実行過程

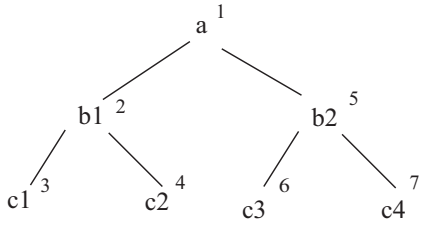
基本的な実行順序 - 深さ優先 (depth-first)

例 1

```
a :- b1, b2.   % 確定節 (rule) b1 /\ b2 -> a

b1 :- c1, c2.
b2 :- c3, c4.

% 単位節 (fact)
c1.  c2.  c3.  c4.
```



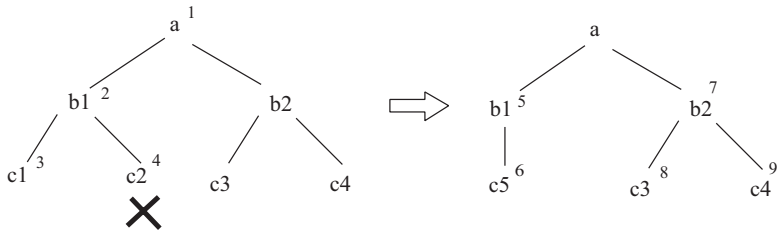
後戻り (backtrack)

例 2

```
a :- b1, b2.   % 確定節 (rule) b1 /\ b2 -> a

b1 :- c1, c2.
b1 :- c5.
b2 :- c3, c4.

% 単位節 (fact)
c1.  c3.  c4.  c5.
c2 :- fail.
```



単一化と実行過程

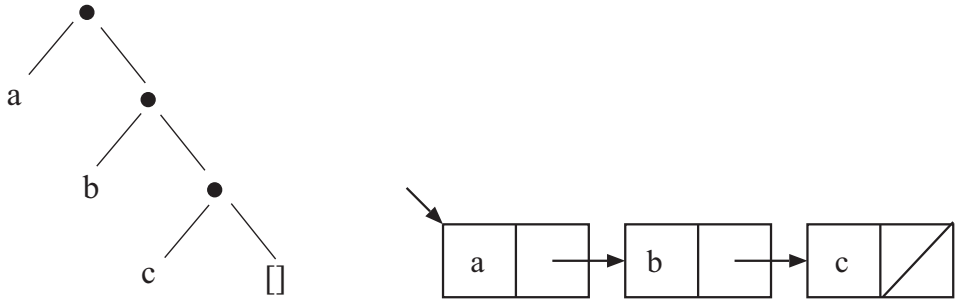
単一化 (unification)
直観的には，述語 (ゴール) 同士の変数に矛盾のないような代入をして見た目をそろえること
head unification
ゴール節のゴールと確定節のヘッド部のゴールの間の単一化．この結果確定節のボディ部に単一化の結果が反映される．

```
sum(0,0).
sum(N,M) :- N1 is N-1, sum(N1,M1), M is N+M1.

    ?- sum(3,X).
      |   sum(N,M) :- N1 is N-1, sum(N1,M1), M is N+M1.
単一化|   /      { N=3, X=M }
      |  /
      V
    ?- N1 is 3-1, sum(N1,M1), X is 3+M1.
      |  第 1 ゴールの実行と成功
      V
    ?- sum(2,M1), X is 3+M1.   (i)
      |   sum(Nr,Mr) :- Nr1 is Nr-1, sum(Nr1,Mr1), Mr is Nr+Mr1.
単一化|   /      { Nr=2, M1=Mr }
      |  /
      V
    ?- Nr1 is 2-1, sum(Nr1,Mr1), M1 is 2+Mr1.
      |  第 1 ゴールの実行と成功
      V
    ?- sum(1,Mr), M1 is 2+Mr1.   (ii)
      |  sum(Nrr,Mrr) :- Nrr1 is Nrr-1, sum(Nrr1,Mrr1), Mrr is Nrr+Mrr1.
単一化|   /      { Nrr=1, Mr=Mrr }
      |  /
      V
    ?- Nrr1 is 1-1, sum(Nrr1,Mrr1), Mr1 is 1+Mrr1.
      |  第 1 ゴールの実行と成功
      V
    ?- sum(0,Mrr1), Mr1 is 1+Mrr1.  (iii)
      |  sum(0,0).
単一化|   /       { Mrr1=0 }
      |  /
      V
    ?- Mr1 is 1+0.  (iii) の残ったゴールを実行
      |
      |  { Mr1=1 }
      V
    ?- M1 is 2+1.   (ii) の残ったゴールを実行
      |
      |  { M1=3 }
      V
    ?- X is 3+3.   (i) の残ったゴールを実行
      |
      |  { X=6 }
      V
    true           すべてのゴールが true になって終了
```

リスト構造（教科書 3.1,3.2 節）

[a,b,c]



- リストは頭部 (head) と尾部 (tail) から成る．
- **尾部はまたリストである．**
- リストの要素の数をリストの長さという．
- H を頭部，T を尾部とするリストを [H|T] と表現する．
  表記がまぎらわしいが，要はリストが 2 引数から構成されるデータ構造であり，T はリストになることを理解すること．

実行結果行なわれる単一化 (unification)

```
?- [H|T]=[a,b,c].
       H=a, T=[b,c]
?- [H|T]=[c].
       H=c, T=[]
?- [H1,H2|T]=[a,b,c].
       H1=a, H2=b, T=[c]
```

例 1：メンバー (教科書 p.69 参照)

```
member(X,[X|T]).
member(X,[Y|T]) :- member(X,T).
```

例 2：リストのコピー

```
copy_list([],[]).
copy_list([X|X1],[X|Y1]) :- copy_list(X1,Y1)
```

例 3：リストの連接 (教科書 pp.69-70 conc と同じ)

```
append([],Y,Y).
append([X|Xs],Y,[X|Zs]) :- append(Xs,Y,Zs).
```

練習問題

1. L1, L2 を要素がすべて整数であるようなリストとする．L2 の各要素が L1 の各要素を 2 倍した値になっている関係を表す述語 double_num(L1,L2) を定義せよ．たとえば double_num([1,2,3],L) は L=[2,4,6] となって成功する．

演習問題 (r3)

以下の問題において，リストの先頭は 0 番目ではなく 1 番目と数える．また，(5)(6) はリストを使用しない．最初に正しい解が得られればよく，別解を求める必要はない．

(7) のレポートには以下の点を記述せよ．(i) 論理的意味，(ii) 実際に動かしたときの動作 (**結果ではなく，どのゴールとどの節のヘッドが単一化され，どのゴールが呼ばれるなどの動作を記述する**)，(iii) 自分が正しいプログラムができなかった場合は間違った点とその理由 (正しいプログラムができていた場合は「正しくできた」の一言でよい)，(iv) 新たな知見や疑問点 (もしあれば)．

∗ 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) リスト L の長さが N であるという関係を表す述語 list_length(L,N) を定義せよ．たとえば，list_length([a,b,c],N) は N=3 となって成功する．

(2) 要素がすべて整数であるようなリスト L の要素の和が S であるという関係を表す述語 sum_list(L,S) を定義せよ．たとえば，sum_list([1,2,3],S) は S=6 となって成功する．

(3) リスト L1 の要素がすべて整数であるとする．L1 の要素の中で偶数のみを取り出したリストが L2 であるような関係を表す述語 even_list(L1,L2) のプログラムを作成せよ．組み込みオペレータ mod を使用してよい．(mod(N,M) は整数 N を M で割った余りを返す.) たとえば even_list([3,5,4,10,8],L2) は L2=[4,10,8] となって成功する．

(4) リスト L の要素がすべて整数であるとする．L の要素の中で偶数の個数が C 個であるという関係を表す述語 number_of_evens(L,C) のプログラムを作成せよ．たとえば number_of_evens([3,5,4,10,8],C) は C=3 となって成功する．

(5) 図 3.1 は図 2.1 の有向グラフに，各エッジの距離が付加されたものである．このグラフにおいて，与えられた 2 点 N，M およびその間の距離 L の関係を表す述語 dist2(N,M,L) を定義せよ．ただし，2 点を入力，距離を出力として正常動作すればよいものとする．(複数解が存在する場合，全解が求められることを確認せよ．)

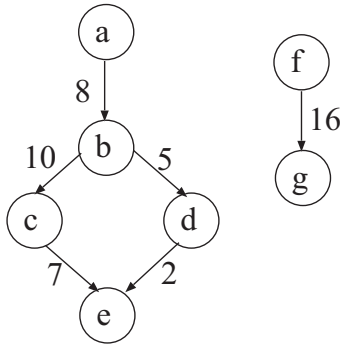


図 3.1

(6)∗ exp(N,M,X) を X が N の M 乗であるような関係を表すとするとき，exp を定義せよ．(ただし，N,M,X には 0 以上の自然数のみが入力されるのと考えてよい．) たとえば exp(3,4,81) は成功し，exp(3,4,X) は X = 81 を返す．また，実行時には第 1, 第 2 引数には入力されるものとし，exp(3,X,81), exp(X,4,81) などは計算できなくてよい．

(7) 解答例 r2_5 についてレポートせよ．