# MPRI
# PRFSYS

## Oct. 21st

## Martin-Löf's Type Theory

# We really need normalization

In naive set theory we had $R =_\beta \neg R$

Now we do not have such reductions over propositions, but still:

$g(0) = S(0)$
$g(S(\_)) = 0$

$f(n) = g(f(n))$

*(non termination)*

- $g(n) = 0 \vee g(n) = 1$
- $g(f(0)) = 0 \implies g(g(f(0))) = 1 \implies 0=1$
- $g(f(0)) = 1 \implies g(g(f(0))) = 0 \implies 0=1$

# Going further: System T

Adding primitive natural numbers and a recursor

In Coq:

```
Inductive nat : Type :=
  |  O : nat
  |  S : nat -> nat.
```

Additional atomic type: N

Additional constants: $0 : N, S : N \to N$

Additional operators: $R_T : N \to T \to (N \to T \to T) \to T$

$$R\ 0\ t_0\ t_S \quad \triangleright \quad t_0$$

$$R\ (S\ n)\ t_0\ t_S \quad \triangleright \quad t_S\ n\ (R\ n\ t_0\ t_S)$$

# A terminating programming language

- Incomplete (no chance to have all algorithms)
- But quite expressive
- Can be made even more expressive (general inductive types, System F…)

$$\text{add} = \lambda\, n.\, \lambda\, m.\ \ (R_N\ \ n \quad m \quad \lambda p.\lambda r.\ S(r))$$

$$\text{mul} = \lambda\, n.\, \lambda\, m.\ \ (R_N\ \ n \quad 0 \quad \lambda p.\lambda r.\ (\text{add}\ m\ r))$$

quite straightforward for all primitive recursive functions

# A non-primitive recursive function

(of course, Ackermann)

$Ack(0, b) = S(b)$

$Ack(S(a), 0) = Ack(a, 1)$

$Ack(S(a), S(b)) = Ack(a, Ack(S(a), b))$

$$Ack = \lambda\, a.\, \lambda\, b.\, R_{N \to N}\ a\ (\lambda x.\ (S\ x))\ (\lambda\ \_.\ \lambda\, f.\ \lambda\, x.\ R_N\ x\ (f\ 1)\ (\lambda\ \_.\ \lambda\, r.\ (f\ r))\ b)$$

# A glance to a wider world: "ordinals"

System T is a small scale model for more general inductive types

$0_o$ : ord
$S_o$ : ord → ord
lim : (N → ord) → ord

idea: lim ($u_0$, $u_1$, … $u_k$ …) is the upper bound of the sequence

so $u_k < \lim(u_0, u_1, \ldots u_k \ldots)$

or formally $(f\ k) < \lim f$

$R^o_T$ : ord → T → (ord → T → T ) → ((N → ord) →(N → T) → T) → T

$R^o\ 0_o\ t_0\ t_S\ t_l \quad \triangleright \quad t_0$

$R^o\ (S_o\ n)\ t_0\ t_S\ t_l \quad \triangleright \quad t_S\ n\ (R^o\ n\ t_0\ t_S\ t_l)$

$R^o\ (\lim f)\ t_0\ t_S\ t_l \quad \triangleright \quad t_l\ f\ \lambda n.(R^o\ (f\ n)\ t_0\ t_S\ t_l)$

# From λ→Σ+ to Type Theory

We have:

- Dependent types, encodes FOL up to cut–elimination for logical cuts (for $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$)
- System T: allows the definition of complex computations

$$\mathrm{R_T} : \mathrm{T} \to (\mathrm{N} \to \mathrm{T} \to \mathrm{T}) \to \mathrm{N} \to \mathrm{T}$$

Let us add the induction axiom :

$$\mathrm{R_P} : (\mathrm{P}\ 0) \to (\Pi\ \mathrm{x} : \mathrm{N}\ .\ (\mathrm{P}\ \mathrm{n}) \to (\mathrm{P}\ (\mathrm{S}\ \mathrm{n}))) \to \Pi\ \mathrm{n} : \mathrm{N}\ .\ (\mathrm{P}\ \mathrm{n})$$

$$(\mathrm{R_P}\ \ \mathrm{p_0}\ \mathrm{p_S}\ \ 0) \quad \triangleright \quad \mathrm{p_0}$$

$$(\mathrm{R_P}\ \ \mathrm{p_0}\ \mathrm{p_S}\ \ (\mathrm{S}\ \mathrm{n})) \quad \triangleright \quad \mathrm{p_S}\ \mathrm{n}\ \ (\mathrm{R_P}\ \ \mathrm{p_0}\ \mathrm{p_S}\ \mathrm{n})$$

# Subject reduction

$(\mathrm{R_P}\ \mathrm{p_0}\ \mathrm{p_S}\ \ 0) \quad \triangleright \quad \mathrm{p_0}$

$\mathrm{R_P} : (\mathrm{P}\ 0) \to (\Pi\ \mathrm{x} : \mathrm{N}\ .\ (\mathrm{P}\ \mathrm{n}) \to (\mathrm{P}\ (\mathrm{S}\ \mathrm{n}))) \to \Pi\ \mathrm{n} : \mathrm{N}\ .\ (\mathrm{P}\ \mathrm{n})$

Suppose $(\mathrm{R_P}\ \mathrm{p_0}\ \mathrm{p_S}\ \ 0) : \mathrm{T}$

so $\mathrm{p_0}$: $(\mathrm{P}\ 0)$

so $\mathrm{p_S}$: $(\Pi\ \mathrm{x} : \mathrm{N}\ .\ (\mathrm{P}\ \mathrm{n}) \to (\mathrm{P}\ (\mathrm{S}\ \mathrm{n})))$

so: $0 : \mathrm{N}$ (it is)

so: $\mathrm{T} =_\beta \mathrm{P}\ 0$

so: $\mathrm{p0} : \mathrm{T}$

Obviously similar for

$(\mathrm{R_P}\ \mathrm{p_0}\ \mathrm{p_S}\ \ (\mathrm{S}\ \mathrm{n})) \quad \triangleright \quad \mathrm{p_S}\ \mathrm{n}\ \ (\mathrm{R_P}\ \mathrm{p_0}\ \mathrm{p_S}\ \mathrm{n})$

# Equality and equality cuts

$\mathrm{I_T} : \mathrm{T} \to \mathrm{T} \to \mathrm{Type}$ (provided T : Type)

$\mathrm{refl_T} : \Pi\ \mathrm{x} : \mathrm{T}\ .\ \mathrm{I_T}\ \mathrm{x}\ \mathrm{x}$

$\mathrm{L_P} : \Pi\ \mathrm{x} : \mathrm{T}.\ \Pi\ \mathrm{y} : \mathrm{T}.\ \mathrm{P}\ \mathrm{x} \to \mathrm{I_T}\ \mathrm{x}\ \mathrm{y} \to \mathrm{P}\ \mathrm{y}$

formally:

$$\frac{\Gamma \vdash \mathrm{T} : \mathrm{Type}}{\Gamma \vdash \mathrm{I_T} : \mathrm{T} \to \mathrm{T} \to \mathrm{Type}}$$

$\mathrm{L_P}\ \mathrm{a}\ \mathrm{b}\ \mathrm{p}\ (\mathrm{refl_T}\ \mathrm{c}) \rhd \mathrm{p}$

$\mathrm{L_P}\ \mathrm{a}\ \mathrm{b}\ \mathrm{p}\ (\mathrm{refl_T}\ \mathrm{c}) : \mathrm{Q}$

so : $\mathrm{P} : \mathrm{T} \to \mathrm{Type}, \quad \mathrm{a} : \mathrm{T}, \quad \mathrm{b} : \mathrm{T}, \quad \mathrm{p} : \mathrm{P}\ \mathrm{a}, \quad \mathrm{Q} =_\beta \mathrm{P}\ \mathrm{b}$

since $(\mathrm{refl_T}\ \mathrm{c}) : \mathrm{I_T}\ \mathrm{c}\ \mathrm{c}$, we have $\mathrm{I_T}\ \mathrm{c}\ \mathrm{c} =_\beta \mathrm{I_T}\ \mathrm{a}\ \mathrm{b}$, so $\mathrm{a} =_\beta \mathrm{b} =_\beta \mathrm{c}$

so by conversion, $\mathrm{p} : \mathrm{P}\ \mathrm{b}$

# A last detail

We want to talk about falsity

$$\frac{\Gamma \text{ wf}}{\Gamma \vdash \bot : \text{Type}} \qquad \frac{\Gamma \vdash t : \bot \quad \Gamma \vdash T : \text{Type}}{\Gamma \vdash \text{efq}_T(t) : T}$$

Nothing more to be added

# Normalization

We can still map normalization to the corresponding system without dependent types. That is System T with product and sum types

| | |
|---|---|
| $\mathrm{N}$ | $\mathrm{N}$ |
| $\mathrm{I_T\ t\ u}$ | $\mathrm{N}$ |
| $\Sigma\ \mathrm{x{:}A.B}$ | $\mathrm{A \times B}$ |
| $\mathrm{A+B}$ | $\mathrm{A+B}$ |
| $\Pi\ \mathrm{x{:}A.B}$ | $\mathrm{A \to B}$ |

# Closed normal terms

A closed normal term of type … is of the form(s) …

| | |
|---|---|
| $\mathrm{N}$ | $0,\ \mathrm{S(t)}$ |
| $\mathrm{I_T\ t\ u}$ | $\mathrm{refl_T\ v}$ |
| $\Sigma\ \mathrm{x{:}A.B}$ | $(\mathrm{a},\ \mathrm{b})$ |
| $\mathrm{A{+}B}$ | $\mathrm{i(a)},\ \ \mathrm{j(b)}$ |
| $\Pi\ \mathrm{x{:}A.B}$ | $\lambda\ \mathrm{x{:}A'.t}$ |
| $\bot$ | none |

# The constructive paradise

By normalizing a proof of $\Sigma\ x{:}A.B$, we obtain $t{:}A$ and $p : B[x \setminus t]$

By normalizing a proof of $A+B$, we obtain either $i(A)$ or $j(b)$

From a proof $t : \Pi\ x{:}A.\ \Sigma\ y{:}B.C$ we can obtain:

$f : A \to B$

$p : \Pi\ x{:}A.\ C[y \setminus (f\ x)]$

$f \equiv \lambda\ x{:}A\ .\ \pi_1(t\ x)$

$p \equiv \lambda\ x{:}A\ .\ \pi_2(t\ x)$

(furthermore $f$ is typable in System T)

There is no term $t$ s.t. $[A{:}\mathrm{Type}] \vdash t : A + (A \to \bot)$

There are closed $A{:}\mathrm{Type}$ with no term $t$ s.t. $[A{:}\mathrm{Type}] \vdash t : A + (A \to \bot)$

(and other variants)

# Limits of this Type Theory

Suppose we have $[] \vdash \mathrm{p} : \mathrm{I_N}\ 0\ (\mathrm{S}\ 0) \to \bot$

by erasing type dependencies, we get: $[] \vdash |p| : \mathrm{N} \to \bot$

and thus a closed term of type $\bot$ (in System T or MLTT)

Hence: 0≠1 is not provable in MLTT.

Indeed having a discrimination predicate means having "really dependent types" :

EQZ 0 $\triangleright$ $\top$

EQZ (S _) $\triangleright$ $\bot$

What happens in Rocq ?

# Historical CoC

```
nat_rec : Π P : nat→ Type. P 0 →
                  (Π m : nat. P m → P (S m)) →
                    Π n:nat. P n
```

```
nat_rect : Π P : nat→ Kind. P 0 →
                  (Π m : nat. P m → P (S m)) →
                    Π n:nat. P n
```

We can then define

EQZ ≡ (nat_rect λ x:nat.Type

True

λ m:nat. λ X:Type. False)

# Modern Rocq

```
Definition EQZ (n:nat) : Type :=
  match n with
   | 0 => True
   | S _ => False
end.
```

In Rocq, operators like $R_T$ are not primitive, but built by combining:

- pattern-matching
- structural recursion

# Universes

In Rocq: $\mathrm{Type}_1 : \mathrm{Type}_2 : \mathrm{Type}_3 : \ldots$
together with pattern-matching (R operators) towards all $\mathrm{Type}_i$

It is interesting to look at the restrictions over eliminations of inductive types to keep the system consistent (but done in 2-7-2 ?)

Martin-Löf's proposal:
An addition type $\mathrm{U}$: $\mathrm{Type}$ (universe)
"constructors" for this type:

$\mathrm{n}$:$\mathrm{U}$

$\mathrm{bot} : \mathrm{U}$

$\pi : \Pi\ \mathrm{a}{:}\mathrm{U}.\ ((\mathrm{tr}\ \mathrm{a}) \to \mathrm{U}) \to \mathrm{U}$

$\ldots$

with:

$\mathrm{tr} : \mathrm{U} \to \mathrm{Type}$

$\mathrm{tr}\ \mathrm{n} \rhd \mathrm{nat}$

$\mathrm{tr}\ \mathrm{bot} \rhd \bot$

$\mathrm{tr}\ (\pi\ \mathrm{a}\ \mathrm{f}) \to \Pi\ \mathrm{x}{:}\mathrm{tr}\ \mathrm{a}.\ \mathrm{tr}(\mathrm{f}\ \mathrm{x})$

# Type Universes in Martin-Löf style

Martin-Löf's proposal:

An addition type $\mathrm{U}$: $\mathrm{Type}$ (universe)

"constructors" for this type:

$\mathrm{n}$:$\mathrm{U}$

$\mathrm{bot} : \mathrm{U}$

$\pi : \Pi\ \mathrm{a}{:}\mathrm{U}.\ (\mathrm{tr}\ \mathrm{a}) \to \mathrm{U}) \to \mathrm{U}$

$\ldots$

with:

$\mathrm{tr} : \mathrm{U} \to \mathrm{Type}$

$\mathrm{tr}\ \mathrm{n} \rhd \mathrm{nat}$

$\mathrm{tr}\ \mathrm{bot} \rhd \bot$

$\mathrm{tr}\ (\pi\ \mathrm{a}\ \mathrm{f}) \to \Pi\ \mathrm{x}{:}\mathrm{tr}\ \mathrm{a}.\ \mathrm{tr}(\mathrm{f}\ \mathrm{x})$

U is an inductive type which allows to model all types (except U)
tr is defined by pattern-matching + recursion
but tr occurs in the definition of U: so-called inductive-recursive type
(this last feature is not available in Rocq)

What does a universe hierarchy look like in this setting ?
Which extension would be paradoxical ?

# Rocq Exercise

Construct div2 using only:

- Definition
- nat_rec (or match with + Fixpoint)

```
Definition P2 n :=
          {p : nat & {n = p + p}+{ n = S (p + p)}} .

div2 (n : nat) : P2 n
```