

logical verification lecture 10
2011 05 10
lambda2 and prop2

logic		lambda	Coq
prop1 intuitionistic classical	~	λ^{\rightarrow}	ind pred
pred1 prop2	~	λP $\lambda 2$	ind data prog ext

identity functions in λ^{\rightarrow}

$\lambda x : \text{nat}. x : \text{nat} \rightarrow \text{nat}$

$\lambda x : \text{bool}. x : \text{bool} \rightarrow \text{bool}$

identity functions in Coq

```
Definition natid : nat -> nat :=  
  fun n : nat => n.
```

```
Definition boolid: bool -> bool :=  
  fun b : bool => b.
```

polymorphic identity in $\lambda 2$

in the polymorphic λ -calculus, or $\lambda 2$, or system F :

$\lambda A : * . \lambda x : A . x : \Pi A : * . A \rightarrow A$

instantiation by application:

$$\frac{\lambda a : * . \lambda x : a . x : \Pi a : * . a \rightarrow a \quad \text{nat} : *}{(\lambda a : * . \lambda x : a . x) \text{ nat} : \text{nat} \rightarrow \text{nat}}$$

polymorphic λ -calculus

we can abstract over variables a in $*$

intuitively: variables over data-types

polymorphic identity in Coq

```
Definition polyid : forall A:Set, A->A :=  
  fun A:Set => fun x:A => x.
```

```
Check polyid nat.  
polyid nat  
  : nat -> nat
```

further encodings

- in Coq: logical connectives are defined using inductive types
we can define them in `prop2`
- in Coq: various data-types are defined using inductive types
we can define them in `lambda2`

natural numbers in $\lambda 2$

- $N = \Pi a: * . a \rightarrow (a \rightarrow a) \rightarrow a$
- $n = \lambda a: * . \lambda o: a . \lambda s: a \rightarrow a . s^n o$

defining successor in two different ways

- $\lambda n: N . \lambda a: * . \lambda o: a . \lambda s: a \rightarrow a . s (n a o s)$
- $\lambda n: N . \lambda a: * . \lambda o: a . \lambda s: a \rightarrow a . (n a (s o) s)$

booleans in $\lambda 2$

- $B = \Pi a: * . a \rightarrow a \rightarrow a$
- $T = \lambda a: * . \lambda x: a . \lambda y: a . x$
- $F = \lambda a: * . \lambda x: a . \lambda y: a . y$

defining some functions on booleans

- $\text{not} = \lambda b: B . \lambda a: * . \lambda u: a . \lambda v: a . (b a v u)$
- $\text{and} = \lambda b: B . \lambda b': B . \lambda a: * . \lambda u: a . \lambda v: a . (b a (b' a u v) v)$
- $\text{or} = \lambda b: B . \lambda b': B . \lambda a: * . \lambda u: a . \lambda v: a . (b a u (b' a u v))$

expressiveness of polymorphic types

Curry-Howard-De Bruijn isomorphism

we can define logical connectives (last week)

we can define various datatypes

prop1 $\sim \lambda \rightarrow$
pred1 $\sim \lambda P$
prop2 $\sim \lambda 2$

prop2: formulas as types

prop2: proofs as terms

a	a
$A \rightarrow B$	$\Pi x:A. B$
$\forall a. B$	$\Pi a:*. B$

proof rules correspond to typing rules

implication introduction and abstraction

$$\frac{B}{A \rightarrow B} \quad \frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash A \rightarrow B : *}{\Gamma \vdash \lambda x : A. b : A \rightarrow B}$$

implication elimination and application

$$\frac{A \rightarrow B \quad A}{B} \quad \frac{\Gamma \vdash P : A \rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash (P a) : B}$$

quantification introduction and abstraction

$$\frac{B}{\forall a. B} \quad \frac{\Gamma, a : * \vdash b : B \quad \Gamma \vdash \Pi a : *. B : *}{\Gamma \vdash \lambda a : *. b : \Pi a : *. B}$$

quantification elimination and application

$$\frac{\forall a. B}{B[a := A]} \quad \frac{\Gamma \vdash P : \Pi a : *. B \quad \Gamma \vdash A : *}{\Gamma \vdash (P A) : B[a := A]}$$