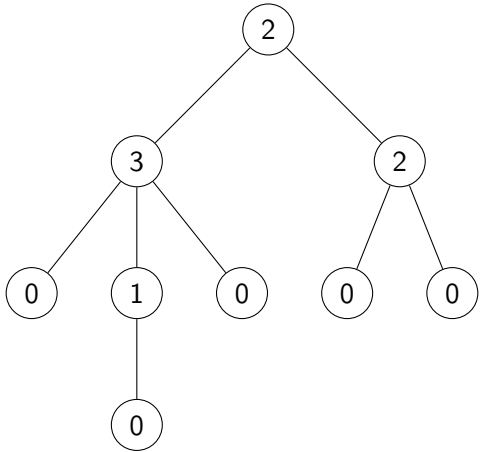
MLTT: $\mathcal{W}$-types

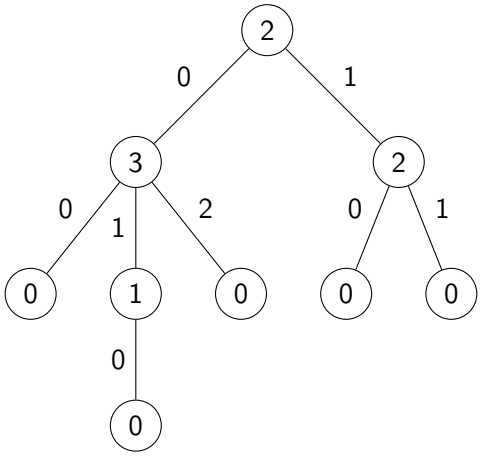
Mikhail Ushakov Rutger Broekhoff

November 21, 2024

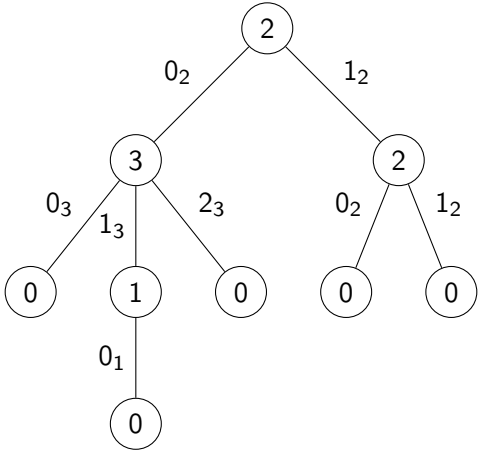
A tree



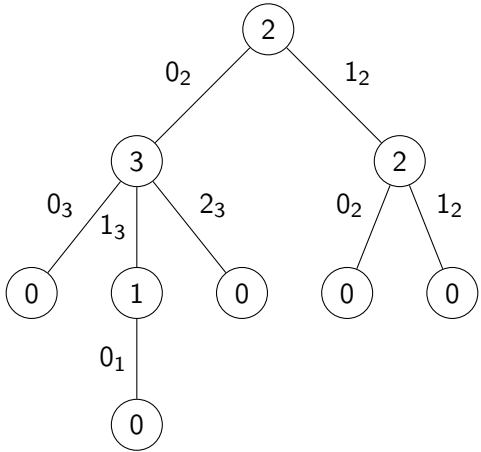
A tree



A tree



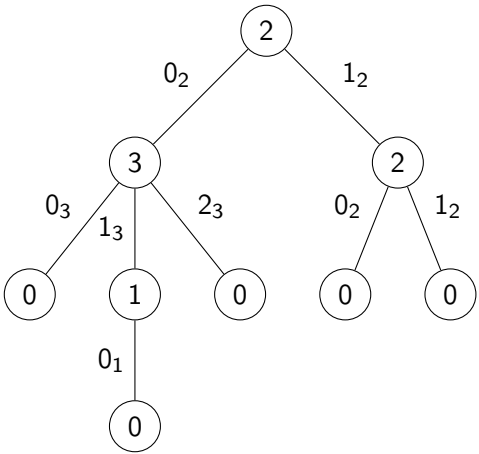
A tree



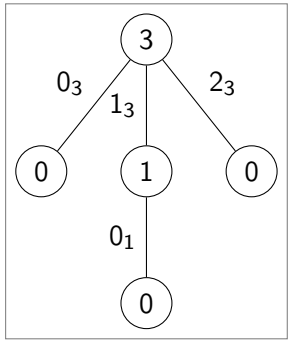
For every node x, labels for edges to descendants are $0_x, 1_x, \ldots, (x-1)_x$.
We say $0_n, 1_n, \ldots, (n-1)_n : \mathbb{N}_n$.

A tree of trees

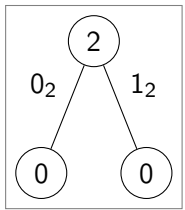
Let $f(x)$ give the subtrees for the root node (2) given the node label $x : \mathbb{N}_2$.



$f(0) =$

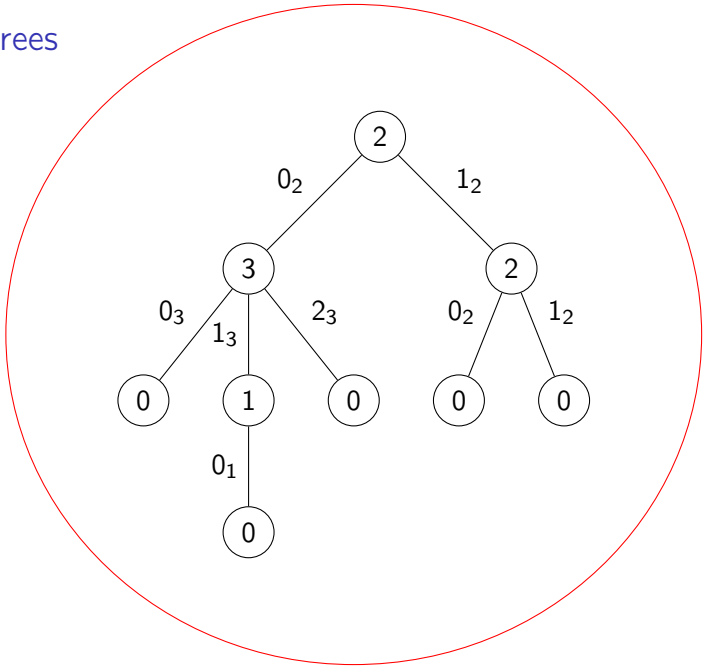


$f(1) =$

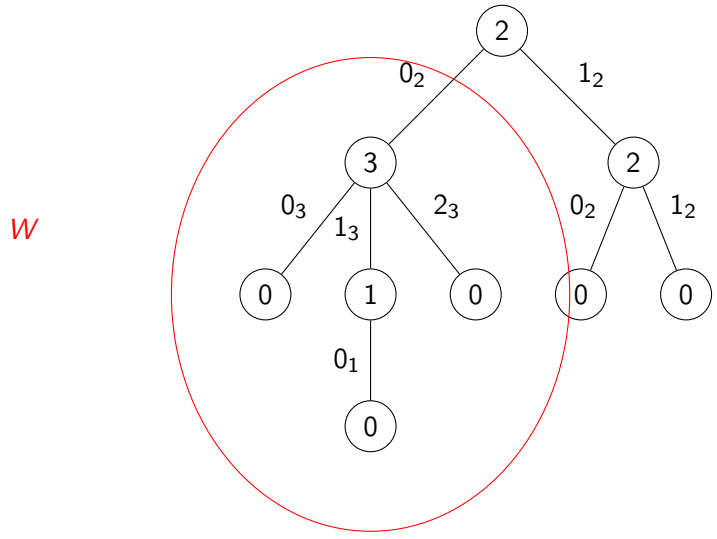


A tree of trees of trees

w

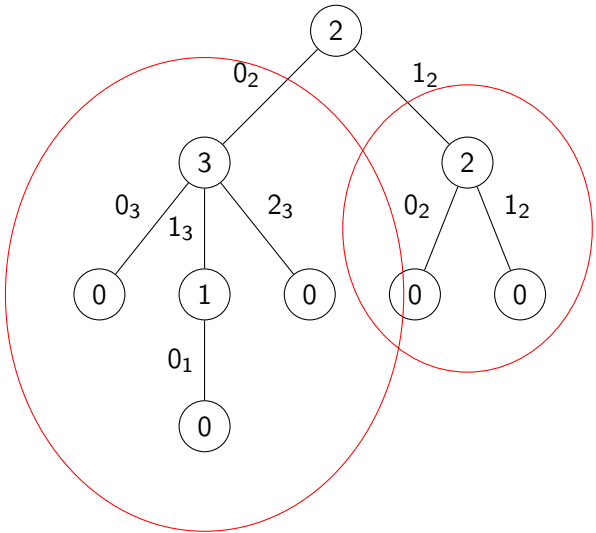


A tree of trees of trees



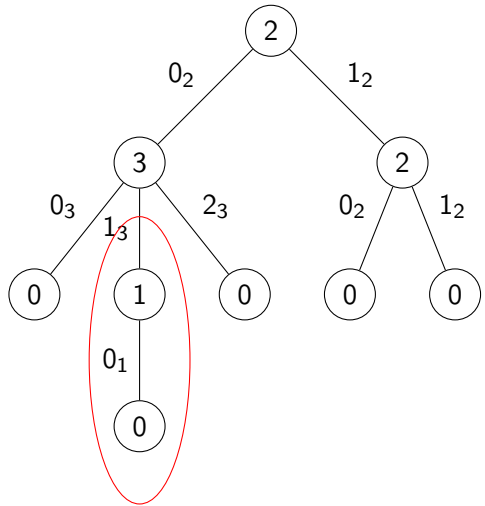
A tree of trees of trees

w



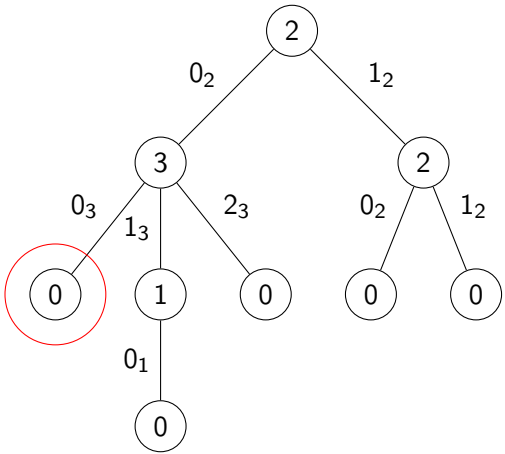
A tree of trees of trees

w



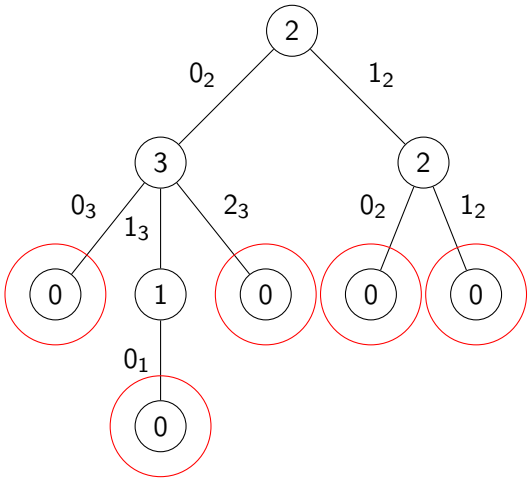
A tree of trees of trees

w



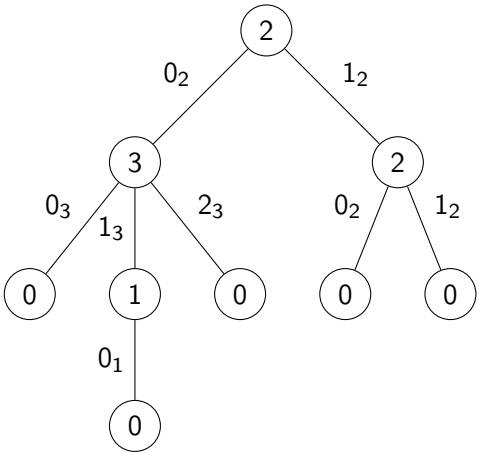
A tree of trees of trees

w

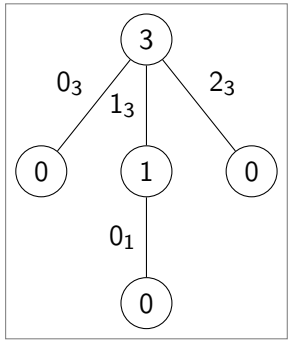


A tree of trees

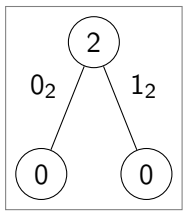
Let $f : \mathbb{N}_2 \to W$ give the subtrees for the root node (2) given some node label in $\mathbb{N}_2$.



$f(0) =$

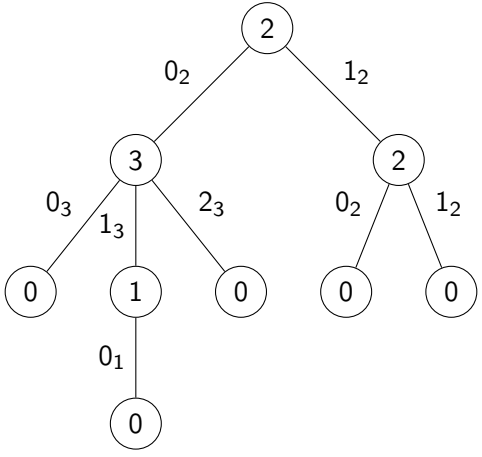


$f(1) =$

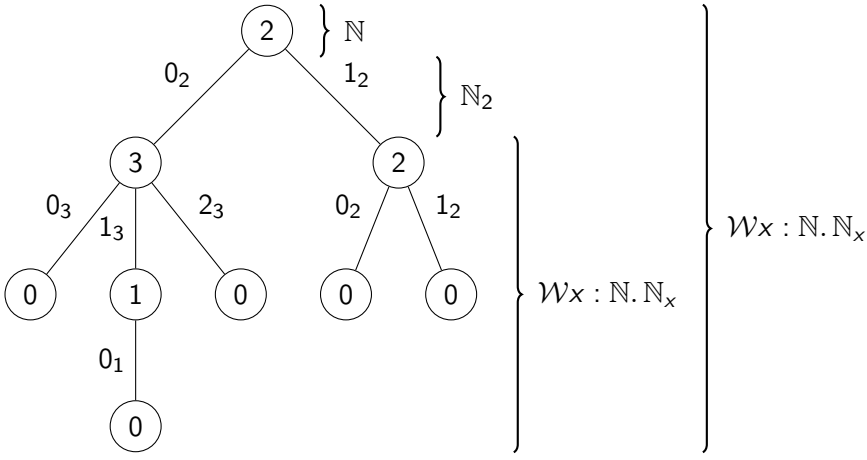


A tree of trees

~~Let $f : \mathbb{N}_2 \to W$ give the subtrees for the root node (2) given some node label in $\mathbb{N}_2$.~~
Let $f : \mathbb{N}_n \to W$ give the subtrees for some node (n) given some node label in $\mathbb{N}_n$.



A generic tree



$$\frac{\Gamma \vdash a : \mathbb{N} \qquad \Gamma \vdash b : \mathbb{N}_a \to \mathcal{W}x : \mathbb{N}.\mathbb{N}_x}{\Gamma \vdash \mathsf{sup}\ a\ b : \mathcal{W}x : \mathbb{N}.\mathbb{N}_x}$$

Writing a subtree as a $\mathcal{W}$-type

$$\frac{\Gamma \vdash a : \mathbb{N} \qquad \Gamma \vdash b : \mathbb{N}_a \to \mathcal{W}x : \mathbb{N}.\, \mathbb{N}_x}{\Gamma \vdash \mathsf{sup}\ a\ b : \mathcal{W}x : \mathbb{N}.\, \mathbb{N}_x}$$

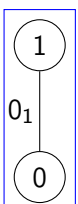
$$\boxed{\textcircled{0}} = \mathsf{sup}\ 0\ \lambda x : \mathbb{N}_0.\ \left\{\langle \text{no cases}, x : \mathbb{N}_0\rangle\right.$$

Writing a subtree as a $\mathcal{W}$-type

$$\frac{\Gamma \vdash a : \mathbb{N} \qquad \Gamma \vdash b : \mathbb{N}_a \to \mathcal{W}x : \mathbb{N}.\, \mathbb{N}_x}{\Gamma \vdash \mathsf{sup}\ a\ b : \mathcal{W}x : \mathbb{N}.\, \mathbb{N}_x}$$

$= \mathsf{sup}\ 0\ \lambda x : \mathbb{N}_0.$

$\left\{ \langle \text{no cases}, x : \mathbb{N}_0 \rangle \right.$



$= \mathsf{sup}\ 1\ \lambda x : \mathbb{N}_1.$

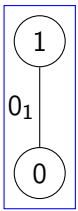
$\left\{ x = 0_1 \to \right.$ 

Writing a subtree as a $\mathcal{W}$-type

$$\frac{\Gamma \vdash a : \mathbb{N} \qquad \Gamma \vdash b : \mathbb{N}_a \to \mathcal{W}x : \mathbb{N}.\, \mathbb{N}_x}{\Gamma \vdash \sup\ a\ b : \mathcal{W}x : \mathbb{N}.\, \mathbb{N}_x}$$

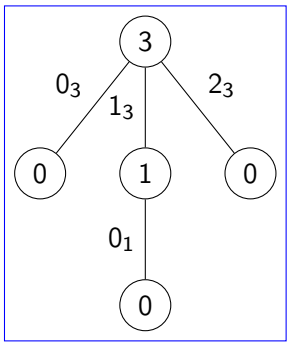
$= \sup\ 0\ \lambda x : \mathbb{N}_0.$

$\{\langle$no cases, $x : \mathbb{N}_0\rangle$



$= \sup\ 1\ \lambda x : \mathbb{N}_1.$

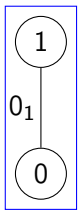
$\{x = 0_1 \to$ 



$=$

$\sup\ 3\ \lambda x : \mathbb{N}_3.$

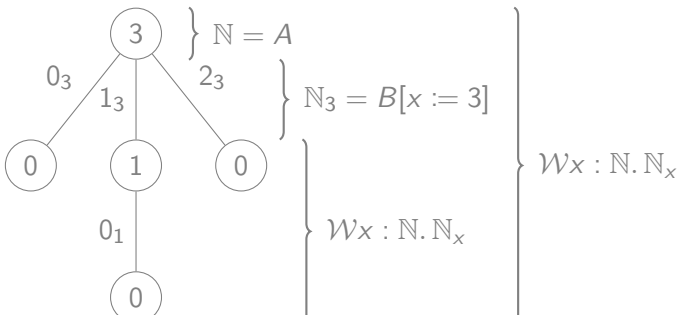
$\{x = 0_3 \to$ 

$x = 1_3 \to$ 

$x = 2_3 \to$ 

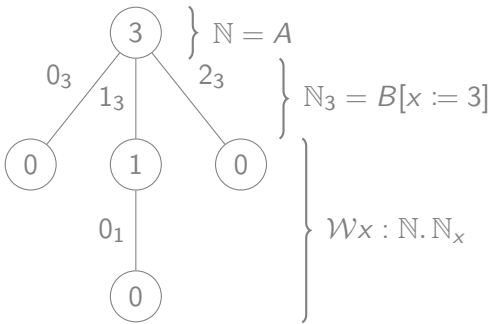
Typing rules

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash (\mathcal{W}x : A.\, B) : *} \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[x \coloneqq a] \to \mathcal{W}x : A.\, B}{\Gamma \vdash \mathsf{sup}\ a\ b : \mathcal{W}x : A.\, B}$$



Elimination rule

$$\frac{\Gamma \vdash f : \Pi \overbrace{x : A}^{\text{node value}}.\, \Pi \overbrace{y : B\ x \to (\mathcal{W} x : A.\, B)}^{\text{branching function}}.\, \overbrace{(\Pi v : B\ x.\, C\ (y\ v))}^{\text{'IH'}} \to C\ (\text{sup}\ x\ y) \qquad \Gamma \vdash w : (\mathcal{W} x : A.\, B)}{\Gamma \vdash \textbf{wrec}\ w\ f : C\ w}$$

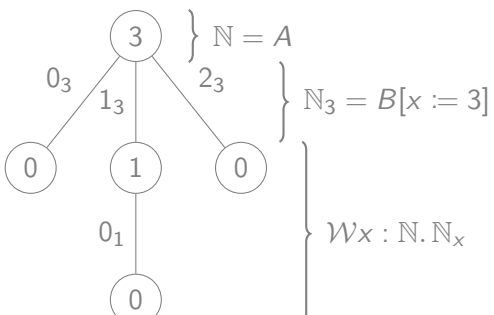


$$\frac{\Gamma \vdash A : *\qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash (\mathcal{W} x : A.\, B) : *}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[x := a] \to \mathcal{W} x : A.\, B}{\Gamma \vdash \text{sup}\ a\ b : \mathcal{W} x : A.\, B}$$

Term computation

$$\frac{\Gamma \vdash f : \Pi x : A.\, \Pi y : B\ x \to (\mathcal{W} x : A.\, B).\, (\Pi v : B\ x.\, C\ (y\ v)) \to C\ (\sup\ x\ y)}{\Gamma \vdash \mathbf{wrec}\ (\sup\ a\ b)\ f = f\ a\ b\ (\lambda v : B\ a.\, \mathbf{wrec}\ (b\ v)\ f) : C\ (\sup\ a\ b)}$$



$$\frac{\Gamma \vdash A : *\qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash (\mathcal{W} x : A.\, B) : *}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[x := a] \to \mathcal{W} x : A.\, B}{\Gamma \vdash \sup\ a\ b : \mathcal{W} x : A.\, B}$$

$\mathcal{W}$-types in Coq

We reuse the definition from Freek (also lecture 5, CIC):

```
Inductive W (A : Type) (B : A → Type) : Type :=
  | sup : ∀ x : A, (B x → W A B) → W A B.
Arguments sup {_ _} _ _ : assert.
```

equivalent to the formation and introduction rules that we saw before:

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : *}{\Gamma \vdash (\mathcal{W}x : A.\, B) : *} \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[x \coloneqq a] \to \mathcal{W}x : A.\, B}{\Gamma \vdash \mathsf{sup}\ a\ b : \mathcal{W}x : A.\, B}$$

$\mathcal{W}$-types in Coq: elimination

Then we get the following recursion principle:

```
Check W_rec : ∀ (A : Type) (B : A → Type) (P : W A B → Set),
  (∀ (x : A) (w : B x → W A B), (∀ b : B x, P (w b)) → P (sup x w)) →
  ∀ w : W A B, P w.
```

equivalent to the elimination rule we saw before:

$$\frac{\Gamma \vdash w : (\mathcal{W}x : A.\, B) \qquad \Gamma \vdash f : \Pi x : A.\, \Pi y : B\ x \to (\mathcal{W}x : A.\, B).\, (\Pi v : B\ x.\, C\ (y\ v)) \to C\ (\mathsf{sup}\ x\ y)}{\Gamma \vdash \mathbf{wrec}\ w\ f : C\ w}$$

$\mathcal{W}$-types in Coq: elimination computation

The computation rule we saw before is also automatically valid:

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B\ a \to (\mathcal{W}x : A.\ B) \qquad \Gamma \vdash f : \Pi x : A.\ \Pi y : B\ x \to (\mathcal{W}x : A.\ B).\ (\Pi v : B\ x.\ C\ (y\ v)) \to C\ (\mathsf{sup}\ x\ y)}{\Gamma \vdash \mathbf{wrec}\ (\mathsf{sup}\ a\ b)\ f = f\ a\ b\ (\lambda v : B\ a.\ \mathbf{wrec}\ (b\ v)\ f) : C\ (\mathsf{sup}\ a\ b)}$$

In Coq, the we can prove that it is correct by definitional equality:

```
Lemma W_equality A (B : A → Type) (C : W A B → Set) f (a : A) (b : B a → W A B) :
  W_rec A B C f (sup a b) = f a b (λ v, W_rec A B C f (b v)).
Proof. reflexivity. Qed.
```

Intermezzo: finite sets

$$\forall n \in \mathbb{N}.\, \mathbb{N}_n : *$$

Introduction rule:

$$0_n, 1_n, 2_n, \ldots, (n-1)_n : \mathbb{N}_n$$

Elimination rule:

$$\frac{c : \mathbb{N}_n \qquad c_0 : C\, 0_n \qquad \cdots \qquad c_{n-1} : C\, (n-1)_n}{\mathbf{R}_n\ c\ c_0\ \cdots\ c_{n-1} : C\, c}$$

Computation rule:

$$\frac{m_n : \mathbb{N}_n \qquad c_0 : C\, 0_n \qquad \cdots \qquad c_m : C\, m_n \qquad \cdots \qquad c_{n-1} : C\, (n-1)_n}{\mathbf{R}_n\ m_n\ c_0\ \cdots\ c_{n-1} = c_m : C\, m_n}$$

The elimination rule for $\mathbb{N}_0$:

$$\frac{c : \mathbb{N}_0}{\mathbf{R}_0\, c : C\ c} \quad \leftarrow \quad \text{N.B.: note that } Cc \text{ can be any type.}$$

Natural numbers as W-types

$$\mathbb{N}_{\mathcal{W}} \coloneqq \mathcal{W}x : \mathbb{N}_2.\, \mathbf{R}_2\, x\, \mathbb{N}_0\, \mathbb{N}_1$$

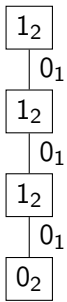
$$\mathbb{N}_{\mathcal{W}} \coloneqq \mathcal{W}x : \mathsf{bool}.\, \mathsf{if}\ x\ \mathsf{then}\ \mathbb{N}_2\ \mathsf{else}\ \bot$$

Inhabitant of the type

Encoding natural number $3 \in \mathbb{N}$ using the $\mathbb{N}_{\mathcal{W}}$:

$$\begin{array}{l}\mathsf{sup}\ 1_2\ (\lambda x_1 : \mathbb{N}_1. \\ \quad \mathsf{sup}\ 1_2\ (\lambda x_2 : \mathbb{N}_1. \\ \quad\quad \mathsf{sup}\ 1_2\ (\lambda x_3 : \mathbb{N}_1. \\ \quad\quad\quad \mathsf{sup}\ 0_2\ (\lambda x_4 : \mathbb{N}_0. \mathbf{R}_0\, x_4))))\end{array}$$

Or, represented as a tree (the root is at the top):



Typing rules

Consider the number 1 in $\mathbb{N}_{\mathcal{W}}$:

$$\mathit{one} \coloneqq \mathsf{sup}\ 1_2\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{sup}\ 0_2\ (\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2))$$

Type checking of the term one:

$$\dfrac{\dfrac{}{\Gamma \vdash 1_2 : \mathbb{N}_2} \quad \dfrac{\dfrac{\dfrac{}{\Gamma, x_1 : \mathbb{N}_1 \vdash 0_2 : \mathbb{N}_2} \quad \dfrac{\dfrac{}{\Gamma, x_1 : \mathbb{N}_1, x_2 : \mathbb{N}_0 \vdash \mathbf{R}_0 : \mathbb{N}_0 \to \mathbb{N}_{\mathcal{W}}} \quad \dfrac{}{\Gamma, x_1 : \mathbb{N}_1, x_2 : \mathbb{N}_0 \vdash x_2 : \mathbb{N}_0}}{\Gamma, x_1 : \mathbb{N}_1, x_2 : \mathbb{N}_0 \vdash \mathbf{R}_0\, x_2 : \mathbb{N}_{\mathcal{W}}}}{\Gamma, x_1 : \mathbb{N}_1 \vdash \lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2 : \mathbb{N}_0 \to \mathbb{N}_{\mathcal{W}}}}{\Gamma, x_1 : \mathbb{N}_1 \vdash \mathsf{sup}\ 0_2\ (\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2) : \mathbb{N}_{\mathcal{W}}}}{\Gamma \vdash \lambda x_1 : \mathbb{N}_1.\, \mathsf{sup}\ 0_2\ (\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2) : \mathbb{N}_1 \to \mathbb{N}_{\mathcal{W}}}}{\Gamma \vdash \mathsf{sup}\ 1_2\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{sup}\ 0_2\ (\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2)) : \mathbb{N}_{\mathcal{W}}}$$

Defining an addition operation for $\mathbb{N}_{\mathcal{W}}$

$$\mathsf{add} \coloneqq \lambda n\, m : \mathbb{N}_{\mathcal{W}}.\, \mathbf{wrec}\, n\, \mathsf{f}$$
$$\mathsf{f} \coloneqq \lambda x\, y\, z.\, \mathbf{R}_2\, x\, m\, (\mathsf{sup}\, 1_2\, z)$$

```
Definition Wnat_add' (n : Wnat) :
  Wnat → Wnat :=
    Wnat_rec _
      (λ (x : bool),
        match x with
        | true ⇒ λ w f, sup true f
        | false ⇒ λ _ _, n
        end).
```

```
Fixpoint Wnat_add (n m : Wnat) : Wnat :=
  match n with
  | sup true w ⇒
      sup true (λ _, Wnat_add (w tt) m)
  | sup false _ ⇒ m
  end.
```

```
Definition Wnat_add'' (n : Wnat) :
  Wnat → Wnat :=
    Wnat_rec _
      (λ (x : bool),
        bool_rec _
          (λ w f, sup true f)
          (λ _ _, n)
          x).
```

Computing $1 + 1$ in $\mathbb{N}_{\mathcal{W}}$

(recall $\mathsf{f} \coloneqq \lambda x\, y\, z.\, \mathbf{R}_2\, x\, m\, (\mathsf{sup}\, 1_2\, z)$)

$\mathsf{zero} \coloneqq \mathsf{sup}\ 0_2\ (\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2)$

$\mathsf{one} \coloneqq \mathsf{sup}\ 1_2\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})$

$\mathsf{add\ one\ one} = (\lambda m : \mathbb{N}_{\mathcal{W}}.\, \mathbf{wrec}\, \mathsf{one}\, \mathsf{f})\ \mathsf{one}$

$=$

Computing $1 + 1$ in $\mathbb{N}_{\mathcal{W}}$

(recall $\mathsf{f} \coloneqq \lambda x\, y\, z.\, \mathbf{R}_2\, x\, m\, (\sup 1_2\, z)$)

$\mathsf{zero} \coloneqq \sup\ 0_2\ (\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2)$

$\mathsf{one} \coloneqq \sup\ 1_2\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})$

$\mathsf{add\ one\ one} = (\lambda m : \mathbb{N}_{\mathcal{W}}.\, \mathbf{wrec}\, \mathsf{one}\, \mathsf{f})\ \mathsf{one}$

$= (\lambda m : \mathbb{N}_{\mathcal{W}}.\, \mathbf{wrec}\, (\sup\ 1_2\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero}))\, \mathsf{f})\ \mathsf{one}$

$= (\lambda m : \mathbb{N}_{\mathcal{W}}.\, \mathsf{f}\, 1_2\, (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\, (\lambda v : \mathbb{N}_1.\, \mathbf{wrec}\ ((\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ v)\ \mathsf{f}))\ \mathsf{one}$

$= (\lambda m.\, \mathbf{R}_2\, 1_2\, m\, (\sup\ 1_2\ (\lambda v : \mathbb{N}_1.\, \mathbf{wrec}\ ((\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ v)\ \mathsf{f})))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\lambda v : \mathbb{N}_1.\, \mathbf{wrec}\ ((\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ v)\ \mathsf{f}))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\mathbf{wrec}\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ \mathsf{f}))\ \mathsf{one}$

Computing $1 + 1$ in $\mathbb{N}_{\mathcal{W}}$

(recall $\mathsf{f} \coloneqq \lambda x\, y\, z.\, \mathbf{R}_2\, x\, m\, (\sup 1_2\, z)$)

$\mathsf{zero} \coloneqq \sup\ 0_2\ (\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2)$

$\mathsf{one} \coloneqq \sup\ 1_2\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})$

$\mathsf{add\ one\ one} = (\lambda m : \mathbb{N}_{\mathcal{W}}.\, \mathbf{wrec}\, \mathsf{one}\, \mathsf{f})\ \mathsf{one}$

$= (\lambda m : \mathbb{N}_{\mathcal{W}}.\, \mathbf{wrec}\, (\sup\ 1_2\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero}))\, \mathsf{f})\ \mathsf{one}$

$= (\lambda m : \mathbb{N}_{\mathcal{W}}.\, \mathsf{f}\ 1_2\ (\lambda x_1 : \mathbb{N}_1.\mathsf{zero})\ (\lambda v : \mathbb{N}_1.\, \mathbf{wrec}\ ((\lambda x_1 : \mathbb{N}_1.\mathsf{zero})\ v)\ \mathsf{f}))\ \mathsf{one}$

$= (\lambda m.\, \mathbf{R}_2\, 1_2\, m\, (\sup\ 1_2\ (\lambda v : \mathbb{N}_1.\, \mathbf{wrec}\ ((\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ v)\ \mathsf{f})))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\lambda v : \mathbb{N}_1.\, \mathbf{wrec}\ ((\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ v)\ \mathsf{f}))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\mathbf{wrec}\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ \mathsf{f}))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\mathsf{f}\ 0_2\ (\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2)\ (\lambda v : \mathbb{N}_0.\, (\mathbf{wrec}\ ((\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2)\ v)\ \mathsf{f}))))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\mathbf{R}_2\, 0_2\, m\, (\sup\ 1_2\ (\lambda v : \mathbb{N}_0.\, \mathbf{wrec}\ ((\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2)\ v)\ \mathsf{f}))))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\lambda x_0 : \mathbb{N}_1.\, m))\ \mathsf{one}$

$= \sup\ 1_2\ \lambda x_0 : \mathbb{N}_1.\, \mathsf{one}$

Computing $1 + 1$ in $\mathbb{N}_{\mathcal{W}}$

(recall $\mathsf{f} \coloneqq \lambda x\, y\, z.\, \mathbf{R}_2\, x\, m\, (\sup 1_2\, z)$)

$\mathsf{zero} \coloneqq \sup\ 0_2\ (\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2)$

$\mathsf{one} \coloneqq \sup\ 1_2\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})$

$\mathsf{add\ one\ one} = (\lambda m : \mathbb{N}_{\mathcal{W}}.\, \mathbf{wrec}\, \mathsf{one}\, \mathsf{f})\ \mathsf{one}$

$= (\lambda m : \mathbb{N}_{\mathcal{W}}.\, \mathbf{wrec}\, (\sup\ 1_2\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero}))\, \mathsf{f})\ \mathsf{one}$

$= (\lambda m : \mathbb{N}_{\mathcal{W}}.\, \mathsf{f}\ 1_2\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ (\lambda v : \mathbb{N}_1.\, \mathbf{wrec}\ ((\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ v)\ \mathsf{f}))\ \mathsf{one}$

$= (\lambda m.\, \mathbf{R}_2\, 1_2\, m\, (\sup\ 1_2\ (\lambda v : \mathbb{N}_1.\, \mathbf{wrec}\ ((\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ v)\ \mathsf{f})))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\lambda v : \mathbb{N}_1.\, \mathbf{wrec}\ ((\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ v)\ \mathsf{f}))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\mathbf{wrec}\ (\lambda x_1 : \mathbb{N}_1.\, \mathsf{zero})\ \mathsf{f}))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\mathsf{f}\ 0_2\ (\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2)\ (\lambda v : \mathbb{N}_0.\, (\mathbf{wrec}\ ((\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2)\ v)\ \mathsf{f}))))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\mathbf{R}_2\, 0_2\, m\, (\sup\ 1_2\ (\lambda v : \mathbb{N}_0.\, \mathbf{wrec}\ ((\lambda x_2 : \mathbb{N}_0.\, \mathbf{R}_0\, x_2)\ v)\ \mathsf{f}))))\ \mathsf{one}$

$= (\lambda m.\, \sup\ 1_2\ (\lambda x_0 : \mathbb{N}_1.\, m))\ \mathsf{one}$

$= \sup\ 1_2\ \lambda x_0 : \mathbb{N}_1.\, \mathsf{one}$

Intermezzo: ordinal numbers

Introduction rules:

$$0 : \mathcal{O} \qquad \frac{\alpha : \mathcal{O}}{\alpha' : \mathcal{O}} \qquad \frac{b : \mathbb{N} \to \mathcal{O}}{\sup\ b : \mathcal{O}}$$

Elimination rule:

$$\frac{e : (\Pi x : \mathcal{O}.\, \Pi y : C\,x.\, C\,x') \qquad \begin{array}{c} c : \mathcal{O} \qquad d : C\ 0 \\ f : \Pi z : (\mathbb{N} \to \mathcal{O}).\, \Pi w : (\Pi n : \mathbb{N}.\, C\,(z\,n)).\, C\ (\sup z) \end{array}}{\mathbf{ordrec}\ c\ d\ e\ f : C\,c}$$

Term Computation:

$$\frac{\ldots}{\mathbf{ordrec}\ 0\ d\ e\ f = d : C\ 0} \qquad \frac{\ldots}{\mathbf{ordrec}\ \alpha'\ d\ e\ f = e\ \alpha\ (\mathbf{ordrec}\ \alpha\ d\ e\ f) : C\ \alpha'}$$

$$\frac{\ldots}{\mathbf{ordrec}\ (\sup\ b)\ d\ e\ f = f\ b\ (\lambda x.\, \mathbf{ordrec}\ (b\,x)\ d\ e\ f) : C\ (\sup\ b)}$$

Ordinal numbers as a $\mathcal{W}$-type

$$\mathcal{O}_{\mathcal{W}} \coloneqq \mathcal{W}x : \mathbb{N}_3.\, \mathbf{R}_3\, x\, \mathbb{N}_0\, \mathbb{N}_1\, \mathbb{N}$$

Zero ordinal:

$$0 := \mathsf{sup}\ 0_3\ (\lambda x : \mathbb{N}_0.\, \mathbf{R}_0\, x_0)$$

Successor:

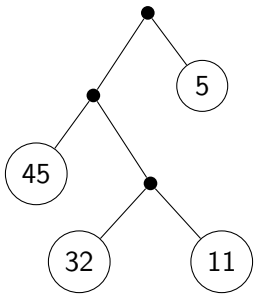
$$\mathsf{S} \coloneqq \lambda\alpha : \mathcal{O}_{\mathcal{W}}.\, \mathsf{sup}\ 1_3\ (\lambda x : \mathbb{N}_1.\, \alpha)$$

Supremum:

$$\begin{aligned} \mathsf{lub} \quad \coloneqq \quad & \lambda\alpha : \mathcal{O}_{\mathcal{W}}. \\ & (\mathsf{sup}\ 2_3\ (\lambda x : \mathbb{N}.\, \mathbf{natrec}\ x\ \alpha\ (\lambda a : \mathbb{N}.\, \lambda b : \mathcal{O}_{\mathcal{W}}.\, \mathsf{S}\ b))) \end{aligned}$$

Binary trees

Consider a binary tree with $\mathbb{N}$s at the leaves

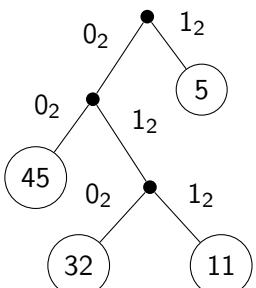


How could we turn this into a $\mathcal{W}$-type?

Recall $\mathsf{sup} : A \to (B\ x \to \mathcal{W}x : A.\ B) \to \mathcal{W}x : A.\ B$

Binary trees as a $\mathcal{W}$-type

Let us take edge labels in $\mathbb{N}_2$:



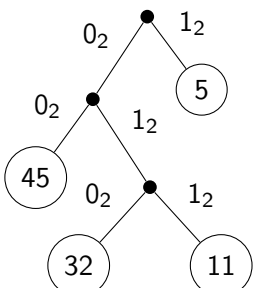
Now, the root node of every subtree either

i) has no value and has two descendants

ii) has a value but has no descendants (it is a leaf)

Binary trees as a $\mathcal{W}$-type

Let us take edge labels in $\mathbb{N}_2$:



Now, the root node of every subtree either

i) has no value and has two descendants

ii) has a value but has no descendants (it is a leaf)

So the node value must depend on the node type, and so must the amount of descendants.

Intermezzo: dependent pairs

Perhaps you recall (lecture 5, CIC):

```
Inductive sigT (A : Type) (P : A → Type) : Type :=
  | existT : ∀ x : A, P x → sigT A P.
Arguments sigT [_] _.
```

In Coq, `@sigT A (λ x : A, B).` is usually written as `{ x : A & B }`.
For example:

```
Definition test := { x : bool & if x then nat else string }.

Check existT _ true 0 : test.
Check existT _ false "asdf" : test.
Fail Check existT _ true "asdf" : test.
Fail Check existT _ false 0 : test.
```

Intermezzo: dependent pairs

We will use the notation $\Sigma x : A.\, B$. Then

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[x := a]}{\Gamma \vdash (a, b) : \Sigma x : A.\, B}$$

We have the eliminator ~~**srec**~~ for Σ-types:

$$\frac{\Gamma \vdash c : (\Sigma x : A.\, B) \qquad \Gamma \vdash d : \Pi a : A.\, \Pi b : B[x := a].\, C\ a\ b}{\Gamma \vdash \mathbf{srec}\ c\ d : C\ c}$$

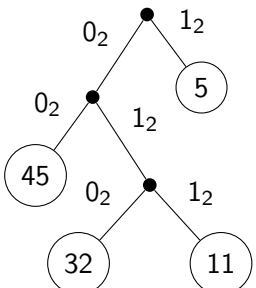
So, for example, we can project out the components:

$$\pi_1 = \lambda c.\, \mathbf{srec}\ c\ (\lambda xy.\, x)$$
$$\pi_2 = \lambda c.\, \mathbf{srec}\ c\ (\lambda xy.\, y)$$

Binary $\mathbb{N}$ trees as a $\mathcal{W}$-type

$$\text{Node type } t \coloneqq \Sigma x : \mathbb{N}_2.\, \mathbf{R}_2\, x\, \mathbb{N}\, \mathbb{N}_1 \quad \sim \quad \Sigma x : \text{bool}.\, \text{if } x \text{ then unit else } \mathbb{N}$$

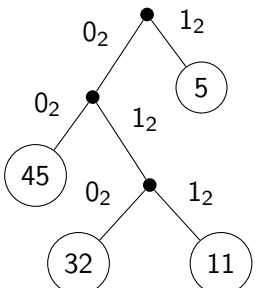
$$\text{Tree type } \mathcal{T} \coloneqq \mathcal{W} x : t.\, \mathbf{R}_2\, (\pi_1\ x)\, \mathbb{N}_0\, \mathbb{N}_2 \quad \sim \quad \mathcal{W} x : t.\, \text{if } \pi_1\ x \text{ then } \mathbb{N}_2 \text{ else } \bot$$

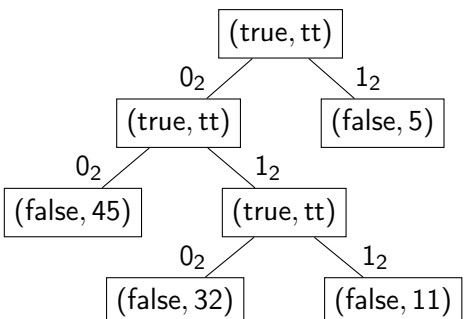


Binary $\mathbb{N}$ trees as a $\mathcal{W}$-type

$$\text{Node type } t \coloneqq \Sigma x : \mathbb{N}_2.\, \mathbf{R}_2\, x\, \mathbb{N}\, \mathbb{N}_1 \quad \sim \quad \Sigma x : \text{bool}.\, \text{if } x \text{ then unit else } \mathbb{N}$$

$$\text{Tree type } \mathcal{T} \coloneqq \mathcal{W} x : t.\, \mathbf{R}_2\, (\pi_1\ x)\, \mathbb{N}_0\, \mathbb{N}_2 \quad \sim \quad \mathcal{W} x : t.\, \text{if } \pi_1\ x \text{ then } \mathbb{N}_2 \text{ else } \bot$$

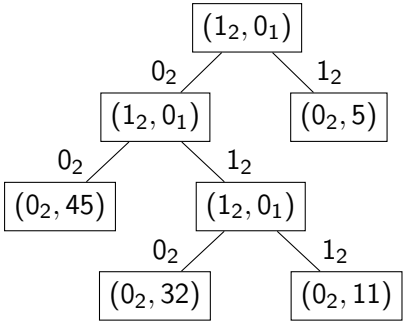


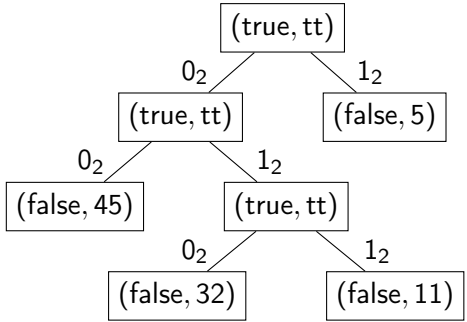


Binary $\mathbb{N}$ trees as a $\mathcal{W}$-type

$$\text{Node type } t \coloneqq \Sigma x : \mathbb{N}_2.\, \mathbf{R}_2\, x\, \mathbb{N}\, \mathbb{N}_1 \quad \sim \quad \Sigma x : \text{bool}.\, \text{if } x \text{ then unit else } \mathbb{N}$$

$$\text{Tree type } \mathcal{T} \coloneqq \mathcal{W} x : t.\, \mathbf{R}_2\, (\pi_1\ x)\, \mathbb{N}_0\, \mathbb{N}_2 \quad \sim \quad \mathcal{W} x : t.\, \text{if } \pi_1\ x \text{ then } \mathbb{N}_2 \text{ else } \bot$$

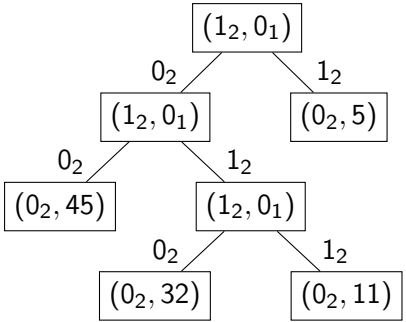


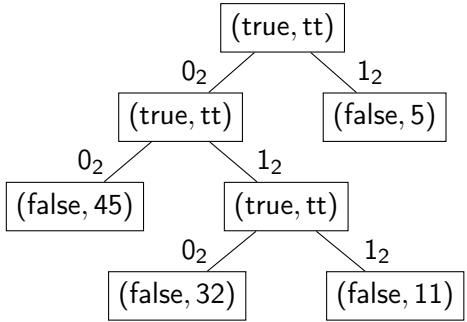


Binary $\mathbb{N}$ trees as a $\mathcal{W}$-type

$$\text{Node type } t \coloneqq \Sigma x : \mathbb{N}_2.\, \mathbf{R}_2\, x\, \mathbb{N}\, \mathbb{N}_{\mathcal{W}}\, \mathbb{N}_1 \quad \sim \quad \Sigma x : \text{bool}.\, \text{if } x \text{ then unit else } \mathbb{N}$$

$$\text{Tree type } \mathcal{T} \coloneqq \mathcal{W} x : t.\, \mathbf{R}_2\, (\pi_1\ x)\, \mathbb{N}_0\, \mathbb{N}_2 \quad \sim \quad \mathcal{W} x : t.\, \text{if } \pi_1\ x \text{ then } \mathbb{N}_2 \text{ else } \bot$$





Thank you for listening!

Questions?