Mixed Logic and Storage Operators

Karim Nour

Mixed Logic and Storage Operators

Karim NOUR
LAMA - Equipe de Logique, Université de Chambéry
73376 Le Bourget du Lac
e-mail nour@univ-savoie.fr

Abstract

In 1990 J-L. Krivine introduced the notion of storage operators. They are λ-terms which simulate call-by-value in the call-by-name strategy and they can be used in order to modelize assignment instructions. J-L. Krivine has shown that there is a very simple second order type in $AF2$ type system for storage operators using Gődel translation of classical to intuitionistic logic.
In order to modelize the control operators, J-L. Krivine has extended the system $AF2$ to the classical logic. In his system the property of the unicity of integers representation is lost, but he has shown that storage operators typable in the system $AF2$ can be used to find the values of classical integers. In this paper, we present a new classical type system based on a logical system called mixed logic. We prove that in this system we can characterize, by types, the storage operators and the control operators. We present also a similar result in the M. Parigot's $\lambda\mu$-calculus.

1 Introduction

In 1990, J.L. Krivine introduced the notion of storage operators (see [4]). They are closed λ-terms which allow, for a given data type (the type of integers, for example), to simulate in λ-calculus the "call by value" in a context of a "call by name" (the head reduction) and they can be used in order to modelize assignment instructions. J.L. Krivine has shown that the formula $\forall x\{N^*[x] \to \neg\neg N[x]\}$ is a specification for storage operators for Church integers : where $N[x]$ is the type of integers in $AF2$ type system, and the operation $*$ is the simple Gődel translation from classical to intuitionistic logic which associates to every formula F the formula F^* obtained by replacing in F every atomic formula by its negation (see [3]).

The latter result suggests many questions :

- Why do we need a Gődel translation ?
- Why do we need the type $N^*[x]$ which characterize a class larger than integers ?

In order to modelize the control operators, J-L. Krivine has extended the system $AF2$ to the classical logic (see [6]). His method is very simple : it consists of adding a new constant, denoted by C, with the declaration $C : \forall X\{\neg\neg X \to X\}$ which axiomatizes classical logic over intuitionistic logic. For the constant C, he adds a new reduction rule : $(Ctt_1...t_n) \to (t \quad \lambda x(x \quad t_1...t_n))$ which is a particular case of a rule given by Felleisen for control operator (see [1]). In this system the property of the unicity of integers representation is lost, but J-L. Krivine has shown that storage operators typable in the intuitionistic system $AF2$ can be used to find the values of classical integers [1](see [6]).

[1]The idea of using storage operators in classical logic is due to M. Parigot (see [19])

The latter result suggests also many questions :

- What is the relation between classical integers and the type $N^*[x]$?
- Why do we need intuitionistic logic to modelize the assignment instruction and classical logic to modelize the control operators ?

In this paper, we present a new classical type system based on a logical system called mixed logic. This system allows essentially to distinguish between classical proofs and intuitionistic proofs. We prove that, in this system, we can characterize, by types, the storage operators and the control operators. This results give some answers to the previous questions.

We present at the end (without proof) a similar result in the M. Parigot's $\lambda\mu$-calculus.

Acknowledgement. We wish to thank J.L. Krivine, and C. Paulin for helpful discussions. We don't forget the numerous corrections and suggestions from R. David and N. Bernard.

2 Pure and typed λ-calculus

- Let $t, u, u_1, ..., u_n$ be λ-terms, the application of t to u is denoted by $(t)u$. In the same way we write $(t)u_1...u_n$ instead of $(...((t)u_1)...)u_n$.
- $Fv(t)$ is the set of free variables of a λ-term t.
- The β-reduction (resp. β-equivalence) relation is denoted by $u \rightarrow_\beta v$ (resp. $u \simeq_\beta v$).
- The notation $\sigma(t)$ represents the result of the simultaneous substitution σ to the free variables of t after a suitable renaming of the bounded variables of t.
- We denote by $(u)^n v$ the λ-term $(u)...(u)v$ where u occurs n times, and $\overline{u}$ the sequence of λ-terms $u_1, ..., u_n$. If $\overline{u} = u_1, ..., u_n$ $n \geq 0$, we denote by $(t)\overline{u}$ the λ-term $(t)u_1...u_n$.
- Let us recall that a λ-term t either has a head redex [i.e. $t = \lambda x_1...\lambda x_n(\lambda xu)vv_1...v_m$, the head redex being $(\lambda xu)v$], or is in head normal form [i.e. $t = \lambda x_1...\lambda x_n(x)v_1...v_m$]. The notation $u \succ v$ means that v is obtained from u by some head reductions. If $u \succ v$, we denote by $h(u, v)$ the length of the head reduction between u and v.

Lemma 2.1 *(see[3])*
1) If $u \succ v$, then, for any substitution σ, $\sigma(u) \succ \sigma(v)$, and $h(\sigma(u), \sigma(v))$=h(u,v).
2) If $u \succ v$, then, for every sequence of λ-terms $\overline{w}$, there is a w, such that $(u)\overline{w} \succ w$, $(v)\overline{w} \succ w$, and $h((u)\overline{w}, w) = h((v)\overline{w}, w) + h(u, v)$.

Remark. Lemma 2.1 shows that to make the head reduction of $\sigma(u)$ (resp. of $(u)\overline{w}$) it is equivalent - same result, and same number of steps - to make some steps in the head reduction of u, and after make the head reduction of $\sigma(v)$ (resp. of $(v)\overline{w}$). □

- The types will be formulas of second order predicate logic over a given language. The logical connectives are $\perp$ (for absurd), $\rightarrow$, and $\forall$. There are individual (or first order) variables denoted by $x, y, z, ...$, and predicate (or second order) variables denoted by $X, Y, Z, ...$

- We do not suppose that the language has a special constant for equality. Instead, we define the formula $u = v$ (where u, v are terms) to be $\forall Y(Y(u) \to Y(v))$ where Y is a unary predicate variable. Such a formula will be called an equation. We denote by $a \approx b$, if $a = b$ is a consequence of a set of equations.

- The formula $F_1 \to (F_2 \to (... \to (F_n \to G)...))$ is also denoted by $F_1, F_2, ..., F_n \to G$. For every formula A, we denote by $\neg A$ the formula $A \to \bot$. If $\overline{v} = v_1, ..., v_n$ is a sequence of variables, we denote by $\forall \overline{v} A$ the formula $\forall v_1 ... \forall v_n A$.

- Let t be a λ-term, A a type, $\Gamma = x_1 : A_1, ..., x_n : A_n$ a context, and E a set of equations. We define by means of the following rules the notion "t is of type A in Γ with respect to E" ; this notion is denoted by $\Gamma \vdash_{AF2} t : A$:

  (1) $\Gamma \vdash_{AF2} x_i : A_i$ $1 \leq i \leq n$.

  (2) If $\Gamma, x : A \vdash_{AF2} t : B$, then $\Gamma \vdash_{AF2} \lambda x t : A \to B$.

  (3) If $\Gamma \vdash_{AF2} u : A \to B$, and $\Gamma \vdash_{AF2} v : A$, then $\Gamma \vdash_{AF2} (u)v : B$.

  (4) If $\Gamma \vdash_{AF2} t : A$, and x is not free in Γ, then $\Gamma \vdash_{AF2} t : \forall x A$.

  (5) If $\Gamma \vdash_{AF2} t : \forall x A$, then, for every term u, $\Gamma \vdash_{AF2} t : A[u/x]$.

  (6) If $\Gamma \vdash_{AF2} t : A$, and X is not free in Γ, then $\Gamma \vdash_{AF2} t : \forall X A$.

  (7) If $\Gamma \vdash_{AF2} t : \forall X A$, then, for every formulas G, $\Gamma \vdash_{AF2} t : A[G/X]$.

  (8) If $\Gamma \vdash_{AF2} t : A[u/x]$, and $u \approx v$, then $\Gamma \vdash_{AF2} t : A[v/x]$.

This typed λ-calculus system is called $AF2$ (for Arithmétique Fonctionnelle du second ordre).

Theorem 2.1 *(see [2]) The $AF2$ type system has the following properties :*
1) Type is preserved during reduction.
2) Typable λ-terms are strongly normalizable.

We present now a syntaxical property of system $AF2$ that we will use afterwards.

Theorem 2.2 *(see [8]) If in the typing we go from $\Gamma \vdash_{AF2} t : A$ to $\Gamma \vdash_{AF2} t : B$, then we may assume that we begin by the $\forall$-elimination rules, then by the equationnal rule, and finally by the $\forall$-introduction rules.*

- We define on the set of types the two binary relations $\lhd$ and $\approx$ as the least reflexive and transitive binary relations such that :

  - $\forall x A \lhd A[u/x]$, if u is a term of language ;
  - $\forall X A \lhd A[F/X]$, if F is a formula of language ;
  - $A \approx B$ if and only if $A = C[u/x]$, $B = C[v/x]$, and $u \approx v$.

3 Pure and typed λC-calculus

3.1 The $C2$ type system

We present in this section the J-L. Krivine's classical type system.

- We add a constant C to the pure λ-calculus and we denote by ΛC the set of new terms also called λC-terms. We consider the following rules of reduction, called rules of head C-reduction.

  1) $(\lambda x u)tt_1...t_n \rightarrow (u[t/x])t_1...t_n$ for every $u, t, t_1, ..., t_n \in \Lambda C$.

  2) $(C)tt_1...t_n \rightarrow (t)\lambda x(x)t_1...t_n$ for every $t, t_1, ..., t_n \in \Lambda C$, x being a λ-variable not appearing in $t_1, ..., t_n$.

- For any λC-terms t, t', we shall write $t \succ_C t'$ if t' is obtained from t by applying these rules finitely many times. We say that t' is obtained from t by head C-reduction.

- A λC-term t is said β-normal if and only if t does not contain a β-redex.

- A λC-term t is said C-solvable if and only if $t \succ_C (f)t_1, ..., t_n$ where f is a variable.

It is easy to prove that : if $t \succ_C t'$, then, for any substitution σ, $\sigma(t) \succ_C \sigma(t')$.

- We add to the $AF2$ type system the new following rule :

$$(0)\ \Gamma \vdash C : \forall X\{\neg\neg X \rightarrow X\}$$

  This rule axiomatizes the classical logic over the intuitionistic logic. We call $C2$ the new type system, and we write $\Gamma \vdash_{C2} t : A$ if t is of type A in the context Γ.

It is clear that $\Gamma \vdash_{C2} t : A$ if and only if $\Gamma, C : \forall X\{\neg\neg X \rightarrow X\} \vdash_{AF2} t : A$.

Theorem 3.1 *(see [6])*
1) If $\Gamma \vdash_{C2} t : A$, *and* $t \rightarrow_\beta t'$, *then* $\Gamma \vdash_{C2} t' : A$.
2) If $\Gamma \vdash_{C2} t :\bot$, *and* $t \succ_C t'$, *then* $\Gamma \vdash_{C2} t' :\bot$.
3) If A *is an atomic type, and* $\Gamma \vdash_{C2} t : A$, *then* t *is* C*-solvable.*

3.2 The $M2$ type system

In this section, we present the system $M2$. This system allows essentialy to distinguish between classical proofs and intuitionistic proofs

We assume that for every integer n, there is a countable set of special n-ary second order variables denoted by X_C, Y_C, Z_C...., and called classical variables.

Let X be an n-ary predicate variable or predicate symbol. A type A is said to be ending with X if and only if A is obtained by the following rules :

- $X(t_1, ..., t_n)$ ends with X;

- If B ends with X, then $A \to B$ ends with X for every type A ;

- If A ends with X, then $\forall v A$ ends with X for every variable v.

A type A is said to be a classical type if and only if A ends with $\bot$ or a classical variable.

We add to the $AF2$ type system the new following rules :

$(0')$ $\Gamma \vdash C : \forall X_C\{\neg\neg X_C \to X_C\}$

$(6')$ If $\Gamma \vdash t : A$, and X_C has no free occurence in Γ, then $\Gamma \vdash t : \forall X_C A$.

$(7')$ If $\Gamma \vdash t : \forall X_C A$, and G is a classical type, then $\Gamma \vdash t : A[G/X_C]$.

We call $M2$ the new type system, and we write $\Gamma \vdash_{M2} t : A$ if t is of type A in the context Γ.

We extend the definition of $\lhd$ by : $\forall X_C A \lhd A[G/X_C]$ if G is a classical type.

Lemma 3.1 *If A is a classical type and $A \lhd B$ (or $A \approx B$), then B is a classical type.*

Proof Easy. □

3.3 The logical properties of $M2$

We denote by $LAF2$, $LC2$, and $LM2$ the underlying logic systems of respectively $AF2$, $C2$, and $M2$ type systems.

With each classical variable X_C, we associate a special variable X^* of $AF2$ having the same arity as X_C. For each formula A of $LM2$, we define the formula A^* of $LAF2$ in the following way :

- If $A = D(t_1, ..., t_n)$ where D is a predicate symbol or a predicate variable, then $A^*=A$;

- If $A = X_C(t_1, ..., t_n)$, then $A^*= \neg X^*(t_1, ..., t_n)$;

- If $A = B \to C$, then $A^*= B^* \to C^*$;

- If $A = \forall x B$, then $A^*=\forall x B^*$.

- If $A = \forall X B$, then $A^*=\forall X B^*$.

- If $A = \forall X_C B$, then $A^*=\forall X^* B^*$.

A^* is called the Gődel translation of A.

Lemma 3.2 *If G is a classical type of $LM2$, then $\vdash_{LAF2} \neg\neg G^* \longleftrightarrow G^*$.*

Proof It is easy to prove that $\vdash_{LAF2} G^* \to \neg\neg G^*$.
We prove $\vdash_{LAF2} \neg\neg G^* \to G^*$ by induction on G.

- If $G = \bot$, then $G^*=\bot$, and $\vdash_{LAF2} ((\bot \to \bot) \to \bot) \to \bot$.

- If $G = X_C(t_1, ..., t_n)$, then $G^*{=}\neg X^*(t_1, ..., t_n)$, and $\vdash_{LAF2} \neg\neg\neg X^*(t_1, ..., t_n) \rightarrow \neg X^*(t_1, ..., t_n)$.

- If $G = A \rightarrow B$, then B is a classical type and $G^* = A^* \rightarrow B^*$. By the induction hypothesis, we have $\vdash_{LAF2} \neg\neg B^* \rightarrow B^*$. Since $\vdash_{LAF2} \neg\neg(A^* \rightarrow B^*) \rightarrow (\neg\neg A^* \rightarrow \neg\neg B^*)$, we check easily that $\vdash_{LAF2} \neg\neg(A^* \rightarrow B^*) \rightarrow (A^* \rightarrow B^*)$.

- If $G = \forall v G'$ where $v = x$ or $v = X$, then G' is a classical type and $G^*{=}\forall v G'^*$. By the induction hypothesis, we have $\vdash_{LAF2} \neg\neg G'^* \rightarrow G'^*$. Since $\vdash_{LAF2} \neg\neg\forall v G'^* \rightarrow \forall v \neg\neg G'^*$, we check easily that $\vdash_{LAF2} \neg\neg\forall v G'^* \rightarrow \forall v G'^*$.

- If $G = \forall X_C G'$, then G' is a classical type and $G^*{=}\forall X^* G'^*$. By the induction hypothesis, we have $\vdash_{LAF2} \neg\neg G'^* \rightarrow G'^*$. Since $\vdash_{LAF2} \neg\neg\forall X^* G'^* \rightarrow \forall X^* \neg\neg G'^*$, we check easily that $\vdash_{LAF2} \neg\neg\forall X^* G'^* \rightarrow \forall X^* G'^*$. □

Lemma 3.3 *Let A, G be formulas of $LM2$, t a term, x a first order variable, and X a second order variable. We have :*
1) $(A[t/x])^* = A^*[t/x]$.
2) $(A[G/X])^* = A^*[G^*/X]$.

Proof By induction on A. □

Lemma 3.4 *Let A be a formula of $LM2$, G a classical type, and X_C a classical variable.*
$\vdash_{LAF2} (A[G/X_C])^* \longleftrightarrow A^*[\neg G^*/X_C]$.

Proof By induction on A.

- If $A = D(t_1, ..., t_n)$ where D is a predicate variable or a predicate symbol, then $A^*{=}A$, and $\vdash_{LAF2} A \longleftrightarrow A$.

- If $A = X_C(t_1, ..., t_n)$, then $A^*{=}\neg X^*(t_1, ..., t_n)$, and, by Lemma 3.2, $\vdash_{LAF2} \neg\neg G^* \longleftrightarrow G^*$.

- If $A = B \rightarrow C$, then $A^* = B^* \rightarrow C^*$. By the induction hypothesis, we have $\vdash_{LAF2} (B[G/X_C])^* \longleftrightarrow B^*[\neg G^*/X_C]$ and $\vdash_{LAF2} (C[G/X_C])^* \longleftrightarrow C^*[\neg G^*/X_C]$. Therefore $\vdash_{LAF2} \{(B[G/X_C])^* \rightarrow (B[G/X_C])^*\} \longleftrightarrow \{B^*[\neg G^*/X_C] \rightarrow C^*[\neg G^*/X_C]\}$.

- If $A = \forall v A'$, where $v = x$ or $v = X$, then $A^*{=}\forall v A'^*$. By the induction hypothesis, we have $\vdash_{LAF2} (A'[G/X_C])^* \longleftrightarrow A'^*[\neg G^*/X_C]$. Therefore $\vdash_{LAF2} (\forall v A'[G/X_C])^* \longleftrightarrow \forall v A'^*[\neg G^*/X_C]$.

- If $A = \forall Y_C A'$, then $A^*{=}\forall Y^* A'^*$. By the induction hypothesis, we have $\vdash_{LAF2} (A'[G/X_C])^* \longleftrightarrow A'^*[\neg G^*/X_C]$. Therefore $\vdash_{LAF2} (\forall Y_C A'[G/X_C])^* \longleftrightarrow (\forall Y_C A')^*[\neg G^*/X_C]$. □

Theorem 3.2 *If $A_1, ..., A_n \vdash_{LM2} A$, then $A_1{}^*, ..., A_n{}^* \vdash_{LAF2} A^*$.*

Proof By induction on the proof of A and using Lemmas 3.2, 3.3, and 3.4. □

Corollary 3.1 *Let $A, A_1, ..., A_n$ be formulas of $LAF2$.*
$A_1, ..., A_n \vdash_{LM2} A$ if and only if $A_1, ..., A_n \vdash_{LAF2} A$.

Proof We use Theorem 3.2. □

With each predicate variable X of $C2$, we associate a classical variable X_C having the same arity as X. For each formula A of $LC2$, we define the formula A^C of $M2$ in the following way :

- If $A = D(t_1, ..., t_n)$ where D is a constant symbol, then $A^C = A$;

- If $A = X(t_1, ..., t_n)$ where X is a predicate symbol, then $A^C = X_C(t_1, ..., t_n)$;

- If $A = B \to C$, then $A^C = B^C \to C^C$;

- If $A = \forall x B$, then $A^C = \forall x B^C$;

- If $A = \forall X B$, then $A^C = \forall X_C B^C$.

A^C is called the classical translation of A.

Theorem 3.3 *Let $A_1, ..., A_n, A$ be formulas of $LC2$.*
$A_1, ..., A_n \vdash_{LC2} A$ if and only if $A_1^C, ..., A_n^C \vdash_{LM2} A^C$.

Proof By induction on the proof of A. □

4 Properties of $M2$ type system

By corollary 3.1, we have that a formula is provable in system $LAF2$ if and only if it is provable in system $LC2$. This resultat is not longer valid if we decorate the demonstrations by terms. We will give some conditions on the formulas in order to obtain such a result.

We define two sets of types of $AF2$ type system : Ω^+ (set of $\forall$-positive types), and Ω^- (set of $\forall$-negative types) in the following way :

- If A is an atomic type, then $A \in \Omega^+$, and $A \in \Omega^-$;

- If $T \in \Omega^+$, and $T' \in \Omega^-$, then, $T' \to T \in \Omega^+$, and $T \to T' \in \Omega^-$;

- If $T \in \Omega^+$, then $\forall x T \in \Omega^+$;

- If $T \in \Omega^-$, then $\forall x T \in \Omega^-$;

- If $T \in \Omega^+$, then $\forall X T \in \Omega^+$;

- If $T \in \Omega^-$, and X has no free occurence in T, then $\forall X T \in \Omega^-$.

Lemma 4.1 *1) If $A \in \Omega^+$ (resp. $A \in \Omega^-$) and $A \approx B$, then $B \in \Omega^+$ (resp. $B \in \Omega^-$).*
2) If $A \in \Omega^-$ and $A \lhd B \to C$, then $B \in \Omega^+$ and $C \in \Omega^-$.

Proof Easy. □

Theorem 4.1 *Let $A_1, ..., A_n$ be $\forall$-negative types, A a $\forall$-positive type of $AF2$ which does not end with $\bot$, $B_1, ..., B_m$ classical types, and t a β-normal λC-term.*
If $\Gamma = x_1 : A_1, ..., x_n : A_n, y_1 : B_1, ..., y_m : B_m \vdash_{M2} t : A$, then t is a normal λ-term, and $x_1 : A_1, ..., x_n : A_n \vdash_{AF2} t : A$.

Proof We argue by induction on t.

- If t is a variable, we have two cases :

- If $t = x_i$ $1 \leq i \leq n$, this is clear.

- If $t = y_j$ $1 \leq j \leq m$, then $A = \forall \overline{v} B$ where $B_j \lhd B'_j$ and $B'_j \approx B$. Therefore, by Lemma 3.1, A is a classical type. A contradiction.

- If $t = \lambda x u$, then $\Gamma, x : E \vdash_{M2} u : F$, and $A = \forall \overline{v}(E' \to F')$ where $E \approx E'$, $F \approx F'$ and $\overline{v}$ does not appear in Γ. First, by Lemma 4.1, $E \in \Omega^-$ and $F \in \Omega^+$, and then, by the induction hypothesis, u is a normal λ-term, and $x_1 : A_1, ..., x_n : A_n, x : E \vdash_{AF2} u : F$. Therefore t is a normal λ-term, and $x_1 : A_1, ..., x_n : A_n \vdash_{AF2} t : A$.

- If $t = (x)u_1...u_r$ $r \geq 1$, we have two cases :

  - If $t = x_i$ $1 \leq i \leq n$, then $A_i \lhd B_1 \to C_1$, $C'_i \lhd B_{i+1} \to C_{i+1}$ $1 \leq i \leq r-1$, $C'_r \lhd D$, $A = \forall v D'$, where $C'_i \approx C_i$ $1 \leq i \leq r$, $D' \approx D$, and $\Gamma \vdash_{M2} u_i : B_i$ $1 \leq i \leq r$. Since A_i is a $\forall$-negative types, we prove (by induction and using Lemma 4.1) that for all $1 \leq i \leq r$ B_i is a $\forall$-positive types. By the induction hypothesis we have u_i is a normal λ-term, and $x_1 : A_1, ..., x_n : A_n \vdash_{AF2} u_i : B_i$. Therefore t is a normal λ-term, and $x_1 : A_1, ..., x_n : A_n \vdash_{AF2} t : A$.

  - If $t = y_j$ $1 \leq j \leq m$, then $B_j \lhd B_1 \to C_1$, $C'_i \lhd B_{i+1} \to C_{i+1}$ $1 \leq i \leq r-1$, $C'_r \lhd D$, $A = \forall v D'$, where $C'_i \approx C_i$ $1 \leq i \leq r$, $D' \approx D$, and $\Gamma \vdash_{M2} u_i : B_i$ $1 \leq i \leq r$. Therefore, by Lemma 3.1, A is a classical type. A contradiction.

- If $t = (C)uu_1...u_r$ $r \geq 0$, then there is a classical type E such that $\Gamma \vdash_{M2} u : \neg\neg E$, $E \lhd B_1 \to C_1$, $C'_i \lhd B_{i+1} \to C_{i+1}$ $1 \leq i \leq r-1$, $C'_r \lhd D$, $A = \forall v D'$, where $C'_i \approx C_i$ $1 \leq i \leq r$, $D' \approx D$, and $\Gamma \vdash_{M2} u_i : B_i$ $1 \leq i \leq r$. Therefore, by Lemma 3.1, A is a classical type. A contradiction. □

Corollary 4.1 *Let A be a $\forall$-positive type of $AF2$ and t a β-normal λC-term.*
If $\vdash_{M2} t : A$, then t is a normal λ-term, and $\vdash_{AF2} t : A$.

Proof We use Theorem 4.1. □

As for relation betwen the systems $C2$ and $M2$, we have the following result.

Theorem 4.2 *Let $A_1, ..., A_n, A$ be types of $C2$, and t a λC-term.*
$A_1, ..., A_n \vdash_{C2} t : A$ if and only if $A_1^C, ..., A_n^C \vdash_{M2} t : A^C$.

Proof By induction on the typing of t. □

5 The integers

- Each data type can be defined by a second order formula. For example, the type of integers is the formula : $N[x] = \forall X\{X(0), \forall y(X(y) \to X(sy)) \to X(x)\}$ where X is a unary predicate variable, 0 is a constant symbol for zero, and s is a unary function symbol for successor. The formula $N[x]$ means semantically that x is an integer if and only if x belongs to each set X containing 0 and closed under the successor function s.
  The λ-term $\underline{0} = \lambda x \lambda f x$ is of type $N[0]$ and represents zero.
  The λ-term $\underline{s} = \lambda n \lambda x \lambda f (f)((n)x)f$ is of type $\forall y(N[y] \to N[s(y)])$ and represents the successor function.

- A set of equations E is said to be adequate with the type of integers if and only if :

  - $s(a) \not\approx 0$;

  - If $s(a) \approx s(b)$, then so is $a \approx b$.

  In the rest of the paper, we assume that all sets of equations are adequate with the type of integers.

- For each integer n, we define the Church integer $\underline{n}$ by $\underline{n} = \lambda x \lambda f (f)^n x$.

5.1 The integers in $AF2$

The system $AF2$ has the property of the unicity of integers representation.

Theorem 5.1 *(see [2]) Let n be an integer. If $\vdash_{AF2} t : N[s^n(0)]$, then $t \simeq_\beta \underline{n}$.*

The propositional trace $N = \forall X\{X, (X \to X) \to X\}$ of $N[x]$ also defines the integers.

Theorem 5.2 *(see [2]) If $\vdash_{AF2} t : N$, then, for a certain n, $t \simeq_\beta \underline{n}$.*

Remark A very important property of data type is the following (we express it for the type of integers) : in order to get a program for a function $f : N \to N$ it is sufficient to prove $\vdash \forall x(N[x] \to N[f(x)])$. For example a proof of $\vdash \forall x(N[x] \to N[p(x)])$ from the equations $p(0) = 0$, $p(s(x)) = x$ gives a λ-term for the predecessor in Church intergers (see [2]). □

5.2 The integers in $C2$

The situation in system $C2$ is more complex. In fact, in this system the property of unicity of integers representation is lost and we have only one operational characterization of these integers.

Let n be an integer. A classical integer of value n is a closed λC-term θ_n such that $\vdash_{C2} \theta_n : N[s^n(0)]$.

Theorem 5.3 *(see [6] and [12]) Let n be an integer, and θ_n a classical integer of value n.*

- if $n = 0$, then, for every distinct variables x, g, y : $(\theta_n)xgy \succ_C (x)y$;

- if $n \neq 0$, then there is $m \geq 1$ and a mapping $I : \{0, ..., m\} \to N$, such that for every distinct variables $x, g, x_0, x_1, ..., x_m$:

$(\theta_n)xgx_0 \succ_C (g)t_1 x_{r_0}$;

$(t_i)x_i \succ_C (g)t_{i+1}x_{r_i}$ $1 \leq i \leq m$;

$(t_m)x_m \succ_C (x)x_{r_m}$;

where $I(0) = n$, $I(r_m) = 0$, and $I(i+1) = I(r_i) - 1$ $0 \leq i \leq m-1$.

We will generalize this result.

Let O be a particular unary predicate symbol. The typed system $C2_O$ is the typed system $C2$ where we replace the rules (2) and (7) by :

(2_O) If $\Gamma, x : A \vdash_{C2_O} t : B$, A and B are not ending with O, then $\Gamma \vdash_{C2_O} \lambda x t : A \to B$.

(7_O) If $\Gamma \vdash_{C2_O} t : \forall X A$, and G is not ending with O, then $\Gamma \vdash_{C2_O} t : A[G/X]$.

We define on the types of $C2_O$ a binary relation $\lhd_O$ as the least reflexive and transitive binary relation such that :

$\forall x A \lhd_O A[u/x]$ if u is a term of language ;

$\forall X A \lhd_O A[G/X]$ if G is a type which is not ending with O.

Lemma 5.1 *a) If* $\Gamma \vdash_{C2_O} t :\bot$, *and* $t \succ_C t'$, *then* $\Gamma \vdash_{C2_O} t' :\bot$.
b) If $\Gamma \vdash_{C2_O} t : A$, *and* A *is an atomic type, then* t *is* C*-solvable.*

Proof a) It is enough to do the proof for one step of reduction. We have two cases :

- If $t = (\lambda x u) v v_1 ... v_m$, then $t' = (u[v/x]) v_1 ... v_m$, $\Gamma, x : F \vdash_{C2_O} u : G$, F and G are not ending with O, $G' \lhd_O F_1 \to G_1$, $G'_j \lhd_O F_{j+1} \to G_{j+1}$ $1 \leq j \leq m-1$, $G_m \approx \bot$, $G_j \approx G'_j$ $1 \leq j \leq m-1$, $\Gamma \vdash_{C2_O} v : F$, and $\Gamma \vdash_{C2_O} v_j : F_j$ $1 \leq j \leq m$. It is easy to check that $\Gamma \vdash_{C2_O} u[v/x] : G$, then $\Gamma \vdash_{C2_O} t' :\bot$.

- If $t = (C) v v_1 ... v_m$, then $t' = (v) \lambda x (x) v_1 ... v_m$, and there is a type A which is not ending with O such that : $A' \lhd_O F_1 \to G_1$, $G'_j \lhd_O F_{j+1} \to G_{j+1}$ $1 \leq j \leq m-1$, $G_m \approx \bot$, $A \approx A'$, $G_j \approx G'_j$ $1 \leq j \leq m$, $\Gamma \vdash_{C2_O} v : \neg\neg A$, and $\Gamma \vdash_{C2_O} v_j : F_j$ $1 \leq j \leq m$. It is easy to check that $\Gamma, x : A \vdash_{C2_O} (x) v_1 ... v_m :\bot$, but A is not ending with O, then $\Gamma \vdash_{C2_O} \lambda x (x) v_1 ... v_m : \neg A$, and $\Gamma \vdash_{C2_O} t' :\bot$.

b) Indeed, a typing of $C2_O$ may be seen as a typing of $C2$. □

Lemma 5.2 *a) If* $\Gamma \vdash_{C2_O} t : O(a)$, *and* $t \succ_C t'$, *then* $t = t'$.
b) If $\Gamma = y_1 : A_1, ..., y_n : A_n, x_1 : O(a_1), ..., x_m : O(a_m) \vdash_{C2_O} t : O(a)$, *and all* A_i $1 \leq i \leq n$ *are not ending with* O, *then* t *is one of* x_i, *and* $a_i \approx a$ $1 \leq i \leq n$.

Proof a) It is enough to do the proof for one step of reduction. We have two cases :

- If $t = (\lambda x u) v v_1 ... v_m$, then $t' = (u[v/x]) v_1 ... v_m$, $\Gamma, x : F \vdash_{C2_O} u : G$, F and G are not ending with O, $G' \lhd_O F_1 \to G_1$, $G'_j \lhd_O F_{j+1} \to G_{j+1}$ $1 \leq j \leq m-1$, $G_m \approx O(a)$, $G_j \approx G'_j$ $1 \leq j \leq m-1$, $\Gamma \vdash_{C2_O} v : F$, and $\Gamma \vdash_{C2_O} v_j : F_j$ $1 \leq j \leq m$. Therefore G_j $1 \leq j \leq m$ is not ending with O, which is impossible since $G_m \approx O(a)$.

- If $t = (C) v v_1 ... v_m$, then $t' = (v) \lambda x (x) v_1 ... v_m$, and there is a type A which is not ending with O such that : $A' \lhd_O F_1 \to G_1$, $G'_j \lhd_O F_{j+1} \to G_{j+1}$ $1 \leq j \leq m-1$, $G_m \approx O(a)$, $A \approx A'$, $G_j \approx G'_j$ $1 \leq j \leq m$, $\Gamma \vdash_{C2_O} v : \neg\neg A$, and $\Gamma \vdash_{C2_O} v_j : F_j$ $1 \leq j \leq m$. A is not ending with O, therefore G_j $1 \leq j \leq m$ is not ending with O, which is impossible since $G_m \approx O(a)$.

b) By Lemma 5.1, we have $t \succ_C (f) t_1 ... t_r$, and, by a), $t = (f) t_1 ... t_r$. Therefore $\Gamma \vdash_{C2_O} (f) t_1 ... t_r : O(a)$.

- If $f = x_i$ $1 \leq i \leq m$, then $r = 0$, $t = x_i$, and $O(a_i) \approx O(a)$, then $a_i \approx a$.

- If $f = y_j$ $1 \leq j \leq k$, then $A_j \lhd_O F_1 \to G_1$, $G'_k \lhd_O F_{k+1} \to G_{k+1}$ $1 \leq k \leq r-1$, $G_r \approx O(a)$, $G_k \approx G'_k$ $1 \leq k \leq r$, and $\Gamma \vdash_{C2_O} t_k : F_k$ $1 \leq k \leq r$. Since A_j is not ending with O, then G_k $1 \leq k \leq r$ is not ending with O, which is impossible since $Cr \approx O(a)$. □

Let V be the set of variables of λC-calculus.
Let P be an infinite set of constants called stack constants [2].
We define a set of λC-terms ΛCP by :

- If $x \in V$, then $x \in \Lambda CP$;

- If $t \in \Lambda CP$, and $x \in V$, then $\lambda xt \in \Lambda CP$;

- If $t \in \Lambda CP$, and $u \in \Lambda CP \bigcup P$, then $(t)u \in \Lambda CP$.

In other words, $t \in \Lambda CP$ if and only if the stack constants are in argument positions in t.

Let σ be a function defined on $V \bigcup P$ such that :

- If $x \in V$, then $\sigma(x) \in \Lambda CP$;

- If $p \in P$, then $\sigma(p) = \overline{t} = t_1, ..., t_n$, $n \geq 0$, $t_i \in \Lambda CP \bigcup P$ $1 \leq i \leq n$.

We define $\sigma(t)$ for all $t \in \Lambda CP$ by :

- $\sigma((u)v) = (\sigma(u))\sigma(v)$ if $v \notin P$;

- $\sigma(\lambda xu) = \lambda x\sigma(u)$;

- $\sigma((t)p) = (t)\overline{t}$ if $\sigma(p) = \overline{t}$.

σ is said to be a P-substitution.

We consider, on the set ΛCP, the following rules of reduction :

1) $(\lambda xu)tt_1...t_n \rightarrow (u[t/x])t_1...t_n$ for all $u, t \in \Lambda CP$ and $t_1, ..., t_n \in \Lambda CP \bigcup P$;

2) $(C)tt_1...t_n \rightarrow (t)\lambda x(x)t_1...t_n$ for all $t \in \Lambda CP$ and $t_1, ..., t_n \in \Lambda CP \bigcup P$, and x being λ-variable not appearing in $t_1, ..., t_n$.

For any $t, t' \in \Lambda CP$, we shall write $t \rhd_C t'$, if t' is obtained from t by applying these rules finitely many times.

Lemma 5.3 *If $t \rhd_C t'$, then $\sigma(t) \rhd_C \sigma(t')$ for all P-substitution σ.*

Proof Easy. □

Lemma 5.4 *Let $t \in \Lambda CP$ such that the stack constants of t are among $p_1, ..., p_m$.*
If $t \succ_C t'$, and $\Gamma = \Gamma', p_1 : O(a_1), ..., p_m : O(a_m) \vdash_{C2_O} t : \bot$, then $t' \in \Lambda CP$ and $t \rhd_C t'$.

Proof It is enough to do the proof for one step of reduction. We have two cases :

- If $t = (\lambda xu)vv_1...v_m$, then, $t' = (u[v/x])v_1...v_m$, $\Gamma, x : F \vdash_{C2_O} u : G$, F and G is not ending with O, and $\Gamma \vdash_{C2_O} v : F$. Therefore $u, v \in \Lambda CP$, and so $t' \in \Lambda CP$ and $t \rhd_C t'$.

[2] The notion of stack constants taken from a manuscript of J-L. Krivine

- If $t = (C)vv_1...v_m$, then, $t' = (v)\lambda x(x)v_1...v_m$, and there is a type A which is not ending with O such that $\Gamma \vdash_{C2_O} v : \neg\neg A$. Therefore $v \in \Lambda CP$, and so $t' \in \Lambda CP$ and $t \rhd_C t'$. □

Theorem 5.4 *Let n be an integer, θ_n a classical integer of value n, and x, g two distinct variables.*

- If $n = 0$, then for every stack constant p, we have : $(\theta_n)xgp \succ_C (x)p$.

- If $n \neq 0$, then there is $m \geq 1$, and a mapping $I : \{0, ..., m\} \to N$, such that for all distinct stack constants $p_0, p_1, ..., p_m$, we have :

$(\theta_n)xgp_0 \succ_C (g)t_1p_{r_0}$;

$(t_i)p_i \succ_C (g)t_{i+1}p_{r_i}$ $1 \leq i \leq m-1$;

$(t_m)p_m \succ_C (x)p_{r_m}$

where $I(0) = n$, $I(r_m) = 0$, and $I(i+1) = I(r_i) - 1$ $0 \leq i \leq m-1$.

Proof We denote, in this proof, the term $s^i(0)$ by i.
If $\vdash_{C2} \theta_n : N[n]$, then $\vdash_{C2_O} \theta_n : [O(0) \to\bot], \forall y\{[O(y) \to\bot] \to [O(sy) \to\bot]\}, O(n) \to\bot$, then $\Gamma_1 = x : O(0) \to\bot, g : \forall y\{[O(y) \to\bot] \to [O(sy) \to\bot]\}, p_0 : O(n) \vdash_{C2_O} (\theta_n)xgp_0 :\bot$, therefore, by Lemma 5.1, $(\theta_n)xgp_0$ is C-solvable, and three cases may be seen :

- If $(\theta_n)xgp_0 \succ_C (p_0)t_1...t_r$, then $r = 0$, and there is a term a, such that $O(a) \approx\bot$. This is impossible.

- If $(\theta_n)xgp_0 \succ_C (x)t_1...t_r$, then $r = 1$, and $\Gamma_1 \vdash_{C2_O} t_1 : O(0)$. Therefore, by Lemma 5.2, $t_1 = p_0$, and so $n = 0$.

- If $(\theta_n)xgp_0 \succ_C (g)t_1...t_r$, then $r = 2$, $\Gamma_1 \vdash_{C2_O} t_1 : O(a) \to\bot$, $\Gamma_1 \vdash_{C2_O} t_2 : O(s(a'))$, and $a \approx a'$. By Lemma 5.2, we have $t_2 = p_0$, and $s(a') \approx n$, then $a \approx n-1$. Therefore $(\theta_n)xgp_0 \succ_C (g)t_1p_0$, and $\Gamma_1 \vdash_{C2_O} t_1 : O(n-1) \to\bot$. Let $I(0) = n$.

We prove that : if $\Gamma_i = g : \forall y\{[O(y) \to\bot] \to [O(sy) \to\bot]\}, x : O(0) \to\bot, p_0 : O(I(0)), ..., p_i : O(I(i)) \vdash_{C2_O} (t_i)p_i :\bot$, then :
$(t_i)p_i \succ_C (g)t_{i+1}p_{r_i}$, and $\Gamma_i \vdash_{C2_O} t_{i+1} : O(I(r_i) - 1) \to\bot$
or
$(t_i)p_i \succ_C (x)p_{r_i}$, and $I(r_i) = 0$.
$\Gamma_i \vdash_{C2_O} (t_i)p_i :\bot$, therefore, by Lemma 5.1, $(t_i)p_i$ est C-solvable, and three cases may be seen :

- If $(t_i)p_i \succ_C (p_j)u_1...u_r$ $0 \leq j \leq i$, then $r = 0$, and there is a term a, such that $O(a) \approx\bot$. This is impossible.

- If $(t_i)p_i \succ_C (x)u_1...u_r$, then $r = 1$, and $\Gamma_i \vdash_{C2_O} u_1 : O(0)$. Therefore, by Lemma 5.2, $u_1 = p_{r_i}$, and $I(r_i) = 0$.

- If $(t_i)p_i \succ_C (g)u_1...u_r$, then $r = 2$, $\Gamma_i \vdash_{C2_O} u_1 : O(a) \to\bot$, $\Gamma_i \vdash_{C2_O} u_2 : O(s(a'))$, and $a \approx a'$. By Lemma 5.2, we have $u_2 = p_{r_i}$, and $s(a') \approx I(r_i)$, then $a \approx I(r_i) - 1$. Therefore $(t_i)p_i \succ_C (g)t_{i+1}p_{r_i}$, and $\Gamma_i \vdash_{C2_O} t_{i+1} : O(I(r_i) - 1) \to\bot$. Let $I(i+1) = I(r_i) - 1$.

This construction always terminates. Indeed, if not, the λC-term $(((\theta_n)\lambda xx)\lambda xx)p_0$ is not C-solvable. This is impossible, since $p_0 :\bot \vdash_{C2} (((\theta_n)\lambda xx)\lambda xx)p_0 :\bot$. □

Corollary 5.1 *Let n be an integer, θ_n a classical integer of value n, and x, g two distinct variables.*

- If $n = 0$, then, for every stack constant p, we have : $(\theta_n)xgp \rhd_C (x)p$.

- If $n \neq 0$, then there is $m \geq 1$, and a mapping $I : \{0, ..., m\} \to N$, such that for all distinct stack constants $p_0, p_1, ..., p_m$, we have :

$(\theta_n)xgp_0 \rhd_C (g)t_1 p_{r_0}$;

$(t_i)p_i \rhd_C (g)t_{i+1}p_{r_i}$ $1 \leq i \leq m-1$;

$(t_m)p_m \rhd_C (x)p_{r_m}$

where $I(0) = n$, $I(r_m) = 0$, and $I(i+1) = I(r_i) - 1$ $0 \leq i \leq m-1$.

Proof We use Lemma 5.4. □

Corollary 5.2 *Let n be an integer, and θ_n a classical integer of value n.*

- If $n = 0$, then, for every $\lambda C - terms$ $a, F, \overline{u}$, we have : $(\theta_n)aF\overline{u} \succ_C (a)\overline{u}$.

- If $n \neq 0$, then there is $m \geq 1$, and a mapping $I : \{0, ..., m\} \to N$, such that for all $\lambda C - terms$ $a, F, \overline{u_0}, \overline{u_1}, ..., \overline{u_m}$, we have :

$(\theta_n)aF\overline{u_0} \succ_C (g)t_1\overline{u_{r_0}}$;

$(t_i)\overline{u_i} \succ_C (g)t_{i+1}\overline{u_{r_i}}$ $1 \leq i \leq m-1$;

$(t_m)\overline{u_m} \succ_C (a)\overline{u_{r_m}}$

where $I(0) = n$, $I(r_m) = 0$, and $I(i+1) = I(r_i) - 1$ $0 \leq i \leq m-1$.

Proof We use Lemma 5.3. □

5.3 The integers in $M2$

According to the results of section 4, we can obtain some results concerning the integers in the system $M2$.

Theorem 5.5 *Let n be an integer. If $\vdash_{M2} t : N[s^n(0)]$, then, $t \simeq_\beta \underline{n}$.*

Proof We use Theorem 4.1. □

Let n be an integer. By Theorem 4.2, a classical integer of value n is a closed λC-term θ_n such that $\vdash_{M2} \theta_n : N^C[s^n(0)]$.

Theorem 5.6 *Let n be an integer, θ_n a classical integer of value n, and x, g two distinct variables.*

- If $n = 0$, then, for every stack constant p, we have : $(\theta_n)xgp \rhd_C (x)p$.

- If $n \neq 0$, then there is $m \geq 1$, and a mapping $I : \{0, ..., m\} \to N$, such that for all distinct stack constants $p_0, p_1, ..., p_m$, we have :

$(\theta_n)xgp_0 \rhd_C (g)t_1 p_{r_0}$;

$(t_i)p_i \rhd_C (g)t_{i+1}p_{r_i}$ $1 \leq i \leq m-1$;

$(t_m)p_m \rhd_C (x)p_{r_m}$

where $I(0) = n$, $I(r_m) = 0$, *and* $I(i+1) = I(r_i) - 1$ $0 \leq i \leq m-1$.

Proof We use Theorem 4.2. □

6 Storage operators

6.1 Storage operators for Church integers

Let T be a closed λ-term. We say that T is a storage operator for Church integers if and only if for every $n \geq 0$, there is a λ-term $\tau_n \simeq_\beta \underline{n}$, such that for every λ-term $\theta_n \simeq_\beta \underline{n}$, there is a substitution σ, such that $(T)\theta_n f \succ (f)\sigma(\tau_n)$.

Examples If we take :
$T_1 = \lambda n((n)\delta)G$ where $G = \lambda x\lambda y(x)\lambda z(y)(\underline{s})z$ and $\delta = \lambda f(f)\underline{0}$
$T_2 = \lambda n\lambda f(((n)f)F)\underline{0}$ where $F = \lambda x\lambda y(x)(\underline{s})y$,
then it is easy to check that : for every $\theta_n \simeq_\beta \underline{n}$, $(T_i)\theta_n f \succ (f)(\underline{s})^n\underline{0}$ ($i = 1$ or 2) (see [3] and [8]).
Therefore T_1 and T_2 are storage operators for Church integers. □

It is a remarkable fact that we can give simple types to storage operators for Church integers. We first define the simple Gődel translation F^* of a formula F : it is obtained by replacing in the formula F, each atomic formula A by $\neg A$. For example :

$$N^*[x] = \forall X\{\neg X(0), \forall y(\neg X(y) \to \neg X(sy)) \to \neg X(x)\}$$

It is well known that, if F is provable in classical logic, then F^* is provable in intuitionistic logic.

We can check that $\vdash_{AF2} T_1, T_2 : \forall x\{N^*[x] \to \neg\neg N[x]\}$. And, in general, we have the following Theorem :

Theorem 6.1 *(see [3] and [10]) If* $\vdash_{AF2} T : \forall x\{N^*[x] \to \neg\neg N[x]\}$, *then* T *is a storage operator for Church integers.*

6.2 Storage operators for classical integers

The storage operators play an important role in classical type systems. Indeed, they can be used to find the value of a classical integer.

Theorem 6.2 *(see [6] and [7]) If* $\vdash_{AF2} T : \forall x\{N^*[x] \to \neg\neg N[x]\}$, *then for every* $n \geq 0$, *there is a* λ*-term* $\tau_n \simeq_\beta \underline{n}$, *such that for every classical integer* θ_n *of value* n, *there is a substitution* σ, *such that* $(T)\theta_n f \succ_C (f)\sigma(\tau_n)$.

Corollary 6.1 *If* $\vdash_{AF2} T : \forall x\{N^*[x] \to \neg\neg N[x]\}$, *then for every* $n \geq 0$ *and for every classical integer* θ_n *of value* n, *there is a* λ*-term* τ_n, *such that* $(T)\theta_n\lambda xx \succ_C \tau_n \to_\beta \underline{n}$.

Proof We use Theorem 6.2. □

Remark. Theorem 6.2 cannot be generalized for the system $C2$. Indeed, let $T = \lambda\nu\lambda f(f)(C)(T_i)\nu$ ($i = 1$ or 2).

$\nu : N^*[x], f : \neg N[x] \vdash_{C2} (T_i)\nu : \neg\neg N[x] \Longrightarrow$
$\nu : N^*[x], f : \neg N[x] \vdash_{C2} (C)(T_i)\nu : N[x] \Longrightarrow$
$\nu : N^*[x], f : \neg N[x] \vdash_{C2} (f)(C)(T_i)\nu : \bot \Longrightarrow$
$\vdash_{C2} T : \forall x\{N^*[x] \to \neg\neg N[x]\}$

Since for every λC-term θ, $(T)\theta f \succ_C (f)(C)(T_i)\theta$, then it is easy to check that there is not a λC-term $\tau_n \simeq_\beta \underline{n}$ such that for every classical integer θ_n of value n, there is a substitution σ, such that $(T)\theta_n f \succ_C (f)\sigma(\tau_n)$. □

We will see that in system $M2$ we have a similar result to Theorem 6.2.

Let T be a closed λC-term. We say that T is a storage operator for classical integers if and only if for every $n \geq 0$, there is a λC-term $\tau_n \simeq_\beta \underline{n}$, such that for every classical integers θ_n of value n, there is a substitution σ, such that $(T)\theta_n f \succ_C (f)\sigma(\tau_n)$.

Theorem 6.3 *If $\vdash_{M2} T : \forall x\{N^C[x] \to \neg\neg N[x]\}$, then T is a storage operator for classical integers.*

The type system M is the subsystem of $M2$ where we only have propositional variables and constants (predicate variables or predicate symbols of arity 0). So, first order variable, function symbols, and finite sets of equations are useless. The rules for typed are 0′) 1), 2), 3), 6), 6′), 7) and 7′) restricted to propositional variables. With each predicate variable (resp. predicate symbol) X, we associate a predicate variable (resp. a predicate symbol) $X^\diamond$ of M type system. For each formula A of $M2$, we define the formula $A^\diamond$ of F_C obtained by forgetting in A the first order part. If $\Gamma = x_1 : A_1, ..., x_n : A_n$ is a context of $M2$, then we denote by $\Gamma^\diamond$ the context $x_1 : A_1^\diamond, ..., x_n : A_n^\diamond$ of M. We write $\Gamma \vdash_M t : A$ if t is typable in M of type A in the context Γ.
We have obviously the following property : if $\Gamma \vdash_{M2} t : A$, then $\Gamma^\diamond \vdash_M t : A^\diamond$.

Theorem 6.3 is a consequence of the following Theorem.

Theorem 6.4 *If $\vdash_M T : N^C \to \neg\neg N$, then for every $n \geq 0$, there is an $m \geq 0$ and a λC-term $\tau_m \simeq_\beta \underline{m}$, such that for every classical integer θ_n of value n, there is a substitution σ, such that $(T)\theta_n f \succ_C (f)\sigma(\tau_m)$.*

Indeed, if $\vdash_{M2} T : \forall x\{N^C[x] \to \neg\neg N[x]\}$, then $\vdash_M T : N^C \to \neg\neg N$. Therefore for every $n \geq 0$, there is an $m \geq 0$ and $\tau_m \simeq_\beta \underline{m}$, such that for every classical integer θ_n of value n, there is a substitution σ, such that $(T)\theta_n f \succ_C (f)\sigma(\tau_m)$. We have $\vdash_{M2} \underline{n} : N^C[s^n(0)]$, then $f : \neg N[s^n(0)] \vdash_{M2} (T)\underline{n} f : \bot$, therefore $f : \neg N[s^n(0)] \vdash_{M2} (f)\underline{m} : \bot$ and $\vdash_{M2} \underline{m} : N[s^n(0)]$. Therefore $n = m$. and T is a storage operator for classical integers. □

In order to prove Theorem 6.4, we shall need some Lemmas.

Lemma 6.1 *If* $\Gamma, \nu : N^C \vdash_M (\nu)\overline{d} :\bot$, *then* $\overline{d} = a, b, d_1, ..., d_r$ *and there is a classical type* F, *such that :* $\Gamma, \nu : N^C \vdash_M a : F$; $\Gamma, \nu : N^C \vdash_M b : F \to F$; $F \lhd E_1 \to F_1$, $F_i \lhd E_{i+1} \to F_{i+1}$ $1 \leq i \leq r-1$; $F_r \lhd \bot$; *and* $\Gamma, \nu : N^C \vdash_M c_i : E_i$ $1 \leq i \leq r$.

Proof We use Theorem 2.2. □

Lemma 6.2 *If* F *is a classical type and* $\Gamma, x : F \vdash_M (x)\overline{d} :\bot$, *then* $\overline{d} = d_1, ..., d_r$; $F \lhd E_1 \to F_1$; $F_i \lhd E_{i+1} \to F_{i+1}$ $1 \leq i \leq r-1$; $F_r \lhd \bot$; *and* $\Gamma, x : F \vdash_M c_i : E_i$ $1 \leq i \leq r$.

Proof We use Theorem 2.2. □

Lemma 6.3 *Let* t *be a* β*-normal* λC*-term, and* $A_1, ..., A_n$ *a sequence of classical types.*
If $x_1 : A_1, ..., x_n : A_n \vdash_M t : N$, *then there is an* $m \geq 0$ *such that* $t = \underline{m}$.

Proof We use Theorems 4.1 and 5.2. □

Let ν and f be two fixed variables.
We denote by $x_{n,a,b,\overline{c}}$ (where n is an integer, a, b two λ-terms, and $\overline{c}$ a finite sequence of λ-terms) a variable which does not appear in $a, b, \overline{c}$.

Theorem 6.5 *Let* n *be an integer. There is an integer* m *and a finite sequence of head reductions* $\{U_i \succ_C V_i\}_{1 \leq i \leq r}$ *such that :*
1) $U_1 = (T)\nu f$ *and* $V_r = (f)\tau_m$ *where* $\tau_m \simeq_\beta \underline{m}$;
2) $V_i = (\nu)ab\overline{c}$ *or* $V_i = (x_{l,a,b,\overline{c}})\overline{d}$ $0 \leq l \leq n-1$;
3) If $V_i = (\nu)ab\overline{c}$, *then* $U_{i+1} = (a)\overline{c}$ *if* $n = 0$ *and* $U_{i+1} = ((b)x_{n-1,a,b,\overline{c}})\overline{c}$ *if* $n \neq 0$;
4) If $V_i = (x_{l,a,b,\overline{c}})\overline{d}$ $0 \leq l \leq n-1$, *then* $U_{i+1} = (a)\overline{d}$ *if* $l = 0$ *and* $U_{i+1} = ((b)x_{l-1,a,b,\overline{d}})\overline{d}$ *if* $l \neq 0$.

Proof A good context Γ is a context of the form $\nu : N^C, f : \neg N, x_{n_1,a_1,b_1,\overline{c_1}} : F_1, ..., x_{n_p,a_p,b_p,\overline{c_p}} : F_p$ where F_i is a classical type, $0 \leq n_i \leq n-1$, and $1 \leq i \leq p$.
We will prove that there is an integer m and a finite sequence of head reductions $\{U_i \succ_C V_i\}_{1 \leq i \leq r}$ such that we have 1), 2), 3), 4), and there is a good context Γ such that $\Gamma \vdash_M V_i :\bot$ $1 \leq i \leq r$.

We have $\vdash_M T : N^C \to \neg\neg N$, then $\nu : N^C, f : \neg N \vdash_M (T)\nu f :\bot$, and by Lemmas 6.1 and 6.2, $(T)\nu f \succ_C V_1$ where $V_1 = (f)\tau$ or $V_1 = (\nu)ab\overline{c}$.
Assume that we have the head reduction $U_k \succ_C V_k$ and $V_k \neq (f)\tau$.

- If $V_k = (\nu)ab\overline{c}$, then, by the induction hypothesis, there is a good context Γ such that $\Gamma \vdash_M (\nu)ab\overline{c} :\bot$. By Lemma 6.1, there is a classical type F, such that $\Gamma \vdash_M a : F$; $\Gamma \vdash_M b : F \to F$; $\overline{c} = c_1, ..., c_s$; $F \lhd E_1 \to F_1$; $F_i \lhd E_{i+1} \to F_{i+1}$ $1 \leq i \leq s-1$; $F_s \lhd \bot$; and $\Gamma \vdash_M c_i : E_i$ $1 \leq i \leq s$.

  - If $n = 0$, let $U_{k+1} = (a)\overline{c}$. We have $\Gamma \vdash_M U_{k+1} :\bot$.

  - If $n \neq 0$, let $U_{k+1} = ((b)x_{n-1,a,b,\overline{c}})\overline{c}$. The variable $x_{n-1,a,b,\overline{c}}$ is not used before. Indeed, if it is, we check easily that the λC-term $(T)\underline{n}f$ is not solvable; but that is impossible because $f : \neg N \vdash_M (T)\underline{n}f :\bot$. Therefore $\Gamma' = \Gamma, x_{n-1,a,b,\overline{c}} : F$ is a good context and $\Gamma' \vdash_M U_{k+1} :\bot$.

- If $V_k = (x_{l,a,b,\overline{c}})\overline{d}$, then, by the induction hypothesis, there is a good context Γ such that $\Gamma \vdash_M (x_{l,a,b,\overline{c}})\overline{d} :\bot$. Then there is a classical type F such that $x_{l,a,b,\overline{c}} : F$ is in the context Γ. By Lemma 6.2, $\overline{c} = d_1, ..., d_s$; $F \lhd E_1 \to F_1$; $F_i \lhd E_{i+1} \to F_{i+1}$ $1 \leq i \leq s-1$; $F_s \lhd \bot$; and $\Gamma \vdash_M c_i : E_i$ $1 \leq i \leq s$.

- If $l = 0$, let $U_{k+1} = (a)\overline{c}$. We have $\Gamma \vdash_M U_{k+1} : \bot$.

- If $l \neq 0$, Let $U_{k+1} = ((b)x_{l-1,a,b,\overline{d}})\overline{d}$. The variable $x_{l-1,a,b,\overline{d}}$ is not used before. Indeed, if it is, we check that the λC-term $(T)\underline{n}f$ is not solvable; but this is impossible because $f : \neg N \vdash_M (T)\underline{n}f : \bot$. Then $\Gamma' = \Gamma, x_{l-1,a,b,\overline{d}} : F$ is a good context and $\Gamma' \vdash_M U_{k+1} : \bot$.

Therefore there is a good context Γ' such that $\Gamma' \vdash_M U_{k+1} : \bot$. Then, by Lemmas 6.1 and 6.2, $U_{k+1} \succ_C V_{k+1}$ where $V_{k+1} = (f)\tau$ or $V_{k+1} = (\nu)ab\overline{c}$ or $V_{k+1} = (x_{l,a,b,\overline{c}})\overline{d}$ $0 \leq l \leq n-1$.
This construction always terminates. Indeed, if not, we check that the λC-term $(T)\underline{n}f$ is not solvable; but this is impossible because $f : \neg N \vdash_M (T)\underline{n}f : \bot$.
Therefore there is $r \geq 0$ and a good context Γ such that $\Gamma \vdash_M V_r = (f)\tau : \bot$, and $\Gamma \vdash_M \tau : N$. Therefore, by Lemma 6.3, there is an $m \geq 0$ such that $\tau \simeq_\beta \underline{m}$. □

Let T be a λC-term such that $\vdash_M T : N^C \to \neg\neg N$. By Theorem 6.5, there is an integer s and a finite sequence of head reductions $\{U_i \succ_C V_i\}_{1 \leq i \leq r}$ such that :
1) $U_1 = (T)\nu f$ and $V_r = (f)\tau_s$ where $\tau_s \simeq_\beta \underline{s}$;
2) $V_i = (\nu)ab\overline{c}$ or $V_i = (x_{l,a,b,\overline{c}})\overline{d}$ $0 \leq l \leq n-1$;
3) If $V_i = (\nu)ab\overline{c}$, then $U_{i+1} = (a)\overline{c}$ if $n = 0$ and $U_{i+1} = ((b)x_{n-1,a,b,\overline{c}})\overline{c}$ if $n \neq 0$;
4) If $V_i = (x_{l,a,b,\overline{c}})\overline{d}$ $0 \leq l \leq n-1$, then $U_{i+1} = (a)\overline{d}$ if $l = 0$ and $U_{i+1} = ((b)x_{l-1,a,b,\overline{d}})\overline{d}$ if $l \neq 0$.

Let θ_n be a classical integer of value n, and x, g two distinct variables. By Theorem 5.6 we have :
If $n = 0$, then for every stack constant p, we have : $(\theta_n)xgp \rhd_C (x)p$.
If $n \neq 0$, then there is $m \geq 1$, and a mapping $I : \{0, ..., m\} \to N$, such that for all distinct stack constants $p_0, p_1, ..., p_m$, we have :
$(\theta_n)xgp_0 \rhd_C (g)t_1p_{r_0}$;
$(t_i)p_i \rhd_C (g)t_{i+1}p_{r_i}$ $1 \leq i \leq m-1$;
$(t_m)p_m \rhd_C (x)p_{r_m}$
where $I(0) = n$, $I(r_m) = 0$, and $I(i+1) = I(r_i) - 1$ $0 \leq i \leq m-1$.

Lemma 6.4 *If $n = 0$, then $(T)\theta_n f \succ_C (f)\tau[\theta_n/\nu]$.*

Proof We prove by induction that for every $1 \leq i \leq r$, we have $(T)\theta_n f \succ_C V_i[\theta_n/\nu]$.
For $i = 1$, $(T)\theta_n f = \{(T)\nu f\}[\theta_n/\nu] = U_1[\theta_n/\nu] \succ_C V_1[\theta_n/\nu]$.
Assume it is true for i, and prove it for $i+1$.
$(T)\theta_n f \succ_C V_i[\theta_n/\nu] = \{(\nu)ab\overline{c}\}[\theta_n/\nu] = \{(\theta_n)ab\overline{c}\}[\theta_n/\nu] = \{(\theta_n)xgp\}[a/x, b/g, \overline{c}/p][\theta_n/\nu]$. Since $(\theta_n)xgp \succ_C (x)p$, then $(T)\theta_n f \succ_C \{(a)\overline{c}\}[\theta_n/\nu] = U_{i+1}[\theta_n/\nu] \succ_C V_{i+1}[\theta_n/\nu]$.
So, for $i = r$, we have $(T)\theta_n f \succ_C V_r[\theta_n/\nu] = \{(f)\tau\}[\theta_n/\nu] = (f)\tau[\theta_n/\nu]$. □

We assume now that $n \geq 1$.

A $k - \lambda C$-term is a λC-term of the forme $V_k[\tau_1/y_1]...[\tau_p/y_p][\theta_n/\nu]$ such that :
- $Fv(V_k) \subseteq \{\nu, f, y_1, ..., y_p\}$
- for every $1 \leq i \leq p$, $y_i = x_{n_i,a_i,b_i,\overline{c}_i}$ and $\tau_i = t_{m_i}[a_i/x, b_i/g, \overline{d_0}/p_0, ..., \overline{d_{m_i-1}}/p_{m_i-1}]$ where $I(m_i) = n_i$
- for every $0 \leq k \leq m_i - 1$, there is $1 \leq l \leq r$ such that $U_l = (a_i)\overline{d_k}$ if $I(k) = 0$ and $U_r =$

$(b_i)x_{I(k)-1,a_i,b_i,\overline{d_k}}\overline{d_k}$ if $I(k) > 0$.

To simplify, a $k - \lambda C$-term is denoted by $V_k[]$.

Lemma 6.5 *Let $1 \leq i \leq r-1$ and $V_i[]$ an $i - \lambda C$-term. If $(T)\theta_n f \succ_C V_i[]$, then there is $1 \leq j \leq r$ and a $j - \lambda C$-term $V_j[]$ such that $V_j[] \succ_C V_j[]$ and either $V_i[] \neq V_j[]$ or $i < j$*

Proof There are only two possibilities. 1) $V_i = (\nu)ab\overline{c}$; 2) $V_i = (x_{\alpha,a,b,\overline{c}})\overline{d}$.
We now examine each of this cases.
1) If $V_i = (\nu)ab\overline{c}$, then $V_i[] = \{(\theta_n)ab\overline{c}\}[] = \{(\theta_n)xgp_0\}[a/x, b/g, \overline{c}/p_0][]$. Since $(\theta_n)xgp_0 \rhd_C (g)t_1p_{r_0} = (g)t_1p_0$, then $V_i[] \succ_C \{(b)t_1[a/x, b/g, \overline{c}/p_0]\overline{c}\}[] =$
$\{(b)x_{n-1,a,b,\overline{c}}\overline{c}\}[t_1[a/x, b/g, \overline{c}/p_0/x_{n-1,a,b,\overline{c}}][] = U_{i+1}[] \succ_c V_{i+1}[]$. Let $j = i+1$. We have $i < j$ and $I(1) = I(r_0) - 1 = I(0) - 1 = n - 1$.
2) If $V_i = (x_{\alpha,a,b,\overline{c}})\overline{d}$, then $V_i[] = \{(t_\beta[a/x, b/g, \overline{d_0}/p_0, ..., \overline{d_{\beta-1}}/p_{\beta-1}])\overline{d}\}[]$ where $I(\beta) = \alpha$.
If $I(\beta) = \alpha \neq 0$, then $U_{i+1} = (b)x_{\alpha-1,a,b,\overline{d}}\overline{d} = (b)x_{I(\beta)-1,a,b,\overline{d}}\overline{d}$, and if $I(\beta) = \alpha \neq 0$, then $U_{i+1} = (a)\overline{d}$.
We consider the following two cases.

- If $\beta \leq m$, then $(t_\beta)p_\beta \rhd_C (g)t_{\beta+1}p_{r_\beta}$, so that
$V_i[] \succ_C \{(g)t_{\beta+1}p_{r_\beta}\}[a/x, b/g, \overline{d_0}/p_0, ..., \overline{d_{\beta-1}}/p_{\beta-1}, \overline{d}/p_\beta][] =$
$\{(b)t_{\beta+1}\overline{d_{r_\beta}}\}[a/x, b/g, \overline{d_0}/p_0, ..., \overline{d_{\beta-1}}/p_{\beta-1}, \overline{d}/p_\beta][]$.
Since $\beta \neq m$, then $I(r_\beta) \neq 0$. By the hypothesis there is $1 \leq j \leq r$ such that $U_j = (b)x_{I(r_\beta)-1,a,b,\overline{d_{r_\beta}}}\overline{d_{r_\beta}}$.
Therefore
$V_i[] \succ_C U_j[t_{\beta+1}[a/x, b/g, \overline{d_0}/p_0, ..., \overline{d_{\beta-1}}/p_{\beta-1}, \overline{d}/p_\beta]/x_{I(r_\beta)-1,a,b,\overline{d_{r_\beta}}}][] = U_j[] \succ_C V_j[]$.
If $V_i[] = V_j[]$, then the head C-reduction $(t_\beta)p_\beta \rhd_C (g)t_{\beta+1}p_{r_\beta}$ must be an identity, in other words $(t_\beta)p_\beta = (g)t_{\beta+1}p_{r_\beta}$ and therefore $\beta = r_\beta$. And so $j = i + 1 > i$.

- If $\beta = m$, then $(t_\beta)p_\beta = (t_m)p_m \rhd_C (x)p_{r_m}$, so that
$V_i[] \succ_C \{(x)p_{r_m}\}[a/x, b/g, \overline{d_0}/p_0, ..., \overline{d_{m-1}}/p_{m-1}][] = (a)t_{r_m}\}[]$.
Since $I(r_m) = 0$, then by the hypothesis there is $1 \leq j \leq r$ such that $U_j = (a)t_{r_m}$. Therefore $V_i[] \succ_C U_j[] \succ_C V_j[]$.
If $V_i[] = V_j[]$, then the head C-reduction $(t_m)p_m \rhd_C (x)p_{r_m}$ must be an identity, in other words $(t_m)p_m \rhd_C (x)p_{r_m}$ and therefore $m = r_m$. And so $j = i + 1 > i$. □

Corollary 6.2 *There is a substitution σ such that $(T)\theta_n f \succ_C (f)\sigma(\tau)$.*

Proof $(T)\theta_n f = \{(T)\nu f\}[\theta_n/\nu] = U_1[\theta_n/\nu] \succ_C V_1[\theta_n/\nu]$. By Lemma 6.5 we obtaine a sequence $V_{i_1}[]$, $V_{i_2}[]$, ... , $V_{i_k}[]$, ... such that $(T)\theta_n f \succ_C V_{i_s}[]$ and if $V_{i_s}[] \neq V_{i_{s+1}}[]$ then $i_s \leq i_{s+1}$. This sequence is necessarily finite, indeed $f : \neg N \vdash_M (T)\theta_n f : \bot$. If $V_{i_s}[] = V_{i_{s+1}}[] = ... = V_{i_{s+\alpha}}[]$, then $i_s < i_{s+1} < ... < i_{s+\alpha}$ and $\alpha \leq r$. Therefore there is s such that $V_{i_s} = (f)\tau$, then $(T)\theta_n f \succ_C V_{i_s}[] = \{(f)\tau\}[] = (f)\tau[]$. □

Then, by Lemma 6.4 and Corollary 6.2, T is a storage operator for classical integers.

6.3 General Theorem

In this subsection, we give (without proof) a generalization of Theorem 6.3.

Let T be a closed λC-term, and D, E two closed types of $AF2$ type system. We say that T is a storage operator for the pair of types (D, E) iff for every λ-term $\vdash_{AF2} t : D$, there is λ-term τ'_t and λC-term τ_t, such that $\tau'_t \simeq_\beta \tau_t$, $\vdash_{AF2} \tau'_t : E$, and for every $\vdash_{C2} \theta_t : D$, there is a substitution σ, such that $(T)\theta_t f \succ_C (f)\sigma(\tau_t)$.

Theorem 6.6 *Let D, E two $\forall$-positive closed types of $AF2$ type system, such that E does not contain $\bot$. If $\vdash_{M2} T : D^C \to \neg\neg E$, then T is a storage operator for the pair (D, E).*

7 Operational characterization of λC-terms of type $\forall X_C\{\bot \to X_C\}$ and $\forall X_C\{\neg\neg X_C \to X_C\}$

Let **A** (for Abort) the λC-term $\lambda x(C)\lambda yx$.

Behaviour of A :

$$(\mathbf{A}\ )tt_1...t_n \succ_C ((C)\lambda yt)t_1...t_n \succ_C (\lambda yt)\lambda x(x)t_1...t_n \succ_C t.$$

Typing of A :

$$x :\bot \vdash_{M2} \lambda yx : \neg\neg X_C \Longrightarrow x :\bot \vdash_{M2} (C)\lambda yx : X_C \Longrightarrow \vdash_{M2} \mathbf{A} : \forall X_C\{\bot \to X_C\}$$

Theorem 7.1 *If $\vdash_{M2} T : \forall X_C\{\bot \to X_C\}$, then for every integer n, and for all $\lambda C - terms\ t, t_1, ..., t_n$, $(T)tt_1...t_n \succ_C t$.*

Proof. Let $O_1, ..., O_n$ be new predicate symbols of arity 0 different from $\bot$. Let $A = O_1, ..., O_n \to \bot$. If $\vdash_{M2} T : \forall X_C\{\bot \to X_C\}$, then $\vdash_{M2} T :\bot \to A$, and $\Gamma = x :\bot, x_1 : O_1, ..., x_n : O_n \vdash_{M2} (T)xx_1...x_n :\bot$. Therefore $(T)xx_1...x_n \succ_C (f)u_1...u_r$ and $\Gamma \vdash_{M2} (f)u_1...u_r :\bot$.

- If $f = x_i$ $1 \leq i \leq n$, then $r = 0$, and $O_i = \bot$. A contradiction.

- If $f = x$, then $r = 0$, and $(T)xx_1...x_n \succ_C x$, therefore, for every integer n, and for all λC-terms $t, t_1, ..., t_n$, $(T)tt_1...t_n \succ_C t$. □

The constant C satisfies the following relations :
$(C)tt_1...t_n \succ_C (t)U$ and
$(U)y \succ_C (y)t_1...t_n$ where y is a new variable.

Let $C' = \lambda x(C)\lambda d(x)\lambda y(x)\lambda z(d)y$.

$x : \neg\neg X_C, y : X_C, z : X_C, d : \neg X_C \vdash_{M2} (d)y :\bot \Longrightarrow$
$x : \neg\neg X_C, y : X_C, d : \neg X_C \vdash_{M2} (x)\lambda z(d)y :\bot \Longrightarrow$
$x : \neg\neg X_C, d : \neg X_C \vdash_{M2} (x)\lambda y(x)\lambda z(d)y :\bot \Longrightarrow$
$x : \neg\neg X_C \vdash_{M2} (C)\lambda d(x)\lambda y(x)\lambda z(d)y : X_C \Longrightarrow$
$\vdash_{M2} C' : \forall X_C\{\neg\neg X_C \to X_C\}$.

The λC-term C' satisfies the following relations :
$(C')tt_1...t_n \succ_C (t)U$,

$(U)y \succ_C (t)V$, and
$(V)z \succ_C (y)t_1...t_n$ where y, z are new variables.

In general, we have the following characterization.

Theorem 7.2 *If $\vdash_{M2} T : \forall X_C\{\neg\neg X_C \to X_C\}$, then there is an integer m, such that, for every integer n, and for all λC-terms $t, t_1, ..., t_n$:*

$(T)tt_1...t_n \succ_C (t)V_1$,

$(V_i)y_i \succ_C (t)V_{i+1}$ $1 \leq i \leq m-1$, *and*

$(V_m)y_m \succ_C (y_i)t_1...t_n$ *where $y_1, ..., y_m$ are new variables.*

Proof Let O be a new predicate symbol of arity 0 different from $\perp$. We define as in section 3, the system $M2_O$. And we check easily that this system has the same results as Lemmas 5.1, 5.2, 5.3 and 5.4.
Let p be a stack constant and $A = O \to \perp$. If $\vdash_{M2} T : \forall X_C\{\neg\neg X_C \to X_C\}$, then $\vdash_{M2_O} T : \neg\neg A \to A$, and $\Gamma = x : \neg\neg A, p : O \vdash_{M2_O} (T)xp : \perp$. Therefore $(T)xp \succ_C (f)u_1...u_r$, and $\Gamma \vdash_{M2_O} (f)u_1...u_r : \perp$.

- If $f = p$, then $r = 0$, and $O = \perp$. A contradiction.

- If $f = x$, then, $(T)xp \rhd_C (x)U_1$, and $\Gamma \vdash_{M2_O} U_1 : \neg A$.

We prove (by induction) that if $\Gamma, y_1 : A, ..., y_{i-1} : A \vdash_{M2_O} U_i : \neg A$, then $[(U_i)y_i \rhd_C (x)U_{i+1}$, and $\Gamma, y_1 : A, ..., y_i : A \vdash_{M2_O} U_{i+1} : \neg A]$ or $[(U_i)y_i \rhd_C (y_j)p$ $1 \leq j \leq i]$.
The sequence $(U_i)_{i \geq 0}$ is not infinite. Indeed, if it is, the λC-term $((T)\lambda x(x)z)p$ is not C-solvable; but this is impossible, because $z : A, p : O \vdash_{M2} ((T)\lambda x(x)z)p : \perp$.
To obtain the Theorem, we replace the constant p by the sequence $\overline{t} = t_1, ..., t_n$ and we put $V_i = U_i[\overline{t}/p]$.
□

8 The $\lambda\mu$-calculus

In this section, we give a similar version to Theorem 6.3 in the M. Parigot's $\lambda\mu$-calculus.

8.1 Pure and typed $\lambda\mu$-calculus

$\lambda\mu$-calculus has two distinct alphabets of variables : the set of λ-variables $x, y, z, ...$, and the set of μ-variables $\alpha, \beta, \gamma, ...$ Terms are defined by the following grammar :

$$t := x \mid \lambda x t \mid (t)t \mid \mu\alpha[\beta]t$$

Terms of $\lambda\mu$-calculus are called $\lambda\mu$-terms.

The reduction relation of $\lambda\mu$-calculus is induced by fives different notions of reduction :

The computation rules

(C_1) $(\lambda x u)v \to u[v/x]$

(C_2) $(\mu\alpha u)v \to \mu\alpha u[v/^*\alpha]$

where $u[\overline{v}/^*\alpha]$ is obtained from u by replacing inductively each subterm of the form $[\alpha]w$ by $[\alpha](w)\overline{v}$.

The simplification rules

(S_1) $[\alpha]\mu\beta u \to u[\alpha/\beta]$

(S_2) $\mu\alpha[\alpha]u \to u$, if α has no free occurence in u

(S_3) $\mu\alpha u \to \lambda x\mu\alpha u[x/^*\alpha]$, if u contains a subterm of the form $[\alpha]\lambda yw$.

Theorem 8.1 *(see [18]) In $\lambda\mu$-calculus, reduction is confluent.*

The notation $u \succ_\mu v$ means that v is obtained from u by some head reductions.
The head equivalence relation is denoted by : $u \sim_\mu v$ if and only if there is a w, such that $u \succ_\mu w$ and $v \succ_\mu w$.

Proofs are written in a natural deduction system with several conclusions, presented with sequents. One deals with sequents such that :
- Formulas to the left of $\vdash$ are labelled with λ-variables ;
- Formulas to the right of $\vdash$ are labelled with μ-variables, except one formula which is labelled with a $\lambda\mu$-term ;
- Distinct formulas never have the same label.

The right and the left parts of the sequents are considered as sets and therefore contraction of formulas is done implicitly.

Let t be a $\lambda\mu$-term, A a type, $\Gamma = x_1 : A_1, ..., x_n : A_n$, and $\triangle = \alpha_1 : B_1, ..., \alpha_m : B_m$. We define by means of the following rules the notion "t is of type A in Γ and $\triangle$". This notion is denoted by $\Gamma \vdash_{FD2} t : A, \triangle$.

The rules (1),...,(8) of $AF2$ type system.

(9) If $\Gamma \vdash_{FD2} t : A, \beta : B, \triangle$, then $\Gamma \vdash_{FD2} \mu\beta[\alpha]t : B, \alpha : A, \triangle$.

Weakenings are included in the rules (2) and (9).

As in typed λ-calculus on can define $\neg A$ as $\to\bot$ and use the previous rules with the following special interpretation of naming for $\bot$: for α a μ-variable, $\alpha :\bot$ is not mentioned.

Example Let $\mathbf{C} = \lambda x\mu\alpha[\phi](x)\lambda y\mu\beta[\alpha]y$.
$x : \neg\neg X, y : X \vdash_{FD2} y : X \Longrightarrow$
$x : \neg\neg X, y : X \vdash_{FD2} \mu\beta[\alpha]y :\bot, \alpha : X \Longrightarrow$
$x : \neg\neg X \vdash_{FD2} \lambda y\mu\beta[\alpha]y : \neg X, \alpha : X \Longrightarrow$
$x : \neg\neg X \vdash_{FD2} \mu\alpha[\phi](x)\lambda y\beta[\alpha]y : X \Longrightarrow$
$\vdash_{FD2} \mathbf{C} : \forall X\{\neg\neg X \to X\}$.

Theorem 8.2 *(see [18] and [20]) The $FD2$ type system has the following properties :*
1) Type is preserved during reduction.
2) Typable $\lambda\mu$-terms are strongly normalizable.

8.2 Classical integers

Let n be an integer. A classical integer of value n is a closed $\lambda\mu$-term θ_n such that $\vdash_{FD2} \theta_n : N[s^n(0)]$.

Let x and f fixed variables, and $N_{x,f}$ be the set of $\lambda\mu$-terms defined by the following grammar :

$$u := x \mid (f)u \mid \mu\alpha[\beta]x \mid \mu\alpha[\beta]u$$

We define, for each $u \in N_{x,f}$ the set $rep(u)$, which is intuitively the set of integers potentially repesented by u :

- $rep(x) = \{0\}$
- $rep((f)u) = \{n+1 \text{ if } n \in rep(u)\}$
- $rep(\mu\alpha[\beta]u) = \bigcap rep(v)$ for each subterm $[\alpha]v$ of $[\beta]u$

The following Theorem characterizes the normal forms of classical integers.

Theorem 8.3 *(see [19]) The normal classical integers of value n are exactly the $\lambda\mu$-terms of the form $\lambda x \lambda f u$ with $u \in N_{x,f}$ without free μ-variable and such that rep(u)=$\{n\}$.*

8.3 General Theorem

In order to define, in this framework, the equivalent of system $M2$, the demonstration of $\neg\neg A \to A$ should not be allowed for all formulas A, and thus we should prevent the occurrence of some formulas on the right. Thus the following definition.

Let t be a $\lambda\mu$-term, A a type, $\Gamma = x_1 : A_1, ..., x_n : A_n$, and $\triangle = \alpha_1 : B_1, ..., \alpha_m : B_m$ where B_i $1 \leq i \leq m$ is a classical type. We define by means of the following rules the notion "t is of type A in Γ and $\triangle$", this notion is denoted by $\Gamma \vdash_{M2} t : A, \triangle$.

The rules of $DL2$ type system.

($6'$) If $\Gamma \vdash t : A, \triangle$, and X_C has no free occurence in Γ, then $\Gamma \vdash t : \forall X_C A, \triangle$.

($7'$) If $\Gamma \vdash t : \forall X_C A, \triangle$, and G is a classical type, then $\Gamma \vdash t : A[G/X_C], \triangle$.

Let T be a closed $\lambda\mu$-term. We say that T is a storage operator for classical integers if and only if for every $n \geq 0$, there is $\lambda\mu$-term $\tau_n \simeq_\beta \underline{n}$, such that for every classical integers θ_n of value n, there is a substitution σ, such that $(T)\theta_n f \sim_\mu \mu\alpha[\alpha](f)\sigma(\tau_n)$.

Theorem 8.4 *If $\vdash_{M2} T : \forall x\{N^C[x] \to \neg\neg N[x]\}$, then T is a storage operator for classical integers.*

References

[1] M. Felleisein *The Calculi of $\lambda_v - CS$ conversion: a syntactic theory of control and state in imperative higher order programming.*
Ph. D. dissertation, Indiana University, 1987.

[2] J.L. Krivine *Lambda-calcul, types et modèles*
Masson, Paris 1990.

[3] J.L. Krivine *Opérateurs de mise en mémoire et traduction de Gődel*
Archiv for Mathematical Logic 30, 1990, pp. 241-267.

[4] J.L. Krivine *Lambda-calcul, évaluation paresseuse et mise en mémoire*
Thearetical Informatics and Applications. Vol. 25,1 p. 67-84 , 1991.

[5] J.L. Krivine *Mise en mémoire (preuve générale)*
Manuscript, 1993.

[6] J.L. Krivine *Classical logic, storage operators and 2nd order lambda-calculus*
Ann. Pure and Applied Logic 68 (1994) p. 53-78.

[7] J.L. Krivine *A general storage theorem for integers in call-by-name λ-calculus*
Th. Comp. Sc. (to appear).

[8] K. Nour *Opérateurs de mise en mémoire en lambda-calcul pur et typé*
Thèse de Doctorat, Université de Chambéry, 1993.

[9] K. Nour and R. David *Storage operators and directed λ-calculus*
Journal of symbolic logic, vol 60, n 4, p. 1054-1086, 1995.

[10] K. Nour *Une preuve syntaxique d'un Théorème de J.L. Krivine sur les opérateurs de mise en mémoire*
C.R. Acad. Sci Paris, t. 318, Série I, p. 201-204, 1994.

[11] K. Nour *Opérateurs de mise en mémoire et types $\forall$-positifs*
Thearetical Informatics and Applications (to appear).

[12] K. Nour *Entiers intuitionnistes et entiers classiques en λC-calcul*
Thearetical Informatics and Applications, vol 29, n 4, p. 293-313, 1995.

[13] K. Nour *Quelques résultats sur le λC-calcul*
C.R. Acad. Sci Paris, t. 320, Série I, p. 259-262, 1995.

[14] K. Nour *A general type for storage operators*
Mathematical Logic Quarterly, 41 p. 505-514, 1995.

[15] K. Nour *La valeur d'un entier classique en $\lambda\mu$-calcul*
Submitted to Archive for Mathematical Logic.

[16] K. Nour *Caractérisation opérationnelle des entiers classiques en λC-calcul*
C.R. Acad. Sci Paris, t. 320, Série I, p. 1431-1434, 1995.

[17] M. Parigot *Free deduction : an analyse of computations in classical logic*
Proc. Russian Conference on Logic Programming, St Petersburg (Russia), 1991, Springer LNCS 592, pp. 361-380.

[18] M. Parigot *$\lambda\mu$-calculus : an algorithm interpretation of classical natural deduction*
Proc. International Conference on Logic Programming and Automated Reasoning, St Petersburg (Russia), 1992, Springer LNCS 624, pp. 190-201.

[19] M. Parigot *Classical proofs as programs*
To appear in Proc. 3rd Krut Gődel Colloquium KGC'93, Springer Lectures Notes in Computer Science.

[20] M. Parigot *Strong normalization for second order classical deduction*
To appear in Proc.LICS 1993.