

バックトラック (後戻り) 機構 [修正版 : 2026/06/01]

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行く (別の節とヘッドユニフィケーションしようとする)

r7(5) 文法 (\mathcal{G}_2) になかったもののみがデータとして与えられるとする .

(\mathcal{G}_2)

`Term ::= Alphabet | Digit | f(Term) | g(Term,Term)`

`Alphabet ::= a | b`

`Digit ::= 0 | 1`

与えられた項 Term に出現するファンクタ f の数と, g の数の比較をし, 多い方が F であるという関係を表す述語 `compare_fnumbers(Term,F)` のプログラムを作成せよ . 同数の場合は, `F=same` とせよ .

`compare_fnumbers` を以下のように定義すると, 冗長であるだけでなく (なぜ冗長か?) `n_of_f`, `n_of_g` の定義によっては正しい解が得られない場合がある .

`compare_fnumbers(T,f) :- n_of_f(T,N1), n_of_g(T,N2), N1 > N2.`

`compare_fnumbers(T,g) :- n_of_f(T,N1), n_of_g(T,N2), N1 < N2.`

`compare_fnumbers(T,same).`

`n_of_g` の定義が program 1 のようになっているとする . (`n_of_f` も同様 .)

`% program 1`

`n_of_g(f(T),N) :- n_of_g(T,N).`

`n_of_g(g(T1,T2),N) :-`

`n_of_g(T1,N1), n_of_g(T2,N2), N is N1+N2+1.`

`n_of_g(_,0).`

`compare_fnumbers(g(f(a),a),F)` を実行すると, `compare_fnumbers` の第 1 節と単一化されて最初の 2 つのゴールが成功する (このとき `n_of_g` は `n_of_g` の第 2 節と単一化されている) . この結果, `N1=1`, `N2=1` となり, `compare_fnumbers` の 3 つ目のボディゴールで失敗する .

この後バックトラックして `n_of_g` を行くと, 今度は `n_of_g` の第 3 節と単一化され別解として `N2=0` が得られる . (実際は 2 回バックトラックが起こる .) この結果, `N1=1`, `N2=0` となって `compare_fnumbers` の第 1 節が成功し, 結果として `F=f` となる .

これを避けるためには `n_of_g` を program 2 のように定義する必要がある . このプログラムでは第 1 引数 T によって成功する節は高々 1 個であるようになっている (決定的プログラミング) .

`% program 2`

`n_of_g(T,0) :- alphabet(T).`

`n_of_g(T,0) :- digit(T).`

`n_of_g(f(T),N) :- n_of_g(T,N).`

`n_of_g(g(T1,T2),N) :-`

`n_of_g(T1,N1), n_of_g(T2,N2), N is N1+N2+1.`

別解として ‘!’ (cut operator) の利用が考えられる．**カットオペレータについては特に演習をせず，解説だけにとどめる．**

- 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
 $H :- B_1, \dots, B_m, !, \dots, B_n.$
 - ! 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)
 - B_1, \dots, B_m 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

`n_of_g` を program 3 のように定義すると `compare_fnumbers` の中でバックトラックが起ころうてもこのゴールが別解を探索することはない．

```
% program 3
n_of_g(f(T),N) :- !, n_of_g(T,N).
n_of_g(g(T1,T2),N) :- !,
    n_of_g(T1,N1), n_of_g(T2,N2), N is N1+N2+1.
n_of_g(_,0).
```

なお，`compare_fnumbers` は

```
compare_fnumbers(T,A) :- n_of_f(T,N1), n_of_g(T,N2), cmp(N1,N2,A).
```

として，`cmp` を別途定義するのが望ましい．

練習問題

1. 名前と年齢の組を要素とするリスト L に対して、すべての参加者の名前のリストが NL であるという関係を表す述語 `attendants(L,NL)` のプログラムを作成せよ。たとえば、`attendants([(ann,80),(bob,40),(jim,20),(liz,16),(tom,65)],NL)` は $NL = [ann,bob,jim,liz,tom]$ となって成功する。
2. 名前と年齢の組を要素とするリスト L に対して、すべての参加者の平均年齢が Y であるという関係を表す述語 `average_age(L,Y)` のプログラムを以下の手順で作成せよ。まず、1と同様にしてすべての参加者の年齢のリストが YL であるという関係を表す新たな述語を作成し、続いて YL の平均値を `r5(2)` で作成した `avarage` を用いて求める。たとえば、`average_age([(ann,80),(bob,40),(jim,20),(liz,16),(tom,65)], Y)` は $Y = 44.2$ となって成功する。

演習問題 (r8)

* のついている問題はオプションなのでできる者のみ解答せよ。

- (1) 名前と年齢の組を要素とするリスト L に対して、年齢が60歳以上の参加者の名前のリストが NL であるという関係を表す述語 `senior_attendants(L,NL)` のプログラムを作成せよ。たとえば、`senior_attendants([(ann,80),(bob,40),(jim,20),(liz,16),(tom,65)], NL)` は $NL = [ann,tom]$ となって成功する。
- (2) 英語と数学の点数の組を要素とするリスト L に対して(つまり各要素は各学生の成績に相当する)、英語の点数が数学の点数以上になっている人の人数が N であるという関係を表す述語 `english_better(L,N)` のプログラムを作成せよ。たとえば、`english_better([(55,80),(60,60),(90,20)], N)` は $N=2$ となって成功する。
- (3) 英語と数学の点数の組を要素とするリスト L に対して、英語または数学の点数が60点未満である人数が N であるという関係を表す述語 `failures(L,N)` のプログラムを作成せよ。たとえば、`failures([(55,20),(55,80)], N)` は $N=2$ 、`failures([(55,80),(60,60),(90,20)], N)` は $N=2$ となってそれぞれ成功する。
- (4) `r5(2)` で作成した述語 `average` を使ってあるテストに対する全受験者の英語の点数のリスト E と、数学の点数のリスト M に対して、英語の平均点と数学の平均点を求め、高い方が R であるという関係を表す述語 `compare_value(E,M,R)` のプログラムを作成せよ。ただし、 R の値は、英語の平均点が高ければ `english`、数学の平均点が高ければ `math`、同点ならば `same` とする。たとえば `compare_value([90,75,72], [55,68,83], R)` は $R=english$ となって成功する。必要ならば指定されたもの以外の述語も定義すること。
- (5) 名前と年齢の組を要素とするリスト L に対して、年齢が60歳以上を `Senior`、20歳以上60歳未満を `Adult`、20歳未満を `Junior` というリストに要素を分類する述語 `classify(L,Senior,Adult,Junior)` のプログラムを作成せよ。たとえば、`classify([(ann,80),(bob,40),(jim,20),(liz,16),(tom,65)], S, A, J)` は $S = [(ann,80),(tom,65)], A = [(bob,40),(jim,20)], J = [(liz,16)]$ となって成功する。
- (6)* $L1$ を `red(N)`, `white(N)`, `blue(N)` を要素としてもつリストとする。 N は1以上の自然数で、それぞれの色ごとの出現順を表す。 $L1$ を `red`, `white`, `blue` の色の順になるように並び替えたリストが $L2$ であるという関係を表す述語 `sort_by_color(L1,L2)` のプログラムを作成せよ。ただし、同一色の中での順番は変わらない。たとえば、

`sort_by_color([red(1),white(1),blue(1),white(2),red(2)], L2)` は
`L2 = [red(1),red(2),white(1),white(2),blue(1)]` となって成功する . (Hint: `r5(7)*` を参考
にせよ .)

- (7) 今回の演習問題 r8 (1)-(6) の解答にあたって生成 AI を少しでも使用したものの番号を
すべて記述せよ .