Realizing the axiom of dependent choice

Jean-Louis Krivine

PPS Group, University Paris 7, CNRS

krivine@pps.jussieu.fr

Edinburgh, March 26, 2003

The extended Curry-Howard correspondence

We want to get programs from *usual* mathematical proofs
and also *understand* these programs.
A possible framework for real mathematics is :

Second order classical logic with the axiom of dependent choice.

We know how to get ordinary λ-terms from proofs
in second order *intuitionistic logic* with the only logical symbols $\forall, \to$.
Therefore, we have to interpret two axioms : the law of Peirce
and the dependent choice axiom.
The method is : extend the λ-calculus with new instructions
but restrict to weak head reduction.
This works also for classical ZF set theory with dependent choice
(not considered in this talk).

An advertising page

Advantages of this method

- We get a pleasant mathematical theory (essential).
- We get a *non-trivial* extension of forcing and a whole new class of models of ZF set theory (not done in this talk).
- We interpret usual concepts of programming such as pointers, imperative call by value, system clock, system boot, ... For instance, in this talk, we use the *system clock* in order to interpret the countable choice axiom.
- This framework is completely open : we may add new *typed* instructions in order to interpret other independent formulas
(a measurable cardinal, for example).

Drawbacks

- None.

Let us now explain the framework.

The λ_c-calculus

Λ_c (resp. Λ_c^0) is the set of arbitrary (resp. closed) *λ_c-terms*.
Π is the set of *stacks*. They are built following these rules :

1. Any variable x, and the constant cc are λ_c-terms.
2. If t, u are λ_c-terms and x is a variable, then $(t)u$ and $\lambda x\, t$ are λ_c-terms.
3. If π is a stack, the constant k_π is a λ_c-term (called a *continuation*).

A stack is a sequence $\pi = t_1. \ldots .t_n.\rho$ of closed λ_c-terms t_i
ended with a *stack constant* ρ (the *bottom* of the stack) ;
$t.\pi$ denotes the stack obtained by *pushing* t on the *top* of π.

The constant cc is an example of instruction.
We may add other instructions and give, for each of them,
the corresponding *rule of reduction*.

Execution of processes

A *process* is a couple : $t \star \pi$ with $t \in \Lambda_c^0$, $\pi \in \Pi$.
A process can be performed, a λ_c-term alone cannot.
t is called the *head* of the process $t \star \pi$.
At each moment, the head is the active part of the process.
The rules of reduction for processes are (with $\pi, \pi' \in \Pi$ and $t, u \in \Lambda_c^0$) :

$tu \star \pi \succ t \star u.\pi$ *(push)* $\quad$ $\mathsf{cc} \star t.\pi \succ t \star k_\pi.\pi$ *(store the stack)*

$\lambda x\, t \star u.\pi \succ t[u/x] \star \pi$ *(pop)* $\quad$ $k_\pi \star t.\pi' \succ t \star \pi$ *(restore the stack)*

For each new instruction χ, we give a rule of reduction for χ.
For instance, if χ is a *stop* instruction, the rule is :

$\chi \star \pi \succ t \star \rho$ for no process $t \star \rho$.

In the following, we use a *'quote'* instruction χ with the rule :

$\chi \star t.\pi \succ t \star n_t.\pi$ $\quad$ n_t is a Church integer

which is the number of the term t in a fixed recursive enumeration of Λ_c^0.

Typing in classical 2nd order logic

The only logical symbols are $\to$, $\forall$ and function symbols on individuals.
$\bot$ is defined as $\forall X\, X$; $A \wedge B$ as $\forall X\{(A, B \to X) \to X\}$;
$\exists x\, F[x]$ as $\forall X\{\forall x(F[x] \to X) \to X\}$; $x = y$ as $\forall X(Xx \to Xy)$; etc.
Let Γ denote $x_1 : A_1, \ldots, x_n : A_n$ *(a context)*. Typing rules are :

1. $\Gamma \vdash x_i : A_i$ $(1 \leq i \leq n)$
2. $\Gamma \vdash t : A \to B$, $\Gamma \vdash u : A \;\Rightarrow\; \Gamma \vdash tu : B$.
3. $\Gamma, x : A \vdash t : B \;\Rightarrow\; \Gamma \vdash \lambda x\, t : A \to B$.
4. $\Gamma \vdash t : (A \to B) \to A \;\Rightarrow\; \Gamma \vdash \mathsf{cc}\, t : A$.
5. $\Gamma \vdash t : A \;\Rightarrow\; \Gamma \vdash t : \forall x\, A$ (resp. $\forall X\, A$) if x (resp. X) is not free in Γ.
6. $\Gamma \vdash t : \forall x\, A \Rightarrow \Gamma \vdash t : A[\tau/x]$ for every term τ.
7. $\Gamma \vdash t : \forall X\, A \Rightarrow \Gamma \vdash t : A[\Phi(x_1, \ldots, x_n)/Xx_1 \ldots x_n]$ for each formula Φ.

The comprehension scheme for second order logic is included in 7,
the law of Peirce in 4.

Realizability

The notion of *model* is the usual one, for a second order language with only function symbols on individuals.

Simply the set of truth values is now $\mathcal{P}(\Pi)$ instead of $\{0, 1\}$.

Thus, a model $\mathcal{M}$ is a set M of individuals ($M = \mathbb{N}$ in this talk), together with an interpretation $f_{\mathcal{M}} : M^k \to M$ of each k-ary function symbol f.

2nd order variables of arity k are valued in $\mathcal{P}(\Pi)^{M^k} = \mathcal{P}(\Pi \times M^k)$.

To define realizability, let $\perp\!\!\!\perp$ be a fixed *saturated* set of processes, i.e. :

$$t \star \pi \in \perp\!\!\!\perp,\ t' \star \pi' \succ t \star \pi \Rightarrow t' \star \pi' \in \perp\!\!\!\perp$$

Let $t \in \Lambda_c^0$ and $\mathsf{P} \subset \Pi$ be a truth value.

We say that $t \Vdash \mathsf{P}$ (t realizes P) iff $(\forall \pi \in \mathsf{P})\, t \star \pi \in \perp\!\!\!\perp$.

We can now define the truth value and the realizability for any formula.

Realizability (cont.)

Let A be a closed 2nd order formula with parameters in M and $\mathcal{P}(\Pi \times M^k)$.
Its truth value, defined below, is a subset of Π denoted by $\|A\|$.
Thus, we say that $t \Vdash A$ (t *realizes* A) iff $(\forall \pi \in \|A\|)\, t \star \pi \in \perp\!\!\!\perp$.

Definition of $\|A\|$, by induction on A :
A atomic i.e. $R(a_1, \ldots, a_k)$, $R \in \mathcal{P}(\Pi)^{M^k}$, $a_i \in M$: evident.

$\|A \to B\| = \{t.\pi\ ;\ t \Vdash A,\ \pi \in \|B\|\}$; $\qquad \|\forall x\, A\| = \bigcup_{a \in M} \|A[a/x]\|$

$\|\forall X\, A\| = \bigcup \{\|A[R/X]\|;\ R \in \mathcal{P}(\Pi \times M^k)\}$

Thus $\qquad t \Vdash \forall x\, A \Leftrightarrow (\forall a \in M)\ t \Vdash A[a/x]$
and $\qquad t \Vdash \forall X\, A \Leftrightarrow (\forall R \in \mathcal{P}(\Pi \times M^k))\ t \Vdash A[R/X]$.
The set $\{t \in \Lambda_c^0; t \Vdash A\}$ is denoted by $|A|$.

Proofs and terms

Realizability is a fundamental tool because it is compatible with classical second order deduction :

Adequation lemma.

If $x_1 : \Phi_1, \ldots, x_n : \Phi_n \vdash t : \Phi$ *and if* $t_i \Vdash \Phi_i \, (1 \leq i \leq n)$

then $t[t_1/x_1, \ldots, t_n/x_n] \Vdash \Phi$.

This property is useful for two reasons :

1. To solve the *specification problem* for a given theorem Φ, i.e. to understand the common behaviour of λ_c-terms extracted from proofs of Φ.
A very interesting but difficult problem. We can use the adequation lemma and study the behaviour of λ_c-terms which *realize* Φ.

2. To extend the λ_c-calculus with new instructions which will realize given axioms or independent formulas (like AC, CH, etc).
In this talk, we will deal with the axiom of countable choice.

Definitions and remarks

A *proof-like term* is a closed λ_c-term which contains no continuation.
We say that *the formula Φ is realized* if $\tau \Vdash \Phi$ for a proof-like term τ. Thus :

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

The truth values $\emptyset$ and Π are denoted by $\top$ and $\bot$.
There is no other iff $⫫ = \emptyset$. It is a degenerate case
in which we get the usual two-valued notion of model.
If $⫫ \neq \emptyset$, then $\tau \Vdash \bot$ for some $\tau \in \Lambda_c^0$: take $t \star \pi \in ⫫$ and $\tau = k_\pi t$.
The choice of $⫫$ is generally done according to the theorem Φ for which we want to solve the specification problem. Let us take a trivial example :
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 $⫫ = \{\mathsf{p}\,;\mathsf{p} \succ t \star \pi\}$ and $\|X\| = \{\pi\}$. QED
Example : $\theta = \lambda x\, \mathsf{cc}\lambda k\, kx$.

Integers

The language has a function symbol for each recursive function.
The set of individuals is $\mathbb{N}$. Let $Int(x) \equiv \forall X[\forall y(Xy \to Xsy), X0 \to Xx]$.
Unfortunately, the recurrence axiom $\forall x\, Int(x)$ is not realized. But

(*) $\quad \forall x_1 \ldots \forall x_k\{Int(x_1), \ldots, Int(x_k) \to Int(f(x_1, \ldots, x_k))\}$

is realized for each function symbol f.
Therefore, the symbol f keeps its intended meaning in the new model.
Proof. Let ϕ be a λ-term which computes f (unary) and T the *storage operator* defined in the next slide. Then $T\phi \Vdash \forall x\{Int(x) \to Int(fx)\}$. QED
Now, if we prove a formula Φ using the recurrence axiom, we know that the *restricted formula* Φ^{Int} is provable without it, using formulas (*). Therefore :
If Φ is provable with realized axioms and the recurrence axiom,
then Φ^{Int} is realized.

Imperative call-by-value

Remark. $s^n 0 \Vdash Int(n)$ if s is a λ-term for the successor.
Define $T = \lambda f \lambda n(n)\lambda g\, g \circ s.f.0$ (*storage operator* [4]).
Theorem. *If* $(\forall \pi \in \|X\|)\ f \star s^n 0.\pi \in \perp\!\!\!\perp$ *then* $Tf \Vdash Int(n) \to X$.
Proof. Let $\|Pj\| = \{s^{n-j}0.\pi;\ \pi \in \|X\|\}$ for $0 \le j \le n$;
$\|Pj\| = \emptyset$ for $j > n$. Then $\lambda g\, g \circ s \Vdash \forall x(Px \to Psx)$ and $f \Vdash P0$.
If $\nu \Vdash Int(n)$ and $\pi \in \|X\|$, then $0.\pi \in \|Pn\|$; thus $\nu \star \lambda g\, g \circ s.f.0.\pi \in \perp\!\!\!\perp$
which gives $Tf \star \nu.\pi \in \perp\!\!\!\perp$. QED

Let $\nu \in \Lambda_c^0$, $\nu \Vdash Int(n)$; i.e. ν "behaves like" the integer n.
In the λ_c-term $f\nu$ this data is *called by name* by the program f.
In the λ_c-term $Tf\nu$ the same data is *called by value* by f.

I name this *imperative* call-by-value, to avoid confusion with
the well-known notion of (functional) call-by-value.
It is only defined for data types (booleans, integers, trees, . . .)

The countable axiom of choice

It is the following axiom scheme (for any formula F) :

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

In order to realize this formula, let $n \mapsto \pi_n$ be a fixed surjection of $\mathbb{N}$ onto Π.

We define a *new instruction* χ by the reduction rule :

$$\chi \star \phi.\pi \succ \phi \star s^n 0.\pi$$

for every $\phi \in \Lambda_c^0$ and $\pi \in \Pi$; n is any integer such that $\pi_n = \pi$

The simplest way (at first sight) to implement this is to choose a *recursive bijection* for the function $n \mapsto \pi_n$.

We shall examine later other possibilities.

The intuitionistic countable choice axiom

We now show that χ almost realizes countable choice axiom :

Theorem. There exists $U : \mathbb{N}^3 \to \mathcal{P}(\Pi)$ such that
$\chi \Vdash \forall x\{\forall n(Int[n] \to F[x, U(x,n,y)/Xy]) \to \forall X\, F[x, X]\}$.

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

The intuitionistic countable choice axiom (cont.)

We have shown that the following axiom scheme is realized (by $\lambda x\, x\chi$) :

$\exists U \forall x\{\forall n(Int[n] \to F[x, U(x,n,y)/Xy]) \to \forall X\, F[x, X]\}$

It may be called the intuitionistic countable choice axiom.

Indeed, the predicate U has been *explicitly* given.

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 :

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

Interpretation

The following variant χ' of χ also realizes the intuitionistic countable choice.
Let $n \mapsto t_n$ be a fixed surjection of $\mathbb{N}$ onto Λ_c^0. The rule of reduction is :

$$\chi' \star t.\pi \succ t \star s^n 0.\pi$$

where n is any integer such that $t_n = t$.

The surjections $n \mapsto \pi_n$ or $n \mapsto t_n$ are arbitrary. If, for example, $n \mapsto t_n$ is a recursive bijection, we may consider $s^n 0$ as an index of t.

The instruction χ' is then similar to the *'quote'* of LISP.

Another possibility is that the integer n is given by an *oracle*, in other words, in an *interactive way*. The only condition is that $n \mapsto t_n$ must be functional, i.e. the integers given for different terms must be different.

A very simple implementation of such an oracle is a *clock* :
just increment a counter at each reduction step
and give its value when asked, i.e. when χ arrives in head position.

A simple example

Theorem. Let $\theta[\chi]$ be obtained by a proof of $\exists x[Int(x) \wedge f(x) = 0]$ in $\mathrm{PA}_2 +$ Dep. Ch., with f recursive. Let κ be a stop instruction. Then

$$\theta \star V\kappa.\pi \succ \kappa s^n 0 \star \pi \text{ with } f(n) = 0.$$

$V\kappa$ is $T\lambda x\lambda y(y)(\kappa)x$, T is the storage operator.

Proof. We have $\theta \Vdash \forall x[Int(x), f(x) = 0 \to X] \to X$. Now take $\|X\| = \{\pi\}$ and $⫫ = \{\mathsf{p}\,;\mathsf{p} \succ \kappa s^n 0 \star \pi \text{ with } f(n) = 0\}$.
We simply have to show that $V\kappa \Vdash \forall x[Int(x), f(x) = 0 \to X]$
i.e. by the call-by-value theorem, that $t \star \kappa s^n 0.\pi \in ⫫$
if $t \Vdash \forall X(X f(n) \to X0)$ (which is $f(n) = 0$).
If $f(n) = 0$, then $t \Vdash \forall X(X \to X)$ and $\kappa s^n 0 \star \pi \in ⫫$. Thus $t \star \kappa s^n 0.\pi \in ⫫$.
If $f(n) \neq 0$, then $t \Vdash \top \to \bot$, hence $t \star \kappa s^n 0.\pi \in ⫫$. QED

Remark. κ is clearly a *pointer to an integer*. In the program, we wrote $V\kappa$, because we want it to point to a *computed* integer.
It is the intuitive meaning of *imperative call-by-value*.

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. **J.-L. Krivine** *Dependent choices, 'quote' and the clock.* To appear in Th. Comp. Sc.

4. **J.-L. Krivine** *A general storage theorem for integers in call-by-name λ-calculus.* Th. Comp. Sc. 129, pp. 79-94 (1994).

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