

State 1

```
equal(Nil, Cons(Zero, Nil)) ... False
```

State 1

```
equal
x: Nil
y: Cons(Zero,Nil)
= case x of
  Nil ->
    case y of
      Nil -> •
      Cons(a: Zero,b: Nil) -> False
      Cons(x'::,y':) -> •
```

False

State 2

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

False

State 2

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

False

State 3

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

True

State 4

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

True

State 4

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

State 4

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

True

True of

State 5

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

False

State 5

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

False