Proofs and programs

Karim Nour

Université Savoie Mont Blanc
LAMA - LIMD

Seminaires de LIM - Université de la Réunion
26 october 2023

Outline

Introduction

Aims of my recherch domain :

- **Writing programs that are provably correct.**
- **Finding the algorithmic content of mathematical proofs.**

Introduction

This relationship between proofs and programs is called :

Curry-Howard correspondence

- **Curry (1958)** observes that a fragment of Hilbert-style deductions coincides to a fragment of combinatory logic.
- **Howard (1969)** observes that the natural deduction proof system can be interpreted as λ-calculus.
- **Griffin (1990)** uses classical logic in order to give types to escape instructions.

Introduction

How to realise this correspondence ?

- **Choose a fairly expressive logic :**
  set theory, second order logic, higher order logic, . . .

- **Code the proofs by objects that will be our programs.**

Introduction

- We find in the literature several systems to illustrate this relation : proofs/programs.
- We will present a very simple and very effective version developed in France since the 90s.
- It allows
  - **to correctly program functions on data types,**
  - **to find the algorithmic content of any proof.**

Outline

$\mathcal{AF}_2$ (Krivine 1988)

- We choose as logic : the second-order intuitionistic logic.
- We can quantify on objects and predicates.
- We only need two connectors $\rightarrow$ and $\forall$.
- This logic is very expressive : it makes it possible to
  - code the data types,
  - code a very large class of mathematical proofs.

Formulas

We need :

- Constants and functions (with arities)
- First order variables : $x, y, z, \dots$
- Predicate variables (with arities) : $X, Y, Z, \dots$

Definition (Terms)

- *A conctant and a first order variable is a term.*
- *If f is an n-ary function and $t_1, \dots, t_n$ are terms, then $f(t_1, \dots, t_n)$ is a term.*

Formulas

Definition (Formulas)

- *If t_1, t_2 are terms, then $t_1 = t_2$ is a formula.*
- *If X is an n-ary predicate and $t_1, \dots, t_n$ are terms, then $X(t_1, \dots, t_n)$ is a formula.*
- *If F_1, F_2 are formulas, then $F_1 \to F_2$ is a formula.*
- *If F is a formula and x a first order variable, then $\forall x, F$ is a formula.*
- *If F is a formula and X a predicate variable, then $\forall X, F$ is a formula.*

Coding of connectors

We write $F, G \to H$ instead of $F \to (G \to H)$.

Definition (Coding of connectors)

- $\perp = \forall X, X$.
- $\neg F = F \to \perp$.
- $F_1 \wedge F_2 = \forall X, \{[F_1, F_2 \to X] \to X\}$.
- $F_1 \vee F_2 = \forall X, \{([F_1 \to X), (F_2 \to X) \to X] \to X\}$.
- $\exists x, F = \forall Y, \{[\forall x, (F \to Y)] \to Y\}$.
- $\exists X, F = \forall Y, \{[\forall X, (F \to Y)] \to Y\}$.

Formulas (example)

x is an integer $\iff$ x is in the smallest set containing 0 and stable under the sucessor function s.

$$\underbrace{\forall X, \{ \overbrace{\forall y, (X(y) \to X(s(y)))}^{X \textit{ stable by } s}, \overbrace{X(0)}^{X \textit{ contains } 0} \to X(x) \}}_{\mathbb{N}[x] \quad \textit{means} \quad x \textit{ is an integer}}$$

- 0 is a constant,
- s is a unary function,
- x, y are variables,
- X is a unary predicate variable.

Sequents

$$\underbrace{A_1, \ldots, A_n}_{\Gamma} \vdash A$$

- Γ is a finite set of formulas.
- A is a formula.

We write : "A is provable from the formulas in Γ".

Proof rules

$$\frac{}{\Gamma, A \vdash A}\ ax$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \to_i \qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \to_e$$

Proof rules

$$\frac{\Gamma \vdash A[x]}{\Gamma \vdash \forall x,\ A[x]}\ \forall_i^1$$

x is free in Γ

$$\frac{\Gamma \vdash A[X]}{\Gamma \vdash \forall X,\ A[X]}\ \forall_i^2$$

X is free in Γ

$$\frac{\Gamma \vdash \forall x,\ A[x]}{\Gamma \vdash A[t]}\ \forall_e^1$$

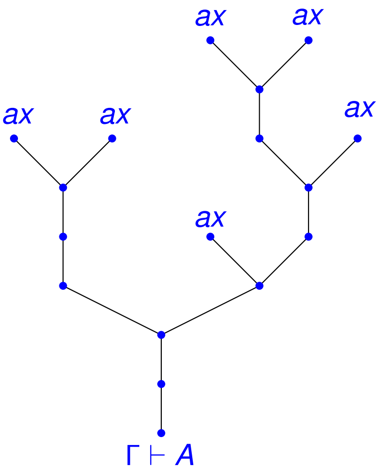
t is a term

$$\frac{\Gamma \vdash \forall X,\ A[X]}{\Gamma \vdash A[F]}\ \forall_e^2$$

F is a formula

$$\frac{\Gamma \vdash A[u] \qquad u = v}{\Gamma \vdash A[v]}\ =$$

Proofs



Simplifications

- We have a notion of reduction on proofs.
- The goal is to simplify the proofs and avoid the use of lemmas.
- This kind of reduction on proofs allow us to prove that the notion of provability is semi-decidable.
- These reductions will allow us to execute programs based on proofs.

Outline

Sequent

Definition

A context Γ *is a set of typing assumptions*

$$\Gamma = x_1 : A_1, \ldots, x_n : A_n$$

where $x_1, \ldots, x_n$ *are* λ*-variables and* $A_1, \ldots, A_n$ *are formulas.*

Definition

The typing relation

$$\Gamma \vdash M : T$$

indicates that M *is a program of type* T *in context* Γ*.*

Coding

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}\ \to_i \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash (M)N : B}\ \to_e$$

Coding

$$\frac{\Gamma \vdash M : A[x]}{\Gamma \vdash M : \forall x, A[x]} \forall_i^1$$

x is free in Γ

$$\frac{\Gamma \vdash M : A[X]}{\Gamma \vdash M : \forall X, A[X]} \forall_i^2$$

X is free in Γ

$$\frac{\Gamma \vdash M : \forall x, A[x]}{\Gamma \vdash M : A[t]} \forall_e^1$$

t is a term

$$\frac{\Gamma \vdash M : \forall X, A[X]}{\Gamma \vdash M : A[F]} \forall_e^2$$

F is a formula

$$\frac{\Gamma \vdash M : A[u] \qquad u = v}{\Gamma \vdash M : A[v]} =$$

Programs and reduction

The λ-calculus is based on a set of λ-variables

$$\mathcal{V} = \{x, y, z, \ldots\}$$

Definition (λ-terms)

The definition of the programs is given by the grammar :

$$\mathcal{T} \quad ::= \quad \mathcal{V} \quad | \quad \lambda\mathcal{V}.\mathcal{T} \quad | \quad (\mathcal{T})\,\mathcal{T}$$

Definition (β-reduction)

$$(\lambda x.M)N \quad \triangleright_\beta \quad M[x := N]$$

Reduction rules

Definition

- *We write* $M \triangleright_\beta M'$ *if* M *reduces to* M' *in one step of* β*-reduction.*
- *We write* $M \triangleright_\beta^* M'$ *if* $M \triangleright_\beta M_1 \triangleright_\beta M_2 \triangleright_\beta \cdots \triangleright_\beta M_k = M'$.

Theorem (Confluence)

If $M \triangleright_\beta^* M_1$ *and* $M \triangleright_\beta^* M_2$*, then* $\exists\, M'$ *such that* $M_1 \triangleright_\beta^* M'$ *and* $M_2 \triangleright_\beta^* M'$.

Theorem (Subject reduction)

If $\Gamma \vdash M : T$ *and* $M \triangleright_\beta^* N$ *, then* $\Gamma \vdash N : T$.

Normalization

Definition

- *A λ-term that does not reduce is called normal form.*
- *A λ-term M is strongly normalizable, if there exists no infinite reduction path out of M. That is, any possible sequence of reductions eventually leads to a normal form.*

Theorem (Strong Normalization)

If $\Gamma \vdash M : T$, then M is strongly normalizable.

Proofs : Girard (1972) and Krivine (1988).

Outline

1. Introduction
2. Proofs as Programs
   - Proofs
   - Programs
   - Programming theorem
   - A few developments
3. Classical logic
   - The calculus
   - The reduction μ'
   - General system

Church's numerals

Example

$$\underline{n} = \lambda f.\lambda x.\overbrace{(f)\ldots(f)}^{n \text{ times}}x$$

$$\underline{s} = \lambda n.\lambda f.\lambda x.(f)((n)f)x$$

$$\underline{+} = \lambda m.\lambda n.\lambda f.\lambda x.((m)f)((n)f)x$$

$$\underline{\times} = \lambda m.\lambda n.\lambda f.\lambda x.((m)(n)f)x$$

Lemma

$$(\underline{s})\underline{n} \quad \triangleright_\beta^* \quad \underline{n+1}$$

$$((\underline{+})\underline{m})\underline{n} \quad \triangleright_\beta^* \quad \underline{m+n}$$

$$((\underline{\times})\underline{m})\underline{n} \quad \triangleright_\beta^* \quad \underline{m\times n}$$

Properties

Example

For all $n \in \mathbb{N}$, $\vdash \underline{n} : \mathbb{N}[s^n(0)]$.

Theorem

For all $n \in \mathbb{N}$, *if* $\vdash M : \mathbb{N}[s^n(0)]$, *then* $M \triangleright_\beta^* \underline{n}$.

Example

$\vdash \underline{s} : \forall x, \{\mathbb{N}[x] \to \mathbb{N}[s(x)]\}$.

$+$ and $\times$

$$\begin{cases} 0+y & = & y, \\ s(x)+y & = & s(x+y). \end{cases}$$

$$\begin{cases} 0\times y & = & 0, \\ s(x)\times y & = & (x\times y)+y. \end{cases}$$

We can prove :

- $\vdash \underline{+} : \forall x, \forall y, \{\mathbb{N}[x], \mathbb{N}[y] \to \mathbb{N}[x+y]\}$.
- $\vdash \underline{\times} : \forall x, \forall y, \{\mathbb{N}[x], \mathbb{N}[y] \to \mathbb{N}[x\times y]\}$.

Predessor

$$\begin{cases} p(0) & = & 0, \\ p(s(x)) & = & x. \end{cases}$$

We can :

- prove $\vdash \forall x, \{\mathbb{N}[x] \to \mathbb{N}[p(x)]\}$,
- find a program P such that $\vdash P : \forall x, \{\mathbb{N}[x] \to \mathbb{N}[p(x)]\}$,
- $(P)\underline{0} \ \triangleright_\beta^* \ \underline{0}$ and $\forall n \in \mathbb{N}^*, (P)\underline{n} \ \triangleright_\beta^* \ \underline{n-1}$.

$$P = \lambda n.(((n)\lambda a.\lambda b.((b)((a)\lambda x.\lambda y.y))$$

$$(\lambda n.\lambda f.\lambda x.(f)((n)f)x)(a)\lambda x.\lambda y.y)\lambda c.((c)\lambda x.\lambda f.x)\lambda x.\lambda f.(f)x)\lambda x.\lambda y.x$$

Programming theorem

Theorem (J.-L. Krivine (1988))

If $\vdash F : \forall x_1, \ldots \forall x_n, \{\mathbb{N}[x_1], \ldots, \mathbb{N}[x_n] \to \mathbb{N}[f(x_1, \ldots, x_n)]\}$,
then F *is a correct program for the function* f.

PROPRE : PROgrammation avec des **PRE**uves
Manoury, Parigot and Simonot (1992)

inf

$$\begin{cases} inf(0,y) & = & 0, \\ inf(x,0) & = & 0, \\ inf(s(x),s(y)) & = & s(inf(x,y)). \end{cases}$$

- Maurey has given a λ-term

  $$\lambda n.\lambda m.((n)\lambda f.\lambda g.(g)f)\lambda x.n)((m)\lambda f.\lambda g.(g)f)\lambda x.m$$

  that computes the inf function in time $O(inf)$.

- Krivine has shown that this λ-term cannot be typed of type $\mathbb{N}, \mathbb{N} \to \mathbb{N}$.

inf

- The λ-term

$$\lambda n.\lambda m.((n)A)\lambda p.\underline{0})m$$
$$A = \lambda u.\lambda v(((v)H)\lambda c.((c)\underline{0})\underline{0})\lambda a.\lambda b.b$$
$$H = \lambda w.\lambda c.((c)(\underline{s})(w)\lambda a.\lambda b.a)))(\underline{s})(u)(w)\lambda a.\lambda b.a$$

  computes the inf function in time $O(inf^2)$ and has the type $\forall x, \forall y, \{\mathbb{N}[x], \mathbb{N}[y] \to \mathbb{N}[inf(x, y)]$.

- David (2009) has given a λ-term that computes the inf function in time $O(inf\ .\ ln(inf))$ and has the type $\forall x, \forall y, \{\mathbb{N}[x], \mathbb{N}[y] \to \mathbb{N}[inf(x, y)]$.

GCD

$$\begin{cases} gcd(0,y) &= y, \\ gcd(s(x),0) &= s(x), \\ gcd(s(x),s(y)) &= gcd(s(min(x,y)),dif(x,y)), \\ min(0,y) &= 0, \\ min(s(x),0) &= 0, \\ min(s(x),s(y)) &= s(min(x,y)), \\ dif(0,y) &= y, \\ dif(s(x),0) &= s(x), \\ dif(s(x),s(y)) &= dif(x,y). \end{cases}$$

We can :

- prove $\vdash \forall x, \forall y, \{\mathbb{N}[x], \mathbb{N}[y] \to \mathbb{N}[gcd(x,y)]\}$,
- find a program *GCD* such that $\vdash GCD : \forall x, \forall y, \{\mathbb{N}[x], \mathbb{N}[y] \to \mathbb{N}[gcd(x,y)]\}$.

Outline

Questions

We are interested in :

- **Studying the syntactical properties of the (typed and untyped) programs.**
- **Understanding the connection between proofs and their algorithmic content.**
- **Finding good semantics for logical system :** "set theory, . . . "

Storage operator

Definition

- *Let O be a predicat constant, and for any formula F, we denote $\neg' F = F \to O$.*
- $N'[x] = \forall X, \{\forall y, [\neg' X(y) \to \neg' X(s(y))], \neg' X(0) \to \neg' X(x)\}$.
- *Let* $\mathcal{SO} = \lambda \nu.((\nu)F)\delta$ *where* $F = \lambda z.\lambda y.(z)\lambda x.(y)(\underline{s})x$ *and* $\delta = \lambda f.(f)\underline{0}$.

Example

- *We have* $\vdash \mathcal{SO} : \forall x, \{N'[x] \to \neg'\neg' N[x]\}$.
- *We have* $\forall F, \forall n \in \mathbb{N}, \forall \theta_n \rhd^*_\beta \underline{n}$, $((\mathcal{SO})F)\theta_n \rhd^*_{name} (F)(\underline{s})^n\underline{0}$.

Storage operator

- The λ-term $\mathcal{SO}$ is called a storage operator ; it enables simulating call by value using call by name computation.
- We can find other typed storage operators. These operators play a significant role in different circumstances.
- What is the connection between the type of these operators and their computational behavior ?

Storage operator

Theorem (Krivine 1989)

If $\Gamma \vdash M : \forall x, \{N'[x] \to \neg'\neg' N[x]\}$, *then M is a storage operator.*

- This theorem generalizes to other data types.
- Multiple possible proofs, even for other types.
- Mixed logic has allowed for a better understanding of this result that remains unclear.

Other examples (Krivine 1996)

Completeness theorem of first-order classical logic :

A formula true in all models is provable.

- Rigorous Formalization.
- Intuitionistic proof.
- Decompiler.

Outline

Proof and coding

Goal : Extending the previous system to classical logic.
(Parigot 1992)

Two rules :

$$\frac{\Gamma, \neg B \vdash \bot}{\Gamma \vdash B}\ \bot_i \qquad\qquad \frac{\Gamma, \neg A \vdash A}{\Gamma, \neg A \vdash \bot}\ \bot_e$$

Two codings :

$$\frac{\Gamma, \alpha : \neg B \vdash M : \bot}{\Gamma \vdash \mu\alpha.M : B}\ \bot_i \qquad\qquad \frac{\Gamma, \alpha : \neg A \vdash M : A}{\Gamma, \alpha : \neg A \vdash [\alpha]M : \bot}\ \bot_e$$

Reduction rules

Definition

$$(\lambda x.M)N \quad \triangleright_\beta \quad M[x := N]$$

$$(\mu\alpha.M)N \quad \triangleright_\mu \quad \mu\alpha.M[[\alpha]U := [\alpha](U)N]$$

$$[\alpha]\mu\beta.M \quad \triangleright_\rho \quad M[\beta := \alpha]$$

$$\mu\alpha.[\alpha]M \quad \triangleright_\theta \quad M \qquad \mathit{if}\ \alpha \notin FV(M)$$

$$\mu\alpha.\mu\beta.M \quad \triangleright_\varepsilon \quad \mu\alpha.M_\beta$$

Reduction rules

Definition

- *We write* $M \triangleright M'$ *if* M *reduces to* M' *in one step of reduction.*
- *We write* $M \triangleright^* M'$ *if* $M \triangleright M_1 \triangleright M_2 \triangleright \cdots \triangleright M_k = M'$.

Theorem (Subject reduction)

If $\Gamma \vdash M : T$ *and* $M \triangleright^* N$ *, then* $\Gamma \vdash N : T$.

Theorem (Confluence)

If $M \triangleright^* M_1$ *and* $M \triangleright^* M_2$*, then* $\exists\, M'$ *such that* $M_1 \triangleright^* M'$ *and* $M_2 \triangleright^* M'$.

Proof : Py (2001)

Normalization

Definition

- *A $\lambda\mu$-term that does not reduce is called normal form.*
- *A $\lambda\mu$-term M is strongly normalizable, if there exists no infinite reduction path out of M. That is, any possible sequence of reductions eventually leads to a normal form.*

Theorem (Strong Normalization)

If $\Gamma \vdash M : T$, then M is strongly normalizable.

Proof : Parigot (1997).

Example 1 (Abort instruction)

$$\dfrac{\dfrac{\dfrac{x : \bot, \alpha : \neg X \vdash x : \bot}{x : \bot \vdash \mu\alpha.x : X}}{\vdash \lambda x.\mu\alpha.x : \bot \to X}}{\vdash \underbrace{\lambda x.\mu\alpha.x}_{\mathcal{A}} : \forall X, \{\bot \to X\}}$$

Lemma

We have $\forall\, n \in \mathbb{N}, \forall\, M, M_1, \ldots, M_n$

$$(\mathcal{A})M\; M_1 \ldots M_n \quad \triangleright^* \quad \mu\alpha.M$$

Example 2 (call/cc instruction)

$$\dfrac{x : \neg X \to X, \alpha : \neg X \vdash x : \neg X \to X \qquad \dfrac{\dfrac{x : \neg X \to X, \alpha : \neg X, y : X \vdash y : X}{x : \neg X \to X, \alpha : \neg X, y : X \vdash [\alpha]y : \bot}}{x : \neg X \to X, \alpha : \neg X \vdash \lambda y.[\alpha]y : \neg X}}{\dfrac{\dfrac{\dfrac{\dfrac{x : \neg X \to X, \alpha : \neg X \vdash (x)\lambda y.[\alpha]y : X}{x : \neg X \to X, \alpha : \neg X \vdash [\alpha](x)\lambda y.[\alpha]y : \bot}}{x : \neg X \to X \vdash \mu\alpha.[\alpha](x)\lambda y.[\alpha]y : X}}{\vdash \lambda x.\mu\alpha.[\alpha](x)\lambda y.[\alpha]y : (\neg X \to X) \to X}}{\vdash \underbrace{\lambda x.\mu\alpha.[\alpha](x)\lambda y.[\alpha]y}_{\mathcal{CC}} : \forall X, \{(\neg X \to X) \to X\}}}$$

Lemma

We have $\forall\, n \in \mathbb{N}, \forall\, M, M_1, \ldots, M_n$

$$(\mathcal{CC})M\ M_1 \ldots M_n \ \triangleright^*\ \mu\alpha.[\alpha](M)\lambda y.[\alpha](y)M_1 \ldots M_n$$

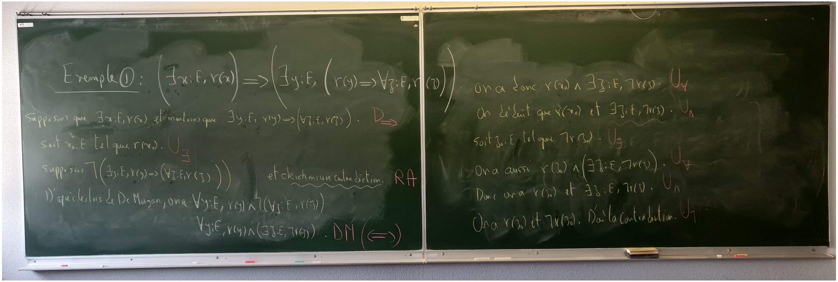
Example 3 (Krivine 2005)

The smoker's paradox : $\exists x, \forall y, \{F(x) \to F(y)\}$.

There is a person such that if they smoke, everyone else smokes too.

- Classical proof.
- Object-Oriented language instruction.

Example 3


Exemple ① : (∃x:E, r(x)) ⇒ (∃y:E, (r(y) ⇒ ∀z:E, r(z)))
Supposons que ∃x:E, r(x) et montrons que ∃y:E, r(y) ⇒ (∀z:E, r(z)). D⇒
Soit x₀:E tel que r(x₀). U∃
Supposons ¬(∃y:E, r(y) ⇒ (∀z:E, r(z))) et cherchons une contradiction. RA
D'après les lois de De Morgan, on a ∀y:E, r(y) ∧ ¬(∀z:E, r(z))
∀y:E, r(y) ∧ (∃z:E, ¬r(z)). DM (⇐⇒)
On a donc r(x₀) ∧ ∃z:E, ¬r(z). U∀
On déduit que r(x₀) et ∃z:E, ¬r(z). U∧
Soit z₀:E tel que ¬r(z₀). U∃
On a aussi r(z₀) ∧ (∃z:E, ¬r(z)). U∀
Donc on a r(z₀) et ∃z:E, ¬r(z). U∧
On a r(z₀) et ¬r(z₀). D'où la contradiction. U¬

Example 4

Let *Thm*1 :

$$\forall x_1, \forall x_2, [U_1(x_1), U_2(x_2) \to R(x_1) \vee R(x_2)] \quad \to$$
$$[\forall y_1, (U_1(y_1) \to R(y_1))] \ \vee \ [\forall y_2, (U_2(y_2) \to R(y_2))]$$

Let $\mathcal{P}1$:

$$\lambda f.\mu\alpha.[\alpha]\lambda y.\lambda z.(z)\lambda x_1.\mu\alpha_1.[\alpha]\lambda y'.\lambda z'.(y')\lambda x_2.\mu\alpha_2.$$
$$((((f)x_1)x_2)\lambda u.[\alpha_1]u)\lambda v.[\alpha_2]v$$

Lemma

We have $\vdash \mathcal{P}1 : Thm1$.

Example 5

$$\begin{array}{lcl} \forall n:\mathbb{N}, P(n) & = & \forall n, \mathbb{N}[n] \to P(n) \\ \exists n:\mathbb{N}, P(n) & = & \exists n, \mathbb{N}[n] \wedge P(n) \\ n \leq m & = & \exists k:\mathbb{N}, n+k=m. \end{array}$$

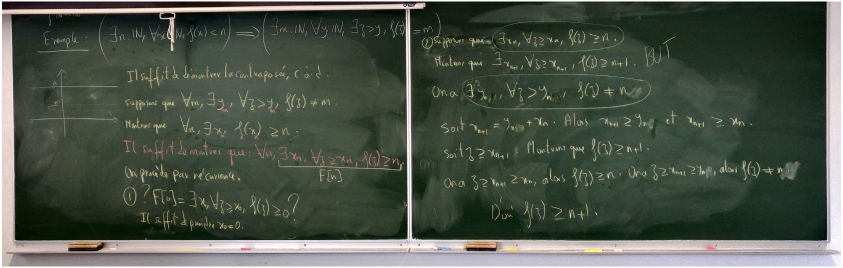
Let *Thm2* :

$$[\exists M:\mathbb{N}, \forall n:\mathbb{N},\ f(n) \leq M] \quad \to$$

$$[\exists m:\mathbb{N}, \forall k:\mathbb{N}, \exists k':\mathbb{N}, (k \leq k') \wedge (f(k') = m)]$$

We can find a $\lambda\mu$-term $\mathcal{P}2$ such that $\vdash \mathcal{P}2 : Thm2$.

Example 5


Exemple : $(\exists n:\mathbb{N}, \forall x:\mathbb{N}, f(x) < n) \Longrightarrow (\exists m:\mathbb{N}, \forall y:\mathbb{N}, \exists z > y, f(z) = m)$
Il suffit de démontrer la contraposée, c.à.d.
Supposons que $\forall m, \exists y_m, \forall z > y_m, f(z) \neq m$.
Montrons que $\forall n, \exists x_n, f(x_n) \geq n$.
Il suffit de montrer que : $\forall n, \underbrace{\exists x_n, \forall z \geq x_n, f(z) \geq n}_{F[n]}$.
On procède par récurrence.
① ? $F[0] = \exists x_0, \forall z \geq x_0, f(z) \geq 0$?
Il suffit de prendre $x_0 = 0$.
② Supposons que $\exists x_n, \forall z \geq x_n, f(z) \geq n$.
Montrons que $\exists x_{n+1}, \forall z \geq x_{n+1}, f(z) \geq n+1$. BUT
On a $\exists y_n, \forall z > y_n, f(z) \neq n$
Soit $x_{n+1} = y_n + x_n$. Alors $x_{n+1} \geq y_n$ et $x_{n+1} \geq x_n$.
Soit $z \geq x_{n+1}$. Montrons que $f(z) \geq n+1$.
On a $z \geq x_{n+1} \geq x_n$, alors $f(z) \geq n$. On a $z \geq x_{n+1} \geq y_n$, alors $f(z) \neq n$.
D'où $f(z) \geq n+1$.

Problem

Let $\theta = \lambda f.\lambda x.\mu\alpha.[\alpha](f)(f)\mu\beta.[\alpha](f)(f)(f)\mu\delta.[\beta](f)(f)\mu\gamma.[\beta](f)x$.

We have $\vdash \theta : \mathbb{N}[s^3(0)]$.

- We lose the property of unique integer representation.
- How do we recognize the value of an integer ?

Solutions (Parigot 1993)

- We can recognize the value of an integer using external algorithms by locating the redundant part.
- We can use storage operators to determine the value of an integer :

$$((\mathcal{S}\mathcal{O})\theta)\lambda x.x \quad \triangleright^* \quad \underline{3}.$$

- Add a new reduction rule μ', which is the symmetry of the reduction rule μ :

$$\theta \quad \triangleright^* \quad \underline{3}.$$

Outline

The reduction μ'

Definition

$$(\mu\alpha.M)N \quad \triangleright_{\mu} \quad \mu\alpha.M[[\alpha]U := [\alpha](U)N]$$

$$(N)\mu\alpha.M \quad \triangleright_{\mu'} \quad \mu\alpha.M[[\alpha]U := [\alpha](N)U]$$

Problems

- We lose the confluence of the system.

$$(\mu\alpha.x)\mu\beta.y \ \triangleright_{\mu}\ \mu\alpha.x$$

$$(\mu\alpha.x)\mu\beta.y \ \triangleright_{\mu'}\ \mu\beta.y$$

- We lose the subject reduction property.
  Raffalli (2001)

- We lose the strong normalization property.
  Battyanyi (2007)

Simply typed calculus (Battyanyi & Nour 2021)

In simply typed $\lambda\mu$-calculus (without $\forall$), we have :

- The uniqueness of integer representation.
- The subject reduction property.
- The weak normalization property.

Outline

Coding

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}\ \to_i \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash (M)N : B}\ \to_e$$

$$\frac{\Gamma, \alpha : \neg B \vdash M : \perp}{\Gamma \vdash \mu\alpha.M : B}\ \perp_i \qquad \frac{\Gamma, \alpha : \neg A \vdash M : A}{\Gamma, \alpha : \neg A \vdash [\alpha]M : \perp}\ \perp_e$$

$$\frac{\Gamma \vdash M : A[x]}{\Gamma \vdash ?M : \forall x,\ A[x]}\ \forall_i^1 \qquad \frac{\Gamma \vdash M : A[X]}{\Gamma \vdash ?M : \forall X,\ A[X]}\ \forall_i^2$$

$$\frac{\Gamma \vdash M : \forall x,\ A[x]}{\Gamma \vdash !M : A[t]}\ \forall_e^1 \qquad \frac{\Gamma \vdash M : \forall X,\ A[X]}{\Gamma \vdash !M : A[F]}\ \forall_e^2$$

$$\frac{\Gamma \vdash M : A[u] \qquad u = v}{\Gamma \vdash M : A[v]}\ =$$

Reduction rules

$$
\begin{array}{lll}
(\lambda x.M)N & \triangleright_\beta & M[x := N] \\
(\mu\alpha.M)N & \triangleright_\mu & \mu\alpha.M[[\alpha]U := [\alpha](U)N] \\
(N)\mu\alpha.M & \triangleright_{\mu'} & \mu\alpha.M[[\alpha]U := [\alpha](N)U] \\
!?M & \triangleright_\forall & M \\
[\alpha]\mu\beta.M & \triangleright_\rho & M[\beta := \alpha] \\
\mu\alpha.[\alpha]M & \triangleright_\theta & M \qquad \text{if } \alpha \notin FV(M) \\
\mu\alpha.\mu\beta.M & \triangleright_\varepsilon & \mu\alpha.M_\beta
\end{array}
$$

Questions

1. The uniqueness of integer representation.
2. The subject reduction property.
3. The weak normalization property.
4. The algorithmic content of some mathematical proofs.