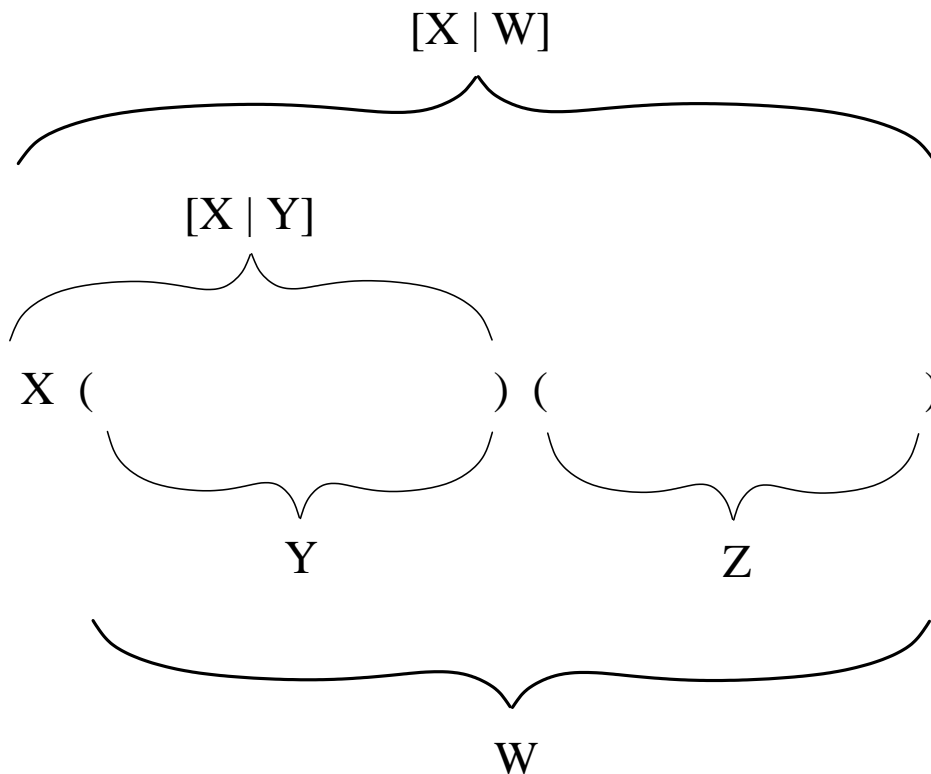


## Appending lists in SCHEME and in Prolog

```
(define append (lambda (X Y)
  (cond ((null? X) Y)
        (#t (cons (car X)
                    (append (cdr X) Y) )))))
```

[1] `append([], U, U)`.

[2] `append([X | Y], Z, [X | W]) :- append(Y, Z, W)`.



[1]  $\text{append}([], U, U)$ .

[2]  $\text{append}([X | Y], Z, [X | W]) \text{ :- } \text{append}(Y, Z, W)$ .

?-  $\text{append}([a,b], [c], L)$ . rule [2]

$X_1 \rightarrow a \quad Y_1 \rightarrow [b] \quad Z_1 \rightarrow [c] \quad L \rightarrow [a | W_1]$

$\text{append}([b], [c], W_1)$ . rule [2]

$X_2 \rightarrow b \quad Y_2 \rightarrow [] \quad Z_2 \rightarrow [c] \quad W_1 \rightarrow [b | W_2]$

$\text{append}([], [c], W_2)$ . rule [1]

$U_3 \rightarrow [c] \quad W_2 \rightarrow [c]$

<empty goal list> **SUCCESS**

What happened to L?

L was bound to  $[a | W_1]$

$W_1$  was bound to  $[b | W_2]$

$W_2$  was bound to  $[c]$

So L was bound to  $[a | [b | [c]]] = [a, b, c]$

## Input parameters may become output parameters

[1] `append([], U, U).`

[2] `append([X | Y], Z, [X | W]) :- append(Y, Z, W).`

?- `append(V, [b,c], [a,b,c]).` rule [2]

$V \rightarrow [a | Y_1] \quad Z_1 \rightarrow [b,c] \quad X_1 \rightarrow a \quad W_1 \rightarrow [b,c]$

`append(Y1, [b,c], [b,c]).` rule [1]

$Y_1 \rightarrow [] \quad U_2 \rightarrow [b,c]$

so  $V$  was bound to  $[a | Y_1]$

and  $Y_1$  was bound to  $[]$

so  $V$  was ultimately bound to  $[a | []] = [a]$

Exercise: Verify that the goal

`append(V, [b,c], [b,c,a]).`

will FAIL.

## More input parameters become output parameters

[1] `append([], U, U).`

[2] `append([X | Y], Z, [X | W]) :- append(Y, Z, W).`

?- `append(L, M, [a,b,c]).` rule [1]

$L \rightarrow [] \quad M = U_1 \rightarrow [a,b,c]$

After the “Yes” response type “;” for more answers:

?- `append(L, M, [a,b,c]).` use rule [2] this time

$L \rightarrow [a | Y_1] \quad M \rightarrow Z_1 \quad X_1 \rightarrow a \quad W_1 \rightarrow [b,c]$

`append(Y1, Z1, [b,c]).` rule [1]

$Y_1 \rightarrow [] \quad Z_1 = U_2 \rightarrow [b,c]$

so in the second answer

L is bound to `[a | Y1]` and Y<sub>1</sub> is bound to `[]`

M and Z<sub>1</sub> are bound to `[b,c]`

thus we have  $L = [a | []] = [a]$ , &  $M = [b,c]$

What are the other answers that Prolog will find?