Proving identities about programs

Four aids for doing this

(1) Function definition

(2) Previously proved results

(3) Principle of induction on lists

Suppose $P(l)$ is a property about lists

$$\left.\begin{array}{l} P(()) \text{ and} \\ P(l) \Rightarrow P((\text{cons } x\ l)) \end{array}\right\} \Rightarrow \forall l\, P(l)$$

1

Prove $\forall l_1 l_2.$ (length (append l1 l2)) = (+ (length l1) (length l2))

Consider an arbitrary l2

$P(l_1)$ = (length (append l1 l2)) = (+ (length l1) (length l2))

We want to prove $\forall l_1\, P(l_1)$

Base case $P(())$

LHS = (length (append () l2))

= (length l2)

RHS = (+ (length ()) (length l2))

= (+ 0 (length l2))

= length l2

Ind. Hyp: (length (append l l2)) = (+ (length l) (length l2))

Ind. step / To show: (length (append (cons x l) l2)) =

(+ (length (cons x l)) (length l2))

LHS = (length (cons x (append l l2)))

= (+ 1 (length (append l l2)))

= (+ 1 (+ (length l) (length l2)))

= (+ (+ 1 (length l)) (length l2))

= (+ (length (cons x l)) (length l2))

= RHS

$\forall l_1 \forall l_2$: (length (append l1 l2)) = (+ (length l1) (length l2))

---

2 Prove $\forall l$ (append l ()) = l

Base case: Show (append () ()) = ()

Ind. hyp: Assume (append l1 '()) = l1

Show (append (cons x l1) ()) = (cons x l1)

LHS = (append (cons x l1) ()) {Defn. append}

= (cons x (append l1 ())) {Ind. hyp}

= (cons x l1)

= RHS

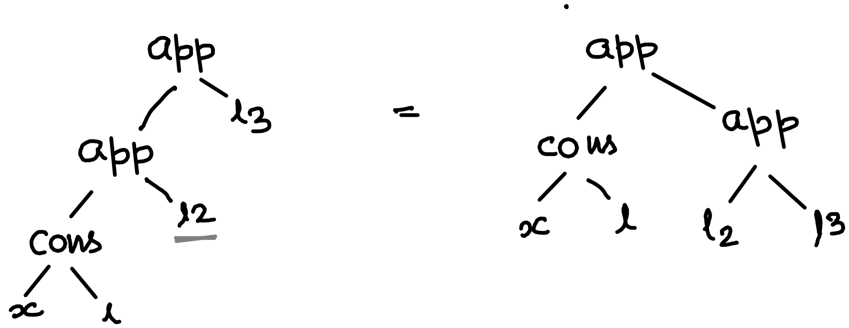
---

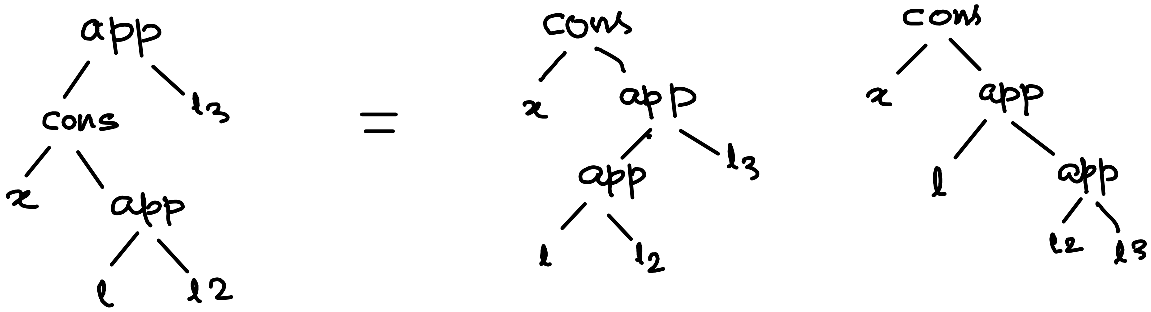
3 Prove (append (append l1 l2) l3) = (append l1 (append l2 l3))

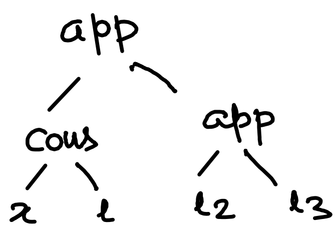
(i) Base case – (append (append () l2) l3) = (append () (append l2 l3))

(ii) Assume (append (append l l2) l3) = (append l (append l2 l3))

Show (append (append (cons x l) l2) l3) = (append (cons x l) (append l2 l3))







(4)

<u>Show</u> $\forall l$ (rev (rev l)) = l

Induction on l.

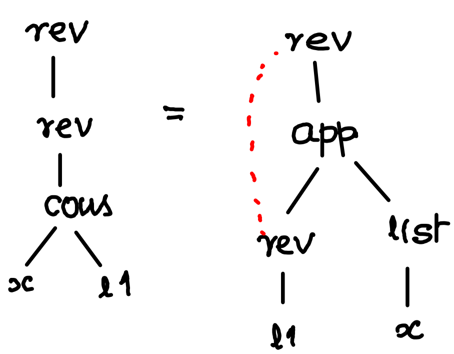
<u>Base</u> (rev (rev ())) = ()

LHS = (rev ())

= ()

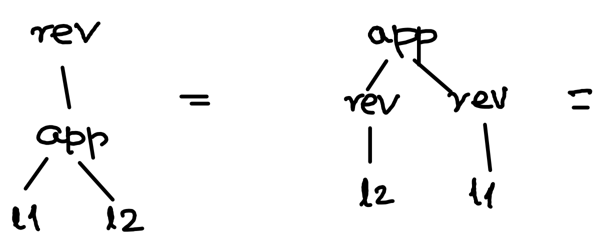
<u>Assume</u> (rev (rev l1)) = l1

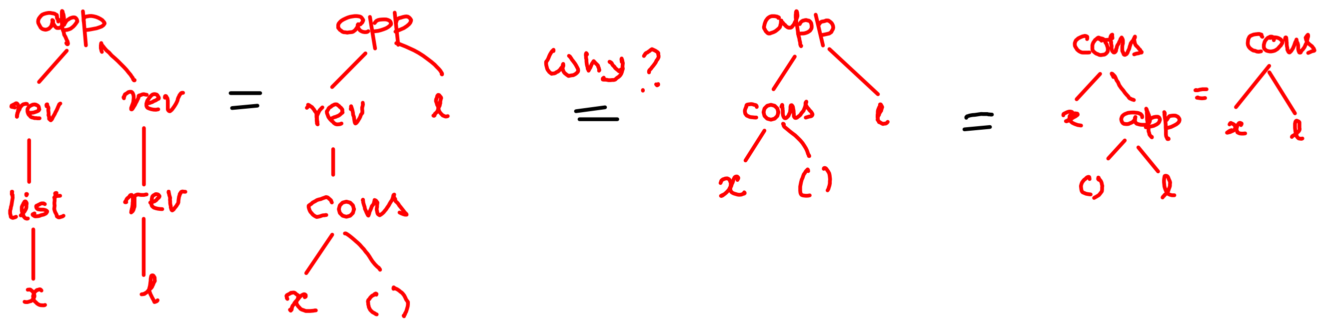
<u>Show</u> (rev (rev (cons x l1))) = (cons x l1)



Bringing the two revs together will allow us to apply the induction hypothesis:

<u>Conjecture</u>





(5) Show rev (app l1 l2) = (app (rev l2) (rev l1))

Induction on l1

Base : rev (app () l2)

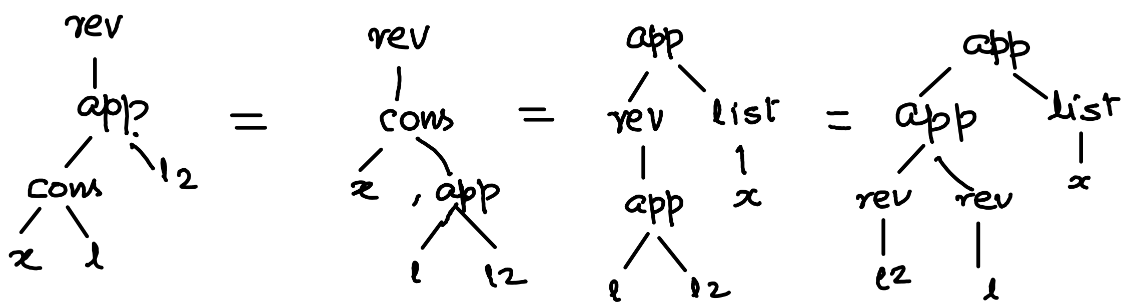
LHS = rev (app l2)

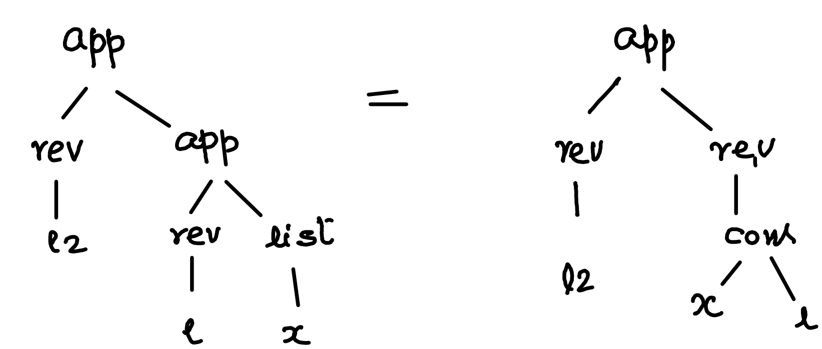
RHS = (app (rev l2) (rev l1))

= (app (rev l2) ())

= (rev l2) (Previous result)

Assume - (rev (app l l2)) = (app (rev l2) (rev l))





⑥ **Prove** map f (append x y) = (append (map f x)
(map f y))

Induct on x

Base case . map f (append () y)

= map f y =

= (append () (map f y))

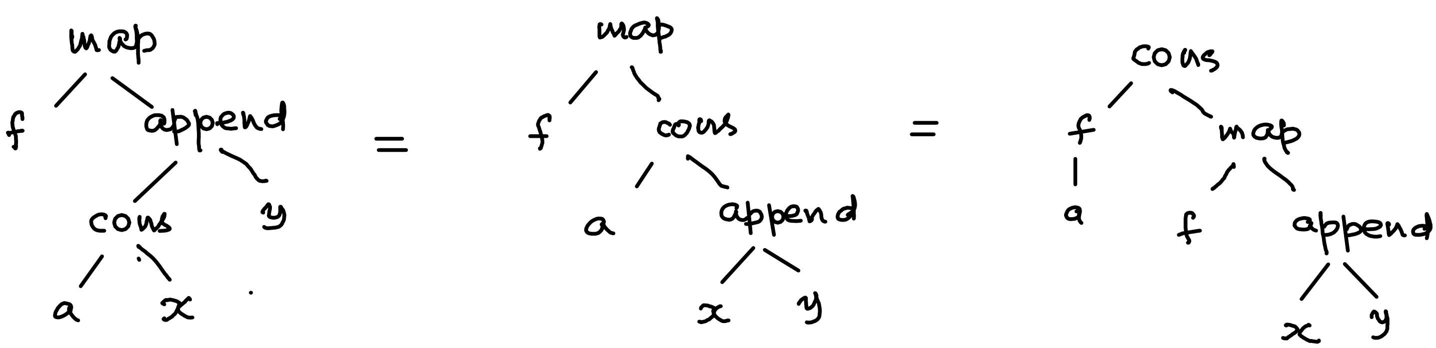
= (append (map f ()) (map f y))

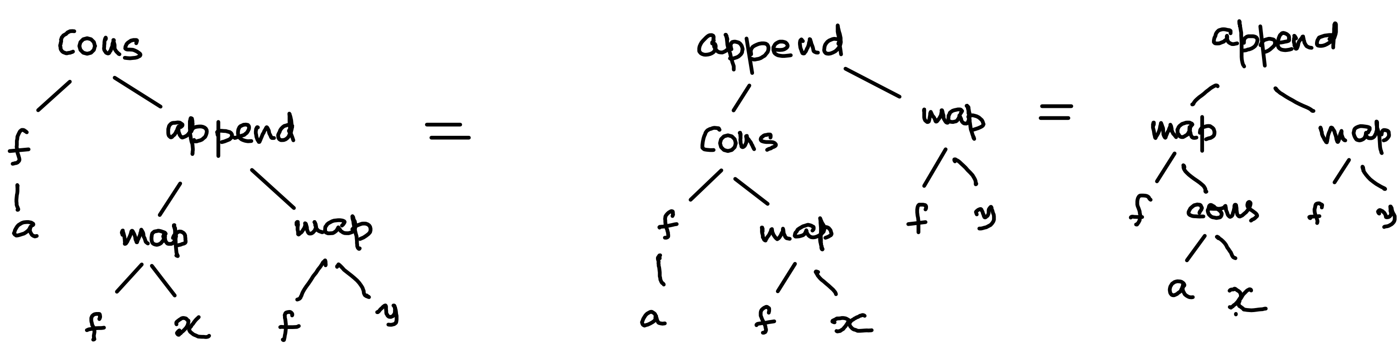
Assume (map f (append x y))

= (append (map f x) (map f y)) .

·To prove (map f (append (cons a x) y))

= (append (map f (cons a x)) (map f y))



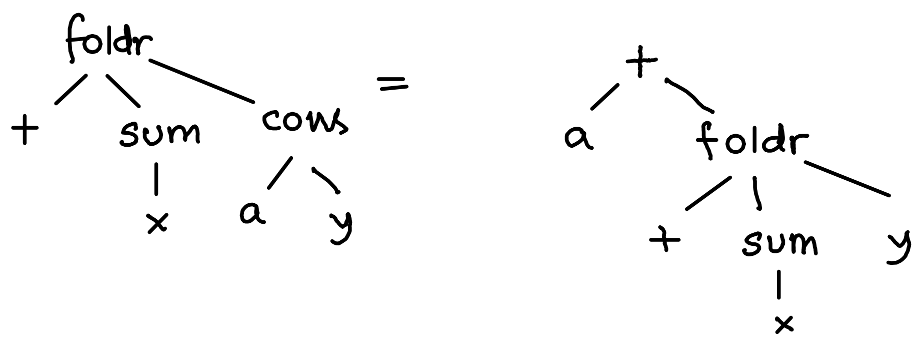


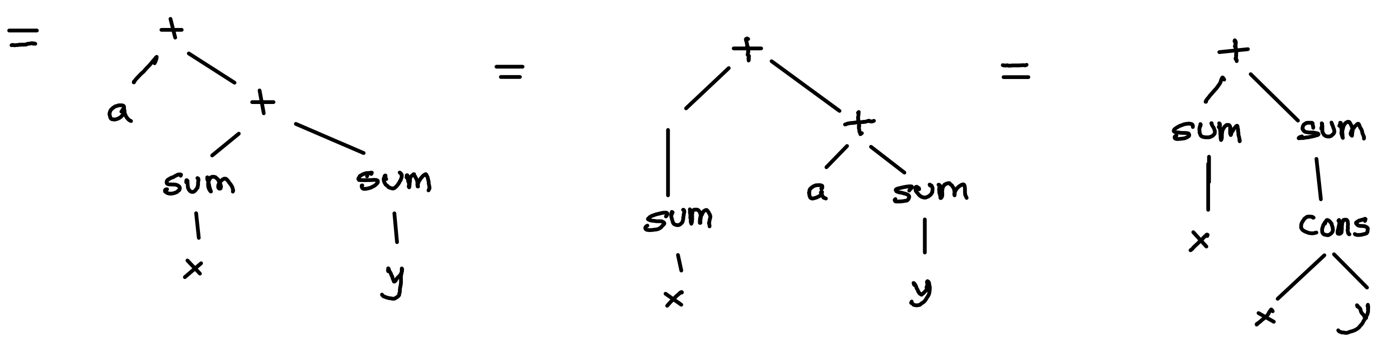
7

(foldr (+) (sum x) y) = (+ (sum x) (sum y))

Base case: easy

Assume (foldr + (sum x) y) = (+ (sum x) (sum y))

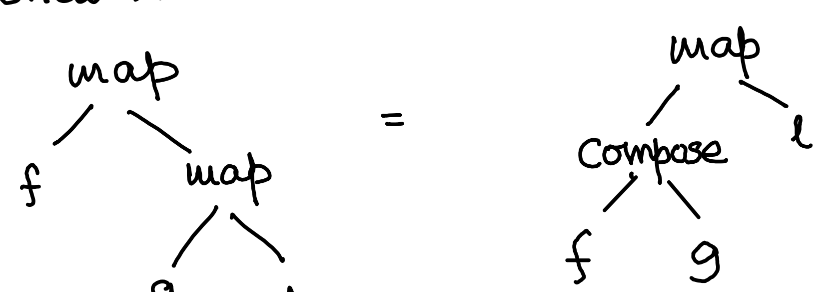




8

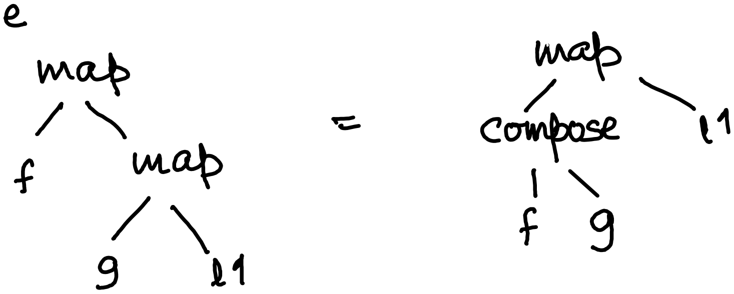
(define (compose f g) (lambda (x) (f (g x))))

Show that

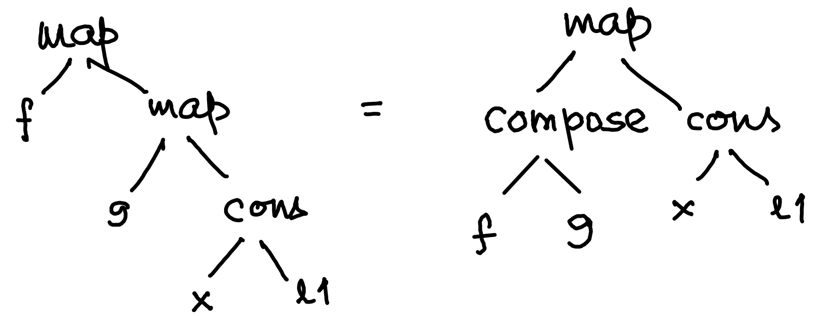


Base case — Easy

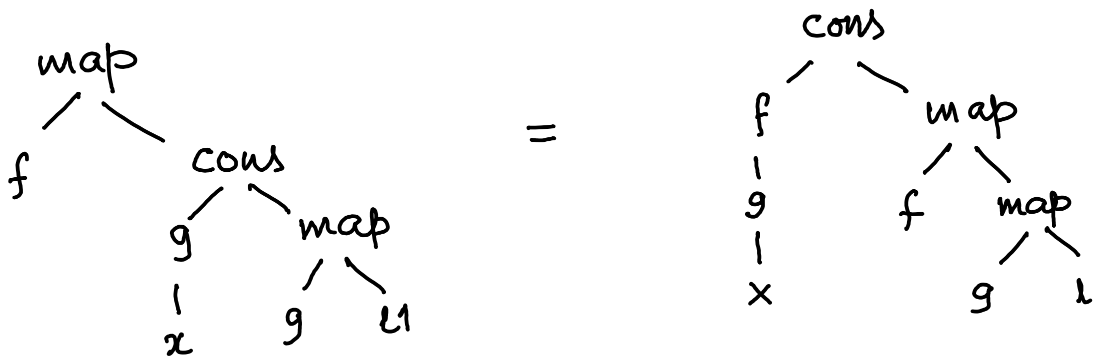
Assume

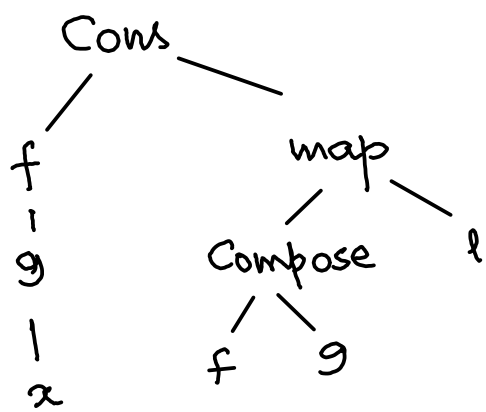


Show

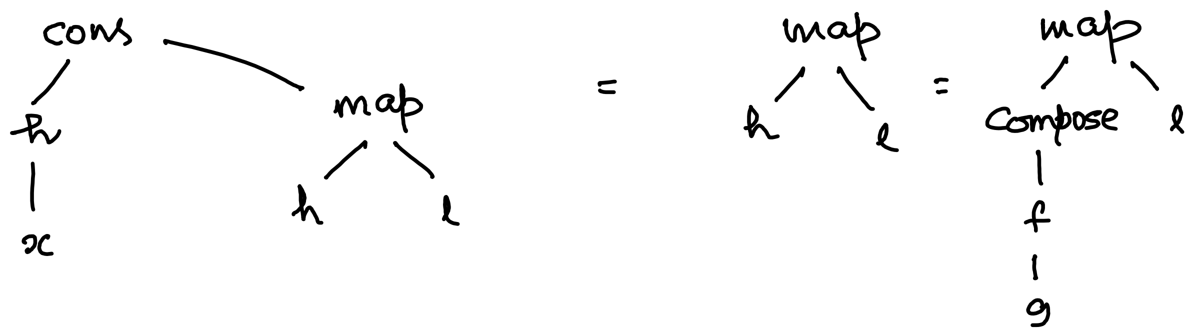


LHS =





let h be (compose f g)



9

```
(define (mf l) (map f l))
(define (append* l)
    (if (null? l)   '()
        (append  (car l)
                 (append* (cdr l) ))))
```

Show that

(map f (append* l)) =

(append* (map mf l))

Base case: Easy

Assume:

Prove: