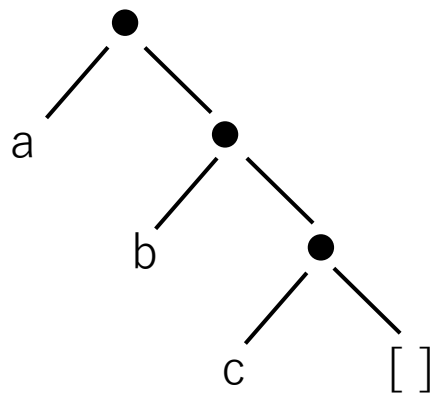
リストの要素のチェック

member

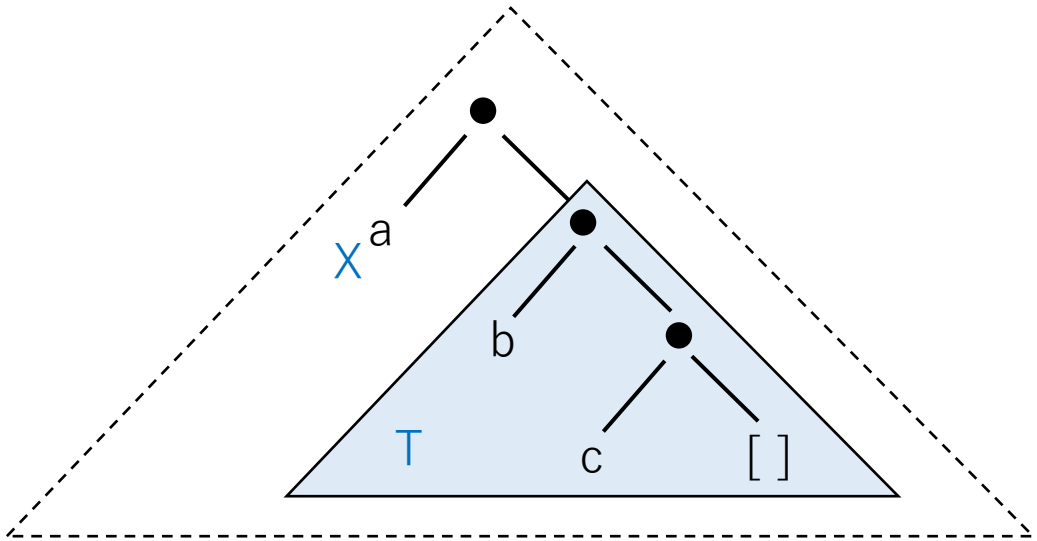
```
member(X,[X|T]).
member(X,[Y|T]) :- member(X,T)
```

(1) :- member(a,[a,b,c])の実行



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

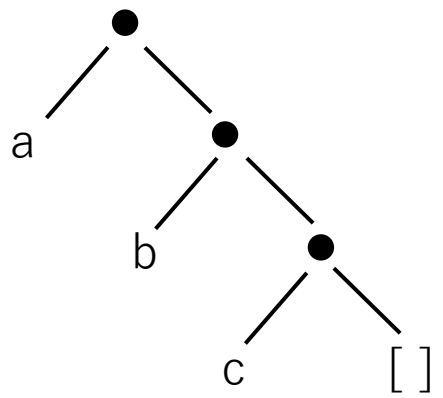
member(a,[a,b,c])



リストの第1要素が一致するので成功して終了

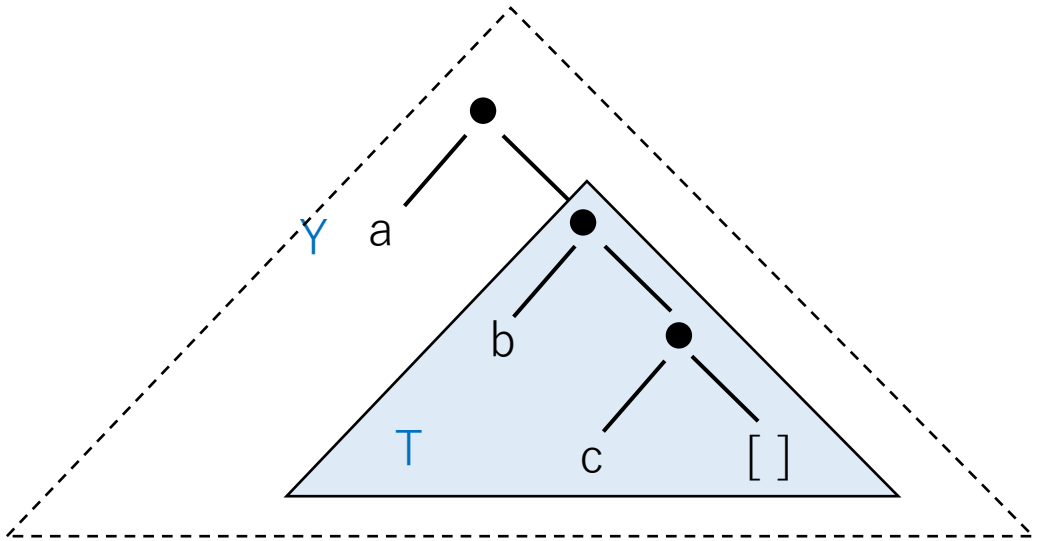
```
member(X,[X|T]).
member(X,[Y|T]) :- member(X,T)
```

(2) :- member(b,[a,b,c])の実行



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

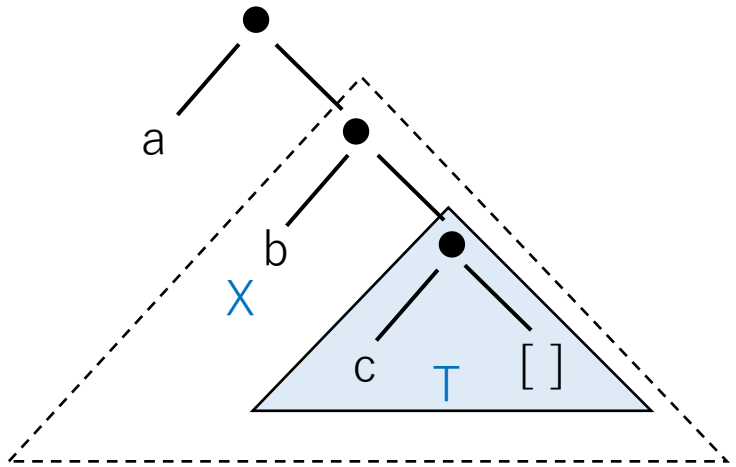
member(b,[a,b,c])



リストの第1要素が一致しないのでリストの残りを調べる

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

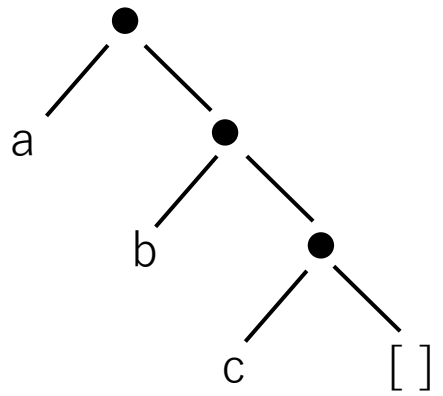
member(b,[b,c])



残りのリスト[b,c] の第1要素が一致するので成功して終了

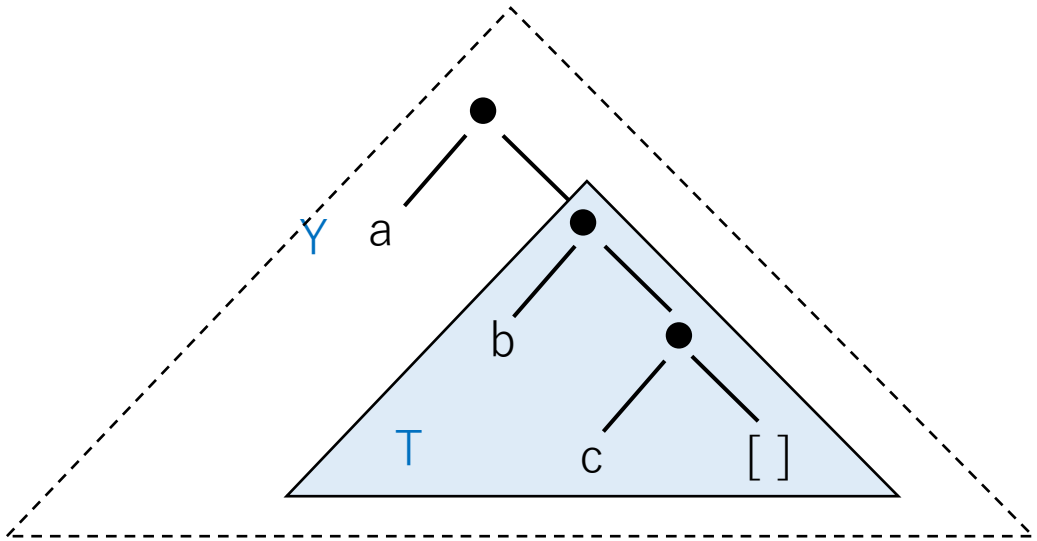
```
member(X,[X|T]).
member(X,[Y|T]) :- member(X,T)
```

(3) :- member(c,[a,b,c])の実行



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

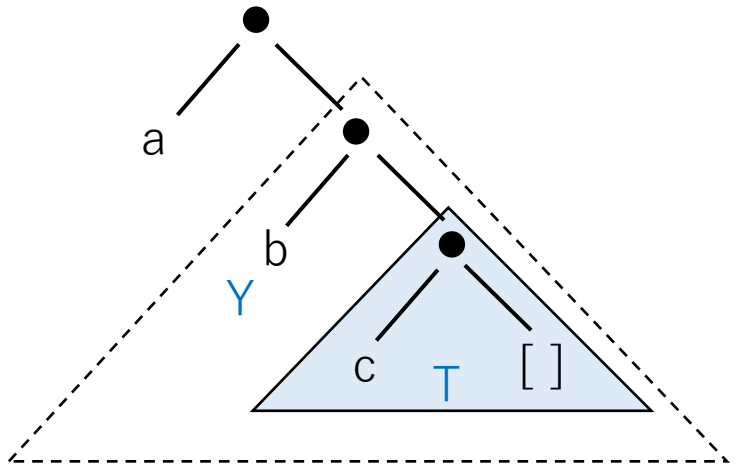
member(c,[a,b,c])



リストの第1要素が一致しないのでリストの残りを調べる

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

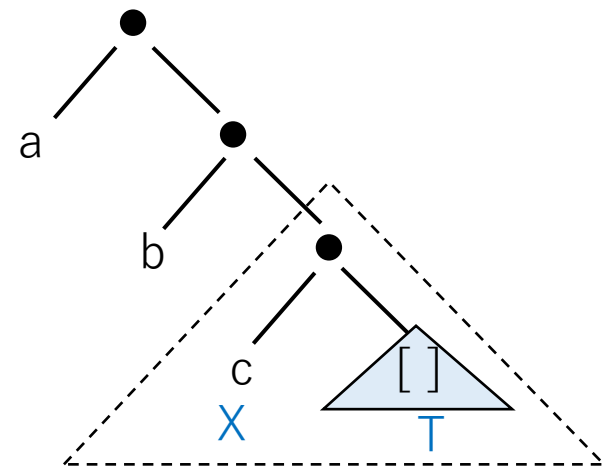
member(c,[b,c])



残りのリスト[b,c] の第1要素が一致しないのでリストの残りを調べる

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

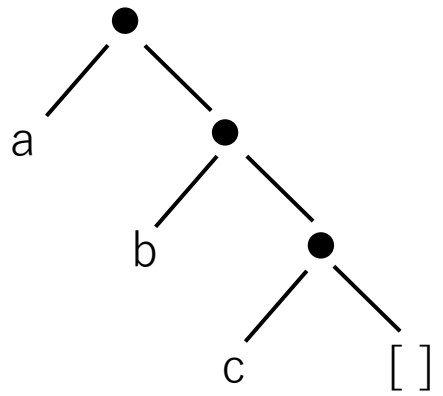
member(c,[c])



残りのリスト[c] の第1要素が一致するので成功して終了

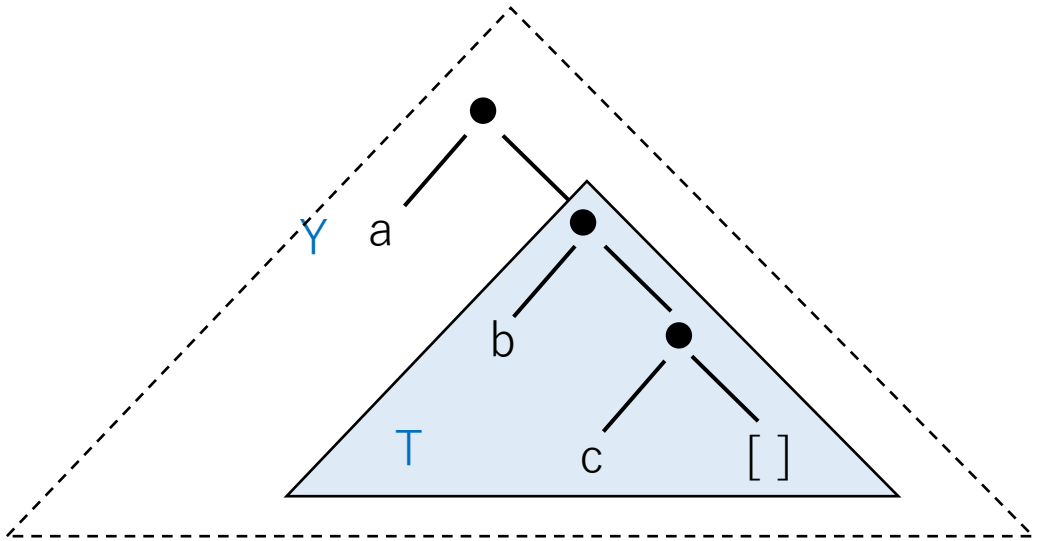
```
member(X,[X|T]).
member(X,[Y|T]) :- member(X,T)
```

(4) :- member(d,[a,b,c])の実行



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

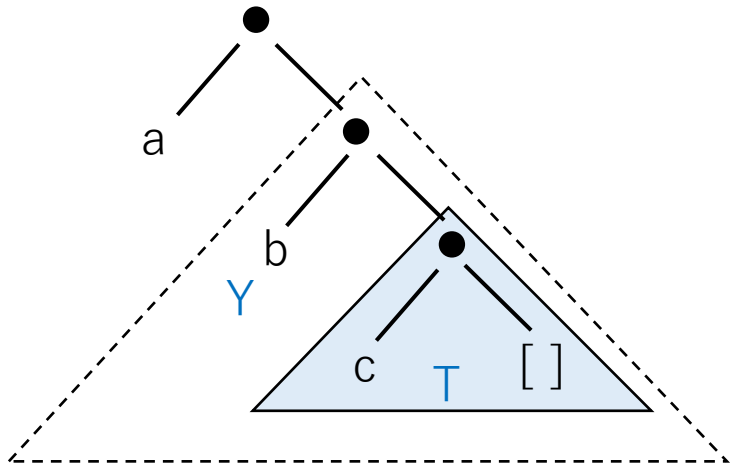
member(d,[a,b,c])



リストの第1要素が一致しないのでリストの残りを調べる

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

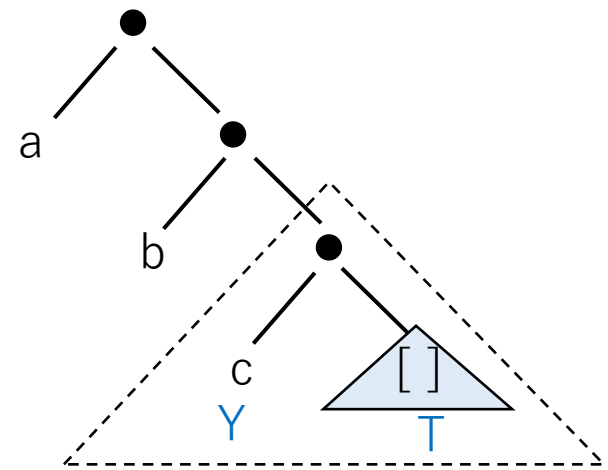
member(d,[b,c])



残りのリスト[b,c] の第1要素が一致しないのでリストの残りを調べる

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

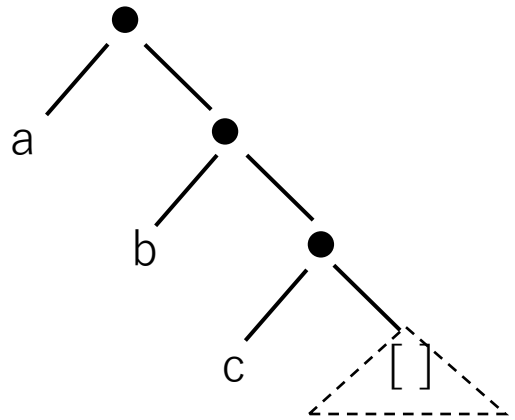
member(d,[c])



残りのリスト[c] の第1要素が一致しないのでリストの残りを調べる

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

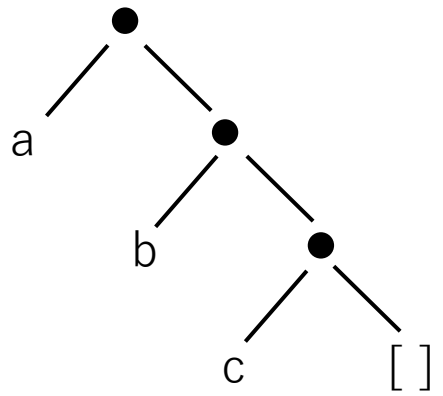
member(d,[ ])



単一化できる節がないので失敗して終了

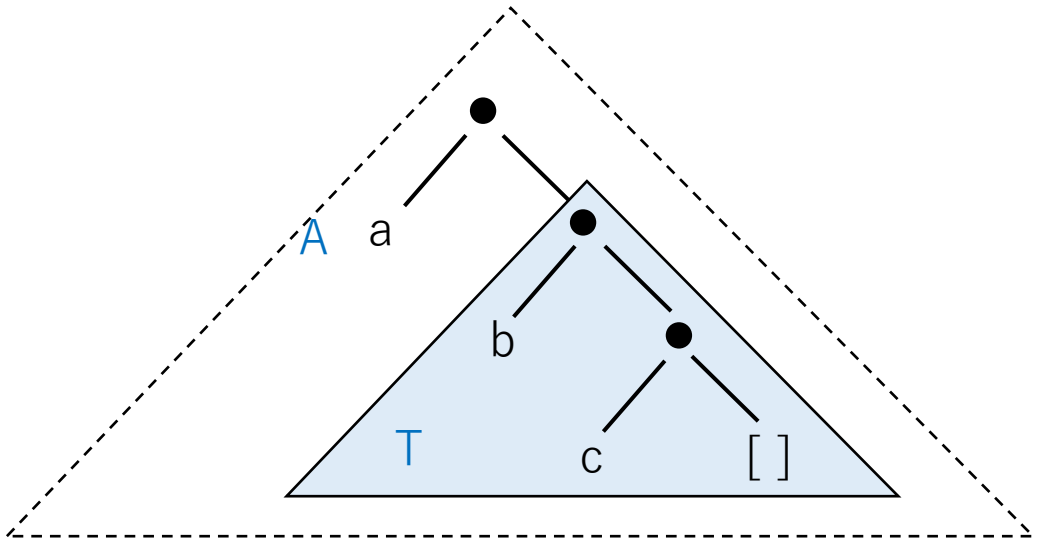
```
member(X,[X|T]).
member(X,[Y|T]) :- member(X,T)
```

(5) :- member(A,[a,b,c])の実行



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

member(A,[a,b,c])

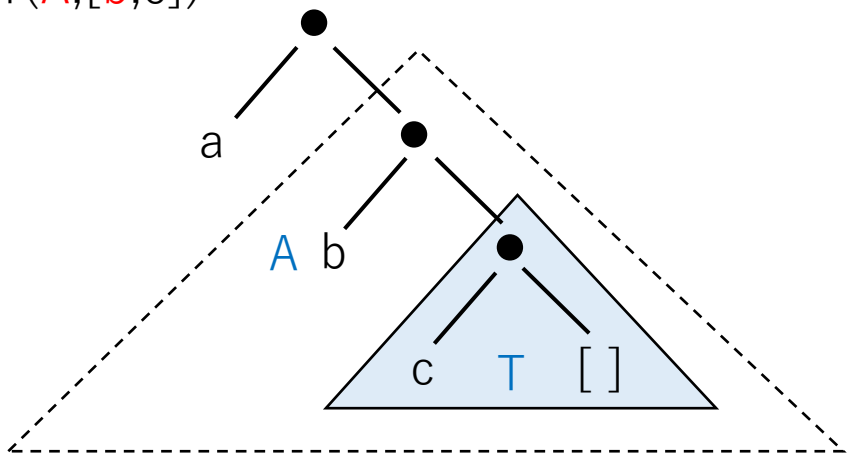


A=a となり成功して終了

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

強制的にバックトラックをかけると，member(A,[a,b,c]) が第2節と単一化を試みる

member(A,[b,c])

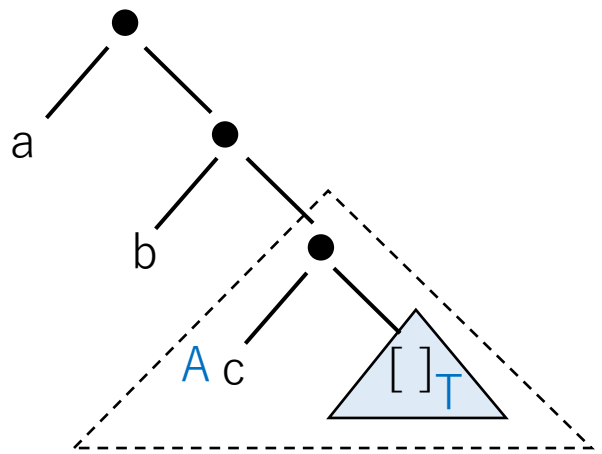


A=b となり成功して終了

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

強制的にバックトラックをかけると，member(A,[b,c]) が第2節と単一化を試みる

member(A,[c])



A=c となり成功して終了

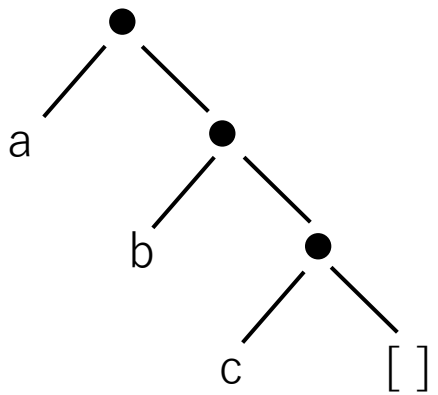
これ以外に解はない

リストのコピー

copy_list

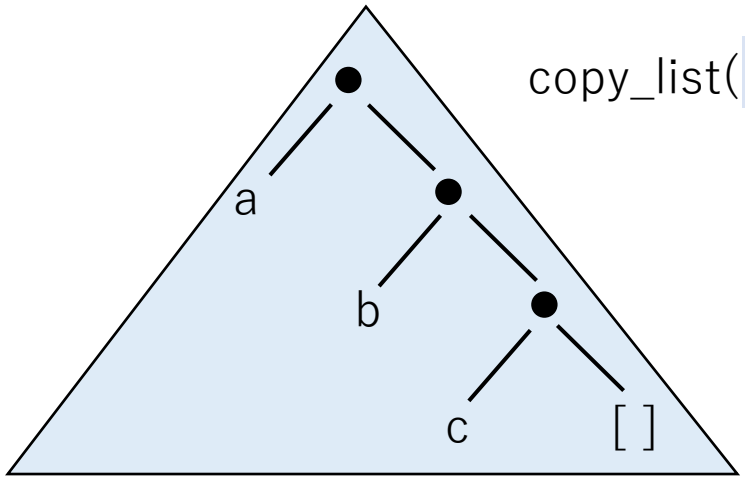
```
copy_list([],[]).
copy_list([X|X1],[X|Y1]) :- copy_list(X1,Y1)
```

copy_list([a,b,c],L)

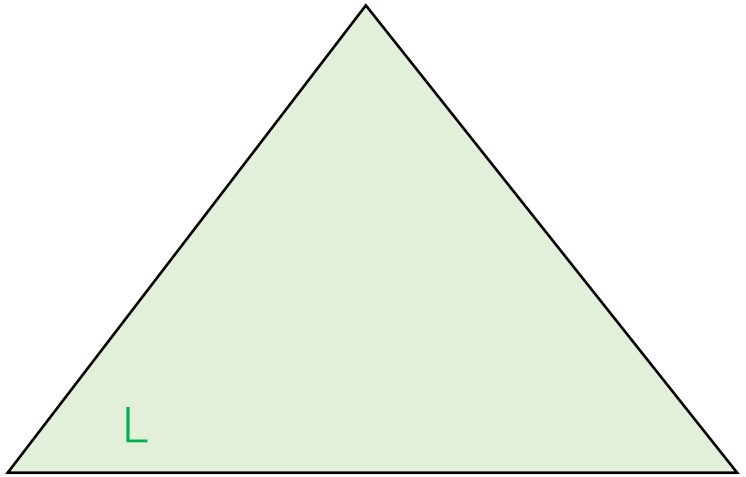


```
copy_list([],[]).
copy_list([X|X1],[X|Y1]) :- copy_list(X1,Y1)
```

copy_list([a,b,c],L)

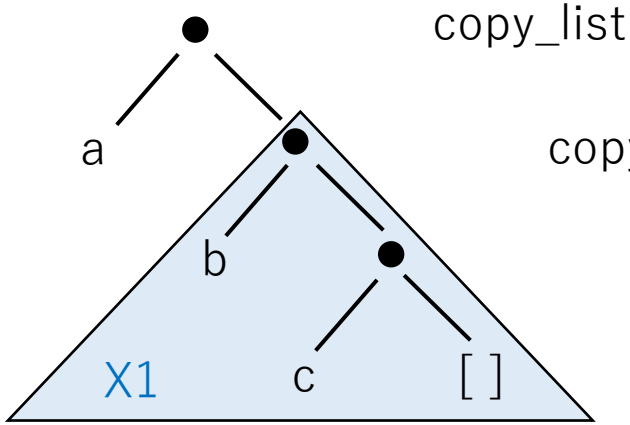


copy_list([a,b,c],L)



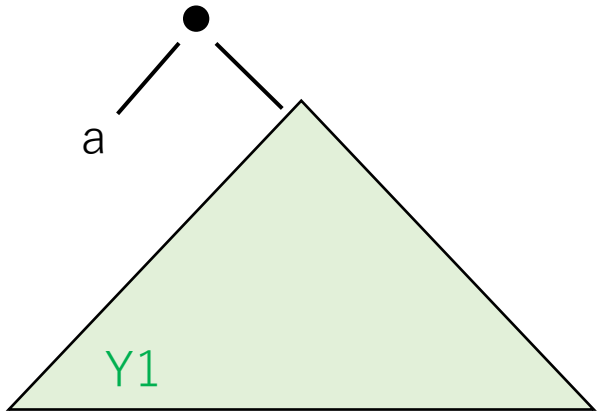
```
copy_list([],[]).
copy_list([X|X1],[X|Y1]) :- copy_list(X1,Y1)
           a       a
```

copy_list([a,b,c],L)



copy_list([a,b,c],L)

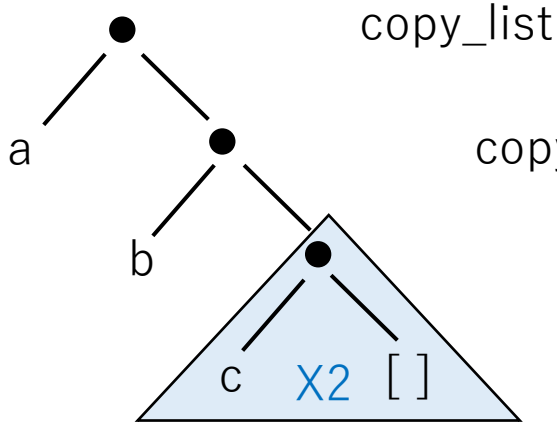
copy_list([b,c],L1)　L=[a|L1]



まず a をコピーし、残りは別の copy_list に委託する

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

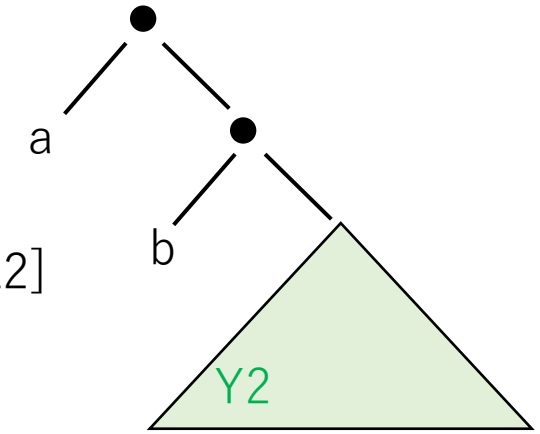
copy_list([a,b,c],L)



copy_list([a,b,c],L)

copy_list([b,c],L1)

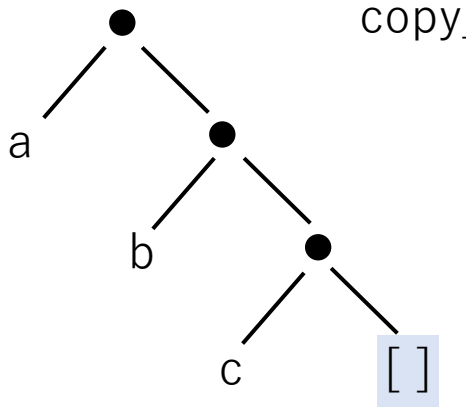
copy_list([c],L2)　L1=[b|L2]



次に b をコピーし、残りは別の copy_list に委託する

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

copy_list([a,b,c],L)

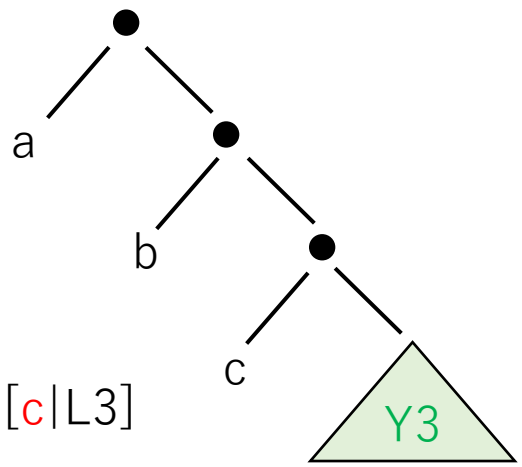


copy_list([a,b,c],L)

copy_list([b,c],L1)

copy_list([c],L2)

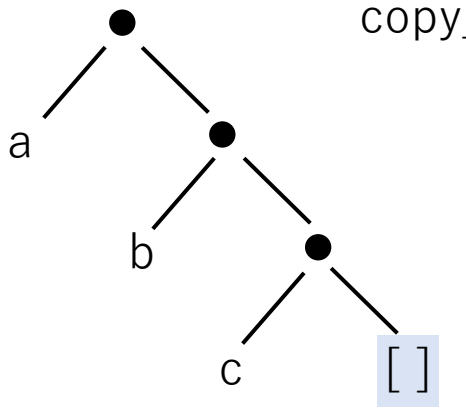
copy_list([ ],L3)　L2=[c|L3]



次に c をコピーし、残りは別の copy_list に委託する

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

copy_list([a,b,c],L)

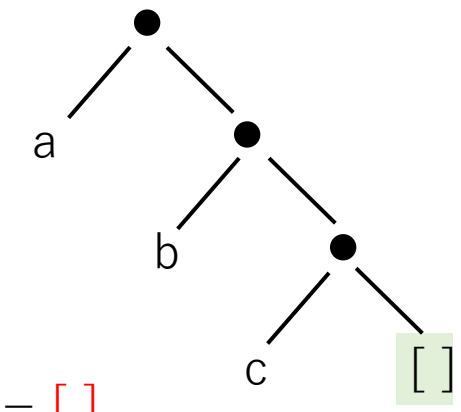


copy_list([a,b,c],L)

copy_list([b,c],L1)

copy_list([c],L2)

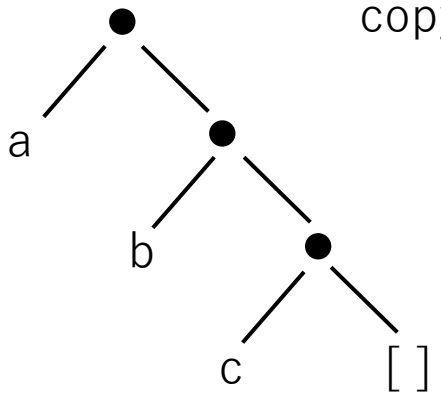
copy_list([ ],[ ])　L3 = [ ]



最後は [ ] を [ ] にコピーする

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

copy_list([a,b,c],L)

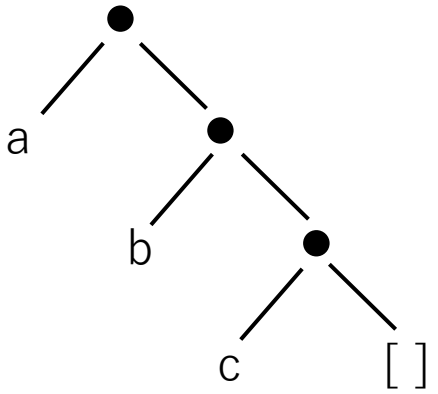


copy_list([a,b,c],L)　　L=[a|L1]

copy_list([b,c],L1)　L1=[b|L2]

copy_list([c],L2)　　L2=[c|L3]

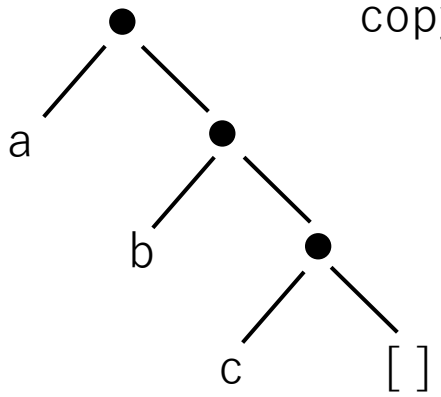
copy_list([ ],[ ])　　L3 = [ ]



L3 の値が決まるので L2 の値が決まる

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

copy_list([a,b,c],L)

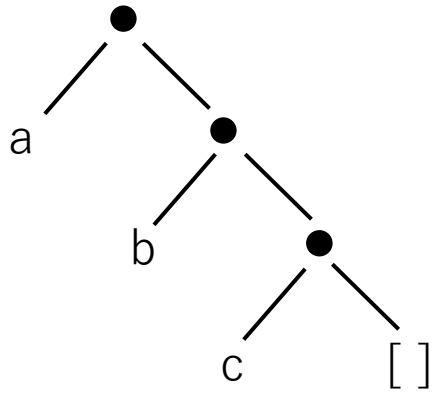


copy_list([a,b,c],L)　　L=[a|L1]

copy_list([b,c],L1)　L1=[b|L2]=[b,c]

copy_list([c],L2)　　L2=[c|L3]=[c]

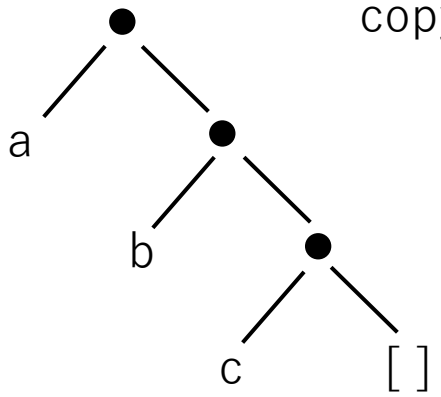
copy_list([ ],[ ])



L2 の値が決まるので L1 の値が決まる

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

copy_list([a,b,c],L)

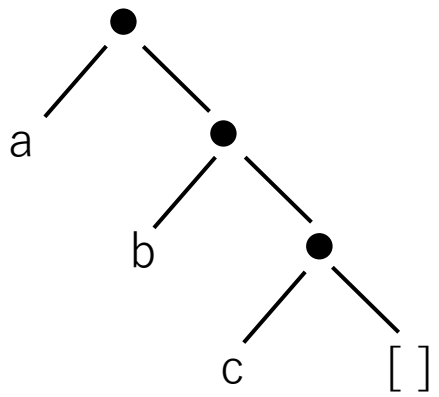


copy_list([a,b,c],L)　L=[a|L1] = [a,b,c]

copy_list([b,c],L1)　L1=[b|L2]=[b,c]

copy_list([c],L2)　L2=[c|L3]=[c]

copy_list([ ],[ ])



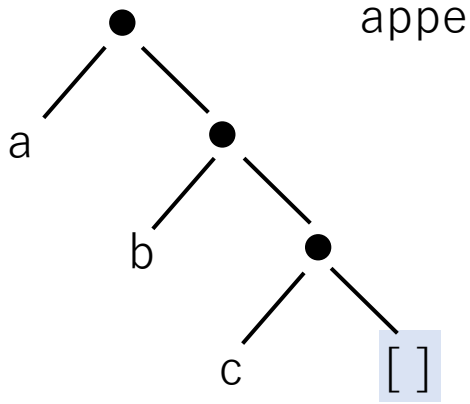
L1 の値が決まるので L の値が決まる

リストの連結

append

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

append([a,b,c],[d,e],L)

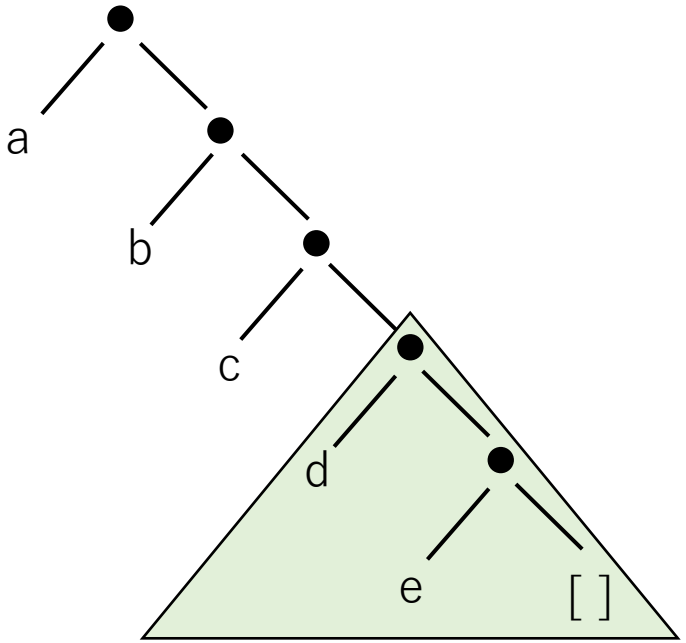


append([a,b,c],[d,e],L)

append([b,c],[d,e],L1)

append([c],[d,e],L2)

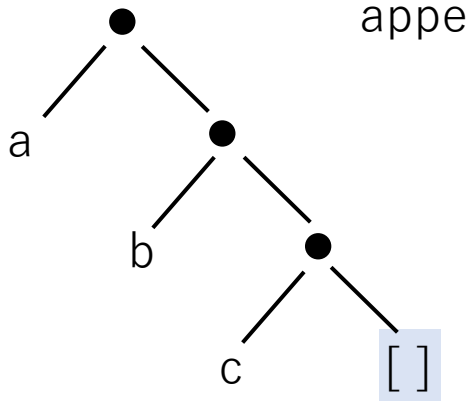
append([ ],[d,e], L3)



copy_list との違い：最後に [ ] をコピーせず，2つ目のリストをつなぐ

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

append([a,b,c],[d,e],L)

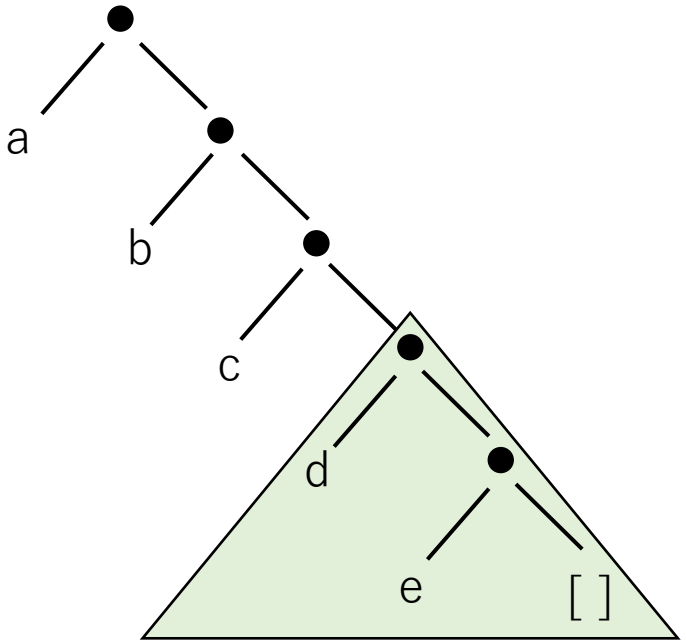


append([a,b,c],[d,e],L)　L=[a|L1] = [a,b,c,d,e]

append([b,c],[d,e],L1)　L1=[b|L2]=[b,c,d,e]

append([c],[d,e],L2)　L2=[c|L3]=[c,d,e]

append([ ],[d,e],[d,e])　L3=[d,e]



copy_list との違い：最後に [ ] をコピーせず，2つ目のリストをつなぐ