

```

正例 ..... E+      class(dog,mammal). min([2],[1]).
負例 ..... E-      :-class(dog,fish). :-class(dog,bird).
背景知識 ... BK    has_legs(dog,4). has_milk(dog). append([W|X],Y,[W|Z]):-append(X,Y,Z).
モード宣言
modeh ...         仮説の頭部に現れるものを宣言 . 学習させたい対象を宣言する .
modeb ...         仮説の本体部に現れるものを宣言 . 背景知識を宣言する .

```

```

:- modeh(1,class(+animal,#class))?
:- modeb(1,has_milk(+animal))?
:- modeb(1,has_gills(+animal))?
:- modeb(1,has_covering(+animal,#covering))?

```

+animal,#covering はタイプ情報 . +は入力, -は出力, #は定数を意味する .

```

:- modeh(1,reverse([+int|+list],-list))?
:- modeb(1,reverse(+list,-list))?
:- modeb(1,append(+list,[+int],-list))?

```

ここで 'reverse' は事例であり, かつ本体部に再帰的に現れるものである . modeh,modeb の両方に宣言されている .

各モードの第 1 引数は, recall number である . (Progol では非決定的な述語を許すので, 述語呼び出しが成功した場合, 他の解を (recall -1) 回までリトライされる . 呼び出しは失敗するまで続けられる .)

タイプ情報

```

animal(dog). animal(dolphin). animal(platypus). animal(bat).
animal(trout). animal(herring). animal(shark). animal(eel).
animal(lizard). animal(crocodile). animal(t_rex). animal(turtle).
animal(snake). animal(eagle). animal(ostrich). animal(penguin).

class(mammal). class(fish). class(reptile). class(bird).

covering(hair). covering(none). covering(scales). covering(feathers).

habitat(land). habitat(water). habitat(air).

```

以下のような宣言方法も可能 .

1 Progol の概要

1.1 Progol の問題設定

Progol とは, S.Muggleton らによって開発された, 帰納論理プログラミングシステムである . 帰納論理プログラミングは, BK を背景知識, E+ を正例, E- を負例とし,

$$\begin{cases} BK \not\models E^+ \\ BK \wedge E^- \not\models \square \end{cases} \text{ のとき, } \begin{cases} BK \wedge H \models E^+ \\ BK \wedge H \not\models E^- \end{cases}$$

なる仮説 H を求めるものである .

1.2 Progol の仮説生成の大まかな流れ

Progol は,

- (1) 正例集合の一つの要素と背景知識から, 最弱仮説 (Most Specific Hypothesis; MSH) を生成し,
- (2) MSH によって形成される候補仮説束内を A*-like 探索アルゴリズムに従い探索し, 最適な仮説を発生する,
- (3) 得られた仮説によって説明される正例を正例集合から取り除く, という処理を, 正例集合が空になるまで繰り返すことで, 仮説を生成する . この内 (2) の A*-like 探索では, 仮説評価のために, 仮説によって被覆される事例の数を計算する . この計算は被覆計算と呼ばれる . また (3) における処理を, 集合被覆アルゴリズムと呼ぶ .

Progol4.4 における仮説の評価基準は以下の通り .

$$F = \frac{p}{p - (n + c + h)}$$

p 仮説によってカバー (被覆) される正事例の数
n 仮説によってカバー (被覆) される負事例の数
c 仮説の本体部の長さ
h 仮説の頭部に現れる変数間の入出力関係を完成させるために, 本体部に追加する必要のあるリテラル数の最小値
m 定数
P 正事例の数

1.3 Progol の起動方法

入力データを, file.pl とすると,

```
%*tozaki/bin/progol4.4 file <Return>
```

と入力することで Progol が起動し, 学習が始まる . また, サンプルデータを "tozaki/progollex に置く .

```

animal(X) :- class(X, _).
class(X) :- class(_, X).
covering(X) :- has_covering(_, X).
habitat(X) :- habitat(_, X).

ilist(□). ilist([H|T]) :- int(H), ilist(T).

```

determination 宣言 ……複数の概念事例を学習する際、各概念事例がどの背景知識と共に一般化されるかを宣言する。

```

:- determination(grandfather_of/2, parent_of/2)?
:- determination(grandfather_of/2, mother_of/2)?
:- determination(grandfather_of/2, father_of/2)?
:- determination(parent_of/2, father_of/2)?
:- determination(parent_of/2, mother_of/2)?

```

commutative 宣言 ……背景知識の+type 引数の順番を入れ換えても、-type 変数の値が変わらないことを宣言する。(例えば、足し算 $\text{plus}(+X, +Y, -Z)$ は、第1引数と第2引数の順番を入れ換えても、Zの値は変わらない。このようなものを宣言する。)

```

:- commutative(mult/3)?
:- commutative(plus/3)?

```

パラメタ Progol には多くのパラメタが存在する。以下、特に重要だと思われるものについて簡単に説明する。

- c …… 仮説本体部に含まれるリテラル数の最大値。":-set(c,4)?" のように記述し、1以上の整数を値として取る。デフォルト値は4である。
- i …… 最弱仮説生成の際の変数深度の上限。":-set(i,3)?" のように記述し、1以上の整数を値として取る。デフォルト値は3である。
- noise …… 許容されるノイズの上限。(被覆される負事例*100)/(被覆される正事例+被覆される負事例)として計算される。":-set(noise,0)?" のように記述し、0以上100以下の整数を値として取る。デフォルト値は0である。
- nodes …… A*-like 探索において生成される候補仮説数の上限。":-set(nodes,200)?" のように記述し、1以上の整数を値として取る。デフォルト値は200である。
- h …… Prolog 実行における推論深度の最大値。":-set(h,30)?" のように記述し、1以上の整数を値として取る。デフォルト値は30である。

1.5 Progol の出力結果の読み方

動物分類問題における Progol の実行結果の一部を図1に示す。図中の%で始まるところは、後から付けたコメントである。

```

CProgol Version 4.4

[: - set(h,100)? - Time taken 0.00s]
[: - modeb(1,class(=animal,#class))? - Time taken 0.00s]
[: - modeb(1,has_milk(=animal))? - Time taken 0.00s]
[: - modeb(1,has_gills(=animal))? - Time taken 0.00s]

..... %モード宣言の読み込み.

[Generalising class(dog,mammal).]
[Most specific clause is] %最弱仮説の生成

class(A,mammal) :- has_milk(A), has_covering(A,hair), has_legs(A),
4), homeothermic(A), habitat(A,land).

[C:-28,4,10,0 class(A,mammal).] %探索の開始
[C:8,4,0,0 class(A,mammal) :- has_milk(A).]
[C:5,3,0,0 class(A,mammal) :- has_covering(A,hair).]
[C:-4,4,3,0 class(A,mammal) :- homeothermic(A).]
[4 explored search nodes]
f=8,p=4,r=0,h=0
[Result of search is]

class(A,mammal) :- has_milk(A). %この探索によって得られた仮説

[4 redundant clauses retracted] %上記の仮説によって被覆された事例の数
[Generalising class(snake,reptile).] %残った事例を対象に、帰納推論を続ける
[Most specific clause is]

class(A,reptile) :- has_covering(A,scales), has_legs(A,0), has_eggs(A),
habitat(A,land).

[C:-28,4,10,0 class(A,reptile).]
[C:0,4,2,0 class(A,reptile) :- has_covering(A,scales).]
[2 explored search nodes]
f=0,p=4,r=2,h=0
[No compression] %探索の失敗、適切な仮説が発見できなかったことを表す.

.....

class(snake,reptile). %最後に、最終的に得られた仮説集合が提示される.
class(A,mammal) :- has_milk(A).
class(A,fish) :- has_gills(A).
class(A,bird) :- has_covering(A,feathers).
class(A,reptile) :- has_covering(A,scales), has_legs(A,4).
[Total number of clauses = 5]

```

図 1: Progol の実行結果の一部

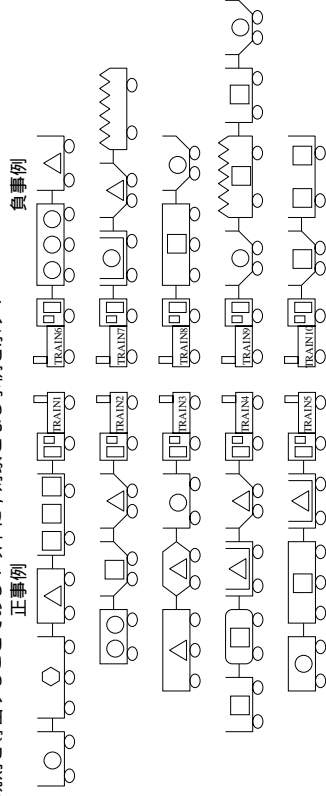
1.6 うまくいかないときは？

- (1) データがまったく一般化されない場合
 - モード宣言が正しいかどうかを確認する。特にタイプを確認する。
 - プログラムを読み込んだ時, `syntax error` などが出ていないか確認する。
- (2) 仮説が出るのだが, 正しくない(不十分, 未完など)
 - その仮説だけでは説明できない事例を加えてみる。
 - その仮説に反する負事例を加えてみる。
- 与える値の範囲を制限する。例えば, リスト中に現れる整数を 1 ~ 5 の値に制限するなど。
- 正事例の順番を入れ替えてみる。
- (3) 仮説の生成に時間がかかる。
 - パラメータの設定を変えてみる。

2 練習問題

2.1 貨物列車の分類問題

与えられた問題は, 連結されている車の情報をを用いて, 東行きの貨物列車と西行きの貨物列車とを分ける規則を導出することである。以下に, 対象となる事例を示す。



貨物列車の分類問題

問題 1. 問題の Prolog による表現:

<code>east(X).</code>	貨物列車 X は東へ向かう
<code>has_car(X, Y).</code>	車 Y は貨物列車 X に連結されている
<code>in_front(X, Y).</code>	車 X は車 Y の前にある
<code>open(X).</code>	車 X は屋根がない
<code>closed(X).</code>	車 X は屋根がある
<code>short(X).</code>	車 X は長さが短い
<code>long(X).</code>	車 X は長さが長い
<code>load(X, Y, Z).</code>	車 X は Y の形をした荷物を Z 個積んでいる
<code>shape(X, Y).</code>	車 X の形は Y である
<code>wheels(X, Y).</code>	車 X にはタイヤが Y 個ついている

という述語を用いて, この問題を Prolog を用いて表現してみよう。

- 問題 2. モード宣言の設定: 次にモード宣言を定義してみよう。モード宣言には, `modeh, modeb` の区別の他に, 引数の入出力情報や, タイプを記述する必要がある。これらが, どの様になるか, 考えてみよう。
- 問題 3. タイプ情報の記述: 次に, モード宣言に従って, タイプ情報を記述しよう。
- 問題 4. 実際の学習: 作成した入カファイルを用いて, 実際の学習を行ってみよう。

2.2 血縁関係の学習

図 2 に血縁関係のデータベースを示す。

<code>parent(mary, tim).</code>	<code>parent(mary, sally).</code>	<code>parent(bob, tim).</code>
<code>parent(bob, sally).</code>	<code>parent(jack, jimmy).</code>	<code>parent(betty, jimmy).</code>
<code>parent(sam, peg).</code>	<code>parent(jane, peg).</code>	<code>parent(tom, mary).</code>
<code>parent(tom, jane).</code>	<code>parent(tom, jack).</code>	<code>parent(joseph, tom).</code>
<code>parent(joseph, elizabeth).</code>	<code>parent(joseph, peter).</code>	<code>parent(elizabeth, mary).</code>
<code>parent(elizabeth, jack).</code>	<code>parent(elizabeth, jack).</code>	<code>parent(peter, joan).</code>
<code>parent(ellen, joan).</code>	<code>parent(virginia, tom).</code>	<code>parent(virginia, elizabeth).</code>
<code>parent(virginia, peter).</code>	<code>parent(martin, joseph).</code>	<code>parent(cathy, joseph).</code>
<code>female(mary).</code>	<code>female(sally).</code>	<code>male(tim).</code>
<code>female(jane).</code>	<code>female(betty).</code>	<code>male(bob).</code>
<code>female(peg).</code>	<code>female(elizabeth).</code>	<code>male(jack).</code>
<code>female(virginia).</code>	<code>female(cathy).</code>	<code>male(peter).</code>
<code>female(ellen).</code>	<code>female(hellen).</code>	<code>male(martin).</code>
<code>father(tom, mary).</code>	<code>father(tom, jane).</code>	<code>mother(mary, tim).</code>
<code>father(tom, jack).</code>	<code>father(bob, tim).</code>	<code>mother(jane, peg).</code>
<code>father(bob, sally).</code>	<code>father(sam, peg).</code>	<code>mother(elizabeth, mary).</code>
<code>father(jack, jimmy).</code>	<code>father(peter, joan).</code>	<code>mother(elizabeth, jack).</code>
<code>father(joseph, tom).</code>	<code>father(joseph, elizabeth).</code>	<code>mother(virginia, elizabeth).</code>
<code>father(joseph, peter).</code>	<code>father(martin, joseph).</code>	<code>mother(cathy, joseph).</code>
<code>grandfather(tom, tim).</code>	<code>grandfather(tom, sally).</code>	<code>grandfather(tom, peg).</code>
<code>grandfather(tom, jimmy).</code>	<code>grandfather(joseph, mary).</code>	<code>grandfather(joseph, jane).</code>
<code>grandfather(joseph, jack).</code>	<code>grandfather(joseph, mary).</code>	<code>grandfather(joseph, joan).</code>
<code>grandfather(joseph, jack).</code>	<code>grandfather(joseph, joan).</code>	<code>grandfather(martin, tom).</code>
<code>grandfather(martin, elizabeth).</code>	<code>grandfather(martin, peter).</code>	
<code>grandmother(elizabeth, tim).</code>	<code>grandmother(elizabeth, sally).</code>	<code>grandmother(elizabeth, peg).</code>
<code>grandmother(elizabeth, jimmy).</code>	<code>grandmother(virginia, mary).</code>	<code>grandmother(virginia, jane).</code>
<code>grandmother(virginia, jack).</code>	<code>grandmother(virginia, mary).</code>	<code>grandmother(virginia, jane).</code>
<code>grandmother(virginia, joan).</code>	<code>grandmother(virginia, joan).</code>	<code>grandmother(cathy, tom).</code>
<code>grandmother(cathy, elizabeth).</code>	<code>grandmother(cathy, peter).</code>	

図 2: 血縁関係のデータ

問題 1. `father` という概念を学習させてみよう。

- (1) モード宣言はどうなりますか?
- (2) タイプ情報はどうなりますか?
- (3) 負事例はどの様に準備しますか?

問題 2. 問題 1. で作ったファイルに情報を追加する形で, `mother` という概念を学習させてみよう。

- (1) モード宣言はどうなりますか?
- (2) タイプ情報はどうなりますか?
- (3) 負事例はどの様に準備しますか?

問題 3. 問題 2. で作ったファイルに情報を追加する形で, grandfather の概念を学習させてみよう.

- (1) モード宣言はどうなりますか?
 - (2) determination 宣言を使ってみましょう.
- 問題 4. 問題 3. で作ったファイルに情報を追加する形で, grandmother の概念を学習させてみよう.
- (1) モード宣言はどうなりますか?
 - (2) determination 宣言を使ってみましょう.

問題 5. 問題 4. 同様にして, greatgrandfather や, greatgrandmother の概念を学習させてみよう.

2.3 論理プログラムの自動合成

以下, いくつかの論理プログラムの自動合成問題を示す. それぞれに対して, 特に,

- (1) 背景知識として利用可能なプログラムの決定
- (2) モード宣言の設定
- (3) タイプ情報の設定

に注意して入力ファイルを作成し, 実際に Progol を用いて学習させてみよう.

1.reverse 第一引数のリストを反転したリストを第二引数に返すプログラム, reverse を学習させてみよう. この問題に対する正負事例は, 例えば, 以下のようになります.

```
正事例
reverse([1,2,3],[3,2,1]).
reverse([1,3],[3,1]).
reverse([2,3],[3,2]).
reverse([1],[1]).
reverse([1,2,3],[4,2,1]).
reverse([3],[3]).
reverse([],[]).
.....
```

ヒント:reverse プログラムは,

```
reverse([],[]).
reverse([A|B],C):- reverse(B,D), append(D,[A],C).
```

と定義されます.

2.sumlist 第一引数の整数リストの要素の和を, 第二引数に返すプログラム, sumlist を学習させてみよう. sumlist の正負事例は, 例えば, 以下のようになります.

```
正事例
sumlist([],0).
sumlist([3,4,5,6],18).
sumlist([1,2,3,4],10).
sumlist([9,1,2],12).
sumlist([1,2],3).
sumlist([4,5,6],15).
sumlist([2,3,4],9).
sumlist([5,6],11).
.....
```

ヒント: sumlist プログラムは,

```
sumlist([],0).
sumlist([A|B],C):- sumlist(B,D),plus(A,D,C).
```

と定義されます.

3.sampling 第二引数のリストの, 第一引数番目の要素を第三引数に返すプログラム, sampling を学習させてみよう. sampling の正負事例は, 例えば, 以下のようになります.

```
正事例
sampling(3,[a,b,c],c).
sampling(3,[b,c,d],d).
sampling(2,[c,d],d).
sampling(2,[b,c],c).
sampling(1,[d],d).
sampling(1,[b],b).
sampling(1,[c],c).
.....
```

ヒント: sampling プログラムは,

```
sampling(1,[A|B],A).
sampling(A,[B|C],D):-minus(A,1,E),sampling(E,C,D).
```

と定義されます.

3 Progolによる検定の方法

3.1 χ^2 検定の方法

以下に, Progol を利用した χ^2 検定のやり方を示す .

1. 問題ファイルの準備 :

- (a) モード宣言と背景知識からなるファイルを用意する (bk.pl)
- (b) 正負事例を, 訓練集合 (training.pl) とテスト集合 (test.pl) に分ける

2. Progol での操作:

- (a) Progol を interactive モードで起動

```
% tozaki/bin/progol4.4
```
- (b) 訓練集合と背景知識を読み込む

```
|- consult(bk)?
```
- (c) 仮説を生成する .

```
|- generalise(Pred/Arity)?
```
- (d) χ^2 検定を行う

```
|- test(test)?
```
- (e) Progol を「終了する」

```
|- quit?
```

3.2 クロスバリデーション (CV)

CV を行う場合, 事例集合を N 回に分割する (train_i.pl, train₀.pl ..., train_N.pl) . そして, train_i.pl をテスト集合に, そのほかの事例を訓練集合として, N 回 Progol を実行させる .

3.3 検定結果の読み方

検定結果の例

```
[False positive:] eastbound(train10).
[False negative:] - eastbound(train14).

[PREDICATE eastbound/1]

Contingency table=
  P | -----A----- ^A
  | | 3 | 1 | 4
  | | 2.0 | 2.0 |
  ~P | 1 | 3 | 4
  | | 2.0 | 2.0 |
  ~~~~~
  | | 4 | 4 | 8

[Overall accuracy= 75.00% +/- 15.31%]
[Chi-square = 0.50]
[Without Yates correction = 2.00]
[Chi-square probability = 0.4795]
```

検定結果の読み方

False positive ... 正事例のうち,間違って負事例と判別された事例

False negative ... 負事例のうち,間違って正事例と判断された事例

Contingency table ... P は 予測結果 (Predicted Result) を, A は実際の正負 (Actual Result) を意味する

Chi-square ... Yates 補正した χ^2 値

Chi-square probability ... 自由度 1, χ^2 値 Chi-square の場合の確率

A 例: krki

ファイル名と内容:

```
krki_bk.pl   モード宣言と背景知識
krki_train.pl 訓練集合
krki_test.pl テスト集合
```

これらのファイルは、

```
"/home/tozaki/progolx/krki"
```

に置いてある

実行のトレース (一部編集してある。また % はコメント)

```
% Progol の起動
% "tozaki/bin/progol4.4
CProgol Version 4.4
% 背景知識の読み込み
% consult(krki_bk)?
[:- set(i,1)? - Time taken 0.00s]
[:- set(c,2)? - Time taken 0.00s]
[:- set(noise,10)? - Time taken 0.00s]
[:- modeh(1,illegal(+rf,+rf,+rf,+rf,+rf,+rf))?
  - Time taken 0.00s]
[:- modeb(1,lt(+rf,+rf))? - Time taken 0.00s]
[:- modeb(1,adj(+rf,+rf))? - Time taken 0.00s]
[:- commutative(adj)/2? - Time taken 0.00s]
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(krki_bk)? - Time taken 0.03s]

% 訓練集合の読み込み
% consult(krki_train)?
[Testing for contradictions]
[No contradictions found]
yes
[:- consult(krki_train)? - Time taken 0.11s]

% 仮説の生成 (illegal という述語名で引数が6)
% generalise(illegal/6)?
[Generalising illegal(5,2,5,2,7,3).]
[Most specific clause is]

illegal(A,B,A,B,C,D) :- lt(A,C), lt(B,A), lt(B,C), lt(B,D), lt(D,
A), lt(D,C), adj(A,A), adj(B,B), adj(B,D), adj(C,C),
adj(D,D).

[ C:149,6,0,0 illegal(A,B,A,B,C,D). ]
.....
.....

% 得られた結果
illegal(A,B,A,B,C,D).
illegal(A,B,C,D,C,E).
illegal(A,B,C,D,E,D).
illegal(A,B,C,D,A,E) :- adj(B,E).
illegal(A,B,C,D,E,F) :- adj(A,E), adj(B,F).
```

[Total number of clauses = 5]

yes

```
[:- generalise(illegal/6)? - Time taken 15.50s]
```

% カイ二乗検定

```
[:- test(krki_test)?
```

```
[False negative:] :- illegal(4,3,4,7,4,0).
```

```
[False negative:] :- illegal(1,5,1,7,1,1).
```

```
[PREDICATE illegal/6]
```

```
Contingency table=
P |-----A-----~A
| | 165 | 21 | 167
| | 56.7 | 110.3 |
~P | 0 | 319 | 319
| | 108.3 | 210.7 |
|-----A-----~A
| | 165 | 321 | 486
```

```
[Overall accuracy= 99.59% +/- 0.29%]
```

```
[Chi-square = 472.79]
```

```
[Without Yates correction = 477.19]
```

```
[Chi-square probability = 0.0000]
```

yes

```
[:- test(krki_test)? - Time taken 0.07s]
```

% Progol の終了

```
[:- quit?
```