Normalization Properties of $\lambda\mu$-Calculus Using Realizability Semantics

Péter Battyányi
University of Debrecen, Hungary

Karim Nour
Université Savoie Mont Blanc, France

The simply typed lambda calculus

The λ-terms:

$$\Lambda = \mathcal{V} \mid (\lambda\mathcal{V}.\Lambda) \mid (\Lambda)\Lambda,$$

where $\mathcal{V} = \{x, y, z, \ldots\}$ are variables.

We adopt Krivine's notation: $(M)N$ stands for the application of N to M.

Moreover,

$$(M_1)M_2 \ldots M_{n-1}M_n \rightleftharpoons (\ldots(M_1)M_2)\ldots)M_{n-1})M_n.$$

The simply typed lambda calculus

The types are built from a set $\mathcal{V}_{\mathbb{T}}$ of atomic types with the connective $\rightarrow$:

$$\mathbb{T} \;:=\; \mathcal{V}_{\mathbb{T}} \;\mid\; \mathbb{T} \rightarrow \mathbb{T}$$

Typing rules

A context Γ: a finite set of declarations of the form $x : A$.
Let $|\Gamma| = \{A \mid x : A \in \Gamma \text{ for some } x\}$. Then:

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

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

The λ-term M is called typable with type A, if $\Gamma \vdash M : A$ is derivable using the rules and axioms.

An example

Let $\Gamma = \{x : A \to (B \to C),\ y : A \to B,\ z : A\}$. Then,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : A \to (B \to C)} \quad \dfrac{}{\Gamma \vdash z : A}}{\Gamma \vdash (x)z : B \to C} \quad \dfrac{\dfrac{}{\Gamma \vdash y : A \to B} \quad \dfrac{}{\Gamma \vdash z : A}}{\Gamma \vdash (y)z : B}}{\Gamma \vdash ((x)z)(y)z : C}}{x : A \to (B \to C), y : A \to B \vdash \lambda z.((x)z)(y)z : A \to C}}{x : A \to (B \to C) \vdash \lambda yz.((x)z)(y)z : (A \to B) \to (A \to C)}}{\vdash \lambda xyz.((x)z)(y)z : (A \to (B \to C)) \to ((A \to B) \to (A \to C))}$$

Types as formulas

When we omit the indications of the terms, we obtain a correct derivation of the end formula.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{|\Gamma| \vdash A \to (B \to C)} \quad \dfrac{}{|\Gamma| \vdash A}}{|\Gamma| \vdash B \to C} \quad \dfrac{\dfrac{}{|\Gamma| \vdash A \to B} \quad \dfrac{}{|\Gamma| \vdash A}}{|\Gamma| \vdash B}}{|\Gamma| \vdash C}}{A \to (B \to C), A \to B \vdash A \to C}}{A \to (B \to C) \vdash (A \to B) \to (A \to C)}}{\vdash (A \to (B \to C)) \to ((A \to B) \to (A \to C))}$$

The Curry-Howard isomorphism

This phenomenon is referred to as the Curry–Howard isomorphism and was first observed by Curry in the 1930s and (re)discovered by Howard in 1968.
It was widely believed that the Curry–Howard isomorphism is valid only for intuitionistic logic, until Griffin and Murthy discovered in the 1990s that certain terms expressing control can be given the type $\neg\neg A \to A$.

The classical case

From that point on, several calculi realizing the Curry-Howard isomorphism for classical logic have been discovered. For example,

- Rehof and Sorensen's λ_{Δ},
- Curien and Herbelin's $\bar{\lambda}\mu\tilde{\mu}$,
- Krivine's $\lambda\mathcal{C}$,
- Berardi and Barbanera's λ^{Sym},
- Parigot's $\lambda\mu$.

Definitions and notation

- Let $\mathcal{V}_\lambda = \{x, y, z, \dots\}$ denote a set of λ-variables and $\mathcal{V}_\mu = \{\alpha, \beta, \gamma, \dots\}$ denote a set of μ-variables, respectively. Formation rules for $\lambda\mu$-terms:

$$\mathcal{T} \;:=\; \mathcal{V}_\lambda \;\mid\; \lambda\mathcal{V}_\lambda.\mathcal{T} \;\mid\; (\mathcal{T})\mathcal{T} \;\mid\; [\mathcal{V}_\mu]\mathcal{T} \;\mid\; \mu\mathcal{V}_\mu.\mathcal{T}$$

- In a $\lambda\mu$-term, the λ and μ operators bind the variables in their scope. We consider terms modulo correct renaming of bound variables.

Definitions and notation

The types are built from a set $\mathcal{V}_{\mathbb{T}}$ of atomic types and the constant $\bot$ with the connectives $\neg$ and $\to$.

$$\mathbb{T} \;:=\; \mathcal{V}_{\mathbb{T}} \cup \{\bot\} \;\mid\; \neg\mathbb{T} \;\mid\; \mathbb{T} \to \mathbb{T}$$

A context: a pair (Γ, Θ).

- Γ: finite set of declarations of the form $x : A$
- Θ: finite set of declarations of the form $\alpha : \neg A$

If $x : A \to \bot$, we may also write $\neg A$ for $A \to \bot$.

Definitions and notation

The typing rules are:

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

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

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

Definitions and notation

Below, let $\Gamma = \{x : \neg B \to \neg A,\ y : A,\ z : B\}, \Theta = \{\alpha : \neg B\}$ and $\Gamma' = \Gamma \setminus \{z : B\}$. Then

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma;\Theta \vdash z : B}{\Gamma;\Theta \vdash [\alpha]z : \bot}}{\Gamma';\Theta \vdash \lambda z.[\alpha]z : \neg B}}{\Gamma';\Theta \vdash (x)\lambda z.[\alpha]z : \neg A} \quad \Gamma';\Theta \vdash y : A}{\Gamma';\Theta \vdash ((x)\lambda z.[\alpha]z)y : \bot}}{\Gamma' \vdash \mu\alpha.((x)\lambda z.[\alpha]z)y : B}}{x : \neg B \to \neg A \vdash \lambda y.\mu\alpha.((x)\lambda z.[\alpha]z)y : A \to B}}{\vdash \lambda xy.\mu\alpha.((x)\lambda z.[\alpha]z)y : (\neg B \to \neg A) \to (A \to B)}$$

Definitions and notation

If we omit the terms annotating the type formulas, we obtain a full fledged proof of $(\neg B \to \neg A) \to (A \to B)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{|\Gamma| \vdash B^{(1)} \qquad |\Gamma| \vdash \neg B^{(2)}}{|\Gamma| \vdash \bot}}{|\Gamma'| \vdash \neg B}\,(1) \qquad |\Gamma'| \vdash \neg B \to \neg A^{(4)}}{|\Gamma'| \vdash \neg A} \qquad |\Gamma'| \vdash A^{(3)}}{\dfrac{\dfrac{\dfrac{|\Gamma'| \vdash \bot}{\neg B \to \neg A, A \vdash B}\,(2)}{\neg B \to \neg A \vdash A \to B}\,(3)}{\vdash (\neg B \to \neg A) \to (A \to B)}\,(4)}$$

Reduction rules

We use several types of substitution in the $\lambda\mu$-calculus.

- β-substitution: $M[x := N]$ is obtained from M by exchanging the free occurrences of x in M with N.
- μ-substitution: $M[\alpha :=_r N]$ is obtained from M when we replace all subterm occurrences $[\alpha]U$ of M by $[\alpha](U)N$ recursively.
- μ'-substitution: $M[\alpha :=_l N]$ is obtained from M when we replace all subterm occurrences $[\alpha]U$ of M by $[\alpha](N)U$ in a recursive manner.
- ρ-substitution: $M[\alpha := \beta]$ is obtained from M by replacing the free occurrences of α in M by β

The notation M_α stands for the erasure of every occurrence of $[\alpha]$ from M.

Reduction rules

With this in hand, we are able to define the reduction rules.

- β-reduction: $(\lambda x^A.M^B)N^A : B \to_\beta M^B[x := N] : B$
- μ-reduction: $(\mu\alpha^{\neg(A\to B)}.M^\perp)N^A : B \to \mu\alpha^{\neg B}.M^\perp[\alpha :=_r N] : B$
- μ'-reduction: $(N)^{A\to B}\mu\alpha^{\neg A}.M^\perp : B \to \mu\alpha^{\neg B}.M^\perp[\alpha :=_l N] : B$
- ρ-reduction: $[\alpha^{\neg A}]\mu\beta^{\neg A}.M^\perp : \perp \to_\rho M^\perp[\beta := \alpha] : \perp$
- θ-reduction: $\mu\alpha^{\neg A}.[\alpha^{\neg A}]M^A : A \to_\rho M^A : A$ provided $\alpha \notin fv(M)$
- ε-reduction: $\mu\alpha^{\neg A}.\mu\beta^{\neg\perp}.M^\perp : A \to_\varepsilon \mu\alpha^{\neg A}.M^\perp_{\beta} : A$.

Reduction rules

- β-reduction: eliminates subsequent occurrences of $\rightarrow_i$ and $\rightarrow_e$.

$$\dfrac{\dfrac{\Gamma, x : A; \Theta \vdash M : B}{\Gamma; \Theta \vdash \lambda x.M : A \rightarrow B}\ \rightarrow_i \qquad \Gamma \vdash N : A}{\Gamma; \Theta \vdash (\lambda x.M)N : B}\ \rightarrow_e \rightsquigarrow \Gamma; \Theta \vdash M[x := N] : B$$

where $\Gamma \vdash M[x := N] : B$ is obtained from $\Gamma \vdash M : B$ by exchanging all hypotheses $x : A$ with the derivation $\Gamma \vdash N : A$.

Reduction rules

- μ-reduction: μ propagates an argument toward occurrences marked by α.
- Given the declaration $\alpha : A \to B$, we have $M : \bot$. We consider the declaration $\alpha : B$ and replace in M each subterm of the form $[\alpha]P$ by $[\alpha](P)N$ to obtain a term of type B.
- At the proof level, μ-reduction corresponds to the simplification of a reasoning by contradiction, when an application of a $\bot$-elimination is replaced by $\bot$-eliminations of lower type complexity.

Reduction rules

$$\dfrac{\dfrac{\Gamma;\alpha:\neg(A\to B),\Theta\vdash M:\bot}{\Gamma;\Theta\vdash\mu\alpha.M:A\to B}\,\bot_e \qquad \Gamma;\Theta\vdash N:A}{\Gamma;\Theta\vdash(\mu\alpha.M)N:B}\,\to_e \rightsquigarrow\ \Gamma;\Theta\vdash\mu\alpha.M[\alpha:=_r N]:B$$

Locally,

$$[\alpha^{\neg(A\to B)}]U^{A\to B}\rightsquigarrow\ [\alpha^{\neg B}](U^{A\to B})N^{A}$$

Reduction rules

- μ'-reduction: μ' propagates the right-side functional part toward occurrences marked by an application with the μ-variable
- It is the symmetric counterpart of rule μ. Given the declaration $\alpha : \neg A$, we have $\mu\alpha.M : A$ by an application of $\bot_e$. If we also have $N : A \to B$, then $(N)\mu\alpha.M : B$ and $(N)\mu\alpha.M : B \to \mu\alpha.M[\alpha :=_I N]$. It suffices to start with the declaration $\alpha : B$ and to replace in M each subterm of the form $[\alpha]P$ by $[\alpha](N)P$.
- Here, we observe that there is no direct relation between the two types associated with α before and after the reduction.

Reduction rules

$$\dfrac{\Gamma;\Theta \vdash N : A \to B \qquad \dfrac{\Gamma;\alpha : \neg A, \Theta \vdash M : \bot}{\Gamma;\Theta \vdash \mu\alpha.M : A}\,\bot_e}{\Gamma;\Theta \vdash (N)\mu\alpha.M : B}\,\to_e \rightsquigarrow\ \Gamma;\Theta \vdash \mu\alpha.M[\alpha :=_I N] : B$$

Locally,

$$[\alpha^{\neg A}]U^A \rightsquigarrow\ [\alpha^{\neg B}](N^{A\to B})U^A$$

Simplification rules

- ρ-reduction: If we obtained a derivation of $[\beta]\mu\alpha.M : \bot$, then, necessarily, $\alpha, \beta : \neg A$ for some A. Hence, ρ-reduction can be interpreted as a simplification of proofs: it is enough to replace every occurrence of the assumption $\alpha : \neg A$ by the same type of assumption $\beta : \neg A$ throughout the derivation.
- θ-reduction: Given the declaration $\alpha : \neg A$, we have $M : A$, hence $\mu\alpha.[\alpha]M : A$. This term reduces simply to M, provided that α does not appear free in M.
- ε-reduction: Given the declaration $\alpha : \neg A$ and the relation $\mu\alpha.\mu\beta.M : A$, we necessarily have $\beta : \neg\bot$. We eliminate $\mu\beta$ as well as all occurrences of $[\beta]$ in M to obtain a term of type A. Hence, ε-rule removes a proof of $\bot$ obtained from $\bot$ by contradiction.

Why the μ'-rule?

Proposition

Let t be a (closed) term of the second order propositional λ-calculus. Then

$$t : \mathcal{N} = X \to (X \to X) \to X \qquad \text{iff} \qquad t = \lambda x \lambda f.(f)^n x$$

for some $n \in \mathbb{N}$.

On the other hand,

$$\lambda x \lambda f.\mu\alpha[\alpha](f)\mu\beta[\alpha](f)\mu\gamma[\beta](f)(f)\mu\beta[\gamma](f)\mu\gamma[\beta]x$$

is a $\lambda\mu$-term of type $\mathcal{N}$ with value 3.

Why the μ'-rule?

In Parigot's original calculus, a proof that n is a natural number may normalize to a term different from the Church numeral $\underline{n}$.

- Such terms are called *classical natural numbers*.
- Their effective numerical value may be difficult to recover.

Possible solutions

- An algorithm for extracting the value of a classical natural number (Parigot).
- A Krivine-type storage operator to simulate call-by-vallue in a call-by-name language (Nour).
- The introduction of the symmetric counterpart of μ as a new rule, which is called the μ'-rule (Parigot).

Reduction rules

Theorem (B.; Nour 2022)

Let x, f be two different λ-variables. If $\vdash M : \mathcal{N}$ and $((M)x)f \in \mathcal{WN}_{\beta\mu\mu'\rho\theta\varepsilon}$, then $\exists\, n \in \mathbb{N}$: $((M)x)f \twoheadrightarrow_{\beta\mu\mu'\rho\theta\varepsilon} (f)^n x$.

In particular, to extract the value of a $\lambda\mu$-term of type $\mathcal{N}$, we have make use of all the reductions $\beta\mu\mu'\rho\theta\varepsilon$.

Reduction rules

Let $M = \lambda x.\mu\alpha.[\alpha]\lambda g.(g)\mu\beta.\mu\gamma.[\alpha]\lambda f.(f)x$. Let $x : X$, $f : X \to X$, $g : X \to X$, $\alpha : \neg((X \to X) \to X)$, $\beta : \neg X$ and $\gamma : \neg\bot$, then $\vdash M : \mathcal{N}$. Furthermore,

$$
\begin{array}{rcl}
((M)x)f & = & (\underline{(\lambda x.\mu\alpha.[\alpha]\lambda g.(g)\mu\beta.\mu\gamma.[\alpha]\lambda f.(f)x)x})f \\
& \to_\beta & \underline{(\mu\alpha.[\alpha]\lambda g.(g)\mu\beta.\mu\gamma.[\alpha]\lambda f.(f)x)f} \\
& \to_\mu & \mu\alpha.[\alpha](\lambda g.(g)\mu\beta.\mu\gamma.[\alpha]\underline{(\lambda f.(f)x)f})f \\
& \to_\beta & \mu\alpha.[\alpha]\underline{(\lambda g.(g)\mu\beta.\mu\gamma.[\alpha](f)x)f} \\
& \to_\beta & \mu\alpha.[\alpha](f)\underline{\mu\beta.\mu\gamma.[\alpha](f)x} \\
& \to_\varepsilon & \mu\alpha.[\alpha](f)\underline{\mu\beta.[\alpha](f)x} \\
& \to_{\mu'} & \mu\alpha.\underline{[\alpha]\mu\beta.[\alpha](f)x} \\
& \to_\rho & \underline{\mu\alpha.[\alpha](f)x} \\
& \to_\theta & (f)x.
\end{array}
$$

Why de Groote's syntax?

The typing rule representing reductio ad absurdum by Parigot:

$$\frac{\Gamma; \alpha : \neg A, \Theta \vdash M : A}{\Gamma; \Theta \vdash \mu\alpha.[\alpha]M : \bot}\ \bot_i$$

De Groote split it into two rules:

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

Why de Groote's syntax?

In Parigot's original calculus (when $\Gamma = \{x : A,\ y : \neg\neg A\}, \Theta = \{\alpha : \neg A,\ \beta : \neg\bot,\ \gamma : \neg\bot\}$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma;\Theta \vdash x : A}{\Gamma;\Theta \vdash [\alpha]x : \bot}}{\Gamma;\Theta \setminus \{\beta : \neg\bot\} \vdash \mu\beta.[\alpha]x : \bot}}{\Gamma \setminus \{x : A\};\Theta \setminus \{\beta : \neg\bot\} \vdash \lambda x.\mu\beta.[\alpha]x : \neg A}}{\Gamma \setminus \{x : A\};\Theta \setminus \{\beta : \neg\bot\} \vdash (y)\lambda x.\mu\beta.[\alpha]x : \bot}}{\Gamma \setminus \{x : A\};\Theta \setminus \{\beta : \neg\bot\} \vdash [\gamma](y)\lambda x.\mu\beta.[\alpha]x : \bot}}{y : \neg\neg A;\gamma : \neg\bot \vdash \mu\alpha.[\gamma](y)\lambda x.\mu\beta.[\alpha]x : A}}{\gamma : \neg\bot \vdash \lambda y.\mu\alpha.[\gamma](y)\lambda x.\mu\beta.[\alpha]x : \neg\neg A \to A}$$

The hypothesis $\gamma : \neg\bot$ never disappears from the set of assumptions.

Why de Groote's syntax?

Possible solutions

- de Groote's modification of the syntax.
- λ-top calculus due to Herbelin and Ariola.

In what follows, we choose de Groote's version of the $\lambda\mu$-calculus.

Why de Groote's syntax?

The previous proof in de Groote's terms (assuming $\Gamma = \{x : A,\ y : \neg\neg A\}$, $\Theta = \{\alpha : \neg A\}$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma;\Theta \vdash x : A}{\Gamma;\Theta \vdash [\alpha]x : \bot}}{\Gamma \setminus \{x : A\};\Theta \vdash \lambda x.[\alpha]x : \neg A}}{\Gamma \setminus \{x : A\};\Theta \vdash (y)\lambda x.[\alpha]x : \bot}}{y : \neg\neg A \vdash \mu\alpha.(y)\lambda x.[\alpha]x : A}}{\vdash \lambda y.\mu\alpha.(y)\lambda x.[\alpha]x : \neg\neg A \to A}$$

Two calculi, two normalization problems

We study two reduction systems:

Without μ'

$\beta\mu\rho\varepsilon\theta$

With μ'

$\beta\mu\mu'\rho\varepsilon\theta$

Strong vs. weak normalization

Definition

1. $M \in \mathcal{SN}_{\mathcal{R}}$ if *every* $\mathcal{R}$-reduction sequence starting from M is finite.
2. $M \in \mathcal{WN}_{\mathcal{R}}$ if *some* $\mathcal{R}$-reduction sequence starting from M reaches a normal form.

Theorem

1. *(Parigot, 1993) Let $\mathcal{R} = \beta\mu\rho\varepsilon\theta$ in the typed $\lambda\mu$-calculus. Then $\mathcal{R}$ is strongly normalizing.*
2. *(B.; Nour, 2022) Let $\mathcal{R} = \beta\mu\mu'\rho\varepsilon\theta$ in the simply typed $\lambda\mu$-calculus. Then $\mathcal{R}$ is weakly normalizing.*

We remark that $\mu\mu'$ together with ρ is not strongly normalizing and together with ε is not even weakly normalizing (B., 2007).

Why a new proof method?

Earlier normalization proofs were mainly syntactic and arithmetical.

They typically required

- inductions on several measures;
- separate treatment of different reduction rules;
- technical commutation lemmas;
- substantial reconstruction when new reductions were added.

Goal

Develop a single semantic framework in which the desired normalization property follows by choosing an appropriate parameter for the interpretation.

The realizability strategy

The basic idea is to interpret each type A by a set of terms $\mathcal{I}(A)$, where $\mathcal{I}(A) \subseteq \perp\!\!\!\perp$.

The main goal is:

$$\Gamma; \Theta \vdash M : A \quad \Longrightarrow \quad M \in \mathcal{I}(A).$$

This would imply

$$M \text{ is typable} \quad \Longrightarrow \quad M \in \perp\!\!\!\perp.$$

Thus, by assigning different computational meanings to $\perp\!\!\!\perp$, we obtain different properties of the calculus. Hence, the semantics is parametrized by $\perp\!\!\!\perp$.

The realizability strategy[1]

The case of the simply typed lambda calculus with $\perp\!\!\!\perp = SN_\beta$ (Tait, 1967; Girard, 1971):

Definition

Let Λ denote the simply typed λ-calculus terms.

1. If $\mathcal{A}, \mathcal{B} \subseteq \Lambda$, then
$$\mathcal{A} \to \mathcal{B} = \{F \in \Lambda \mid (\forall a \in \mathcal{A})(Fa \in \mathcal{B})\}.$$
2. $[\![A]\!] = \perp\!\!\!\perp (= SN_\beta)$ if $A \in \mathcal{V}_{\mathbb{T}}$.
3. $[\![X]\!] = [\![U]\!] \to [\![V]\!]$, where $X = U \to V$.

[1] Based on Sørensen; Urzyczyn: Lectures on the Curry-Howard isomorphism

The realizability strategy

Definition

1 A set $\mathcal{S} \subseteq \perp\!\!\!\perp$ is *saturated* if

❶ For all $n \geq 0$ and $M_1, \ldots, M_n \in \perp\!\!\!\perp$:

$$(x)\, M_1 \cdots M_n \in \mathcal{S}$$

❷ For all $n \geq 1$ and $M_1, \ldots, M_n \in \perp\!\!\!\perp$

$$(M_0[x := M_1])\, M_2 \cdots M_n \in \mathcal{S} \;\Rightarrow\; (\lambda x.M_0)\, M_1\, M_2 \cdots M_n \in \mathcal{S}$$

2

$$\mathbb{S} = \{X \subseteq \Lambda \mid X \text{ is saturated}\}.$$

The realizability strategy

Lemma

1. $\perp\!\!\!\perp (= \mathcal{SN}_\beta) \in \mathbb{S}$;
2. $\mathcal{A}, \mathcal{B} \in \mathbb{S} \Rightarrow \mathcal{A} \to \mathcal{B} \in \mathbb{S}$;
3. $X \in \Pi \Rightarrow [\![X]\!] \in \mathbb{S}$. In particular, if $x : X$, then $x \in [\![X]\!]$.

The realizability strategy

Definition

1. A valuation is a map $\rho : \mathcal{V} \to \Lambda$, where $\mathcal{V}$ is the set of term variables. The valuation $\rho(x := N)$ is defined by

$$\rho(x := N)(y) = \begin{cases} N & \text{if } x = y, \\ \rho(y) & \text{otherwise.} \end{cases}$$

2. Let ρ be a valuation. Then

$$[\![M]\!]_\rho = M[x_1 := \rho(x_1), \ldots, x_n := \rho(x_n)],$$

where $\mathrm{FV}(M) = \{x_1, \ldots, x_n\}$.

The realizability strategy

Definition

3 Let ρ be a valuation. Then

$$\rho \models M : A \quad \text{iff} \quad [\![M]\!]_\rho \in [\![A]\!].$$

Also, $\rho \models \Gamma$ iff $\rho(x) \in [\![A]\!]$ for all $x : A \in \Gamma$.

4

$$\Gamma \models M : A \quad \text{iff} \quad \forall \rho : \ \rho \models \Gamma \ \Rightarrow \ \rho \models M : A.$$

The realizability strategy

Lemma (Adequacy)

$$\Gamma \vdash M : A \quad \Longrightarrow \quad \Gamma \models M : A.$$

Theorem

$$\Gamma \vdash M : A \quad \Rightarrow \quad M \in \mathrm{SN}_\beta.$$

Proof.

If $\Gamma \vdash M : A$, then $\Gamma \models M : A$. For each $x : X \in \Gamma$, let $\rho(x) = x$. Then $x \in [\![X]\!]$ holds since $[\![X]\!] \in \mathbb{S}$. Then $\rho \models \Gamma$, and we have

$$M = [\![M]\!]_\rho \in [\![A]\!] \subseteq \mathrm{SN}_\beta.$$

The case of the $\lambda\mu$-calculus

The main difference lies in the following typing rules:

$$\alpha : \neg A, M : \bot \implies [\alpha]M : \bot,$$
$$M : \bot, \alpha : \neg A \implies \mu\alpha.M : A..$$

If we interpret $[\![\bot]\!] = \perp\!\!\!\perp$, the interpretation must allow us to pass from

$$M \in \perp\!\!\!\perp \qquad \text{to} \qquad \mu\alpha.M \in [\![A]\!].$$

Crucial idea (Parigot, 1997)

For a set $\mathcal{X}$ of finite sequences of terms in $\perp\!\!\!\perp$, define

$$\mathcal{X} \rightsquigarrow \perp\!\!\!\perp = \{M \mid (M)\bar{P} \in \perp\!\!\!\perp \text{ for any initial segment } \bar{P} \text{ of an arbitrary element of } \mathcal{X}\}.$$

Intuitively:

- $\mathcal{X}$ describes a collection of finite sequences of arguments,
- $M \in \mathcal{X} \rightsquigarrow \perp\!\!\!\perp$ means that M sends all such sequences into $\perp\!\!\!\perp$.

This is the operation used to construct the reducibility candidates.

Saturation

A saturated set $\mathcal{S} \subseteq \Lambda$ is stable under term construction and expansion with β- and μ. In our case:

$$
\begin{aligned}
M \in \mathcal{S}, x \in \mathcal{V}_\lambda &\implies \lambda x.M \in \mathcal{S},\\
M \in \mathcal{S}, \alpha \in \mathcal{V}_\mu &\implies \mu\alpha.M \in \mathcal{S},\\
M \in \mathcal{S}, \alpha \in \mathcal{V}_\mu &\implies [\alpha]M \in \mathcal{S},\\
x \in \mathcal{V}_\lambda, \bar{N} \in \perp\!\!\!\perp^{<\omega} &\implies (x)\bar{N} \in \mathcal{S},\\
M \in \mathcal{T}, \bar{P} \in \mathcal{T}^{<\omega}, N \in \mathcal{S}, (M[x := N])\bar{P} \in \mathcal{S} &\implies (\lambda x.M)N\bar{P} \in \mathcal{S},\\
M \in \mathcal{T}, \bar{N} \in \mathcal{S}^{<\omega}, \mu\alpha.M[\alpha :=_r \bar{N}] \in \mathcal{S} &\implies (\mu\alpha.M)\bar{N} \in \mathcal{S}.
\end{aligned}
$$

Semantic consequence

Definition (The $\rightsquigarrow$ operator)

Let $\perp\!\!\!\perp$ be a saturated set and consider $\mathcal{K}, \mathcal{L} \subseteq \perp\!\!\!\perp$, and $\mathcal{X} \subseteq \perp\!\!\!\perp^{<\omega}$. Then:

$$\mathcal{K} \rightsquigarrow_{\perp\!\!\!\perp} \mathcal{L} = \{M \in \perp\!\!\!\perp \mid \forall N \in \mathcal{K}, (M)N \in \mathcal{L}\},$$

$$\mathcal{X} \rightsquigarrow_{\perp\!\!\!\perp} \perp\!\!\!\perp = \{M \in \perp\!\!\!\perp \mid \forall \bar{N} \in \mathcal{X}, \text{if } \bar{P} \sqsubseteq \bar{N}, \text{ then } (M)\bar{P} \in \perp\!\!\!\perp\},$$

where $\bar{P} \sqsubseteq \bar{N}$ means that $\bar{P}$ is an initial segment of $\bar{N}$.

Definition ($\perp\!\!\!\perp$-saturated sets)

Let $\perp\!\!\!\perp$ be saturated. We call $\mathcal{S}$ a $\perp\!\!\!\perp$-saturated set if the following property holds:

$$\exists \mathcal{X}_{\mathcal{S}} \subseteq \perp\!\!\!\perp^{<\omega}, \mathcal{S} = \mathcal{X}_{\mathcal{S}} \rightsquigarrow_{\perp\!\!\!\perp} \perp\!\!\!\perp.$$

Weakest precondition

For a semantic set $\mathcal{S}$, define its weakest precondition

$$\mathcal{S}^{\perp\!\!\!\perp} = \bigcup \{\mathcal{X} \subseteq \perp\!\!\!\perp^{<\omega} \mid \mathcal{S} = \mathcal{X} \rightsquigarrow \perp\!\!\!\perp\}.$$

The crucial property is

$$\mathcal{S} = \mathcal{S}^{\perp\!\!\!\perp} \rightsquigarrow \perp\!\!\!\perp.$$

Interpretation

- We define a $\perp\!\!\!\perp$-model $\mathcal{M}$ in a way such that, for every $\mathcal{S} \in \mathcal{M}$, $\mathcal{S}$ is $\perp\!\!\!\perp$-saturated.
- If $\mathcal{M}$ is a $\perp\!\!\!\perp$-model, then an $\perp\!\!\!\perp$-interpretation $\mathfrak{I}$ is a function such that
  - $\mathfrak{I}(\bot) = \perp\!\!\!\perp$,
  - $\mathfrak{I}(A \to B) = \mathfrak{I}(A) \rightsquigarrow \mathfrak{I}(B)$.

These conditions are tailored so that the interpretation of every type remains inside $\perp\!\!\!\perp$. That is, $\mathfrak{I}(A) \subseteq \perp\!\!\!\perp$ for any type A.

The adequacy theorem

Let $M \in \mathcal{T}$ and σ be the simultaneous substitution $[(x_i := M_i)_{1\leq i\leq n}; (\alpha_j :=_r \bar{N}_j)_{1\leq j\leq m}]$. Then $M\sigma$ is obtained from the $\lambda\mu$-term M by replacing each x_i by M_i and replacing inductively each subterm of the form $[\alpha_j]U$ in M by $[\alpha_j](U)\bar{N}_j$.

Theorem (General correctness theorem)

Let $\mathcal{M}$ be a $\perp\!\!\!\perp$-model, $\mathfrak{I}$ an $\mathcal{M}$-interpretation,
$\Gamma = \{x_i : A_i\}_{1\leq i\leq n}$, $\Theta = \{\alpha_j : \neg B_j\}_{1\leq j\leq m}$,
$M_i \in \mathfrak{I}(A_i)$ for all $1 \leq i \leq n$, $\bar{P}_j \sqsubseteq \bar{N}_j \in \mathfrak{I}(B_j)^{\perp\!\!\!\perp}$ for all $1 \leq j \leq m$ and
$\sigma = [(x_i := M_i)_{1\leq i\leq n}; (\alpha_j :=_r \bar{P}_j)_{1\leq j\leq m}]$.
If $\Gamma; \Theta \vdash M : A$, then $M\sigma \in \mathfrak{I}(A)$.

The adequacy theorem

Proof.

By induction on the derivation $\Gamma; \Theta \vdash M : A$.

We consider only the case $\bot_e$. In this case, $\Gamma; \Theta \vdash M = \mu\alpha.N$, hence, $\Gamma; \Theta', \alpha : \neg A \vdash N : \bot$, where $\Theta = \Theta' + (\alpha : A)$.

- Since $\varnothing \in \mathfrak{I}(A)^{\perp\!\!\!\perp}$ and $N\sigma = N\sigma'$ where $\sigma' = \sigma + [\alpha :=_r \varnothing]$, we obtain, by IH, $N\sigma \in \mathfrak{I}(\bot) = \perp\!\!\!\perp$, hence, $M\sigma = \mu\alpha.N\sigma \in \perp\!\!\!\perp$.
- Let $\bar{P} \sqsubseteq \bar{N} \in \mathfrak{I}(A)^{\perp\!\!\!\perp}$. We have $(M\sigma)\bar{P} = (\mu\alpha.N\sigma)\bar{P} \twoheadrightarrow_\mu \mu\alpha.N\sigma''$ where $\sigma'' = \sigma + [\alpha :=_r \bar{P}]$. By IH, $N\sigma'' \in \perp\!\!\!\perp = \mathfrak{I}(\bot)$ and $\mu\alpha.N\sigma'' = \mu\alpha.N\sigma[\alpha :=_r \bar{P}] \in \perp\!\!\!\perp$, hence $(M\sigma)\bar{P} \in \perp\!\!\!\perp$. Therefore $M\sigma = \mu\alpha.N\sigma \in \mathfrak{I}(A)$.

Correctness

Corollary (Correctness lemma)

If $\Gamma; \Theta \vdash M : A$ *and* $\perp\!\!\!\perp$ *is a saturated set, then* $M \in \perp\!\!\!\perp$.

Proof.

Assume $\Gamma = \{x_i : A_i\}_{1 \leq i \leq n}$ and $\Theta = \{\alpha_j : \neg B_j\}_{1 \leq j \leq m}$. Let $\mathcal{M}$ be the $\perp\!\!\!\perp$-model containing only the set $\perp\!\!\!\perp$ and $\mathfrak{I}$ the $\mathcal{M}$-interpretation defined by $\mathfrak{I}(X) = \perp\!\!\!\perp$ for all $X \in \mathcal{V}_{\mathbb{T}}$. Since $x_i \in \mathfrak{I}(A_i)$ for all $1 \leq i \leq n$ and $\emptyset \in \mathfrak{I}(B_j)^{\perp\!\!\!\perp}$ for all $1 \leq j \leq m$, then, by the general correctness lemma,
$M = M[(x_i := x_i)_{1 \leq i \leq n}; (\alpha_j :=_r \emptyset)_{1 \leq r \leq j}] \in \mathfrak{I}(A) \subseteq \perp\!\!\!\perp$. □

The Parameter $\perp\!\!\!\perp$

The semantics is parameterized by the set of terms $\perp\!\!\!\perp$. Different choices of $\perp\!\!\!\perp$ will yield different consequences.

To prove a normalization property for all typable terms, it suffices to prove that the corresponding set $\perp\!\!\!\perp$ is saturated.

Case I: Strong normalization

Take

$$\perp\!\!\!\perp = \mathcal{T}_t \cap \mathcal{SN}_{\beta\mu\rho\varepsilon\theta}.$$

The main technical work is to verify the six saturation conditions.

Result

The set $\perp\!\!\!\perp$ is saturated.

Hence, by the correctness lemma,

$$M \in \mathcal{T}_t \quad \Longrightarrow \quad M \in \mathcal{SN}_{\beta\mu\rho\varepsilon\theta}.$$

Why μ' changes the picture

Reductions μ, μ' and ρ together destroy strong normalization.

Let

$$U = \mu\alpha.[\alpha][\alpha]x, \qquad M = (\mu\beta.U)U.$$

Taking $x : \bot, \alpha : \neg\bot, \beta : \neg(\bot \to \bot)$, M is typable, but there is a reduction sequence

$$M \longrightarrow_{\mu'} \mu\alpha.[\alpha](\mu\beta.U)[\alpha](\mu\beta.U)x \longrightarrow_{\mu} \mu\alpha.[\alpha](\mu\beta.U)[\alpha]\mu\beta.U \longrightarrow_{\rho} \mu\alpha.[\alpha]M(\longrightarrow_{\theta} M).$$

Therefore

$$M \notin \mathcal{T}_t \cap \mathcal{SN}_{\beta\mu\mu'\rho\varepsilon\theta}.$$

Case II: Weak normalization

The case of $\beta\mu\mu'\rho\varepsilon\theta$-reduction must concern *weak*, rather than strong, normalization. Now take

$$\perp\!\!\!\perp' = \mathcal{T}_t \cap \mathcal{WN}_{\beta\mu\mu'\rho\varepsilon\theta}.$$

Result

$\perp\!\!\!\perp'$ is saturated.

Hence,

$$M \in \mathcal{T}_t \quad \Longrightarrow \quad M \in \mathcal{WN}_{\beta\mu\mu'\rho\varepsilon\theta}.$$

In both cases, the assumption of typability is essential for the proofs to work.

Summary

Advantages

- All reduction rules are treated simultaneously. Logical (β, μ, μ') and simplification rules $(\rho, \varepsilon, \theta)$ need not be separated.
- The parameter $\perp\!\!\!\perp$ makes the proof "reusable".

Disadvantage

Unlike the "combinatorial" proofs, realizability proofs do not allow us to extract normalization algorithms from the proofs.