LMFI
Second-order quantification and fixed-points in logic
First lecture: Gödel's System T

Alexis Saurin

4th january 2022

1 On the weak expressiveness of STLC

Definition 1.1

Extended polynomials *are the functions generated by* 0, 1, *and the identity function the operations of addition, multiplication and conditional.*

Theorem 1.2 (***Schwichtenberg and Statman***)

The arithmetical functions definable in simply-typed λ*-calculus over type* Nat *are exactly the extended polynomials.*

$\mathsf{Nat} = (o \to o) \to (o \to o)$. $[0] = o$

$\mathsf{Nat} = [2]$. $[n+1] = [n] \to [n]$.

$\boxed{[n+2],\ n \geq 0}$ $\underline{\mathsf{Nat}^i = [i+2]}$.

Several solutions are available to improve this expressiveness issue. We shall now consider an option investigated by Gödel, extending the simply-typed λ-calculus with types for pairs of objects, atomic types for booleans and naturals and constructions for conditional branching and a recursor.

Another option that will be investigated in the following lecture will consist in allowing the λ-terms to be polymorphic, that is to be applied to arguments of variable types : this will be the core of System F and of the connection with second-order logic.

2 Gödel's system T. 2.1 Types and terms of system T

Several systems have been considered to increase the class of (total) functions that can be represented in the typed setting. ***Gödel's System T*** is such a system, extending the simply-typed λ-calculus with product types ($U \times V$), a type for booleans (Bool), with a type for natural numbers (Nat) and with the following term constructions :

(i) pairs and projections : $\langle t, u\rangle, \pi_1(t), \pi_2(t)$;

(ii) boolean constants and a boolean test : $\mathsf{true}, \mathsf{false}, \mathsf{if}\ t\ \mathsf{then}\ u\ \mathsf{else}\ v$;

(iii) constants for representing natural numbers and a recursor for each type A : $\mathsf{S}(t), 0, \mathsf{Rec}(t, u, v)$.

$A \to B$

$U \times V$.

$\left.\begin{matrix}\mathsf{Bool} \\ \mathsf{Nat} \\ A\end{matrix}\right\} \mathcal{T}_{At}$.

Definition 2.1 (*simple types for system* **T**)

We consider a countable set $\mathcal{T}_{\mathsf{At}}$ *of atomic types containing* Nat *and* Bool. T-*types are defined inductively as*

$$T, U, V ::= A \mid U \times V \mid U \to V \qquad A \in \mathcal{T}_{\mathsf{At}}.$$

$$\frac{}{x^U : U}\ (Var)\quad (x \in \mathcal{V}^U) \qquad \frac{t : T}{\lambda x^U.t : U \to T}\ (Abs)\quad (x \in \mathcal{V}^U) \qquad \frac{t : U \to T \quad u : U}{(t)u : T}\ (App)$$

const / $\wedge_i$

$$\frac{u : U \quad v : V}{\langle u, v\rangle : U \times V}\ (Prod) \qquad \frac{t : U_1 \times U_2}{\pi_1(t) : U_1}\ (Proj_1) \qquad \frac{t : U_1 \times U_2}{\pi_2(t) : U_2}\ (Proj_2)$$

Destructors

$\wedge_e^1$ / $\wedge_e^2$.

$$\frac{}{\mathsf{true} : \mathsf{Bool}}\ (\mathsf{true}) \qquad \frac{}{\mathsf{false} : \mathsf{Bool}}\ (\mathsf{false}) \qquad \frac{}{0 : \mathsf{Nat}}\ (0) \qquad \frac{t : \mathsf{Nat}}{\mathsf{S}(t) : \mathsf{Nat}}\ (\mathsf{S})$$

$$\frac{A \vee B \quad \begin{matrix}[A] \\ \vdots \\ C\end{matrix} \quad \begin{matrix}[B] \\ \vdots \\ C\end{matrix}}{C}\ \vee_e$$

$$\frac{t : \mathsf{Bool} \quad u : U \quad v : U}{\mathsf{if}\ t\ \mathsf{then}\ u\ \mathsf{else}\ v : U}\ (\mathsf{If}) \qquad \frac{t : \mathsf{Nat} \quad u : \mathsf{Nat} \to (U \to U) \quad v : U}{\mathsf{Rec}(t, u, v) : U}\ (\mathsf{Rec})$$

Definition 2.3 (T-*reduction relation*)

We define the **T-reduction**, *written* $\longrightarrow_{\mathsf{T}}$, *as the least compatible relation on* T*-terms, containing typed* β*-reduction as well as :*

$$
\begin{array}{rcl}
\pi_i(\langle t_1, t_2\rangle) & \longrightarrow_{\mathsf{T}} & t_i \\
\mathsf{if\ true\ then}\ t\ \mathsf{else}\ u & \longrightarrow_{\mathsf{T}} & t \\
\mathsf{if\ false\ then}\ t\ \mathsf{else}\ u & \longrightarrow_{\mathsf{T}} & u \\
\mathsf{Rec}(0, v, w) & \longrightarrow_{\mathsf{T}} & w \ : U \\
\mathsf{Rec}(\mathsf{S}(t), v, w) & \longrightarrow_{\mathsf{T}} & (v)t\mathsf{Rec}(t, v, w)
\end{array}
$$

A **T-normal form** *is a* T*-term that does not* $\longrightarrow_{\mathsf{T}}$*-reduce to any* T*-term.*

• $\forall t : \mathsf{Nat}$ closed, $\exists n \in \mathbb{N}, t \longrightarrow^*_{\mathsf{T}} S^n(0)$.

• $\forall t : \mathsf{Bool}$ closed. $t \longrightarrow^*_{\mathsf{T}}$ true or $t \longrightarrow^*_{\mathsf{T}}$ false

once we have proved the normalization thm.

Proposition 2.4

Assume that t is a **closed** T*-normal. Prove that :*

- *If* $t : \mathsf{Nat}$, *then there exists* $n \in \mathbb{N}$ *such that* $t = \mathsf{S}^n(0)$ *;*
- *If* $t : \mathsf{Bool}$, *then* $t = \mathsf{true}$ *or* $t = \mathsf{false}$ *;*
- *If* $\vdash t : A \times B$, *then* $t = \langle u, v\rangle$ *;*
- *If* $t : U \to V$, *then* $t = \lambda x.\, u$.

$$
\mathit{ar})\quad (x \in \mathcal{V}^U) \qquad \frac{t : T}{\lambda x^U.t : U \to T}\ (\mathit{Abs})\quad (x \in \mathcal{V}^U) \qquad \frac{t : U \to T \quad u : U}{(t)u : T}\ (\mathit{App})
$$

$$
\frac{u : U \quad v : V}{\langle u, v\rangle : U \times V}\ (\mathit{Prod}) \qquad \frac{t : U_1 \times U_2}{\pi_1(t) : U_1}\ (\mathit{Proj}_1) \qquad \frac{t : U_1 \times U_2}{\pi_2(t) : U_2}\ (\mathit{Proj}_2)
$$

$$
\frac{}{\mathsf{true} : \mathsf{Bool}}\ (\mathsf{true}) \qquad \frac{}{\mathsf{false} : \mathsf{Bool}}\ (\mathsf{false}) \qquad \frac{}{0 : \mathsf{Nat}}\ (0) \qquad \frac{t : \mathsf{Nat}}{\mathsf{S}(t) : \mathsf{Nat}}\ (\mathsf{S})
$$

$$
\frac{t : \mathsf{Bool} \quad u : U \quad v : U}{\mathsf{if}\ t\ \mathsf{then}\ u\ \mathsf{else}\ v : U}\ (\mathsf{If}) \qquad \frac{t : \mathsf{Nat} \quad u : \mathsf{Nat} \to (U \to U) \quad v : U}{\mathsf{Rec}(t, u, v) : U}\ (\mathsf{Rec})
$$

2.2 Strong normalization theorem

Definition 2.5 (*Neutral* T*-term*)

A T*-term is* **neutral** *if it is not of the form* $\lambda x^U : t$, $\langle t, u\rangle$, true, false, 0 *or* $\mathsf{S}(t)$.

The sets $\mathsf{Neut}^{\mathsf{SN}}(U)$, $\mathsf{SNorm}(U)$ are adapted to T-terms ***without any change*** (but the dependency of $\mathsf{Neut}^{\mathsf{SN}}(U)$ with $\mathsf{RED}^{\mathsf{SN}}(U)$...) :

Definition 2.6 ($\mathsf{Neut}^{\mathsf{SN}}(T)$)

$$\mathsf{Neut}^{\mathsf{SN}}(T) = \{t \in \Lambda;\ t \text{ est neutre de type } T \text{ et } \forall t', t \longrightarrow t', t' \in \mathsf{RED}^{\mathsf{SN}}(T)\}$$

Definition 2.7 ($\mathsf{SNorm}(T)$)

$$\mathsf{SNorm}(T) = \{t \in \Lambda;\ t \text{ fortement normalisable de type } T\}.$$

Definition 2.8

- $\mathsf{RED}^{\mathsf{SN}}(X) = \mathsf{SNorm}(X)$
- $\mathsf{RED}^{\mathsf{SN}}(U \to V) = \{t : U \to V; \forall u \in \mathsf{RED}^{\mathsf{SN}}(U), (t)\, u \in \mathsf{RED}^{\mathsf{SN}}(V)\}$.
- $\mathsf{RED}^{\mathsf{SN}}(U_1 \times U_2) = \{t : U_1 \times U_2 \mid \forall i \in \{1, 2\}, \pi_i(t) \in \mathsf{RED}^{\mathsf{SN}}(U_i)\}$.

Lemma 2.9 (*Adapation*)

For every type T, *one has* $\mathsf{Neut}^{\mathsf{SN}}(T) \subseteq \mathsf{RED}^{\mathsf{SN}}(T) \subseteq \mathsf{SNorm}(T)$.

Lemma 2.14 (*Adequation*)

Let $t : U$ *with free variables among* $x_1^{T_1}, \ldots, x_n^{T_n}$. *For any* $(u_i \in \mathsf{RED}^{\mathsf{SN}}(T_i))_{1 \le i \le n}$, *one has* $t\{u_i/x_i\} \in \mathsf{RED}^{\mathsf{SN}}(U)$.

Normalization proof:

$t : U$, prove t is SN

We need to prove $t \in \mathsf{SNorm}(U)$.

It is sufficient to prove that t is reducible. (by adaptation lemma).

$t\{x_i^{T_i}/x_i\} = t$

$x_i^{T_i} \in \mathsf{Neut}^{\mathsf{SN}}(T_i)$.

$\subseteq \mathsf{Red}^{\mathsf{SN}}(T_i)$.

By adequation, $t = t\{x_i^{T_i}/x_i, 1 \le i \le n\} \in \mathsf{Red}^{\mathsf{SN}}(U)$

Lemma 2.10

For any type T, $\mathsf{RED}^{\mathsf{SN}}(T)$ is closed by β-reduction :

$$t \in \mathsf{RED}^{\mathsf{SN}}(T), \qquad t \xrightarrow{\;T\;} t' \qquad \Rightarrow \qquad t' \in \mathsf{RED}^{\mathsf{SN}}(T).$$

Démonstration : The lemma is proved by induction on the structure of type T.

- If T is atomic, as in STLC
- If $T = U \to V$, as in STLC → $t \in \mathrm{Red}(T)$.
- If $T = U_1 \times U_2$, then let $t : T$ such that $t \longrightarrow t'$. Since t is reducible, its projection are : $\pi_i(t) \in \mathsf{RED}^{\mathsf{SN}}(U_i), i \in \{1, 2\}$. By applying induction hypothesis on U_1 and U_2, we know that $\mathsf{RED}^{\mathsf{SN}}(U_i)$ are closed by reduction and since $\pi_i(t) \longrightarrow \pi_i(t')$ with $i \in \{1, 2\}$, we have that $\pi_i(t') \in \mathsf{RED}^{\mathsf{SN}}(U_i)$ for $i \in \{1, 2\}$. Therefore $t' \in \mathsf{RED}^{\mathsf{SN}}(T)$.

□

Démonstration of lemma 2.9 : The proof is by induction on the structure of type T :

- If $T = X$, as for STLC
- If $T = U \to V$, as for STLC
- If $T = U_1 \times U_2$, then :
  - $\mathsf{Neut}^{\mathsf{SN}}(T) \subseteq \mathsf{RED}^{\mathsf{SN}}(T)$:
    Let $t \in \mathsf{Neut}^{\mathsf{SN}}(T)$. Since t is neutral, $\pi_i(t)$ cannot be a redex itself : its redexes are necessarily in t, so that its one-step reducts are all of the form $\pi_i(t')$ with $t \longrightarrow t'$. Since $t \in \mathsf{Neut}^{\mathsf{SN}}(T)$, $t' \in \mathsf{RED}^{\mathsf{SN}}(T)$ and $\pi_i(t') \in \mathsf{RED}^{\mathsf{SN}}(U)_i, i \in \{1, 2\}$.
    Therefore we have that $\pi_i(t), i \in \{1, 2\}$ are neutral and all their one-step reducts are reducible : $\pi_i(t) \in \mathsf{Neut}^{\mathsf{SN}}(U)_i, i \in \{1, 2\}$.
    By induction hypothesis on U_1 and U_2, $\pi_i(t) \in \mathsf{RED}^{\mathsf{SN}}(U_i), i \in \{1, 2\}$.
    By definition of reducibility at product type, one concludes that $t \in \mathsf{RED}^{\mathsf{SN}}(T)$ as expected.
  - $\mathsf{RED}^{\mathsf{SN}}(T) \subseteq \mathsf{SNorm}(T)$:
    Assume that $t \in \mathsf{RED}^{\mathsf{SN}}(T)$. The $\pi_1(t) \in \mathsf{RED}^{\mathsf{SN}}(U_1)$ by definition and, by induction hypothesis on U_1, $\pi_1(t) \in \mathsf{SNorm}(U_1)$. The longest reduction from t is certainly at ~~least as long as that from $\pi_1(t)$ so there~~ is only finite reduction sequence from t and ~~$t \in \mathsf{SNorm}(T)$~~.
    $\Rightarrow t \in \mathsf{SNorm}(T)$.

□

The longest reduction from $\pi_1(t)$

$t = t_0 \to t_1 \to t_2 \to \cdots \to t_n$

$\pi_1(t_0) \to \pi_1(t_1) \to \pi_1(t_2) \to \cdots \to \pi_1(t_n)$.

$\pi_i(t) \to u$

$\exists t', t \to t'$ & $u = \pi_i(t')$.

$\in \mathrm{Red}(U_1 \times U_2)$

$\in \mathrm{Red}(U_i)$.

$\pi_i(t) \in \mathrm{Neut}(U_i) \subseteq \mathrm{Red}(U_i)$.

Lemma 2.12

$$(\forall u \in \mathsf{RED}^{\mathsf{SN}}(U), v\{u/x\} \in \mathsf{RED}^{\mathsf{SN}}(V)) \Rightarrow \forall u \in \mathsf{RED}^{\mathsf{SN}}(U), (\lambda x.v)u \in \mathsf{RED}^{\mathsf{SN}}(V).$$

$\longrightarrow \lambda x.u \in Red(U \to V).$

The following is a corresponding result for pairs :

Lemma 2.13

$$\forall u \in \mathsf{RED}^{\mathsf{SN}}(U), v \in \mathsf{RED}^{\mathsf{SN}}(V), \langle u, v\rangle \in \mathsf{RED}^{\mathsf{SN}}(U \times V).$$

Démonstration : By adaptation lemma, one can reason using the strong normalisation of u, v and therefore reason by induction on the sum of the length of the longest reductions from u and v to show that $\pi_i(\langle u, v\rangle)$ is reducible.

First notice that this term is neutral. Therefore, to show that is it reducible, it is sufficient to show that every one-step reduct is reducible from which one deduce that $\pi_i(\langle u, v\rangle) \in \mathsf{Neut}^{\mathsf{SN}}(U)$ and, by adaptation, that it is reducible.

$\pi_i(\langle u, v\rangle)$ reduces (i) either to u (resp. v) which is reducible, (ii) or to $\pi_i(\langle u', v\rangle)$ with $u \longrightarrow u'$. u' is reducible since reducibility is closed by reduction and its longest reduction is shorter than that of u so by induction hypothesis, $\pi_i(\langle u', v\rangle)$ is reducible, (iii) or to $\pi_i(\langle u, v'\rangle)$ with $v \longrightarrow v'$ which is reducible by exactly the same reasoning as in (ii).

Therefore both projections of $\langle u, v\rangle$ are reducible showing that $\langle u, v\rangle \in \mathsf{RED}^{\mathsf{SN}}(U \times V)$. □

Proposition 2.11

The following holds :

1. $0 \in \mathsf{RED}^{\mathsf{SN}}(\mathsf{Nat})$.
2. $\mathsf{true}, \mathsf{false} \in \mathsf{RED}^{\mathsf{SN}}(\mathsf{Bool})$.
3. $\forall t \in \mathsf{RED}^{\mathsf{SN}}(\mathsf{Nat}), \mathsf{S}(t) \in \mathsf{RED}^{\mathsf{SN}}(\mathsf{Nat})$.
4. $\forall t \in \mathsf{RED}^{\mathsf{SN}}(\mathsf{Bool}), \forall u, v \in \mathsf{RED}^{\mathsf{SN}}(U),$ if t then u else $v \in \mathsf{RED}^{\mathsf{SN}}(\mathsf{Bool})$.
5. $\forall t \in \mathsf{RED}^{\mathsf{SN}}(\mathsf{Nat}), \forall u \in \mathsf{RED}^{\mathsf{SN}}(\mathsf{Nat} \to (U \to U)), \forall v \in \mathsf{RED}^{\mathsf{SN}}(U), \mathsf{Rec}(t, u, v) \in \mathsf{RED}^{\mathsf{SN}}(U)$.

$\Gamma \quad \lambda x^U . x^U : U \to U$. (church style)

Curry style: $\lambda x.x = I$.

$\vdash I : U \to U$ for any U.

$\forall X(X \to X)$.

$\vdash I : \forall X(X \to X)$.

$\forall t \in F$ st $\vdash t : \forall X.(X \to X)$,

then $t \to \lambda x.x$.

$\vdots$

Lemma 2.14 (*Adequation*)

Let $t : U$ with free variables among $x_1^{T_1}, \dots, x_n^{T_n}$. For any $(u_i \in \mathsf{RED}^{\mathsf{SN}}(T_i))_{1 \le i \le n}$, one has $t\{u_i/x_i\} \in \mathsf{RED}^{\mathsf{SN}}(U)$.

$\langle u\{u_i/x_i\}, v\{u_i/x_i\}\rangle$

$= \langle u, v\rangle \{u_i/x_i\}$.

Démonstration du lemme 2.14: One reason by induction on the structure of $t : T$.

- If $t = x_i^{T_i}$, as for STLC.
- If $t = \lambda x^U.t'$, as for STLC.
- If $t = (u)v$, as for STLC.
- If $t = \langle u, v\rangle$, then by induction hypothesis, both $u\{u_i/x_i\}$ and $v\{u_i/x_i\}$ are reducible and by the previous lemma $t\{u_i/x_i\}$ is reducible. ✓
- If $t = \pi_1(u)$ (resp $\pi_2(u)$), then by induction hypothesis $u\{u_i/x_i\}$ is reducible which implies that $\pi_1(u\{u_i/x_i\})$ is reducible by definition.
- If t is some T-constant, it is reducible (since $0 \in \mathsf{RED}^{\mathsf{SN}}(\mathsf{Nat})$, $\mathsf{true}, \mathsf{false} \in \mathsf{RED}^{\mathsf{SN}}(\mathsf{Bool})$).
- If $t = \mathsf{S}(u)$, then by induction hypothesis, $u\{u_i/x_i\}$ is reducible and so is $\mathsf{S}(u\{u_i/x_i\})$.
- If $t = \mathsf{if}\ u\ \mathsf{then}\ v\ \mathsf{else}\ w$, then by induction hypothesis, $u\{u_i/x_i\}$, $v\{u_i/x_i\}$, $w\{u_i/x_i\}$ are reducible and so is $\mathsf{if}\ u\{u_i/x_i\}\ \mathsf{then}\ v\{u_i/x_i\}\ \mathsf{else}\ w\{u_i/x_i\}$.
- If $t = \mathsf{Rec}(u, v, w)$, then by induction hypothesis, $u\{u_i/x_i\}$, $v\{u_i/x_i\}$, $w\{u_i/x_i\}$ are reducible and so is $\mathsf{Rec}(u\{u_i/x_i\}, v\{u_i/x_i\}, w\{u_i/x_i\})$. $= t\{u_i/x_i\} \in Red(U)$

$u : Nat,\ v : Nat \to (U \to U),\ w : U$

□

$\forall T$ type of system T and $\forall t : T$, t has only finite reduction sequences.

Theorem 2.15

System T *is strongly normalizing.*

Démonstration : Let $t : T$ of free variables $(x_i^{T_i})_{1 \le i \le n}$. Be adaptation lemma 2.9 for any $1 \le i \le n$, $x_i^{T_i} \in \mathsf{RED}^{\mathsf{SN}}(T_i)$ since variables of type T are neutral and normal and therefore in $\mathsf{Neut}^{\mathsf{SN}}(T)$.

Adequation lemma 2.14 ensures that $t\left\{x_i^{T_i}/x_i, 1 \le i \le n\right\} = t$ is reducible of type T $(\in \mathsf{RED}^{\mathsf{SN}}(T))$.

By using adaptation lemma once more, one has $t \in \mathsf{RED}^{\mathsf{SN}}(T) \subseteq \mathsf{SNorm}(T)$ which allows to conclude that t is strongly normalizing.

□

2.3 Expressive power of system T

The extended expressiveness of T that was mentioned in the start is expressed by the following theorem :

Theorem 2.16

The functions that can be represented in system T *are the recursive functions which can be proved to be total functions in first-order Peano arithmetics* (PA).

≠ Second-order logic

System F.

2.3 Expressive power of system T

Simple arithmetical functions represented by T-terms.

The successor function can be written as :

$\boxed{\mathsf{Succ} \triangleq \lambda x^{\mathsf{Nat}}.\mathsf{S}(x) \text{ or}}$

$\mathsf{Succ}' \triangleq \lambda x^{\mathsf{Nat}}.\mathsf{Rec}(x, \lambda y^{\mathsf{Nat}}.\lambda z^{\mathsf{Nat}}.z, \mathsf{S}(0))$

Addition can be defined as :

$$\mathsf{Add} \triangleq \lambda x^{\mathsf{Nat}}.\lambda y^{\mathsf{Nat}}.\mathsf{Rec}(x, \lambda z^{\mathsf{Nat}}.\mathsf{Succ}, y).$$

Multiplication can be defined in the same way,

$$\mathsf{Mult} \triangleq \lambda x^{\mathsf{Nat}}.\lambda y^{\mathsf{Nat}}.\mathsf{Rec}(x, \lambda z^{\mathsf{Nat}}.(\mathsf{Add})y, 0).$$

Exponentiation can be defined in the same way,

$$\mathsf{Exp} \triangleq \lambda x^{\mathsf{Nat}}.\lambda y^{\mathsf{Nat}}.\mathsf{Rec}(y, \lambda z^{\mathsf{Nat}}.(\mathsf{Mult})x, \mathsf{S}(0)).$$

$$\mathsf{Pred} \triangleq \lambda x^{\mathsf{Nat}}.\mathsf{Rec}(x, \overbrace{\lambda y^{\mathsf{Nat}}.\lambda z^{\mathsf{Nat}}.y}^{v}, 0).$$

$$\mathsf{Subt} \triangleq \lambda x^{\mathsf{Nat}}.\lambda y^{\mathsf{Nat}}.\mathsf{Rec}(y, \lambda z^{\mathsf{Nat}}.\mathsf{Pred}, x).$$

Pred $S(t)$

$\rightarrow (v)\,t\ \mathrm{Rec}(t, v, 0)$

$\rightarrow t$

$S^k(t).$

$= S(S(\ldots(t))).$

$$\begin{cases} succ = \lambda n.\lambda f \lambda x.(f)\,\underline{(n) f x}. \\ succ' = \lambda n\, \lambda f \lambda x.\,(n) f\,(f) x. \end{cases}$$

$succ\ \overline{n} \rightarrow \lambda f \lambda x.\,(f)(\overline{n}) f x. \rightarrow \rightarrow \lambda f \lambda x.\,\underbrace{(f)(f)\ldots(f)}_{n+1}x$

$\boxed{\lambda f \lambda x.\,(f)(f)\ldots(f)x}$

Ackermann-Peter function in T.

$$A(m,n) \triangleq \begin{cases} n+1 & \text{if } m=0 \\ A(m-1,1) & \text{if } m>0 \text{ and } n=0 \\ A(m-1, A(m, n-1)) & \text{if } m>0 \text{ and } n>0 \end{cases}$$

In order to represent A in T, we would need a T-term A such that

$$\begin{array}{lll} (\mathsf{A})0n & \longrightarrow^{\star}_{\mathsf{T}} & \mathsf{S}(n) \\ (\mathsf{A})\mathsf{S}(m)0 & \longrightarrow^{\star}_{\mathsf{T}} & (\mathsf{A})m\mathsf{S}(0) \\ (\mathsf{A})\mathsf{S}(m)\mathsf{S}(n) & \longrightarrow^{\star}_{\mathsf{T}} & (\mathsf{A})m(\mathsf{A})\mathsf{S}(m)n \end{array}$$

we only have a recursor, not minimization scheme construct. How to find a solution ?

Let us consider A, by currying, not as a function of two arguments but as a family of unary functions $(A_m)_{m\in\mathbb{N}}$ from $\mathbb{N}$ to $\mathbb{N}$. We then notice that the definition becomes :

$$A_0(n) \triangleq n+1$$
$$A_{m+1}(n) \triangleq \begin{cases} A_m(1) & \text{if } n=0 \\ A_m(A_{m+1}(n-1)) & n>0 \end{cases}$$

The effect of A_{m+1} on n is to iterate A_m $n+1$ times over 1 : $A_{m+1}(n) = A_m(A_{m+1}(n-1)) = A_m(A_m(A_{m+1}(n-2))) = A_m(A_m(A_m(A_{m+1}(n-3)))) = \cdots = A_m(A_m(A_m(A_m(\ldots(A_m(1)\ldots)))))!$

$\underline{A_{m+1}(n) = iter(A_m, n)}$ where $\underline{iter(f,0)} = \underline{f(1)}$ and $\underline{iter(f,n+1)} = f(\underline{iter(f,n)})$.

Rec : Nat → ((Nat → U → U) →
(U → U)).

U = Nat.

U = Nat → Nat.

Now, we see clearly how to complete the definition of A :

Consider $\mathsf{Iter} \triangleq \lambda f^{\mathsf{Nat}\to\mathsf{Nat}}.\lambda x^{\mathsf{Nat}}.\mathsf{Rec}(x, \lambda y^{\mathsf{Nat}}.f, (f)\mathsf{S}(0))$ to represent the iteration function described above.

We can define the T-term representing Ackermann-Peter function as

$$\mathsf{A} \triangleq \lambda x^{\mathsf{Nat}}.\mathsf{Rec}(x, \lambda z^{\mathsf{Nat}}.\mathsf{Iter}, \mathsf{Succ}).$$

4 lectures by Alexis.

↳ T, SO, System F.

---

5 lectures by Thomas.

↳ MSO

Alexis is back (3 lectures)

↳ completeness, (many-sorted logics / HO logic).

↳ fixed points μ-calculus

Thomas again on MSO (finite / infinite models)

then corresp. MSO / μ-calculus.