

MPRI 2-7-1

week 4 - Oct. 5th

Functions in HOL

base types : ι and o

HOL rules for \Rightarrow and \forall

constants: $0, S, + \times$

Axioms: $\forall x. 0+x = x, \forall x y. S(x) + y = S(x + y),$

$\forall x. 0 \times x = 0, \forall x y. S(x) \times y = x \times y + y,$

$\forall x. 0 \neq S(x),$ injectivity of S

induction

Can be extended with more base types and induction principles

Can be extended with the excluded middle

Implemented and used in real systems : HOL, HOL-light, Isabelle-HOL...

Very simple model

Model of simply typed λ -calculus, $|i| \equiv \mathbb{N}$, $|o| \equiv \{0, 1\}$

$|\Rightarrow| \equiv$ boolean implication

$|\forall_T|(A) \equiv \min_{\alpha \in |T|} |A|(\alpha)$

$|0| \equiv 0$, $|S| \equiv x \mapsto x+1$, ...

The formalism enjoys cut-elimination property

Intuitionistic proofs are constructive

The smallest set such that :

- ▶ $even(0)$
- ▶ $\forall x. even(x) \Rightarrow even(S(S(x)))$

Any set closed by the two properties contains *even* :

$$(even\ n) \equiv \forall X : 1 \rightarrow 0 .$$

$$(X\ 0) \Rightarrow$$

$$(\forall y. (X\ y) \Rightarrow (X\ (S\ (S\ y)))) \Rightarrow$$

$$(X\ n)$$

A Proof by induction

$$\forall X : \mathbb{1} \rightarrow \mathbb{0} .$$

$$(X \ 0) \Rightarrow$$

$$(\forall y . (X \ y) \Rightarrow (X \ (S \ (S \ y)))) \Rightarrow$$

$$(X \ n)$$

$$\left\{ \begin{array}{l} (\text{even } x) \Rightarrow \exists y . x = y + y \\ P \equiv \lambda x . \exists y . x = y + y \end{array} \right.$$

$$\exists y . 0 = y + y$$

$$(\exists y . 0 = y + y) \Rightarrow$$

$$(\forall x . \exists y . x = y + y \Rightarrow \exists y . (S \ (S \ x)) = y + y) \Rightarrow$$

$$\exists y . x = y + y$$

$$\exists y . x = y + y \Rightarrow \exists y . (S \ (S \ x)) = y + y$$

two induction cases to prove

A more advanced inductive predicate

What is a *strongly normalizing* term ?

No infinite path : $t \triangleright t_1 \triangleright t_2 \triangleright t_3 \triangleright \dots$

Define it inductively ?

$t \in \text{SN}$ iff $\forall t', t \triangleright t' \Rightarrow t' \in \text{SN}$

The smallest set s.t. $(\forall t', t \triangleright t' \Rightarrow t' \in \text{SN}) \Rightarrow t \in \text{SN}$

Only one clause !

base case : t is normal (then it is SN)

$(\text{SN } u) \equiv$

$\forall X : \Lambda \rightarrow \text{O} .$

$(\forall t : \Lambda . (\forall t' : \Lambda . (\beta t t') \Rightarrow X t') \Rightarrow X t)$

$\Rightarrow (X u)$

Using this definition

$$\forall X : \Lambda \rightarrow o . (\forall t : \Lambda . (\forall t' : \Lambda . (\beta t t') \Rightarrow X t') \Rightarrow t) \Rightarrow (X u)$$

Can we prove $(\beta u u)$ is false ?

$$(\forall t : \Lambda . (\forall t' : \Lambda . (\beta t t') \Rightarrow \neg(\beta t' t')) \Rightarrow \neg(\beta t t)) \Rightarrow \neg(\beta u u)$$

$$\forall t : \Lambda . (\forall t' : \Lambda . (\beta t t') \Rightarrow \neg(\beta t' t')) \Rightarrow \neg(\beta t t)$$

given t , $\forall t' : \Lambda . (\beta t t') \Rightarrow \neg(\beta t' t)$ show $\neg(\beta t t)$

$$(\beta t t) \Rightarrow \neg(\beta t t) \quad \text{indeed entails } \neg(\beta t t)$$

Specifying a recursive function

$$\text{Ack}(0, n) = (\text{S } n)$$

$$\text{Ack}(\text{S } m, 0) = \text{Ack}(m, (\text{S } 0))$$

$$\text{Ack}(\text{S } m, \text{S } n) = \text{Ack}(m, \text{Ack}(\text{S } m, n))$$

$\lambda a:i.\lambda b:i.\lambda r:i.$

$\forall X : i \rightarrow i \rightarrow i \rightarrow o .$

$(\forall n. (X 0 n (\text{S } n))) \Rightarrow$

$(\forall m. \forall r. (X m (\text{S } 0) r) \Rightarrow (X (\text{S } m) 0 r)) \Rightarrow$

$(\forall m. \forall n. \forall r. \forall r'. (X (\text{S } m) n r') \Rightarrow (X m r' r) \Rightarrow (X (\text{S } m)(\text{S } n) r)) \Rightarrow$

$(X a b r)$

Proving the existence of a recursive function

$$\begin{aligned}
 \text{Ack} \equiv & \lambda a:i.\lambda b:i.\lambda r:i. \\
 & \forall X : i \rightarrow i \rightarrow i \rightarrow o. \\
 & (\forall n. (X 0 n (S n))) \Rightarrow \\
 & (\forall m. \forall r. (X m (S 0) r) \Rightarrow (X (S m) 0 r)) \Rightarrow \\
 & (\forall m. \forall n. \forall r. \forall r'. (X (S m) n r') \Rightarrow (X m r' r) \Rightarrow (X (S m)(S n) r)) \Rightarrow \\
 & (X a b r)
 \end{aligned}$$

$$\forall a . \forall b . \exists r . (\text{Ack } a \ b \ r) \quad \text{by induction}$$

$$\text{induction over } a : \quad \forall b . \exists r . (\text{Ack } a \ b \ r)$$

$$\forall b . \exists r . (\text{Ack } 0 \ b \ r)$$

$$\forall b . \exists r . (\text{Ack } a \ b \ r) \Rightarrow \forall b . \exists r . (\text{Ack } (S \ a) \ b \ r)$$

Proving the existence of a recursive function

$\text{Ack} \equiv \lambda a:i.\lambda b:i.\lambda r:i.$

$\forall X : i \rightarrow i \rightarrow i \rightarrow o .$

$(\forall n. (X 0 n (S n)) \Rightarrow$

$(\forall m. \forall r. (X m (S 0) r) \Rightarrow (X (S m) 0 r)) \Rightarrow$

$(\forall m. \forall n. \forall r. \forall r'. (X (S m) n r') \Rightarrow (X m r' r) \Rightarrow (X (S m)(S n) r)) \Rightarrow$

$(X a b r)$

induction over a : $\forall b . \exists r . (\text{Ack } a b r)$

$\forall b . \exists r . (\text{Ack } 0 b r)$

$\forall b . \exists r . (\text{Ack } a b r) \Rightarrow \forall b . \exists r . (\text{Ack } (S a) b r)$

induction over b : $\exists r . (\text{Ack } (S a) b r)$

$\exists r . (\text{Ack } (S a) 0 r)$

$\exists r . (\text{Ack } (S a) (S b) r)$

Naming functions: Hilbert operator

Extending the language

$\varepsilon(P)$ "The" object verifying P ("choice operator")

$$\frac{\vdash (P t)}{\vdash (P \varepsilon(P))}$$

"If one guy can do it, it's ε "

$$\boxed{\exists x . P x \iff P \varepsilon(P)}$$

(can be used instead of \exists)

$$\text{exp_f a b} = \varepsilon(\lambda x . (\text{exp a b x}))$$

$$\text{exp_f} = \lambda a . \lambda b . \varepsilon(\lambda x . (\text{exp a b x}))$$

We can show $\text{exp_f a 0} = 1$, $\text{exp_f a (S b)} = a \times \text{exp_f a b}$

The proof of these equations can be mechanized

What do we miss ?

Computations !

Architecture of HOL implementations (in a nutshell)

- ▶ One does not construct the proof derivation (as a tree data structure)
- ▶ ML was invented as the meta-language of HOL implementations !
- ▶ Safety architecture :
 - An abstracted datatype for judgements $\Gamma \vdash A$
 - Only a few simple tactics allow to construct these judgements
 - These tactics correspond to logical rules
 - These tactics are the Trusted Computing Base
 - More complex tactics are assembled on top of those tactics (using ML)

How *unconstructive* is the ε operator ?

Remarks:

1. $\forall x . \forall y . x=y \vee x \neq y$ is provable in HA
2. $(A \vee \neg A) \wedge (B \vee \neg B) \Rightarrow (A \wedge B) \vee \neg(A \wedge B)$
3. $(A \vee \neg A) \wedge (B \vee \neg B) \Rightarrow (A \vee B) \vee \neg(A \vee B)$
4. $(A \vee \neg A) \wedge (B \vee \neg B) \Rightarrow (A \Rightarrow B) \vee \neg(A \Rightarrow B)$

Why is classical arithmetic undecidable ?

$$\forall x. A(x) \vee \neg A(x) \text{ does not entail } (\forall x. A(x)) \vee \neg(\forall x. A(x))$$

$$\text{does not entail } (\exists x. A(x)) \vee \neg(\exists x. A(x))$$

with ε , Heyting arithmetic becomes classical !

We prove that for any proposition A , $\vdash A \vee \neg A$ holds (is provable) by induction over the size of A

1. $\forall x . \forall y . x=y \vee x \neq y$ is provable in HA
2. $(A \vee \neg A) \wedge (B \vee \neg B) \Rightarrow (A \wedge B) \vee \neg(A \wedge B)$
3. $(A \vee \neg A) \wedge (B \vee \neg B) \Rightarrow (A \vee B) \vee \neg(A \vee B)$
4. $(A \vee \neg A) \wedge (B \vee \neg B) \Rightarrow (A \Rightarrow B) \vee \neg(A \Rightarrow B)$

Suppose we know :

$$\forall x . A(x) \vee \neg A(x) \quad (\exists x . A(x)) \vee \neg(\exists x . A(x))$$

$$\qquad \qquad \qquad (\exists x . A(x)) \vee \neg(\exists x . A(x))$$

We prove that for any proposition A , $\vdash A \vee \neg A$ holds (is provable)
by induction over the size of A number of connectives

We prove that for any proposition A , $\vdash A \vee \neg A$ holds (is provable)

by induction over the size of A

Suppose we know : $\vdash A \vee \neg A$ for all A with n number of connectives

$\vdash \forall x. A(x) \vee \neg A(x)$ let us prove $\vdash (\exists x. A(x)) \vee \neg(\exists x. A(x))$

by I.H : $\vdash A(\mathcal{E}(A)) \vee \neg A(\mathcal{E}(A))$

if $A(\mathcal{E}(A))$, then $\exists x. A(x)$ (trivial)

if $\neg A(\mathcal{E}(A))$: $\exists x. A(x)$ entails $A(\mathcal{E}(A))$, thus \perp .

so $\neg(\exists x. A(x))$

We prove that for any proposition A , $\vdash A \vee \neg A$ holds (is provable)

by induction over the size of A

Suppose we know : $\vdash A \vee \neg A$ for all A with n number of connectives

$\vdash \forall x. A(x) \vee \neg A(x)$ let us prove $\vdash (\exists x. A(x)) \vee \neg(\exists x. A(x))$

by I.H : $\vdash A(\mathcal{E}(A)) \vee \neg A(\mathcal{E}(A))$

if $A(\mathcal{E}(A))$, then $\exists x. A(x)$ (trivial)

if $\neg A(\mathcal{E}(A))$: $\exists x. A(x)$ entails $A(\mathcal{E}(A))$, thus \perp .

so $\neg(\exists x. A(x))$

$(\exists x. A(x)) \vee \neg(\exists x. A(x))$

We prove that for any proposition A , $\vdash A \vee \neg A$ holds (is provable) by induction over the size of A

Suppose we know :

$\vdash \forall x. A(x) \vee \neg A(x)$ let us prove $\vdash (\forall x. A(x)) \vee \neg(\forall x. A(x))$

~~$\vdash A(\varepsilon(A)) \vee \neg A(\varepsilon(A))$~~

by I.H : $\vdash A(\varepsilon(\neg A)) \vee \neg A(\varepsilon(\neg A))$

if $\neg A(\varepsilon(\neg A))$, then $\neg(\forall x. A(x))$ (trivial)

if $A(\varepsilon(\neg A))$: $\exists x. \neg A(x)$ entails $\neg A(\varepsilon(\neg A))$, thus \perp .

so $\neg(\exists x. \neg A(x))$

now, given x , we can show $\neg A(x) \Rightarrow \exists y. \neg A(y) \Rightarrow \perp$

so $\forall x. \neg\neg A(x)$ but $A(x) \vee \neg A(x)$

so $\forall x. A(x)$

Summing up

In other words :

Heyting arithmetic with Hilbert operator = Peano + Hilbert operator

Computing with epsilon is not easy

However, HOL (without epsilon and EM) is constructive

I was asked : what is the difference between HOL and system F ?

HOL : formalism quantification over propositions

System F : type system quantification over types

Link : when we view proofs as λ -terms (starting next week)

Normalization of system F (actually F_ω) will allow to show cut elimination in HOL