決定的プログラミングとカットオペレータ

- Prolog は，単一化に失敗したら**バックトラック (後戻り)** して別の解をさがしにいく (別の節とヘッドユニフィケーションしようとする)
- **カットオペレータ**を挿入して余計な部分を探索しないように制御する

  ```
  H :- B1, ..., Bm, !, ..., Bn.
  ```

  - ！より左側にはバックトラックしない (これらの別解を求めることはしない)
  - B1,...,Bm からは (たとえ複数存在したとしても) 高々1 つの解しか求まらない
- カットを挿入する位置は，そのゴールが成功すればその節の選択が確定するところ
- 無駄な計算 (新たな解が得られないことがわかっている計算部分) を省くのが目的
- カットを挿入せずに，各節を相互排他的 (単一データに対して単一節のみヘッドユニフィケーションに成功するよう) に記述する方法もある (決定的プログラミング)

例：最大値の計算

```
/* max の定義が def1,def2 それぞれの場合について do(3,2) を実行すると,
まず max の第 1 節と単一化され，その後 3<0 を実行して失敗する．その後 ... */
do(X,Y) :- max(X,Y,Z), Z<0.

/* def1 バックトラックして第 2 節のボディ部 3<2 で失敗して終了 */
max(X,Y,X) :- X>=Y.
max(X,Y,Y) :- X<Y.

/* def2 バックトラックせずすぐに終了 */
max(X,Y,X) :- X>=Y, !.
max(X,Y,Y).
```

例：リストの要素の抽出

```
/* member の定義が def1,def2 それぞれの場合について do を実行すると,
まず member の第 1 節と単一化されて X=a となり，a=b を実行して失敗する．その後... */
do :- member(X,[a,b,c]), X=b.

/* def1 バックトラックして第 2 節との単一化が成功し，X=b となって b=b が成功する．
member(X,[X|L]).
member(X,[Y|L]) :- member(X,L).

/* def2 バックトラックせずすぐに終了して失敗する．*/
member(X,[X|L]) :- !.
member(X,[Y|L]) :- member(X,L).
```

全解探索

- 全解を収集するメタ的な述語 **setof,bagof** (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 の解代入のリストが L である．
- setof の場合，L は重複はなくソートされている．bagof の場合，L は重複もありソートされていない．
- 例：右のプログラムに対し実行すると，以下のようになる．

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

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

```
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).
```

Prolog における否定の扱い

- 失敗による否定 (Negation as Failure)
- 論理的否定と異なる
- 不完全知識を扱う手法の 1 つ
- 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) not G が成功する

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
?-
```

練習問題

1. male(X),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 gather_men(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 number_of_men(N) のプログラムを作成せよ．

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

2. 上のデータベースに対して`:- female(pam).`を実行するとどうなるか確かめよ．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).`を実行するとどうなるか確かめ，なぜそうなるのか理由を考察せよ．

3. 有向グラフについて，edge(N,M), start(N), goal(N) 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 N 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，edge(N,M), start(N), goal(N) 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かめよ．(注：setof は使用しない．)

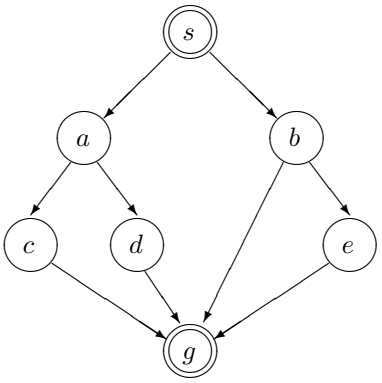

図 9.1

演習問題 (r9)

∗ のついている問題はオプション課題.
カットオペレータは特に使う必要なない．(もちろん使って効率化をはかってもかまわない).
注意：**必要に応じて自分で述語を定義すること．**たとえば (5) は引数 1 つの dfs という述語のみでは解けない.

(1) r9 の練習問題 1 のデータベースに対して，それ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) を組み込み述語 append を使って作成せよ．実行すると L=[bob, jim, tom, ann, liz] となって成功する.

(2) r1 の練習問題で作成したデータベースにおいて X,Y が兄弟姉妹であることを表す述語 siblings(X,Y) を定義せよ．(年齢は不問とするため，siblings(pat,ann) および siblings(ann,pat) はともに成功する．) この述語を用いて X の兄弟姉妹の数 (自分自身は数にいれない) が N であることを表す n_of_siblings(X,N) のプログラムを setof を用いて作成せよ．たとえば，n_of_siblings(pat,N) は N=1 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 $N_1/V_1, \ldots, N_k/V_k$ に対して，$V_i (i = 1, \ldots, k)$ 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) 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．Path は初期ノードを先頭とするリストの形で表すものとする．たとえば，dfs(Path) の解の 1 つは Path=[s,a,c,g] である．まず，単一解を求め，次に dfs に対して setof を使って 4 個の異なる解が求められることを確かめよ．(Hint: 探索したノードを前から順に Path にいれていく．)

(6)∗ ループのない (評価値つき) 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.

(7) r9 の練習問題 2 についてレポートせよ．レポートには以下を記述すること．(i) r9(1) 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.

(8) 以下のプログラムを実行した結果および考察をレポートせよ.

```
picnic1(Day) :- holiday(Day), sunny(Day).
picnic2(Day) :- holiday(Day), !, sunny(Day).
holiday(saturday).
holiday(sunday).
sunny(sunday).
```

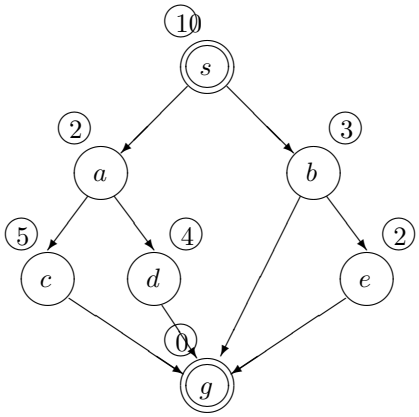

図 9.2