

logical verification lecture 11  
2011 05 13  
prop2 and lambda2

wat hier?

## identity

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

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

```
Definition polyid : forall A : Set, A -> A :=  
  fun (A : Set) => fun (a : A) => a.
```

## lists

```
Inductive natlist : Set :=  
  natnil : natlist  
| natcons : nat -> natlist -> natlist.
```

```
Inductive boollist : Set :=  
  boolnil : boollist  
| boolcons : bool -> boollist -> boollist.
```

```
Inductive polylist (X : Set) : Set :=  
  polynil : polylist X  
| polycons : X -> polylist X -> polylist X.
```

## pairs

```
Inductive natprod : Set :=  
| natpair : nat -> nat -> natprod.
```

```
Inductive boolprod : Set :=  
| boolpair : bool -> bool -> boolprod.
```

```
Inductive polyprod (X Y :Set) : Set :=  
| polypair : X -> Y -> polyprod X Y
```

## instantiation

using application

## bintrees

```
Inductive natbintree : Set :=  
  natleaf : natbintree  
| natnode :  
  natbintree -> nat -> natbintree -> natbintree.
```

```
Inductive boolbintree : Set :=  
| boolleaf : boolbintree  
| boolnode :  
  boolbintree -> bool -> boolbintree -> boolbintree.
```

```
Inductive polybintree (X : Set) : Set :=  
  polyleaf : polybintree X  
| polynode :  
  polybintree X -> X -> polybintree X -> polybintree X.
```

## lambda2: examples

```
λa: Set. λx: a. x  
λa: Prop. λx: a. x  
λa: *. λx: a. x
```

```
λa: Set. λx: a. λb: Set. λy: a → b. (y x)  
λa: Prop. λx: a. λb: Prop. λy: a → b. (y x)  
λa: *. λx: a. λb: *. λy: a → b. (y x)
```

```
λa: Set. λb: Set. λx: a → b. λy: a. (x y)  
λa: Prop. λb: Prop. λx: a → b. λy: a. (x y)  
λa: *. λb: *. λx: a → b. λy: a. (x y)
```

## lambda2: instantiation

by application and beta-reduction

## natural numbers in lambda2

- $N = \Pi a: * . a \rightarrow (a \rightarrow a) \rightarrow a$
- $n = \lambda a: * . \lambda o: a. \lambda s: a \rightarrow a. s^n o$
- $\text{successor} = \lambda n: N. \lambda a: * . \lambda o: a. \lambda s: a \rightarrow a. s (n a o s)$
- $\text{successorb} = \lambda n: N. \lambda a: * . \lambda o: a. \lambda s: a \rightarrow a. (n a (s o) s)$

## booleans in lambda2

- $B = \Pi a: * . a \rightarrow a \rightarrow a$
- $T = \lambda a: * . \lambda x: a. \lambda y: a. x$
- $F = \lambda a: * . \lambda x: a. \lambda y: a. y$
- $\text{not} = \lambda b: B. \lambda a: * . \lambda u: a. \lambda v: a. (b a v u)$
- $\text{and} = \lambda b: B. \lambda b': B. \lambda a: * . \lambda u: a. \lambda v: a. (b a (b' a u v) v)$
- $\text{or} = \lambda b: B. \lambda b': B. \lambda a: * . \lambda u: a. \lambda v: a. (b a u (b' a u v))$

## Curry-Howard-De Bruijn isomorphism

$\text{prop1} \sim \lambda \rightarrow$   
 $\text{pred1} \sim \lambda P$   
 $\text{prop2} \sim \lambda 2$

back to the examples

$\lambda a : \star . \lambda x : a . x : \Pi a : \star . \Pi x : a . a$

$\lambda a : \star . \lambda x : a . \lambda b : \star . \lambda y : a \rightarrow b . (y x) : \Pi a : \star . \Pi x : a . \Pi b : \star . \Pi y : a \rightarrow b . b$

$\lambda a : \star . \lambda b : \star . \lambda x : a \rightarrow b . \lambda y : a . (x y) : \Pi a : \star . \Pi b : \star . \Pi x : a \rightarrow b . \Pi y : a . b$

beta-reduction corresponds to detour-elimination