

# Paramcoq-iff: Parametricity and Uniformity of Propositions

Abhishek Anand and Greg Morrisett



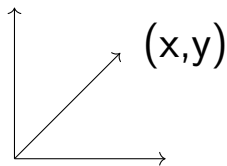
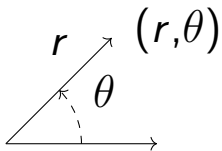
Cornell University

(paper and slides are on the wiki)

# Parametricity: Reynold's vision

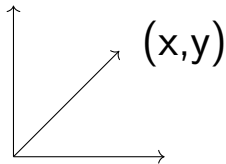
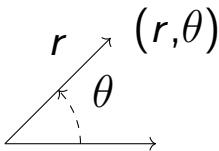
Reynolds, "Types, Abstraction and Parametric Polymorphism", 1983

## Complex Variables



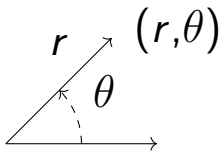


## Complex Variables





## Complex Variables

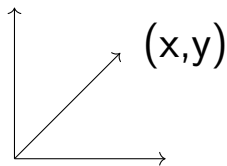


+

×

|| ||

⋮



# Parametricity: Reynold's vision



## Complex Variables



$\mathbb{C} := \{$

$\mathbb{C} : \text{Type}$

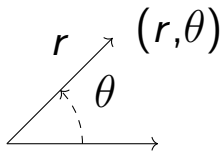
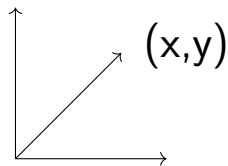
$+: \mathbb{C} \rightarrow \mathbb{C} \rightarrow \mathbb{C}$

$\times: \mathbb{C} \rightarrow \mathbb{C} \rightarrow \mathbb{C}$

$|| \ ||: \mathbb{C} \rightarrow \mathbb{R}$

$\vdots$

$\}$



# Parametricity: Reynold's vision



## Complex Variables



$\mathcal{C}I := \{$

$\mathbb{C}$ : **Type**

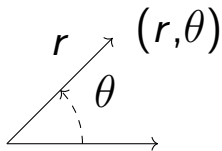
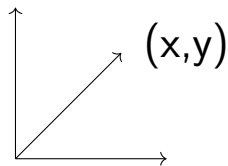
$+: \mathbb{C} \rightarrow \mathbb{C} \rightarrow \mathbb{C}$

$\times: \mathbb{C} \rightarrow \mathbb{C} \rightarrow \mathbb{C}$

$|| \ ||: \mathbb{C} \rightarrow \mathbb{R}$

$\vdots$

$\}$



$2^{nd}$  lecture onwards :  $\mathcal{C}I \rightarrow \dots$

# Parametricity: Reynold's vision



## Complex Variables



$\mathbb{C}I := \{$

$\mathbb{C}$ : **Type**

$+: \mathbb{C} \rightarrow \mathbb{C} \rightarrow \mathbb{C}$

$\times: \mathbb{C} \rightarrow \mathbb{C} \rightarrow \mathbb{C}$

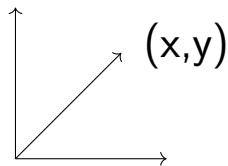
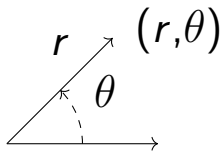
$|| \ ||: \mathbb{C} \rightarrow \mathbb{R}$

$\vdots$

$\}$

behaves uniformly

$2^{nd}$  lecture onwards :  $\mathbb{C}I \rightarrow \text{...}$





## System F

System F : ~~proofs~~

System F : proofs

⋮

System F : proofs

⋮

Coq/Agda

Bernardy 2011, Keller and Lasson 2012

System F : proofs

⋮

Coq/Agda : complex analysis course

Bernardy 2011, Keller and Lasson 2012

System F : proofs

⋮

Coq/Agda : complex analysis course

Bernardy 2011, Keller and Lasson 2012

program translation

System F : proofs

⋮

Coq/Agda : complex analysis course

Bernardy 2011, Keller and Lasson 2012

program translation

computations ✓

System F : ~~proofs~~

⋮

Coq/Agda : complex analysis course

Bernardy 2011, Keller and Lasson 2012

program translation

computations ✓

logic?



# Missing: Uniformity of Coq's logic

$f: \forall T:\text{Type}, T \rightarrow \text{bool}$

# Missing: Uniformity of Coq's logic

$f: \forall T:\text{Type}, T \rightarrow \text{bool}$

$\llbracket f \rrbracket$

# Missing: Uniformity of Coq's logic

$f: \forall T:\text{Type}, T \rightarrow \text{bool}$

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 = f T_2 ta_2$

# Missing: Uniformity of Coq's logic

constant



$f: \forall T:\text{Type}, T \rightarrow \text{bool}$

$$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$$
$$(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$$
$$f T_1 ta_1 = f T_2 ta_2$$

# Missing: Uniformity of Coq's logic

constant

$f: \forall T:\text{Type}, T \rightarrow \text{bool}$

$\lambda\_, \text{True}$

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (\downarrow T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 = f T_2 ta_2$

# Missing: Uniformity of Coq's logic

$f: \forall T:\text{Type}, T \rightarrow \text{boolProp}$

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 = f T_2 ta_2$

# Missing: Uniformity of Coq's logic

undecidable predicates

$f: \forall T:\text{Type}, T \rightarrow \text{boolProp}$

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 = f T_2 ta_2$

# Missing: Uniformity of Coq's logic

undecidable predicates

$f: \forall T:\text{Type}, T \rightarrow \text{boolProp}$

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 = f T_2 ta_2$



# Missing: Uniformity of Coq's logic

undecidable predicates

$f: \forall T:\text{Type}, T \rightarrow \text{boolProp}$

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 \rightarrow f T_2 ta_2 \rightarrow \text{Prop}$

# Missing: Uniformity of Coq's logic

undecidable predicates

$f: \forall T:\text{Type}, T \rightarrow \text{boolProp}$

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 \rightarrow f T_2 ta_2 \rightarrow \text{Prop}$

# Missing: Uniformity of Coq's logic

undecidable predicates

$f: \forall T:\text{Type}, T \rightarrow \text{boolProp}$

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 \rightarrow f T_2 ta_2 \rightarrow \text{Prop}$

$\lambda_{--}, \text{True}$

# Missing: Uniformity of Coq's logic

undecidable predicates

$f: \forall T:\text{Type}, T \rightarrow \text{boolProp}$

utterly useless

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 \rightarrow f T_2 ta_2 \rightarrow \text{Prop}$

$\lambda_{--}, \text{True}$

# Missing: Uniformity of Coq's logic

$f: \forall T: \text{Type}, T \rightarrow \text{boolProp}$

$\llbracket f \rrbracket: \forall (T_1 T_2: \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1: T_1) (ta_2: T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 \rightarrow f T_2 ta_2 \rightarrow \text{Prop}$

$\llbracket \text{bool} \rrbracket := \lambda (b_1 b_2: \text{bool}), b_1 = b_2$

$\llbracket \text{Prop} \rrbracket := \lambda (b_1 b_2: \text{Prop}), b_1 \rightarrow b_2 \rightarrow \text{Prop}$

# Missing: Uniformity of Coq's logic

$f: \forall T:\text{Type}, T \rightarrow \text{Prop}$

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 \rightarrow f T_2 ta_2 \rightarrow \text{Prop}$

$\llbracket \text{Prop} \rrbracket := \lambda (b_1 b_2: \text{Prop}), b_1 \rightarrow b_2 \rightarrow \text{Prop}$

# Missing: Uniformity of Coq's logic

$f: \forall T:\text{Type}, T \rightarrow \text{Prop}$

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 \leftrightarrow f T_2 ta_2$

$\llbracket \text{Prop} \rrbracket := \lambda (b_1 b_2: \text{Prop}), b_1 \rightarrow b_2 \rightarrow \text{Prop}$

# Missing: Uniformity of Coq's logic

$f: \forall T:\text{Type}, T \rightarrow \text{Prop}$

$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 \leftrightarrow f T_2 ta_2$

$\llbracket \text{Prop} \rrbracket := \lambda (b_1 b_2: \text{Prop}), b_1 \rightarrow b_2 \rightarrow \text{Prop}$   
 $\times b_1 \leftrightarrow b_2$



# Impossibility

$$f: \forall T:\text{Type}, T \rightarrow \text{Prop}$$
$$\llbracket f \rrbracket: \forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type}) \\ (ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow \\ f T_1 ta_1 \leftrightarrow f T_2 ta_2$$
$$\llbracket \text{Prop} \rrbracket := \lambda (b_1 b_2: \text{Prop}), b_1 \rightarrow b_2 \rightarrow \text{Prop} \\ \times b_1 \leftrightarrow b_2$$

# Impossibility

?

$f: \forall T:\text{Type}, T \rightarrow \text{Prop}$

$\forall (T_1 T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r ta_1 ta_2 \rightarrow$   
 $f T_1 ta_1 \leftrightarrow f T_2 ta_2$

# Impossibility

$\lambda (T:\text{Type}) (\_ : T), \forall (a\ b:T), a=b$



$f: \forall T:\text{Type}, T \rightarrow \text{Prop}$

$\forall (T_1\ T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r\ ta_1\ ta_2 \rightarrow$   
 $f\ T_1\ ta_1 \leftrightarrow f\ T_2\ ta_2$

# Impossibility

$\lambda (T:\text{Type}) (\_ : T), \forall (a\ b:T), a=b$

$\downarrow$   
 $f: \forall T:\text{Type}, T \rightarrow \text{Prop}$

$\forall (T_1\ T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r\ ta_1\ ta_2 \rightarrow$   
 $f\ T_1\ ta_1 \leftrightarrow f\ T_2\ ta_2$

$\uparrow$   
unit

$\uparrow$   
bool

# Parametricity is too liberal

$\lambda (T:\text{Type}) (\_ : T), \forall (a\ b:T), a=b$

$f: \forall T:\text{Type}, T \rightarrow \text{Prop}$

$\lambda \_, \text{True}$

$\forall (T_1\ T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r\ ta_1\ ta_2 \rightarrow$   
 $f\ T_1\ ta_1 \leftrightarrow f\ T_2\ ta_2$

unit

bool

# Parametricity is too liberal

$\lambda (T:\text{Type}) (\_ : T), \forall (a\ b:T), a=b$

$f: \forall T:\text{Type}, T \rightarrow \text{Prop}$

Minimally restrict parametricity relations

$\lambda \_ \_, \text{True}$

$\forall (T_1\ T_2 : \text{Type}) (T_r: T_1 \rightarrow T_2 \rightarrow \text{Type})$   
 $(ta_1 : T_1) (ta_2 : T_2), T_r\ ta_1\ ta_2 \rightarrow$   
 $f\ T_1\ ta_1 \leftrightarrow f\ T_2\ ta_2$

unit

bool

# Undecidable proposition: example from CertiCoq

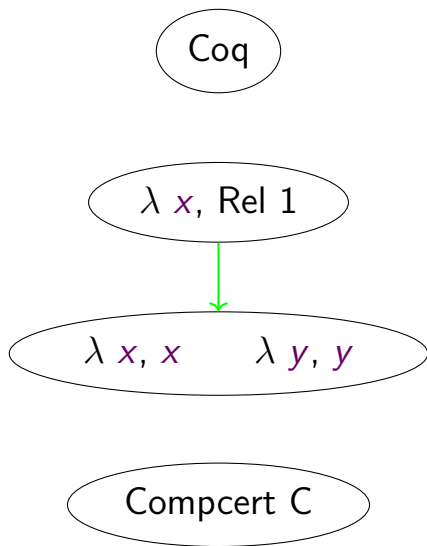


Coq

The diagram consists of two ovals. The top oval is smaller and contains the text 'Coq'. The bottom oval is larger and contains the text 'Compcert C'. They are vertically aligned in the center of the slide.

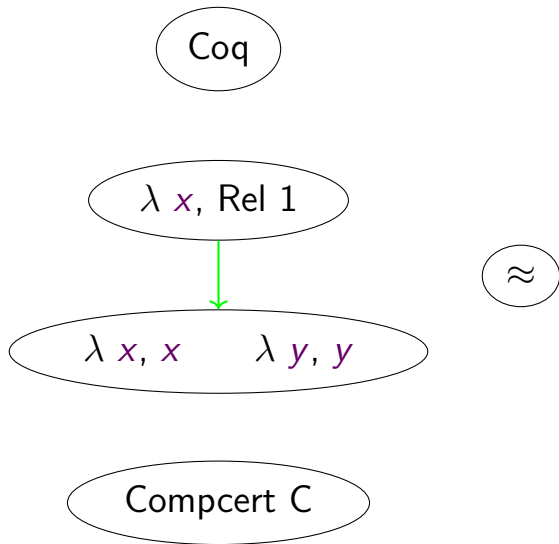
Compcert C

# Undecidable proposition: example from CertiCoq

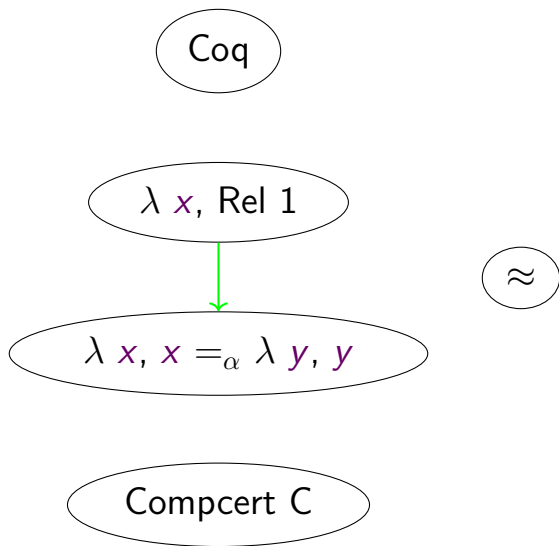




# Undecidable proposition: example from CertiCoq



# Undecidable proposition: example from CertiCoq



$$\llbracket s \rrbracket := \lambda(x\ x_2 : \text{Prop}), x \rightarrow x_2 \rightarrow s$$

$$\llbracket s \rrbracket := \lambda(x\ x_2 : \mathbf{Prop}), x \rightarrow x_2 \rightarrow s$$

$$\begin{aligned} \llbracket \forall x:A. B \rrbracket &:= \lambda(f : \forall x : A. B)(f_2 : \forall x_2 : A_2. B_2), \\ &\quad \forall(x : A)(x_2 : A_2)(x_r : \llbracket A \rrbracket x\ x_2), \\ &\quad \llbracket B \rrbracket (f\ x)(f_2\ x_2) \end{aligned}$$

$$\llbracket s \rrbracket := \lambda(x\ x_2 : \mathbf{Prop}), x \rightarrow x_2 \rightarrow s$$

$$\begin{aligned} \llbracket \forall x : A. B \rrbracket &:= \lambda(f : \forall x : A. B)(f_2 : \forall x_2 : A_2. B_2), \\ &\quad \forall(x : A)(x_2 : A_2)(x_r : \llbracket A \rrbracket x\ x_2), \\ &\quad \llbracket B \rrbracket (f\ x)(f_2\ x_2) \end{aligned}$$

$$\llbracket s \rrbracket := \lambda(x\ x_2 : \mathbf{Prop}), x \rightarrow x_2 \rightarrow s$$

$$\begin{aligned} \llbracket \forall x:A. B \rrbracket &:= \lambda(f : \forall x : A. B)(f_2 : \forall x_2 : A_2. B_2), \\ &\quad \forall(x : A)(x_2 : A_2)(x_r : \llbracket A \rrbracket x\ x_2), \\ &\quad \llbracket B \rrbracket (f\ x)(f_2\ x_2) \end{aligned}$$

$$\llbracket s \rrbracket := \lambda(x\ x_2 : \mathbf{Prop}), x \rightarrow x_2 \rightarrow s$$

$$\begin{aligned} \llbracket \forall x : A. B \rrbracket &:= \lambda(f : \forall x : A. B)(f_2 : \forall x_2 : A_2. B_2), \\ &\quad \forall(x : A)(x_2 : A_2)(x_r : \llbracket A \rrbracket x\ x_2), \\ &\quad \llbracket B \rrbracket (f\ x)(f_2\ x_2) \end{aligned}$$

# AnyRel translation (Keller and Lassel 2012)

$$\Gamma \vdash a : T \quad \Rightarrow \quad \llbracket \Gamma \rrbracket \vdash \llbracket a \rrbracket : (\llbracket T \rrbracket \ a \ a_2)$$

$$\llbracket s \rrbracket := \lambda(x \ x_2 : \mathbf{Prop}), x \rightarrow x_2 \rightarrow s$$

$$\begin{aligned} \llbracket \forall x : A. B \rrbracket &:= \lambda(f : \forall x : A. B)(f_2 : \forall x_2 : A_2. B_2), \\ &\quad \forall(x : A)(x_2 : A_2)(x_r : \llbracket A \rrbracket x \ x_2), \\ &\quad \llbracket B \rrbracket (f \ x)(f_2 \ x_2) \end{aligned}$$



# AnyRel translation (Keller and Lassen 2012)

$$\Gamma \vdash a : T \quad \Rightarrow \quad \llbracket \Gamma \rrbracket \vdash \llbracket a \rrbracket : (\llbracket T \rrbracket \ a \ a_2)$$

$$\llbracket s \rrbracket := \lambda(x \ x_2 : \mathbf{Prop}), x \rightarrow x_2 \rightarrow s$$

$$\begin{aligned} \llbracket \forall x : A. B \rrbracket &:= \lambda(f : \forall x : A. B)(f_2 : \forall x_2 : A_2. B_2), \\ &\quad \forall(x : A)(x_2 : A_2)(x_r : \llbracket A \rrbracket x \ x_2), \\ &\quad \llbracket B \rrbracket (f \ x)(f_2 \ x_2) \end{aligned}$$

# AnyRel translation (Keller and Lasson 2012)

$$\Gamma \vdash a : T \quad \Rightarrow \quad \llbracket \Gamma \rrbracket \vdash \llbracket a \rrbracket : (\llbracket T \rrbracket \ a \ a_2)$$

$$\llbracket s \rrbracket := \lambda(x \ x_2 : \mathbf{Prop}), x \rightarrow x_2 \rightarrow s$$

$$\begin{aligned} \llbracket \forall x : A. B \rrbracket &:= \lambda(f : \forall x : A. B)(f_2 : \forall x_2 : A_2. B_2), \\ &\quad \forall(x : A)(x_2 : A_2)(x_r : \llbracket A \rrbracket x \ x_2), \\ &\quad \llbracket B \rrbracket (f \ x)(f_2 \ x_2) \end{aligned}$$

$$\llbracket x \rrbracket := x_r$$

$$\llbracket \lambda x : A. B \rrbracket := \lambda(x : A)(x_2 : A_2)(x_r : \llbracket A \rrbracket x \ x_2), \llbracket B \rrbracket$$

$$\llbracket (A \ B) \rrbracket := (\llbracket A \rrbracket \ B \ B_2 \ \llbracket B \rrbracket)$$

# AnyRel translation (Keller and Lasson 2012)

$$\Gamma \vdash a : T \quad \Rightarrow \quad \llbracket \Gamma \rrbracket \vdash \llbracket a \rrbracket : (\llbracket T \rrbracket \ a \ a_2)$$

$$\llbracket s \rrbracket := \lambda(x \ x_2 : \mathbf{Prop}), x \rightarrow x_2 \rightarrow s$$

$$\begin{aligned} \llbracket \forall x : A. B \rrbracket &:= \lambda(f : \forall x : A. B)(f_2 : \forall x_2 : A_2. B_2), \\ &\quad \forall(x : A)(x_2 : A_2)(x_r : \llbracket A \rrbracket x \ x_2), \\ &\quad \llbracket B \rrbracket (f \ x)(f_2 \ x_2) \end{aligned}$$

$$\llbracket x \rrbracket := x_r$$

$$\llbracket \lambda x : A. B \rrbracket := \lambda(x : A)(x_2 : A_2)(x_r : \llbracket A \rrbracket x \ x_2), \llbracket B \rrbracket$$

$$\llbracket (A \ B) \rrbracket := (\llbracket A \rrbracket \ B \ B_2 \ \llbracket B \rrbracket)$$

# Our translation

$$\llbracket s \rrbracket := \lambda(x\ x_2 : \text{Prop}), x \rightarrow x_2 \rightarrow s$$

# Our translation

$$\llbracket \text{Prop} \rrbracket_{iso} := \lambda(x\ x_2 : \text{Prop}), x \rightarrow x_2 \rightarrow s \quad \times \quad x \leftrightarrow x_2$$

# Our translation

$$\begin{aligned} \llbracket \text{Prop} \rrbracket_{iso} &:= \lambda(x\ x_2 : \text{Prop}), x \rightarrow x_2 \rightarrow s \quad \times \quad x \leftrightarrow x_2 \\ \llbracket \text{Type}_i \rrbracket_{iso} &:= \lambda(x\ x_2 : \text{Type}_i), \{R : x \rightarrow x_2 \rightarrow s \& P\ R\} \end{aligned}$$

# Our translation

$$\begin{aligned} \llbracket \text{Prop} \rrbracket_{iso} &:= \lambda(x\ x_2 : \text{Prop}), x \rightarrow x_2 \rightarrow s \quad \times \quad x \leftrightarrow x_2 \\ \llbracket \text{Type}_i \rrbracket_{iso} &:= \lambda(x\ x_2 : \text{Type}_i), \{R : x \rightarrow x_2 \rightarrow s \& P\ R\} \end{aligned}$$



weakest condition

# Our translation

$$\begin{aligned} \llbracket \text{Prop} \rrbracket_{iso} &:= \lambda(x \ x_2 : \text{Prop}), x \rightarrow x_2 \rightarrow s \quad \times \quad x \leftrightarrow x_2 \\ \llbracket \text{Type}_i \rrbracket_{iso} &:= \lambda(x \ x_2 : \text{Type}_i), \{R : x \rightarrow x_2 \rightarrow s \& P \ R\} \end{aligned}$$



weakest condition

up next: weakest conditions for relations of types  
occurring in propositions



# Propositions in Coq

- $\forall$
- Inductives



$A : \text{Type}$

$B : A \rightarrow \text{Prop}$

$(\forall (x : A). B\ x) \text{Prop}$

$A:\text{Type}$   
 $\updownarrow$   
 $\llbracket A \rrbracket$   
 $A_2:\text{Type}$

$B:A \rightarrow \text{Prop}$

$(\forall (x:A). B\ x) \text{Prop}$


$$\begin{array}{c} A:\text{Type} \\ \updownarrow \llbracket A \rrbracket \\ A_2:\text{Type} \end{array}$$
$$\begin{array}{c} B:A \rightarrow \text{Prop} \\ \updownarrow \llbracket B \rrbracket \\ B_2:A_2 \rightarrow \text{Prop} \end{array}$$
$$(\forall (x:A). B \ x) \text{Prop}$$

$$\begin{array}{c} A:\text{Type} \\ \updownarrow \llbracket A \rrbracket \\ A_2:\text{Type} \end{array}$$

$$\begin{array}{c} B:A \rightarrow \text{Prop} \\ \updownarrow \llbracket B \rrbracket \\ B_2:A_2 \rightarrow \text{Prop} \end{array}$$

$$\llbracket B \rrbracket \Rightarrow \forall (a:A) (a_2:A_2), \llbracket A \rrbracket a a_2 \rightarrow (B a \leftrightarrow B_2 a_2)$$

$$(\forall (x:A). B x)$$

$$\begin{array}{c} A:\text{Type} \\ \updownarrow \\ \llbracket A \rrbracket \end{array}$$

$$A_2:\text{Type}$$

$$\begin{array}{c} B:A \rightarrow \text{Prop} \\ \updownarrow \\ \llbracket B \rrbracket \end{array}$$

$$B_2:A_2 \rightarrow \text{Prop}$$

$$\llbracket B \rrbracket \Rightarrow \forall (a:A) (a_2:A_2), \llbracket A \rrbracket a a_2 \rightarrow (B a \leftrightarrow B_2 a_2)$$

$$(\forall (x:A). B x) \quad \Leftrightarrow \quad (\forall (x_2:A_2). B_2 x_2)$$

$A := \text{True}, A_2 := \text{False}$

$A : \text{Type}$

$\Uparrow \llbracket A \rrbracket$

$A_2 : \text{Type}$

$B := B_2 := \lambda \_, \text{False}$

$B : A \rightarrow \text{Prop}$

$\Uparrow \llbracket B \rrbracket$

$B_2 : A_2 \rightarrow \text{Prop}$

$\llbracket B \rrbracket \Rightarrow \forall (a : A) (a_2 : A_2), \llbracket A \rrbracket a a_2 \rightarrow (B a \leftrightarrow B_2 a_2)$

$(\forall (x : A). B x) \quad \Leftrightarrow \quad (\forall (x_2 : A_2). B_2 x_2)$

$$\begin{array}{c} A:\text{Type} \\ \updownarrow \llbracket A \rrbracket \\ A_2:\text{Type} \end{array}$$

$$\begin{array}{c} B:A \rightarrow \text{Prop} \\ \updownarrow \llbracket B \rrbracket \\ B_2:A_2 \rightarrow \text{Prop} \end{array}$$

$$\llbracket B \rrbracket \Rightarrow \forall (a:A) (a_2:A_2), \llbracket A \rrbracket a a_2 \rightarrow (B a \leftrightarrow B_2 a_2)$$

$$\begin{aligned} \text{Total } \llbracket A \rrbracket &:= \forall (a:A), \{a_2:A_2 \ \& \ \llbracket A \rrbracket a a_2\} \\ &\quad \wedge \forall (a_2:A_2), \{a:A \ \& \ \llbracket A \rrbracket a a_2\} \end{aligned}$$

$$(\forall (x:A). B x) \quad \Leftrightarrow \quad (\forall (x_2:A_2). B_2 x_2)$$



# Propositions in Coq

- $\forall$
- Inductives
  - equality (`Coq.Init.Logic.eq`)
  - $\vdots$

# Propositions in Coq

- $\forall$  ✓
- Inductives
  - equality (`Coq.Init.Logic.eq`)
  - $\vdots$

A:Type

x:A

y:A

(eq A x y) Prop

$A:\text{Type}$                        $x:A$                        $y:A$

$\updownarrow$   
 $\llbracket A \rrbracket$

$A_2:\text{Type}$

$(\text{eq } A \ x \ y) \text{ Prop}$

$$\begin{array}{c} A:\text{Type} \\ \updownarrow \\ \llbracket A \rrbracket \\ \updownarrow \\ A_2:\text{Type} \end{array}$$

$$\begin{array}{c} x:A \\ \updownarrow \\ \llbracket x \rrbracket : \llbracket A \rrbracket \quad x \quad x_2 \\ \updownarrow \\ x_2:A_2 \end{array}$$

$$\begin{array}{c} y:A \\ \updownarrow \\ \llbracket y \rrbracket : \llbracket A \rrbracket \quad y \quad y_2 \\ \updownarrow \\ y_2:A_2 \end{array}$$

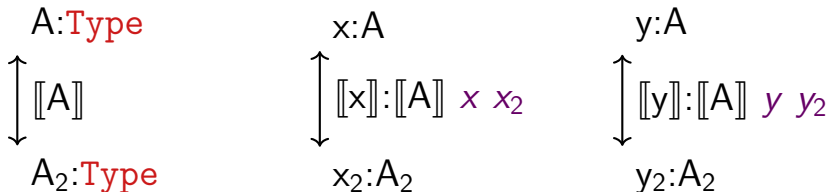
$(\text{eq } A \ x \ y) \text{ Prop}$

$$\begin{array}{c} A:\text{Type} \\ \updownarrow \\ \llbracket A \rrbracket \\ \updownarrow \\ A_2:\text{Type} \end{array}$$

$$\begin{array}{c} x:A \\ \updownarrow \\ \llbracket x \rrbracket : \llbracket A \rrbracket \quad x \quad x_2 \\ \updownarrow \\ x_2:A_2 \end{array}$$

$$\begin{array}{c} y:A \\ \updownarrow \\ \llbracket y \rrbracket : \llbracket A \rrbracket \quad y \quad y_2 \\ \updownarrow \\ y_2:A_2 \end{array}$$

$$(\text{eq } A \quad x \quad y) \quad \Leftrightarrow \quad (\text{eq } A_2 \quad x_2 \quad y_2)$$



$\text{OneToOne } \llbracket A \rrbracket := \forall (x \ y : A) (x_2 \ y_2 : A_2),$   
 $\llbracket A \rrbracket \ x \ x_2 \rightarrow \llbracket A \rrbracket \ y \ y_2 \rightarrow (\text{eq } A \ x \ y \leftrightarrow \text{eq } A_2 \ x_2 \ y_2)$

$(\text{eq } A \ x \ y) \quad \leftrightarrow \quad (\text{eq } A_2 \ x_2 \ y_2)$

# General Inductives



# W props

```
Inductive IWP (I A : Type) (B : A → Type)  
  (AI : A → I) (BI : ∀ (a : A), B a → I)  
  : ∀ (i:I), Prop :=  
node : ∀ (a : A)  
  (brs : ∀ b : B a, IWP I A B AI BI (BI a b)),  
  IWP I A B AI BI (AI a).
```

Total

Total

Inductive IWP ( $I : A : \text{Type}$ ) ( $B : A \rightarrow \text{Type}$ )  
 ( $AI : A \rightarrow I$ ) ( $BI : \forall (a : A), B a \rightarrow I$ )  
 :  $\forall (i : I), \text{Prop} :=$   
 node :  $\forall (a : A)$   
 ( $brs : \forall b : B a, \text{IWP } I A B AI BI (BI a b)$ ),  
 IWP  $I A B AI BI (AI a)$ .

Total

Total

Inductive IWP ( $I A : \text{Type}$ ) ( $B : A \rightarrow \text{Type}$ )  
 ( $AI : A \rightarrow I$ ) ( $BI : \forall (a : A), B a \rightarrow I$ )  
 :  $\forall (i:I), \text{Prop} :=$   
 node :  $\forall (a : A)$   
 ( $brs : \forall b : B a, \text{IWP } I A B AI BI (BI a b)$ ),  
 IWP  $I A B AI BI (AI a)$ .

IWP  $A B AI BI i \leftrightarrow \text{IWP } A_2 B_2 AI_2 BI_2 i_2$

Total

Total

Inductive IWP ( $I A : \text{Type}$ ) ( $B : A \rightarrow \text{Type}$ )  
 ( $AI : A \rightarrow I$ ) ( $BI : \forall (a : A), B a \rightarrow I$ )  
 :  $\forall (i:I), \text{Prop} :=$   
 node :  $\forall (a : A)$   
 ( $brs : \forall b : B a, \text{IWP } I A B AI BI (BI a b)$ ),  
 IWP  $I A B AI BI (AI a)$ .

IWP  $A B AI BI i \leftrightarrow \text{IWP } A_2 B_2 AI_2 BI_2 i_2$

# Recap: Propositions in Coq

- $\forall$
- Inductives

# Recap: Propositions in Coq

- $\forall$  ✓
- Inductives ✓

# Recap: Propositions in Coq

- $\forall$  ✓
- Inductives ✓

Summary:

# Recap: Propositions in Coq

- $\forall$  ✓
- Inductives ✓

Summary:

- Total and OneToOne for types mentioned in  
Props  $\Rightarrow$  uniformity of props



# Recap: Propositions in Coq

- $\forall$  ✓
- Inductives ✓

Summary:

- Total and OneToOne for types mentioned in Props  $\Rightarrow$  uniformity of props
- One or both may not be needed, depending on where the type occurs

# Total and OneToOne for types

$\llbracket \lambda (A:\text{Type}), (\forall x:A \quad \dots):\text{Prop} \rrbracket$

# Total and OneToOne for types

$\llbracket \lambda (A:\text{Type}), (\forall x:A \dots):\text{Prop} \rrbracket :=$

$\lambda (A:\text{Type}) (A_2:\text{Type})$   
 $(A_r:\{A_r: A \rightarrow A_2 \rightarrow \text{Type} \mid \text{Total } A_r\}), \dots$

# Total and OneToOne for types

$\llbracket \lambda (A:\text{Type}), (\forall x:A \quad \dots):\text{Prop} \rrbracket$

# Total and OneToOne for types

$\llbracket \lambda (A:\text{Type}), (\forall x:A \times A, \dots):\text{Prop} \rrbracket$

## Total and OneToOne for types

$\llbracket \lambda (A:\text{Type}), (\forall x:A \times A, \dots):\text{Prop} \rrbracket$

Total and OneToOne for composite types

# Canonical types in Coq

- $\forall$
- inductives
- universes

$$\begin{aligned} \llbracket \forall x: A. B \rrbracket &:= \\ &\lambda(f: \forall x: A. B)(f_2: \forall x_2: A_2. B_2), \\ &\quad \forall(x: A)(x_2: A_2) (x_r: \llbracket A \rrbracket x x_2), \quad \llbracket B \rrbracket(f x)(f_2 x_2) \end{aligned}$$



$\llbracket \forall x: A. B \rrbracket :=$

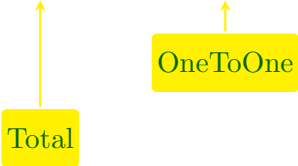
$\lambda(f: \forall x: A. B)(f_2: \forall x_2: A_2. B_2),$

$\forall(x: A)(x_2: A_2) (x_r: \llbracket A \rrbracket x x_2), \llbracket B \rrbracket (f x)(f_2 x_2)$

$$\llbracket \forall x: A. B \rrbracket :=$$
$$\lambda(f: \forall x: A. B)(f_2: \forall x_2: A_2. B_2),$$
$$\forall(x: A)(x_2: A_2) (x_r: \llbracket A \rrbracket x x_2), \llbracket B \rrbracket (f x)(f_2 x_2)$$

Total

Total, OneToOne, proof irrelevant

$$\llbracket \forall x. A.B \rrbracket :=$$
$$\lambda(f:\forall x:A.B)(f_2:\forall x_2:A_2.B_2),$$
$$\forall(x:A)(x_2:A_2) (x_r:\llbracket A \rrbracket x x_2), \llbracket B \rrbracket(f x)(f_2 x_2)$$


The diagram illustrates the semantic definition of the universal quantifier  $\forall$ . It shows the expression  $\llbracket \forall x. A.B \rrbracket$  defined as a lambda function. The lambda function takes two arguments:  $f$  (representing a function from  $A$  to  $B$ ) and  $f_2$  (representing a function from  $A_2$  to  $B_2$ ). The body of the lambda function is  $\forall(x:A)(x_2:A_2) (x_r:\llbracket A \rrbracket x x_2), \llbracket B \rrbracket(f x)(f_2 x_2)$ . Below the expression, two yellow boxes labeled 'Total' and 'OneToOne' have arrows pointing to the semantic expression. 'Total' points to the first argument of the lambda function, and 'OneToOne' points to the body of the lambda function.

function extensionality

## W types: Total and OneToOne

```
Inductive IWT (I A : Type) (B : A → Type)
  (AI : A → I) (BI : ∀ (a : A), B a → I)
  : ∀ (i:I), Type :=
node : ∀ (a : A)
  (brs : ∀ b : B a, IWT I A B AI BI (BI a b)),
IWT I A B AI BI (AI a).
```

# W types: Total and OneToOne

OneToOne, irr

Total

Total, OneToOne, irr

Inductive IWT (*I* *A* : Type) (*B* : *A* → Type)  
 (*AI* : *A* → *I*) (*BI* : ∀ (*a* : *A*), *B* *a* → *I*)  
 : ∀ (*i* : *I*), Type :=  
node : ∀ (*a* : *A*)  
 (*brs* : ∀ *b* : *B* *a*, IWT *I* *A* *B* *AI* *BI* (*BI* *a* *b*)),  
 IWT *I* *A* *B* *AI* *BI* (*AI* *a*).

## W types: Total and OneToOne

OneToOne, irr

Total

Inductive IWT ( $I$   $A$  : Type) ( $B$  :  $A \rightarrow$  Type)  
( $AI$  :  $A \rightarrow I$ ) ( $BI$  :  $\forall (a : A), B\ a \rightarrow I$ )  
:  $\forall (i:I),$  Type :=  
node :  $\forall (a : A)$   
( $brs$  :  $\forall b : B\ a, IWT\ I\ A\ B\ AI\ BI\ (BI\ a\ b)$ ),  
 $IWT\ I\ A\ B\ AI\ BI\ (AI\ a)$ .

function extensionality, proof irrelevance

# OneToOne of $[[\text{Type}_i]]$ is not provable

# OneToOne of $\llbracket \text{Type}_i \rrbracket$ is not provable

$$\llbracket \text{Type}_j \rrbracket_{iso} := \lambda (x \ x_2 : \text{Type}_j),$$
$$\{ R : x \rightarrow x_2 \rightarrow \text{Type}_j \ \& \ \text{Total } R \ \& \ \text{OneToOne } R \}$$



# OneToOne of $\llbracket \text{Type}_i \rrbracket$ is not provable

$$\llbracket \text{Type}_j \rrbracket_{iso} := \lambda (x \ x_2 : \text{Type}_j),$$
$$\{R : x \rightarrow x_2 \rightarrow \text{Type}_j \ \& \ \text{Total } R \ \& \ \text{OneToOne } R\}$$

$\text{Type}_0 : \text{Type}_1$

# OneToOne of $\llbracket \text{Type}_i \rrbracket$ is not provable

$$\llbracket \text{Type}_j \rrbracket_{iso} := \lambda (x \ x_2 : \text{Type}_j), \\ \{ R : x \rightarrow x_2 \rightarrow \text{Type}_j \ \& \ \text{Total } R \ \& \ \text{OneToOne } R \}$$

$\text{Type}_0 : \text{Type}_1$

$$\llbracket \text{Type}_0 \rrbracket_{iso} \text{ nat nat} \\ \llbracket \text{Type}_0 \rrbracket_{iso} \text{ nat (list unit)}$$

# OneToOne of $\llbracket \text{Type}_i \rrbracket$ is not provable

$$\llbracket \text{Type}_j \rrbracket_{iso} := \lambda (x \ x_2 : \text{Type}_j), \\ \{ R : x \rightarrow x_2 \rightarrow \text{Type}_j \ \& \ \text{Total } R \ \& \ \text{OneToOne } R \}$$

$\text{Type}_0 : \text{Type}_1$

$$\llbracket \text{Type}_0 \rrbracket_{iso} \text{ nat } \text{nat}$$
$$\llbracket \text{Type}_0 \rrbracket_{iso} \text{ nat } (\text{list unit})$$

# Fail on Proposition mentioning higher universes

$$\begin{aligned} \llbracket \text{Set} \rrbracket_{iso} &:= \lambda (x \ x_2 : \text{Set}), \\ &\{ R : x \rightarrow x_2 \rightarrow \text{Set} \ \& \ \text{Total } R \ \& \ \text{OneToOne } R \} \\ \llbracket \text{Type}_j \rrbracket_{iso} &:= \lambda (x \ x_2 : \text{Type}_j), x \rightarrow x_2 \rightarrow \text{Type}_j \end{aligned}$$

# Fail on Proposition mentioning higher universes

$$\begin{aligned} \llbracket \text{Set} \rrbracket_{iso} &:= \lambda (x \ x_2 : \text{Set}), \\ &\{ R : x \rightarrow x_2 \rightarrow \text{Set} \ \& \ \text{Total } R \ \& \ \text{OneToOne } R \} \\ \llbracket \text{Type}_j \rrbracket_{iso} &:= \lambda (x \ x_2 : \text{Type}_j), x \rightarrow x_2 \rightarrow \text{Type}_j \end{aligned}$$

# Fail on Proposition mentioning higher universes

$$\begin{aligned} \llbracket \text{Set} \rrbracket_{iso} &:= \lambda (x \ x_2 : \text{Set}), \\ &\{ R : x \rightarrow x_2 \rightarrow \text{Prop} \ \& \ \text{Total } R \ \& \ \text{OneToOne } R \} \\ \llbracket \text{Type}_j \rrbracket_{iso} &:= \lambda (x \ x_2 : \text{Type}_j), x \rightarrow x_2 \rightarrow \text{Type}_j \end{aligned}$$

# Fail on Proposition mentioning higher universes

$$\llbracket \text{Set} \rrbracket_{iso} := \lambda (x \ x_2 : \text{Set}),$$
$$\{ R : x \rightarrow x_2 \rightarrow \text{Prop} \ \& \ \text{Total } R \ \& \ \text{OneToOne } R \}$$
$$\llbracket \text{Type}_j \rrbracket_{iso} := \lambda (x \ x_2 : \text{Type}_j), x \rightarrow x_2 \rightarrow \text{Type}_j$$

Implicitly subtyping:

$$(\lambda (x : \text{Type}_j), b) \text{ nat}$$

# Fail on Proposition mentioning higher universes

$\llbracket \text{Set} \rrbracket_{iso} := \lambda (x \ x_2 : \text{Set}),$   
 $\{ R : x \rightarrow x_2 \rightarrow \text{Prop} \ \& \ \text{Total } R \ \& \ \text{OneToOne } R \}$   
 $\llbracket \text{Type}_j \rrbracket_{iso} := \lambda (x \ x_2 : \text{Type}_j), \ x \rightarrow x_2 \rightarrow \text{Type}_j$

Implicitly subtyping:

$(\lambda (x : \text{Type}_j), b) \text{ nat}$

$(\lambda (x \ x_2 : \text{Type}_j) \ x_r, b) \text{ nat nat } \llbracket \text{nat} \rrbracket_{iso}$



# Fail on Proposition mentioning higher universes

$$\llbracket \text{Set} \rrbracket_{iso} := \lambda (x \ x_2 : \text{Set}),$$
$$\{ R : x \rightarrow x_2 \rightarrow \text{Prop} \ \& \ \text{Total } R \ \& \ \text{OneToOne } R \}$$
$$\llbracket \text{Type}_j \rrbracket_{iso} := \lambda (x \ x_2 : \text{Type}_j), x \rightarrow x_2 \rightarrow \text{Type}_j$$

Implicit subtyping:

$$(\lambda (x : \text{Type}_j), b) \text{ nat}$$
$$(\lambda (x \ x_2 : \text{Type}_j) \ x_r, b) \text{ nat nat } \llbracket \text{nat} \rrbracket_{iso}$$

## Recap: Total and OneToOne for types

- $\forall$  ✓
- inductives ✓
- universes ✓
  - Set ✓
  - Type ✓

# IsoRel translation

$$\begin{aligned} \llbracket \lambda x : A, B \rrbracket_{iso} := \\ \lambda (x : A) (x_2 : A_2) (x_r : \quad \llbracket A \rrbracket x x_2), \llbracket B \rrbracket \end{aligned}$$

$$\begin{aligned} \llbracket \lambda x : A, B \rrbracket_{iso} &:= \\ \lambda (x : A) (x_2 : A_2) (x_r : \pi_1 \llbracket A \rrbracket x x_2), &\llbracket B \rrbracket \end{aligned}$$

# MinIsoRel translation

$$\llbracket \text{Set} \rrbracket_{iso} := \lambda (x \ x_2 : \text{Set}),$$
$$\{ R : x \rightarrow x_2 \rightarrow \text{Prop} \ \& \quad \quad \quad \}$$

# MinIsoRel translation

$$\llbracket \text{Set} \rrbracket_{iso} := \lambda (x \ x_2 : \text{Set}),$$
$$\{ R : x \rightarrow x_2 \rightarrow \text{Prop} \ \& \ \text{OneToOne } R \}$$

Definition POne :=

$$\lambda (T : \text{Set}) (f : \text{nat} \rightarrow T), f \ 0 = f \ (S \ 0).$$

# MinIsoRel translation

$$\llbracket \text{Set} \rrbracket_{iso} := \lambda (x \ x_2 : \text{Set}),$$
$$\{ R : x \rightarrow x_2 \rightarrow \text{Prop} \ \& \ \text{Total } R \ \& \ }$$

Definition PTot :=

$$\lambda (T : \text{Set}) (f : T \rightarrow \text{nat}), \forall (t : T), f \ t = 0.$$

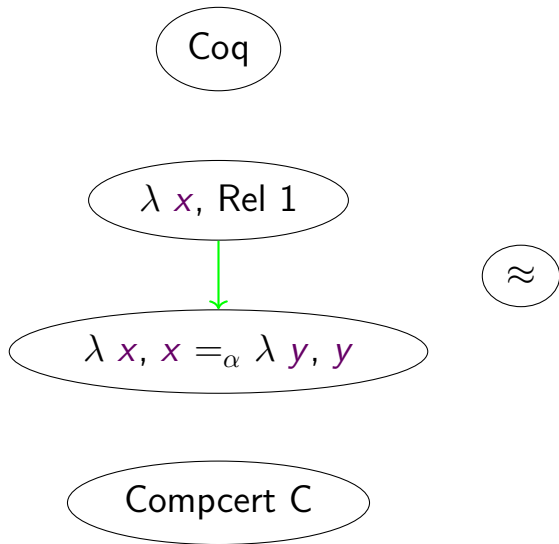


# MinIsoRel translation

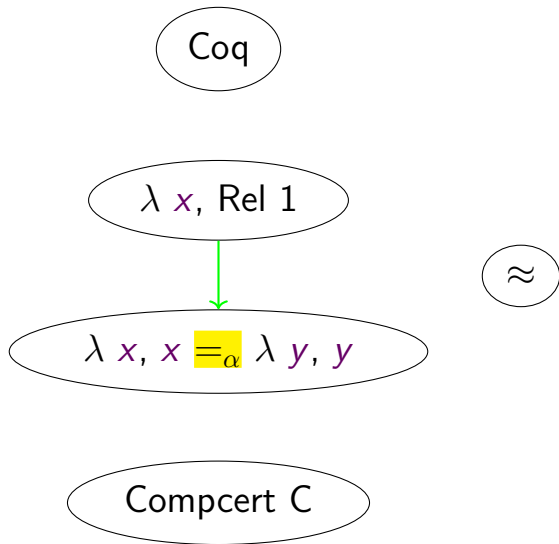
$\llbracket \text{Set} \rrbracket_{iso} := \lambda (x \ x_2 : \text{Set}),$   
 $\{ R : x \rightarrow x_2 \rightarrow \text{Prop} \ \& \quad \quad \quad \}$

Definition PNone :=  
 $\lambda (T : \text{Set}) (f : T \rightarrow \text{nat}) (a \ b : T) , (f \ a = f \ b).$

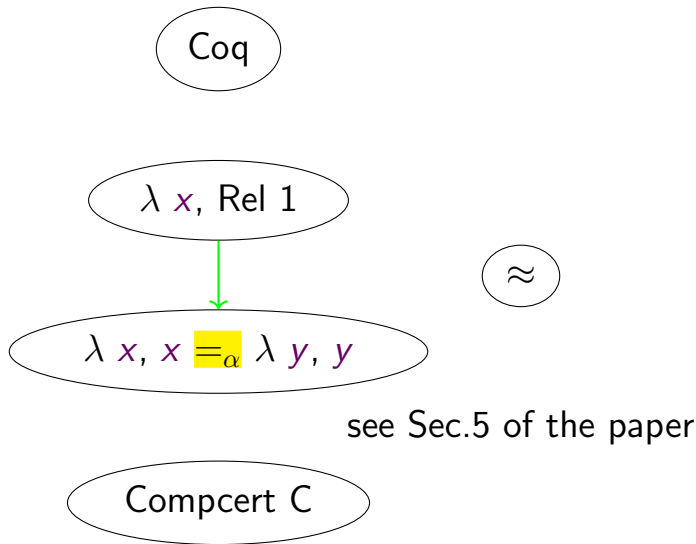
# The example from CertiCoq, again



# The example from CertiCoq, again



# The example from CertiCoq, again



## universe polymorphic inductives

**Inductive**  $\text{list@}\{i\} (A : \text{Type@}\{i\}) : \text{Type@}\{i\}$   
 $:= \text{nil} : \text{list } A \mid \text{cons} : A \rightarrow \text{list } A \rightarrow \text{list } A.$

## universe polymorphic inductives

**Inductive**  $\text{list@}\{i\} (A : \text{Type@}\{i\}) : \text{Type@}\{i\}$   
 $:= \text{nil} : \text{list } A \mid \text{cons} : A \rightarrow \text{list } A \rightarrow \text{list } A.$

**Fixpoint**  $\text{list\_RF@}\{i\} (A \ A_2 : \text{Type@}\{i\})$   
 $(A\_R : A \rightarrow A_2 \rightarrow \text{Type@}\{i\})$   
 $(l : \text{list@}\{i\} \ A) (l_2 : \text{list@}\{i\} \ A_2) : \text{Type@}\{i\} :=$   
**match**  $l, l_2$  **with**  
 $\mid \text{nil}, \text{nil} \Rightarrow \text{True}$   
 $\mid \text{cons } h \ tl, \text{cons } h_2 \ tl_2 \Rightarrow$   
 $(A\_R \ h \ h_2 \times \text{list\_RF } A \ A_2 \ A\_R \ tl \ tl_2) \% \text{type}$   
 $\mid \_, \_ \Rightarrow \text{False}$   
**end.**

## universe polymorphic inductives

**Inductive**  $\text{list@}\{i\} (A : \text{Type@}\{i\}) : \text{Type@}\{i\}$   
 $:= \text{nil} : \text{list } A \mid \text{cons} : A \rightarrow \text{list } A \rightarrow \text{list } A.$

**Fixpoint**  $\text{list\_RF@}\{i\} (A \ A_2 : \text{Type@}\{i\})$   
 $(A\_R : A \rightarrow A_2 \rightarrow \text{Type@}\{i\})$   
 $(l : \text{list@}\{i\} A) (l_2 : \text{list@}\{i\} A_2) : \text{Type@}\{i\} :=$   
**match**  $l, l_2$  **with**  
 $\mid \text{nil}, \text{nil} \Rightarrow \text{True}$   
 $\mid \text{cons } h \ tl, \text{cons } h_2 \ tl_2 \Rightarrow$   
 $(A\_R \ h \ h_2 \times \text{list\_RF } A \ A_2 \ A\_R \ tl \ tl_2) \% \text{type}$   
 $\mid \_, \_ \Rightarrow \text{False}$   
**end.**

## universe polymorphic inductives

```
Inductive list@{i} (A : Type@{i}) : Type@{i}  
:= nil : list A | cons : A → list A → list A.
```

```
Fixpoint list_RF@{i} (A A2: Type@{i})  
  (A_R : A → A2 → if i=Set then Prop else i )  
  (l : list@{i} A) (l2 : list@{i} A2): Type@{i} :=  
match l, l2 with  
| nil,nil ⇒ True  
| cons h tl, cons h2 tl2 ⇒  
  (A_R h h2 × list_RF A A2 A_R tl tl2)%type  
| _,_ ⇒ False  
end.
```



- explicit universe subtyping markers
- more expressive syntax for universe polymorphism
- registering translations
- extracting template-coq/`Ast.v:Term` to `coq/kernel/Term.constr`?
- better support for nested fixes