

テストデータの実行とトレースの利用

例：第 8 回講義の練習問題 1 のプログラム

```

-----
attendants([], []).
attendants([(Name, Age) | L], [Name | S]) :- attendants(L, S).

test8_1(NL) :- attendants( [ (ann, 80), (bob, 40), (jim, 20), (liz, 16), (tom, 65) ], NL ).
-----

```

swipl 上での実行 (変数名は何でもよい)

```

?- test8_1(X).
X = [ann, bob, jim, liz, tom].

```

トレースモードの利用。ENT をタイプすることでゴールの呼び出し/終了が 1 段階ずつ表示される。

```

?- trace, test8_1(X).
Call: (11) test8_1(_1748) ?
Call: (12) attendants([(ann, 80), (bob, 40), (jim, 20), (liz, 16), (tom, 65)], _1748) ?
Call: (13) attendants([(bob, 40), (jim, 20), (liz, 16), (tom, 65)], _2326) ?
Call: (14) attendants([(jim, 20), (liz, 16), (tom, 65)], _2376) ?
Call: (15) attendants([(liz, 16), (tom, 65)], _2426) ?
Call: (16) attendants([(tom, 65)], _2476) ?
Call: (17) attendants([], _2526) ?
Exit: (17) attendants([], []) ?
Exit: (16) attendants([(tom, 65)], [tom]) ?
Exit: (15) attendants([(liz, 16), (tom, 65)], [liz, tom]) ?
Exit: (14) attendants([(jim, 20), (liz, 16), (tom, 65)], [jim, liz, tom]) ?
Exit: (13) attendants([(bob, 40), (jim, 20), (liz, 16), (tom, 65)], [bob, jim, liz, tom]) ?
Exit: (12) attendants([(ann, 80), (bob, 40), (jim, 20), (liz, 16), (tom, 65)], [ann, bob, jim, liz, tom]) ?
Exit: (11) test8_1([ann, bob, jim, liz, tom]) ?
X = [ann, bob, jim, liz, tom].

```

trace モードで s とタイプするとそのゴールの展開は表示されず結果が表示される。下の例では Call: (14) の行で s とタイプすることで、このゴールを実行した結果が Exit: (14) として表示されている。

```

[trace] ?- trace, test8_1(X).
Call: (11) test8_1(_5048) ?
Call: (12) attendants([(ann, 80), (bob, 40), (jim, 20), (liz, 16), (tom, 65)], _5048) ?
Call: (13) attendants([(bob, 40), (jim, 20), (liz, 16), (tom, 65)], _5616) ?
Call: (14) attendants([(jim, 20), (liz, 16), (tom, 65)], _5666) ? s
Exit: (14) attendants([(jim, 20), (liz, 16), (tom, 65)], [jim, liz, tom]) ?
Exit: (13) attendants([(bob, 40), (jim, 20), (liz, 16), (tom, 65)], [bob, jim, liz, tom]) ?
Exit: (12) attendants([(ann, 80), (bob, 40), (jim, 20), (liz, 16), (tom, 65)], [ann, bob, jim, liz, tom]) ?
Exit: (11) test8_1([ann, bob, jim, liz, tom]) ?
X = [ann, bob, jim, liz, tom].

```

長いまたは深く入れ子になったデータの表示方法

長いまたは深く入れ子になったデータは， default では ... と省略表示されるので，省略なしに見たい場合は以下のいずれかを行う．

[方法その 1] write の実行

exec(X) の結果 X を省略なしで表示させたい場合， ?- exec(X), write(X). を実行する．

[方法その 2] 表示モード切替え

trace モードで w とタイプすると省略なし， p とタイプすると，表示モードを省略ありに切替えることができる．また， trace 環境下で h とタイプすると，入力可能なコマンド一覧が表示されるので，そちらも参照のこと．

使用例

```
?- trace,foo(X).
```

```
X = [1, 2, 3, 4, 5, 6, 7, 8, 9|...] <-- ここで w とタイプ
```

```
X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] <--ここで p とタイプ
```

```
X = [1, 2, 3, 4, 5, 6, 7, 8, 9|...]
```

```
Yes
```

```
?-
```

Prolog における否定の扱い

そもそも古典論理において命題 P に対して「 $\neg P$ が真だと証明できる」と「 P が真だと証明できない」は別物である。

Prolog においては後者を以下のように扱う。

- 失敗による否定 (Negation as Failure)
- 論理的否定 (命題の否定が正しい) と異なる
- 不完全知識を扱う手法の 1 つ (不完全知識については「知識情報処理」の講義参照)
- 陽に記述されていない事実についてはその否定が成り立つと考える
- ゴール (原子論理式) G を証明しようとして失敗すれば (ヘッドを単一化できる節がない) $\text{not } G$ が成功する

全解探索

- 全解を収集するメタ的な述語 `setof`, `bagof`, `findall` (詳細は text pp.183-184 を参照.)
- ゴール `setof(X,P,L)` の意味
ゴール P が成功するすべての X の解のリストが L である。
 L は重複はなくソートされている。
- 例: 下のプログラム (データベース) に対し `setof` を実行した結果を示す。(処理系によって L の要素の順番が変わる可能性あり)

```
?- setof(N,edge(a,N),L).  
N = N,  
L = [c,d]
```

```
?- setof((N,M),edge(N,M),L).  
N = N,  
M = M,  
L = ([a,c),(a,d),(b,e),(b,g),(c,g),(d,g),(e,g),(s,a),(s,b)])
```

```
?- setof(N/M,edge(N,M),L).  
N = N,  
M = M,  
L = ([a/c,a/d,b/e,b/g,c/g,d/g,e/g,s/a,s/b])
```

```
edge(s,a).  
edge(s,b).  
edge(a,c).  
edge(a,d).  
edge(b,e).  
edge(b,g).  
edge(c,g).  
edge(d,g).  
edge(e,g).
```

練習問題

1. $\text{male}(X), \text{female}(X)$ をそれぞれ X が男性である, X が女性である, を表す述語とする. 以下のデータベースに対する男性の集合が Men であるという関係を表す述語 $\text{gather_men}(\text{Men})$ のプログラムを setof を使って作成せよ. 次に, このデータベースに対して男性の数が N 人であることを表す述語 $\text{number_of_men}(N)$ のプログラムを作成せよ. 必要に応じて自分で述語を定義すること.

```
male(tom).    female(liz).  
male(bob).    female(pat).  
male(jim).
```

2. ループのない有向グラフについて, $\text{edge}(N,M), \text{start}(N), \text{goal}(M)$ をそれぞれノード N からノード M へのエッジがある, ノード N が初期ノードである, ノード M が目標ノードである, ということを表す述語とする. 初期ノードから目標ノードへの経路があるかどうかを判定する述語 exist_path のプログラムを作成せよ (Hint: 「初期ノードから目標ノードへの経路がある」とは「 S が初期ノードであり, かつ, G が目標ノードであり, かつ, S から G へつながっている」と考えよ) また, 図 9.1 の有向グラフで初期ノードが s , 目標ノードが g のとき, $\text{edge}(N,M), \text{start}(N), \text{goal}(M)$ をそれぞれ具体的データとして記述し, exist_path が成功することを確認せよ. 必要に応じて自分で述語を定義すること (注: setof は使用しない.)
3. ループのない有向グラフについて, ノード N からノード M への経路は P であるという関係を表す述語 $\text{path}(N,M,P)$ のプログラムを作成せよ. 経路は N から M に行くときに経由するノードを通る順に記述したリストとする. たとえば $\text{path}(s,g,P)$ は $P=[s,a,c,g]$ となって成功する (注: setof は使用しない.)

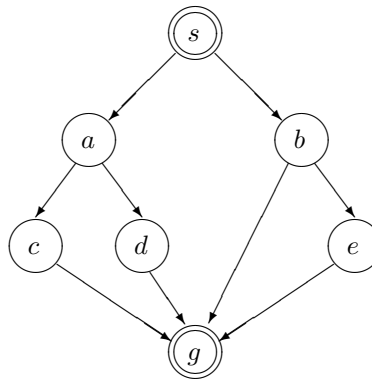


図 9.1

演習問題 (r9)

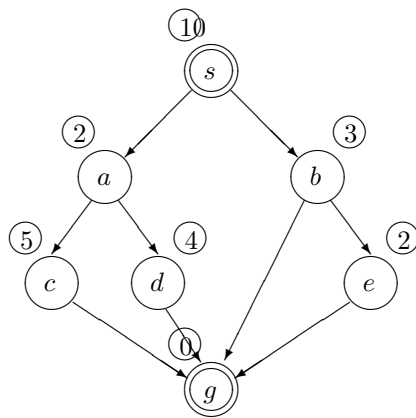
* のついている問題はオプション課題 .

注意 : 必要に応じて自分で述語を定義すること .

- (1) r9 の練習問題 1 で作成したデータベースにおいてそこに含まれるすべてのメンバーのリストが L であるという関係を表す述語 `all_members(L)` を `setof` および 2 つのリストを接続する組み込み述語 `append` を使って作成せよ . 実行すると $L=[\text{bob}, \text{jim}, \text{tom}, \text{pat}, \text{liz}]$ となって成功する .
- (2) r1 の練習問題 1 で作成したデータベースにおいて Y の祖先の数が N 人であるという関係を表す述語 `n_of_ancestors(Y,N)` のプログラムを `setof` を用いて作成せよ . たとえば , `n_of_ancestors(jim,N)` は $N=4$ となって成功する .
- (3) `node(N,V)` をノードの名前 N とその評価値 V を表す述語とする . `node(s,10)`. `node(a,2)`. `node(b,3)`. `node(c,5)`. `node(d,4)`. `node(e,2)`. `node(g,0)`. というデータベースが与えられたとき , `node(N,V)` と単一化が成功するような解を N/V の形にしてリストにまとめたものが L であるという関係を表す述語 `node_list(L)` のプログラムを `setof` を使って作成せよ . 実行すると $L = [a/2, b/3, c/5, d/4, e/2, g/0, s/10]$ となる .
- (4) (3) で得られたリストの各要素 $N_1/V_1, \dots, N_k/V_k$ に対して , $V_i (i = 1, \dots, k)$ の平均値 `Average` を求める `a_value(Average)` のプログラムを作成せよ . `a_value(A)` を実行すると , $A=3.71429$ となって成功する .
- (5) 練習問題 2 を参考に , ループのない有向グラフとそのノード 2 つが初期ノード , 目標ノードとして与えられたとき , 初期ノードから目標ノードに到る経路 `Path` を縦型探索 (深さ優先探索) を使って求める `dfs(Path)` のプログラムを作成し , 図 9.1 のグラフに関して全解が求まることを確認せよ . 縦型探索の場合は左から順に経路を探索するため , 練習問題 3 の解を利用すればよい .
- (6)* ループのない (予測コストつき) 有向グラフとそのノード 2 つが初期ノード , 目標ノードとして与えられたとき , 初期ノードから目標ノードに到る経路 `Path` を山登り法を使って求める `hc(Path)` のプログラムを作成し , 図 9.2 のグラフに関して `hc(Path)` を実行すると `Path=[s,a,d,g]` となって成功することを確認せよ . ノードに付加された数値は予測コスト (そのノードからゴールまでのヒューリスティック関数の値) である .
- (7) r9 の練習問題 1 で作成したデータベースに対して以下をレポートせよ . (i) このデータベースに対して :- `female(pam)`. を実行した結果およびその理由 . (ii) 次に , このデータベースに

```
female(X) :- not(male(X)).
```

という節を追加したとき , :- `female(pam)`. を実行した結果およびその理由 .



9.2