Realizability : a machine for Analysis and set theory

Jean-Louis Krivine

PPS Group, University Paris 7, CNRS

krivine@pps.jussieu.fr

Aussois, june 2007

Introduction

In this tutorial, we will develop a method to obtain programs
from mathematical proofs in Analysis and set theory.
The Curry-Howard or proof-program correspondence began in the 1960's,
for a very weak logical system (intuitionistic propositional logic with $\to$).
Until 1990, everybody believed that this correspondence
worked only for intuitionistic logic.
Then, a key step was made by T. Griffin (a SCHEM'ist) :
the extension to classical propositional logic.

Now, we will explain how to extend this correspondence
to the whole of mathematics.
But why is it worth working hard for this ?

Is this theory interesting or useful ?

In mathematics : the theory of classical realizability is a far reaching extension of Cohen’s forcing. It gives a new class of models of ZF and new intuitions, coming from computer science, to work with them.

In computer science : a methodology for the fundamental problems of software specification and software correctness.

In epistemology : the extension of the proof-program correspondence to *every* mathematical proof, clearly gives a new and deep insight on the nature of mathematics.

But, in this short tutorial, we do not tackle any of these topics.

Introduction (cont.)

We go through the following steps :

First, build the processor (low level machine) which will execute our programs.

Then we have two problems to solve :

1st problem

For each mathematical *proof*, find a *program* which we can run in this machine.

2nd problem (*specification problem)*

Understand the behaviour of these programs

i.e. the *specification* associated with a given *theorem*.

The first problem is now completely solved, but the second is far from being so.

Usual λ-calculus

The *λ-terms* are defined as follows, from a given denumerable set of *λ-variables* :

- Each variable is a λ-term.
- If t is a λ-term and x a variable, then $\lambda x\, t$ is a term (abstraction).
- If t, u are terms, then $(t)u$ is a term (application).

Notations. $((t)u_1)\dots u_n$ is also denoted by $t u_1 \dots u_n$ or $(t)u_1 \dots u_n$.

The substitution is denoted by $t[u_1/x_1, \dots, u_n/x_n]$
(replace, in t, each free occurrence of x_i with u_i).

λ-calculus is very important in computer science, because it is the core of every programming language.

It is a very nice structure, with many properties (Church-Rosser, standardization, ...) which has been deeply investigated.

But, in the following, *nothing else than the definition above* is used about λ-calculus.

A machine in symbolic form

The machine is the program side of the proof-program correspondence.
In these talks, I use only a machine in symbolic form,
not an explicit implementation.
We execute a *process* $t \star \pi$; t is (provisionally) a closed λ-term,
π is a *stack*, that is a sequence $t_1 \bullet t_2 \bullet \ldots \bullet t_n \bullet \pi_0$ where t_i is a closed λ-term
and π_0 is a *stack constant*, i.e. a marker for the bottom of the stack.
We denote by $t \bullet \pi$ the stack obtained by "pushing" t on the top of the stack π.
The set of all stacks will be denoted by Π.

Execution rules for processes (*weak head reduction* of λ-calculus) :
$tu \star \pi \succ t \star u \bullet \pi$ (push)
$\lambda x\, t \star u \bullet \pi \succ t[u/x] \star \pi$ (pop)

This symbolic machine will be used to follow the execution of programs
written in an extension of λ-calculus with new instructions.

A machine in symbolic form (cont.)

We get a better approximation of a “real” machine by eliminating substitution.
The execution rules are a little more complicated (*head linear reduction*) :

$\lambda x_1 \ldots \lambda x_k t u \star t_1 \bullet \ldots \bullet t_k \bullet \pi \succ \lambda x_1 \ldots \lambda x_k t \star t_1 \bullet \ldots \bullet t_k \bullet v \bullet \pi$

with $v = (\lambda x_1 \ldots \lambda x_k u) t_1 \ldots t_k$

(in particular, for $k = 0$, $t u \star \pi \succ t \star u \bullet \pi$)

$\lambda x_1 \ldots \lambda x_k x_i \star t_1 \bullet \ldots \bullet t_k \bullet \pi \succ t_i \star \pi.$

It is necessary to add new instructions, because such simple machines can only handle ordinary λ-terms, i.e. programs obtained from proofs in *pure intuitionistic logic*.
Observe that some of these instructions will be *incompatible with β-reduction.*
Not a problem, because β-reduction plays no real role in the following.

These two methods of execution are essentially equivalent.
For real machine implementation, we use *head linear reduction*
which is much more efficient. But *weak head reduction* is better for easy reading ;
I shall use it during this tutorial, and indicate, from time to time,
the small changes which are necessary for head linear reduction.
Observe that head linear reduction needs the introduction of combinators
or instructions, in order to avoid garbage.
For example, it is better to introduce a new fixpoint instruction Y
with the reduction rule : $\mathsf{Y} \star t \bullet \pi \succ t \star \mathsf{Y} t \bullet \pi$.
The usual Curry fixpoint $Y = \lambda f A_f A_f$ with $A_f = \lambda x(f)(x)x$
would give the following (equivalent, but not very readable) result :
$Y \star t \bullet \pi \succ t \star ((\lambda f\, A_f)t)(\lambda f\, A_f)t \bullet \pi$.
This phenomenon does not arise with weak head reduction.

Examples of λ-terms

Identity : $I = \lambda x\, x$; a program which does nothing.

$\lambda x\, x \star t \centerdot \pi \succ t \star \pi.$

Booleans : true $= \lambda x \lambda y\, x$, false $= \lambda x \lambda y\, y$.

$\lambda x \lambda y\, x \star t \centerdot u \centerdot \pi \succ t \star \pi.$

$\lambda x \lambda y\, y \star t \centerdot u \centerdot \pi \succ u \star \pi.$

Integers : $\lambda f \lambda x (f) \ldots (f) x$ denoted by $\lambda f \lambda x (f)^n x$ or $\underline{n}$ (Church integers).

$\underline{n} \star t \centerdot u \centerdot \pi \succ (t)^n u \star \pi.$

Operations on integers :

Successor : $s = \lambda n \lambda f \lambda x (f)(n) f x$; Sum : $\lambda m \lambda n \lambda f \lambda x (m f)(n) f x$;

Product : $\lambda m \lambda n \lambda f (m)(n) f$; Predecessor : $\lambda n (n \lambda f \lambda x \lambda y ((f)(s) x) x) \underline{0}\,\underline{0}\,\underline{0}$.

Endless loop : $\delta\delta$ with $\delta = \lambda x\, x x$; $\delta\delta \star \pi \succ \delta\delta \star \pi.$

Fixpoint : $\mathsf{Y} = AA$ with $A = \lambda a \lambda f (f)(a) a f$; $\mathsf{Y} t \star \pi \succ t \star \mathsf{Y} t \centerdot \pi.$

Call-by-value on integers

Let v be a program which computes an integer n, for example Sum$\underline{2}\,\underline{3}$.
Then you don't have $v \star t \centerdot u \centerdot \pi \succ (t)^n u \centerdot \pi$. In other words,
the terms Sum$\underline{2}\,\underline{3}$ and $\underline{5}$ do not behave in the same way.
If you need the actual value of v, for instance to execute the loop :
"`for i=1 to n`", then you must compute v first
which is known as "call-by-value" in programming languages.
You need to implement this feature by a λ-term, for instance :
CV $= \lambda f \lambda x \lambda k \lambda n(n \lambda g\, g \circ f)kx$ with $g \circ f = \lambda y(g)(f)y$.

It has the following execution :
CV$\star t \centerdot u \centerdot \kappa \centerdot v \centerdot \pi \succ \kappa \star (t)^n u \centerdot \pi.$ (see page 47).

Second order formulas

We consider second order formulas with $\to$ and $\forall$ as the only logical symbols.

We have individual variables, function symbols on individuals

and predicate variables of any arity.

Thus, an atomic formula is $X t_1 \dots t_k$, where $t_1, \dots, t_k$ are terms.

Notations. $(A_1 \to (A_2 \to \dots \to (A_k \to A)\dots))$ is denoted $A_1, A_2, \dots, A_k \to A$.

Let X be a propositional variable (predicate variable of arity 0).

$\perp$ is defined as $\forall X X$; $\neg A$ is $A \to \perp$;

$A \wedge B$ is $\forall X((A, B \to X) \to X)$; $A \vee B$ is $\forall X((A \to X), (B \to X) \to X)$;

$\exists Y\{A_1, \dots, A_k\}$ means $\forall X[\forall Y(A_1, \dots, A_k \to X) \to X]$.

$x = y$ is defined as $\forall X(Xx \to Xy)$.

Intuitionistic Curry-Howard correspondence

Intuitionistic natural deduction is given by the following usual rules :

$A_1, \dots, A_k \vdash A_i$

$A_1, \dots, A_k, A \vdash B \quad \Rightarrow \quad A_1, \dots, A_k \vdash A \to B$

$A_1, \dots, A_k \vdash A \to B,\ \ A_1, \dots, A_k \vdash A \quad \Rightarrow \quad A_1, \dots, A_k \vdash B$

$A_1, \dots, A_k \vdash A \quad \Rightarrow \quad A_1, \dots, A_k \vdash \forall x\, A$ and $\forall X\, A$

(if x, X are not free in $A_1, \dots, A_k$)

$A_1, \dots, A_k \vdash \forall x\, A \to A[t]$

$A_1, \dots, A_k \vdash \forall X\, A \to A[F/X x_1 \dots x_k]$

(comprehension scheme)

Notice that the axiom $\vdash \bot \to F$ is a particular case of this scheme.

The properties of equality are easily proved using this scheme :

$\forall x \forall y (x = y \to y = x)$; $\forall x \forall y (x = y, F(x) \to F(y))$ for any formula F.

(reflexivity and transitivity are trivial).

Intuitionistic Curry-Howard correspondence (cont.)

These rules become rules for typing λ-terms, as follows :

$x_1{:}A_1, \ldots, x_k{:}A_k \vdash x_i{:}A_i$

$x_1{:}A_1, \ldots, x_k{:}A_k, x{:}A \vdash t{:}B \quad \Rightarrow \quad x_1{:}A_1, \ldots, x_k{:}A_k \vdash \lambda x\, t{:}A \to B$

$x_1{:}A_1, \ldots, x_k{:}A_k \vdash t{:}A \to B,\ \ u{:}A \quad \Rightarrow \quad x_1{:}A_1, \ldots, x_k{:}A_k \vdash tu{:}B$

$x_1{:}A_1, \ldots, x_k{:}A_k \vdash t{:}A \quad \Rightarrow \quad x_1{:}A_1, \ldots, x_k{:}A_k \vdash t{:}\forall x\, A$ and $t{:}\forall X\, A$

(if x, X are not free in $A_1, \ldots, A_k$)

$x_1{:}A_1, \ldots, x_k{:}A_k \vdash \lambda x\, x{:}\forall x\, A \to A[t]$

$x_1{:}A_1, \ldots, x_k{:}A_k \vdash \lambda x\, x{:}\forall X\, A \to A[F/X x_1 \ldots x_k]$

(comprehension scheme)

In this way, we get programs from proofs in *pure* (i.e. without axioms) *intuitionistic logic*. It is the very first step of our work.

Realizability

We know that proofs in pure intuitionistic logic give λ-terms.
But *pure intuitionistic*, or even *classical*, logic is not sufficient
to write down mathematical proofs.
We need *axioms*, such as *extensionality, infinity, choice, …*
Axioms are not theorems, they have no proof !
How can we find suitable programs for them ?

The solution is given by the theory of classical realizability.
We define, for each mathematical formula Φ :

- the set of stacks which *are against* Φ, denoted by $\|\Phi\|$
- the set of closed terms t which *realize* Φ, which is written $t \Vdash \Phi$.

We first choose a set of processes, denoted by $⫫$, which is *saturated*, i.e.

$$t \star \pi \in ⫫,\ \ t' \star \pi' \succ t \star \pi \ \Rightarrow\ t' \star \pi' \in ⫫$$

Realizability (cont.)

Equivalently, we can choose the complement $⫫^c$ of $⫫$, which is *closed by execution*

i.e. $t \star \pi \in ⫫^c,\ \ t \star \pi \succ t' \star \pi' \ \Rightarrow\ t' \star \pi' \in ⫫^c$

The set $\|\Phi\|$ and the property $t \Vdash \Phi$ are defined by induction on the formula Φ.

They are connected as follows :

$$t \Vdash \Phi \Leftrightarrow (\forall \pi \in \|\Phi\|)\, t \star \pi \in ⫫$$

There are three steps of induction, because our logical symbols are
the arrow : $\to$, the first and second order universal quantifiers : $\forall x$, $\forall X$.

1. $\|\Phi \to \Psi\| = \{t \centerdot \pi;\ t \Vdash \Phi, \pi \in \|\Psi\|\}$.

In words : if the term t realizes the formula Φ
and the stack π is against the formula Ψ
then the stack $t \centerdot \pi$ (push t on the top of π) is against the formula $\Phi \to \Psi$.

Realizability (cont.)

2. $\|\forall x\,\Phi(x)\| = \bigcup\{\|\Phi(a)\|;\, a \in \mathbb{N}\}$
This means that the domain of first order variables is $\mathbb{N}$.
In words : a stack is against $\forall x\,\Phi(x)$ if it is against $\Phi(a)$ for some integer a.

3. Let X be a predicate variable of arity k. Then
$\|\forall X\,\Phi(X)\| = \bigcup\{\|\Phi[\mathscr{X}/X]\|;\, \mathscr{X} : \mathbb{N}^k \to \mathscr{P}(\Pi)\}$ where Π is the set of all stacks.
This means that the domain of k-ary predicate variables is $\mathscr{P}(\Pi)^{\mathbb{N}^k}$.

It follows that $t \Vdash \forall x\,\Phi(x) \Leftrightarrow (\forall a \in \mathbb{N})\; t \Vdash \Phi(a)$ and
$t \Vdash \forall X\,\Phi(X) \Leftrightarrow (\forall \mathscr{X} \in \mathscr{P}(\Pi)^{\mathbb{N}^k})\; t \Vdash \Phi[\mathscr{X}/X]$

We have defined $\|\Phi\|$ and $t \Vdash \Phi$ for every closed second order formula Φ *with parameters*. Parameters of arity k are functions $\mathscr{X} : \mathbb{N}^k \to \mathscr{P}(\Pi)$.
A closed atomic formulas is $\mathscr{X}(n_1, \ldots, n_k)$. Its truth value is obvious.

Realizability (cont.)

We see that realizability theory is exactly model theory, in which the truth value set is $\mathscr{P}(\Pi)$ instead of $\{0,1\}$, Π being the set of stacks.
We are indeed considering “ standard ” second order models :
the domain of individuals is $\mathbb{N}$
the domain for k-ary predicate variables is $\mathscr{P}(\Pi)^{\mathbb{N}^k}$ (instead of $\{0,1\}^{\mathbb{N}^k}$).
For each function $f:\mathbb{N}^k \to \mathbb{N}$, we have the k-ary function symbol f
with its natural interpretation.
The truth values $\emptyset$ and Π are denoted by $\top$ and $\bot$. Therefore :
$t \Vdash \top$ for every term t ; $t \Vdash \bot \Rightarrow t \Vdash F$ for every F.
Warning. In our *realizability* models, the set of individuals is $\mathbb{N}$.
But, the 2-*valued* models we shall get from them are *non-standard*, i.e.
they may contain *non-standard integers* and even
individuals which are *not integers at all*.

The adequation lemma

In order to get a model, we have only to choose the saturated set $\perp\!\!\!\perp$.

The case $\perp\!\!\!\perp = \emptyset$ is degenerate : we get back the usual two-valued model theory.

The lemma below is the analog of the *soundness lemma* for our notion of model.

It is an essential tool for the proof-program correspondence.

Adequation lemma.

If $x_1{:}\Phi_1, \dots, x_n{:}\Phi_n \vdash t{:}\Phi$ *and if* $t_i \Vdash \Phi_i \, (1 \le i \le n)$ *then* $t[t_1/x_1, \dots, t_n/x_n] \Vdash \Phi$.

In particular : *If* $\vdash t{:}\Phi$ *then* $t \Vdash \Phi$.

The proof is a simple induction on the length of the derivation of $\dots \vdash t{:}\Phi$.

In the following, we shall more and more use semantic realizability $t \Vdash \Phi$ instead of syntactic typability $\vdash t{:}\Phi$.

The language of mathematics

The proof-program correspondence is well known for *intuitionistic logic*.
Now we have

Mathematics ≡ Classical logic + some axioms

that is Mathematics ≡ Intuitionistic logic + Peirce's law + some axioms

For each axiom $\mathscr{A}$, we choose a closed λ-term which realizes $\mathscr{A}$, *if there is one*.
If not, *we extend our machine* with some *new instruction* which realizes $\mathscr{A}$,
if we can devise such an instruction.
Now, there are essentially two possible axiom systems for mathematics :

1. *Analysis*, i.e. second order classical logic
with the axioms of *recurrence* and *dependent choice*.
2. *ZFC*, i.e. Zermelo-Fraenkel set theory with the full axiom of choice.

Thus, we now have many axioms to deal with.
First of all, we must settle the *law of Peirce* : $((A \to \bot) \to A) \to A$.

Peirce's law

We adapt to our machine the solution found by Tim Griffin in 1990.

We add to the λ-calculus an instruction denoted by cc. Its reduction rule is :

$$\mathsf{cc} \star t \centerdot \pi \succ t \star \mathsf{k}_\pi \centerdot \pi$$

k_π is a *continuation*, i.e. a pointer to a location where the stack π is saved.

In our symbolic machine, it is simply a λ-constant, indexed by π.

Its execution rule is $\mathsf{k}_\pi \star t \centerdot \pi' \succ t \star \pi$.

Therefore cc saves the current stack and k_π restores it.

Using the theory of classical realizability, we show that $\mathsf{cc} \Vdash (\neg A \to A) \to A$.

In this way, we extend the Curry-Howard correspondence to every proof in *pure* (i.e. without axiom) *classical logic* : we now have the new typing rule

$$x_1{:}A_1, \ldots, x_k{:}A_k \vdash \mathsf{cc}{:}(\neg A \to A) \to A$$

Let us check that $\mathsf{cc} \Vdash (\neg A \to A) \to A$:

Peirce's law (cont.)

Take $t \Vdash \neg A \to A$ and $\pi \in \|A\|$. For every $u \Vdash A$, we have $u \star \pi \in \perp\!\!\!\perp$, therefore $\mathsf{k}_\pi \star u \,.\, \pi' \in \perp\!\!\!\perp$ for every stack π'. Thus $\mathsf{k}_\pi \Vdash A \to \perp$ and $\mathsf{k}_\pi \,.\, \pi \in \|\neg A \to A\|$.

It follows that $t \star \mathsf{k}_\pi \,.\, \pi \in \perp\!\!\!\perp$ thus $\mathsf{cc} \star t \,.\, \pi \in \perp\!\!\!\perp$. QED

This extended λ-calculus will be called *λ_c-calculus*.

The set of closed λ_c-terms is denoted by Λ_c.

A closed λ_c-term which contains no continuation is called a *proof-like term*.

We say that *the formula Φ is realized* if there is a proof-like term τ such that $\tau \Vdash \Phi$ *for every choice of* $\perp\!\!\!\perp$. Thus :

- Every λ_c-term which comes from a proof is proof-like.
- If the axioms are realized, every provable formula is realized.

If $\perp\!\!\!\perp \neq \emptyset$, then $\tau \Vdash \perp$ for some λ_c-term τ : take $t \star \pi \in \perp\!\!\!\perp$ and $\tau = \mathsf{k}_\pi t$.

Observe that τ is not a proof-like term.

A useful trick

Let V be *any set of terms*. Then we can define the truth value $\|V \to \Phi\|$ when Φ is a truth value, for example a closed formula with parameters.

The definition is $\|V \to \Phi\| = \{v \,\bullet\, \pi;\ v \in V, \pi \in \|\Phi\|\}$.

For example $\{\xi\} \to \Phi$ and $\neg V$ have truth values.

Theorem. Let Φ be a truth value and V a set of terms.

Then $(V \to \Phi) \leftrightarrow (\neg\Phi \to \neg V)$ is (uniformly) realized :

$\lambda f \lambda k\, k \circ f \Vdash (V \to \Phi) \to (\neg\Phi \to \neg V)$;

$\lambda h \lambda v \mathrm{cc} \lambda k (h) k v \Vdash (\neg\Phi \to \neg V) \to (V \to \Phi)$.

Theorem. Let X be a truth value and $X^- = \{\mathrm{k}_\pi;\ \pi \in \|X\|\}$.

Then $\neg X^- \leftrightarrow X$ is (uniformly) realized.

Indeed $\lambda x \lambda y\, y x \Vdash X \to \neg X^-$ and $\mathrm{cc} \Vdash \neg X^- \to X$.

In fact, the very definition of cc gives $\mathrm{cc} \Vdash (X^- \to X) \to X$.

A useful trick (cont.)

It follows that, in order to realize $X \vee A$ or $\neg X \to A$, it is sufficient (but often much easier) to realize $X^- \to A$. Indeed, we have :

Theorem. If $\xi \star \mathsf{k}_\pi . \rho \in ⫫$ for all $\pi \in \|X\|$ and $\rho \in \|A\|$,
then $\lambda y\, \mathsf{cc} \lambda k(y)(\xi)k \Vdash (A \to X) \to X$.
In other words, $\lambda x \lambda y\, \mathsf{cc} \lambda k(y)(x)k \Vdash (X^- \to A) \to ((A \to X) \to X)$.

Let $\eta \Vdash A \to X$ and $\pi \in \|X\|$. We must show that $\lambda y\, \mathsf{cc} \lambda k(y)(\xi)k \star \eta . \pi \in ⫫$.
Now this process reduces to $\eta \star \xi \mathsf{k}_\pi . \pi$. Thus, it is sufficient to show $\xi \mathsf{k}_\pi \Vdash A$.
But, if $\rho \in \|A\|$, we have $\xi \mathsf{k}_\pi \star \rho \succ \xi \star \mathsf{k}_\pi . \rho$ which is in $⫫$ by hypothesis. QED

First simple theorems

The choice of $\perp\!\!\!\perp$ is generally done according to the theorem Φ for which we want to solve the specification problem. Let us take two simple examples.

Theorem. If θ comes from a proof of $\forall X(X \to X)$ (with any realized axioms) then $\theta \star t\,.\,\pi \succ t \star \pi$ i.e. θ behaves like $\lambda x\, x$.

Proof. Take $\perp\!\!\!\perp = \{\mathsf{p}\,;\, \mathsf{p} \succ t \star \pi\}$ and $\|X\| = \{\pi\}$.

Thus $t \Vdash X$ and $\theta \star t\,.\,\pi \in \perp\!\!\!\perp$. QED

Example : $\theta = \lambda x\, \mathrm{cc}\lambda k\, kx$.

Dual proof. Take $\perp\!\!\!\perp^c = \{\mathsf{p}\,;\, \theta \star t\,.\,\pi \succ \mathsf{p}\}$ and $\|X\| = \{\pi\}$.

Thus, $\theta \star t\,.\,\pi \in \perp\!\!\!\perp^c$; since $\pi \in \|X\|$ and $\theta \Vdash X \to X$, we have $t \not\Vdash X$ and therefore $t \star \pi \in \perp\!\!\!\perp^c$. QED

First simple theorems (cont.)

The formula $\mathrm{Bool}(x) \equiv \forall X(X1, X0 \to Xx)$ is equivalent to $x=1 \vee x=0$.

Theorem. If θ comes from a proof of $\mathrm{Bool}(1)$, then $\theta \star t \centerdot u \centerdot \pi \succ t \star \pi$

i.e. θ behaves like the boolean $\lambda x \lambda y\, x$.

Proof. Take $\perp\!\!\!\perp = \{\mathsf{p}\ ;\ \mathsf{p} \succ t \star \pi\}$, $\|X1\| = \{\pi\}$ and $\|X0\| = \emptyset = \|\top\|$.

Thus $t \Vdash X1$, $u \Vdash X0$ and $\theta \star t \centerdot u \centerdot \pi \in \perp\!\!\!\perp$. QED

Dual proof. Take $\perp\!\!\!\perp^c = \{\mathsf{p}\ ;\ \theta \star t \centerdot u \centerdot \pi \succ \mathsf{p}\}$, $\|X1\| = \{\pi\}$ and $\|X0\| = \emptyset = \|\top\|$. We have $u \Vdash X0$, $\pi \in \|X1\|$, $\theta \Vdash X1, X0 \to X1$ and $\theta \star t \centerdot u \centerdot \pi \in \perp\!\!\!\perp^c$.

Thus $t \not\Vdash X1$ and $t \star \pi \in \perp\!\!\!\perp^c$. QED

Another example : $\exists x(Px \to \forall y\, Py)$

Write this theorem $\forall x[(Px \to \forall y\, Py) \to \bot] \to \bot$. We must show :
$z{:}\forall x[(Px \to \forall y\, Py) \to \bot] \vdash ?{:}\bot$. We get $z{:}(Px \to \forall y\, Py) \to \bot$,
$z{:}(Px \to \forall y\, Py) \to Px$, $\mathsf{cc}\, z{:}Px$, $\mathsf{cc}\, z{:}\forall x\, Px$, $\lambda d\, \mathsf{cc}\, z{:}Px \to \forall y\, Py$
and $z\lambda d\, \mathsf{cc}\, z{:}\bot$. Finally we have obtained the program $\theta = \lambda z\, z\lambda d\, \mathsf{cc}\, z$.
Let us find a characteristic feature in the behaviour of *all terms* θ
such that $\vdash \theta{:}\exists x(Px \to \forall y\, Py)$. Let $\alpha_0, \alpha_1, \ldots$ and $\varpi_0, \varpi_1, \ldots$
be a fixed sequence of terms and of stacks. We define a *new instruction* κ ;
its reduction rule uses two players named $\exists$ and $\forall$ and is as follows :

$$\kappa \star \xi . \pi \succ \xi \star \alpha_i . \varpi_j$$

where i is first chosen by $\exists$, then j by $\forall$.
The player $\exists$ wins iff the execution arrives at $\alpha_i \star \varpi_i$ for some $i \in \mathbb{N}$.

$\exists x(Px \to \forall y\, Py)$ (cont.)

Theorem. If $\vdash \theta{:}\forall x[(Px \to \forall y\, Py) \to \bot] \to \bot$, there is a winning strategy for $\exists$ when we execute the process $\theta \star \kappa \,.\, \pi$ (for any stack π).

Proof. Let $⫫$ be the set of processes for which there is a winning strategy for $\exists$. Define a realizability model on $\mathbb{N}$, by setting $\|Pn\| = \{\varpi_n\}$. Thus $\alpha_n \Vdash Pn$.

Suppose that $\xi \Vdash Pi \to \forall y\, Py$ for some $i \in \mathbb{N}$. Then :

$\xi \star \alpha_i \,.\, \varpi_j \in ⫫$ for every j and it follows that $\kappa \star \xi \,.\, \pi \in ⫫$, for any stack π : indeed, a strategy for $\exists$ is to play i and to continue with a strategy for $\xi \star \alpha_i \,.\, \varpi_j$ if $\forall$ plays j. It follows that $\kappa \Vdash (Pi \to \forall y\, Py) \to \bot$ and therefore :

$\kappa \Vdash \forall x[(Px \to \forall y\, Py) \to \bot]$. Thus, $\theta \star \kappa \,.\, \pi \in ⫫$ for every stack π. QED

$\exists x(Px \to \forall y\, Py)$ (cont.)

Dual proof. Suppose there is no winning strategy for $\exists$, starting from $\theta \star \kappa \bullet \pi$.
Then there is one for $\forall$: to play so that $\exists$ never has a winning strategy.
Suppose that $\forall$ has chosen such a strategy and define $\perp\!\!\!\perp^c$
to be the set of processes we can reach from $\theta \star \kappa \bullet \pi$.
Set, as before, $\|Pn\| = \{\varpi_n\}$. Then $\alpha_n \Vdash Pn$ because $\alpha_n \star \varpi_n$ is not reached.
Then $\kappa \nVdash \forall x[(Px \to \forall y\, Py) \to \perp]$ because $\theta \star \kappa \bullet \pi \notin \perp\!\!\!\perp$.
Thus, there is an $i \in \mathbb{N}$ and a $\xi \Vdash Pi \to \forall y\, Py$ s.t. $\kappa \star \xi \bullet \pi' \in \perp\!\!\!\perp^c$.
At this moment, $\exists$ can play α_i and $\forall$ will play ϖ_j by his strategy.
Thus $\xi \star \alpha_i \bullet \varpi_j \in \perp\!\!\!\perp^c$ because this process is reached.
This contradicts the property of ξ because $\alpha_i \Vdash Pi$ and $\varpi_j \in \|\forall y\, Py\|$. QED

$\exists x(Px \to \forall y\, Py)$ (cont.)

For instance, if $\theta = \lambda z\, z \lambda d\, \mathsf{cc}\, z$, we have :

$\theta \star \kappa \centerdot \pi \succ \kappa \star \lambda d\, \mathsf{cc} \kappa \centerdot \pi \succ \lambda d\, \mathsf{cc} \kappa \star \alpha_{i_0} \centerdot \varpi_{j_0}$ if $\exists$ plays i_0 and $\forall$ plays j_0.

We get $\mathsf{cc} \star \kappa \centerdot \varpi_{j_0} \succ \kappa \star \mathsf{k}_{\varpi_{j_0}} \centerdot \varpi_{j_0}$. A winning strategy for $\exists$ is now to play j_0 :

if $\forall$ plays j_1, this gives $\mathsf{k}_{\varpi_{j_0}} \star \alpha_{j_0} \centerdot \varpi_{j_1} \succ \alpha_{j_0} \star \varpi_{j_0}$.

Remark. The program θ does not gives explicitly a winning strategy.

Programs associated with proofs of *arithmetical theorems* will give such strategies,

i.e. will play in place of $\exists$.

We shall return to this topic later and consider the general case :

true first order formulas.

Axioms for mathematics

Let us now consider the usual axiomatic theories which formalize mathematics.

• Analysis is written in second order logic. There are three groups of axioms :

1. Equations such as $\forall x(x+0=x)$, $\forall x \forall y(x+sy=s(x+y))$, $\ldots$

and inequations such as $s0 \neq 0$.

2. The recurrence axiom $\forall x \,\mathrm{int}(x)$, which says that each individual (1st order object) is an integer. The formula $\mathrm{int}(x)$ is : $\forall X\{\forall y(Xy \to Xsy), X0 \to Xx\}$.

3. The axiom of dependent choice :

If $\forall X \exists Y\, F(X,Y)$, then there exists a sequence X_n such that $F(X_n, X_{n+1})$.

Analysis is sufficient to formalize a very important part of mathematics including the theory of functions of real or complex variables, measure and probability theory, partial differential equations, analytic number theory, Fourier analysis, etc.

Axioms for mathematics (cont.)

- Axioms of ZFC can be classified in three groups :

1. Equality, extensionality, foundation.
2. Union, power set, substitution, infinity.
3. Choice : Any product of non void sets is non void ;
   possibly other axioms such as CH, GCH, large cardinals, ...

In order to realize axioms 1 and 2 (i.e. ZF), we must interpret ZF
in a conservative extension called ZF_ε which is much simpler to realize.
The λ_c-terms for ZF are rather complicated, but do not use new instructions.
I found rather recently the solution for AC and CH.
New instructions are needed, again very well known in programming languages :
read and write on global variables (”side effects”).
We get, for AC and CH, very complicated programs (λ-terms)
written with these instructions.

Realizability models of analysis

For the moment, we consider realizability models of 2nd order logic.
For these models, the domain of individuals is $\mathbb{N}$
and the domain of k-ary predicate variables is $\mathscr{P}(\Pi)^{\mathbb{N}^n}$.
The only left free choice is $\perp\!\!\!\perp$.
But it is important to remember that these domains are used only
for computing the truth values of formulas : $\|\forall x\,\Phi(x)\| = \bigcup_{n\in\mathbb{N}} \|\Phi(n)\|$.
For example, it does not mean that the formula : “ every individual is an integer ”
that is the recurrence axiom $\forall x \forall X[\forall y(Xy \to Xsy), X0 \to Xx]$ is realized.
Indeed, for the most usual choices of $\perp\!\!\!\perp$, the *negation* of this formula is realized.
In order to grasp this strange situation, we absolutely need *ordinary 2-valued models.*
We now explain how to get them.

Coherence

In fact, the situation is even worse, because there are
some useful examples of $\perp\!\!\!\perp$ for which $\perp$ *is realized*. For instance :
$\perp\!\!\!\perp$ = the set of processes the execution of which is infinite ; we have $\delta\delta \Vdash \perp$.

Now, by adequation lemma, the set of realized formulas is closed by
classical deduction. If this set is consistent, we say that $\perp\!\!\!\perp$ is *coherent*.
This means that there is no proof-like term θ such that $\theta \Vdash \perp$.
In other words, for every proof-like term θ, there is a stack π such that $\theta \star \pi \notin \perp\!\!\!\perp$.
From now on, we consider only the case when $\perp\!\!\!\perp$ is coherent.
Examples. Let p_0 be some given process ; then
$\perp\!\!\!\perp = \{\mathsf{p};\, \mathsf{p} \succ \mathsf{p}_0\}$ is coherent if there is at least 2 stack constants ;
$\perp\!\!\!\perp = \{\mathsf{p};\, \mathsf{p}_0 \not\succ \mathsf{p}\}$ is not coherent : consider the proof-like terms $\delta\delta 0$ and $\delta\delta 1$.

2-valued realizability models

Let $\perp\!\!\!\perp$ be a coherent saturated set of processes. Then the set of realized closed formulas is closed under derivation in classical logic and does not contain $\perp$.
It is therefore consistent and we obtain, in this way, 2-valued models
of second order logic or of set theory.
We shall see that these models are very different from the model we started with.
As told before, there exist individuals which are not integers ;
but there are also non-standard integers in the following strong sense :
there is a unary predicate P such that the formulas
$\exists x[\text{int}(x) \land Px]$ and $\neg Pn$ are realized for each integer n.

The Boolean algebra $\mathscr{P}(\Pi)$

Every coherent $\perp\!\!\!\perp$ gives a *Boolean structure* on the set $\mathscr{P}(\Pi)$ of truth values :
for $\mathscr{X}, \mathscr{Y} \subset \Pi$, define :

$$\mathscr{X} \le \mathscr{Y} \Leftrightarrow \text{there is a proof-like term } \theta \text{ s.t. } \theta \Vdash \mathscr{X} \to \mathscr{Y}$$

It is easy to prove that this is a Boolean preorder on $\mathscr{P}(\Pi)$, with $\mathscr{X}^c = \|\neg\mathscr{X}\|$ and
$\inf(\mathscr{X}, \mathscr{Y}) = \|\mathscr{X} \wedge \mathscr{Y}\| = \|\forall X((\mathscr{X}, \mathscr{Y} \to X) \to X)\|$ or $\|(\mathscr{X}, \mathscr{Y} \to \perp) \to \perp\|$,
$\sup(\mathscr{X}, \mathscr{Y}) = \|\mathscr{X} \vee \mathscr{Y}\| = \|\forall X((\mathscr{X} \to X), (\mathscr{Y} \to X) \to X)\|$
or $\|(\mathscr{X} \to \perp), (\mathscr{Y} \to \perp) \to \perp\|$.
Let $\mathscr{B} = \mathscr{P}(\Pi)/\simeq$ be this Boolean algebra.
Every closed formula has a value in $\mathscr{P}(\Pi)$ and therefore a value in $\mathscr{B}$.
We get, in this way, *Boolean models* of second order logic or set theory.
Using any ultrafilter on $\mathscr{B}$, we obtain again the 2-valued realizability models described in the last slide.

Remarks on 2-valued models

We use the following terminology : the *standard model* of analysis is $(\mathbb{N}, 2^{\mathbb{N}})$.
Given $\perp\!\!\!\perp$, we have the *realizability model* associated with $\perp\!\!\!\perp$, which is $(\mathbb{N}, \mathscr{P}(\Pi)^{\mathbb{N}})$ with the definition of truth value of closed 2nd order formulas.
Then, we have the *2-valued realizability models*, we have just defined.
For any closed second order formula F the following conditions are equivalent :
• $\mathscr{M} \models F$ for every 2-valued model $\mathscr{M}$ associated with $\perp\!\!\!\perp$
• there exists a proof-like term θ s.t. $\theta \Vdash F$

Notice that every predicate and every function on individuals which is defined in the standard model is also defined in the 2-valued realizability models (because we can put them in the language). But, in these models, there are many individuals and predicates which are not named in the language. For example, non-standard integers or non integers.

Axioms of analysis : equations

Axioms : $\neg(0 = s0)$; $p0 = 0$; $\forall x(psx = x)$; $\forall x(x + 0 = x)$; $\forall x(x.0 = 0)$;
$\forall x \forall y(x + sy = s(x + y))$; $\forall x \forall y(x.sy = xy + x)$

Such equations and inequations are very easy to realize.

Theorem. Any true equation is realized by $\lambda x\, x$.

Any true inequation is realized by $\lambda x\, xt$ for an arbitrary t.

Proof. $x = y$ is defined by $\forall X(Xx \to Xy)$ in second order logic. QED

Very useful definition. Define a new predicate $x \neq y$ by setting :
$\|n \neq p\| = \emptyset = \|\top\|$ if $n \neq p$ and $\|n \neq p\| = \Pi = \|\bot\|$ if $n = p$.

Theorem. $\lambda x \lambda y\, yx \Vdash \forall x \forall y[x \neq y \to \neg(x = y)]$ and
$\lambda x\, xt \Vdash \forall x \forall y[\neg(x = y) \to x \neq y]$ for any t.

This means we can use the predicate $x \neq y$ in place of $\neg(x = y)$.

Another important Boolean algebra

The predicate $x^2 = x$ defines a set $\mathscr{B}$ of individuals, which is a *Boolean algebra*.

For example, $\forall x \forall y[x^2 = x, y^2 = y \to (x + y - xy)^2 = x + y - xy]$ is a consequence of true equations (associativity, commutativity and distributivity).

Another way : realize $\forall x \forall y[x^2 = x, (x + y - xy)^2 \neq x + y - xy \to y^2 \neq y]$, i.e.

$\forall x(x^2 = x, 1 \neq 1 \to \bot) \cap \forall x(x^2 = x, x^2 \neq x \to \bot)$ i.e.

$\forall x(1 \neq 1 \to x^2 \neq x) \cap \forall x(x^2 \neq x \to x^2 \neq x)$ realized by $\lambda x\, x$.

Lemma. Every element $\neq 1$ of $\mathscr{B}$ is not a successor.

Indeed, $(x+1)^2 = x+1$ gives $x^2 + x = 0$ thus $x = 0$. QED

In most interesting models, the algebra $\mathscr{B}$ is not trivial, i.e. $\mathscr{B} \neq \{0, 1\}$.

This shows that there are individuals which are not integers.

Let us give an example.

A non trivial Boolean algebra $\mathscr{B}$

Set $⫫ = \{\mathsf{p} \in \Lambda \star \Pi;\ \mathsf{p} \succ I \star \pi_0\}$; I is $\lambda x\, x$, π_0 is a fixed stack constant.

Lemma. $|\top, \bot \to \bot| \cap |\bot, \top \to \bot| = |\top, \top \to \bot|$.

It is clearly sufficient to prove $\subset$. Let $t \in |\top, \bot \to \bot| \cap |\bot, \top \to \bot|$, $\pi \in \Pi$,

$\kappa = \mathsf{k}_{\pi_0} I \Vdash \bot$, $\omega = \delta\delta$ and a, b be two fresh constants.

Suppose that $t \star a . b . \pi \succ a \star \pi'$; then $t \star \omega . \kappa . \pi \succ \omega \star \pi''$,

which contradicts $t \Vdash \top, \bot \to \bot$. Therefore, during the execution of $t \star a . b . \pi$

neither a nor b comes in head position. Since $t \star u . \kappa . \pi \succ I \star \pi_0$

it follows that $t \star u . v . \pi \succ I \star \pi_0$. This shows that $t \Vdash \top, \top \to \bot$. QED

Now, $|\forall x(x \neq 1, x \neq 0 \to x^2 \neq x)| = |\top, \bot \to \bot| \cap |\bot, \top \to \bot|$;

this shows that $\lambda x\, x00 \Vdash \neg \forall x(x \neq 1, x \neq 0 \to x^2 \neq x)$.

This formula means that $\mathscr{B}$ is a non trivial Boolean algebra.

An atomless Boolean algebra $\mathscr{B}$

An atom of $\mathscr{B}$ is a minimal element of $\mathscr{B} \setminus \{0\}$. We show that, in the above model, the algebra $\mathscr{B}$ has no atom ; thus, it is not only non trivial, but even infinite.
The fact that $\mathscr{B}$ is atomless is expressed by the formula :
$\forall x(x^2 = x, x \neq 0 \to \exists y(y^2 = y \wedge xy \neq 0 \wedge xy \neq x)$ i.e.
$\forall x[\forall y(xy \neq 0, xy \neq x \to y^2 \neq y), x \neq 0 \to x^2 \neq x]$.
The truth value of this formula is :
$|\forall y(y \neq 0, y \neq 1 \to y^2 \neq y), \top \to \bot| \cap |\forall y(0 \neq 0, 0 \neq 0 \to \bot), \bot \to \bot|$.
We have just seen that $|\forall y(y \neq 0, y \neq 1 \to y^2 \neq y)| = |\top, \top \to \bot|$.
Thus, we get $|(\top, \top \to \bot), \top \to \bot| \cap |(\bot, \bot \to \bot), \bot \to \bot|$
which is realized by $\lambda x \lambda y\, xyy$.

Exercise on this model

We have shown that any element of $\mathscr{B}\setminus\{1\}$ has no predecessor (in every model). But, in this model, the converse is false, i.e.
there are individuals without predecessor that are not in $\mathscr{B}$.
We show that the formula $\exists x[x^2 \neq x \wedge \forall y(x \neq sy)]$ that is
$\forall x[x^2 \neq x, \forall y(x \neq sy) \to \perp] \to \perp$ is realized. We have
$|\forall y(n \neq sy)| = \top$ if $n = 0$ and $\perp$ if $n \neq 0$. Therefore
$|\forall x[x^2 \neq x, \forall y(x \neq sy) \to \perp]| = \bigcap_n |n^2 \neq n, \forall y(n \neq sy) \to \perp|$
$= |\perp, \top \to \perp| \cap |\perp, \perp \to \perp| \cap |\top, \perp \to \perp| = |\top, \top \to \perp|$ by the lemma above.
Thus $|\exists x[x^2 \neq x \wedge \forall y(x \neq sy)]| = |(\top, \top \to \perp) \to \perp|$
and this formula is realized by $\lambda x\, x00$.

We have now many examples of *non integers*.
We have not yet given an example of a *non-standard integer*.
A much more difficult problem is : does there exist an *ultrafilter on $\mathscr{B}$* ?

Intersection of types

Let $F(x)$ be any second order formula. It is interesting to compare the truth values $|F(1) \wedge F(0)|$ and $|F(1)| \cap |F(0)|$. We show that $|F(1)| \cap |F(0)|$ is equivalent to the formula $\forall x [x^2 = x \to F(x)]$. This means that :

i) $\forall x [x^2 = x \to F(x)] \to |F(1)| \cap |F(0)|$ and

ii) $|F(1)| \cap |F(0)| \to \forall x [x^2 = x \to F(x)]$ are both realized.

(i) is realized by $\lambda x\, x I$ (put $x = 1, 0$ in $x^2 = x \to F(x)$).

Now $\forall x [x^2 = x \to F(x)]$ is equivalent to $\forall x [\neg F(x) \to x^2 \neq x]$

the value of which is $|\neg\neg F(1)| \cap |\neg\neg F(0)|$. But we have

$\lambda x\, x I \Vdash |F(1)| \cap |F(0)| \to |\neg\neg F(1)| \cap |\neg\neg F(0)|$.

We have found the meaning of $F(1) \cap F(0)$

which is clearly stronger than $F(1) \wedge F(0)$.

Axioms of analysis : recurrence

The proper recurrence axiom is $\forall x\,\mathrm{int}(x)$, where $\mathrm{int}(x)$ is the formula :

$$\forall X[\forall y(Xy \to Xsy), X0 \to Xx]$$

This axiom *cannot be realized*, even by means of new instructions ;
thus, in realizability models, there are individuals which are not integers.

There are two solutions, which are logically equivalent for integers ;
but they correspond to very different programming styles.
The first method is to *discard* the recurrence axiom
and restrict first order quantifiers to the formula $\mathrm{int}(x)$.
The second method is the same we shall use to realize axioms of ZF.
We define a new equality $\simeq$ on individuals, which allows to realize
the recurrence axiom : every individual becomes *equivalent* to an integer.
It uses a fixpoint combinator and the programming style is LISP's.

Recurrence axiom, 1st method

The language has a function symbol for each recursive function.

Let $\text{int}(x) \equiv \forall X[\forall y(Xy \to Xsy), X0 \to Xx]$.

Theorem. If a second order formula Φ is provable with the recurrence axiom, then the restricted formula Φ^{int} is provable without it, using the axioms $\forall x_1 \ldots \forall x_k\{\text{int}(x_1),\ldots,\text{int}(x_k) \to \text{int}(f(x_1,\ldots,x_k))\}$ for each symbol f.

Now, we only need to realize these new axioms. There are two ways of doing this :

• Prove this formula from *true equations*.

Examples. The successor s : $\text{int}(x) \to \text{int}(sx)$ is provable with no equation.

Addition : $\text{int}(x), \text{int}(y) \to \text{int}(x+y)$ is provable with the equations :

$x+0=x;\ x+sy=s(x+y), \ldots$

This works for a very large class of recursive functions :

the provably total functions in second order arithmetic.

Recurrence axiom (cont.)

• The second method works for *every recursive function* f.

Assume, for simplicity, that f is unary. We have two lemmas.

Lemma. If τ is a closed λ-term, $\tau \simeq_\beta \underline{n}$ (Church integer), then $\tau \Vdash \text{int}(s^n 0)$.

Define $T = \lambda f \lambda n(n) \lambda g\, g \circ s \,\bullet f \bullet 0$ (*storage operator* [5]).

Storage lemma. *If* $(\forall \pi \in \|X\|)\, \kappa \star s^n 0 \bullet \pi \in \perp\!\!\!\perp$ *then* $T\kappa \Vdash$ *int*$(n) \to X$.

Proof. Let $\|Pj\| = \{s^{n-j} 0 \bullet \pi;\ \pi \in \|X\|\}$ for $0 \le j \le n$;

$\|Pj\| = \emptyset$ for $j > n$. Then $\lambda g\, g \circ s \Vdash \forall x(Px \to Psx)$ and $\kappa \Vdash P0$.

Thus, if $\nu \Vdash$ int(n) then $\nu \star \lambda g\, g \circ s \bullet \kappa \bullet \pi \in \perp\!\!\!\perp$ which gives $T\kappa \star \nu \bullet \pi \in \perp\!\!\!\perp$. QED

We can state this result as follows : $T \Vdash \forall X \forall n\{(\{s^n 0\} \to X) \to (\text{int}(n) \to X)\}$

i.e. the formula int(n) may be replaced with $\{s^n 0\}$ when computing truth values, at the cost of introducing the program T.

Recurrence axiom (cont.)

Finally, we realize the axiom we need :

Theorem. Let τ be a closed λ-term which computes the recursive function f.
Then $T\lambda x\tau x \Vdash \forall x[\mathrm{int}(x) \to \mathrm{int}(f(x))]$.

By the storage lemma, we only need to prove that $\lambda x\tau x \star s^n 0 . \pi \in \perp\!\!\!\perp$
for $\pi \in \|\mathrm{int}(f(n))\|$. But this follows from the first lemma,
since $\tau s^n 0 \simeq_\beta \underline{r}$ with $r = f(n)$. QED

In the storage lemma, we may be replace the λ-terms $s, \underline{0}$ with arbitrary terms t, u.
The proof is not changed.

Generalized storage lemma. *If $(\forall \pi \in \|X\|)\kappa \star t^n u . \pi \in \perp\!\!\!\perp$ then $\mathsf{CV}tu\kappa \Vdash \mathrm{int}(n) \to X$ with $\mathsf{CV} = \lambda f\lambda x\lambda k\lambda n(n\lambda g\, g\circ f)kx$ and $g\circ f = \lambda y(g)(f)y$.*

Imperative call-by-value

Let $\nu \in \Lambda_c$ such that $\vdash \nu{:}\mathrm{int}(s^n 0)$; i.e. ν "behaves like" the integer n.

In the λ_c-term $\kappa\nu$ this data is *called by name* by the program κ.

In the λ_c-term $T\kappa\nu$ the same data is *called by value* by κ,

which means it is computed first (in the form $s^n 0$).

Theorem. If $\vdash \nu{:}\mathrm{int}(s^n 0)$, then $T\kappa \star \nu \centerdot \pi \succ \kappa \star s^n 0 \centerdot \pi$

and $\mathrm{CV}\star t \centerdot u \centerdot \kappa \centerdot \nu \centerdot \pi \succ \kappa \star t^n u \centerdot \pi$.

Let $⫫ = \{\mathrm{p};\ \mathrm{p} \succ \kappa \star s^n 0 \centerdot \pi\}$. Then $T\kappa \star \nu \centerdot \pi \in ⫫$, by the storage lemma. QED

I name this behaviour *imperative* call-by-value, to avoid confusion with the notion of call-by-value used in functional languages (LISP, ML), and because it is very similar to the usual notion of call-by-value in imperative languages.

It is only defined for data types (booleans, integers, trees, ...)

Computing recursive functions

So, we can discard the recurrence axiom and replace it with the formulas :
$\forall x_1 \ldots \forall x_k \{\mathrm{int}(x_1), \ldots, \mathrm{int}(x_k) \to \mathrm{int}(f(x_1, \ldots, x_k))\}$ for each symbol f.
These formulas make sense, because there exist *individuals which are not integers*.
Theorem. If $\vdash \phi : \forall \vec{x}\{\overrightarrow{\mathrm{int}}(\vec{x}) \to \mathrm{int}(f\vec{x})\}$, then ϕ computes the function f, i.e. :
if $\underline{\vec{n}}$ is a sequence of Church integers, then $T\kappa \star \phi \underline{\vec{n}} . \pi \succ \kappa \star s^p 0 . \pi$ with $p = f(\vec{n})$.
This works for every data type : Booleans, integers, sums, products and lists
of data types, etc. Here, we only use the types of integers and of Booleans.
$\mathrm{Bool}(x) \equiv \forall X(X1, X0 \to Xx)$. For this type we have :
Theorem. If $\vdash \phi : \forall x\{\mathrm{int}(x) \to \mathrm{Bool}(f(x))\}$, then
$\phi \star \hat{n} . t . u . \pi \succ t \star \pi$ if $f(n) = 1$; $\phi \star \hat{n} . t . u . \pi \succ u \star \pi$ if $f(n) = 0$
where $\hat{n}$ is any closed λ-term β-equivalent to the Church integer n.

Remarks on head linear reduction

If we use the *head linear reduction machine*, the storage lemma is no longer true : the storage operator $T = \lambda f \lambda n(n)\lambda g\, g \circ s \,.\, f \,.\, 0$ introduces garbage.
We define a storage instruction $\mathbf{T}$ and an auxiliary instruction $\mathbf{U}$
with the following execution rules :
$\mathbf{T} \star \phi . \nu . \pi \succ \nu \star \mathbf{U} . \phi . 0 . \pi$ and $\mathbf{U} \star g . \xi . \pi \succ g \star s\xi . \pi$.
Storage lemma. *If* $(\forall \pi \in \|X\|)\, \phi \star s^n 0 . \pi \in \perp\!\!\!\perp$ *then* $\mathbf{T}\phi \Vdash$ *int*$(n) \to X$.
Proof. Let $\|Pj\| = \{s^{n-j} 0 . \pi;\ \pi \in \|X\|\}$ for $0 \le j \le n$;
$\|Pj\| = \emptyset$ for $j > n$. Then $\mathbf{U} \Vdash \forall x(Px \to Psx)$ and $\phi \Vdash P0$.
Thus, if $\nu \Vdash$ int(n) then $\nu \star \mathbf{U} . \phi . \pi \in \perp\!\!\!\perp$ which gives $\mathbf{T}\phi \star \nu . \pi \in \perp\!\!\!\perp$. QED

Recurrence axiom, 2nd method

Theorem. $\mathsf{Y} \Vdash \forall x[\forall y(Xy \to sy \neq x) \to \neg Xx] \to \forall x \neg Xx$
where $\mathsf{Y} = AA$ with $A = \lambda a \lambda f(f)(a)af$ is the Turing fixpoint combinator.
Its execution rule is $\mathsf{Y} \star t . \pi \succ t \star \mathsf{Y}t . \pi$.

Remark. In *head linear reduction*, Y must be an instruction with this reduction rule.

Now, this formula says that the relation $sy = x$ is well founded.
From this, it is easy to *prove* that every individual x can be uniquely written as $x = x_0 + n$, where n is an integer and x_0 has no predecessor.
We have defined an equivalence relation on individuals and we consider integers as equivalence classes. The class 0 is the set of individuals without predecessor.
The recurrence axiom $\forall X \forall x[\forall y(Xy \to Xsy), X0 \to Xx]$ which we cannot realize, is replaced with : $\forall X \forall x[\forall y(Xy \to Xsy), \forall y(y \simeq 0 \to Xy) \to Xx]$
which is provable from the well foundedness of $sy = x$.

Fixpoint and well foundedness

We prove more generally :

Theorem. Let $x \sqsubset y$ be well founded on integers and $\phi(x,y)$ its characteristic function. Then $\mathsf{Y} \Vdash \forall X\{\forall x[\forall y(Xy \to \phi(y,x) \neq 1) \to \neg Xx] \to \forall x \neg Xx\}$.

Proof. Let $t \Vdash \forall x[\forall y(\mathscr{X}(y) \to \phi(y,x) \neq 1) \to \neg\mathscr{X}(x)]$
for some $\mathscr{X} : \mathbb{N} \to \mathscr{P}(\Pi)$. We prove $\mathsf{Y}t \Vdash \neg\mathscr{X}(n)$ by induction on n,
following $\sqsubset$. Let $u \Vdash \mathscr{X}(n)$, we must prove $\mathsf{Y} \star tu\pi \in \perp\!\!\!\perp$, i.e. $t \star \mathsf{Y}t.u.\pi \in \perp\!\!\!\perp$.
It is sufficient to prove $\mathsf{Y}t \Vdash \forall y(\mathscr{X}(y) \to \phi(y,n) \neq 1)$.
Now, if $y \sqsubset n$, this is true because $\mathsf{Y}t \Vdash \mathscr{X}(y) \to \perp$, by induction hypothesis ;
else this is also true because $\|\phi(y,n) \neq 1\| = \emptyset$. Q.E.D.

Non standard integers (1st example)

Let a_n, π_n be given sequences of λ-constants (instructions) and stack constants.

Define a realizability model by setting $\perp\!\!\!\perp = \{\mathsf{p} \in \Lambda \star \Pi;\ \exists n(\mathsf{p} \succ a_n \star \pi_n)\}$.

In this model, define a unary predicate P by $\|Pn\| = \{\pi_n\}$.

Since $a_n \Vdash Pn$, every 2-valued realizability model satisfies Pn for every $n \in \mathbb{N}$.

We show that there are such models with non-standard integers :

more precisely, the formula $\forall x[\mathrm{int}(x) \to Px]$ is not realized.

Indeed, consider a proof-like term $\theta \Vdash \forall x[\mathrm{int}(x) \to Px]$

and choose n such that a_n is not in θ.

Then $\theta \star n . \pi_n \in \perp\!\!\!\perp$, i.e. $\theta \star n . \pi_n \succ a_n \star \pi_n$ which is impossible.

Non standard integers (cont.)

Suppose now we have an instruction σ with the following execution rule :

$\sigma \star t \mathbin{\bullet} \pi \succ t \star \underline{n} \mathbin{\bullet} \pi_n$ where π_n is the stack constant of π.

Then $\sigma \Vdash \forall x[\mathrm{int}(x) \to Px] \to \bot$

i.e. *there are non-standard integers in every 2-valued realizability model*.

Indeed, let $t \Vdash \forall x[\mathrm{int}(x) \to Px]$ and $\pi \in \Pi$.

We must show that $\sigma \star t \mathbin{\bullet} \pi \in \perp\!\!\!\perp$, i.e. $t \star \underline{n} \mathbin{\bullet} \pi_n \in \perp\!\!\!\perp$.

This follows from the hypothesis on t.

Instructions similar with σ will be used in order to realize
the axiom of dependent choice.

Examples of arithmetical theorems

Theorem. Let $\vdash \theta : \exists x[\text{int}(x) \wedge f(x) = 0]$, with f recursive. Let κ be a stop instruction. Then $\theta \star T\kappa \centerdot \pi \succ \kappa \star s^n 0 \centerdot \pi$ with $f(n) = 0$; T is the storage operator.

Proof. We have $\theta \Vdash \forall x[\text{int}(x) \to f(x) \neq 0] \to \bot$.

Now take $\perp\!\!\!\perp = \{\text{p} \,;\, \text{p} \succ \kappa \star s^n 0 \centerdot \pi$ with $f(n) = 0\}$.

We simply have to show that $T\kappa \Vdash \forall x[\text{int}(x) \to f(x) \neq 0]$ i.e. by the storage lemma, that $\kappa \star s^n 0 \centerdot \pi \in \perp\!\!\!\perp$ for every n such that $\pi \in \| f(n) \neq 0 \|$.

But this means that $\| f(n) \neq 0 \| \neq \emptyset$ and thus $f(n) = 0$. QED

Remark. κ is clearly a *pointer to an integer*. In the program, we wrote $T\kappa$, because we want it to point to a *computed* integer.

It is the intuitive meaning of *imperative call-by-value*.

Examples of arithmetical theorems (cont.)

We consider now an arithmetical theorem $\{\exists x \forall y [f(x,y) \neq 0]\}^{\text{int}}$.
Define a game with two players $\exists$ and $\forall$: $\exists$ plays an integer m, $\forall$ answers by n ;
the play stops as soon as $f(m,n) \neq 0$ and then $\exists$ wins ;
thus $\forall$ wins if and only if the play does not stop.
Intuitively, $\exists$ is the " defender " of the theorem and
$\forall$ " attacks " it, searching to exhibit a counter-example.
It is clear that $\exists$ has a winning strategy if and only if $\mathbb{N} \models \exists x \forall y [f(x,y) \neq 0]$; then,
there is an obvious strategy for $\exists$: simply play successively $0, 1, 2, \ldots$
We show that a proof of $\{\exists x \forall y [f(x,y) \neq 0]\}^{\text{int}}$ gives an explicit programming
of a winning strategy for the " defender ".
Usually, this strategy is much more efficient than the trivial one.

Programming a winning strategy

Let us add to our symbolic machine, an instruction κ which allows an interactive execution. Its execution rule is :

$$\kappa \star s^n 0 \centerdot \xi \centerdot \pi \succ \xi \star s^p 0 \centerdot \pi_{np}$$

for $n, p \in \mathbb{N}$; π_{np} is a stack constant.

This execution rule is non deterministic since p is arbitrary. Intuitive meaning :
in the left hand side, the program (the player $\exists$), plays the integer n and prepares a handler ξ for the answer of $\forall$; in the right hand side, the attacker $\forall$ plays p ;
π_{np} store the information about this move.

Theorem. If $\vdash \theta : \{\exists x \forall y (f(x,y) \neq 0)\}^{\text{int}}$, then *every reduction* of $\theta \star T\kappa \centerdot \pi$ gives $\xi \star s^p 0 \centerdot \pi_{np}$ with $f(n,p) \neq 0$ (T is the storage operator).

This means that the process $\theta \star T\kappa \centerdot \pi$ acts as a winning strategy for $\exists$.

Programming a winning strategy (cont.)

Proof. Take for $\perp\!\!\!\perp$ the set of processes *every* reduction of which gives $\xi \star s^p 0 \centerdot \pi_{np}$ with $f(n,p) \neq 0$. We must show that $\theta \star T\kappa \centerdot \pi \in \perp\!\!\!\perp$.

Now $\theta \Vdash \forall x[\mathrm{int}(x), \forall y(\mathrm{int}(y) \to f(x,y) \neq 0) \to \bot] \to \bot$.

Therefore, by definition of $\Vdash$, it is sufficient to show that :

$T\kappa \Vdash \forall x[\mathrm{int}(x), \forall y(\mathrm{int}(y) \to f(x,y) \neq 0) \to \bot]$.

By the storage lemma, we only need to show that :

if $\xi \Vdash \forall y(\mathrm{int}(y) \to f(n,y) \neq 0)$ then $\kappa \star s^n 0 \centerdot \xi \centerdot \pi \in \perp\!\!\!\perp$, i.e.

$\xi \star s^p 0 \centerdot \pi_{np} \in \perp\!\!\!\perp$ for every $p \in \mathbb{N}$.

If $f(n,p) \neq 0$, this is true by definition of $\perp\!\!\!\perp$.

Else, $\pi_{np} \in \| f(n,p) \neq 0 \| = \Pi$, hence the result, by hypothesis on ξ. QED

Programming a winning strategy (cont.)

Remark. κ can be considered as a *pointer* to the *object* (n, ξ) consisting of the integer n and the handler ξ (data and method). Moreover, the integer n is *called by value* which is guaranteed by writing $T\kappa$ instead of κ.

Example. We take the theorem $\{\exists x \forall y[f(x) \le f(y)]\}^{\mathrm{int}}$ where f is recursive.
Let $\phi(x, y)$ be the characteristic function of the well founded relation $f(x) < f(y)$.
The formula is $\forall x[\mathrm{int}(x), \forall y(\mathrm{int}(y) \to \phi(y, x) \ne 1) \to \bot] \to \bot$.
A particular case of the result page 51 is :
$\mathsf{Y} \Vdash \forall x[\forall y(\mathrm{int}(y) \to \phi(y, x) \ne 1) \to \neg\mathrm{int}(x)] \to \forall x \neg\mathrm{int}(x)$.
Thus, we get $\theta = \lambda h(\mathsf{Y}\lambda x \lambda n\, hnx)0$. It is easily checked that the process $\theta \star T\kappa \,.\, \pi$ gives the following strategy, much better than the trivial one :
$\exists$ plays 0 ; if $\forall$ plays p and if $f(p) < f(0)$, then $\exists$ plays p and so on.

The axiom of dependent choice

We need a *new instruction* in our machine. Any of the following two will work :

1. The signature. Let $t \mapsto n_t$ be a function from closed terms into the integers, which is *very easily computable* and *“practically” one-to-one*. It means that the one-to-one property has to be true only for the terms which appear during the execution of a given process. And also that we never try to compute the inverse function.

We define an instruction σ with the following reduction rule :

$$\sigma \star t \bullet \pi \succ t \star \underline{n_t} \bullet \pi.$$

A simple way to implement such an instruction is to take for n_t the *signature* of the term t, given by a standard algorithm, such as MD5 or SHA1.

Indeed, these functions are almost surely one-to-one for the terms which appear during a finite execution of a given process.

The axiom of dependent choice (cont.)

2. The clock. It is denoted as $\hbar$ and its reduction rule is :

$$\hbar \star t \,\bullet\, \pi \succ t \star \underline{n} \,\bullet\, \pi$$

where $\underline{n}$ is a Church integer which is the current time (for instance, the number of reduction steps from the boot).

Both instructions, the clock and the signature, can be given (realize) the same type, which is not DC but a formula DC′ which *implies DC in classical logic*.
By means of this proof, we get a λ-term $\gamma[\mathrm{cc}, \sigma]$ or $\gamma[\mathrm{cc}, \hbar]$ which has the type DC.
The instructions σ, $\hbar$ appear only inside this λ-term γ.
By looking at its behavior, we find that the integers produced by these instructions are only compared with each other.
No other operation is performed on these integers.

” Proof ” of the dependent choice axiom

For simplicity, we consider the *countable choice axiom* :

$$\exists Z \forall x (F[x, Z(x,y)/Xy] \to \forall X\, F[x,X])$$

Indeed, the dependent choice is the same formula in which the *individual parameter* x is replaced with a *predicate parameter*. There is nothing to change in the following proofs, because the parameter does not play any role.

We use a variant of the instruction σ with the following reduction rule :

$$\sigma \star t \,\bullet\, \pi \succ t \star \underline{n}_\pi \,\bullet\, \pi$$

($\pi \mapsto n_\pi$ is a given recursive bijection of Π onto $\mathbb{N}$).

Theorem. There exists a ” predicate ” $U : \mathbb{N}^3 \to \mathscr{P}(\Pi)$ such that $\sigma \Vdash \forall x\{\forall n(\mathrm{int}[n] \to F[x, U(x,n,y)/Xy]) \to \forall X\, F[x,X]\}$.

The dependent choice axiom (cont.)

The usual countable choice axiom follows easily, *but not intuitionistically*.
Simply define, for each x, the unary predicate $Z(x,\bullet)$ as $U(x,n,\bullet)$ for the first integer n s.t. $\neg F[x,U(x,n,y)/Xy]$, or as $\mathbb{N}$ if there is no such integer :

$$Z(x,z) \equiv \forall n\{\mathrm{int}(n), \forall p(\mathrm{int}(p), p<n \to F[x,U(x,p,y)/Xy]),$$
$$\neg F[x,U(x,n,y)/Xy] \to U(x,n,z)\}.$$

Proof. By definition of $\|\forall X\, F[x,X]\|$, we have :
$\pi \in \|\forall X\, F[x,X]\| \Leftrightarrow (\exists R \in \mathscr{P}(\Pi)^{\mathbb{N}})\, \pi \in \|F[x,R/X]\|$.
By countable choice, we get a function $U : \mathbb{N}^3 \to \mathscr{P}(\Pi)$ such that
$\pi \in \|\forall X\, F[x,X]\| \Leftrightarrow \pi \in \|F[x,U(x,n_\pi,y)/Xy]\|$.
Let $x \in \mathbb{N}$, $t \Vdash \forall n(\mathrm{int}[n] \to F[x,U(x,n,y)/Xy])$ and $\pi \in \|\forall X\, F[x,X]\|$.
We must show that $\sigma \star t \centerdot \pi \in \perp\!\!\!\perp$ and, by the rule for σ,
it suffices to show $t \star \underline{n_\pi} \centerdot \pi \in \perp\!\!\!\perp$. But this follows from
$\underline{n_\pi} \Vdash \mathrm{int}(s^{n_\pi}0)$, $\pi \in \|F[x,U(x,n_\pi,y)/Xy]\|$ (by definition of U) and
$t \Vdash \mathrm{int}(s^{n_\pi}0) \to F[x,U(x,n_\pi,y)/Xy]$. QED

Instructions for dependent choice

This proof gives a rather complicated term γ containing $\hbar$ and cc which realizes the dependent choice axiom. It is much clearer to give it as an instruction ; in any case, this is necessary if we use *head linear reduction*.

We introduce four instructions $\gamma, \mathsf{E}, \mathsf{U}_0, \mathsf{U}_1$. Their execution rules are :

$\gamma \star t . \pi \succ \mathsf{E} \star t . \underline{n} . \pi$ where n is the *current time* or the *number of the stack π*.

$\mathsf{E} \star t . m . \pi \succ t \star ((\mathsf{U}_0)(\mathsf{E}t)m)\mathsf{k}_\pi . ((\mathsf{U}_1)(\mathsf{E}t)m)\mathsf{k}_\pi . \pi$

$\mathsf{U}_0 \star t . m . k . u . \rho \succ u \star t . m . k . \rho$

$\mathsf{U}_1 \star t . \underline{n} . \mathsf{k}_\pi . u . t' . \underline{n}' . \mathsf{k}_{\pi'} . \rho \succ u \star \rho$ if $n = n'$;

$\succ t' \star \underline{n} . \pi$ if $n < n'$

$\succ t \star \underline{n}' . \pi'$ if $n' < n$

π, π', ρ are arbitrary stacks ; t, t', m, k, u are arbitrary terms ;
$\underline{n}, \underline{n}'$ are integers *in the form introduced by γ*.

Instructions for dependent choice (cont.)

We now show that γ realizes the dependent choice, as follows : given a formula $F[Y]$ with some parameters we do not write, we explicitly define a unary predicate $V:\mathbb{N}\to\mathscr{P}(\Pi)$ and prove that $\gamma \Vdash \forall Y(\forall y(Vy \leftrightarrow Yy) \to F[Y]) \to \forall Y\, F[Y]$.

Remark. It will be clear, from the definition of V, that if the formula is $F[X,Y]$ with the parameter X, then $V:\mathscr{P}(\Pi)^{\mathbb{N}}\times\mathbb{N}\to\mathscr{P}(\Pi)$, i.e. $V:\mathscr{P}(\Pi)^{\mathbb{N}}\to\mathscr{P}(\Pi)^{\mathbb{N}}$.

Thus, we have $\gamma \Vdash \forall X\{\forall Y(\forall y(V[X](y) \leftrightarrow Yy) \to F[X,Y]) \to \forall Y\, F[X,Y]\}$

which is stronger than dependent choice (*non extensional axiom of choice*).

We first define a binary predicate $U:\mathbb{N}^2\to\mathscr{P}(\Pi)$, such that for every stack π :

$\pi \in \|\forall X\, F[X]\| \Rightarrow \pi \in \|F[U_n]\|$ where $\pi = \pi_n$ (n is the number of π).

U_n is the unary predicate defined by $U_n(y) = U(n,y)$.

Since $\|\forall X\, F[X]\| = \bigcup\{F[V];\ V:\mathbb{N}\to\mathscr{P}(\Pi)\}$, this is a simple application of the countable choice axiom.

Instructions for dependent choice (cont.)

Define now a unary predicate $V : \mathbb{N} \to \mathscr{P}(\Pi)$ in the following way :

$V(y) \equiv \forall n\{\bigcap_{m<n}|\{\underline{m}\} \to F[U_m]|, \{\underline{n}\}, \Phi_n \to U_n y\}$

with $\Phi_n = \{\mathsf{k}_\pi;\ \pi \in \|F[U_n]\|\}$.

Intuitively, V is U_n for the first integer n such that $\neg F[U_n]$ if there is one, and $\mathbb{N}$ otherwise. Indeed, $\{\underline{m}\}$ stands for $\mathsf{int}(m)$ and Φ_n for $\neg F[U_n]$.

Lemma. $\mathsf{E} \Vdash \forall n(\forall y(Vy \leftrightarrow U_n y) \to F[U_n]) \to \bigcap_n |\{n\} \to F[U_n]|$.

Remark. This lemma will be used with the stronger hypothesis :

$\forall Y(\forall y(Vy \leftrightarrow Yy) \to F[Y])$ which is an abbreviation for :

$\forall y(Vy \to Yy), \forall y(Yy \to Vy) \to F[Y]$.

We prove, by induction on n, that

if $t \Vdash \forall n\{\forall y(Vy \leftrightarrow U_n y) \to F[U_n]\}$ and $\pi \in \|F[U_n]\|$ then $\mathsf{E} \star t \centerdot \underline{n} \centerdot \pi \in \perp\!\!\!\perp$.

Using the rule for E, it suffices to show that :

i) $(((\mathsf{U}_0)(\mathsf{E})t)\underline{n})\mathsf{k}_\pi \Vdash \forall y(Vy \to U_n y)$

ii) $(((\mathsf{U}_1)(\mathsf{E})t)\underline{n})\mathsf{k}_\pi \Vdash \forall y(U_n y \to Vy)$.

Proof of (i). Let $v \Vdash Vy$ and $\rho \in \|U_n y\|$; we must show that

$\mathsf{U}_0 \star \mathsf{E}\, t \,.\, \underline{n} \,.\, \mathsf{k}_\pi \,.\, v \,.\, \rho \in \perp\!\!\!\perp$, i.e. $v \star \mathsf{E}\, t \,.\, \underline{n} \,.\, \mathsf{k}_\pi \,.\, \rho \in \perp\!\!\!\perp$.

By the induction hypothesis, we have $\mathsf{E}\, t \in \bigcap_{m<n} |\{m\} \to F[U_m]|$.

By hypothesis on π, we have $\mathsf{k}_\pi \in \Phi_n$. Hence the result, by definition of $V(y)$.

Proof of (ii). Let $u \Vdash U_n y$, $\eta' \in \bigcap_{m<n'} |\{m\} \to F[U_m]|$, $n' \in \mathbb{N}$, $\pi' \in \|F[U_{n'}]\|$ and $\rho \in \|U_{n'} y\|$. We have to show that $\mathsf{U}_1 \star \mathsf{E}\, t \,.\, \underline{n} \,.\, \mathsf{k}_\pi \,.\, u \,.\, \eta' \,.\, \underline{n}' \,.\, \mathsf{k}_{\pi'} \,.\, \rho \in \perp\!\!\!\perp$.

If $n = n'$, this is $u \star \rho \in \perp\!\!\!\perp$, which is true by the hypothesis on u and ρ.

If $n < n'$, this is $\eta' \star \underline{n} \,.\, \pi \in \perp\!\!\!\perp$, which is true by the hypothesis on η' and π.

If $n' < n$, this is $\mathsf{E}\, t \star \underline{n}' \,.\, \pi' \in \perp\!\!\!\perp$. But, by the induction hypothesis, we have :

$\mathsf{E}\, t \Vdash \{\underline{n}'\} \to F[U_{n'}]$, hence the result since, by hypothesis, $\pi' \in \|F[U_{n'}]\|$. QED

Theorem. $\gamma \Vdash \forall Y\{\forall y(Vy \leftrightarrow Yy) \to F[Y]\} \to \forall Y\, F[Y]$.

Let $t \Vdash \forall Y\{\forall y(Vy \leftrightarrow Yy) \to F[Y]\}$ and $\pi \in \|\forall Y\, F[Y]\|$.

Thus, we have $\pi \in \|F[U_n]\|$ where n is the number of π ($\pi = \pi_n$).

By the lemma above, it follows that $\mathsf{E} \star t \bullet \underline{n} \bullet \pi \in \perp\!\!\!\perp$, which gives the result, using the execution rule of γ. QED

As explained before, if $F \equiv F[X, Y]$ has a second order unary predicate parameter X then $V : \mathscr{P}(\Pi)^{\mathbb{N}} \to \mathscr{P}(\Pi)^{\mathbb{N}}$ is defined by

$V[X, y] \equiv \forall n\{\bigcap_{m<n} |\{\underline{m}\} \to F[X, U_m[X]]|, \{\underline{n}\}, \Phi_n \to U_n[X, y]\}$.

We have $\gamma \Vdash \forall X\{\forall Y(\forall y(V[X](y) \leftrightarrow Yy) \to F[X, Y]) \to \forall Y\, F[X, Y]\}$

and $V[X]$ is a choice function which is *non-extensional* :

the formula $\forall x(Xx \leftrightarrow X'x) \to \forall y(V[X, y] \leftrightarrow V[X', y])$ is not realized.

Nevertheless, we get the dependent choice, because we can iterate the function V, which gives the desired sequence $V^n[X_0]$ of predicates.

Example

We prove the following formula intuitionistically from the axiom of choice :
$\forall a[(Ra \to \forall x\, Rx) \to \bot] \to \bot$, which we denote $\exists^* a[Ra \to \forall x\, Rx]$.
Take $F[X] \equiv X \neq \emptyset \to X \cap R \neq \emptyset$ i.e. $F[X] \equiv \exists x\, Xx \to \exists x\{Xx, Rx\}$.
By the axiom of choice : $\gamma \Vdash \forall X\{\forall x(Vx \leftrightarrow Xx) \to F[X]\} \to \forall X\, F[X]$.
As every formula, $F[X]$ is compatible with extensionality.
Thus $F[V] \vdash \forall X\{\forall x(Vx \leftrightarrow Xx) \to F[X]\}$.
We easily show $\forall X\, F[X] \vdash \forall x\, Rx$: take $Xx \equiv (x = y)$.
Now, we have to show $\vdash \exists^* a(Ra \to F[V])$, i.e. $\exists^* a(\exists x\, Vx, Ra \to \exists x\{Vx, Rx\})$. By the intuitionistic rule : $A \to \exists^* a\, B(a) \vdash \exists^* a[A \to B(a)]$, we have now to show :
$\exists x\, Vx \to \exists^* a(Ra \to \exists x\{Vx, Rx\})$. It is sufficient to prove :
$\exists x\, Vx \to \exists^* a(Ra \to Va \wedge Ra)$ i.e. $\exists x\, Vx \to \exists^* a(Ra \to Va))$ or finally
$\exists x\, Vx \to \exists^* a\, Va$ which is trivial.

Thus, we obtain $\vdash \exists^* a[Ra \to \forall x\, Rx]$ by an intuitionistic proof from the non-extensional axiom of choice. The program we get contains γ but no occurrence of cc. Here it is :

$B = \lambda k(k)\lambda r$

$(\gamma\lambda x\lambda y\lambda\alpha\lambda u((u)(x)(\alpha)\lambda x(k)\lambda r(\gamma\lambda x'\lambda y'\lambda\alpha\lambda u((u)(x')(y)x)r)JI)r)JI$

with $I = \lambda x\, x$, $J = \lambda x\, xI$.

We have checked that this term implements correctly a winning strategy for $\exists$ in the game associated with the formula $\forall a[(Ra \to \forall x\, Rx) \to \bot] \to \bot$.

The standard realizability model of Analysis

Realizability models are obtained by choosing a set $\perp\!\!\!\perp$ which must be *saturated* and *coherent*. Let $\perp\!\!\!\perp^c$ be the complement of $\perp\!\!\!\perp$. The conditions on $\perp\!\!\!\perp^c$ are :

p $\in \perp\!\!\!\perp^c$, p $\succ$ q $\Rightarrow$ q $\in \perp\!\!\!\perp^c$ (saturation) ;

for every proof-like term ξ there is a stack π s.t. $\xi \star \pi \in \perp\!\!\!\perp^c$ (coherence).

Let $\xi \mapsto \pi_\xi$ be a one-one map from proof-like terms into stack constants.

If $\xi \star \pi_\xi \in \perp\!\!\!\perp^c$ for every ξ, the set $\perp\!\!\!\perp$ is obviously coherent. The set of all processes obtained by executing $\xi \star \pi_\xi$ will be called the *thread* generated by the proof-like term ξ, and $\xi \star \pi_\xi$ is the *boot* of this thread.

Thus, $\perp\!\!\!\perp^c =$ the union of all threads is a somewhat canonical way to define $\perp\!\!\!\perp$.

We have thus $\perp\!\!\!\perp^c = \{$p; there is a proof-like ξ s.t. $\xi \star \pi_\xi \succ$ p$\}$

We call this model the *standard realizability model*.

Nevertheless, as we shall see, it contains non standard integers.

$\mathscr{B}$ in the standard realizability model

We show that *the Boolean algebra $\mathscr{B}$* of individuals x s.t. $x^2 = x$ *is non trivial*.

Theorem. Let $d_0 = \delta\delta 0$ and $d_1 = \delta\delta 1$, with $\delta = \lambda x\, xx$. Then :

$\lambda x(\mathsf{cc})\lambda k((x)(k)d_0)(k)d_1 \Vdash \forall x(x \neq 1, x \neq 0 \to x^2 \neq x) \to \bot$.

We know that $|\forall x(x \neq 1, x \neq 0 \to x^2 \neq x)| = |\top, \bot \to \bot| \cap |\bot, \top \to \bot|$.

Let $t \in |\top, \bot \to \bot| \cap |\bot, \top \to \bot|$ and $\pi \in \Pi$. We must show that :

$\lambda x(\mathsf{cc})\lambda k((x)(k)d_0)(k)d_1 \star t \,.\, \pi \in \perp\!\!\!\perp$ that is $t \star \mathsf{k}_\pi d_0 \,.\, \mathsf{k}_\pi d_1 \,.\, \pi \in \perp\!\!\!\perp$. If this is not true, by hypothesis on t, we have $\mathsf{k}_\pi d_0, \mathsf{k}_\pi d_1 \not\Vdash \bot$. Therefore, both terms appear in head position in some thread ; since they both contain the stack constant of π, these threads are the same one ; thus d_0 and d_1 appear in head position in the same thread, which is absurd. QED

We now show that this Boolean algebra is *atomless*.

Theorem. Let $\theta = \lambda x \lambda y \, \mathrm{cc} \lambda k((x)(k)y0)((x)(k)y1)(k)y2$. Then we have :
$\theta \Vdash \forall x[\forall y(xy \neq 0, xy \neq x \to y^2 \neq y), x \neq 0 \to x^2 \neq x]$
(which means that the Boolean algebra $\mathscr{B}$ has no atom).

A simple computation shows that we have to prove i) and ii) :

i) $\theta \Vdash (\bot, \bot \to \bot), \bot \to \bot$.

Let $t \in |\bot, \bot \to \bot|$ and $u \in |\bot|$.

We have to show that $\theta \star t.u.\pi \in \perp\!\!\!\perp$ i.e. $= t \star \mathrm{k}_\pi u0.((t)(\mathrm{k}_\pi)u1)(\mathrm{k}_\pi)u2.\pi \in \perp\!\!\!\perp$.

But $u \Vdash \bot \Rightarrow \mathrm{k}_\pi u\xi \Vdash \bot$ for all ξ. Since $t \Vdash \bot, \bot \to \bot$, it follows that $((t)(\mathrm{k}_\pi)u1)(\mathrm{k}_\pi)u2 \Vdash \bot$ and therefore $t \star \mathrm{k}_\pi u0.((t)(\mathrm{k}_\pi)u1)(\mathrm{k}_\pi)u2.\pi \in \perp\!\!\!\perp$.

ii) $\theta \Vdash |\top, \bot \to \bot| \cap |\bot, \top \to \bot|, \top \to \bot$.

Let $t \in |\top, \bot \to \bot| \cap |\bot, \top \to \bot|$ and $u \in \Lambda_c$.

Again, we have to show that $t \star \mathrm{k}_\pi u0.((t)(\mathrm{k}_\pi)u1)(\mathrm{k}_\pi)u2.\pi \in \perp\!\!\!\perp$.

Let $t \in |\top, \bot \to \bot| \cap |\bot, \top \to \bot|$ and $u \in \Lambda_c$.

Again, we have to show that $t \star \mathsf{k}_\pi u0.((t)(\mathsf{k}_\pi)u1)(\mathsf{k}_\pi)u2.\pi \in \perp\!\!\!\perp$.

If this is not true, the hypothesis on t gives successively :

$\mathsf{k}_\pi u0 \not\Vdash \bot$ and $((t)(\mathsf{k}_\pi)u1)(\mathsf{k}_\pi)u2 \not\Vdash \bot$; and then $\mathsf{k}_\pi u1 \not\Vdash \bot$ and $\mathsf{k}_\pi u2 \not\Vdash \bot$.

It follows that $\mathsf{k}_\pi u0, \mathsf{k}_\pi u1, \mathsf{k}_\pi u2$ all appear in head position in some thread.

Since they contain k_π, these threads are the same (their stack constant is the same).

Suppose, for example, that $\mathsf{k}_\pi u0$ appears first in head position,

then $\mathsf{k}_\pi u1$, and then $\mathsf{k}_\pi u2$. We have thus :

$\mathsf{k}_\pi u0 \star \pi_0 \succ u \star \pi \succ \cdots \succ \mathsf{k}_\pi u1 \star \pi_1 \succ u \star \pi \succ \cdots \succ \mathsf{k}_\pi u2 \star \pi_2 \succ u \star \pi \succ \cdots$

But such an execution is clearly impossible because, at the second appearance

of the process $u \star \pi$, we enter in a loop and can never arrive at $\mathsf{k}_\pi u2 \star \pi_2$. QED

Thus, the standard realizability model contains *non integers*.

We now show it contains also *non-standard integers*.

A generic non-standard integer

Let $n \mapsto \xi_n$ be a fixed recursive enumeration of proof-like terms. We define a unary predicate G by setting :

$\|Gn\| = \Pi_n$ i.e. the set of stacks which end with the constant π_{ξ_n}.

We assume there is no instruction which changes the stack constant.

It follows that π_ξ is the only one which appears in the thread $\xi \star \pi_\xi$.

Since $\bigcup_n \Pi_n = \Pi$, we get $\|\forall x\, Gx\| = \Pi$, thus $I \Vdash \neg\forall x\, Gx$.

We show that Gn is realized for each integer n. Indeed suppose that :

$\delta\delta 0 \not\Vdash Gn$ and $\delta\delta 1 \not\Vdash Gn$ with $\delta = \lambda x\, xx$.

Then, $\xi_n \star \pi_{\xi_n} \succ \delta\delta 0 \star \pi_0$ and $\xi_n \star \pi_{\xi_n} \succ \delta\delta 1 \star \pi_1$ which is impossible.

It follows that the predicate G contains every *standard* integer,

but not every individual. Does it contain every integer ?

A generic non-standard integer (cont.)

Let ς (for "self") be a new instruction with the following reduction rule :

$\varsigma \star t \centerdot \pi \succ t \star \underline{n} \centerdot \pi$; n is the integer such that $\pi \in \Pi_n$.

Then $\varsigma \Vdash \forall x(\mathrm{int}(x) \to Gx) \to \bot$.

Indeed, if $t \Vdash \forall x(\mathrm{int}(x) \to Gx)$ and $\pi \in \Pi_n$, then $\underline{n} \Vdash \mathrm{int}(n)$ and $\pi \in \|Gn\|$.

Thus $t \star \underline{n} \centerdot \pi \in \perp\!\!\!\perp$ and $\varsigma \star t \centerdot \pi \in \perp\!\!\!\perp$.

It follows that the predicate $\neg G$ contains at least one *non-standard integer*.

Theorem. $\lambda x \lambda y\, y \Vdash \forall x \forall y\{\neg Gx, x \neq y \to Gy\}$.

By the trick on page 23, it is sufficient to show that :

$\lambda x \lambda y\, y \star \mathsf{k}_\pi \centerdot t \centerdot \rho \in \perp\!\!\!\perp$ with $\pi \in \|Gn\| = \Pi_n$, $t \Vdash n \neq p$ and $\rho \in \|Gp\| = \Pi_p$.

If $n \neq p$, this process is in no thread, because

it contains two different stack constants π_{ξ_n} and π_{ξ_p}.

If $n = p$, then $t \Vdash \bot$ and $\lambda x \lambda y\, y \star k_\pi \centerdot t \centerdot \rho \succ t \star \rho$, hence the result. QED

A generic non-standard integer (cont.)

Thus, the predicate $\neg Gx$ consists in *exactly one* individual
and it is a non-standard integer. We call it *the generic integer*.
We add a new individual constant g to our language,
and replace Gx with $x \neq \mathsf{g}$.
The non-standard proof-like term ξ_{g} has remarkable properties.
Indeed, we shall prove that the "generic thread" $\xi_{\mathsf{g}} \star \pi_{\xi_{\mathsf{g}}}$ does not stop
and execute every standard solvable term.
More precisely, for every standard proof-like term $\theta \Vdash \bot, \ldots, \bot \to \bot$
we have $\xi_{\mathsf{g}} \star \pi_{\xi_{\mathsf{g}}} \succ \theta \star \pi$.

The clock in the standard realizability model

The execution rule of the clock instruction $\hbar$ is defined *formally* as follows :
let π_ξ be the stack constant of the current process $\hbar \star t \mathbin{.} \pi$.
" Reboot " $\xi \star \pi_\xi$ until you arrive at $\hbar \star t \mathbin{.} \pi$ (if this never happens, you are stuck).
Let n be the number of steps ; then $\hbar \star t \mathbin{.} \pi \succ t \star \underline{n} \mathbin{.} \pi$.
The *implementation* is much simpler : you only have to set a counter
which is incremented at each step.
Warning : you must check that the current process $\hbar \star t \mathbin{.} \pi$ was not attained before.
In this case, you enter an endless loop.

Definition. A term θ is called *strongly solvable* if $\theta \Vdash \perp \to \perp$.
This means that, if $\theta \star t$ comes in head position in a thread, and t is not proof-like,
then t comes in head position in this thread.
θ is called *solvable* if $\lambda x \theta x \ldots x$ is strongly solvable.
If θ is a usual λ-term, this is the usual notion of solvability.

Theorem. If θ is a (strongly) solvable proof-like term,
then it comes in head position in the generic thread.

Let $\phi_\theta : \mathbb{N}^2 \to \{0,1\}$ be the recursive function such that : $\phi_\theta(n,p) = 1$ iff
θ comes in head position in the thread $\xi_p \star \pi_{\xi_p}$ at the $(n+4)$-th step.
We show that $\hbar\lambda x\lambda y(\theta)(y)x \Vdash \forall p\{\forall n[\mathrm{int}(n) \to \phi_\theta(n,p) \neq 1] \to p \neq \mathsf{g}\}$
(in other words, $\exists n\{\mathrm{int}(n), \phi_\theta(n,\mathsf{g}) = 1\}$).
Let $p \in \mathbb{N}$, $\pi \in \|p \neq \mathsf{g}\|$ and $t \Vdash \forall n[\mathrm{int}(n) \to \phi_\theta(n,p) \neq 1]$.
Suppose that $\hbar\lambda x\lambda y(\theta)(y)x \star t.\pi \notin \perp\!\!\!\perp$. Therefore, this process appears in a thread, which is $\xi_p \star \pi_{\xi_p}$ because $\pi \in \|p \neq \mathsf{g}\|$. Thus, we have :
$\xi_p \star \pi_{\xi_p} \succ \hbar\lambda x\lambda y(\theta)(y)x \star t.\pi \succ \theta \star t\underline{n}.\pi$, where n is the number of steps
in the reduction of $\xi_p \star \pi_{\xi_p}$ until $\hbar\lambda x\lambda y(\theta)(y)x \star t.\pi$. Thus, we obtain $\theta \star t\underline{n}.\pi$
at the $(n+4)$-th step of reduction. Since θ is in head position at this moment,
we have $\phi_\theta(n,p) = 1$. By hypothesis on t, it follows that $t\underline{n} \Vdash \perp$. Now, by hypothesis,
$\theta \Vdash \perp \to \perp$ and therefore $\theta \star t\underline{n}.\pi \in \perp\!\!\!\perp$. This is a contradiction, because
$\theta \star t\underline{n}.\pi$ appears in the thread $\xi_p \star \pi_{\xi_p}$. QED

Theorem. If θ is a proof-like term such that $\theta\underline{m}$ is strongly solvable for each $m \in \mathbb{N}$, then the following formula is realized :
" $\forall m\{\text{int}(m) \to \theta\underline{m}$ comes in head position in the generic thread$\}$ ".
Remark. It follows that the generic thread neither stops, nor loops (take $\theta = 0$).
Let $\psi_\theta : \mathbb{N}^3 \to \{0,1\}$ be the recursive function defined by $\psi_\theta(m,n,p) = 1$ iff
$\theta\underline{m}$ comes in head position at the $(n+4)$-th step in the thread $\xi_p \star \pi_{\xi_p}$. We prove :
$T\lambda m\hbar\lambda x\lambda y(\theta m)(y)x \Vdash \forall p \forall m\{\text{int}(m), \forall n[\text{int}(n) \to \psi_\theta(m,n,p) \neq 1] \to p \neq \mathsf{g}\}$
(in other words $\forall m(\text{int}(m) \to \exists n\{\text{int}(n), \psi_\theta(m,n,\mathsf{g}) = 1\})$).
It is sufficient to prove that, for all integers m, p
and all stacks ρ in $\|\forall n[\text{int}(n) \to \psi_\theta(m,n,p) \neq 1] \to p \neq \mathsf{g}\|$, we have :
$\hbar\lambda x\lambda y(\theta\underline{m})(y)x \star \rho \in \perp\!\!\!\perp$. But this results from the last theorem
and the fact that $\psi_\theta(m,n,p) = \phi_{\theta\underline{m}}(n,p)$. QED

Clock and choice

We check that, with the clock instruction $\hbar$, the axiom of dependent choice is realized.

Theorem. Let $F[x,X]$ be a formula with parameters, X being a unary predicate variable. There exists $\Phi:\mathbb{N}^4 \to \mathscr{P}(\Pi)$ such that :

$\hbar \Vdash \forall x \forall p \forall X\{\forall n(\mathrm{int}(n), F[x,\Phi(n,p,x,y)/Xy] \to \bot), F[x,X] \to p \neq \mathsf{g}\}$.

We define $v:\mathbb{N}^2 \to \Lambda_c$ by putting : $v(n,p) =$ the λ_c-term u which is in second position in the stack, at the n-th execution step in the thread $\xi_p \star \pi_{\xi_p}$. At the n-th step of this execution, we have therefore a process of the form $\tau \star t.u.\pi$.

We define now $\Phi(n,p,x,y)$, (using axiom of choice), in such a way that :

If there exists $X:\mathbb{N} \to \mathscr{P}(\Pi)$ such that $v(n,p) \Vdash F[x,X]$

then $v(n,p) \Vdash F[x,\Phi(n,p,x,y)/Xy]$.

Then Φ has the desired property :

Clock and choice (cont.)

Consider $x, p \in \mathbb{N}$, $X : \mathbb{N} \to \mathscr{P}(\Pi)$, λ_c-terms t, u such that
$t \Vdash \forall n(\mathrm{int}(n), F[x, \Phi(n,p,x,y)/Xy] \to \perp)$, $u \Vdash F[x,X]$
and a stack $\pi \in \|p \neq \mathsf{g}\|$. We must show that $\hbar \star t.u.\pi \in \perp\!\!\!\perp$.
If not, then $\hbar \star t.u.\pi$ appears in a thread, at the n-th step.
By hypothesis on π, this thread is $\xi_p \star \pi_{\xi_p}$.
Thus, we have $u = v(n,p)$, by definition of v, hence $v(n,p) \Vdash F[x,X]$.
By definition of Φ, we get $u = v(n,p) \Vdash F[x, \Phi(n,p,x,y)/Xy]$.
But, since $\underline{n} \Vdash \mathrm{int}(n)$, it follows that $t \star \underline{n}.u.\pi \in \perp\!\!\!\perp$, by hypothesis on t.
This is a contradiction, because this process appears at the $(n+1)$-th step
in the thread $\xi_p \star \pi_{\xi_p}$. QED

Clock and choice (cont.)

It follows that the standard generic model satisfies the formula :
$\forall x \forall X (F[x,X] \to \exists n\{\mathrm{int}(n), F[x,\Phi(n,\mathsf{g},x,y)/Xy]\})$.
Thus, we can define the binary predicate $\Psi(x,y)$ by the formula :
" $\Phi(n,\mathsf{g},x,y)$ for the first integer n such that $F[x,\Phi(n,\mathsf{g},x,y)/Xy]$,
and $\top$ if there is no such integer ".
Then, we have, in the generic model :
$\forall x \forall X (F[x,X] \to F[x,\Psi(x,y)/Xy])$ which is the axiom of choice.

A game on first order formulas

We consider first order formulas written with :
$\rightarrow$, $\forall$, $\top$, $\bot$, $\neq$, predicate constants, function symbols for recursive functions.
A 1st order formula has the form $\forall\vec{x}[\Phi_1,\ldots,\Phi_n \rightarrow A]$ where $\Phi_1,\ldots,\Phi_n$
are 1st order formulas and A is atomic (i.e. $R t_1 \ldots t_k$ or $t_0 \neq t_1$ or $\top$ or $\bot$).
In the following, we only consider *closed* 1st order formulas.
The atomic closed formula $t_0 \neq t_1$ is interpreted as $\top$ (resp. $\bot$)
if it is true (resp. false) in $\mathbb{N}$.
We define a game between two players : $\exists$ (the defender) and $\forall$ (the opponent).
At each step, there are two sets $\mathscr{U},\mathscr{V}$ of closed 1st order formulas
and a set $\mathscr{A}$ of closed atomic formulas. $\mathscr{U}$ and $\mathscr{A}$ increase at each step.
$\mathscr{U}$ (resp. $\mathscr{V}$) is the choice set for $\exists$ (resp. $\forall$).
At the beginning of the game, $\mathscr{A} = \{\bot\}$, $\mathscr{V} = \{\Phi_1,\ldots,\Phi_k\}$ is a given (finite) set
of closed formulas and $\mathscr{U} = \{\Phi_1,\ldots,\Phi_k \rightarrow \bot\}$.

A move of the game is as follows : the player $\forall$ chooses a formula $\Phi \in \mathscr{V}$,
$\Phi \equiv \forall \vec{x}[\Psi_1(\vec{x}), \ldots, \Psi_m(\vec{x}) \to A(\vec{x})]$ and $\vec{i} \in \mathbb{N}^k$.
The atomic formula $A(\vec{i})$ must not be $\top$ (otherwise, $\forall$ has lost).
Then $\Psi_1(\vec{i}), \ldots, \Psi_m(\vec{i})$ *are added* to $\mathscr{U}$ and $A(\vec{i})$ *is added* to $\mathscr{A}$.
The player $\exists$ chooses $\Psi \in \mathscr{U}$, $\Psi = \forall \vec{y}[\Phi_1(\vec{y}), \ldots, \Phi_n(\vec{y}) \to B(\vec{y})]$ and $\vec{j} \in \mathbb{N}^l$
such that $B(\vec{j}) \in \mathscr{A}$ (this is always possible since $\Phi_1, \ldots, \Phi_k \to \bot$ is in $\mathscr{U}$).
$\mathscr{V}$ *is changed* into $\{\Phi_1(\vec{j}), \ldots, \Phi_n(\vec{j})\}$.
$\exists$ wins iff $\forall$ cannot play at some step (every formula of $\mathscr{V}$ ends with $\top$,
in particular if $\mathscr{V} = \emptyset$). Thus $\forall$ wins iff the play is infinite.
Intuitively, at each step, $\forall$ pretends that some formula of $\mathscr{V}$ is false and
each formula of $\mathscr{U}$ is true. Conversely, $\exists$ says that some formula of $\mathscr{U}$ is false
and each formula of $\mathscr{V}$ is true. Both agree that each formula of $\mathscr{A}$ is false.
In fact, the player $\forall$ tries to build a model over $\mathbb{N}$ in which $\mathscr{V}_0$ is not satisfied
and $\exists$ tries to avoid this.

Theorem. i) Any model $\mathscr{M}$ over $\mathbb{N}$ s.t. $\mathscr{M} \not\models \mathscr{V}_0$ gives a winning strategy for $\forall$.
ii) There exists a strategy for $\exists$ with the following property : every play that $\exists$ loses following this strategy, gives a model $\mathscr{M}$ over $\mathbb{N}$ s.t. $\mathscr{M} \not\models \mathscr{V}_0$.

i) We define a strategy for $\forall$ such that, at each step, we have $\mathscr{M} \not\models \mathscr{V}$, $\mathscr{M} \models \mathscr{U}$ and every formula of $\mathscr{A}$ is false in $\mathscr{M}$. This is true at the beginning of the game.
Thus, at each step, $\forall$ can choose $\Phi \in \mathscr{V}$ such that $\mathscr{M} \models \neg\Phi$.
Then $\Phi = \forall\vec{x}[\Psi_1(\vec{x}),\dots,\Psi_m(\vec{x}) \to A(\vec{x})]$ and $\forall$ can choose $\vec{i} \in \mathbb{N}^k$ such that $\mathscr{M} \models \Psi_1(\vec{i}),\dots,\Psi_m(\vec{i})$ and $\neg A(\vec{i})$.
Then, $\forall$ adds $\Psi_1(\vec{i}),\dots,\Psi_m(\vec{i})$ to $\mathscr{U}$ and $A(\vec{i})$ to $\mathscr{A}$.
Thus, $\mathscr{U}$ and the negation of formulas of $\mathscr{A}$ remain true in $\mathscr{M}$.
Then $\exists$ chooses $\Psi \in \mathscr{U}$, $\Psi = \forall\vec{y}[\Phi_1(\vec{y}),\dots,\Phi_n(\vec{y}) \to B(\vec{y})]$ and $\vec{j} \in \mathbb{N}^l$ such that $B(\vec{j}) \in \mathscr{A}$. Therefore, $B(\vec{j})$ is false in $\mathscr{M}$.
Since $\mathscr{M} \models \Psi$, the new set $\mathscr{V} = \{\Phi_1(\vec{j}),\dots,\Phi_n(\vec{j})\}$ is not satisfied by $\mathscr{M}$.

ii) The strategy for $\exists$ is as follows : fix an enumeration of all ordered pairs $(\Psi, \vec{j})$ where $\Psi = \forall \vec{y}[\Phi_1(\vec{y}), \ldots, \Phi_n(\vec{y}) \to B(\vec{y})]$ is a closed formula
and $\vec{j}$ is a finite sequence of integers of the same length as $\vec{y}$.
At each step, $\exists$ chooses the first allowed pair $(\Psi, \vec{j})$, *not chosen before*.
Consider a play which $\exists$ loses with this strategy. $\mathscr{M}$ is the model which satisfies exactly the closed atomic formulas which are never put in $\mathscr{A}$ during the play.
A pair $(\Psi, \vec{j})$ is called *acceptable* if Ψ is put is $\mathscr{U}$ and $B(\vec{j})$ in $\mathscr{A}$ at some step
where $B(\vec{y})$ is the final atom of Ψ.
Every acceptable pair is effectively played by $\exists$ at some step : indeed, let $(\Psi, \vec{j})$ be the first counter-example. At some step during the play, Ψ and $B(\vec{j})$ are respectively in $\mathscr{U}$ and $\mathscr{A}$ and every acceptable pair *before* $(\Psi, \vec{j})$ has been chosen by $\exists$.
At this moment, the strategy tells $\exists$ to play $(\Psi, \vec{j})$.
We prove, by induction, that $\mathscr{M}$ satisfies *every formula Ψ which is put in $\mathscr{U}$*
and the negation of *every formula Φ chosen by $\forall$* during the play.

Proof for Ψ : The result is clear if Ψ is atomic because, if Ψ is both in $\mathscr{U}$ and $\mathscr{A}$ then $(\Psi, \emptyset)$ is acceptable and thus will be chosen by $\exists$; then $\exists$ wins.
Let $\Psi = \forall \vec{y}[\Phi_1(\vec{y}), \ldots, \Phi_n(\vec{y}) \to B(\vec{y})]$. We must show that
$\mathscr{M} \models \Phi_1(\vec{j}), \ldots, \Phi_n(\vec{j}) \to B(\vec{j})$ for every $\vec{j} \in \mathbb{N}^k$.
This is clear if $B(\vec{j})$ is never put in $\mathscr{A}$, because $\mathscr{M} \models B(\vec{j})$.
Otherwise, $(\Psi, \vec{j})$ is acceptable and is chosen by $\exists$ at some step.
Then $\mathscr{V} = \{\Phi_1(\vec{j}), \ldots, \Phi_n(\vec{j})\}$ and $\Phi_1(\vec{j})$, for instance, is chosen by $\forall$.
By induction hypothesis, we have $\mathscr{M} \models \neg\Phi_1(\vec{j})$, which gives the result.
Proof for Φ : Let $\Phi = \forall \vec{x}[\Psi_1(\vec{x}), \ldots, \Psi_m(\vec{x}) \to A(\vec{x})]$; $\forall$ chooses $\vec{i}$
and puts $A(\vec{i})$ in $\mathscr{A}$ and $\Psi_1(\vec{i}), \ldots, \Psi_m(\vec{i})$ in $\mathscr{U}$. By induction hypothesis,
$\mathscr{M} \models \Psi_1(\vec{i}), \ldots, \Psi_m(\vec{i})$; and, by definition, $\mathscr{M} \not\models A(\vec{i})$. Thus $\mathscr{M} \models \neg\Phi$.

It follows that $\mathscr{M} \not\models \mathscr{V}_0$ since, at the beginning of the play, $\forall$ chooses $\Phi \in \mathscr{V}_0$.

QED

Specification of first order formulas

For every closed first order formula Φ, we define an instruction κ_Φ and a set $[\Phi] \subset \Pi$. If Φ is atomic, $\Phi \equiv R\vec{i}$, with $\vec{i} \in \mathbb{N}^k$, we choose a stack constant π_Φ and we set $[\Phi] = \{\pi_\Phi\}$ and also $\|\Phi\| = \{\pi_\Phi\}$.

If $\Phi \equiv \perp$ (resp. $\top$), then $[\Phi] = \Pi$ (resp. $\emptyset$). This settles the case when Φ is $t_0 \neq t_1$.

In general, $\Phi = \forall \vec{x}[\Psi_1(\vec{x}), \ldots, \Psi_n(\vec{x}) \to A(\vec{x})]$ where $A(\vec{x})$ is atomic.

The execution rule of κ_Φ is $\kappa_\Phi \star \xi_1 \centerdot \ldots \centerdot \xi_n \centerdot \pi \succ \xi_j \star \rho$

where $j \in \{1, \ldots, n\}$ and $\rho \in \Pi$ are defined in the following way :

the player $\exists$ first chooses $\vec{i} \in \mathbb{N}^k$ ($\vec{x}$ is of length k) such that $\pi \in [A(\vec{i})]$.

If this is impossible, then $\forall$ wins.

Then, the player $\forall$ chooses $j \in \{1, \ldots, n\}$ and a stack $\rho \in [\Psi_j(\vec{i})]$.

In particular, the player $\forall$ loses when $n = 0$ or $\Psi_j(\vec{i}) \equiv \top$.

Finally we define $[\Phi] = \{\kappa_{\Psi_1(\vec{i})} \centerdot \ldots \centerdot \kappa_{\Psi_n(\vec{i})} \centerdot \pi;\ \vec{i} \in \mathbb{N}^k, \pi \in [A(\vec{i})]\}$.

A game is associated with each process p : p is performed and $\exists$ wins iff the execution terminates with $\kappa_\Phi \star \pi$ where Φ is the closure of an atomic formula and $\pi \in [\Phi]$. In other words, iff $\forall$ cannot play any more.

It is clear that this game is exactly the same as before, but *the process plays the role of* $\exists$ for the choice of formulas ; $\exists$ still chooses the integers.
We shall see below that, by restricting formulas to the set int, the process will completely replace the player $\exists$.

Lemma. Define $\perp\!\!\!\perp = \{$p; $\exists$ has a winning strategy for the game associated with p$\}$. Then, for each closed formula Φ, we have $[\Phi] \subset \|\Phi\|$ and $\kappa_\Phi \Vdash \Phi$.
Proof by induction on Φ. The result is trivial if Φ is atomic.
If $\Phi = \forall \vec{x}[\Psi_1(\vec{x}), \ldots, \Psi_n(\vec{x}) \to A(\vec{x})]$, by induction hypothesis, we have $\kappa_{\Psi_j(\vec{i})} \Vdash \Psi_j(\vec{i})$ and $\pi \in \|A(\vec{i})\|$, which shows that $[\Phi] \subset \|\Phi\|$.

Now, suppose that $\xi_j \Vdash \Psi_j(\vec{i})$ for $1 \le j \le n$ and that $\pi \in \|A(\vec{i})\|$. We have to show that $\kappa_\Phi \star \xi_1 \centerdot \ldots \centerdot \xi_n \centerdot \pi \in \perp\!\!\!\perp$, i.e. that $\exists$ has a winning strategy for the game associated with this process. The strategy is first to choose this $\vec{i}$. Then, we have $\pi \in [A(\vec{i})]$ because $A(\vec{i})$ is atomic. After that, $\forall$ chooses j and $\rho \in [\Psi_j(\vec{i})]$. But, by the induction hypothesis, we have $\rho \in \|\Psi_j(\vec{i})\|$ and therefore $\xi_j \star \rho \in \perp\!\!\!\perp$. Now, $\exists$ can follow the strategy for the game associated with the process $\xi_j \star \rho$. QED

Corollary. If $\theta \Vdash \Phi$ for every $\perp\!\!\!\perp$ (in particular, if $\vdash \theta : \Phi$) and $\pi \in [\Phi]$, then $\exists$ has a winning strategy for the game defined by the process $\theta \star \pi$.

If Φ is $\forall \vec{x}[\Psi_1(\vec{x}), \ldots, \Psi_n(\vec{x}) \to A(\vec{x})]$ where $A(\vec{x})$ is atomic, then $\forall$ begins the play by choosing $\vec{i}$; then, the process $\theta \star \kappa_{\Psi_1(\vec{i})} \centerdot \ldots \centerdot \kappa_{\Psi_n(\vec{i})} \centerdot \pi_{A(\vec{i})}$ is started.

A move of the play happens each time a constant κ_Φ comes in head position. Then $\mathsf{p} = \kappa_\Phi \star \xi_1 \bullet \ldots \bullet \xi_n \bullet \pi_A$ and the process has already chosen, in place of $\exists$, the formulas Φ and A. The player $\exists$ has only to choose the integers $\vec{i}$. Then, $\forall$ chooses $j \in \{1, \ldots, n\}$ and a stack $\rho \in [\Psi_j(\vec{i})]$ i.e. creates new constants of term and stack. The process restarts with $\xi_j \star \rho$.

This corollary allows to classify the proofs of Φ (and more generally the terms which realize Φ) according to the strategies associated with them.

Examples.

i) $\theta = \lambda z\, z \lambda d\, \mathsf{cc}\, z \Vdash \exists x \forall y (Rx \to Ry)$ i.e. $\forall x[\forall y(Rx \to Ry) \to \bot] \to \bot$.

A very simple game : $\exists$ chooses $i \in \mathbb{N}$ or Rk in $\mathscr{U}$; $\forall$ chooses j and puts Rj in $\mathscr{A}$ and Ri in $\mathscr{U}$. The strategy given by θ wins at the second move.

The proofs are characterized by a pair $(m, n) \in \mathbb{N}^2$, with $m \leq n$: n is the number of moves and m is the choice of $\exists$ at the end of the play.

ii) Υ $\Vdash \forall x\{\forall y(Ry \to x \neq sy), Rx \to \bot\} \to \forall x(Rx \to \bot)$.

First, $\forall$ chooses n and puts Φ, Rn in $\mathscr{U}$. In the general move, $\exists$ chooses Rp if possible ($Rp \in \mathscr{U} \cap \mathscr{A}$) and wins ; or he chooses $q \in \mathbb{N}$. Then $\mathscr{V} = \{\forall y(Ry \to q \neq sy), Rq\}$; thus $\forall$ may choose Rq and put it in $\mathscr{A}$; else, if $q \neq 0$, he may put $R(q-1)$ in $\mathscr{U}$.

The strategy for $\exists$ given by Υ, is to always choose the last (and least) p such that $Rp \in \mathscr{U}$.

Another winning strategy is to successively choose $0, 1, \ldots, n$. This forces $\forall$ to put $R0, R1, \ldots, Rn$ in $\mathscr{A}$. Then $\exists$ can play Rn and win.

But this strategy does not correspond to any proof-like term. This is because, during the execution of processes, the choices of $\forall$ appear only in index of κ-instructions. Thus, there is no mean to compute with them, *unless using new instructions.*

Machine replaces man

We consider now a formula Φ^{int}, where Φ is 1st order. We use the notations $\text{int}(\vec{x}) \to F$ for $\text{int}(x_1),\ldots,\text{int}(x_k) \to F$ and $\vec{i}\,.\pi$ for $i_1 . \ldots . i_k . \pi$.

We have $\Phi^{\text{int}} \equiv \forall \vec{x}[\text{int}(\vec{x}), \Psi_1^{\text{int}}(\vec{x}), \ldots, \Psi_m^{\text{int}}(\vec{x}) \to A(\vec{x})]$ where A is atomic.

The game is exactly the same as for Φ.

We define, almost as before, the instructions κ_Φ and the set $[\Phi] \subset \Pi$:

If Φ is atomic, we choose a stack constant π_Φ and we set $[\Phi] = \{\pi_\Phi\} = \|\Phi\|$.

If $\Phi \equiv \perp$ (resp. $\top$), then $[\Phi] = \Pi$ (resp. $\emptyset$). This settles the case when Φ is $t_0 \neq t_1$.

In general, $\Phi = \forall \vec{x}[\Psi_1(\vec{x}), \ldots, \Psi_n(\vec{x}) \to A(\vec{x})]$ where $A(\vec{x})$ is atomic.

The execution rule of κ_Φ is : $\kappa_\Phi \star \vec{i}\,.\xi_1 . \ldots . \xi_n . \pi \succ \xi_j \star \rho$; $\vec{i}$ is a sequence of integers of the form $s^r 0$ of the same length k as $\vec{x}$, such that $\pi \in [A(\vec{i})]$.

If this condition is not fulfilled, the execution loops indefinitely.

$j \in \{1, \ldots, n\}$ and $\rho \in [\Psi_j(\vec{i})]$ are chosen by $\forall$.

Finally we define $[\Phi] = \{\vec{i}\,.T\kappa_{\Psi_1(\vec{i})} . \ldots . T\kappa_{\Psi_n(\vec{i})} . \pi;\ \vec{i} \in \mathbb{N}^k, \pi \in [A(\vec{i})]\}$.

Machine replaces man (cont.)

During the execution of a process, the machine plays in place of $\exists$.

The process implements completely a *strategy* for $\exists$.

The following theorem gives a specification for Π^1_1 consequences of Analysis.

Theorem. Let Φ be a closed 1st order formula. If $\theta \Vdash \Phi^{\text{int}}$ for every $\perp\!\!\!\perp$ (in particular, if $\vdash \theta : \Phi^{\text{int}}$ is provable in Analysis) and $\pi \in [\Phi]$, then the process $\theta \star \pi$ plays a winning strategy for $\exists$.

Examples. i) We have already given examples of the form $[\exists x \forall y (f(x,y) \neq 0)]^{\text{int}}$.

ii) Consider $\Phi \equiv \forall x[\forall y(Xy \to Xsy), X0 \to Xx]$, which is not realized.

But $\Phi^{\text{int}} \equiv \forall x[\text{int}(x), \forall y(\text{int}(y), Xy \to Xsy), X0 \to Xx]$ is provable.

The game : first, $\forall$ chooses n and puts Xn in $\mathscr{A}$ and $\forall y(Xy \to Xsy), X0$ in $\mathscr{U}$.

During the game, $\exists$ chooses Xp if possible (i.e. $Xp \in \mathscr{U} \cap \mathscr{A}$) and wins ;

or he chooses q such that $Xsq \in \mathscr{A}$ and sets $\mathscr{V} = \{Xq\}$.

Then $\forall$ must choose Xq and put it in $\mathscr{A}$.

Computing predecessor

At the beginning of the play, the player $\exists$ has no other choice than to output $n-1$. Thus we have, for $n>0$

$\theta \star n \bullet T\kappa \bullet \kappa_0 \bullet \pi \succ \kappa \star (n-1) \bullet \xi \bullet \eta \bullet \pi$. We have shown :

Theorem. Any proof of $\forall x[\forall y(Xy \to Xsy), X0 \to Xx]^{\text{int}}$ gives a λ-term which computes the predecessor function.

The simplest strategy for $\exists$ is to choose successively $n-1, n-2, \ldots, 0$.

The simplest term $\theta = \lambda m \lambda f \lambda a(m\lambda g \lambda n \lambda y((g)(s)n)(f)ny)00a$ follows this strategy. It is obtained by proving

$f : \forall y(\text{int}(y), Xy \to Xsy) \vdash \lambda g \lambda n \lambda y((g)(s)n)(f)ny : F(x) \to F(sx)$

with $F(x) \equiv \forall y[\text{int}(y), Xy \to X(x+y)]$.

Remark. The formula $\Phi \equiv \forall x[\forall y(Xy \to Xsy), X0 \to Xx]$ cannot be realized, although the game and the winning strategies are the same as for Φ^{int}.
The reason is that the integer n, chosen by $\forall$, appears in the processes only as an index of a κ-instruction. It cannot be compared with 0.
The formula $\Phi' \equiv \forall x[\text{int}(x), \forall y(Xy \to Xsy), X0 \to Xx]$
is (trivially) realized by I which checks if $n = 0$.

By proving Φ^{int}, we compute, in fact, a sequence of integers from $n-1$ to 0.
We can use the following simpler formula if we only want to compute the predecessor : $\Phi_0 \equiv \forall x(\forall y\, Xsy, X0 \to Xx)$.
We note that Φ_0^{int} is provable in Analysis.
Theorem. If $\theta \Vdash \Phi_0^{\text{int}}$, then $\theta \star n \,.\, T\kappa \,.\, a \,.\, \pi \succ \kappa \star (n-1) \,.\, \pi$ for $n > 0$.
Same proof as before.

This can be generalized to compute other functions.

Computing quotient

Consider, for example the following formula Φ :

$\forall x[\forall y\, X7y, \forall y\, X(7y{+}1), \ldots, \forall y\, X(7y{+}6) \to Xx]$

Φ^{int} is provable in Analysis. Exactly the same method shows :

If $\theta \Vdash \Phi^{\text{int}}$, then θ computes the quotient and the remainder by 7 ;

i.e. $\theta \star \underline{n} \centerdot T\kappa_0 \centerdot \ldots T\kappa_6 \centerdot \pi \succ \kappa_r \star \underline{q} \centerdot \pi$ where $n = 7q + r, 0 \leq r \leq 6$.

We can generalize a bit more, with the following useful trick :

Let $a, b \in \mathbb{N}$ and X be a truth value.

Define the predicate $a = b \mapsto X$ as X if $a = b$ and $\top$ if $a \neq b$.

Theorem (trivial). $\lambda x \lambda y\, yx \Vdash \forall x \forall y \forall X\{(x{=}y \mapsto X) \to (x{=}y \to X)\}$

and $\lambda x\, xI \Vdash \forall x \forall y \forall X\{(x{=}y \to X) \to (x{=}y \mapsto X)\}$.

Thus, we can use $x{=}y \mapsto X$ instead of $x{=}y \to X$

at the cost of a little more code.

Computing logarithm

Now consider the formula Φ :

$\forall x\{\forall y \forall z \forall u[2^y{=}z{+}u{+}1 \to X(2^y+z)], X0 \to Xx\}$

which can be read as $\forall x\{\forall y \forall z[z<2^y \to X(2^y+z)], X0 \to Xx\}$.

Φ says that each integer has a logarithm, and Φ^{int} is provable in Analysis.

With the theorem above, any $\theta \Vdash \Phi^{\text{int}}$ is easily transformed into $\eta \Vdash \Psi^{\text{int}}$ with

$\Psi \equiv \forall x\{\forall y \forall z \forall u[2^y{=}z{+}u{+}1 \mapsto X(2^y+z)], X0 \to Xx\}$

Consider a play with Φ (resp. Ψ). The player $\forall$ chooses n and puts

Xn in $\mathscr{A}$, $X0$ and Φ' (resp. Ψ') in $\mathscr{U}$. $\exists$ cannot choose $X0$.

Thus, $\exists$ chooses y, z, u such that $n = 2^y{+}z$.

In the case of Φ, he has no other obligation. But, in the case of Ψ, he must satisfy $2^y{=}z{+}u{+}1$, otherwise he gets the formula $\top$ which is forbidden. Thus

Theorem. If $\eta \Vdash \Psi^{\text{int}}$, then η computes the logarithm ;

i.e. $\eta \star \underline{n} \cdot T\kappa \cdot \pi \succ \kappa \star \underline{p} \cdot \underline{q} \cdot \underline{r} \cdot \pi'$ with $n = 2^p + q$ and $2^p = q{+}r{+}1$.

Well founded recursive relations

Let $f : \mathbb{N}^2 \to \mathbb{N}$ be recursive. The predicate $f(x,y) = 1$ is well founded iff the formula $\forall X \forall z \{\forall x [\forall y (f(x,y) = 1 \to Xy) \to Xx] \to Xz\}$ is true in $\mathbb{N}$.

We show that, in this case, this formula is even *realized*.

Theorem. If the predicate $f(x,y) = 1$ is well founded, then $\mathsf{Y} \Vdash \forall X \forall z \{\forall x [\forall y (f(x,y) = 1 \mapsto Xy) \to Xx] \to Xz\}$.

Let $t \Vdash \forall x [\forall y (f(x,y) = 1 \mapsto Xy) \to Xx]$ and $n \in \mathbb{N}$; we show by induction on n, following the well founded predicate " $f(x,y) = 1$ ", that $\mathsf{Y}t \Vdash Xn$.

Since $\mathsf{Y}t \star \pi \succ t \star \mathsf{Y}t \centerdot \pi$, it suffices to show that $\mathsf{Y}t \Vdash \forall y (f(n,y) = 1 \mapsto Xy)$

i.e. $\mathsf{Y}t \Vdash f(n,p) = 1 \mapsto Xp$. This is trivial if $f(n,p) \neq 1$

and this follows from the induction hypothesis if $f(n,p) = 1$.

Thus, if $\pi \in \|Xn\|$, we have $t \star \mathsf{Y}t \centerdot \pi \in \perp\!\!\!\perp$ and therefore $\mathsf{Y} \star t \centerdot \pi \in \perp\!\!\!\perp$. QED

This shows that a recursive well founded predicate is also well founded *in every realisability model*.

True Π^1_1 formulas

But formulas provable in Analysis are not the only realized formulas.

Indeed, we have the remarkable property :

Theorem. If Φ is a true Π^1_1 formula, then Φ^{int} is realized.

This shows, in particular, that the integers of the realizability models are *elementary equivalent* to standard integers. It is not possible to show the independence of some *arithmetical* (and even Π^1_1) true formula by means of realizability models.

This leaves the possibility open for Σ^1_1 (or more complicated) true formulas, a case which is inaccessible to forcing methods, because of the Shoenfield theorem.

Sketch of proof.

Let Φ be a given Π^1_1 formula. We have associated with Φ a game such that

Φ is true iff the “ trivial ” strategy for $\exists$ is winning.

The trivial strategy is to always play the first allowed move not already played.

True Π^1_1 formulas (cont.)

Now let $f(x,y)=1$ be the recursive predicate which says that

x,y are successive positions chosen by $\forall$ such that, between them,

$\exists$ has applied the trivial strategy.

It is clear that this strategy is winning iff the predicate $f(x,y)=1$ is well founded (each play is finite, which means that every branch is finite).

Now, we have shown page 99 that this predicate is well founded iff

$\mathsf{Y} \Vdash \forall X\{\forall x[\forall y(f(x,y)=1 \mapsto Xy) \to Xx] \to \forall x\, Xx\}$.

But we have just proved that : "$f(x,y)=1$ is well founded" $\to \Phi$.

Let θ be a proof-like term associated with this proof. Then $\theta\mathsf{Y} \Vdash \Phi$. QED

Zermelo-Fraenkel set theory

A first order theory. Its axioms can be classified in three groups :

1. Equality, extensionality, foundation.
2. Union, power set, substitution, infinity.
3. Choice ; possibly other axioms such as CH, GCH, large cardinals.

We can realize the first two groups by λ_c-terms,
i.e. no new instruction is necessary besides cc.
Curiously, equality and extensionality are the most difficult ones. For example,
the first axiom of equality $\forall x(x = x)$ is realized by a λ-term τ
with the reduction rule : $\tau \star t \,.\, \pi \succ t \star \tau \,.\, \tau \,.\, \pi$ (fixed point of $\lambda x \lambda f\, f x x$).
Therefore, we need to consider first a theory with a strong membership relation ε,
without extensionality ; in some sense, $\in$ is defined by means of ε.

ZF$_\varepsilon$ set theory

Three binary symbols $\in, \subset$ and ε (strong membership) ; $x = y$ is $x \subset y \wedge y \subset x$.

• "Definition" of $\in$ and $\subset$:

$\forall x \forall y [x \in y \leftrightarrow (\exists z \varepsilon y)\, x = z]$; $\forall x \forall y [x \subset y \leftrightarrow (\forall z \varepsilon x)\, z \in y]$.

• Foundation : $\forall a [(\forall x \varepsilon a) F(x) \to F(a)] \to \forall a F(a)$ (for every formula F).

• Comprehension : $\forall a \exists b \forall x [x \varepsilon b \leftrightarrow (x \varepsilon a \wedge F(x))]$ (")

• Pair : $\forall a \forall b \exists x [a \varepsilon x \wedge b \varepsilon x]$

• Union : $\forall a \exists b (\forall x \varepsilon a)(\forall y \varepsilon x)\, y \varepsilon b$.

• Power set : $\forall a \exists b \forall x (\exists y \varepsilon b) \forall z (z \varepsilon y \leftrightarrow (z \varepsilon a \wedge F(z, x)))$ (")

• Collection : $\forall a \exists b (\forall x \varepsilon a)[\exists y F(x, y) \to (\exists y \varepsilon b) F(x, y)]$ (")

• Infinity : $\forall a \exists b \{a \varepsilon b \wedge (\forall x \varepsilon b)[\exists y F(x, y) \to (\exists y \varepsilon b) F(x, y)]\}$ (")

This theory is a *conservative extension* of ZF :

1. If $ZF_\varepsilon \vdash F$ (formula of ZF), then $ZF \vdash F$: simply replace ε by $\in$ in ZF_ε.
2. We must show that each axiom of ZF is a consequence of ZF_ε.

ZF$_\varepsilon$ set theory (cont.)

Example. $ZF_\varepsilon \vdash a \subset a$ (and thus $a = a$).
By foundation, assume $\forall x(x \varepsilon a \to x \subset x)$; this gives $\forall x(x \varepsilon a \to x = x)$, thus
$\forall x[x \varepsilon a \to (\exists y \varepsilon a)\, x = y]$, i.e. $\forall x(x \varepsilon a \to x \in a)$, and therefore $a \subset a$.

Now, we define *realizability models for ZF$_\varepsilon$*, which will therefore be also realizability models for ZF. We only need to define $\|F\|$ for atomic formulas F.
Of course, we start with a model of ZF, and we take as atomic formulas :
$a \not\varepsilon b$, $a \notin b$ and $a \subset a$. Then define : $\|a \not\varepsilon b\| = \{\pi \in \Pi;\ (a, \pi) \in b\}$.
We check that all the axioms of ZF$_\varepsilon$, except the first, are realized, without knowing the precise definition of $\|a \notin b\|$, $\|a \subset b\|$, simply because they are defined in ZF.
Foundation. $\mathsf{Y} \Vdash \forall a[\forall x(F(x) \to x \not\varepsilon a) \to \neg F(a)] \to \forall a \neg F(a)$.
This explains why we find $\mathsf{Y} \lambda x \lambda f\, f x x \Vdash \forall x(x = x)$.

ZF$_\varepsilon$ set theory (cont.)

Comprehension. For every set a and every formula $F(x)$, set :
$b = \{(x, t \bullet \pi);\ (x,\pi) \in a,\ t \Vdash F(x)\}$. We easily get $\|x \not\varepsilon b\| = \|F(x) \to x \not\varepsilon a\|$. It follows that $(I, I) \Vdash \forall x[x \not\varepsilon b \leftrightarrow (F(x) \to x \not\varepsilon a)]$.
Other axioms of ZF$_\varepsilon$ are realized in the same way. For example :
Collection. Let a be a set, $Cl(a)$ its transitive closure and $F(x, y)$ a formula.
We set $b = \bigcup\{\Phi(x, t) \times Cl(a);\ x \in Cl(a),\ t \in \Lambda_c\}$ with
$\Phi(x, t) = \{y$ of minimum rank ; $t \Vdash F(x, y)\}$, or $\emptyset$ if there is no such y.
We show that $\|\forall y(F(x, y) \to x \not\varepsilon a)\| \subset \|\forall y(F(x, y) \to y \not\varepsilon b)\|$. Indeed :
suppose $t \Vdash F(x, y)$, $(x,\pi) \in a$. Then $x,\pi \in Cl(a)$, and therefore :
$(y',\pi) \in b$ for some $y' \in \Phi(x, t)$; it follows that $t \Vdash F(x, y')$ and $\pi \in \|y' \not\varepsilon b\|$. Therefore $t \bullet \pi \in \|\forall y(F(x, y) \to y \not\varepsilon b)\|$.
We have proved that $I \Vdash \forall y(F(x, y) \to y \not\varepsilon b) \to \forall y(F(x, y) \to x \not\varepsilon a)$.

ZF$_\varepsilon$ set theory (cont.)

We must now realize the first axioms of ZF$_\varepsilon$ and therefore define the truth values of the atomic formulas : $\|a \notin b\|$, $\|a \subset b\|$, where a, b vary in *a given model of ZFC*.

It would be nice to have :

$\|a \notin b\| = \|\forall z(z \subset a, a \subset z \to z \not\varepsilon b)\|$ and $\|a \subset b\| = \|\forall z(z \notin b \to z \not\varepsilon a)\|$

because we should deduce immediately that I realizes the axioms we need.

Now $\|c \not\varepsilon a\| = \emptyset$ if $rk(a) \le rk(c)$. Thus, the above equations may be written as :

$\|a \notin b\| = \bigcup_{rk(c)<rk(b)} \|(c \subset a, a \subset c \to c \not\varepsilon b)\|$

$\|a \subset b\| = \bigcup_{rk(c)<rk(a)} \|(c \notin b \to c \not\varepsilon a)\|$ i.e.

$\|a \notin b\| = \bigcup_{rk(c)<rk(b)} \Phi(a, b, c, \|c \subset a\|, \|a \subset c\|)$

$\|a \subset b\| = \bigcup_{rk(c)<rk(a)} \Psi(a, b, c, \|c \subset a\|, \|a \subset c\|)$

where Φ, Ψ are functionals defined in ZF.

We simply observe now that this is a correct inductive definition on the ordered pair of ordinals : $(rk(a) \cup rk(b), rk(a) \cap rk(b))$.

ZF$_\varepsilon$ set theory (cont.)

Remark. It is also possible to *define* the relations $x \not\varepsilon y$, $x \subset y$ by formulas with the only symbol $\not\varepsilon$ and then to *prove* the first axioms of ZF$_\varepsilon$ from the others axioms.
We cannot use induction to define these relations, because ordinals are not definable in ZF$_\varepsilon$. But we can use *coinduction.*
Anyway, this method gives complicated λ-terms for the first axioms of ZF$_\varepsilon$
so that we prefer the above method.

Remark. The definition of $t \Vdash x \not\varepsilon y$ and $t \Vdash x \subset y$ is very similar to the defintion of forcing. In fact, the generic models of set theory, which are defined in forcing, are particular cases of realizability models.
Thus, the theory presented here gives completely new models of set theory.
The fact that forcing is a case of realizability, is used to find programs associated with the *axiom of choice* and the *continuum hypothesis*. We build a model by combining both methods ; we call this *iterated realizability* by analogy with *iterated forcing*.

The full axiom of choice

We get a program for the axiom of dependent choice in the same way as in Analysis.
The problem for the *full axiom of choice* is more difficult. It has been solved recently (not yet published). As a bonus, we get also the *continuum hypothesis*.
The proof is too long to be given here ; the result is as follows :
we need two new instructions χ and χ' which appear inside
two very complex λ-terms, together with cc and the clock (or the signature).
The behaviour of these programs is not yet understood.
These new instructions χ, χ' work on the *bottom of the stack*.
Their reduction rules is as follows :

$$\chi \star t \cdot \tau \cdot t_1 \ldots t_n \cdot \pi_0 \succ t \star t_1 \ldots t_n \cdot \tau \cdot \pi_0$$
$$\chi' \star t \cdot t_1 \ldots t_n \cdot \tau \cdot \pi_0 \succ t \star \tau \cdot t_1 \ldots t_n \cdot \pi_0$$

where π_0, as before, is a marker for the bottom of the stack.

The axiom of choice (cont.)

In order to understand the behaviour of these new instructions, we consider processes of the form $<t \star \pi, \tau>$ where τ is a closed term.

The execution rules are as follows :

$<tu \star \pi, \tau> \succ <t \star u . \pi, \alpha_0 \tau>$ $\qquad$ $<\lambda x\, t \star u . \pi, \tau> \succ <t[u/x] \star \pi, \alpha_1 \tau>$

$<\mathsf{cc} \star t . \pi, \tau> \succ <t \star \mathsf{k}_\pi . \pi, \alpha_2 \tau>$ $\qquad$ $<\mathsf{k}_\pi \star t . \rho, \tau> \succ <t \star \pi, \alpha_3 \tau>$

$<\chi \star t . \tau . t_1 \ldots t_n . \pi_0, \tau'> \succ <t \star t_1 \ldots t_n . \tau' . \pi_0, \tau>$

$<\chi' \star t . t_1 \ldots t_n . \tau' . \pi_0, \tau> \succ <t \star \tau . t_1 \ldots t_n . \pi_0, \tau'>$

The α_i are fixed closed terms, which we shall not write explicitly here.

In fact, we get a parallel execution ; χ and χ' are communication instructions.

Conclusion

The conclusion is that we can translate *every mathematical proof*
into a program. We can execute this program in a lazy λ-calculus machine
extended with only four new instructions : cc, σ (or $\hbar$), χ and χ'.
This machine can be implemented rather easily.

The challenge, now, is to understand all these programs,
first of all, the ones we obtained for the axioms of ZFC.
It is very plausible that we shall find, in this way, programs analogous
to the core of an operating system like Unix.
This would give a method to implement such a core on a very firm basis.

References

1. **S. Berardi, M. Bezem, T. Coquand** *On the computational content of the axiom of choice.* J. Symb. Log. 63, pp. 600-622, 1998.
2. **U. Berger, P. Oliva** *Modified bar recursion and classical dependent choice.* Preprint.
3. **T. Coquand.** *A semantics of evidence for classical arithmetic.*
J. Symb. Log. 60, pp. 325-337, 1995.
4. **V. Danos & L. Regnier.** *How abstract machines implement head linear reduction.*
Higher Order and Symbolic Computation (to appear).
5. **T. Griffin.** *A formulæ-as-type notion of control.*
Conf. Record of the 17th A.C.M. Symp. on Principles of Progr. Languages, 1990.
6. **G. Kreisel.** *On the interpretation of non-finitist proofs I-II.*
J. Symb. Log. 16, p. 248-267, 1951. - J. Symb. Log. 17, p. 43-58, 1952.

References (cont.)

7. **J.-L. Krivine** *Typed lambda-calculus in classical Zermelo-Fraenkel set theory.*
Arch. Math. Log. 40, 3, pp. 189-205, 2001.
8. **J.-L. Krivine** *Dependent choices, 'quote' and the clock.*
Th. Comp. Sc. 308, pp. 259-276, 2003.
9. **J.-L. Krivine** *Realizability in classical logic.*
To appear in Panoramas et Synthèses. Société mathématique de France.

Pdf files at http://www.pps.jussieu.fr/˜krivine