, T_0)** の動作例)

E^+ = {flies(1),flies(2),flies(3),
flies(4)},

BG = {bird(1),bird(2),bird(3),bird(4),
(bird(X):- pen(X)),bird(c),
pen(a),pen(b)}.

BG から帰結される基礎リテラル集合を生成する．

M = {bird(1),bird(2),bird(3),bird(4),
bird(a),bird(b),bird(c),
pen(a),pen(b)}.

e_1 =flies(3), e_2 =flies(4) とするとき，e_1, e_2 を基にした RLGG により，

```
flies(X) :- pen(A),pen(B),bird(A),
            bird(B), bird(C),bird(D),···,
            bird(X),···.
```

が得られ，この規則にリテラルの連鎖を適用すると以下の規則を生成する．

```
flies(X) :- bird(X).
```

unfolding [Tamaki 84] は，生成した規則のボディを短縮するために因子化とともに用いる．いま，P をプログラム，C を規則とする．ここで，C が $(H$:- $\beta, A, \gamma)$ の形であり，A は NAF でないリテラル，β, γ は (空かもしれない) リテラルの列であると

する．このとき，P による C の A に関する **unfolding** は，P を以下のプログラムに置換する．

$$(P\backslash\{C\}) \cup \{(H :- \beta, bd(D), \gamma)\theta \mid D \in P \text{ かつ } A \text{ が mgu } \theta \text{ で } hd(D) \text{ と単一化可能}\}$$

ここで，規則 X に対して，$hd(X)$ は X のヘッドを表し，$bd(X)$ は X のボディを表すものとする．例えば，C を `(ab1(X) :- bird(X),pen(X))` とし，P が `(bird(X) :- pen(X))` を含むとき，P による C の `bird(X)` に関する unfolding は，C を `(ab1(X) :- pen(X))` で置き換える．このとき 2 回出現する `pen(X)` は因子化により 1 つにマージされる．

2・3 デフォルト規則の生成

例外を含むデフォルト規則を生成するために，LELP においては Bain と Muggleton による閉世界特殊化 (CWS) [Bain 92b] を改良した**開世界特殊化** (OWS) を用いる．CWS と同様に，OWS により生成される規則は NAF を含むデフォルト規則の形式となる．OWS では閉世界仮説を用いるのではなく，例外は背景知識と一般規則から証明されるリテラルで，かつその補リテラルが負例に含まれることを必要とする．LELP ではこの証明に Prolog によるトップダウン証明を用いたが，解代入が得られる定理証明器であれば何を用いてもよい．OWS のアルゴリズムは次の通りである．

アルゴリズム 2・2

$\underline{OWS(T, E^+, E^-, BG, AB, T')}$

入力: 規則 T，正例 E^+，負例 E^-，背景知識 BG

出力: デフォルト規則 T'，例外 AB

$T' := T$; $AB := \emptyset$;

T 中の変数を含む各規則 $C_i = (H :- B)$ に対して，

$Exc := \{ L\theta \mid \overline{L} \in E^-,\ BG \cup T \models L\theta,$
$BG \cup (T \setminus \{C_i\}) \not\models L\theta \}$;

if $Exc \neq \emptyset$ **then**

if B が NAF リテラル $\backslash+N$ を含む **then**

$AB := AB \cup \{ N\theta \mid L\theta \in Exc \}$

else $N := \mathbf{ab}_i(V_1, \ldots, V_n)$,

ここで $\{V_1, \ldots, V_n\}$ は H 中の全変数であり $\mathbf{ab}_i$ は BG 中には出現しない新しい述語，

$T' := (T' \setminus \{C_i\}) \cup \{ (H :- B, \backslash+N) \}$,

$AB := AB \cup \{ N\theta \mid L\theta \in Exc \}$.

上のアルゴリズムにおいて，各 $\overline{L} \in E^-$ に対して，$BG \cup T$ から帰結されるが $BG \cup (T \setminus \{C_i\})$ からは帰結されないような $L\theta$ が Exc として集められる．これは，C_i が $BG \cup T$ から $L\theta$ を導くために必要であるという条件になっている．このような C_i がアルゴリズム 2·1 のステップ (2) において生成される場合，C_i のヘッド H と $\overline{L} \in E^-$ との間で導出可能である．なお，非単調な効果により，カバーされていた正例がカバーされない場合には，カバーされなくなった正例を結果のプログラム T' に挿入してやればよい．

〔**例 2・3**〕(**例 2·2 の続き**)

背景知識 BG，正例の集合 E^+ は例 2·2 と同じ．

$T = \{$`(flies(X) :- bird(X))`$\}$,

$E^- = \{$`-flies(a), -flies(b), -flies(c)`$\}$.

ここで，$C_1 =$ `(flies(X) :- bird(X))` とすると，C_1 に対して，$Exc =$ `{flies(a), flies(b), flies(c)}` であり，N を `ab1(X)` と置くことにより，

$T' = \{$`(flies(X) :- bird(X), \+ab1(X))`$\}$,

$AB = \{$`ab1(a), ab1(b), ab1(c)`$\}$.

2・4 反例を導く規則の生成

アルゴリズム 2·1 のステップ (4) において，**反例を導く規則**を求める必要がある．例えば一般規則が正の規則である場合は，負例の集合と OWS で求めたデフォルト規則の集合および例外の集合から，負例を導く規則を次のアルゴリズムを用いて求める．

アルゴリズム 2・3 $\underline{Counter(E, AB, D, T)}$

入力: 例 E, デフォルト規則 D，例外 AB

出力: 規則 T

(1) $R := \{ (\overline{H} :- N) \mid N\theta \in AB$ かつ $(H :- B, \backslash+N) \in D \}$.

(2) $T := R \cup \{ e \in E \mid AB \cup R$ によってはカバーされない e の例が存在する $\}$.

例 2·3 では，負例を導く規則は以下のように求まる．

`-flies(Y) :- ab1(Y).`

なお，アルゴリズム 2·3 において，AB が空なら R も空となり，一般化は行われず $T = E$ が出力される．これとは別の方法として，「T が D によってカバーされている正例の導出を妨げない」という制約の下で，負例 E をカバーするように $GenRules(E, AB, T)$ を用いて負例を一般化してしまう方法も考えられる．しかしながら，正例と負例がともに一般化規則によってカバーされるべきときには，**3 章**における並行するデフォルト規則の学習によって扱うこととする．

2・5 例外に関する規則の生成

OWS では例外の集合が基礎アトムの集合 AB として出力される．しかし，例外が何らかの規則性を持っている場合には，一般化を行うことで，**例外に関する規則**を生成する．これらの規則は**デフォルト打ち消し規則**として機能する．

OWS を適用した段階で，例外は述語記号 $\mathbf{ab}_i$ を用

いて表現されている．この述語記号 $\mathtt{ab}_i$ を持つリテラルをヘッドに含む規則が，例外に関する規則であり，ここでも *GenRules* を用いることができる．ところで，例外が特殊な存在であると考えると，例外に関する規則が一般的であり過ぎないようにすることが必要であり，この場合は，与えられた AB 以外のアトムを導かないことが制約となる．生成した規則が一般的であり過ぎないかどうかの検査は，例えば帰結される基礎リテラルを計算することで行うことができる．

アルゴリズム 2・4 *Cancel*(AB, BG, T)

入力: 例外 AB，背景知識 BG

出力: 例外に関する規則 T

(1) *GenRules*(AB, BG, T),
ただし，$BG \cup T$ から帰結される $\mathtt{ab}_i$ 述語を持つすべての基礎リテラルの集合が AB の基礎例の集合と一致するように T を生成する．

アルゴリズム 2·4 において，AB に対して一般化がまったく行われない場合，T は AB と一致し条件を満たす．よって，*Cancel* の出力の存在は保証される．

〔例 2・4〕（例 2·3の続き）

背景知識 BG は例 2·2と同じ，

例外の集合: $AB = \{\mathtt{ab1(a)},\ \mathtt{ab1(b)},\ \mathtt{ab1(c)}\}$.

いま，*GenRules* の出力 T が

$$T_a = \{\mathtt{(ab1(X)\ :-\ bird(X))}\}$$

であるとすると，$BG \cup T_a$ からは，AB 以外にアトム $\mathtt{ab1(1)}$, $\mathtt{ab1(2)}$, $\mathtt{ab1(3)}$, $\mathtt{ab1(4)}$ も帰結されるため，アルゴリズム 2·4 の出力とはならない．一方，

$$T_b = \{\mathtt{(ab1(X)\ :-\ pen(X))},\ \mathtt{ab1(c)}\}$$

を考えると，アルゴリズムの条件を満たす．よって，$T = T_b$ は *Cancel*(AB, BG, T) の出力となる．

2・6 例　　題

LELP のプログラムは Prolog 上で動作し，その仕様は `lelp(例, 背景知識, 結果)` である．例では，+ の付いたものを正例，- の付いたものを負例とする．次の例においては，正の規則を生成するように設定した．

〔例 2・5〕（鳥の例の総括: 図 1 参照）

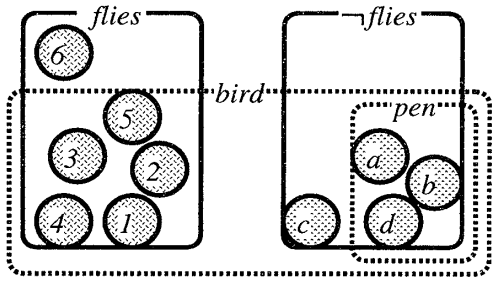

図 1　鳥の例

```
| ?- lelp([+flies(1),+flies(2),+flies(3),
      +flies(4),+flies(5),+flies(6),
      -flies(a),-flies(b),-flies(c),
      -flies(d)],
     [bird(1),bird(2),bird(3),bird(4),bird(5),
      bird(c),(bird(X) :- pen(X)),
      pen(a),pen(b),pen(d)],Rules).

Running Time: 280 (msec)
Rules =
[(flies(A):-bird(A),\+ab1(A)),flies(6),
 (-flies(B):-ab1(B)),
 (ab1(C):-pen(C)),ab1(c)] ?
```

2・7 LELP の正当性

［定理 2・1］ 背景知識 BG と正負例 $E = E^+ \cup E^-$ が与えられ，$BG \cup E$ は無矛盾であると仮定する．H を $LELP1(E^+, E^-, BG, H)$ による出力であるとすると，H は BG と E に関して完全かつ無矛盾である．

《証明》 アルゴリズム 2·1 に沿って証明を行う．ここでは，正の規則を生成する場合を考えるが，負の規則を生成する場合も同様にして証明できる．

まず，ステップ (2) の $GenRules(E^+, BG, T_0)$ では，*GenRules* の満たすべき条件より，T_0 は BG と E^+ に関して完全かつ無矛盾であることから，

$$\forall e \in E^+ \,[(BG \cup T_0 \vdash e) \wedge (BG \cup T_0 \nvdash \bar{e})].$$

次に，ステップ (3) の $OWS(T_0, E^+, E^-, AB, T_1)$ では以下の場合分けを考える．

$\underline{T_0 = E^+}$: *GenRules*(E^+, BG, T_0) が新しい規則を生成していないので，$T_1 = T_0$ かつ $AB = \emptyset$. よって，$\forall e \in E^+ (BG \cup AB \cup T_1 \vdash e)$ かつ $\forall e \in E^+ (BG \cup AB \cup T_1 \nvdash \bar{e})$. また，$BG \cup E$ の無矛盾性の仮定により，$\forall e \in E^- (BG \cup AB \cup T_1 \nvdash \bar{e})$.

$\underline{T_0 \neq E^+}$: ここでは T_0 において，ヘッド H が E^- からのあるリテラルとの間で導出可能であるような規則が $C = (H\ \mathtt{:-}\ B)$ 1つしか存在しない場合を考える．もしそのような規則が複数ある場合については以下の証明を拡張すればよい．このとき，次の場合分けを考える．

(1) $\bar{L} \in E^-$ かつ $BG \cup T_0 \vdash \exists L$ であるようなリテラル L が存在しない場合，$T_1 = T_0$ かつ $AB = \emptyset$ が成立する．このとき，上と同様にして，$\forall e \in E^+ (BG \cup AB \cup T_1 \vdash e)$, $\forall e \in E^+ (BG \cup AB \cup T_1 \nvdash \bar{e})$, および $\forall e \in E^- (BG \cup AB \cup T_1 \nvdash \bar{e})$ が成立する．

(2) $\bar{L} \in E^-$ かつ $BG \cup T_0 \vdash \exists L$ であるよう

なリテラル L が存在する場合，θ を L の証明における解代入とする．このとき，T_1 は規則 $(H$:- $B, \backslash+ N)$ $(N = \mathtt{ab}_i(V_1, \ldots, V_n))$ を含み，AB は $N\theta$ を含む．このとき，$\forall e \in E^+ (BG \cup AB \cup T_1 \vdash e)$ および $\forall e \in E^+ (BG \cup AB \cup T_1 \nvdash \bar{e})$ がやはり成立するとしてよいが，さらに $BG \cup AB \cup T_1 \nvdash L\theta$ も成り立つ．AB はこのような $N\theta$ をすべて含むので，$\forall e \in E^- (BG \cup AB \cup T_1 \nvdash \bar{e})$.

よって，上のいずれの場合においても，以下が成り立つ．

$$\forall e \in E^+ (BG \cup AB \cup T_1 \vdash e),$$

$$\forall e \in E^+ (BG \cup AB \cup T_1 \nvdash \bar{e}),$$

$$\forall e \in E^- (BG \cup AB \cup T_1 \nvdash \bar{e}).$$

次に，ステップ (4) の $Counter(E^-, AB, T_1, T_2)$ について考える．T_2 の構成方法により，

$$\forall e \in E^- (AB \cup T_2 \vdash e).$$

また，T_2 のすべての一般規則は，$(\overline{H}$:- $N)$ (ただし，$(H$:- $B, \backslash+ N) \in T_1$ かつ $N\theta \in AB$) という形式をしているため，$\overline{H}\theta$ が帰結されるときは常に $N\theta \in AB$ である．したがって，T_2 を $BG \cup AB \cup T_1$ に加えても $E = E^+ \cup E^-$ のどのリテラルに関する帰結可能性も変化させることはない．ゆえに，

$$\forall e \in E (BG \cup AB \cup T_1 \cup T_2 \vdash e),$$

$$\forall e \in E (BG \cup AB \cup T_1 \cup T_2 \nvdash \bar{e}).$$

次に，ステップ (5) の $Cancel(AB, BG, T_3)$ においては，T_3 によりカバーされる基礎 $\mathtt{ab}_i$ リテラルの集合が AB の基礎例の集合と一致するという制約の下で，$GenRules(AB, BG, T_3)$ により T_3 を生成する．よってこの制約により，

$$\forall ab_i \in AB\ (BG \cup T_3 \vdash ab_i).$$

さらに，$BG \cup T_3$ は AB の要素の基礎例ではないような新しい $\mathtt{ab}_i$ リテラルを帰結することはない．

最後に，$H = T_1 \cup T_2 \cup T_3$ と置くと，上の結果より，

$$\forall e \in E\,[(BG \cup H \vdash e) \wedge (BG \cup H \nvdash \bar{e})].$$

したがって，H は BG と E に関して完全かつ無矛盾である．□

3. 拡　　張

本章ではより複雑な規則を生成できるようにLELPを拡張する．

3・1 非決定的規則

前章の $LELP1$ アルゴリズム 2·1 のステップ (1) においては，一般規則が正であるか負であるかを判断する必要があった．この決定の方法として，正例および負例の個体全体に対する比率を考慮することも考えられるが，一般にはこうした方法ではうまくいかない．例えば，正例の数が負例の数に近い場合や，正負例が疎に与えられているために個体全体に対する割合が小さい場合には，正負どちらが一般的であるかの判定をくだすのは困難である．この問題に対して，ここでは，並行するデフォルト規則と非決定的規則の2つの解決策を考える．**並行するデフォルト規則**は，例外が正と負の両方の規則に存在するときに生成される．例えば，ほ乳類は通常飛ばないが例外としてコウモリは飛び，鳥類は通常飛ぶが例外としてペンギンは飛ばない．すなわち，飛ぶことに関する例外と飛ばないことに関する例外が並行して存在している．**非決定的規則**は，ある例が2つの規則により正例にも負例にも分類されて矛盾が起きるときに生成される．非決定的規則生成のアルゴリズムは，後述の階層的デフォルト規則と合わせて，アルゴリズム 3·2に示すが，以下で例を用いて説明する．

〔例 **3・1**〕(非決定的規則の生成)

```
| ?- lelp([+flies(1),+flies(2),+flies(3),
       +flies(4),-flies(5),-flies(6),
       -flies(7),-flies(8)],
      [bird(1),bird(2),bird(3),bird(4),
       bird(5),bird(6),bird(7),bird(8)],Rules).
```

この例では正例と負例の割合が一致しており，正の一般規則と負の一般規則の両方を並行するデフォルト規則として生成することにする．正負それぞれに対して，$LELP1$ を並行に使うと次のようになる:

```
(flies(A):-bird(A),\+ab1(A)),
ab1(5),ab1(6),ab1(7),ab1(8),
(-flies(B):-bird(B),\+ab2(B)),
ab2(1),ab2(2),ab2(3),ab2(4)
```

ここには，$Counter$ により生成される規則である `(-flies(A):-ab1(A))` と `(flies(B):-ab2(B))` は含めていない．この理由は，正負両概念を学習しようとする場合にはこれらの反対概念を導く規則が不要になるからである．さて，上の2つの正負の規則は一見正しそうであるが，両規則のボディが $\mathtt{ab}_i$ 述語を除くと `bird(`X`)` で同じであるため，背景知識に `bird(9)` を追加すると，`ab1(9)` と `ab2(9)` がともに証明されず，矛盾を引き起こす．このような場合に，並行するデフォルト規則を非決定的規則に変換する．いま，デフォルト規則のヘッドが γ であるとき，NAF式 $not\,\gamma$ をボディに追加する．この例では，次のようになる．

```
(flies(A):-bird(A),/+ab1(A),/+ -flies(A)),
```

```
ab1(5),ab1(6),ab1(7),ab1(8),
(-flies(B):-bird(B),/+ab2(B),/+flies(B)),
ab2(1),ab2(2),ab2(3),ab2(4)
```

ここで2つの規則は非決定規則として働き，各デフォルト規則は

$$\gamma_i(\boldsymbol{x}) \leftarrow B_i,\ not\ \mathtt{ab}_i(\boldsymbol{x}),\ not\ \overline{\gamma_i(\boldsymbol{x})}$$

の形式で表現される（$\boldsymbol{x}$ は変数の列，B_i はボディのリテラルの積）．この形式の規則は，ELP でデフォルトを表現するのに適したものであることが [Baral 94] において議論されている．このとき，追加された基礎項9に関する一般規則の基礎例は，

```
flies(9) :- bird(9), \+ab1(9), \+-flies(9).
-flies(9) :- bird(9), \+ab2(9), \+flies(9).
```

となる．よって鳥9について，飛ぶ場合と飛ばない場合の2つの解集合が得られる．

なお，非決定的規則を生成する別の例については [Inoue 97] を参照のこと．

3・2 階層的デフォルト

ここでは，さらに例外が階層的な構造を持つような規則を生成できるように LELP を拡張する．最初に打ち消し規則生成のためのアルゴリズム 2·4を修正する．

アルゴリズム 3・1 $Cancel2(AB, BG, T)$

入力: 例外 AB，背景知識 BG

出力: 例外に関する規則 T

(1) $GenRules(AB, BG, T)$,
ただし，$BG \cup T$ から帰結される $\mathtt{ab}_i$ 述語を持つすべての基礎リテラルの集合を $\mathbf{S}$ とし，AB の基礎例の集合を $\mathbf{A}$ とするとき，$|\mathbf{S} \setminus \mathbf{A}| < |\mathbf{A}|$ を満たすように T を生成する．

上記の条件 $|\mathbf{S} \setminus \mathbf{A}| < |\mathbf{A}|$ は，アルゴリズム 2·4におけるより強い条件 $\mathbf{S} = \mathbf{A}$ に代わるものである．ここで，集合 $\mathbf{S} \setminus \mathbf{A}$ は例外の例外を表す．この弱い条件下では，$\mathbf{S}$ は $\mathbf{A}$ を真に包含してもよいが，$\mathbf{S} \setminus \mathbf{A}$ の要素数は $\mathbf{A}$ の要素数を超えてはならない．本条件は，「どの階層においても例外の方が一般例よりも少ない」という**単調性の仮定**を表している．

階層的なデフォルトを学習するには，OWS と $Cancel2$ のアルゴリズムを再帰的に呼び出せばよい．例外が無くなるか一般化できなくなれば終了する．これらの拡張を施したアルゴリズム LELP2 を以下に示す．

アルゴリズム 3・2 $LELP2(E^+, E^-, BG, H)$

入力: (正) 例 E^+，(負) 例 E^-，背景知識 BG

出力: 規則 $H := R_3 \cup R_4 \cup R_5 \cup R_6 \cup R_7 \cup R_8$

(1) (a) E^+ に対する規則のみをデフォルトとするか，(b) E^- に対する規則のみをデフォルトとするか，(c) 並行するデフォルト規則を学習するかを決定する．
if (a) **then** $R_4 = R_6 = R_8 = \emptyset$ とし，(2), (4), (7), (9) を実行;
if (b) **then** $R_3 = R_5 = R_7 = \emptyset$ とし，(3), (5), (8), (10) を実行;
if (c) **then** $R_5 = R_6 = \emptyset$ とし，(2)〜(6), (9)〜(10) を実行;

(2) $GenRules(E^+, BG, R_1)$;

(3) $GenRules(E^-, BG, R_2)$;

(4) $OWS(R_1, E^+, E^-, BG, AB_1, R_3)$;

(5) $OWS(R_2, E^-, E^+, BG, AB_2, R_4)$;

(6) **if** $R_1 \cup R_2$ の中に矛盾を引き起こす規則が存在する **then** $R_3 \cup R_4$ を非決定規則に変換し，$R_5 = R_6 = R_7 = R_8 = \emptyset$ として終了;

(7) $Counter(E^-, AB_1, R_3, R_5)$;

(8) $Counter(E^+, AB_2, R_4, R_6)$;

(9) $ABs(E^+, E^-, AB_1, BG \cup R_3 \cup R_5, R_7)$;

(10) $ABs(E^-, E^+, AB_2, BG \cup R_4 \cup R_6, R_8)$.

アルゴリズム 3・3 $ABs(E^+, E^-, AB, BG, T)$

入力: 例 $E^+ \cup E^-$，例外 AB，背景知識 BG

出力: 規則 T

(1) **if** $AB = \emptyset$ **then** $T := \emptyset$ として終了;

(2) $Cancel2(AB, BG, R)$;

(3) $OWS(R, E^-, E^+, BG, AB_{low}, R')$;

(4) $ABs(E^-, E^+, AB_{low}, BG \cup R', T_{low})$;

(5) $T := R' \cup T_{low}$.

アルゴリズム 3·3において，AB_{low} は例外 AB をカバーするデフォルト打ち消し規則 R に対する例外であり，T_{low} は AB_{low} をカバーする規則である．

〔例 3・2〕（階層的デフォルト規則: 図 2 参照）

```
| ?- lelp([-flies(1),-flies(2),+flies(3),
   +flies(4),+flies(5),-flies(6),-flies(7),
   -flies(8),-flies(9),-flies(10),
   -flies(11),-flies(12)],
```

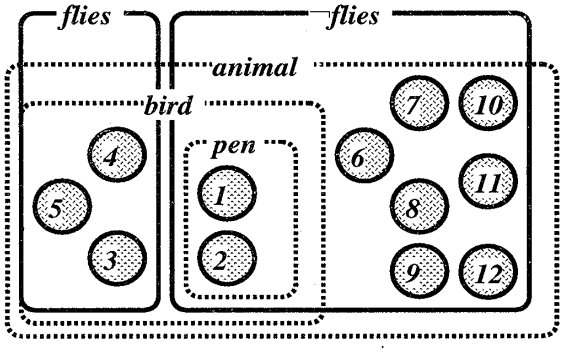

図2 階層の例

```
[pen(1),pen(2),bird(3),bird(4),bird(5),
 (bird(X) :- pen(X)),animal(6),animal(7),
 animal(8),animal(9),animal(10),
 animal(11),animal(12),
 (animal(X) :- bird(X))],Rules).
```

ここでは負例の方が多いために，最初に負の一般規則 (`-flies(A) :- animal(A)`) が生成される．次にOWSによって第一階層の例外集合 {`ab1(3)`, `ab1(4)`, `ab1(5)`} とデフォルト規則 (`-flies(A):- animal(A), \+ ab1(A)`) が生成される．さらに例外集合を基に，打ち消し規則 (`ab1(C) :- bird(C)`) と，反例を導く規則 (`flies(B) :- ab1(B)`) も生成される．ここで打ち消し規則に対してOWSが適用された結果，第二階層の例外集合 {`ab2(1)`, `ab2(2)`} とデフォルト規則が生成される．この時点で例外は存在しなくなり，プログラムは終了する．最終的な `Rules` は次の通り*．

```
(-flies(A):-animal(A),\+ab1(A)),
(flies(B):-ab1(B)),
(ab1(C):-bird(C),\+ab2(C)),
(ab2(D):-pen(D))
```

4. 関 連 研 究

Bain と Muggleton の CWS [Bain 92b] は，[Bain 92a] においてCIGOLとGOLEMの非単調版で用いられている．CWSは層状NLPにおいてNAFを含むデフォルト規則を生成する．CWSは二値の意味論におけるCWAに基づいており，モデルに含まれない基礎アトムを例外とみなす．これに対し，LELPではCWSの代わりにOWSを採用しているため，不完全な情報を三値の意味論のELPで表現することができる．

[Bergadano 95] では，SLDNF導出のトレース情報を用いて層状NLPを学習する $TRACY^{not}$ を提案している．彼らのシステムは事前に仮説空間を必要とするので，例外を表す $\mathtt{ab}_i$ のような新述語は生成できない．それゆえ，デフォルト規則を学習するよりも負の知識やCWAを含む規則を学習することに適している．[Martin 96] は，帰納学習の枠組でNLPに対して三値の意味論を用いたが，ELPを採用せずCWAが適用されるため，やはり2種類の否定が区別されない．

これまでに，学習するプログラムの形式として，フル形式のELPを用いたものはないが，論理否定の出現を制限した形式は [De Raedt 90, Dimopoulos 95] で用いられている．[De Raedt 90] は，最初にILPにおける三値の意味論の重要性を議論した．しかしながら，彼らはNAFを許さなかったので，例外のためのリストが各規則に対して必要となり，AIにおける**制限条件記述問題** (qualification problem) をひき起こす．[Dimopoulos 95] は，階層的な例外を含む規則を学習する方法を提案している．彼らはデフォルト規則を表現するのにNAFを使っておらず，優先関係による独自の非単調な意味論を採用している．一方，われわれはデフォルトを表現するのに標準的なELP形式を用いている．それゆえ，多くの既存のELPのための証明手続きを学習されたプログラムに対して用いることができる．さらに，[Dimopoulos 95] では，各々の負の情報が通常の特殊化過程で用いられるべきか，例外同定過程において用いられるべきかを決定する必要がある．LELPでは，そのような区別はNAFと論理否定の適切な使い分けによって明確に行うことができる．

なお，以上述べた過去の研究ではどれも非決定的規則を生成できない．そのため，複数の拡張を持つデフォルト理論を学習できないという問題点がある．LELPでは非決定的規則を導入することにより，層状プログラムではないプログラムが得られる．

最近になって，LELPに影響された形で，ELPを学習するシステムの提案が [Lamma 98] においても行われた．この研究においては，ELPの意味論にwell-founded semanticsを用いており，学習される規則の形式がLELPとはやや異なっている．

5. お わ り に

本研究では，例外を含む非単調な規則を学習する方法を提案し，学習システムLELPを紹介した．LELPは2種類の否定 (NAFと論理否定) を含む拡張論理プログラムを学習するプログラムの形式として採用し，不完全な情報の下での推論を表現できる．デフォルト規則はOWSを用いて生成され，例外の集合は打ち消し規則へと一般化される．LELPは三値の意味論の範囲内で，並行するデフォルト規則，階層的デフォルト規則，非決定的規則を学習することもできる．

なお *LELP*1 アルゴリズムにおいては，負の情報は正の仮説に対する例外の候補とみなして扱った．しかしながら，実世界では負の知識は学習する概念に無関係であることもある．この点に関して，例外からノイズを分離する方法が [Srinivasan 92] で提案されてい

---

* 本論文におけるアルゴリズム *LELP*2 は [Inoue 97] のものを改訂しており，*Counter* による反例を導く規則の生成を最上位レベルにおいて1度しか行っていない．これにより，本例においても以前のアルゴリズムにおいて出力されていた (`-flies(D):-ab2(D)`) を含めていない．実際にこの規則は冗長であり，[Lamma 98] により指摘された．

る．もう1つのアプローチとしては，各々の概念が例外を持つか否かの情報やCWAが適用できるか否かの情報を加えることがある．これらの拡張はLELPの範囲内で容易に実現できる．

最後に，不完全な知識を表現することができる論理プログラムとしては，ELP以外にもアブダクティブ論理プログラム(ALP)が存在する．[Inoue 98]では，LELPで獲得されたELPを基にしてALPを生成し，アブダクションのための新たな仮説の発見を行うことができるシステムLAELPが提案されており，LELPとLAELPの違いについても論じられている．

謝　辞

LELPの実現に携わった現在CSKの中西博一氏および神戸大学大学院自然科学研究科の中小路宗紀氏，本研究に対して貴重な意見を頂いた富士通研究所の有馬淳氏およびBologna大学のEvelina Lamma助教授，ならびに丁寧なコメントを頂いた査読者の方々に感謝します．なお，本研究の一部は文部省科学研究費補助金(基盤(C), 10680381)による．

◇参考文献◇

[Bain 92a] Bain, M.: Experiments in non-monotonic first-order induction, in [Muggleton 92a], pp. 423–436.

[Bain 92b] Bain, M. and Muggleton, S.: Non-monotonic learning, in [Muggleton 92a], pp. 145–161.

[Baral 94] Baral, C. and Gelfond, M.: Logic programming and knowledge representation, *Journal of Logic Programming*, 19/20, pp. 73–148 (1994).

[Bergadano 95] Bergadano, F., Gunetti, D., Nicosia, M. and Ruffo, G.: Learning logic programs with negation as failure, L. De Raedt (ed.), *Proc. of ILP-95*, pp. 33–51, K. U. Leuven (1995).

[De Raedt 90] De Raedt, L. and Bruynooghe, M.: On negation and three-valued logic in interactive concept-learning, *Proc. of ECAI'90*, pp. 207–212, Pitman (1990).

[Dimopoulos 95] Dimopoulos, Y. and Kakas, A.: Learning non-monotonic logic programs: learning exceptions, N. Lavrač and S. Wrobel (eds.), *Proc. of ECML-95*, pp. 122–137, LNAI 912, Springer (1995).

[Gelfond 91] Gelfond, M. and Lifschitz, V.: Classical negation in logic programs and disjunctive databases, *New Generation Computing*, 9(3,4), pp. 365–385 (1991).

[Helft 89] Helft, N.: Induction as nonmonotonic inference, *Proc. of KR'89*, pp. 149–156, Morgan Kaufmann (1989).

[井上 96] 井上 克巳，中西 博一：失敗による否定を含む規則の自動生成，第25回人工知能学会人工知能基礎論研究会資料(SIG-FAI-9601), pp. 180–187 (1996).

[Inoue 97] Inoue, K. and Kudoh, Y.: Learning extended logic programs, *Proc. of IJCAI-97*, pp. 176–181, Morgan Kaufmann (1997).

[Inoue 98] Inoue, K. and Haneda, H.: Learning abductive and nonmonotonic logic programs, Peter A. Flach and Antonis C. Kakas (eds.), *Abductive and Inductive Reasoning —Essays on their Relation and Integration*, Kluwer Academic Publishers (1998).

[Lamma 98] Lamma, E., Riguzzi, F. and Pereira, L.M.: Learning with extended logic programs, J. Dix and J. Lobo (eds.), *Proceedings of the LP Track at the International NMR '98 Workshop* (Trento, Italy), Universität of Koblenz (1998).

[Lavrač 94] Lavrač, N. and Džeroski, S.: *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood (1994).

[Martin 96] Martin, L. and Vrain, C.: A three-valued framework for the induction of general logic programs, Luc De Raedt (ed.), *Advances in Inductive Logic Programming*, pp. 219–235, IOS Press (1996).

[Muggleton 92a] Muggleton, S. (ed.): *Inductive Logic Programming*, Academic Press, London (1992).

[Muggleton 92b] Muggleton, S. and Feng, C.: Efficient induction of logic programs, in [Muggleton 92a], pp. 281–298.

[Reiter 80] Reiter, R.: A logic for default reasoning, *Artificial Intelligence*, 13, pp. 81–132 (1980).

[Srinivasan 92] Srinivasan, A., Muggleton, S. and Bain, M.: Distinguishing exceptions from noise in non-monotonic learning, *Proc. of ILP-92*, ICOT (1992).

[Tamaki 84] Tamaki, S. and Sato, T.: Unfold/fold transformation of logic programs, *Proc. 2nd Int'l Conf. on Logic Programming*, pp. 127–138 (1984).

〔担当委員：國藤　進〕

著者紹介

井上　克巳(正会員)

1959年生まれ．1982年京都大学工学部数理工学科卒業．1984年同大学院工学研究科数理工学専攻修士課程修了．松下電器産業(株)，(財)新世代コンピュータ技術開発機構，豊橋技術科学大学を経て，1997年より神戸大学工学部助教授．京都大学博士(工学)．人工知能，論理プログラミング，計算機科学に関する教育研究に従事．情報処理学会，AAAI各会員．〈inoue@eedept.kobe-u.ac.jp〉

工藤　嘉晃

1974年生まれ．1997年豊橋技術科学大学工学部情報工学課程卒業．1999年北海道大学大学院工学研究科修士課程電子情報工学専攻修了．現在同博士課程に在学中．帰納的論理プログラミング(ILP)，データベースからの知識発見(KDD)に関する研究に従事．〈kudo@db-ei.eng.hokudai.ac.jp〉

羽根田　博正

1939年生まれ．1972年カリフォルニア大学(バークレイ校)大学院博士課程修了．同年神戸大学工学部助教授．1982年同教授．1997年IEEE産業エレクトロニクス部門学会会長．計算機援用設計・解析に関する研究に従事．Ph.D. 計測自動制御学会，システム制御情報学会，電気学会，電子情報通信学会，情報処理学会，ACM，IEEE(フェロー)各会員．〈haneda@eedept.kobe-u.ac.jp〉