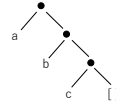


リストの要素のチェック member

1

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

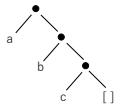
(2) :- member(b,[a,b,c])の実行
```



4

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

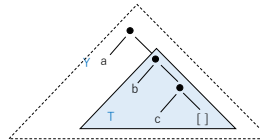
(1) :- member(a,[a,b,c])の実行
```



2

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

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

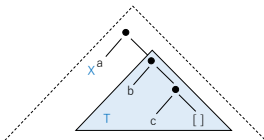


リストの第1要素が一致しないのでリストの残りを調べる

5

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

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

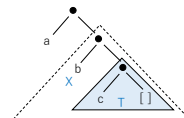


リストの第1要素が一致するので成功して終了

3

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

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

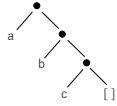


残りのリスト[b,c]の第1要素が一致するので成功して終了

6

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

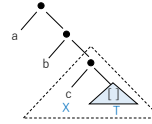
(3) :- member(c,[a,b,c])の実行



7

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

member(c,[c])

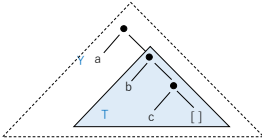


残りのリスト[c] の第1要素が一致するので成功して終了

10

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

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

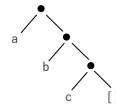


リストの第1要素が一致しないのでリストの残りを調べる

8

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

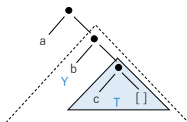
(4) :- member(d,[a,b,c])の実行



11

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

member(c,[b,c])

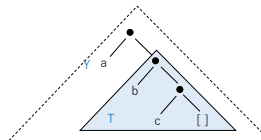


残りのリスト[b,c] の第1要素が一致しないのでリストの残りを調べる

9

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

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

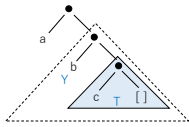


リストの第1要素が一致しないのでリストの残りを調べる

12

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

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

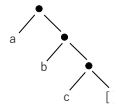


残りのリスト[b,c] の第1要素が一致しないのでリストの残りを調べる

13

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

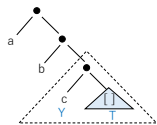
(5) :- member(A,[a,b,c]) の実行
```



16

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

member(d,[c])
```

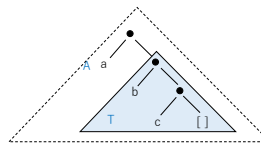


残りのリスト[c] の第1要素が一致しないのでリストの残りを調べる

14

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

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

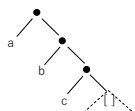


A=a となり成功して終了

17

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

member(d,[ ])
```



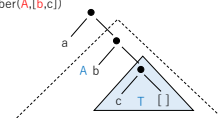
単一化できる節がないので失敗して終了

15

```
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])
```



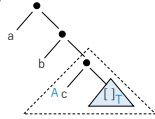
A=b となり成功して終了

18

```
member(X,[X|_]).
member(X,[_|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])

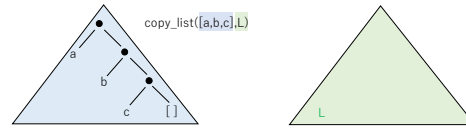


A=c となり成功して終了
これ以外に解はない

19

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

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



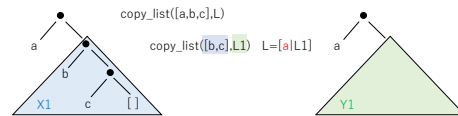
22

リストのコピー
copy_list

20

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

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

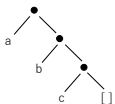


まず a をコピーし、残りは別の copy_list に委託する

23

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

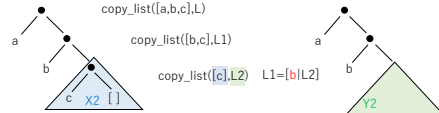
copy_list([a,b,c],L)



21

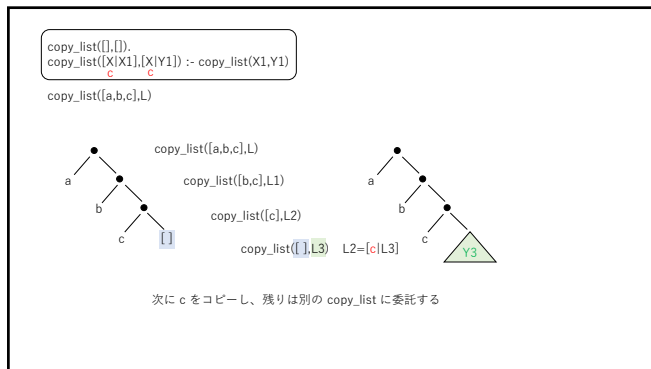
```
copy_list([],[]).
copy_list([X|X1],[X|Y1]) :- copy_list(X1,Y1)
```

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

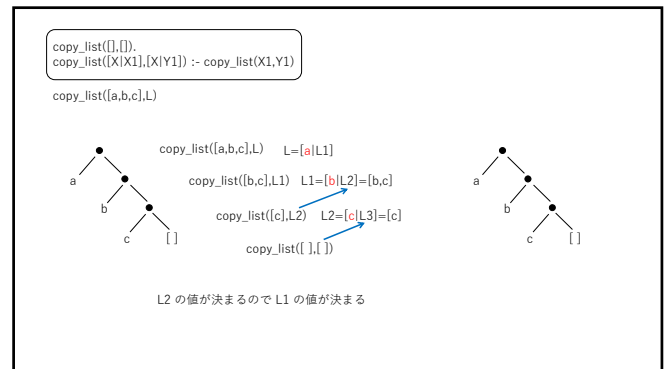


次に b をコピーし、残りは別の copy_list に委託する

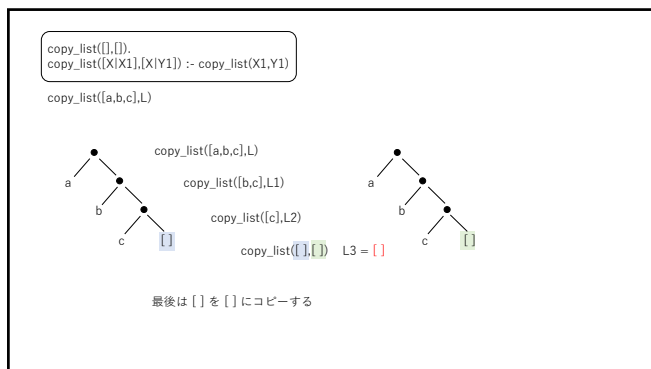
24



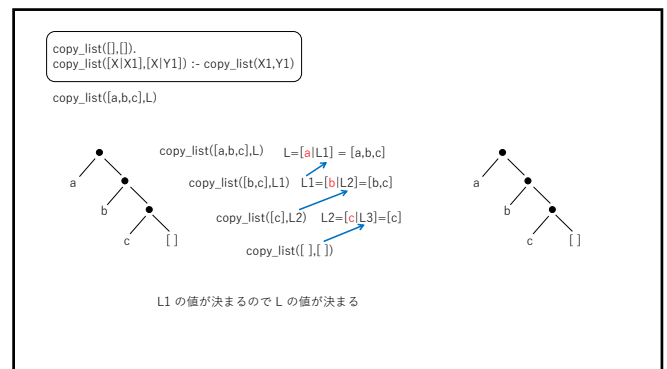
25



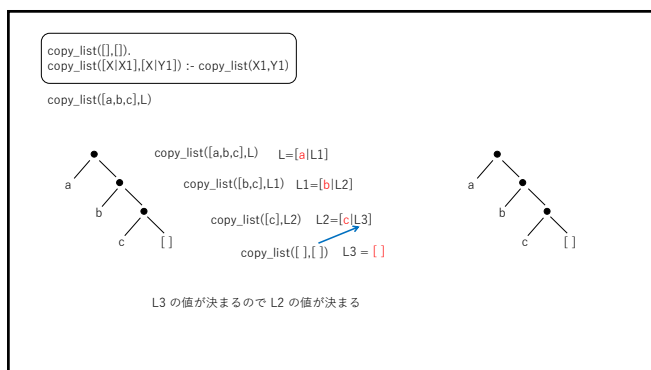
28



26



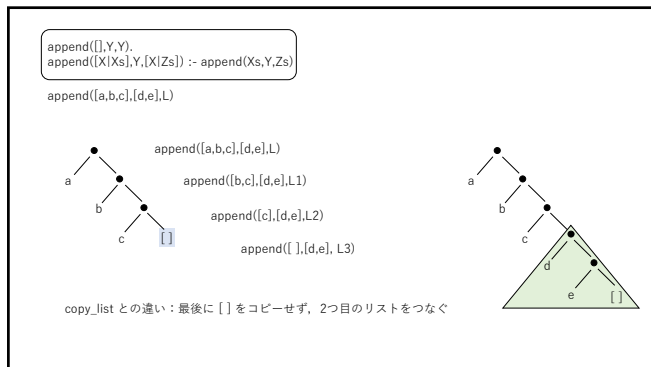
29



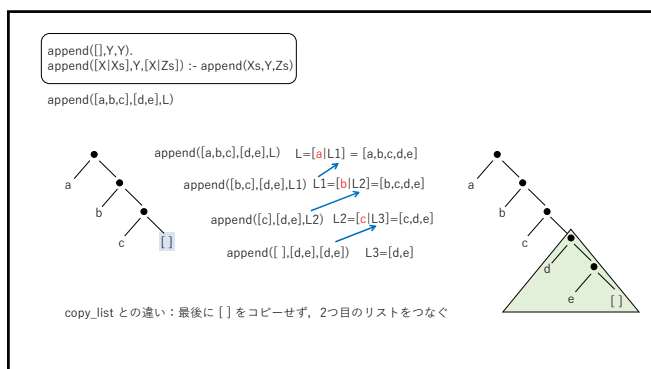
27



30



31



32