

## Unlifting of $M$ at $\sigma$

```
define
  equal(x,y) = case x of
    Nil -> case y of
      Nil -> False
      Cons(a,b) -> False
    Cons(x',y') -> case y of
      Nil -> False
      Cons(a,b) -> case equal_nat(x',a) of
        True -> equal(y',b)
        False -> False
  equal_nat(x,y) = case x of
    Zero -> case y of
      Zero -> True
      Succ(a) -> False
    Succ(x') -> case y of
      Zero -> True
      Succ(a) -> equal_nat(x',a)
in
  equal(Nil,Cons(Zero,Nil))
```

## Interactive term $M$

```
define
  equal(x,y) = case x of
    Nil -> case y of
      Nil -> val v3
      Cons(a,b) -> val v2
    Cons(x',y') -> case y of
      Nil -> val v1
      Cons(a,b) -> case equal_nat(x',a) of
        True -> equal(y',b)
        False -> val v0
  equal_nat(x,y) = case x of
    Zero -> case y of
      Zero -> val v6
      Succ(a) -> val v5
    Succ(x') -> case y of
      Zero -> val v4
      Succ(a) -> equal_nat(x',a)
in
  equal(val v7,val v10)
```

## Value store $\sigma$ for State 1

v0:False  
v1:False  
v2:False  
v3:False  
v4:True

v5:False  
v6:True  
v7:Nil  
v8:Zero  
v9:Nil

v10:Cons(v8:Zero,v9:Nil)

## Unlifting of $M$ at $\sigma$

```
define
  equal(x,y) = case x of
    Nil -> case y of
      Nil -> False
      Cons(a,b) -> False
    Cons(x',y') -> case y of
      Nil -> False
      Cons(a,b) -> case equal_nat(x',a) of
        True -> equal(y',b)
        False -> False
  equal_nat(x,y) = case x of
    Zero -> case y of
      Zero -> True
      Succ(a) -> False
    Succ(x') -> case y of
      Zero -> True
      Succ(a) -> equal_nat(x',a)
in
  equal(Cons(Zero,Nil),Cons(Zero,Nil))
```

## Interactive term $M$

```
define
  equal(x,y) = case x of
    Nil -> case y of
      Nil -> val v3
      Cons(a,b) -> val v2
    Cons(x',y') -> case y of
      Nil -> val v1
      Cons(a,b) -> case equal_nat(x',a) of
        True -> equal(y',b)
        False -> val v0
  equal_nat(x,y) = case x of
    Zero -> case y of
      Zero -> val v6
      Succ(a) -> val v5
    Succ(x') -> case y of
      Zero -> val v4
      Succ(a) -> equal_nat(x',a)
in
  equal(val v7,val v10)
```

## Value store $\sigma$ for State 2

v0:False  
v1:False  
v2:False  
v3:False  
v4:True

v5:False  
v6:True  
v7:Cons(v13:Zero,v14:Nil)  
v8:Zero  
v9:Nil

v10:Cons(v8:Zero,v9:Nil)  
v13:Zero  
v14:Nil

## Unlifting of $M$ at $\sigma$

```
define
  equal(x,y) = case x of
    Nil -> case y of
      Nil -> True
      Cons(a,b) -> False
    Cons(x',y') -> case y of
      Nil -> False
      Cons(a,b) -> case equal_nat(x',a) of
        True -> equal(y',b)
        False -> False
  equal_nat(x,y) = case x of
    Zero -> case y of
      Zero -> True
      Succ(a) -> False
    Succ(x') -> case y of
      Zero -> True
      Succ(a) -> equal_nat(x',a)
in
  equal(Cons(Zero,Nil),Cons(Zero,Nil))
```

## Interactive term $M$

```
define
  equal(x,y) = case x of
    Nil -> case y of
      Nil -> val v3
      Cons(a,b) -> val v2
    Cons(x',y') -> case y of
      Nil -> val v1
      Cons(a,b) -> case equal_nat(x',a) of
        True -> equal(y',b)
        False -> val v0
  equal_nat(x,y) = case x of
    Zero -> case y of
      Zero -> val v6
      Succ(a) -> val v5
    Succ(x') -> case y of
      Zero -> val v4
      Succ(a) -> equal_nat(x',a)
in
  equal(val v7,val v10)
```

## Value store $\sigma$ for State 3

v0:False  
v1:False  
v2:False  
v3:True  
v4:True

v5:False  
v6:True  
v7:Cons(v13:Zero,v14:Nil)  
v8:Zero  
v9:Nil

v10:Cons(v8:Zero,v9:Nil)  
v13:Zero  
v14:Nil

## Unlifting of $M$ at $\sigma$

```
define
  equal(x,y) = case x of
    Nil -> case y of
      Nil -> True
      Cons(a,b) -> False
    Cons(x',y') -> case y of
      Nil -> False
      Cons(a,b) -> case equal_nat(x',a) of
        True -> equal(y',b)
        False -> False
  equal_nat(x,y) = case x of
    Zero -> case y of
      Zero -> True
      Succ(a) -> False
    Succ(x') -> case y of
      Zero -> True
      Succ(a) -> equal_nat(x',a)
in
  equal(Cons(Succ(Zero),Nil),Cons(Zero,Nil))
```

## Interactive term $M$

```
define
  equal(x,y) = case x of
    Nil -> case y of
      Nil -> val v3
      Cons(a,b) -> val v2
    Cons(x',y') -> case y of
      Nil -> val v1
      Cons(a,b) -> case equal_nat(x',a) of
        True -> equal(y',b)
        False -> val v0
  equal_nat(x,y) = case x of
    Zero -> case y of
      Zero -> val v6
      Succ(a) -> val v5
    Succ(x') -> case y of
      Zero -> val v4
      Succ(a) -> equal_nat(x',a)
in
  equal(val v7,val v10)
```

## Value store $\sigma$ for State 4

v0:False  
v1:False  
v2:False  
v3:True  
v4:True

v5:False  
v6:True  
v7:Cons(v13:Zero,v14:Nil)  
v8:Zero  
v9:Nil

v10:Cons(v8:Zero,v9:Nil)  
v13:Succ(v21:Zero)  
v14:Nil  
v21:Zero

## Unlifting of $M$ at $\sigma$

```
define
  equal(x,y) = case x of
    Nil -> case y of
      Nil -> True
      Cons(a,b) -> False
    Cons(x',y') -> case y of
      Nil -> False
      Cons(a,b) -> case equal_nat(x',a) of
        True -> equal(y',b)
        False -> False
  equal_nat(x,y) = case x of
    Zero -> case y of
      Zero -> True
      Succ(a) -> False
    Succ(x') -> case y of
      Zero -> False
      Succ(a) -> equal_nat(x',a)
in
  equal(Cons(Succ(Zero),Nil),Cons(Zero,Nil))
```

## Interactive term $M$

```
define
  equal(x,y) = case x of
    Nil -> case y of
      Nil -> val v3
      Cons(a,b) -> val v2
    Cons(x',y') -> case y of
      Nil -> val v1
      Cons(a,b) -> case equal_nat(x',a) of
        True -> equal(y',b)
        False -> val v0
  equal_nat(x,y) = case x of
    Zero -> case y of
      Zero -> val v6
      Succ(a) -> val v5
    Succ(x') -> case y of
      Zero -> val v4
      Succ(a) -> equal_nat(x',a)
in
  equal(val v7,val v10)
```

## Value store $\sigma$ for State 5

v0:False  
v1:False  
v2:False  
v3:True  
v4:False

v5:False  
v6:True  
v7:Cons(v13:Zero,v14:Nil)  
v8:Zero  
v9:Nil

v10:Cons(v8:Zero,v9:Nil)  
v13:Succ(v21:Zero)  
v14:Nil  
v21:Zero