The Scope of Algebraic Effects

Nicolas Wu
with Maciej Pirog, Tom Schrijvers, and Mauro Jaskelioff

University of Bristol

Shonan Seminar 146
26th March 2019

Effect Handlers

Effect Handlers

Syntax

Syntax represented by the **free monad** for a functor that provides a signature

Semantics

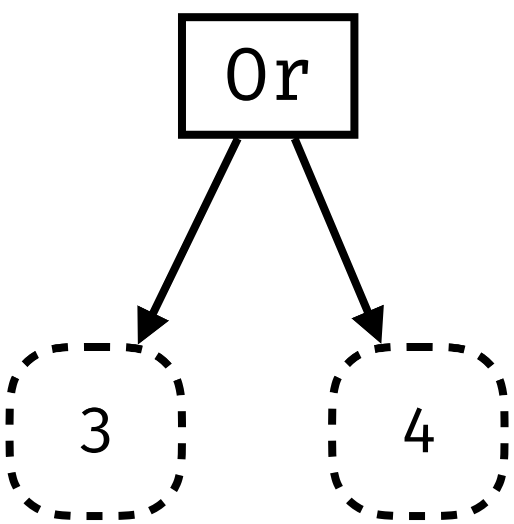
Semantics often in terms of a **fold** over the free monad

Effect Handlers

Syntax

```
data Free f a
  = Var  a
  | Op (f (Free f a))
```

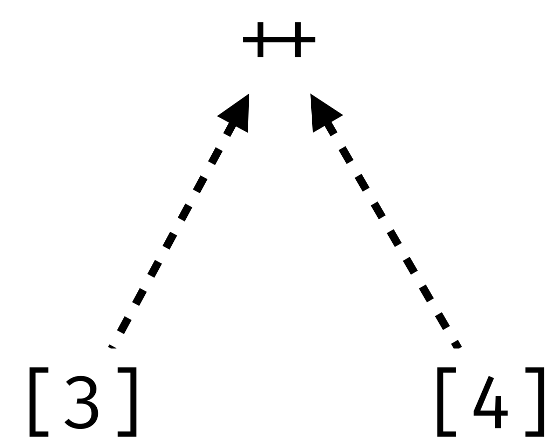
```
data Or k = Or k k
```




```
Op (Or (Var 3) (Var 4))
```

Semantics

```
type Alg f a = f a → a

eval :: Functor f ⇒
  (a → b) → Alg f b →
  Free f a → b
eval gen alg (Var x) = gen x
eval gen alg (Op op) =
  (alg . fmap (eval gen alg)) op
```



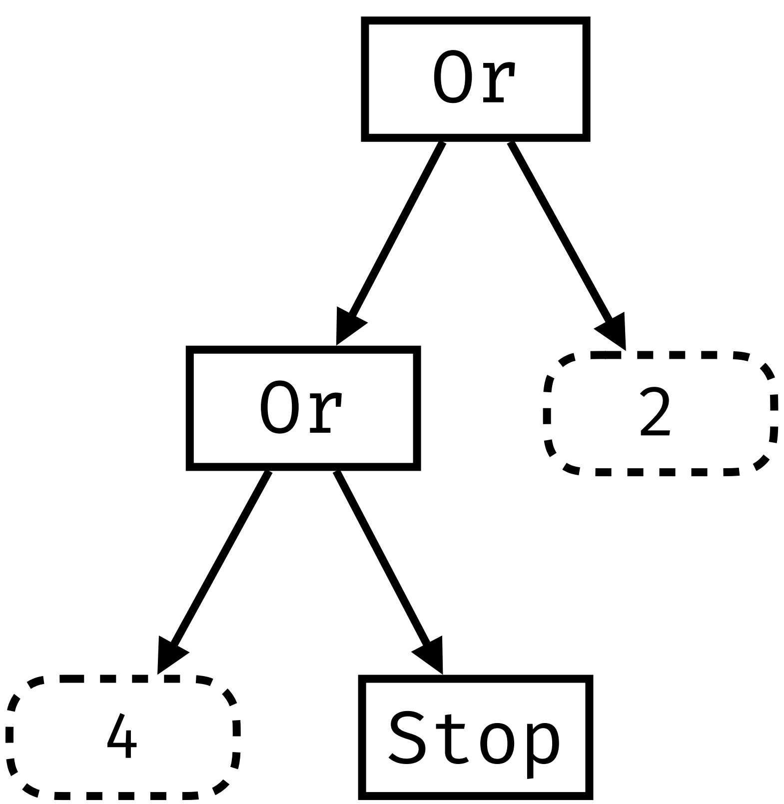

```
[3, 4]
```

Syntax

Extra language features can be added as needed

```
data Or k = Or k k      data Stop k = Stop      data Void k
```

```
data (f :+ sig) a = Eff (f a) | Sig (sig a)
```




```
:: Free (Or :+ Stop :+ Void) Int
≃ Free (Stop :+ Or :+ Void) Int
```

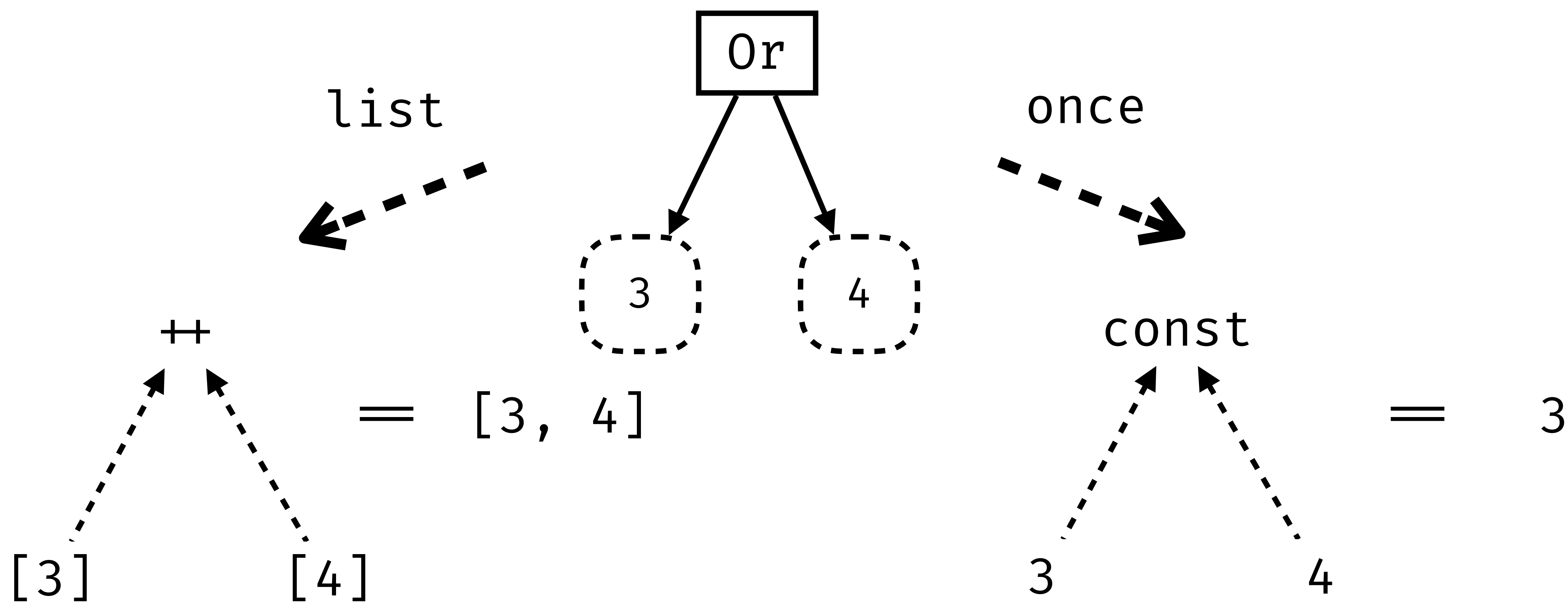
Semantics

Semantics target individual operations

```
data Or k = Or k k
```

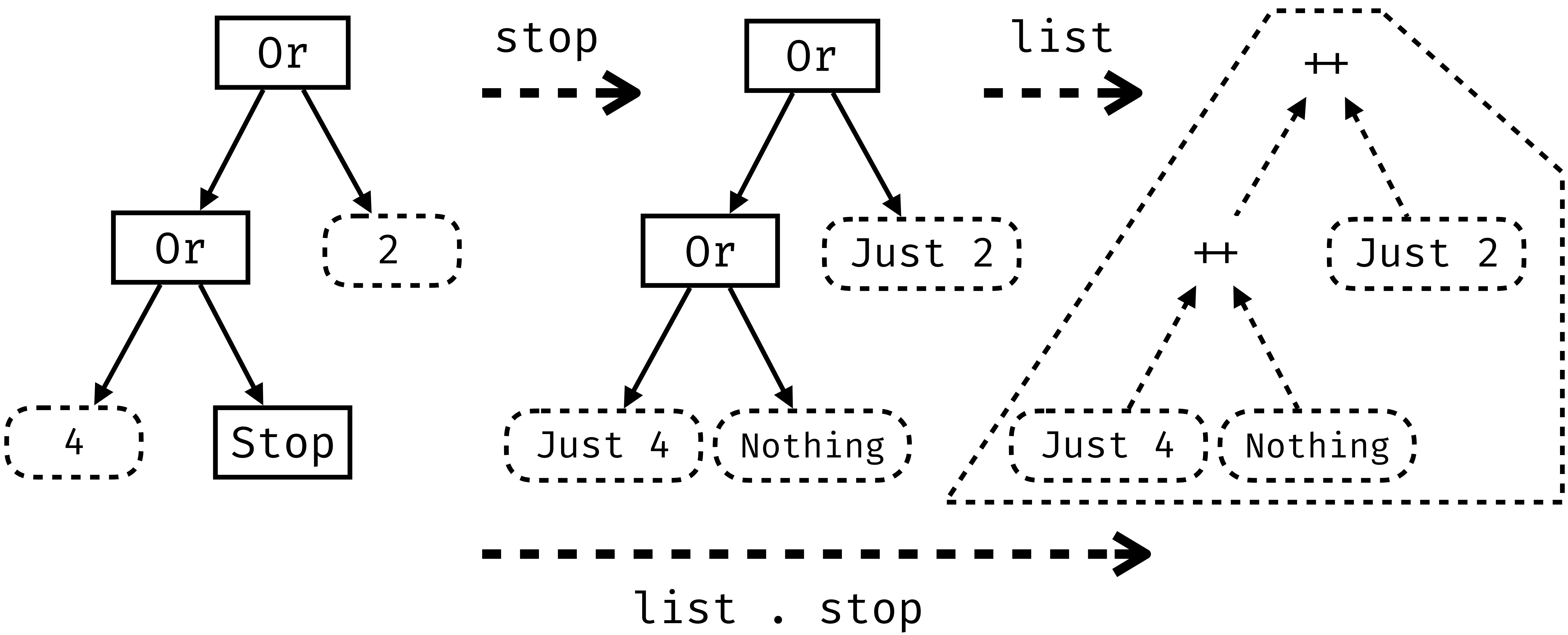
```
list :: Free Or a → [a]
list = eval gen alg where
  gen x = [x]
  alg (Or xs ys) = xs ++ ys
```

```
once :: Free Or a → a
once = eval gen alg where
  gen x = x
  alg (Or xs ys) = const xs ys
                 = xs
```



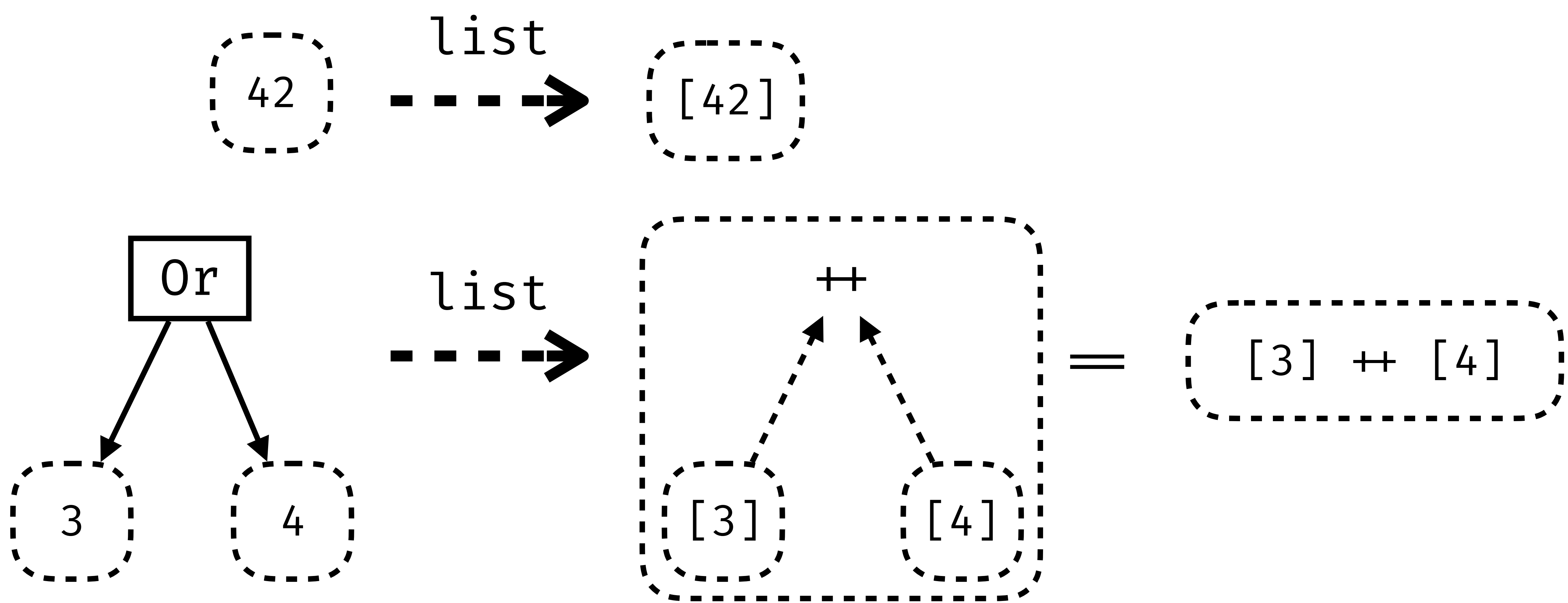
Chained Handlers

Chained Handlers



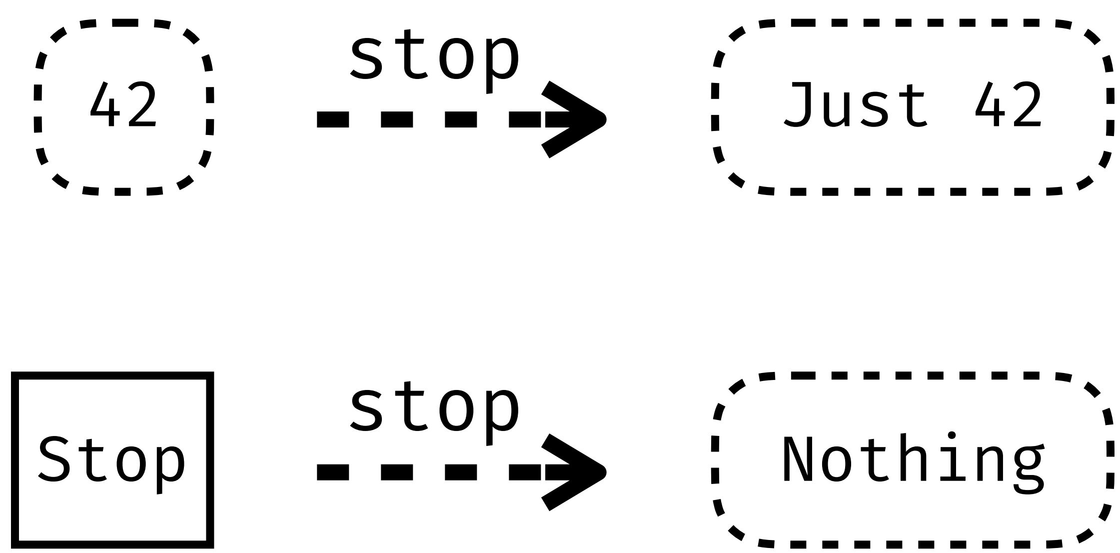
Nondeterminism

```
data Or k = Or k k

list :: Functor f ⇒ Free (Or :+ f) a → Free f [a]
list = eval gen (embed alg) where
  gen x = Var [x]
  alg (Or mx my) = do xs ← mx
                      ys ← my
                      Var (xs ++ ys)
```



Exceptions

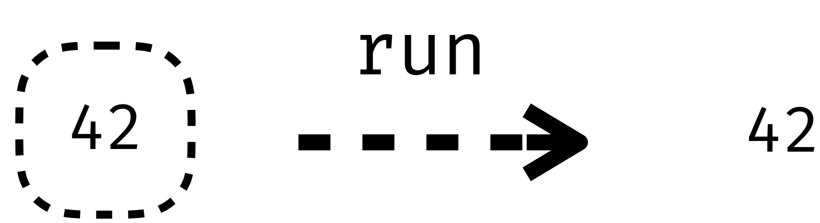
```
data Stop k = Stop

stop :: Functor f ⇒ Free (Stop :+ f) a → Free f (Maybe a)
stop = eval gen (embed alg) where
  gen x = Var (Just x)
  alg :: Alg (Stop) (Free f (Maybe a))
  alg Stop = Var (Nothing)
```



Void

```
data Void k
```

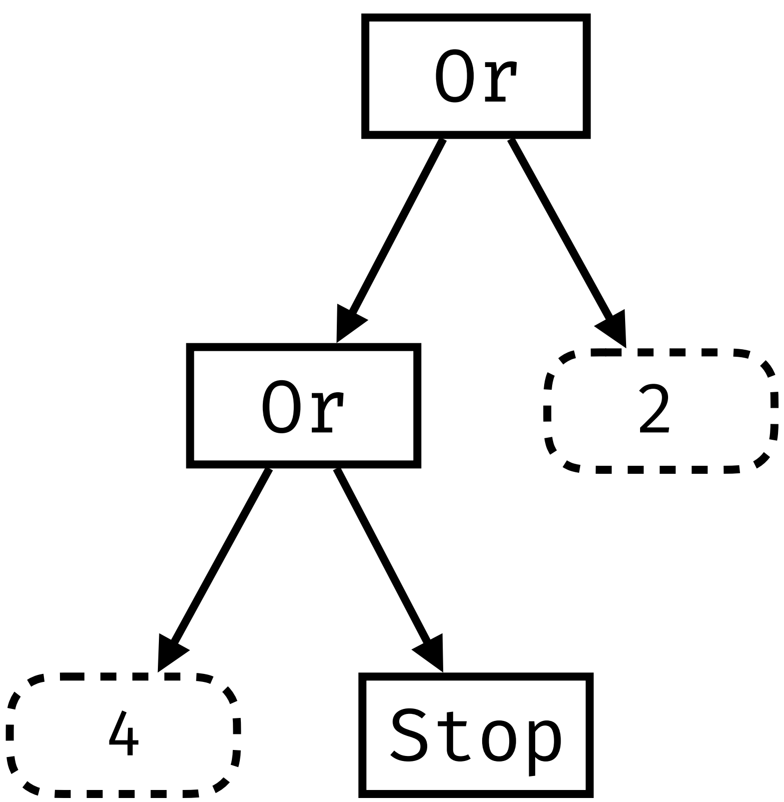
```
run :: Free Void a → a
run = eval gen alg where
  gen = id
  alg = error "unreachable"
```



Local and Global Exceptions

```
global :: Functor sig ⇒
  Free (Or :+ Stop :+ sig) a → Free sig (Maybe [a])
global = stop . list
```

```
local :: Functor sig ⇒
  Free (Stop :+ Or :+ sig) a → Free sig [Maybe a]
local = list . stop
```




```
run . global
-----------→ Nothing
run . local
-----------→ [Just 4, Nothing, Just 2]
```

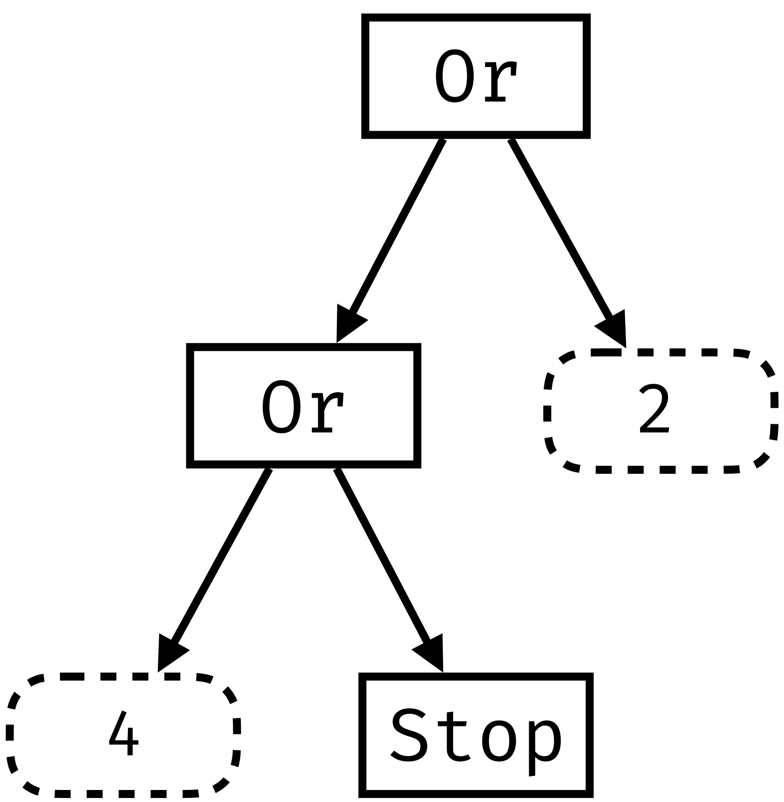
Local and Global Exceptions

```
global :: Functor sig ⇒
  (Or :+ Stop :+ sig) a → Free sig (Maybe
global = stop . list
```

persistent

```
local :: Functor sig ⇒
  (Stop :+ Or :+ sig) a → Free si      a]
local = list . stop
```

provisional



run . global ⇢ Nothing

run . local ⇢ [Just 4, Nothing, Just 2]

Effects Everywhere!

There are lots of algebraic effects, each with various handlers that deal with them

Effect	**Handlers**
Exceptions	catch
Nondeterminism	every, once
Reader	local
Writer	flush
State	exec, run
Threads	spawn, fork

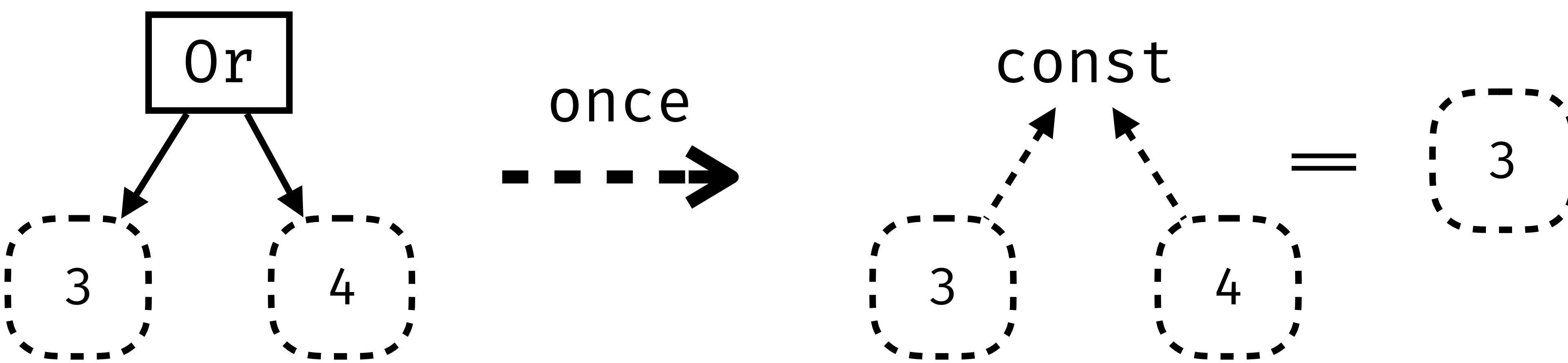
So ... did you spot the fine details that lead to failure in pragmatic programming?

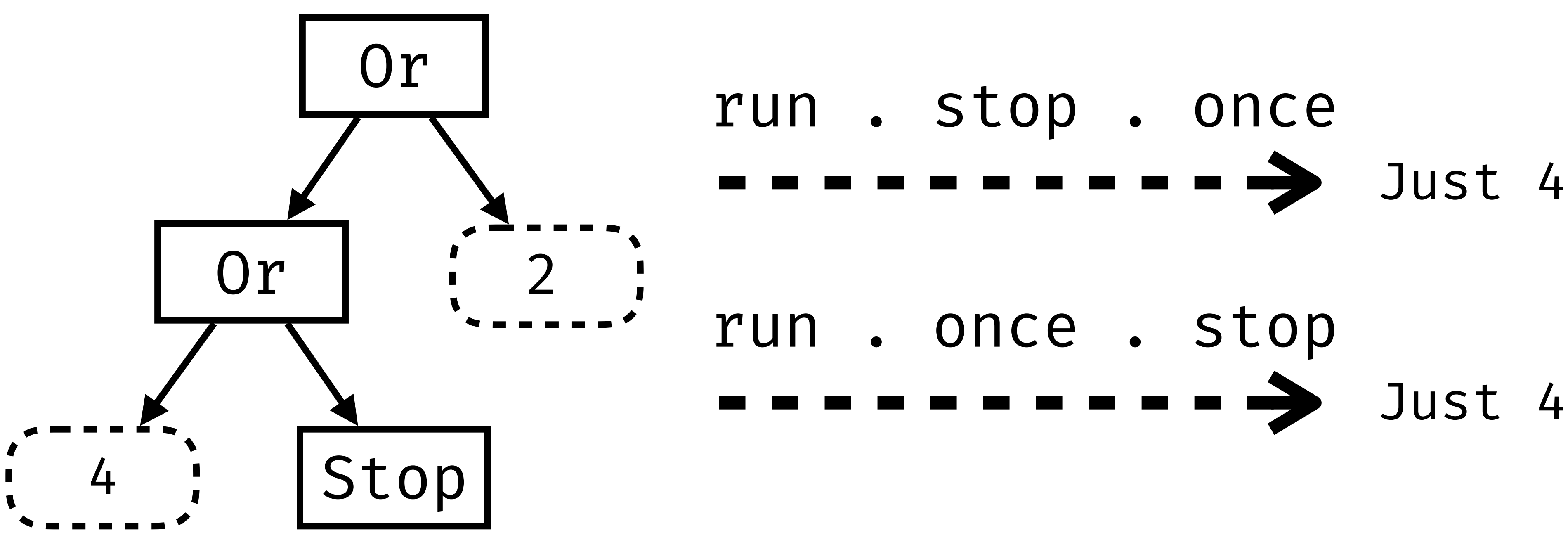
Effects are handled algebraically

Handlers might not be!

The Fine Print

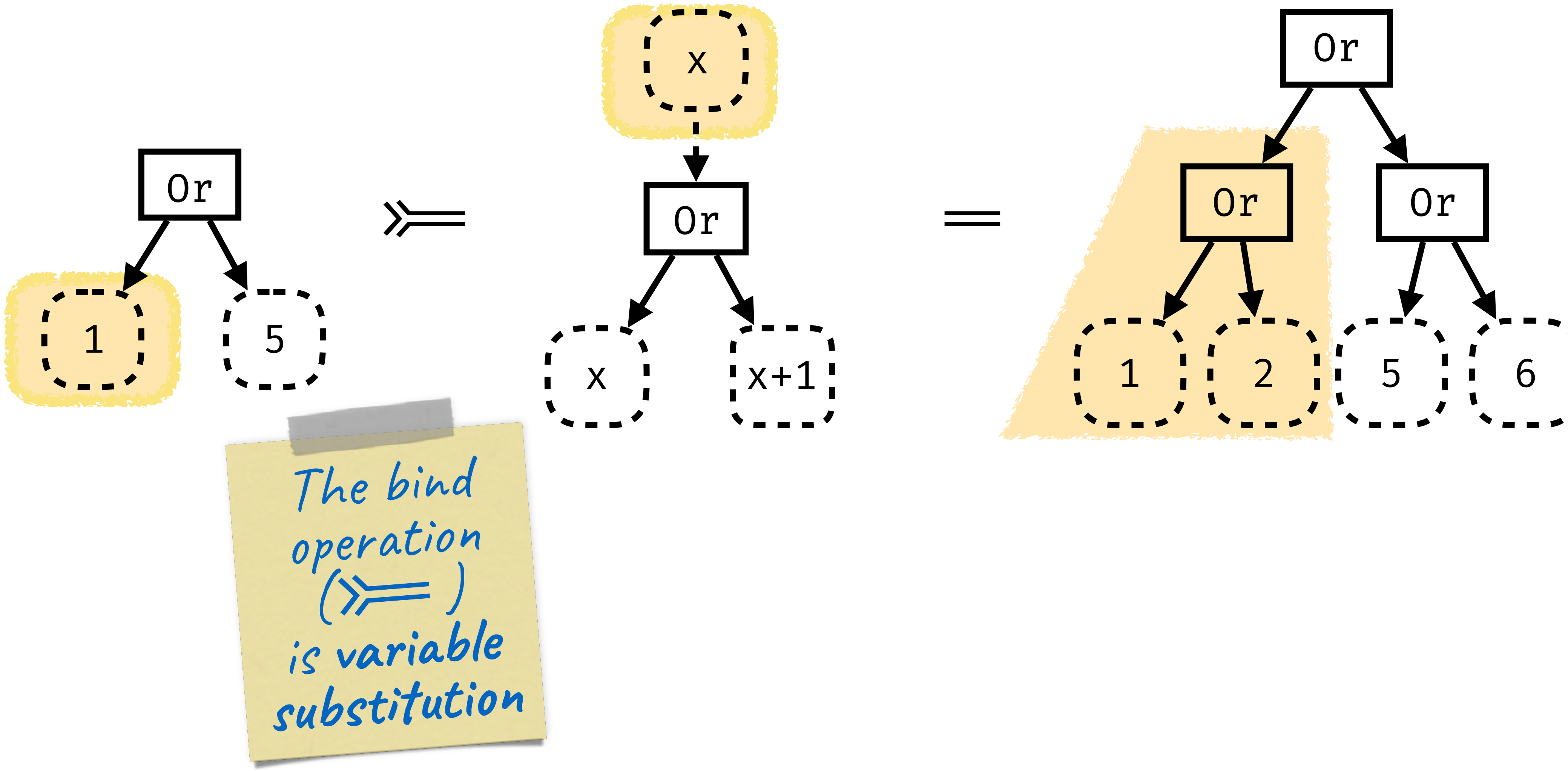
```
once :: Functor f ⇒ Free (Or :+ f) a → Free f a
once = eval gen (embed alg) where
  gen x          = Var x
  alg (Or x y) = const x y
                 = x
```



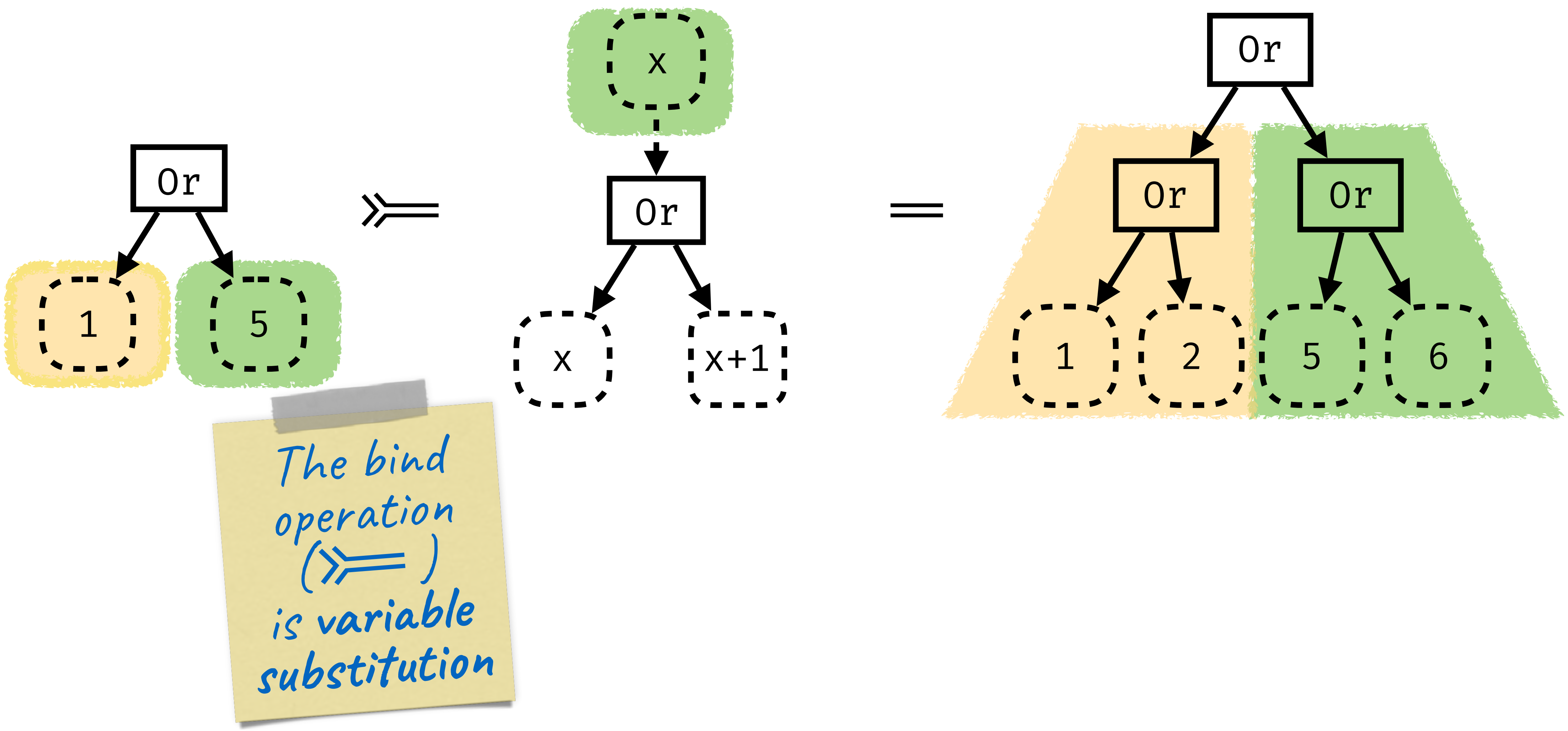


Scoped Operations

Algebraicity



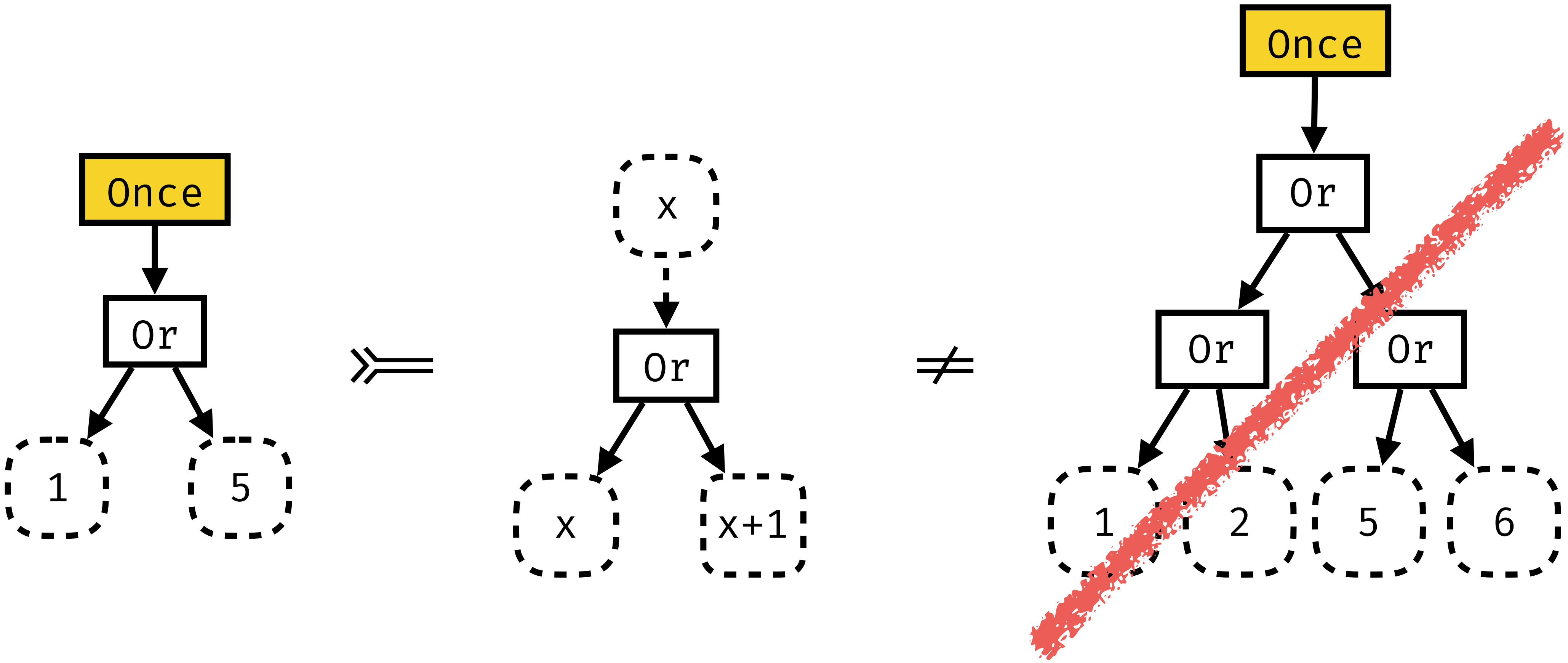
Algebraicity



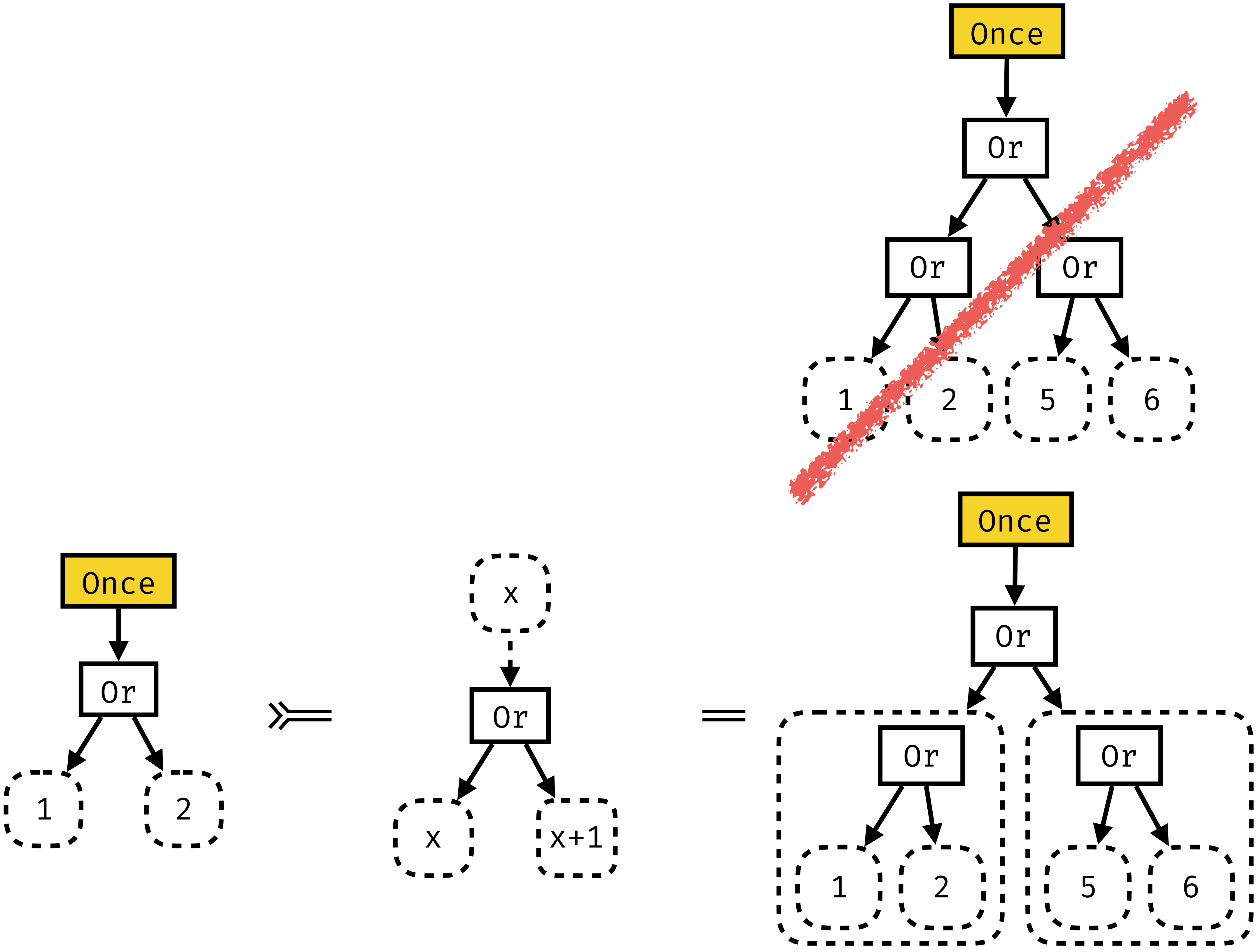
The *or* operation is *algebraic* because
it behaves well with substitution:

$$\mathtt{or(p1,\ p2)} \gg= \mathtt{k} \quad = \quad \mathtt{or(p1} \gg= \mathtt{k,\ p2} \gg= \mathtt{k)}$$

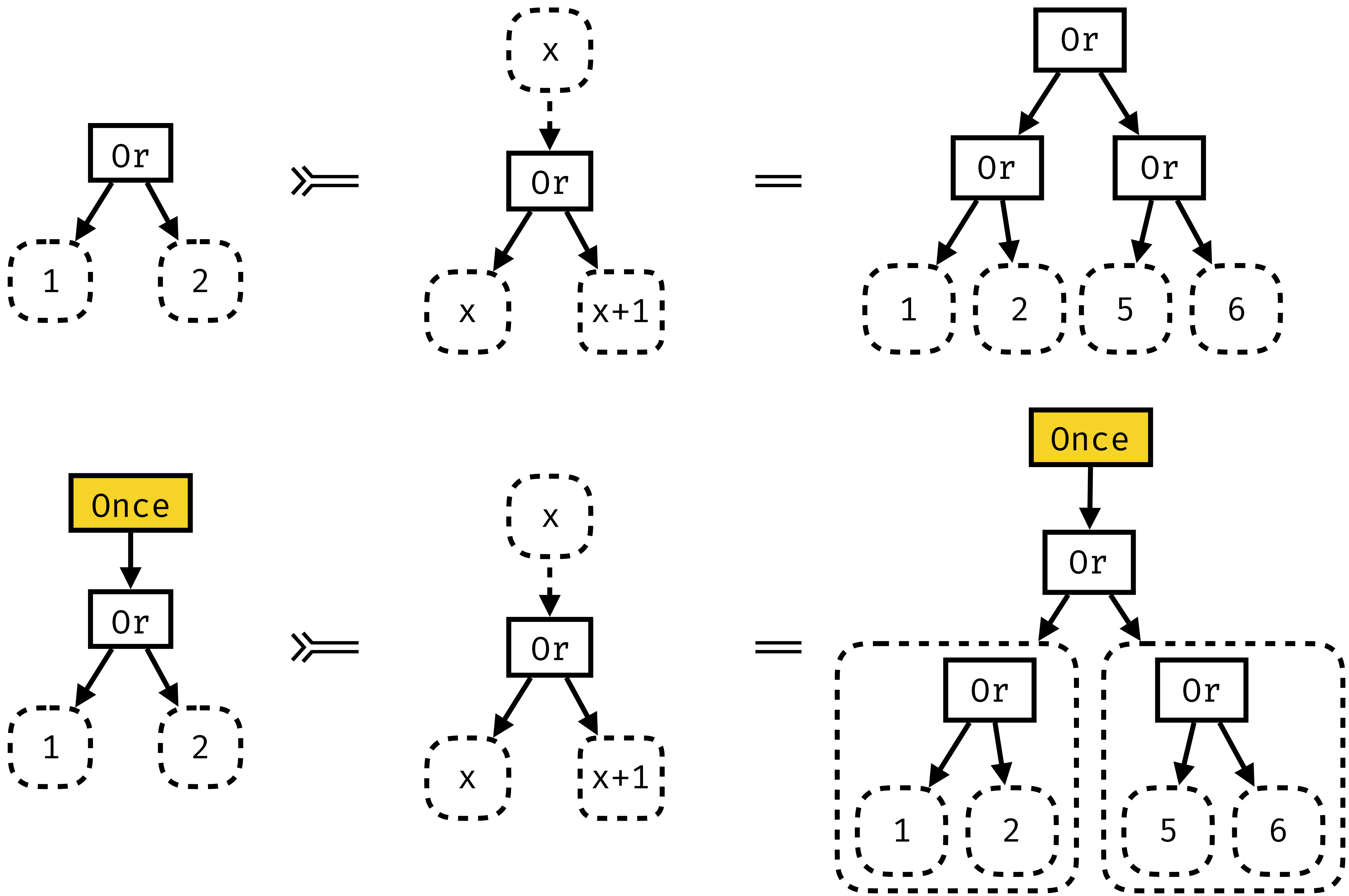
Once



Once

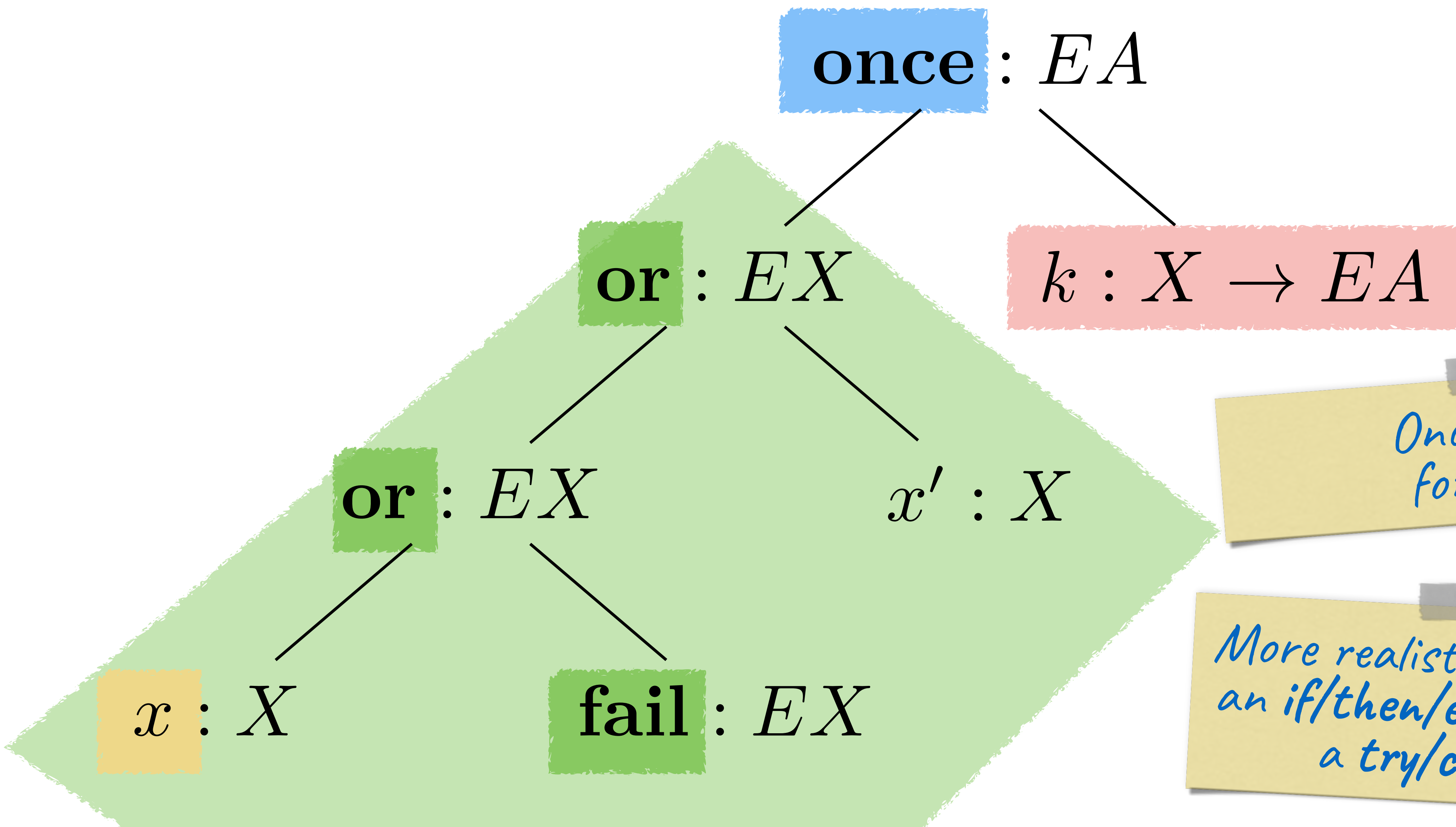


Once



Once

Our goal is to treat *once* like an operation:



Once is Γ = Id for simplicity

More realistically, Γ could be an **if/then/else** construct or a **try/catch** block

$$EA \cong A + \Sigma(EA) + \int^{X \in \mathscr{C}} \Gamma(EX) \times (EA)^X$$

Scoped Free

The coend equation for our explicit substitution is:

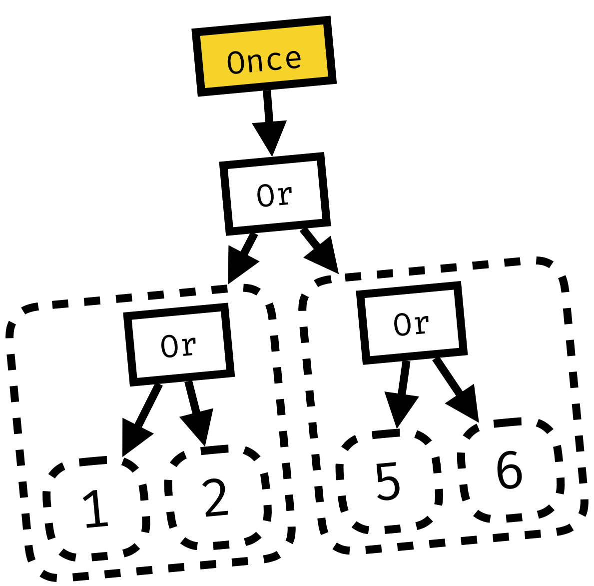
$$EA \cong A \; + \; \Sigma(EA) \; + \int^{X \in \mathscr{C}} \Gamma(EX) \; \times \; (EA)^X$$

it can be reduced to:

$$EA \cong A \; + \; \Sigma(EA) \; + \; \Gamma(E(EA))$$

and this has an easy implementation:

```
data Prog f g a = Var a
                | Op (f (Prog f g a))
                | Scope (g (Prog f g (Prog f g a)))
```



However, the algebras are problematic:

```
alg :: g (Prog f g a) → a
```

this violates step-by-step reduction strategy!

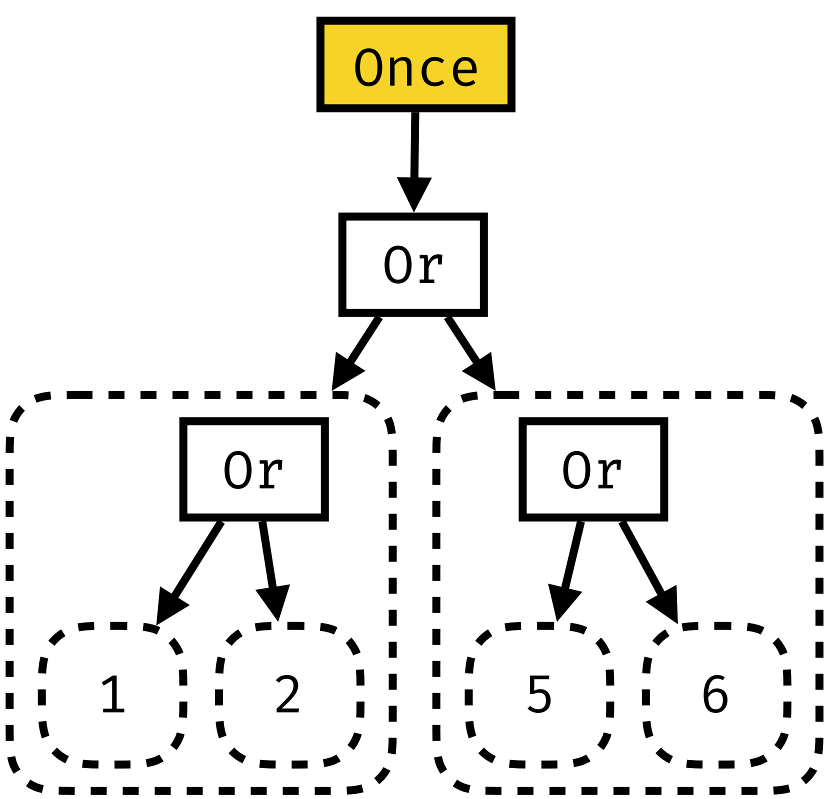
Indexed Carriers

Once Again

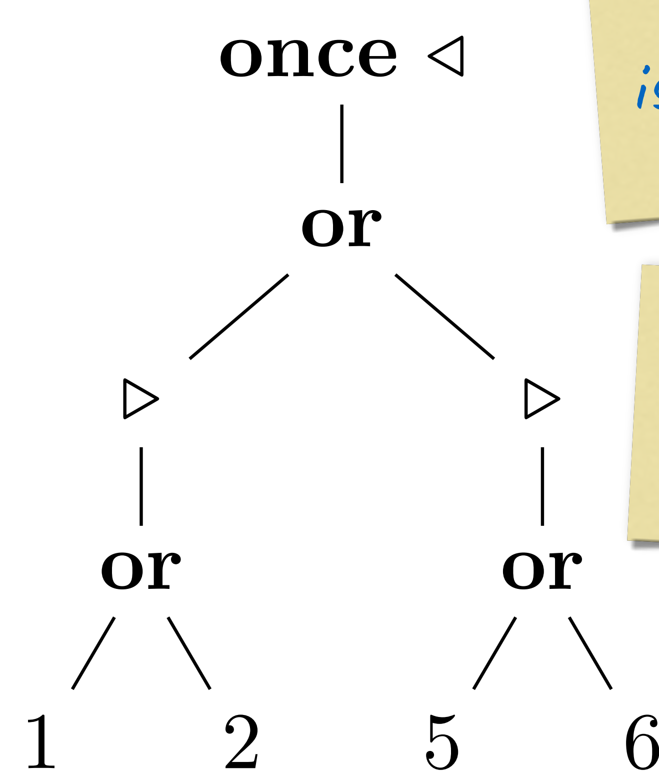
Consider this example:

$$\mathbf{once}(\mathbf{or}(\mathit{return}\,1, \mathit{return}\,5)) \gg\!= \lambda x.\,\mathbf{or}(\mathit{return}\,x, \mathit{return}\,(x+1))$$

As a tree, this becomes:



$\cong$

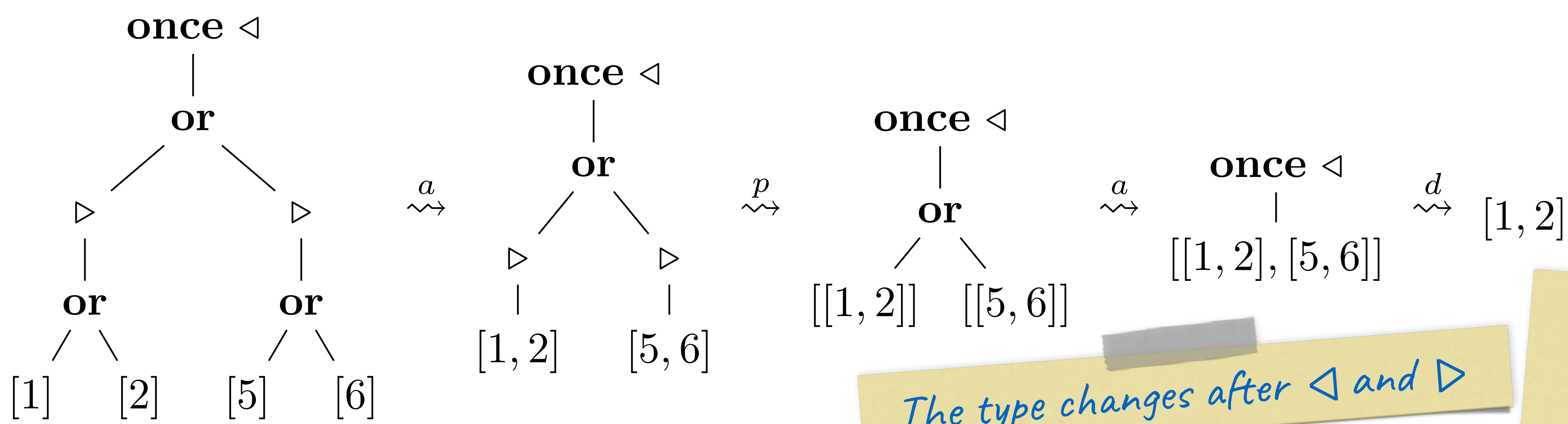


the structure is flattened at the value level

type indexes maintain the structure

Once Again

Starting from the bottom, step-by-step:



The type changes after ◁ and ▷

each level is indexed by a number

We can interpret the tree with these three algebras:

```haskell
data Nat = Z | S Nat

data Alg f g a = Alg { a :: forall n . f (a n) → a n
                     , p :: forall n . a n → a (S n)
                     , d :: forall n . g (a (S n)) → a n
```

$$\langle A, a : \overline{\Sigma} A \to A, d : \overline{\Gamma} \lhd A \to A, p : \rhd A \to A \rangle$$

$$\overline{\Sigma} + \overline{\Gamma} \lhd + \rhd : \mathscr{C}^{|\mathbb{N}|} \to \mathscr{C}^{|\mathbb{N}|}$$

Indexed Carriers

We are working in an indexed category: $\mathscr{C}^{|\mathbb{N}|}$

These endofunctors provide a way of
moving between levels:

$$(\lhd A)_i = A_{i+1}$$

$$(\rhd A)_0 = 0 \qquad (\rhd A)_{i+1} = A_i$$

We can lift the signatures easily enough:

$$(\overline{\Sigma} A)_n = \Sigma(A_n)$$

Lifting to $\mathscr{C}^{|\mathbb{N}|}$

Our computations are in the underlying category

$$\downharpoonleft : \mathscr{C}^{|\mathbb{N}|} \to \mathscr{C} \qquad \upharpoonright : \mathscr{C} \to \mathscr{C}^{|\mathbb{N}|}$$

$$\downharpoonleft A = A_0 \qquad (\upharpoonright X)_0 = X \qquad (\upharpoonright X)_{n+1} = 0$$

$$\begin{tikzcd}
\mathscr{C} \arrow[loop left, "\downharpoonleft M \upharpoonright"] \arrow[r, shift left=2, "\upharpoonright"] \arrow[r, phantom, "\perp"] & \mathscr{C}^{|\mathbb{N}|} \arrow[l, shift left=2, "\downharpoonleft"] \arrow[loop below, "M"] \arrow[r, shift left=2, "F"] \arrow[r, phantom, "\perp"] & (\overline{\Sigma} + \overline{\Gamma}\triangleleft + \triangleright)\text{-}\mathbf{Alg} \arrow[l, shift left=2, "U"]
\end{tikzcd}$$

$$\begin{aligned} M &= UF \\ &= (\overline{\Sigma} + \overline{\Gamma}\triangleleft + \triangleright)^* \end{aligned}$$

Implementation

in Haskell we implement this as follows, where the carrier is indexed:

$$
\begin{aligned}
&\mathbf{data}\ Nat = Zero \mid Succ\ Nat \\
&\mathbf{data}\ Alg\ f\ g\ a = A\ \{ a :: \forall n. f\ (a\ n) \rightarrow a\ n \\
&\qquad\qquad\qquad\quad ,\ d :: \forall n. g\ (a\ (Succ\ n)) \rightarrow a\ n \\
&\qquad\qquad\qquad\quad ,\ p :: \forall n. a\ n \rightarrow a\ (Succ\ n) \}
\end{aligned}
$$

the fold for this is as expected, using p and d where required:

$$
\begin{aligned}
&fold :: (Functor\ f, Functor\ g) \Rightarrow Alg\ f\ g\ a \rightarrow Prog\ f\ g\ (a\ n) \rightarrow a\ n \\
&fold\ alg\ (Var\ x) \quad\ = x \\
&fold\ alg\ (Op\ op) \quad = a\ alg\ (fmap\ (fold\ alg)\ op) \\
&fold\ alg\ (Scope\ sc) = d\ alg\ (fmap\ (fold\ alg \circ fmap\ (p\ alg \circ fold\ alg))\ sc)
\end{aligned}
$$

Once Implementation

$$\textbf{data}\ Carrier_{ND}\ a\ n = ND\ [Carrier_{ND}{}'\ a\ n]$$

$$\begin{aligned}
&\textbf{data}\ Carrier_{ND}{}'\ a :: Nat \to *\textbf{where}\\
&\quad CZ_{ND} :: a \to Carrier_{ND}{}'\ a\ Zero\\
&\quad CS_{ND} :: [Carrier_{ND}{}'\ a\ n] \to Carrier_{ND}{}'\ a\ (Succ\ n)
\end{aligned}$$

$$\begin{aligned}
&gen_{ND} :: a \to Carrier_{ND}\ a\ Zero\\
&gen_{ND}\ x = ND\ [CZ_{ND}\ x]
\end{aligned}$$

$$\begin{aligned}
&alg_{ND} :: Alg\ Choice\ Once\ (Carrier_{ND}\ a)\\
&alg_{ND} = A\ \{..\}\ \textbf{where}\\
&\quad a :: \forall n\ a.\ Choice\ (Carrier_{ND}\ a\ n) \to Carrier_{ND}\ a\ n\\
&\quad a\ Fail && = ND\ [\,]\\
&\quad a\ (Or\ (ND\ l)\ (ND\ r)) && = ND\ (l +\!\!+\ r)\\
&\quad d :: \forall n\ a.\ Once\ (Carrier_{ND}\ a\ (Succ\ n)) \to Carrier_{ND}\ a\ n\\
&\quad d\ (Once\ (ND\ [\,])) && = ND\ [\,]\\
&\quad d\ (Once\ (ND\ (CS_{ND}\ l : _))) && = ND\ l\\
&\quad p :: \forall n\ a.\ Carrier_{ND}\ a\ n \to Carrier_{ND}\ a\ (Succ\ n)\\
&\quad p\ (ND\ l) && = ND\ [CS_{ND}\ l]
\end{aligned}$$

$$\begin{tikzcd}
\begin{array}{c}(\overline{\Sigma}E^+A)_{n+1}\\ =\Sigma EE^{n+1}A\end{array} \arrow[rr, "\Sigma Eq_n"] \arrow[d, "k^{\Sigma}_{n+1}=\mathsf{cons}"'] \arrow[dr, "\Sigma i"] \arrow[drr, phantom, "\mathsf{a}"] & & \Sigma E(M\upharpoonright A)_n \arrow[ddd, "\Sigma i"] \\
EE^{n+1}A \arrow[d, "Eq_n"] \arrow[dr, "i"] \arrow[r, phantom, "\mathsf{b}"] & \begin{array}{c}\Sigma\downharpoonleft M\upharpoonright E^{n+1}A\\ =\downharpoonleft\overline{\Sigma}M\upharpoonright E^{n+1}A\end{array} \arrow[d, "i^{\Sigma}=\downharpoonleft\mathsf{cons}"] \arrow[ddr, bend left, "\Sigma\downharpoonleft M\upharpoonright q_n"] & \\
E(M\upharpoonright A)_n \arrow[d, "i"] \arrow[r, phantom, "\mathsf{c}"] & \downharpoonleft M\upharpoonright E^{n+1}A \arrow[dl, "\downharpoonleft M\upharpoonright q_n"] \arrow[d, phantom, "\mathsf{d}"] & \\
\begin{array}{c}\downharpoonleft M\upharpoonright(M\upharpoonright A)_n\\ =\downharpoonleft M\upharpoonright\downharpoonleft\triangleleft^n M\upharpoonright A\end{array} \arrow[d, "\downharpoonleft M\epsilon"] \arrow[drr, phantom, "\mathsf{e}"] & & \begin{array}{c}\Sigma\downharpoonleft M\upharpoonright(M\upharpoonright A)_n\\ =\Sigma\downharpoonleft M\upharpoonright\downharpoonleft\triangleleft^n M\upharpoonright A\end{array} \arrow[ll, "\downharpoonleft\mathsf{cons}"'] \arrow[d, "\Sigma\downharpoonleft M\epsilon"] \\
\begin{array}{c}\downharpoonleft M\triangleleft^n M\upharpoonright A\\ =\downharpoonleft M\triangleleft^{n+1}\triangleright M\upharpoonright A\end{array} \arrow[d, "\downharpoonleft\lambda^{n+1}"] \arrow[drr, phantom, "\mathsf{f}"] & & \begin{array}{c}\Sigma\downharpoonleft M\triangleleft^n M\upharpoonright A\\ =\Sigma\downharpoonleft M\triangleleft^{n+1}\triangleright M\upharpoonright A\\ =\downharpoonleft\overline{\Sigma}M\triangleleft^{n+1}\triangleright M\upharpoonright A\end{array} \arrow[ll, "\downharpoonleft\mathsf{cons}"'] \arrow[d, "\Sigma\downharpoonleft\lambda^{n+1}"] \\
\downharpoonleft\triangleleft^{n+1}M\triangleright M\upharpoonright A \arrow[d, "\downharpoonleft\triangleleft^{n+1}\mathsf{struct}"] \arrow[drr, phantom, "\mathsf{g}"] & & \begin{array}{c}\Sigma\downharpoonleft\triangleleft^{n+1}M\triangleright M\upharpoonright A\\ =\downharpoonleft\overline{\Sigma}\triangleleft^{n+1}M\triangleright M\upharpoonright A\\ =\downharpoonleft\triangleleft^{n+1}\overline{\Sigma}M\triangleright M\upharpoonright A\end{array} \arrow[ll, "\downharpoonleft\lambda^{\Sigma}=\downharpoonleft\triangleleft^{n+1}\mathsf{cons}"'] \arrow[d, "\Sigma\downharpoonleft\triangleleft^{n+1}\mathsf{struct}=\downharpoonleft\triangleleft^{n+1}\overline{\Sigma}\mathsf{struct}"] \\
\begin{array}{c}\downharpoonleft\triangleleft^{n+1}M\upharpoonright A\\ =(M\upharpoonright A)_{n+1}\end{array} & & \begin{array}{c}\Sigma\downharpoonleft\triangleleft^{n+1}M\upharpoonright A\\ =\Sigma(M\upharpoonright A)_{n+1}\\ =(\overline{\Sigma}M\upharpoonright A)_{n+1}\end{array} \arrow[ll, "\mathsf{cons}_{n+1}=\downharpoonleft\triangleleft^{n+1}\mathsf{cons}"]
\end{tikzcd}$$

$\mathsf{E} = \downharpoonleft \mathsf{M} \upharpoonright$

$$\begin{tikzcd}
\triangleleft E^+ \arrow[rrr, "\eta^M"] \arrow[dr, "\triangleleft\eta^M"] \arrow[dddrrr, bend right, "\mathrm{id}"] \arrow[rr, phantom, "\mathsf{a}"] & & & M\triangleleft E^+ \arrow[dl, "M\triangleleft\eta^M"'] \arrow[dd, "\lambda"] \arrow[dl, phantom, "\mathsf{b}"] \\
& \triangleleft ME^+ \arrow[r, "\eta^M"] \arrow[ddrr, phantom, "\mathsf{d}"] \arrow[drr, bend right, "\mathrm{id}"] \arrow[r, phantom, "\mathsf{c}"'] & M\triangleleft ME^+ \arrow[dr, "\widehat{\lambda}"] & \\
& & & \triangleleft ME^+ \arrow[d, "\triangleleft\widehat{k}"] \\
& & & \triangleleft E^+
\end{tikzcd}$$

$$\begin{tikzcd}
\triangleleft E^+ + \overline{\Sigma}M\triangleleft E^+ + \overline{\Gamma}\triangleleft M\triangleleft E^+ + \triangleright M\triangleleft E^+ \arrow[r, "{[\eta^M,\ \mathsf{cons},\ \mathsf{cons},\ \mathsf{cons}]}"] \arrow[d, "\mathrm{id}+\overline{\Sigma}(\triangleleft\widehat{k}\cdot\lambda E^+)+\overline{\Gamma}\triangleleft(\triangleleft\widehat{k}\cdot\lambda E^+)+\triangleright(\triangleleft\widehat{k}\cdot\lambda E^+)"] & M\triangleleft E^+ \arrow[d, "\triangleleft\widehat{k}\cdot\lambda E^+"] \\
\triangleleft E^+ + \overline{\Sigma}\triangleleft E^+ + \overline{\Gamma}\triangleleft\triangleleft E^+ + \triangleright\triangleleft E^+ \arrow[r, "{[\mathrm{id},\ k^{\Sigma}E,\ k^{\triangleleft}E,\ k^{\triangleright}E]}"] & \triangleleft E^+
\end{tikzcd}$$

$$\begin{tikzcd}
\Gamma\downharpoonleft M\upharpoonright\downharpoonleft M\upharpoonright \arrow[r, "\Gamma\downharpoonleft k\downharpoonleft k"] \arrow[dr, bend right, "\Gamma\downharpoonleft M\upharpoonright\downharpoonleft k"'] \arrow[dd, "\Gamma\downharpoonleft M\epsilon M\upharpoonright"'] \arrow[dr, phantom, "\mathsf{a}"] & \Gamma\downharpoonleft E^+\downharpoonleft E^+ \arrow[r, "="] & \Gamma EE \arrow[dddddddd, "\mathsf{cons}"] \arrow[ddddddl, bend right, "="'] \arrow[ddddddddl, phantom, "\mathsf{e}"] \\
& \Gamma\downharpoonleft M\upharpoonright\downharpoonleft E^+ \arrow[u, "\Gamma\downharpoonleft k\downharpoonleft E^+"'] \arrow[d, "\Gamma\downharpoonleft M\epsilon E^+"] & \\
\Gamma\downharpoonleft MM\upharpoonright \arrow[d, "="] \arrow[r, phantom, "\mathsf{b}"] & \Gamma\downharpoonleft ME^+ \arrow[d, "="] \arrow[r, phantom, "\mathsf{d}"] & \\
\Gamma\downharpoonleft M\triangleleft\!\triangleright M\upharpoonright \arrow[d, "\Gamma\downharpoonleft\lambda\triangleright M\upharpoonright"] & \Gamma\downharpoonleft M\triangleleft\!\triangleright E^+ \arrow[d, "\Gamma\downharpoonleft\lambda\triangleright E^+"] & \\
\Gamma\downharpoonleft\triangleleft M\triangleright M\upharpoonright \arrow[d, "="] & \Gamma\downharpoonleft\triangleleft M\triangleright E^+ \arrow[d, "="] & \\
\downharpoonleft\overline{\Gamma}\triangleleft M\triangleright M\upharpoonright \arrow[r, "\downharpoonleft\overline{\Gamma}\triangleleft M\triangleright k"] \arrow[dd, "\downharpoonleft\,\mathsf{cons\ cons}\upharpoonright"] \arrow[ddr, phantom, "\mathsf{c}"] & \downharpoonleft\overline{\Gamma}\triangleleft M\triangleright E^+ \arrow[d, "\downharpoonleft\overline{\Gamma}\triangleleft Mk^{\triangleright}"] & \\
& \downharpoonleft\overline{\Gamma}\triangleleft ME^+ \arrow[d, "\downharpoonleft\overline{\Gamma}\triangleleft\widehat{k}"] & \\
\downharpoonleft MM\upharpoonright \arrow[d, "\downharpoonleft\mu^M\upharpoonright"] & \downharpoonleft\overline{\Gamma}\triangleleft E^+ \arrow[d, "\downharpoonleft k^{\triangleleft}"] & \\
\downharpoonleft M\upharpoonright \arrow[r, "\downharpoonleft k"] & \downharpoonleft E^+ \arrow[r, "="] & E
\end{tikzcd}$$

$$\begin{tikzcd}
\overline{\Sigma}M\triangleleft E^+ \arrow[rr, "\mathsf{cons}"] \arrow[d, "\overline{\Sigma}\lambda"] \arrow[drr, phantom, "\mathsf{a}"] & & M\triangleleft E^+ \arrow[d, "\lambda"] \\
\overline{\Sigma}\triangleleft ME^+ \arrow[rr, "\lambda^{\Sigma}"] \arrow[dr, "="] \arrow[ddd, "\overline{\Sigma}\triangleleft\widehat{k}"] \arrow[rr, phantom, "\mathsf{b}"', shift right=3] & & \triangleleft ME^+ \arrow[ddd, "\triangleleft\widehat{k}"] \\
& \triangleleft\overline{\Sigma}ME^+ \arrow[ur, "\triangleleft\mathsf{cons}"'] \arrow[d, "\triangleleft\overline{\Sigma}\widehat{k}"] \arrow[l, phantom, "\mathsf{c}", shift right=6] \arrow[r, phantom, "\mathsf{d}", shift right=6] & \\
& \triangleleft\overline{\Sigma}E^+ \arrow[dr, "\triangleleft k^{\Sigma}"] \arrow[d, phantom, "\mathsf{e}"] & \\
\overline{\Sigma}\triangleleft E^+ \arrow[ur, "="] \arrow[rr, "k^{\Sigma}"] & & \triangleleft E^+
\end{tikzcd}$$

$$\begin{tikzcd}
M\upharpoonright\downharpoonleft E^+ \arrow[rr, "k\downharpoonleft E^+"] \arrow[d, "M\epsilon E^+"] \arrow[dr, "Mk^{\mathrm{var}}\downharpoonleft E^+"] & & E^+\downharpoonleft E^+ \arrow[ddd, "="] \\
ME^+ \arrow[d, "="] \arrow[dr, phantom, "\mathsf{a}"] & ME^+\downharpoonleft E^+ \arrow[dd, "="] \arrow[r, phantom, "\mathsf{b}"] & \\
M\triangleleft\!\triangleright E^+ \arrow[d, "M\triangleleft k^{\triangleright}"] & & \\
M\triangleleft E^+ \arrow[r, "="] \arrow[d, "\lambda E^+"] \arrow[dr, phantom, "\mathsf{c}"] & ME^+E \arrow[r, "\widehat{k}E"] & E^+E \arrow[d, "="] \\
\triangleleft ME^+ \arrow[rr, "\triangleleft\widehat{k}"] & & \triangleleft E^+
\end{tikzcd}$$

$$\begin{tikzcd}
& \overline{\Gamma}\triangleleft M\triangleright M\triangleleft^{n-1}\triangleleft \arrow[rr, "\mathsf{struct}"] \arrow[dl, "\overline{\Gamma}\triangleleft M\triangleright\lambda^n"'] \arrow[dr, "\overline{\Gamma}\triangleleft M\triangleright\lambda^{n-1}"] \arrow[d, phantom, "\mathsf{a}"] & & M\triangleleft^{n-1}\triangleleft \arrow[ddd, "\lambda^{n-1}"] \arrow[dddl, phantom, "\mathsf{b}"] \\
\overline{\Gamma}\triangleleft M\triangleright\triangleleft^{n-1}\triangleleft M \arrow[dd, "\overline{\Gamma}\triangleleft M\epsilon"] \arrow[ddrr, phantom, "\mathsf{c}"] & & \overline{\Gamma}\triangleleft M\triangleright\triangleleft^{n-1}M\triangleleft \arrow[ll, "\overline{\Gamma}\triangleleft M\triangleright\triangleleft^{n-1}\lambda"'] \arrow[d, "\overline{\Gamma}\triangleleft M\epsilon"] & \\
& & \begin{array}{c}\overline{\Gamma}\triangleleft M\triangleleft^{n-2}M\triangleleft\\ =\overline{\Gamma}\triangleleft M\triangleleft^{n-1}\triangleright M\triangleleft\end{array} \arrow[d, "\overline{\Gamma}\triangleleft\lambda^{n-1}"] & \\
\begin{array}{c}\overline{\Gamma}\triangleleft M\triangleleft^{n-2}\triangleleft M\\ =\overline{\Gamma}\triangleleft M\triangleleft^{n-1}\triangleright\triangleleft M\end{array} \arrow[dd, bend right, "\mathrm{id}"'] \arrow[dd, bend left, "\overline{\Gamma}\triangleleft M\triangleleft^{n-1}\epsilon"] \arrow[dd, phantom, "\mathsf{d}"] & & \begin{array}{c}\overline{\Gamma}\triangleleft\triangleleft^{n-1}M\triangleright M\triangleleft\\ =\triangleleft^{n-1}\overline{\Gamma}\triangleleft M\triangleright M\triangleleft\end{array} \arrow[r, "\triangleleft^{n-1}\mathsf{struct}"] \arrow[d, "\triangleleft^{n-1}\overline{\Gamma}\triangleleft M\triangleright\lambda"] & \triangleleft^{n-1}M\triangleleft \arrow[ddd, "\triangleleft^{n-1}\lambda"] \arrow[ddd, phantom, "\mathsf{f}"'] \\
& & \triangleleft^{n-1}\overline{\Gamma}\triangleleft M\triangleright\triangleleft M \arrow[d, "\triangleleft^{n-1}\overline{\Gamma}\triangleleft M\epsilon"] & \\
\begin{array}{c}\overline{\Gamma}\triangleleft M\triangleleft^{n-1}M\\ =\overline{\Gamma}\triangleleft M\triangleleft^{n-1}\triangleleft\!\triangleright M\end{array} \arrow[rr, "\overline{\Gamma}\triangleleft M\lambda^{n-1}"] \arrow[dr, "\overline{\Gamma}\triangleleft\lambda^n"] \arrow[drr, phantom, "\mathsf{e}"] & & \begin{array}{c}\triangleleft^{n-1}\overline{\Gamma}\triangleleft MM\\ =\triangleleft^{n-1}\overline{\Gamma}\triangleleft M\triangleleft\!\triangleright M\end{array} \arrow[dl, "\triangleleft^{n-1}\overline{\Gamma}\triangleleft\lambda"] & \\
& \begin{array}{c}\overline{\Gamma}\triangleleft^{n-1}\triangleleft\triangleleft M\triangleright M\\ =\triangleleft^{n-1}\overline{\Gamma}\triangleleft\triangleleft M\triangleright M\\ =\triangleleft^{n-1}\triangleleft\overline{\Gamma}\triangleleft M\triangleright M\end{array} \arrow[rr, "\triangleleft^{n-1}\triangleleft\mathsf{struct}"] & & \triangleleft^{n-1}\triangleleft M
\end{tikzcd}$$