HAL
open science

# Dependent choice, 'quote' and the clock

Jean-Louis Krivine

## To cite this version:

Jean-Louis Krivine. Dependent choice, 'quote' and the clock. Theoretical Computer Science, 2003, 308 (1-3), pp.259-276. 10.1016/S0304-3975(02)00776-4 . hal-00154478

**HAL Id: hal-00154478**

**https://hal.science/hal-00154478v1**

Submitted on 13 Jun 2007

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Dependent choice, 'quote' and the clock

Jean-Louis Krivine

University Paris VII, C.N.R.S.

Equipe Preuves, programmes et systèmes

2 Place Jussieu 75251 Paris cedex 05, France

e-mail krivine@pps.jussieu.fr

September 29, 2002

## Introduction

When using the Curry-Howard correspondence in order to obtain executable programs from mathematical proofs, we are faced with a difficult problem : to interpret each axiom of our axiom system for mathematics (which may be, for example, second order classical logic, or classical set theory) as an instruction of our programming language. This problem has been solved for the axiom of excluded middle by T. Griffin[5], who found that it can be interpreted by means of control instructions like `call-with-current-continuation` in SCHEME, `catch` and `throw` in LISP or `try ... with ...` in CAML. The solution for the comprehension axiom scheme for second order logic was essentially given by G. Takeuti[14], who gave a formulation of this scheme by means of an elimination rule of the second order universal quantifier, and J.Y. Girard who showed in [4] that this rule can be interpreted by the identity instruction $\lambda x\, x$.
In [11], this problem is solved for the axioms of classical Zermelo-Fraenkel set theory, with the axiom of foundation, *but without the axiom of choice.*

Indeed the problem remains for the axiom of choice, which is essential in many proofs. In most cases, we know how to avoid it, but at the price of much longer proofs. The weaker *axiom of countable choice* is fundamental in calculus, measure theory, ... (this very important part of mathematics which can be formalized in second order classical logic). It is certainly unavoidable.

In this paper, we give an interpretation of the axiom of countable choice (and even the slightly stronger *axiom of dependent choice*) in classical second order logic, as a programming instruction. Using the results of [11], the same method applies indeed in ZF set theory. Our solution is to introduce a new instruction in $\lambda$-calculus, which uses an enumeration of $\lambda$-terms. We give two different computational interpretations and in fact implementations, for this instruction : the first is similar to the *quote instruction* of LISP and the second is given in terms of *a clock*.
Such an instruction is rather difficult to handle, because it is incompatible with $\beta$-reduction. Therefore, we must set very precisely our framework in order to be able to

give it a type.
This is done in the next section ; we use essentially the same framework as in [11]. Then we give the interpretation of the axioms of countable and dependent choice in terms of the new instruction. In the last section, we apply this method to first order arithmetical formulas which are provable in classical analysis with choice, in the spirit of the *no-counter-example interpretation* of G. Kreisel. We use the ideas about game semantics for proofs of such formulas which have been developed in [3].

Other computational interpretations of the countable axiom of choice have been given in [1], and recently in [2]. It would be interesting to understand the relation with the present paper.

I want to thank Thierry Coquand and the referee for their very pertinent observations ; and Vincent Danos for an uncountable number of helpful and stimulating discussions.

We now define our programming langage : the $\lambda_c$-calculus, an extension of the $\lambda$-calculus with a control instruction and continuations, which is suitable for classical logic.

# The $\lambda_c$-calculus

We have countably many $\lambda$-*variables* $x, y, \ldots$ and *stack constants* $\pi_0, \pi_1, \ldots$, and we define recursively and simultaneously the set $\Lambda_c$ of $\lambda_c$-*terms* and the set $\Pi$ of *stacks*. A stack is a finite sequence $t_1 \ldots t_n.\rho$ of *closed* $\lambda_c$-terms $t_1, \ldots, t_n$, ended with a stack constant $\rho$ which is called the *bottom* of the stack. The $\lambda_c$-term $t_1$ is called the *top* of the stack, $n$ is its *length*.
The $\lambda_c$-terms are built according to the following rules :

1) A $\lambda$-variable is a $\lambda_c$-term.
2) If $t, u$ are $\lambda_c$-terms $(t)u$ (also denoted by $tu$) is a $\lambda_c$-term.
3) If $t$ is a $\lambda_c$-term and $x$ a $\lambda$-variable, $\lambda x\, t$ is a $\lambda_c$-term.
4) The constant $\mathsf{cc}$ is a (closed) $\lambda_c$-term.
5) Each stack $\pi$ gives, in an injective way, a $\lambda$-constant denoted by $k_\pi$, which is a (closed) $\lambda_c$-term. It is called the *continuation* associated with $\pi$.

The set of closed $\lambda_c$-terms will be denoted by $\Lambda_c^0$. Given a stack $\pi = t_1 \ldots t_n.\rho$ and a closed $\lambda_c$-term $t$, we denote by $t.\pi$ the stack $t.t_1 \ldots t_n.\rho$ obtained by *pushing* $t$ on the *top* of $\pi$.

Given the $\lambda_c$-terms $t, u_1, \ldots, u_n$, we denote by $(t)u_1 \ldots u_n$ or even by $tu_1 \ldots u_n$ the $\lambda_c$-term $(\ldots((t)u_1)u_2 \ldots)u_n$, obtained by giving to $t$ the arguments $u_1, \ldots, u_n$.

## Execution of processes

$\Lambda_c^0 \times \Pi$ is called the set of *processes*. Thus, a process is a pair, denoted by $t \star \pi$, where $t$ is a *closed* $\lambda_c$-term and $\pi$ is a stack ; $t$ is called the *head* of the process.
A process can be performed, a $\lambda_c$-term alone cannot. At each time, the head is the active part of the process, i.e. the part which is executed. Thus, there are four rules of execution, because the head of the process may have one of the forms (2), (3), (4) or (5). The rules are the following, with $\pi$, $\pi' \in \Pi$ and $t, u \in \Lambda_c^0$ :

| | | | |
|---|---|---|---|
| $tu \star \pi \succ t \star u.\pi$ | *(push)* | $\mathsf{cc} \star t.\pi \succ t \star k_\pi.\pi$ | *(store the stack)* |
| $\lambda x\, t \star u.\pi \succ t[u/x] \star \pi$ | *(pop)* | $k_\pi \star t.\pi' \succ t \star \pi$ | *(restore the stack)* |

**Remark.** For example, when $t$ is an ordinary closed $\lambda$-term and $\pi$ is a stack constant, then the reduction of the process $t \star \pi$ is essentially the weak head reduction of $t$.

Now, let $\perp\!\!\!\perp$ be a fixed $\mathsf{cc}$-*saturated* set of processes, i.e. with the property :

If $t \star \pi \in \perp\!\!\!\perp$ and $t' \star \pi' \succ t \star \pi$ then $t' \star \pi' \in \perp\!\!\!\perp$.

A *truth value* is a subset of $\Lambda_c^0$ of the form $\mathsf{P}^\perp$ for any $\mathsf{P} \subset \Pi$, where :

$$\mathsf{P}^\perp = \{t \in \Lambda_c^0 \,;\, (\forall \pi \in \mathsf{P})\, t \star \pi \in \perp\!\!\!\perp\}.$$

The set of truth values is denoted by $\Re_{\perp\!\!\!\perp}$ or simply $\Re$ if there is no ambiguity. The *truth value of a formula*, defined below, will be the set of $\lambda_c$-terms which *realize* this formula.

## Typing in classical second order logic

Types are usual formulas of second order logic, written with the only logical symbols $\to$, $\forall$ and some fixed function symbols on individuals, taken in a set $\mathcal{L}$ (the *language*). First order (also called *individual*) variables are denoted by lower case letters $x, y, \ldots$ and second order (also called *predicate*) variables by upper case letters $X, Y, \ldots$ Each second order variable has an arity in $\mathbb{N}$. Variables of arity 0 are also called *propositional variables.*
Notations : $F_0 \to (F_1 \to \ldots \to (F_n \to G) \ldots)$ is denoted by $F_0, F_1, \ldots, F_n \to G$.
$\perp$ is defined as $\forall X\, X$ ; $\neg F$ as $F \to \perp$ ;
$F \vee G$ as $\forall X[(F \to X), (G \to X) \to X]$ ;
$F \wedge G$ as $\forall X[(F, G \to X) \to X]$ ;
$\exists X\, F[X]$ as $\forall Y\{\forall X(F[X] \to Y) \to Y\}$ ; $\exists x\, F[x]$ as $\forall Y\{\forall x(F[x] \to Y) \to Y\}$ ;
($Y$ is a propositional variable, $X$ has an arbitrary arity).
$x = y$ as $\forall X(Xx \to Xy)$.
We shall use the notation $\exists X\{F_1[X], \ldots, F_k[X]\}$ for the formula :
$\forall Y\{\forall X(F_1[X], \ldots, F_k[X] \to Y) \to Y\}$ and the same for the first order quantifier.
Let $x_1, \ldots, x_k$ be individual variables and $X$ a $k$-ary predicate variable. For any formulas $A$ and $F$ we define $A[F/Xx_1 \ldots x_k]$ by induction on $A$ :

If $X$ is not free in $A$, then $A[F/Xx_1 \ldots x_k]$ is $A$.
If $A$ is $X(t_1, \ldots, t_k)$ then $A[F/Xx_1 \ldots x_k]$ is $F[t_1/x_1, \ldots, t_k/x_k]$.
$(A \to B)[F/Xx_1 \ldots x_k]$ is $A[F/Xx_1 \ldots x_k] \to B[F/Xx_1 \ldots x_k]$.
$(\forall y\, A)[F/Xx_1 \ldots x_k]$ is $\forall y\, A[F/Xx_1 \ldots x_k]$.
$(\forall Y\, A)[F/Xx_1 \ldots x_k]$ is $\forall Y\, A[F/Xx_1 \ldots x_k]$ if $Y$ is a predicate variable $\neq X$
(as usual, $y$ and $Y$ are assumed not to be free in $F$).

If $F$ is atomic of the form $R(x_1, \ldots, x_k)$, where $R$ is either a $k$-ary second order variable or a parameter ($R \in \mathcal{P}(\Pi)^{M^k}$), we shall also write $A[R/X]$ instead of $A[F/Xx_1 \ldots x_k]$.

The deduction rules for classical second order logic are given below, together with the typing rules for $\lambda_c$-terms ; in an expression like ' $t : A$ ', $t$ is a $\lambda_c$-term and $A$ a type, that is a second order formula. Let $\Gamma$ denote $x_1 : A_1, \ldots, x_n : A_n$ *(a context).*

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 $\Gamma$.
6. $\Gamma \vdash t : \forall x\, A \Rightarrow \Gamma \vdash t : A[\tau/x]$ for every term $\tau$ of $\mathcal{L}$.
7. $\Gamma \vdash t : \forall X\, A \Rightarrow \Gamma \vdash t : A[F/Xx_1 \ldots x_k]$ for any formula $F$.

Rule 4 uses Griffin's interpretation of the law of Peirce.
Rule 7 uses the Takeuti-Girard interpretation of the comprehension scheme.

A $\lambda_c$-term is called *proof-like* if it contains no continuation. Clearly, every $\lambda_c$-term which is typed in this system is proof-like.

## Models and realizability

A *model* $\mathcal{M}$ is a set $M$ of individuals (the domain of variation of first order variables), together with an interpretation $f_{\mathcal{M}} : M^k \to M$ for each $k$-ary function symbol $f$ of $\mathcal{L}$. It is also given a set $\perp\!\!\!\perp$ of processes, which is $\mathsf{cc}$-saturated.
The domain of variation of $k$-ary second order variables is $\mathcal{P}(\Pi)^{M^k}$ ($\Pi$ is the set of stacks).
Let $A$ be a closed second order formula with parameters in $M$ (first order parameters) and in $\mathcal{P}(\Pi)^{M^k}$ (second order $k$-ary parameters). Its truth value, defined below, is denoted by $|A|$. Indeed, we have $|A| = \|A\|^{\perp}$ for a certain set of stacks $\|A\|$ (intuitively, $\|A\|$ is an interpretation of the *negation* of $A$). Now, what we must really define is $\|A\|$, which is done by induction on $A$ :

If $A$ is atomic, i.e. $R(t_1, \ldots, t_k)$, then $t_1, \ldots, t_k$ are closed terms with parameters in $M$ and $R \in \mathcal{P}(\Pi)^{M^k}$ is a second order $k$-ary parameter. Let $a_i \in M$ be the value of $t_i$ ; then we set $\|R(t_1, \ldots, t_k)\| = R(a_1, \ldots, a_k) \subset \Pi$.
The other steps of the induction are :

$\|A \to B\| = \{t.\pi \;;\; t \in |A|,\, \pi \in \|B\|\}$ ;
$\|\forall x\, A\| = \bigcup_{a \in M} \|A[a/x]\|$ ; (it follows that $|\forall x\, A| = \bigcap_{a \in M} |A[a/x]|$) ;
$\|\forall X\, A\| = \bigcup \{\|A[R/X]\|;\, R \in \mathcal{P}(\Pi)^{M^k}\}$ if $X$ is a $k$-ary predicate variable
(it follows that $|\forall X\, A| = \bigcap \{|A[R/X]|;\, R \in \mathcal{P}(\Pi)^{M^k}\}$).

We have $\|\perp\| = \Pi$, and thus $|\perp|$ is the least truth value. There is a greatest truth value, denoted by $\top$ which is $\emptyset^{\perp} = \Lambda_c^0$ (the set of all closed $\lambda_c$-terms). We can consider $\top$ as a propositional constant with $\|\top\| = \emptyset$.

**Remark.** We have $|\perp| = \emptyset \Leftrightarrow \perp\!\!\!\perp = \emptyset$.
Indeed, if $\perp\!\!\!\perp = \emptyset$, then $|\perp| = \{t \in \Lambda_c^0;\, (\forall \pi \in \Pi)\, t \star \pi \in \perp\!\!\!\perp\} = \emptyset$. Conversely, if $\perp\!\!\!\perp \neq \emptyset$, take $t \star \pi \in \perp\!\!\!\perp$. Then $k_\pi t \in |\perp|$ since $k_\pi t \star \pi' \succ t \star \pi$.
Notice that, if $\perp\!\!\!\perp = \emptyset$, the set of truth values is $\Re = \{\emptyset, \Lambda_c^0\} = \{\perp, \top\}$ and we have the ordinary notion of model.

We say that $t$ *realizes* $A$ (notation : $t \Vdash A$) if $t \in |A|$ i.e. $(\forall \pi \in \|A\|)\, t \star \pi \in \perp\!\!\!\perp$.

The next theorem says that realizability is compatible with classical deduction. It will be used repeatedly in the following.

**Theorem 1 (Adequation lemma).** *Let $A_1, \ldots, A_k$, $A$ be closed formulas such that $x_1 : A_1, \ldots, x_k : A_k \vdash t : A$ is obtained with the above rules of deduction. If $t_i \Vdash A_i$ for $1 \leq i \leq k$, then $t[t_1/x_1, \ldots, t_k/x_k] \Vdash A$.*

In particular, if $A$ is a closed formula and if we can deduce $\vdash t : A$, then $t \Vdash A$.

We state as a lemma a somewhat stronger and more complicated form of this theorem, which is suitable for an inductive proof.

**Lemma 2.**
*Let $A_1, \ldots, A_k, A$ be formulas, the free variables of which are among $y_1, \ldots, y_m, Y_1, \ldots, Y_n$. Let $b_i \in M$ $(1 \leq i \leq m)$ and $P_j \in \mathcal{P}(\Pi)^{M^{k_j}}$ $(1 \leq j \leq n)$, where $k_j$ is the arity of $Y_j$. Suppose that $x_1 : A_1, \ldots, x_k : A_k \vdash t : A$ is obtained with the above rules of deduction. If $t_i \Vdash A_i[b_1/y_1, \ldots, b_m/y_m, P_1/Y_1, \ldots, P_n/Y_n]$ for $1 \leq i \leq k$, then $t[t_1/x_1, \ldots, t_k/x_k] \Vdash A[b_1/y_1, \ldots, b_m/y_m, P_1/Y_1, \ldots, P_n/Y_n]$.*

We prove this lemma by induction on the length of the derivation of $\Gamma \vdash t : A$ ($\Gamma$ being the context $x_1 : A_1, \ldots, x_k : A_k$). We shall use the notations $t'$ for $t[t_1/x_1, \ldots, t_k/x_k]$, and $A'$ for $A[b_1/y_1, \ldots, b_m/y_m, P_1/Y_1, \ldots, P_n/Y_n]$. We consider the last rule used (the case of rule 1 is trivial) :

If it is rule 2, we have $t = uv$ and $\Gamma \vdash u : A \to B$, $v : A$. We want to show that $t' \in |B'|$, that is $u'v' \star \pi \in \perp\!\!\!\perp$ for every $\pi \in \|B'\|$. But $u'v' \star \pi \succ u' \star v'.\pi$ and the result follows since, by the induction hypothesis, we have $u' \in |A' \to B'|$ and $v' \in |A'|$ and therefore $v'.\pi \in \|A' \to B'\|$.

If it is rule 3, we have $t = \lambda x\, u$, $A = B \to C$ and $\Gamma, x : B \vdash u : C$. We want to show that $\lambda x\, u' \in |B' \to C'|$, that is $\lambda x\, u' \star \pi \in \perp\!\!\!\perp$ for any $\pi \in \|B' \to C'\|$. Now, we have $\pi = v.\varpi$ with $v \in |B'|$ and $\varpi \in \|C'\|$. By the induction hypothesis, $u'[v/x] \Vdash C'$ and thus $u'[v/x] \star \varpi \in \perp\!\!\!\perp$. Therefore, $\lambda x\, u' \star v.\varpi \in \perp\!\!\!\perp$, since $\perp\!\!\!\perp$ is cc-saturated.

If it is rule 4, we have $t = (\mathsf{cc})u$ and $\Gamma \vdash u : (A \to B) \to A$. We want to show that $t' \in |A'|$, that is $(\mathsf{cc})u' \star \pi \in \perp\!\!\!\perp$ for any $\pi \in \|A'\|$. Since $\perp\!\!\!\perp$ is cc-saturated, it suffices to show that $u' \star k_\pi.\pi \in \perp\!\!\!\perp$. Now, we prove that $k_\pi \in |A' \to B'|$ : indeed, if $v \in |A'|$ and $\rho \in \|B'\|$, then $k_\pi \star v.\rho \succ v \star \pi \in \perp\!\!\!\perp$.
By the induction hypothesis, we know that $u' \in |(A' \to B') \to A'|$, so the result follows.

If it is rule 5 with a first order variable $x$, we have $A = \forall x\, B$ and $\Gamma \vdash t : B$. By the induction hypothesis, we have $t' \in |B'[a/x]|$ for every $a \in M$. Thus $t' \in \bigcap_{a \in M} |B'[a/x]| = |\forall x\, B'| = |A'|$.

If it is rule 5 with a second order variable $X$ of arity $p$, we have $A = \forall X\, B$ and $\Gamma \vdash t : B$. By the induction hypothesis, we have $t' \in |B'[R/X]|$ for every $R \in \mathcal{P}(\Pi)^{M^p}$. Thus $t' \in |\forall X\, B'| = |A'|$.

If it is rule 6, we have $A = B[\tau/x]$ and $\Gamma \vdash t : \forall x\, B$. By the induction hypothesis, we get $t' \in |\forall x\, B'|$. Now, if $a$ is the value of $\tau$ in $M$, then $|B'[\tau/x]| = |B'[a/x]| \supset |\forall x\, B'|$ and thus $t \in |B'[\tau/x]| = |A'|$.

If it is rule 7, we have $A = B[\Phi(z_1, \ldots, z_p)/X z_1 \ldots z_p]$ so that :
$A' = B'[\Phi'(z_1, \ldots, z_p)/X z_1 \ldots z_p]$. We have $\Gamma \vdash t : \forall X\, B$ and we must show that $t' \in |A'|$. By the induction hypothesis, we know that $t' \in |\forall X\, B'|$ and the result follows from lemma 3.

Q.E.D.

**Lemma 3.** *Let $\Phi$ (resp. $A$) be a formula with parameters with the only free variables $z_1, \ldots, z_p$ (resp. $X$ of arity $p$). Define $R \in \mathcal{P}(\Pi)^{M^p}$ by :*

$R(a_1,\dots,a_p) = \|\Phi[a_1/z_1,\dots,a_p/z_p]\|$ *for every* $a_1,\dots,a_p \in M$.
*Then* $\|A[\Phi/Xz_1\dots z_p]\| = \|A[R/X]\|$ *and therefore :*
$\|A[\Phi/Xz_1\dots z_p]\| \subset \|\forall X\, A\|$ *and* $|\forall X\, A| \subset |A[\Phi/Xz_1\dots z_p]|$.

We prove $\|A[\Phi/Xz_1\dots z_p]\| = \|A[R/X]\|$ by induction on the length of $A$.
It is trivial if $X$ is not free in $A$, if $A$ is atomic, or if $A = B \to C$.
If $A = \forall x\, B$ then $\|A[\Phi/Xz_1\dots z_p]\| = \bigcup_{a\in M} \|B[\Phi/Xz_1\dots z_p][a/x]\|$
$= \bigcup_{a\in M} \|B[a/x][\Phi/Xz_1\dots z_p]\| = \bigcup_{a\in M} \|B[a/x][R/X]\|$ by induction hypothesis
$= \bigcup_{a\in M} \|B[R/X][a/x]\| = \|\forall x\, B[R/X]\| = \|A[R/X]\|$.
If $A = \forall Y\, B$, with $Y$ of arity $q$ and $Y \neq X$, then :
$\|A[\Phi/Xz_1\dots z_p]\| = \bigcup\{\|B[\Phi/Xz_1\dots z_p][S/Y]\|;\, S \in \mathcal{P}(\Pi)^{M^q}\}$
$= \bigcup\{\|B[S/Y][\Phi/Xz_1\dots z_p]\|;\, S \in \mathcal{P}(\Pi)^{M^q}\}$
$= \bigcup\{\|B[S/Y][R/X]\|;\, S \in \mathcal{P}(\Pi)^{M^q}\}$ by induction hypothesis
$= \bigcup\{\|B[R/X][S/Y]\|;\, S \in \mathcal{P}(\Pi)^{M^q}\} = \|\forall Y\, B[R/X]\| = \|A[R/X]\|$.
Q.E.D.

# Second order arithmetic

From now on, the set of individuals is $\mathbb{N}$ and we suppose that, for each recursive function $\phi : \mathbb{N}^k \to \mathbb{N}$, there is a $k$-ary function symbol $f_\phi$ in the language $\mathcal{L}$, which is, of course, interpreted by $\phi$ in the model. We shall write $\phi$ instead of $f_\phi$, which will cause no problem. There is no other function symbol and therefore, the model is completely defined as soon as we have chosen the set $\perp\!\!\!\perp$.

Let $F$ be a closed formula. We say that $F$ is *realized* if there is a proof-like $\lambda_c$-term $t$ such that $t \Vdash F$ *for every choice of* $\perp\!\!\!\perp$. By theorem 1 :
*The set of realized formulas is closed under deduction in classical second order logic.*

**Theorem 4.** *i) Let* $a, b \in \mathbb{N}$. *Then :*
$\|a = b\| = \{t.\pi;\, t \in \Lambda_c^0, \pi \in \Pi\} = \|\top \to \bot\|$ *if* $a \neq b$
$\|a = b\| = \{t.\pi;\, t \in \Lambda_c^0, \pi \in \Pi, t \star \pi \in \perp\!\!\!\perp\} = \|\forall X(X \to X)\|$ *if* $a = b$.
*ii) Every equational formula :* $\forall x_1 \dots \forall x_k[\tau(x_1,\dots,x_k) = \tau'(x_1,\dots,x_k)]$ *where* $\tau$ *and* $\tau'$ *are terms of* $\mathcal{L}$, *which is true in* $\mathbb{N}$, *is realized by the* $\lambda$*-term* $I = \lambda x\, x$.

i) Trivial, by definition of the formula $a = b$ which is $\forall X(Xa \to Xb)$.
ii) Easy consequence of (i).
Q.E.D.

The axioms of second order Peano arithmetic (denoted by $\mathrm{PA}_2$) can be given in the following way in $\mathcal{L}$, with a constant symbol 0, two unary function symbols $s$ (the successor function) and $p$ (the predecessor function), and two binary symbols $+, *$ :
1. $s0 \neq 0$ ; $p0 = 0$ ; $\forall x(psx = x)$ ; $\forall x(x + 0 = x)$ ; $\forall x(x * 0 = 0)$ ;
$\forall x \forall y(x + sy) = s(x + y)$ ; $\forall x \forall y(x * sy) = x * y + x$ ;
2. $\forall x\, Int[x]$ where $Int[x] \equiv \forall X\{\forall y(Xy \to Xsy), X0 \to Xx\}$.
Formula 2 is the axiom of recurrence ; $Int[x]$ reads as " $x$ is an integer ".
It is easy to check that axioms 1 are realized ; indeed :

by theorem 4(i) we get $|s0 \neq 0| = |(\top \to \bot) \to \bot|$ and therefore $\lambda x(x)t \Vdash s0 \neq 0$ for any $t \in \Lambda_c^0$. Other axioms 1 are equational formulas, and are realized by $I$, by theorem 4(ii).

Strangely enough, axiom 2 is not realized in general : indeed, it is generally impossible to find a proof-like $\lambda_c$-term which realizes simultaneously $Int[0]$ and $Int[s0]$, not to speak of $\forall x\, Int[x]$. Moreover, in most interesting models, *the negation of axiom 2 is realized* by a proof-like $\lambda_c$-term.

But our aim is, given a theorem $\Theta$ of $\mathrm{PA}_2$, i.e. a consequence of axioms 1 and 2, to find properties of any $\lambda_c$-term $t$ built with a proof of $\Theta$. In order to use realizability, we want to get rid of the axiom of recurrence. This can be done by the following well known property :

*Any proof of $\Theta$ in second order classical logic, using axioms 1 and 2, and any set $\mathcal{E}$ of equational formulas of $\mathcal{L}$ true in $\mathbb{N}$, can be transformed into a proof of $\Theta^{Int}$ using axioms 1, $\mathcal{E}$ and the following :*

*3. $\forall x_1 \ldots \forall x_k \{Int[x_1], \ldots, Int[x_k] \to Int[f(x_1, \ldots, x_k)]\}$ for each function symbol $f$ of $\mathcal{L}$.*

$\Theta^{Int}$ is the formula obtained by restricting to $Int$ all first order quantifiers of $\Theta$. It is inductively defined as follows :

If $A$ is atomic, $A^{Int} \equiv A$ ; $(A \to B)^{Int} \equiv A^{Int} \to B^{Int}$ ;
$(\forall X\, A)^{Int} \equiv \forall X\, A^{Int}$ ; $(\forall x\, A)^{Int} \equiv \forall x(Int[x] \to A^{Int})$.

Therefore, it now remains to prove the following :

**Theorem 5.** *Let $f$ be a $k$-ary function symbol of $\mathcal{L}$ (i.e. representing a recursive function). Then the formula $\forall x_1 \ldots \forall x_k \{Int[x_1], \ldots, Int[x_k] \to Int[f(x_1, \ldots, x_k)]\}$ is realized.*

We first need a result about usual $\lambda$-calculus. We will use the following :

Notations.
$\Lambda$ is the set of usual $\lambda$-terms. If $t, u \in \Lambda$, then $t \simeq_\beta t'$ means that $t$ is $\beta$-equivalent to $t'$ ; $t \succ t'$ means that $t$ reduces to $t'$ by *weak head reduction* : one step of weak head reduction is $(\lambda x\, u)vv_1 \ldots v_n \succ (u[v/x])v_1 \ldots v_n$.

The following lemma explains why we use the same symbol $\succ$ for weak head reduction of ordinary $\lambda$-terms and execution of processes.

**Lemma 6.**
*Let $\xi, \eta, t_1, \ldots, t_k \in \Lambda$ be closed ordinary $\lambda$-terms and $\pi \in \Pi$. If $\eta$ is not an application (i.e. $\eta$ is a $\lambda$-constant or $\eta = \lambda x\, \eta'$) and if $\xi \succ (\eta)t_1 \ldots t_k$, then $\xi \star \pi \succ \eta \star t_1. \ldots .t_k.\pi$.*

The proof is by induction on the length of the weak head reduction $\xi \succ (\eta)t_1 \ldots t_k$. The first step is $\xi = (\lambda y\, u)vv_1 \ldots v_n \succ (u[v/y])v_1 \ldots v_n$ and, by induction hypothesis :
$(u[v/y])v_1 \ldots v_n \star \pi \succ \eta \star t_1. \ldots .t_k.\pi$. Now, the first $n-1$ steps of reduction from the left-hand side concern some application ; since $\eta$ is not an application, we have not yet reached the right-hand side after these steps. Therefore, we have :
$(u[v/y]) \star v_1 \ldots v_n.\pi \succ \eta \star t_1. \ldots .t_k.\pi$. Thus :
$\xi \star \pi = (\lambda y\, u)vv_1 \ldots v_n \star \pi \succ (u[v/y]) \star v_1 \ldots v_n.\pi \succ \eta \star t_1. \ldots .t_k.\pi$.
Q.E.D.

Notations.
For $t, u \in \Lambda_c$ we define :

$(t)^n u$ for $n \in \mathbb{N}$ by : $(t)^0 u = u$, $(t)^{n+1} u = (t)(t)^n u$.
$t \circ u$ by $\lambda x(t)(u)x$, where $x$ does not appear in $t, u$.

**Lemma 7.** *Let $f, a$ be $\lambda$-constants and $\xi \in \Lambda$ such that $\xi \simeq_\beta (f)^n a$. Suppose that $\phi \Vdash \forall y(Xy \to Xsy)$ and $\alpha \Vdash X0$. Then $\xi[\phi/f, \alpha/a] \Vdash Xs^n 0$.*

Proof by induction on $n$. If $n = 0$, then $\xi \simeq_\beta a$ and therefore $\xi \succ a$. If $\pi \in \|X0\|$, then $\xi \star \pi \succ a \star \pi$ by lemma 6. Therefore $\xi[\phi/f, \alpha/a] \star \pi \succ \alpha \star \pi \in \perp\!\!\!\perp$.
If $n > 0$, then $\xi \succ (f)\eta$ with $\eta \simeq_\beta (f)^{n-1} a$. Let $\pi \in \|Xs^n 0\|$ ; then $\xi \star \pi \succ f \star \eta.\pi$ by lemma 6 and $\xi[\phi/f, \alpha/a] \star \pi \succ \phi \star \eta[\phi/f, \alpha/a].\pi$. Now, $\phi \in |Xs^{n-1}0 \to Xs^n 0|$ and, by induction hypothesis, $\eta[\phi/f, \alpha/a] \in |Xs^{n-1}0|$. Therefore $\phi \star \eta[\phi/f, \alpha/a].\pi \in \perp\!\!\!\perp$.
Q.E.D.

**Theorem 8.** *Let $n \in \mathbb{N}$ and $\nu \in \Lambda$ such that $\nu \simeq_\beta \lambda f \lambda x(f)^n x$. Then $\nu \Vdash Int[s^n 0]$.*

Let $\phi \Vdash \forall y(Xy \to Xsy)$, $\alpha \Vdash X0$ and $\pi \in \|Xs^n 0\|$ ; we must show that $\nu \star \phi.\alpha.\pi \in \perp\!\!\!\perp$. Since $\nu \simeq_\beta \lambda f \lambda x(f)^n x$, we have $\nu \succ \lambda f\, \eta$, $\eta \succ \lambda a\, \xi$ and $\xi \simeq_\beta (f)^n a$. By lemma 6, we have $\nu \star \phi.\alpha.\pi \succ \lambda f\, \eta \star \phi.\alpha.\pi \succ \eta[\phi/f] \star \alpha.\pi$. Again by lemma 6, we have :
$\eta \star \alpha.\pi \succ \lambda a\, \xi \star \alpha.\pi \succ \xi[\alpha/a] \star \pi$ and thus $\eta[\phi/f] \star \alpha.\pi \succ \xi[\phi/f, \alpha/a] \star \pi$. Finally, we get $\nu \star \phi.\alpha.\pi \succ \xi[\phi/f, \alpha/a] \star \pi$. But, by lemma 7, we have $\xi[\phi/f, \alpha/a] \in |Xs^n 0|$ and thus $\nu \star \phi.\alpha.\pi \succ \xi[\phi/f, \alpha/a] \star \pi \in \perp\!\!\!\perp$.
Q.E.D.

**Theorem 9.** *Let $T = (\lambda f \lambda n(n)\lambda g\, g \circ s)f0$, where $s$ is a $\lambda$-term for the successor (e.g. $s = \lambda n \lambda f \lambda x(f)(n)fx$). If $\phi \in \Lambda_c^0$ is such that $\phi \star s^n 0.\pi \in \perp\!\!\!\perp$ for all $\pi \in \|X\|$, then $T\phi \Vdash Int[s^n 0] \to X$.*

**Remark.** $T$ is called a *storage operator* (cf. [10]). We can understand intuitively what it does by comparing the weak head reductions of $\phi\nu$ and $T\phi\nu$, where $\phi$ and $\nu$ are ordinary $\lambda$-terms, $\nu \simeq_\beta \lambda f \lambda x(f)^n x$ (a Church integer). Then $T\phi\nu \succ (\phi)(s)^n 0$, which means that the computation of the integer argument $\nu$ of $\phi$ takes place first. In other words, $T$ emulates call-by-value inside weak head reduction, which is a call-by-name reduction.

Notice that we use the same symbol $s$ for a $\lambda$-term and a function symbol in $\mathcal{L}$.

PROOF. Let $\nu \Vdash Int[s^n 0]$ and $\pi \in \|X\|$ ; we have to show that $T\phi \star \nu.\pi \in \perp\!\!\!\perp$.
We define a unary second order parameter $P$ by :
$\|P(j)\| = \{s^{n-j}0.\pi\}$ for $0 \leq j \leq n$ and $\|P(j)\| = \emptyset$ for $j > n$.
We have $\phi \Vdash P0$ by hypothesis on $\phi$. We show that $\lambda g\, g \circ s \Vdash \forall x(Px \to Psx)$, which means that $\lambda g\, g \circ s \in |P(j) \to P(j+1)|$ for all $j \in \mathbb{N}$. This is trivial for $j \geq n$.
If $j < n$, let $\xi \in |P(j)|$ ; we must show $\lambda g\, g \circ s \star \xi.s^{n-j-1}0.\pi \in \perp\!\!\!\perp$. But we have :
$\lambda g\, g \circ s \star \xi.s^{n-j-1}0.\pi \succ \xi \circ s \star s^{n-j-1}0.\pi \succ \xi \star s^{n-j}0.\pi \in \perp\!\!\!\perp$ by hypothesis on $\xi$.
Now, we get $T\phi \star \nu.\pi \succ \nu \star \lambda g\, g \circ s.\phi.0.\pi$ which is in $\perp\!\!\!\perp$ because :
$\nu \Vdash \forall x(Px \to Psx), P0 \to Ps^n 0$ by hypothesis, $\phi \Vdash P0$ and $0.\pi \in \|Ps^n 0\|$.
Q.E.D.

We can now prove theorem 5. For simplicity, we suppose $k = 1$, so that we have a recursive function $f : \mathbb{N} \to \mathbb{N}$. Let $\phi \in \Lambda$ be a $\lambda$-term which represents $f$ : for $n \in \mathbb{N}$, we have $\phi s^n 0 \simeq_\beta \lambda f \lambda x(f)^p x$ with $p = f(n)$. Therefore $\phi s^n 0 \succ \lambda f\, \xi$ and, by theorem 8, we have $\lambda f\, \xi \Vdash Int[s^p 0]$. Let $\pi \in \|Int[s^p 0]\|$ ; we have $\lambda f\, \xi \star \pi \in \perp\!\!\!\perp$. Now, by lemma 6, we

have $\phi s^n 0 \star \pi \succ \lambda f\, \xi \star \pi$ ; but this reduction has necessarily at least one step, because $\phi s^n 0 \neq \lambda f\, \xi$ ($\phi s^n 0$ does not begin by $\lambda$). Therefore $\phi \star s^n 0.\pi \succ \lambda f\, \xi \star \pi \in \perp\!\!\!\perp$. Since this is true for every $\pi \in \|Int[s^p 0]\|$, it follows from theorem 9 that $T\phi \Vdash Int[s^n 0] \to Int[s^p 0]$, i.e. $T\phi \Vdash Int[n] \to Int[f(n)]$. Since this is true for each $n \in \mathbb{N}$, we get :
$T\phi \Vdash \forall x(Int[x] \to Int[f(x)])$.

Q.E.D.

# The countable axiom of choice

The axioms of classical analysis are : second order arithmetic and *the axiom of countable choice*. Thus, in order to get programs from proofs in classical analysis, it remains the problem : is it possible to realize this axiom ? This section is devoted to a positive answer to this problem.

The countable axiom of choice is the following axiom scheme :

$$\forall \vec{Y} \exists Z \forall x (\exists X\, F[x, X, \vec{Y}] \to F[x, Z(x,y)/Xy, \vec{Y}]).$$

$F$ is a second order formula in which the 2-ary variable $Z$ does not appear ; $X$ is of arity 1 ; $\vec{Y} = (Y_1, \ldots, Y_n)$ is a finite sequence of second order variables (which are, in fact, parameters).

We will rather write this axiom scheme in the contrapositive form :

(CAC) $$\forall \vec{Y} \exists Z \forall x (F[x, Z(x,y)/Xy, \vec{Y}] \to \forall X\, F[x, X, \vec{Y}]).$$

This means that $Z(x,y)$ is some counterexample to $\forall X\, F[x, X]$, whenever any exists.

In order to realize this scheme, we introduce in the $\lambda_c$-calculus a new constant denoted by $\chi$. We consider a fixed bijection $n \mapsto t_n$ of $\mathbb{N}$ onto $\Lambda_c^0$ (we do not even need that this bijection be recursive) ; let $\phi \mapsto n_\phi$ be the inverse function. For every $n \in \mathbb{N}$, we denote by $\underline{n}$ a fixed $\lambda$-term which is $\simeq_\beta \lambda f \lambda x (f)^n x$, for example $\lambda f \lambda x (f)^n x$ itself, or $s^n 0$, where $s$ is a $\lambda$-term for the successor. We extend the rules of execution of processes by giving the following rule for $\chi$ :

$$\chi \star \phi.\pi \succ \phi \star \underline{n}_\phi.\pi \quad \text{for every } \phi \in \Lambda_c^0 \text{ and } \pi \in \Pi.$$

**Remark.** We shall give below our intuitive interpretations of such a reduction rule.

From now on, of course, when we consider a set $\perp\!\!\!\perp$ of processes, it will be cc-saturated for the extended notion of reduction.

**Theorem 10.**
*Let $F[x, X]$ be a formula with parameters ($X$ being unary). There exists $\Phi : \mathbb{N}^3 \to \mathcal{P}(\Pi)$ such that :* $\chi \Vdash \forall x \{\forall n (Int[n] \to F[x, \Phi(x,n,y)/Xy]) \to \forall X\, F[x, X]\}$.

For each $n \in \mathbb{N}$, define $P_n(\perp\!\!\!\perp) = \{\pi \in \Pi;\ t_n \star \underline{n}.\pi \notin \perp\!\!\!\perp\}$. Now, for every individual $x$, we have $\|\forall X\, F[x, X]\| = \bigcup \{\|F[x, R/X]\|;\ R \in \mathcal{P}(\Pi)^{\mathbb{N}}\}$. Therefore, for every $x, n \in \mathbb{N}$ such that $P_n(\perp\!\!\!\perp) \cap \|\forall X\, F[x, X]\| \neq \emptyset$, there is a function $R : \mathbb{N} \to \mathcal{P}(\Pi)$ such that $P_n(\perp\!\!\!\perp) \cap \|F[x, Ry/Xy]\| \neq \emptyset$. By the axiom of countable choice, we get a function $\Phi : \mathbb{N}^3 \to \mathcal{P}(\Pi)$ which has the following property : if $P_n(\perp\!\!\!\perp) \cap \|\forall X\, F[x, X]\| \neq \emptyset$ then $P_n(\perp\!\!\!\perp) \cap \|F[x, \Phi(x,n,y)/Xy]\| \neq \emptyset$.

Now, let $x$ be fixed (in $\mathbb{N}$, which is the set of individuals) and consider $\pi \in \|\forall X\, F[x,X]\|$ and $\phi \in |\forall n(Int[n] \to F[x,\Phi(x,n,y)/Xy])|$. We must show that $\chi \star \phi.\pi \in \perp\!\!\!\perp$ and, by the rule for $\chi$, it is sufficient to show $\phi \star \underline{n}_\phi.\pi \in \perp\!\!\!\perp$.
If it is not true, we put $n = n_\phi$, so that $t_n = \phi$ and we have $\pi \in P_n(\perp\!\!\!\perp) \cap \|\forall X\, F[x,X]\|$. By definition of $\Phi$, there exists $\pi' \in P_n(\perp\!\!\!\perp) \cap \|F[x,\Phi(x,n,y)/Xy]\|$. From $\pi' \in P_n(\perp\!\!\!\perp)$, we get $t_n \star \underline{n}.\pi' \notin \perp\!\!\!\perp$. But, since $\pi' \in \|F[x,\Phi(x,n,y)/Xy]\|$, by the hypothesis on $\phi$, we have $\phi \star \underline{n}.\pi' \in \perp\!\!\!\perp$, because $\underline{n} \Vdash Int[s^n 0]$ (theorem 8). This is a contradiction, because $\phi = t_n$.
Q.E.D.

From this theorem, we can get a $\lambda_c$-term with $\chi$ which realizes the axiom of countable choice as follows :

**Lemma 11.** *There is a $\lambda_c$-term $\Omega$ such that :*
$\vdash \Omega : \forall Y \exists Z\{Func(Z), \forall x(\forall n[Z(x,n) \to Y(x,n)] \to \forall n[Int(n) \to Y(x,n)])\}$
*where $Func(Z)$ is the formula $\forall x \forall n \forall n'(Z(x,n), Z(x,n') \to n = n')$.*

Remember that $\exists Z\{A,B\}$ is a notation for $\forall X\{\forall Z(A, B \to X) \to X\}$. It is equivalent to $\exists Z(A \wedge B)$.

PROOF. The lemma is trivial, because this formula is clearly provable in classical second order logic : take simply for $Z(x,n)$ the formula which says that $n$ is the first integer such that $\neg Y(x,n)$ if there exists one ; i.e. $Z(x,n)$ is :
$Int[n] \wedge \neg Y(x,n) \wedge \forall p \forall q[Int[p], Int[q], p+q+1 = n \to Y(x,p)]$.
Q.E.D.

Now, by theorem 10, we know that :
$\chi \Vdash \forall x\{\forall n(Int[n] \to F[x,\Phi(x,n,y)/Xy]) \to \forall X\, F[x,X]\}$.
Take for $Y(x,n)$ in lemma 11, the formula $F[x,\Phi(x,n,y)/Xy]$. We get :
$\lambda x(\Omega)\lambda y(x)\lambda a\lambda b(\chi)(y)ab$
$$\Vdash \exists Z\{\mathrm{Func}(Z), \forall x(\forall n(Z(x,n) \to F[x,\Phi(x,n,y)/Xy]) \to \forall X\, F[x,X])\}$$
for any formula $F$. Therefore, if we define the axiom scheme (CCA) as :

(CCA) $\quad \exists U \exists Z\{\mathrm{Func}(Z), \forall x(\forall n(Z(x,n) \to F[x,U(x,n,y)/Xy]) \to \forall X\, F[x,X])\}$

we obtain :
(*) $\quad \Theta \Vdash$ (CCA) for every formula $F$.
with $\Theta = \lambda z(z)\lambda x(\Omega)\lambda y(x)\lambda a\lambda b(\chi)(y)ab$
The axiom scheme (CCA) is trivially equivalent to (CAC) and is realized by a $\lambda_c$-term which is independent of $F$. However, in order to derive (CAC) from (CCA), we need the " theorem scheme of extensionality " :
$$\forall X \forall Y\{\forall x(Xx \leftrightarrow Yx) \to (F[X] \leftrightarrow F[Y])\}.$$
This theorem scheme is proved by (concrete) induction on $F$ and therefore the associated $\lambda_c$-term will eventually depend on $F$.
Now, we can add to our deduction rules 1, ..., 7 the following one :
8. $\vdash \Theta : \exists U \exists Z\{\mathrm{Func}(Z), \forall x(\forall n(Z(x,n) \to F[x,U(x,n,y)/Xy]) \to \forall X\, F[x,X])\}$
Rules 1 to 8 form a deduction system for " classical analysis ", i.e. classical second order logic with countable choice. It follows from (*) that *the adequation lemma (theorem 1) remains valid for these deduction rules.*

# Dependent choice

In this section, we show, by the same method, that the axiom scheme of dependent choice can be realized. This axiom scheme is :

$$\forall X \exists Y\, H[X,Y] \to \forall X_0 \exists Z \{\forall y(X_0(y) \leftrightarrow Z(0,y)),$$
$$\forall k(Int[k] \to H[Z(k,y)/Xy, Z(k+1,y)/Yy])\}$$

for any formula $H$ in which the variable $Z$ does not appear.

Let $<x,y>$ be a binary function symbol, which represents a bijection from $\mathbb{N}^2$ onto $\mathbb{N}$. We assume it to be recursive, so that theorem 5 applies.

**Theorem 12.** *Let $F[X,Y]$ be a formula with parameters ($X,Y$ being unary). For every $X_0 : \mathbb{N} \to \mathcal{P}(\Pi)$, there exists $A : \mathbb{N}^3 \to \mathcal{P}(\Pi)$ such that $A(0,x,y) = X_0(y)$ and*
$$\chi \Vdash \forall x \forall k(\forall n\{Int[n] \to F[A(k,x,y)/Xy, A(k+1,<n,x>,y)/Yy]\}$$
$$\to \forall Y\, F[A(k,x,y)/Xy, Y]).$$

We first prove :

**Lemma 13.** *For every $U : \mathbb{N}^2 \to \mathcal{P}(\Pi)$, there exists $V : \mathbb{N}^2 \to \mathcal{P}(\Pi)$ such that :*
$\chi \Vdash \forall n\{Int[n] \to F[U(x,y)/Xy, V(<n,x>,y)/Yy]\} \to \forall Y\, F[U(x,y)/Xy, Y]$.

The proof is the same as theorem 10. Let $P_n(\perp\!\!\!\perp) = \{\pi \in \Pi;\ t_n \star \underline{n}.\pi \notin \perp\!\!\!\perp\}$ and define $V : \mathbb{N}^2 \to \mathcal{P}(\Pi)$ by the following condition : if $P_n(\perp\!\!\!\perp) \cap \|\forall Y\, F[U(x,y)/Xy, Y]\| \neq \emptyset$ then $P_n(\perp\!\!\!\perp) \cap \|F[U(x,y)/Xy, V(<n,x>,y)/Yy]\| \neq \emptyset$.
Now let $x$ be fixed in $\mathbb{N}$ and consider $\phi \in |\forall n\{Int[n] \to F[U(x,y)/Xy, V(<n,x>,y)/Yy]\}|$ and $\pi \in \|\forall Y\, F[U(x,y)/Xy, Y]\|$. We have to show that $\chi \star \phi.\pi \in \perp\!\!\!\perp$, i.e. $\phi \star \underline{n_\phi}.\pi \in \perp\!\!\!\perp$. If it is not true, we put $n = n_\phi$, so that $t_n = \phi$ and we have :
$\pi \in P_n(\perp\!\!\!\perp) \cap \|\forall Y\, F[U(x,y)/Xy, Y]\|$.
Thus, by definition of $V$, there exists $\pi' \in P_n(\perp\!\!\!\perp) \cap \|F[U(x,y)/Xy, V(<n,x>,y)/Yy]\|$. Since $\pi' \in P_n(\perp\!\!\!\perp)$, we have $t_n \star \underline{n}.\pi' \notin \perp\!\!\!\perp$. But $\pi' \in \|F[U(x,y)/Xy, V(<n,x>,y)/Yy]\|$ and thus, by hypothesis on $\phi$, we get $\phi \star \underline{n}.\pi' \in \perp\!\!\!\perp$. This is a contradiction because $\phi = t_n$.

Q.E.D.

By lemma 13, we have $(\forall U \in \mathcal{P}(\Pi)^{\mathbb{N}^2})(\exists V \in \mathcal{P}(\Pi)^{\mathbb{N}^2})\Phi(U,V)$ where $\Phi(U,V)$ is the formula $\chi \Vdash \forall n\{Int[n] \to F[U(x,y)/Xy, V(<n,x>,y)/Yy]\} \to \forall Y\, F[U(x,y)/Xy, Y]$. Therefore, we obtain theorem 12 by means of an application of the axiom of dependent choice to the formula $\Phi(U,V)$.

Q.E.D.

Now, by theorem 12, the following formula is realized by $\lambda x(x)II\chi$ (with $I = \lambda x\, x$) :

$$(*) \quad \forall X_0 \exists A\{\forall x \forall y[A(0,x,y) \to X_0(y)], \forall x \forall y[X_0(y) \to A(0,x,y)],$$
$$\forall x \forall k(\forall n\{Int[n] \to F[A(k,x,y)/Xy, A(k+1,<n,x>,y)/Yy]\}$$
$$\to \forall Y\, F[A(k,x,y)/Xy, Y])\}.$$

Therefore, in order to realize the axiom of dependent choice, it is sufficient to derive this axiom from formula $(*)$, where $F$ is $\neg H$, in classical second order logic. The derivation is as follows : define inductively the sequence $n_k$ of integers by the conditions $n_0 = 0$ and $n_{k+1} = <n, n_k>$ for the first integer $n$ such that $H[A(k,n_k,y)/Xy, A(k+1,<n,n_k>,y)/Yy]$. The formula $(*)$ and the hypothesis $\forall X \exists Y\, H(X,Y)$ ensure that such an integer always exists. Finally, if we define $Z(k,y)$ by $A(k,n_k,y)$, we get :
$\forall k\{Int[k] \to H[Z(k,y)/Xy, Z(k+1,y)/Yy]\}$ and $\forall y(Z(0,y) \leftrightarrow X_0(y))$.

# Variant and interpretations

We can get the same results by using, instead of $\chi$, a dual instruction $\overline{\chi}$ which works on stacks instead of terms. Consider now a bijection $n \mapsto \pi_n$ of $\mathbb{N}$ onto $\Pi$, and let $\pi \mapsto n_\pi$ be the inverse function. We introduce into the $\lambda_c$-calculus a new constant denoted by $\overline{\chi}$ and we extend the rules of execution of processes by giving the following rule for $\overline{\chi}$ :

$$\overline{\chi} \star \phi.\pi \succ \phi \star \underline{n}_\pi.\pi \quad \text{for every } \phi \in \Lambda_c^0 \text{ and } \pi \in \Pi.$$

We have the analogue of theorem 10 and the proof is even simpler :

**Theorem 14.**
*Let $F[x, X]$ be a formula with parameters ($X$ being unary). There exists $\Phi : \mathbb{N}^3 \to \mathcal{P}(\Pi)$ such that : $\overline{\chi} \Vdash \forall x\{\forall n(Int[n] \to F[x, \Phi(x, n, y)/Xy]) \to \forall X\, F[x, X]\}$.*

For every individual $x$ and every stack $\pi$, we have :

$$\pi \in \|\forall X\, F[x, X]\| \Leftrightarrow (\exists R \in \mathcal{P}(\Pi)^{\mathbb{N}})\, \pi \in \|F[x, R/X]\|.$$

Therefore, by the axiom of countable choice, there exists a function $\Phi : \mathbb{N}^3 \to \mathcal{P}(\Pi)$ such that $\pi \in \|\forall X\, F[x, X]\| \Leftrightarrow \pi \in \|F[x, \Phi(x, n_\pi, y)/Xy]\|$. We show that $\Phi$ has the desired property : consider an individual $x \in \mathbb{N}$, $\phi \in |\forall n(Int[n] \to F[x, \Phi(x, n, y)/Xy])|$ and $\pi \in \|\forall X\, F[x, X]\|$. We must show that $\overline{\chi} \star \phi.\pi \in \perp\!\!\!\perp$ and, by the rule for $\overline{\chi}$, it is sufficient to show $\phi \star \underline{n}_\pi.\pi \in \perp\!\!\!\perp$. But this is clear because :
- by hypothesis on $\phi$, we have $\phi \Vdash Int(s^{n_\pi}0) \to F[x, \Phi(x, n_\pi, y)/Xy]$ ;
- by theorem 8, we have $\underline{n}_\pi \Vdash Int(s^{n_\pi}0)$ ;
- and by hypothesis on $\pi$ and the definition of $\Phi$, we have $\pi \in \|F[x, \Phi(x, n_\pi, y)/Xy]\|$.

Q.E.D.

We get in the same way an analogue of lemma 13. Now, by exactly the same reasoning as above, we realize the axioms of countable choice and of dependent choice, using $\overline{\chi}$ instead of $\chi$.

### The 'quote' interpretation

The rule of reduction for the instruction $\chi$ is :

$$\chi \star \phi.\pi \succ \phi \star \underline{n}_\phi.\pi.$$

This rule makes $\chi$ very similar to the `quote` instruction of LISP. Indeed $\underline{n}_\phi$ may be used in the same way as `(quote` $\phi$`)`, if we assume that $\phi \mapsto n_\phi$ is a recursive bijection from $\Lambda_c^0$ onto $\mathbb{N}$. For example, since $n_u$ is then a recursive function of $n_{tu}$, we can define, using $\chi$, an instruction $\chi'$ such that $\chi' \star \phi.\psi.\pi \succ \phi \star n_\psi.\pi$ : let $p$ be a closed $\lambda$-term such that $(p)\underline{n}_{tu} \simeq_\beta \underline{n}_u$ and let $\chi' = \lambda x \lambda y(\chi)(\lambda d\, x \circ p)y$. As we can see in this example, the instruction $\chi$ is not compatible with $\beta$-reduction : indeed, we cannot replace $(\lambda d\, x \circ p)y$ with $x \circ p$.
As observed by the referee, in the program associated with the axiom of countable choice, the instruction $\chi$ is not used in the same way as the 'quote' instruction is habitually used in LISP. In fact, the only operation which is performed on the integers $n_\phi$ is to compare them, and *$\phi$ is never retrieved from $n_\phi$*. In other words, there is no use of the 'unquote' or 'eval' instruction which, in LISP, is the inseparable companion of 'quote'.
Nevertheless, it is a fact that the instruction $\chi$ can be implemented by means of the

'quote' instruction. Moreover, it is quite possible that, in future work, trying to interpret some other axioms or theorems, someone will use the program 'eval' in interaction with $\chi$.

**The clock interpretation**

We give now another interpretation for the instruction $\chi$ (which is also valid for $\overline{\chi}$).
We observe that the application $n \mapsto t_n$ may be *any surjective map* from $\mathbb{N}$ onto $\Lambda_c^0$. The reduction rule for $\chi$ is then :

$$\chi \star \phi.\pi \succ \phi \star \underline{n}.\pi.$$

where $n$ is any integer such that $t_n = \phi$. This suggests the following interpretation :
$\chi$ is an input instruction and, when it comes in head position, the process $\chi \star \phi.\pi$ waits for some integer $n$ which is provided by some human operator or some external process. Then we have the reduction $\chi \star \phi.\pi \succ \phi \star \underline{n}.\pi$ and the execution goes on. The only constraint is that " $\phi$ must be retrievable from $n$ ", i.e. the integers provided to the processes $\chi \star \phi.\pi$ and $\chi \star \phi'.\pi'$ with $\phi \neq \phi'$, must be different.
A very simple and natural way to obtain this behaviour is to provide the integer $n$ by means of a *clock*, since two different $\lambda_c$-terms cannot appear at the same time. In other words, we may suppose there is a second process running aside the main one, which simply increments an integer at each step of reduction. This process provides the integer $n$ when needed, that is when $\chi$ comes in head position in the main process.
This method gives a completely different way of implementing the instruction $\chi$. I think that the behaviour of the program associated with the axioms of countable or dependent choice is easier to understand with this implementation.

# Arithmetical theorems

In this section, we apply the above results to study the behaviour of programs associated with proofs of arithmetical theorems in $\mathrm{PA}_2 +$ CAC. As an example, we consider first a $\Sigma_2^0$ formula $\Phi$ of the form $\exists x \forall y[\phi(x,y) = 0]$ where $\phi$ is a recursive function.
Consider the following game between two players named $\exists$ and $\forall$ : $\exists$ plays an integer $m$, $\forall$ reply with an integer $n$ ; the game stops at the first moment when $\phi(m,n) = 0$ and then $\exists$ wins ; thus, $\forall$ wins if and only if the game lasts infinitely long.
The following is trivial :
$\exists$ has a winning strategy if and only if $\mathbb{N} \models \Phi$ ; moreover, in this case, there is an obvious winning strategy for $\exists$ : to play successively $0, 1, 2, \ldots$
We will show (theorem 15) that the program associated with a proof of $\Phi$ in $\mathrm{PA}_2 +$ CAC acts as a winning strategy for the above game. We first get rid of the axiom of recurrence as explained above and thus, we consider a proof in " classical analysis " of the formula :

$$\Phi^{Int} \equiv \forall x[Int(x), \forall y(Int(y) \to \phi(x,y) = 0) \to \bot] \to \bot.$$

Now we add the following constants to $\lambda_c$-calculus : $\kappa_{np}$ $(n, p \in \mathbb{N})$ and $\kappa$ which is an *input instruction.* Thus the $\lambda_c$-terms become *interactive programs.* Once again, we extend the rules of execution of processes by giving the following rule for $\kappa$ :

$$\kappa \star s^n 0.\xi.\pi \succcurlyeq \xi \star s^p 0.\kappa_{np}.\pi'$$

for $n, p \in \mathbb{N}$, $\xi \in \Lambda_0$, $\pi, \pi' \in \Pi$ ; $s$ is is a fixed $\lambda$-term for the successor in Church integers.
*This rule is non-deterministic*, since the integer $p$ and the stack $\pi'$ are arbitrary.

The intuitive meaning of this rule is as follows : in the left-hand member the program, which stands for the player $\exists$, proposes the integer $n$ ; then, in the right-hand member, the opponent $\forall$ replies with $p$ and the execution goes on ; $\kappa_{np}$ keeps a trace of the ordered pair $(n, p)$.

**Theorem 15.** *Suppose that* $\vdash \theta : [\exists x \forall y(\phi(x, y) = 0)]^{Int}$ *in classical second order logic with choice. Then* every reduction *of* $\theta \star T\kappa.\pi$ *following* $\succ$ *ends up into* $\kappa_{np} \star \pi'$ *with* $\phi(n, p) = 0$. *$T$ is the storage operator of theorem 9.*

We define $\perp\!\!\!\perp$ as the set of processes *all* reductions of which end up into $\kappa_{np} \star \pi'$ *with* $\phi(n, p) = 0$ for some stack $\pi'$. We must show that $\theta \star T\kappa.\pi \in \perp\!\!\!\perp$ for every $\pi \in \Pi$.
By theorem 1, we have $\theta \Vdash \forall x[Int(x), \forall y(Int(y) \to \phi(x, y) = 0) \to \bot] \to \bot$.
Therefore, by definition of $\Vdash$, it is sufficient to prove that :
$T\kappa \Vdash \forall x[Int(x), \forall y(Int(y) \to \phi(x, y) = 0) \to \bot]$.
Let $n \in \mathbb{N}$ ; we have to show $T\kappa \Vdash Int[s^n 0] \to [\forall y(Int(y) \to \phi(n, y) = 0) \to \bot]$.
By theorem 9, this amounts to show that if $\pi \in \Pi$ and $\xi \Vdash \forall y(Int(y) \to \phi(n, y) = 0)$ then $\kappa \star s^n 0.\xi.\pi \in \perp\!\!\!\perp$. By the very definition of $\perp\!\!\!\perp$, it is sufficient to show that $\xi \star s^p 0.\kappa_{np}.\pi \in \perp\!\!\!\perp$ for every $p \in \mathbb{N}$ and $\pi \in \Pi$. But, by hypothesis on $\xi$, for any $p \in \mathbb{N}$ and $\varpi \in \|\phi(n, p) = 0\|$, we have $\xi \star s^p 0.\varpi \in \perp\!\!\!\perp$. Therefore, it is sufficient to show that $\kappa_{np}.\pi \in \|\phi(n, p) = 0\|$ for any $p \in \mathbb{N}$ and $\pi \in \Pi$.
If $\phi(n, p) = 0$ then, by theorem 4(i), $\|\phi(n, p) = 0\|$ is $\|\forall X(X \to X)\|$ that is :
$\{t.\rho;\ t \in \Lambda_c^0, \rho \in \Pi, t \star \rho \in \perp\!\!\!\perp\}$. But, by definition of $\perp\!\!\!\perp$, we have also $\kappa_{np} \star \pi \in \perp\!\!\!\perp$, so that $\kappa_{np}.\pi \in \|\phi(n, p) = 0\|$.
If $\phi(n, p) \neq 0$, again by theorem 4(i), we have $\|\phi(n, p) = 0\| = \|\top \to \bot\|$ that is :
$\{t.\rho;\ t \in \Lambda_c^0, \rho \in \Pi\}$ ; therefore, we have again $\kappa_{np}.\pi \in \|\phi(n, p) = 0\|$.
Q.E.D.

It follows that any proof of $\Phi$ in classical second order arithmetic with countable axiom of choice provides an interactive program which can stand in for the player $\exists$ and which wins against every opponent.

Indeed, after each reply of the opponent, the program provides an *object* $(s^n 0, \xi)$ made up with an integer $n$ (the provisional solution) and an *exception handler* $\xi$ which is used in case of a relevant reply from the opponent.
These are the two arguments of $\kappa$ which can therefore be seen as a *pointer* to this object.

## The general case

Consider now an arithmetical theorem $\Phi$ of the form :
$\exists x_1 \forall y_1 \ldots \exists x_k \forall y_k(\phi(x_1, y_1, \ldots, x_k, y_k) = 0)$ where $\phi$ is recursive.
The game associated with $\Phi$ is now the following :
A *position* of the game is an integer sequence $n_1 p_1 \ldots n_i p_i$ $(0 \leq i \leq k)$. The player $\exists$ chooses first an *already reached* position $n_1 p_1 \ldots n_i p_i$, with $0 \leq i < k$, and an integer $n_{i+1}$ ; then $\forall$ chooses an integer $p_{i+1}$. The position $n_1 p_1 \ldots n_{i+1} p_{i+1}$ is then reached.
If $i + 1 = k$ and $\phi(n_1, p_1, \ldots, n_k, p_k) = 0$, then the game stops and $\exists$ *won*. In every other case, the play goes on. Thus $\forall$ wins if and only if the game lasts infinitely long.

It is easily seen that $\mathbb{N} \models \Phi$ if and only if the player $\exists$ has a winning strategy for this game. Moreover, we can effectively (and easily) describe such a winning strategy, which

does not even depend on $\Phi$, but only on the number $k$ of quantifiers :
The player $\exists$ uses an effective enumeration of $\mathbb{N}^k$. When he comes to the $k$-uple $n_1 \dots n_k$, he chooses the longest already reached position of the form $n_1 p_1 \dots n_i p_i$. We know that $i < k$ because this position was reached for a $k$-uple of integers different from $n_1 \dots n_k$. Then he successively plays $n_{i+1}, \dots, n_k$ regardless of the choices of $\forall$. Then he takes the next $k$-uple of integers.
If this play is infinite, we get $k$ functions $f_i(x_1, \dots, x_i)$ such that :

$$\mathbb{N} \models \forall x_1 \dots \forall x_k \, \{\phi[x_1, f_1(x_1), x_2, f_2(x_1, x_2), \dots, x_k, f_k(x_1, \dots, x_k)] \neq 0\}$$

which is the Skolem form of $\neg\Phi$ ; thus $\mathbb{N} \models \neg\Phi$.

Conversely, if $\mathbb{N} \models \neg\Phi$, there exists $k$ functions $f_i(x_1, \dots, x_i)$ such that the Skolem form of $\neg\Phi$ is satisfied. Of course, they provide a winning strategy to the opponent $\forall$.

Let us now introduce into the $\lambda_c$-calculus the constants $\kappa_{n_1 p_1 \dots n_i p_i}$ for $0 \leq i \leq k$ (one for each position of the game). Their rule of reduction is as follows :

For $0 \leq i \leq k-2$ : $\quad \kappa_{n_1 p_1 \dots n_i p_i} \star s^{n_{i+1}} 0.\xi.\pi \succ \xi \star s^{p_{i+1}} 0.T\kappa_{n_1 p_1 \dots n_{i+1} p_{i+1}}.\pi'$
($T$ is the storage operator of theorem 9).

For $i = k-1$ : $\quad \kappa_{n_1 p_1 \dots n_{k-1} p_{k-1}} \star s^{n_k} 0.\xi.\pi \succ \xi \star s^{p_k} 0.\kappa_{n_1 p_1 \dots n_k p_k}.\pi'$.

For $i = k$ no reduction is possible.

It is a *non-deterministic* rule, since $p_{i+1}$ and $\pi'$ are arbitrary.

The intuitive meaning of these rules is the following : in the left-hand side, the program, which stands for the player $\exists$, chooses the previously reached position $n_1 p_1 \dots n_i p_i$ and proposes the integer $n_{i+1}$ ; then, in the right-hand side, the opponent $\forall$ replies with $p_{i+1}$ and the execution goes on ; $\kappa_{n_1 p_1 \dots n_{i+1} p_{i+1}}$ keeps a trace of the fact that the position $n_1 p_1 \dots n_{i+1} p_{i+1}$ has been reached.

**Theorem 16.** *Suppose that* $\vdash \theta : [\exists x_1 \forall y_1 \dots \exists x_k \forall y_k (\phi(x_1, y_1, \dots, x_k, y_k) = 0)]^{Int}$ *in classical second order logic with choice. Then* every reduction *of* $\theta \star T\kappa.\pi$ *following* $\succ$ *ends up into* $\kappa_{n_1 p_1 \dots n_k p_k} \star \pi'$ *with* $\phi(n_1, p_1, \dots, n_k, p_k) = 0$.

**Remark.** $\kappa$ is the constant associated with the empty position of the game.

The meaning of the theorem is that each proof of $\Phi^{Int}$ in classical second order logic with axiom of choice gives an interactive program, which wins against every strategy of the opponent, in the game associated with the formula :

$$\Phi \equiv \exists x_1 \forall y_1 \dots \exists x_k \forall y_k (\phi(x_1, y_1, \dots, x_k, y_k) = 0).$$

This theorem is closely related to the *no-counter-example interpretation* of G. Kreisel[7, 8] (see also [3, 6, 13]) : Kreisel has shown that, if $\Phi$ is a theorem of first order Peano arithmetic, then there exists *type recursive functionals* in the sense of [13] $F_i(f_1, \dots, f_k)$ $(1 \leq i \leq k)$ such that :

$$\phi[\xi_1, f_1(\xi_1), \xi_2, f_2(\xi_1, \xi_2), \dots, \xi_k, f_k(\xi_1, \dots, \xi_k)] = 0 \text{ with } \xi_i = F_i(f_1, \dots, f_k)$$

for any functions $f_i : \mathbb{N}^i \to \mathbb{N}$ $(1 \leq i \leq k)$, i.e. for any strategy of the opponent. Theorem 16 associates such a functional *to each proof* of $\Phi$ in classical analysis (with axiom of choice) and gives it as an explicit program which is a winning strategy for the player $\exists$.

PROOF.
We define $\perp\!\!\!\perp$ as the set of processes *all* reductions of which end up into $\kappa_{n_1 p_1 \dots n_k p_k} \star \pi'$

with $\phi(n_1, p_1, \ldots, n_k, p_k) = 0$. We have to show that $\theta \star T\kappa.\pi \in \perp\!\!\!\perp$ for every $\pi \in \Pi$.
For $0 \leq i \leq k$, we set :

$$A_i(x_1, y_1, \ldots, x_i, y_i) \equiv [\exists x_{i+1} \forall y_{i+1} \ldots \exists x_k \forall y_k (\phi(x_1, y_1, \ldots, x_k, y_k) = 0)]^{Int}.$$

By theorem 1, it is sufficient to show that $T\kappa.\pi \in \|A_0\|$ for every stack $\pi$. In fact, we shall show, by decreasing induction from $k$, that for every stack $\pi$ :
$K_{n_1 p_1 \ldots n_i p_i}.\pi \in \|A_i(n_1, p_1, \ldots, n_i, p_i)\|$ for $0 \leq i \leq k$ where
$K_{n_1 p_1 \ldots n_i p_i}$ is $T\kappa_{n_1 p_1 \ldots n_i p_i}$ for $0 \leq i < k$ and $\kappa_{n_1 p_1 \ldots n_k p_k}$ for $i = k$.
We show this first for $i = k$ ; we have to show that :
$\kappa_{n_1 p_1 \ldots n_k p_k}.\pi \in \|\phi(n_1, p_1, \ldots, n_k, p_k) = 0\|$.
If $\phi(n_1, p_1, \ldots, n_k, p_k) \neq 0$ then, by theorem 4(i), we have :
$\|\phi(n_1, p_1, \ldots, n_k, p_k) = 0\| = \|\top \to \bot\|$, hence the result.
If $\phi(n_1, p_1, \ldots, n_k, p_k) = 0$, then, by theorem 4(i), we have :
$\|\phi(n_1, p_1, \ldots, n_k, p_k) = 0\| = \{t.\pi;\, t \in \Lambda_c^0, \pi \in \Pi, t \star \pi \in \perp\!\!\!\perp\}$.
But $\kappa_{n_1 p_1 \ldots n_k p_k} \star \pi \in \perp\!\!\!\perp$ by definition of $\perp\!\!\!\perp$ ; hence the result.
Assuming the property for $i + 1$, we have to show that :
$T\kappa_{n_1 p_1 \ldots n_i p_i} \Vdash$

$$\forall x_{i+1}\{Int(x_{i+1}), \forall y_{i+1}[Int(y_{i+1}) \to A_{i+1}(n_1, p_1, \ldots, n_i, p_i, x_{i+1}, y_{i+1})] \to \bot\}.$$

That is, for every $n_{i+1} \in \mathbb{N}$ :
$T\kappa_{n_1 p_1 \ldots n_i p_i} \Vdash Int(n_{i+1}) \to \{\forall y_{i+1}[Int(y_{i+1}) \to A_{i+1}(n_1, p_1, \ldots, n_i, p_i, n_{i+1}, y_{i+1})] \to \bot\}$.
By theorem 9, it is sufficient to show that $\kappa_{n_1 p_1 \ldots n_i p_i} \star s^{n_{i+1}}0.\xi.\pi \in \perp\!\!\!\perp$ for every stack $\pi$ and every $\xi \in |\forall y_{i+1}[Int(y_{i+1}) \to A_{i+1}(n_1, p_1, \ldots, n_i, p_i, n_{i+1}, y_{i+1})]|$. Now, by definition of $\succ$, this amounts to show that $\xi \star s^{p_{i+1}}0.K_{n_1 p_1 \ldots n_{i+1} p_{i+1}}.\pi \in \perp\!\!\!\perp$ for every $p_{i+1} \in \mathbb{N}$ and $\pi \in \Pi$.
But this is clear, by hypothesis on $\xi$ since, by the induction hypothesis, we have for every stack $\pi$ : $K_{n_1 p_1 \ldots n_{i+1} p_{i+1}}.\pi \in \|A_{i+1}(n_1, p_1, \ldots, n_i, p_i, n_{i+1}, p_{i+1})\|$ ; and $s^{p_{i+1}}0 \in |Int[s^{p_{i+1}}0]|$ by theorem 8.

Q.E.D.

# References


[1] S. Berardi, M. Bezem, T. Coquand. On the computational content of the axiom of choice. J. Symbolic Logic 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. Symbolic Logic 60, pp. 325-337 (1995).

[4] J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse. In: Proc. 2nd Scand. Logic Symp. p. 63-92. North Holland Pub. Co. (1971).

[5] T. Griffin. A formulæ-as-type notion of control. In Conference Record of the 17th A.C.M. Symposium on Principles of Programming Languages (1990).

[6] U. Kohlenbach. On the no-counter-example interpretation. J. Symbolic Logic 64, pp. 1491-1511 (1999).

[7] G. Kreisel. On the interpretation of non-finitist proofs, part I. J. Symbolic Logic 16, pp. 241-267 (1951).

[8] G. Kreisel. On the interpretation of non-finitist proofs, part II: Interpretation of number theory, applications. J. Symbolic Logic 17, pp. 43-58 (1952).

[9] G. Kreisel. Mathematical significance of consistency proofs. J. Symbolic Logic 23, pp. 155-182 (1958).

[10] J.-L. Krivine. A general storage theorem for integers in call-by-name $\lambda$-calculus. Theor. Comp. Sc. 129, p. 79-94 (1994).

[11] J.-L. Krivine. Typed lambda-calculus in classical Zermelo-Fraenkel set theory. Arch. Math. Logic 40 (2001) 3, 189-205.

[12] M. Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. Proc. Logic Progr. and Autom. Reasoning, St Petersbourg. L.N.C.S. 624, p. 190-201 (1992).

[13] J.R. Shoenfield. Mathematical logic. Addison Wesley (1967).

[14] G. Takeuti. Proof Theory. Studies in Logic and Foundations of Mathematics, North-Holland (1987).