SIZED TYPES

TOWARDS TYPE-BASED TERMINATION THAT DOESN'T GET IN THE WAY OF EQUALITY

Andreas **Abel**
Théo **Winterhalter**

CHALMERS

WHY SIZED TYPES?

SIZES CAN GET IN THE WAY OF EQUALITY

OUR PROPOSAL

WHY SIZED TYPES? STRUCTURAL ORDER LACKS COMPOSITIONALITY

```
sub : ℕ → ℕ → ℕ
sub zero     m        = zero
sub (suc n) zero     = suc n
sub (suc n) (suc m) = sub n m
```

WHY SIZED TYPES? STRUCTURAL ORDER LACKS COMPOSITIONALITY

```
div : ℕ → ℕ → ℕ
div zero m = zero
div (suc n) m = suc (div (sub n m) m)
```

WHY SIZED TYPES? STRUCTURAL ORDER LACKS COMPOSITIONALITY

```
div : ℕ → ℕ → ℕ
div zero m = zero
div (suc n) m = suc (div (sub n m) m)
```

Termination checking failed!

WHY SIZED TYPES? THE NATURAL NUMBERS EXAMPLE

```
data Nat : Size → Set where
  zero : ∀ i → Nat (↑i)
  suc  : ∀ j → (n : Nat j) → Nat (↑j)
```

WHY SIZED TYPES? THE NATURAL NUMBERS EXAMPLE

```
data Nat : Size → Set where
  zero : ∀ i → Nat (↑i)
  suc  : ∀ j → (n : Nat j) → Nat (↑j)
```

```
sub : ∀ i → Nat i → Nat ∞ → Nat i
sub .(↑i) (zero i)  m          = (zero i)
sub .(↑i) (suc i n) (zero ∞)   = suc i n
sub .(↑i) (suc i n) (suc ∞ m) = sub i n m
```

WHY SIZED TYPES? THE NATURAL NUMBERS EXAMPLE

```
data Nat : Size → Set where
  zero : ∀ i → Nat (↑i)
  suc  : ∀ j → (n : Nat j) → Nat (↑j)
```

```
sub : ∀ i → Nat i → Nat ∞ → Nat i
```

```
div : ∀ i → Nat i → Nat ∞ → Nat i
div .(↑i) (zero i)   m = (zero i)
div .(↑i) (suc i n) m = suc i (div i (sub i n m) m)
```

WHY SIZED TYPES?

SIZES CAN GET IN THE WAY OF EQUALITY

OUR PROPOSAL

SIZES IN THE WAY OF EQUALITY EXAMPLE

```
gscale : (∀ i → Nat i → Nat i) → (Nat ∞ → Nat ∞) →
         ∀ i → Nat i → Nat ∞
gscale f g .(↑i) (zero i)   = zero ∞
gscale f g .(↑i) (suc i n) = g (suc ∞ (gscale f g i (f i n)))
```

SIZES IN THE WAY OF EQUALITY EXAMPLE

```
gscale : (∀ i → Nat i → Nat i) → (Nat ∞ → Nat ∞) →
         ∀ i → Nat i → Nat ∞
gscale f g .(↑i) (zero i)   = zero ∞
gscale f g .(↑i) (suc i n) = g (suc ∞ (gscale f g i (f i n)))
```

Generalization of `div`

```
div : ∀ i → Nat i → Nat ∞ → Nat i
div .(↑i) (zero i)   m = (zero i)
div .(↑i) (suc i n) m = suc i (div i (sub i n m) m)
```

SIZES IN THE WAY OF EQUALITY EXAMPLE

```
gscale : (∀ i → Nat i → Nat i) → (Nat ∞ → Nat ∞) →
         ∀ i → Nat i → Nat ∞
gscale f g .(↑i) (zero i)   = zero ∞
gscale f g .(↑i) (suc i n) = g (suc ∞ (gscale f g i (f i n)))
```

Generalization of `div`

```
div : ∀ i → Nat i → Nat ∞ → Nat i
div .(↑i) (zero i)   m = (zero i)
div .(↑i) (suc i n) m = suc i (div i (sub i n m) m)
```

```
div ≅ gscale sub (λ x → x)
```

SIZES IN THE WAY OF EQUALITY EXAMPLE

```
gscale : (∀ i → Nat i → Nat i) → (Nat ∞ → Nat ∞) →
         ∀ i → Nat i → Nat ∞
gscale f g .(↑i) (zero i)  = zero ∞
gscale f g .(↑i) (suc i n) = g (suc ∞ (gscale f g i (f i n)))
```

SIZES IN THE WAY OF EQUALITY EXAMPLE

```
gscale : (∀ i → Nat i → Nat i) → (Nat ∞ → Nat ∞) →
         ∀ i → Nat i → Nat ∞
gscale f g .(↑i) (zero i)   = zero ∞
gscale f g .(↑i) (suc i n) = g (suc ∞ (gscale f g i (f i n)))
```

```
scale1 = gscale (λ _ x → x) (λ x → x)
```

SIZES IN THE WAY OF EQUALITY EXAMPLE

```
gscale : (∀ i → Nat i → Nat i) → (Nat ∞ → Nat ∞) →
         ∀ i → Nat i → Nat ∞
gscale f g .(↑i) (zero i)   = zero ∞
gscale f g .(↑i) (suc i n) = g (suc ∞ (gscale f g i (f i n)))
```

```
scale1 = gscale (λ _ x → x) (λ x → x)
```

```
scale1-id : ∀ i → (n : Nat i) → scale1 i n ≡ n
scale1-id .(↑i) (zero i)   = ?1
scale1-id .(↑i) (suc i n) = ?2
```

SIZES IN THE WAY OF EQUALITY EXAMPLE

```
gscale : (∀ i → Nat i → Nat i) → (Nat ∞ → Nat ∞) →
         ∀ i → Nat i → Nat ∞
gscale f g .(↑i) (zero i)   = zero ∞
gscale f g .(↑i) (suc i n) = g (suc ∞ (gscale f g i (f i n)))
```

```
scale1 = gscale (λ _ x → x) (λ x → x)
```

```
scale1-id : ∀ i → (n : Nat i) → scale1 i n ≡ n
scale1-id .(↑i) (zero i)   = ?₁
scale1-id .(↑i) (suc i n) = ?₂
```

```
goal₁ zero ∞ ≡ zero i
```

WHY SIZED TYPES?

SIZES CAN GET IN THE WAY OF EQUALITY

OUR PROPOSAL

OUR PROPOSAL FIRST IDEA

Remove sizes from terms

OUR PROPOSAL FIRST IDEA

Remove sizes from terms

Handle them through subtyping

OUR PROPOSAL FIRST IDEA

Remove sizes from terms ⇒ irrelevance for free

Handle them through subtyping

OUR PROPOSAL FIRST IDEA

Remove sizes from terms ⇒ irrelevance for free

Handle them through subtyping ⇒ overhead with the proofs

OUR PROPOSAL FIRST IDEA

Remove sizes from terms

⇒ irrelevance for free

Handle them through subtyping

⇒ overhead with the proofs

⇒ no arbitrary rank size quantification

OUR PROPOSAL CUMULATIVITY AND BUILT-IN IRRELEVANCE

We feature

Arbitrary rank (first-class) size quantification

OUR PROPOSAL CUMULATIVITY AND BUILT-IN IRRELEVANCE

We feature

Arbitrary rank (first-class) size quantification

Natural numbers with sizes irrelevant for equality

OUR PROPOSAL CUMULATIVITY AND BUILT-IN IRRELEVANCE

We feature

Arbitrary rank (first-class) size quantification

Natural numbers with sizes irrelevant for equality

```
zero a ≡ zero b : Nat c   for a,b < c
```

OUR PROPOSAL CUMULATIVITY AND BUILT-IN IRRELEVANCE

We feature

Arbitrary rank (first-class) size quantification

Natural numbers with sizes irrelevant for equality

$$\texttt{zero}\ a \equiv \texttt{zero}\ b : \texttt{Nat}\ c \quad \text{for } a, b < c$$

Cumulativity for Sized Types

$$\frac{a \leq b}{\texttt{Nat}\ a \leq \texttt{Nat}\ b}$$

OUR PROPOSAL CUMULATIVITY AND BUILT-IN IRRELEVANCE

We feature

Arbitrary rank (first-class) size quantification

Natural numbers with sizes irrelevant for equality

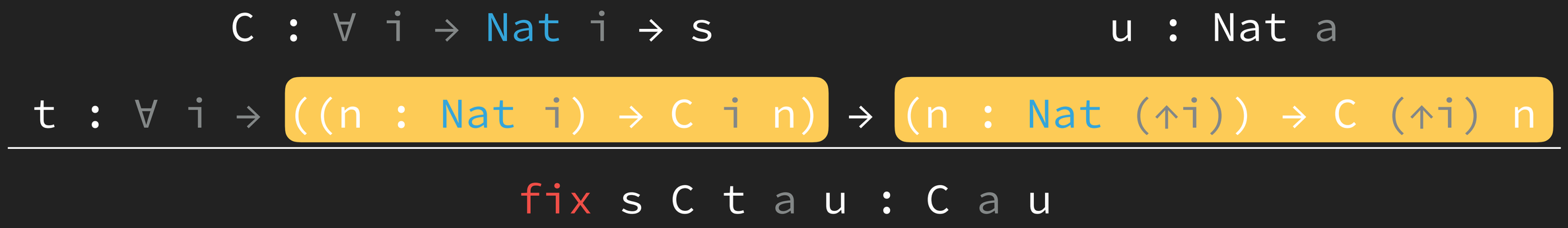
$$\texttt{zero}\ a \equiv \texttt{zero}\ b : \texttt{Nat}\ c \quad \text{for } a, b < c$$

Cumulativity for Sized Types

$$\frac{a \leq b}{\texttt{Nat}\ a \leq \texttt{Nat}\ b} \qquad \frac{U = U' \qquad x{:}U \vdash T \leq T'}{(x{:}U) \to T \leq (x{:}U') \to T'}$$

OUR PROPOSAL TERMINATION

Ensured by fixed point operator



OUR PROPOSAL TERMINATION

Ensured by fixed point operator

$$\frac{C : \forall\ i \to \mathsf{Nat}\ i \to s \qquad u : \mathsf{Nat}\ a \qquad t : \forall\ i \to ((n : \mathsf{Nat}\ i) \to C\ i\ n) \to (n : \mathsf{Nat}\ (\uparrow i)) \to C\ (\uparrow i)\ n}{\mathsf{fix}\ s\ C\ t\ a\ u : C\ a\ u}$$

```
fix s C t (↑a) (zero b)  ↦ t a (λ (n : Nat a) → fix s C t a n) (zero b)
```

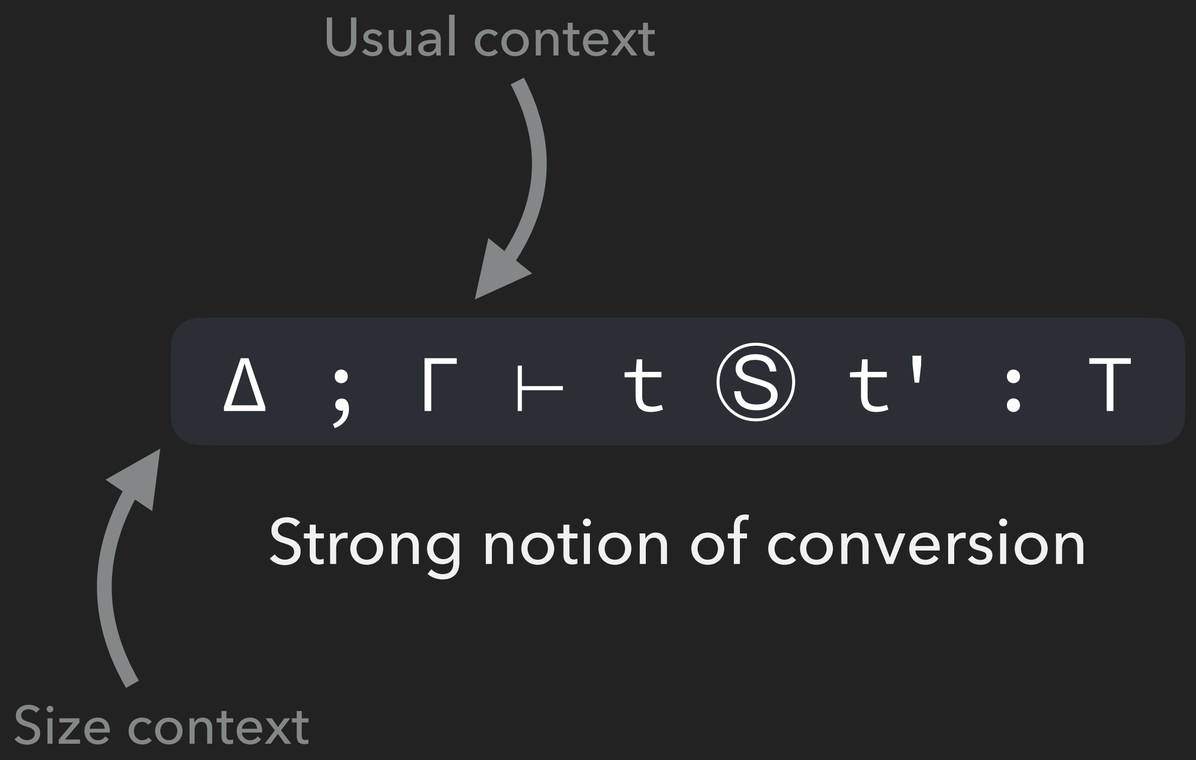
OUR PROPOSAL TERMINATION

Ensured by fixed point operator

$$\frac{C : \forall\ i \to \mathsf{Nat}\ i \to s \qquad u : \mathsf{Nat}\ a \qquad t : \forall\ i \to ((n : \mathsf{Nat}\ i) \to C\ i\ n) \to (n : \mathsf{Nat}\ (\uparrow i)) \to C\ (\uparrow i)\ n}{\mathsf{fix}\ s\ C\ t\ a\ u : C\ a\ u}$$

```
fix s C t (↑a) (zero b)  ↦ t a (λ (n : Nat a) → fix s C t a n) (zero b)
fix s C t (↑a) (suc b u) ↦ t a (λ (n : Nat a) → fix s C t a n) (suc b u)
```

OUR PROPOSAL MODEL / KRIPKE LOGICAL RELATION



OUR PROPOSAL KRIPKE LOGICAL RELATION (OUTLINE)

$$\frac{\forall\ \Delta' \leq \Delta,\ \Delta' \vdash a \Rightarrow \Delta' \ ;\ \Gamma \vdash f\ a \mathbin{\widehat{\textcircled{S}}} f'\ a : T[a/i]}{\Delta \ ;\ \Gamma \vdash f \mathbin{\textcircled{S}} f' : \forall\ i \to T}$$

OUR PROPOSAL KRIPKE LOGICAL RELATION (OUTLINE)

$$\frac{\frac{T \searrow A \quad \Delta;\Gamma \vdash T = A \qquad t \searrow a \quad t' \searrow a'}{\Delta;\Gamma \vdash t = a : A \quad \Delta;\Gamma \vdash t' = a' : A}}{\Delta;\Gamma \vdash a \mathbin{Ⓢ} a' : A}}{\Delta;\Gamma \vdash t :=: t' : T}$$

$$\Delta ; \Gamma \vdash t \mathbin{\widehat{Ⓢ}} t' : T$$

OUR PROPOSAL KRIPKE LOGICAL RELATION (OUTLINE)

$$\frac{\frac{\frac{\frac{T \searrow A \qquad \Delta;\Gamma \vdash T = A}{t \searrow a \qquad t' \searrow a'}}{\Delta;\Gamma \vdash t = a : A \qquad \Delta;\Gamma \vdash t' = a' : A}}{\Delta;\Gamma \vdash a \mathbin{Ⓢ} a' : A}}{\Delta;\Gamma \vdash t :=: t' : T}$$

$$\Delta ; \Gamma \vdash t \mathbin{\widehat{Ⓢ}} t' : T$$

⇒ normalization and subject reduction

OUR PROPOSAL ALGORITHMIC EQUALITY (OUTLINE)

Type-directed Equality

$$\frac{\Delta;\Gamma,\ x{:}U \vdash t\ x \Leftrightarrow t'\ x : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : (x{:}U) \to T}$$

OUR PROPOSAL ALGORITHMIC EQUALITY (OUTLINE)

Type-directed Equality

$$\frac{\Delta;\Gamma,\ x{:}U \vdash t\ x \Leftrightarrow t'\ x : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : (x{:}U) \to T} \qquad \frac{\Delta,i;\Gamma \vdash t\ i \Leftrightarrow t'\ i : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : \forall\ i \to T}$$

OUR PROPOSAL ALGORITHMIC EQUALITY (OUTLINE)

Type-directed Equality

$$\frac{\Delta;\Gamma,\ x{:}U \vdash t\ x \Leftrightarrow t'\ x : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : (x{:}U) \to T} \qquad \frac{\Delta,i;\Gamma \vdash t\ i \Leftrightarrow t'\ i : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : \forall\ i \to T} \qquad \ldots$$

OUR PROPOSAL ALGORITHMIC EQUALITY (OUTLINE)

Type-directed Equality

$$\frac{\Delta;\Gamma,\ x{:}U \vdash t\ x \Leftrightarrow t'\ x : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : (x{:}U) \to T} \qquad \frac{\Delta,i;\Gamma \vdash t\ i \Leftrightarrow t'\ i : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : \forall\ i \to T} \qquad \cdots$$

Structural Equality for neutrals

$$\frac{(x{:}U) \in \Gamma}{\Delta;\Gamma \vdash x \Leftrightarrow x : U}$$

OUR PROPOSAL ALGORITHMIC EQUALITY (OUTLINE)

Type-directed Equality

$$\frac{\Delta;\Gamma,\ x{:}U \vdash t\ x \Leftrightarrow t'\ x : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : (x{:}U) \to T} \qquad \frac{\Delta,i;\Gamma \vdash t\ i \Leftrightarrow t'\ i : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : \forall\ i \to T} \qquad \ldots$$

Structural Equality for neutrals

$$\frac{(x{:}U) \in \Gamma}{\Delta;\Gamma \vdash x \Leftrightarrow x : U} \qquad \frac{\Delta;\Gamma \vdash n \Leftrightarrow n' : (x{:}U) \to T \qquad \Delta;\Gamma \vdash u \Leftrightarrow u' : U}{\Delta;\Gamma \vdash n\ u \Leftrightarrow n'\ u' : T[u/x]}$$

OUR PROPOSAL ALGORITHMIC EQUALITY (OUTLINE)

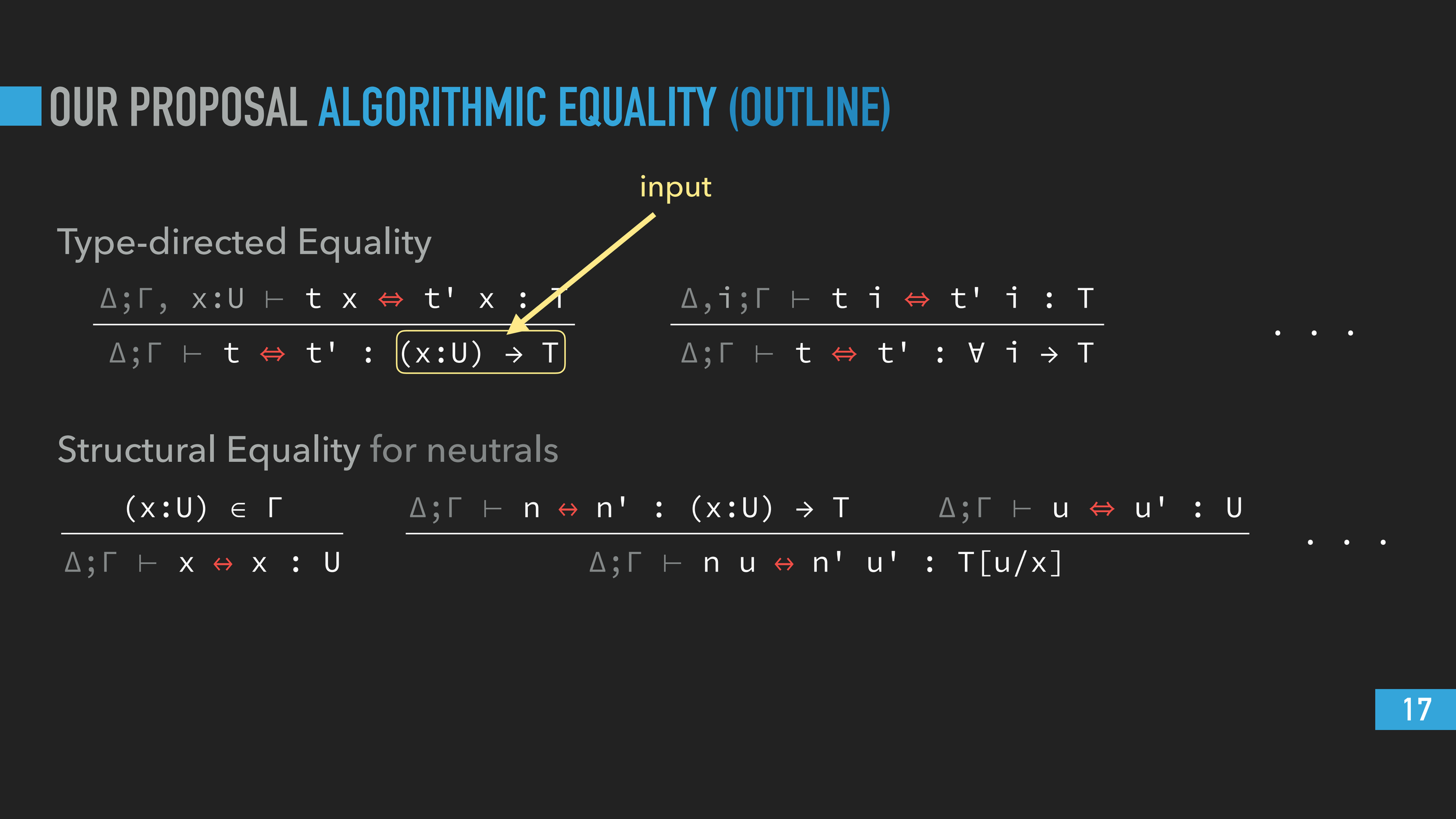
Type-directed Equality

$$\frac{\Delta;\Gamma,\ x{:}U \vdash t\ x \Leftrightarrow t'\ x : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : (x{:}U) \to T} \qquad \frac{\Delta,i;\Gamma \vdash t\ i \Leftrightarrow t'\ i : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : \forall\ i \to T} \qquad \ldots$$

Structural Equality for neutrals

$$\frac{(x{:}U) \in \Gamma}{\Delta;\Gamma \vdash x \Leftrightarrow x : U} \qquad \frac{\Delta;\Gamma \vdash n \Leftrightarrow n' : (x{:}U) \to T \qquad \Delta;\Gamma \vdash u \Leftrightarrow u' : U}{\Delta;\Gamma \vdash n\ u \Leftrightarrow n'\ u' : T[u/x]} \qquad \ldots$$

OUR PROPOSAL ALGORITHMIC EQUALITY (OUTLINE)



Type-directed Equality

$$\frac{\Delta;\Gamma,\ x{:}U \vdash t\ x \Leftrightarrow t'\ x : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : (x{:}U) \to T} \qquad \frac{\Delta,i;\Gamma \vdash t\ i \Leftrightarrow t'\ i : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : \forall\ i \to T} \qquad \cdots$$

Structural Equality for neutrals

$$\frac{(x{:}U) \in \Gamma}{\Delta;\Gamma \vdash x \Leftrightarrow x : U} \qquad \frac{\Delta;\Gamma \vdash n \Leftrightarrow n' : (x{:}U) \to T \qquad \Delta;\Gamma \vdash u \Leftrightarrow u' : U}{\Delta;\Gamma \vdash n\ u \Leftrightarrow n'\ u' : T[u/x]} \qquad \cdots$$

OUR PROPOSAL ALGORITHMIC EQUALITY (OUTLINE)

input

Type-directed Equality

$$\frac{\Delta;\Gamma,\ x{:}U \vdash t\ x \Leftrightarrow t'\ x : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : (x{:}U) \to T} \qquad \frac{\Delta,i;\Gamma \vdash t\ i \Leftrightarrow t'\ i : T}{\Delta;\Gamma \vdash t \Leftrightarrow t' : \forall\ i \to T} \qquad \cdots$$

Structural Equality for neutrals

$$\frac{(x{:}U) \in \Gamma}{\Delta;\Gamma \vdash x \Leftrightarrow x : U} \qquad \frac{\Delta;\Gamma \vdash n \Leftrightarrow n' : (x{:}U) \to T \qquad \Delta;\Gamma \vdash u \Leftrightarrow u' : U}{\Delta;\Gamma \vdash n\ u \Leftrightarrow n'\ u' : T[u/x]} \qquad \cdots$$

output

CONCLUSION

Decidable type system with

CONCLUSION

Decidable type system with

Sized dependent types

CONCLUSION

Decidable type system with

Sized dependent types

Type-based termination checking

CONCLUSION

Decidable type system with

Sized dependent types

Type-based termination checking

Only `Nat` for now

CONCLUSION

Decidable type system with

Sized dependent types

Type-based termination checking

Only `Nat` for now

Future work

CONCLUSION

Decidable type system with

Sized dependent types

Type-based termination checking

Only `Nat` for now

Future work

Inductives / Pattern-matching

CONCLUSION

Decidable type system with

Sized dependent types

Type-based termination checking

Only `Nat` for now

Future work

Inductives / Pattern-matching

CoInductives / Productivity

CONCLUSION

Decidable type system with

Sized dependent types

Type-based termination checking

Only `Nat` for now

Future work

Inductives / Pattern-matching

CoInductives / Productivity

HoTT