for negative connectives

- Interpreted as right adjoint functions
  - \( *I \) := \( b \)
  - \( *E \) := \( e \)
  - \( *\Rightarrow \) := \( \beta \)
  - \( *\Leftarrow \) := identity expansion

---

**Bob Harper, Day 4**

**Proof-relevant equality**

\[ \text{EQ} : \text{Nat} \rightarrow \text{Nat} \rightarrow \text{U} \]

- spec: \( \text{EQ} \) \( m \rightarrow n \equiv \lambda (n) \) \( \text{diff} \)
- informal: \( \text{EQ} \) \( m \rightarrow n \equiv 0 \) \( \text{nil} \) \( \text{inhabited} \)
- EQ is inhabited when \( m \) is \( n \).

**Question:** Given \( f : \text{Nat} \rightarrow \text{Nat} \) is it the case that for \( M, N : \text{Nat} \)

- \( f \) is mathematical function
- \( \text{EQ} \) \( (M, N) \rightarrow \text{EQ} (f M, f N) \)?

- **Yes** \( x, y : \text{Nat} \rightarrow \text{U} \)

\[ \text{EQ} (x, y) \rightarrow \text{EQ} (f x, f y) \]

**Proof:** double induction on \( x, y \).
Adjunctions

Antiparallel functors $F$ and $G$ form an adjunction ("$F \rightleftharpoons G$") if for any $A : C, B : D$ there is a natural bijection of hom sets:

$$\text{bijection } \Phi \left( \frac{C(A \rightarrow G(B))}{D(F(A) \rightarrow B)} \right).$$

- naturality
  $$\begin{array}{ccc}
  C : A & \xrightarrow{a} & A & \xrightarrow{f \circ b} & G(B) & \xrightarrow{G(b)} & G(B') \\
  D : F(A') & \xrightarrow{F(a)} & F(A) & \xrightarrow{f} & B & \xrightarrow{b} & B'
  \end{array}$$

$F : C \rightarrow D$ and $G : D \rightarrow C$ form $F \rightleftharpoons G$ if

**Universal property of unit:**

\[ \exists \eta : \text{id}(C) \rightarrow F : G \]

\[ \forall f : C(A \rightarrow G(B)) \]

\[ \exists ! g : D(F(A) \rightarrow B) \]

\[ g(\eta_A \cdot G(g)) = f \]

**Universal property of counit:**

\[ \exists \varepsilon : G \cdot F \rightarrow \text{id}(D) \]

\[ \forall g : D(F(A) \rightarrow B) \]

\[ \exists ! f : C(A \rightarrow G(B)) \]

\[ F(f) \cdot \varepsilon(B) = g \]

Local soundness from this property:
Case $x = \text{zero} = y$:

\[
\text{if } x \neq y \text{ then refl (f zero)}.
\]

\[
\text{EQ}_\text{Nat}(\text{zero}, \text{zero}) = T \iff \text{aka unit aka } 1
\]

Case $x = \text{zero}, y = \text{suc}(-)$:

\[
\text{EQ}_\text{Nat}(\text{zero, succ } -) = 0 \iff \text{aka void aka } \bot
\]

\[
\text{if } x < 0 \text{ then abort (s) } \iff \text{abort is the only term that is } \bot
\]

Case $x = \text{suc}(-), y = \text{zero}$:

\[
\text{EQ}_\text{Nat}(\text{suc }, \text{zero}) = 0
\]

\[
\text{if } x > 0 \text{ then abort (s) }
\]

\[
x \text{ = suc } (x'), y \text{ = suc } (y')\iff \text{EQ}_\text{Nat}(\text{suc } (x'), \text{ suc } (y')) = \text{EQ}_\text{Nat}(x, y)
\]

**IH:**

**TS:**

**Suppose** $f : N \rightarrow N$

\[
p : \text{EQ}_\text{Nat}(x, y) = \text{EQ}_\text{Nat}(x', y')
\]

\[
\text{Take } f \text{ in } \text{IH} \text{ to be } f \circ \text{suc}
\]

\[
\Rightarrow \text{IH } (f \circ \text{suc})(p) : \text{EQ}_\text{Nat}((f \circ \text{suc})(x'), (f \circ \text{suc})(y'))
\]

\[
\text{TS} : \text{EQ}_\text{Nat}(f x, f y)
\]

\[
\text{This depends on the code whether your code really gives you type equality.}
\]

\[
\text{EQ}_A = \exists x, y. f
\]

\[
\text{EQ}_B = \exists x, y. g
\]

\[
\text{EQ}_{A \times B} = \text{EQ}_A \times \text{EQ}_B = \exists x, y. \text{EQ}_A (f x, f y) \times \text{EQ}_B (g x, g y)
\]

\[
\text{functions are eq if they yield eq values for eq arg } 0
\]

\[
\text{For } x, y \text{ from } A \text{ that are equal}
\]

\[
\text{EQ}_{\text{Nat}:A \rightarrow B} = \text{EQ}_A \rightarrow \text{EQ}_B
\]

\[
\text{Question: } A : U, x : A \text{ if } \text{eq}_A (x)
\]

Is this type inhabitable for $A : U$?

**Consider** $A = \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$
IS: $F : (Nat \rightarrow Nat) \rightarrow Nat \vdash EQ_F(F, F)$

Expand def of $EQ_F(A, B)$:

\[ \forall f, g : Nat \rightarrow Nat \quad EQ_{Nat}(f, g) \rightarrow EQ_{Nat}(f \circ f, g \circ g) \]

\[ \forall f, g, x, y : Nat \quad EQ_{Nat}(f, g) \rightarrow EQ_{Nat}(f(x, y), g(x, y)) \]

This term is not reducible, since the functions are variables, you don't know anything about them.

Church's Law:

(Scientific law?)

Every programming language you can write $\lambda$-is eventually like the $\lambda$-calculus or Turing machines (at least for natural numbers).

Q: Example for "unreasonable" element

\[ A : \]

\[ \exists x : \text{Gold number?} \]

"Turing machine code"
**Alternative points of view:**

1. Functions are extensional
   \[ F : (N \to N) \to N. \]
   \[ F : N \to N, \text{ EQ}_{N \to N}(f, [F(f)]) \]
   (Church's Law does not hold!)
   
   consistent with classical math

2. Functions are not extensional
   \[ F : N \to N, \text{ EQ}_{N \to N}(f, [F(f)]) \]
   (Church's Law holds!)
   
   inconsistent with classical math

Problem with Setoids: You have to provide equality for \( M \).

You could do that wrong.

---

**What is equality?**

"That which everything respects"

Leibniz Principle: two things are equal if you cannot tell them apart no matter what you do to them.

**IDENTIFICATION TYPE**

Equality is no longer a property of two elements, but two elements are connected by an identity relation.

\[ \Gamma \vdash A : \text{Type} \]
\[ \Gamma \vdash M : A \quad \Gamma \vdash N : A \]
\[ \Gamma \vdash \text{Id}_A(M, N) : \text{Type} \]

**Type theory does not know that EQ is equality, i.e. EQ exists outside of type theory, you have to simulate it.**

\[ \Gamma \vdash \text{Id}_A(M, N) : \text{Type} \]

**Formation rule?**

(Induction rule for the type \( \text{Id} \))

**Notation:** \[ M \equiv_A N \text{ type?} \]

**Terms**

\[ \text{refl}_A(x) \]

**Witness of identification from \( a \) to \( b \):**

\[ a : \text{Id}_A(a, b) \]
If this rule was the only introduction rule, $\text{refl}_A$ is the least reflexive relation.

Elimination

$$
\Gamma, x : A \vdash \text{refl}_A(x) \quad \text{proof of equality}
$$

If $C$ is a reflexive relation and $\alpha$ the least reflexive relation (which it is if I from above is the only intro rule), then I can just map out of $C$:

**Rule says:** "If we can construct a proof $\alpha$ using $\text{refl}_A$, we can use $\Gamma$ for the identity proof."

$$
\Gamma, x : A, y : A, z : \text{Id}_A(x, y) \vdash \lambda [x, y, z]. C(x, y, z). \quad \text{(conclusion of (E))}
$$

**Special case "transport"**

$$
x : A \vdash C_x : M
$$

$$
\Gamma, \alpha : \text{Id}_A(M, N) \vdash \lambda x. [x. C(x) : [M/x] C \Rightarrow [N/x] C].
$$

"logically equal" but logical equivalence is relatively weak.