Syntax and Models of a non-Associative Composition of Programs and Proofs

Guillaume Munch-Maccagnoni

December 10th 2013


informatiques mathématiques
Inria

Metaphor

Let us consider a question for a human

$$(1+2)\times(3+4)$$

$$\begin{aligned}
&= (1+2)\times(3+4)\\
&= 3\times(3+4)\\
&= 3\times(3+4)\\
&= 3\times 7\\
&= 3\times 7\\
&= 21
\end{aligned}$$

Metaphor

The same question asked to a machine

Reverse polish notation:

$$12+34+\times$$

	Expression	Stack	
	$1\,2+3\,4+\times$		push
▷	$2+3\,4+\times$	1	push
▷	$+\,3\,4+\times$	$2,1$	add
▷	$3\,4+\times$	3	push
▷	$4+\times$	$3,3$	push
▷	$+\times$	$4,3,3$	add
▷	$\times$	$7,3$	mult
▷		21	

Metaphor

Moral

There may be many equivalent representations but some are better than others.

Metaphor

In this thesis

Language ⟶ Representation ⟶ Model

Language of Proofs, of Programs

Model Mathematical (Denotation), Physical (Implementation)

Representation Two formalisms of which we show the unity: Sequent calculus (after Gentzen), Continuation-based models (after Landin).

The polarisation hypothesis

Introducing the idea of interaction

We take into account the context of a program or proof. Below: the context is modelled with a stack.

	Expression	Stack	
	$\mathsf{match}\,(\lambda x.\iota_1(t))\,u\,\mathsf{with}\,(v\|w)$	$\star$	push
$\triangleright$	$(\lambda x.\iota_1(t))\,u$	$(v\|w)\cdot\star$	push
$\triangleright$	$\lambda x.\iota_1(t)$	$u\cdot(v\|w)\cdot\star$	pop
$\triangleright$	$\iota_1(t[u/x])$	$(v\|w)\cdot\star$	branch
$\triangleright$	v	$t[u/x]\cdot\star$	

The polarisation hypothesis

Introducing the idea of interaction

Interactive aspects of the Curry-Howard correspondence

- In the proof theory of classical logic
- In the continuation-based modelling of evaluation order and control
- Categorical modelling of effects

The polarisation hypothesis

Introducing polarities

Polarities are a legacy of intuitionism:

- Positive connectives: $\exists, \vee$ …
- Negative connectives: $\forall, \to$ …

They become more important when interaction is taken into account.

The polarisation hypothesis

Introducing polarities

1. Classical proofs: polarities determine cut elimination (constructiveness)
2. Continuation-based modelling of programs: polarities determine whether a continuation is meant to be passed or to be applied (evaluation order)
3. Categories: Polarisation amounts to relaxing the associativity of composition ("Blass" phenomenon)

The polarisation hypothesis

Polarisation-related phenomena

- Blass problem in game semantics (Abramsky, Melliès)
- Thunks in call-by-value (Hatcliff and Danvy, Führmann)
- Value-restriction of polymorphism in CBV (Harper and Lillibridge)
- Arithmetical hierarchy in Peano arithmetic
- Focalisation in proof search (Andreoli)
- Orthogonality-based logical relations (Girard, Krivine, Pitts)

And probably more

Contribution

What

We contribute to the understanding of the nature, role and mechanisms of polarisation with:

1. a positive characterisation of polarisation where composition is not associative: duploids;
2. a decomposition of CPS translations in three meaningful steps (for *shift*/*reset* operators and 4 call-by-name variants);
3. a formulae-as-types correspondence for a Curry-style classical natural deduction that satisfies $\neg\neg A \simeq A$

Contribution

How

Introducing **L** calculi, for:

- modelling programming languages (after Landin)
- investigating the structure of proofs (after Gentzen)

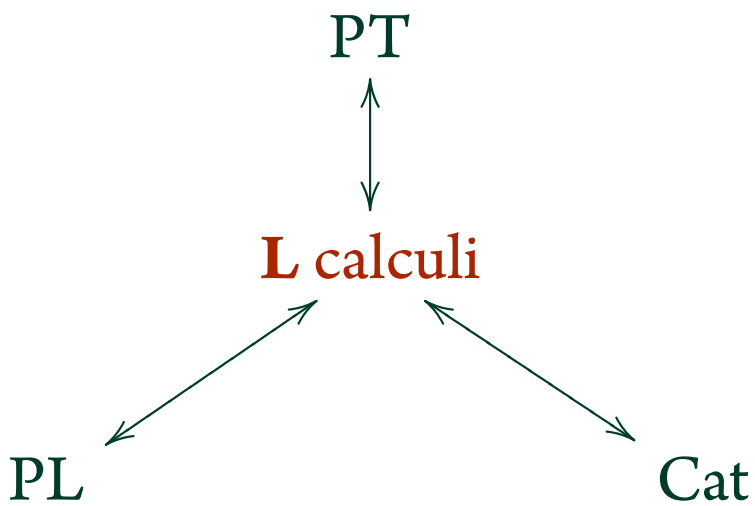
Inspired from:

- the modelling of effects (after Moggi)
- the quest for a link between categorical duality and continuations (after Filinski)
- the interactive notion of construction (after Girard, Krivine)

Contribution

Why

Exhibiting unifying principles behind an extension to interaction of the correspondence between proof theory, programming languages and category theory.



Four ideas

for modelling interaction

1. Explicit contexts, yielding abstract machines.
2. Solving equations with adjoints, yielding a correspondence with sequent calculus.
3. Introducing the $\tilde{\mu}$ binder, uncovering a symmetry with simplifying virtues.
4. Considering polarisation, yielding extensionality.

Abstract machines

First ingredient: Explicit contexts

The same example as before:

$$\begin{aligned}
&\langle \mathsf{match}\ (\lambda x.\iota_1(t))\ u\ \mathsf{with}\ (v|w) \parallel e\rangle \\
&\rhd_{\mathrm{R}} \langle (\lambda x.\iota_1(t))\ u \parallel (v|w)\cdot e\rangle \\
&\rhd_{\mathrm{R}} \langle \lambda x.\iota_1(t) \parallel u\cdot(v|w)\cdot e\rangle \\
&\rhd_{\mathrm{R}} \langle \iota_1(t[u/x]) \parallel (v|w)\cdot e\rangle \\
&\rhd_{\mathrm{R}} \langle v \parallel t[u/x]\cdot e\rangle
\end{aligned}$$

Commands $c ::= \langle t \parallel e\rangle$

Contexts $e ::= \star \mid \mathsf{fst}\cdot e \mid \mathsf{snd}\cdot e \mid (u|v)\cdot e \mid u\cdot e$

(for now contexts are stacks)

Abstract machines

Second ingredient: Solving equations with adjoints

Elimination rules:

$$\langle \mathsf{fst}(u) \,\|\, e \rangle \rhd_{\mathrm{R}} \langle u \,\|\, \mathsf{fst}\cdot e \rangle$$

$$\langle \mathsf{snd}(u) \,\|\, e \rangle \rhd_{\mathrm{R}} \langle u \,\|\, \mathsf{snd}\cdot e \rangle$$

$$\langle \mathsf{match}\, t\, \mathsf{with}\, (u|v) \,\|\, e \rangle \rhd_{\mathrm{R}} \langle t \,\|\, (u|v)\cdot e \rangle$$

$$\langle t\, u \,\|\, e \rangle \rhd_{\mathrm{R}} \langle t \,\|\, u\cdot e \rangle$$

are all "adjoints" (in the terminology borrowed by Girard):

$$\langle \tau^*(t) \,\|\, e \rangle \rhd_{\mathrm{R}} \langle t \,\|\, \tau(e) \rangle$$

Abstract machines

Second ingredient: Solving equations with adjoints

We introduce the adjoint construct c^* with the following reduction rule:

$$\langle c^* \,\|\, e\rangle \rhd_{\mathrm{R}} c[e/\star]$$

(essentially Parigot's μ)

The adjoint is used to solve the reduction rules which are therefore derived:

$$\mathsf{fst}(t) \stackrel{\text{def}}{=} \langle t \,\|\, \mathsf{fst}\cdot\star\rangle^*$$

$$\mathsf{snd}(t) \stackrel{\text{def}}{=} \langle t \,\|\, \mathsf{snd}\cdot\star\rangle^*$$

$$\mathsf{match}\ t\ \mathsf{with}\ (u|v) \stackrel{\text{def}}{=} \langle t \,\|\, (u|v)\cdot\star\rangle^*$$

$$t\ u \stackrel{\text{def}}{=} \langle t \,\|\, u\cdot\star\rangle^*$$

(Classical variant: solving the reduction rules of the operators and C and k_π with μ)

Abstract machines

Reward: Relationship with sequent calculus

$$\dfrac{\Gamma \vdash t : A \to B \qquad \dfrac{\Gamma \vdash u : A \qquad \overline{\Gamma \mid \star : B \vdash \star : B}}{\Gamma \mid u \cdot \star : A \to B \vdash \star : B}}{\dfrac{\langle t \parallel u \cdot \star \rangle : (\Gamma \vdash \star : B)}{\Gamma \vdash \langle t \parallel u \cdot \star \rangle^* : B}}$$

$$= t\ u$$

Calculus L

Third ingredient: $\tilde{\mu}$ binder

The $\tilde{\mu}$ binder is symmetric to the adjoint construct:

$$\langle t \,\|\, \tilde{\mu}x.c \rangle \rhd_{\mathrm{R}} c[t/x]$$

Application. Solving the equations of branching:

$$\langle \iota_1(t) \,\|\, (u|v)\cdot e \rangle \rhd_{\mathrm{R}} \langle u \,\|\, t\cdot e \rangle$$

$$\langle \iota_2(t) \,\|\, (u|v)\cdot e \rangle \rhd_{\mathrm{R}} \langle v \,\|\, t\cdot e \rangle$$

gives:

$$(u|v)\cdot e \overset{\text{def}}{=} (e_1|e_2)$$

where:

$$e_1 \overset{\text{def}}{=} \tilde{\mu}x.\langle u \,\|\, x\cdot e \rangle$$

$$e_2 \overset{\text{def}}{=} \tilde{\mu}x.\langle v \,\|\, x\cdot e \rangle$$

Calculus L

Third ingredient: $\tilde{\mu}$ binder

Consequence

The context is no longer necessarily a stack

Calculus L

Reward

Commutative cuts are derived

$$\dfrac{\dfrac{\dfrac{}{A \vee B}\quad \dfrac{\begin{array}{c}[A]\\ \vdots\\ C \to D\end{array}\quad \dfrac{\begin{array}{c}[B]\\ \vdots\\ C \to D\end{array}}{C \to D}}{C \to D}\quad \begin{array}{c}\vdots\\ C\end{array}}{D}$$

$$\simeq_{\mathrm{R}}$$

$$\dfrac{\dfrac{}{A \vee B}\quad \dfrac{\begin{array}{c}\vdots\\ C\end{array}\quad \dfrac{\begin{array}{c}[A]\\ \vdots\\ C \to D\end{array}}{D}}{D}\quad \dfrac{\begin{array}{c}\vdots\\ C\end{array}\quad \dfrac{\begin{array}{c}[B]\\ \vdots\\ C \to D\end{array}}{D}}{D}}{D}$$

Calculus L

Benefits of $\tilde{\mu}$

- Highlights choices in the evaluation order (Curien and Herbelin)
- Correspondence with sequent calculus (Curien and Herbelin, Wadler)
- Simpler equational theory: no more commutative cuts and related rules
- Completeness of contexts
- Simplifies orthogonality-based logical relations (M.-M.)

$$\mathbf{L} = \mu + \tilde{\mu}$$

Fourth ingredient: Polarisation

Guillaume Munch-Maccagnoni. Focalisation and Classical Realisability. In *Proc. CSL '09*, LNCS. Springer-Verlag, 2009

We define how terms and variables can be of either polarity (in the sense of Girard and Danos, Joinet and Schellinx)

Positive polarity The evaluation of a positive term is strict (call by value)

Negative polarity The evaluation of a negative term is delayed (call by name)

Having both is essential for extensionality

Fourth ingredient: Polarisation

Introducing a negative characterisation of polarisation

$$\mathsf{let}\, x^{\ominus} = (\mathsf{let}\, y^{+} = t_1\, \mathsf{in}\, t_2)\, \mathsf{in}\, t_3$$

vs. $\mathsf{let}\, y^{+} = t_1\, \mathsf{in}\, \mathsf{let}\, x^{\ominus} = t_2\, \mathsf{in}\, t_3$

- Above: computes t_3 first
- Below: computes t_1 first

Negative characterisation of polarisation

Rejecting the hypothesis that composition is associative *a priori*.

So-called "Blass problem" (Abramsky, Melliès)

Fourth ingredient: Polarisation

Non-associativity everywhere

$$(h \circ g) \bullet f \neq h \circ (g \bullet f)$$

- $\bullet$: composition in call by value
- $\circ$: composition in call by name

ML `let y = f x in h (fun () -> g y)` $\neq$ `h (fun () -> g (f x))`

Haskell `(\y->h (g y)) $! (f x)` $\neq$ `h (g $! (f x))`

Direct models

Various approaches to polarisation

1. Simplifying hypotheses: linearity, call-by-value, call-by-name
2. Indirect models: continuation calculi, call-by-push-value
3. Here: we propose to model proofs and programs through a structure where composition is not associative, which characterises polarisation directly.

Direct models

A direct denotational model offers an exact correspondence between operations of the model and constructions of the language.

Direct models

Comparative Table

Evaluation order	By value	By name	Polarised
Direct model	Thunk (Führmann)	Runnable monad (ex: Selinger)	Duploid
Indirect model	Monad T	Co-monad L	Adjunction $F \dashv G$
Programs	Kleisli maps $P \to TQ$	co-Kleisli maps $LN \to M$	Oblique maps $FP \to N$ $\simeq P \to GN$
Syntactic data	Values	Stacks	Both
Completion into	Thunkable expressions	Linear evaluation contexts	Both

The duploid construction

Inspired from Girard, Danos-Joinet-Schellinx, Laurent

Let $\uparrow \dashv \downarrow : \mathscr{N} \to \mathscr{P}$ be an adjunction:

$$\frac{P \to \downarrow N}{\uparrow P \to N} \ (\simeq)$$

P, Q objects of $\mathscr{P}$

N, M objects of $\mathscr{N}$

Polarised translation Recipe for defining a notion of morphism $A \to B$ for any A, B among the objects of $\mathscr{P}$ and $\mathscr{N}$.

Definition An *oblique morphism* $f : P \to N$ is either $P \to \downarrow N$ or $\uparrow P \to N$

The duploid construction

Inspired from Girard, DJS, Laurent

White composition

$$\dfrac{\dfrac{f : P \to N}{f : P \to {\downarrow}N}\,(\simeq) \qquad \dfrac{g : {\downarrow}N \to M}{g : {\downarrow}N \to {\downarrow}M}\,(\simeq)}{\dfrac{g \circ f : P \to {\downarrow}M}{g \circ f : P \to M}\,(\simeq)}\,(\circ)$$

Black composition

$$\dfrac{\dfrac{f : P \to {\uparrow}Q}{f : {\uparrow}P \to {\uparrow}Q}\,(\simeq) \qquad \dfrac{g : Q \to N}{g : {\uparrow}Q \to N}\,(\simeq)}{\dfrac{g \bullet f : {\uparrow}P \to N}{g \bullet f : P \to N}\,(\simeq)}\,(\bullet)$$

The duploid construction

Inspired from Girard, DJS, Laurent

A morphism $A \to B$ is an oblique morphism:

$$\boxed{A^+ \to B^\ominus}$$

with

$$A^+ \stackrel{\text{def}}{=} \begin{cases} \downarrow N & \text{if } A = N \\ P & \text{if } A = P \end{cases}$$

$$A^\ominus \stackrel{\text{def}}{=} \begin{cases} N & \text{if } A = N \\ \uparrow P & \text{if } A = P \end{cases}$$

$g \bullet f$ composition in $\mathscr{N}$ of $A \xrightarrow{f} P \xrightarrow{g} B$

$g \circ f$ composition in $\mathscr{P}$ of $A \xrightarrow{f} N \xrightarrow{g} B$

Duploids

Composition in a duploid is written:

$g \circledcirc f$ when the middle object is of any polarity

$g \circ f$ when the middle object is negative

$g \bullet f$ when the middle object is positive

$$(h \bullet g) \bullet f = h \bullet (g \bullet f)$$

$$(h \circ g) \circ f = h \circ (g \circ f)$$

$$(h \bullet g) \circ f = h \bullet (g \circ f)$$

$$(h \circ g) \bullet f \neq h \circ (g \bullet f)$$

See Loday's duplicial algebras

Jean-Louis Loday. Generalized bialgebras and triples of operads. *arXiv preprint math/0611885*, 2006

Duploids

Linear and thunkable morphisms

Definition

Linear morphism f associates to its right $\quad f \circ (g \circ h) = (f \circ g) \circ h$

Thunkable morphism f associates to its left $\quad h \circ (g \circ f) = (h \circ g) \circ f$

$\mathscr{D}_l$ category of linear morphisms

$\mathscr{D}_t$ category of thunkable morphisms

Duploids

Hom-functor: comparison

Category $\mathscr{C}$: hom-functor $\mathscr{C}(-,=) : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \to \boldsymbol{Set}$

Duploid $\mathscr{D}$: hom-functor $\mathscr{D}(-,=) : {\mathscr{D}_t}^{\mathrm{op}} \times \mathscr{D}_l \to \boldsymbol{Set}$

Duploids

Structure of Shifts

$\mathscr{N}$ sub-category of morphisms $N \to M$.

$\mathscr{P}$ sub-category of morphisms $P \to Q$.

Characterisation of shifts

- $\Uparrow : \mathscr{D}_l \to \mathscr{N}_l$ equivalence of categories
- $\Downarrow : \mathscr{D}_t \to \mathscr{P}_t$ equivalence of categories

Corollary

1. *Adjunction* $\boxed{\Downarrow \dashv \Uparrow}$ *with* $\Downarrow : \mathscr{N} \to \mathscr{P}$ *and* $\Uparrow : \mathscr{P} \to \mathscr{N}$.
2. *Adjunction* $\boxed{\Uparrow \dashv \Downarrow}$ *with* $\Downarrow : \mathscr{N}_l \to \mathscr{P}_t$ *and* $\Uparrow : \mathscr{P}_t \to \mathscr{N}_l$.

Duploids

Functor: comparison

Functor $F : \mathscr{C} \to \mathscr{C}'$ **between categories** gives a natural transformation $\mathscr{C}(-,=) \to \mathscr{C}'(F-,F=)$

Functor $F : \mathscr{D} \to \mathscr{D}'$ **between duploids** preserves linearity and thunkability and gives a natural transformation $\mathscr{D}(-,=) \to \mathscr{D}'(F_t-,F_l=)$

Duploids

Theorem

$$\mathcal{Dupl} \triangleleft \mathcal{Adj}$$

i.e.: the duploid construction (or polarised translation) extends into a functor $j : \mathcal{Adj} \to \mathcal{Dupl}$ *which admits a full and faithful right adjoint* $i : \mathcal{Dupl} \to \mathcal{Adj}$.

$\mathcal{Dupl}$ is the category of duploids and duploid functors.
$\mathcal{Adj}$ is the category of adjunctions and *pseudo-morphisms of adjunctions*.

Results

1. Category $\mathcal{Dupl}$ of duploids and duploid functors.
2. $\mathcal{Dupl} \triangleleft \mathcal{Adj}$, in particular:
   - Every duploid comes from an adjunction,
   - Not all adjunctions come from a duploid.
3. Characterisation of adjunctions that arise from duploids. (Completion of values and stacks + quotient)
4. Internal language / Abstract machines.
5. The duploid is a category if and only if the adjunction is *idempotent*.
6. Kleisli categories correspond to duploids where $\Downarrow$ or $\Uparrow$ are bijective on objects.

Decomposing CPS translations

Continuation-passing style translations

- provide implementations (Sussman and Steele) to programming languages,
- provide denotational semantics (Strachey and Wadsworth) to programming languages,
- are still an active topic.

The goal is to explain continuation-passing-style translations through **L** calculi. To do so we decompose CPS translations for delimited control calculi through CPS translations for **L** calculi.

Decomposing CPS translations

Delimited control operators

- First-class context delimiters model the fact that captured contexts have a finite extent (Felleisen).
- We focused on Danvy and Filinski's *shift/reset* variant. $(S/<\cdot>)$

$$<E[S(\lambda x.t)]> \triangleright <t[\lambda y.<E[y]>/x]>$$
$$<V> \triangleright V$$

- We used the decomposition of Herbelin and his collaborators with the binder $\mu\tilde{\mathrm{tp}}$. This binder models an auxiliary stack of contexts:

$$\langle t \,\|\, e_0 \rangle [e_1, \ldots, e_n]$$

Decomposing CPS translations

- The *shift*/*reset* calculus in call-by-value as well as four call-by-name variants embed, by "*solving equations*", into a single polarised calculus **L** called $\mathbf{L}_{\mathrm{pol},\widehat{\mathrm{tp}}^+}$.
- Method: showing that the CPS translations factor through $\mathbf{L}_{\mathrm{pol},\widehat{\mathrm{tp}}^+}$, thus refining well-known decompositions of $\lambda\mu$ calculi (Laurent and others).
- All the translations preserve equivalences. All the translations starting from **L** calculi simulate reductions as well.
- The call-by-name setting is not simply deduced from the call-by-value setting "by duality".

Involutive negation

- We review the issue of the involution of negation in the proof theory of classical logic
- We give a Curry-style classical natural deduction satisfying $\neg\neg A \simeq A$, which enjoys a formulae-as-types notion of construction.

Motivations

- Explaining an isomorphism of types such as $\neg\neg A \simeq A$ (as in Girard's **LC**).
- We gave in CSL'09 a syntax for Girard's **LC**, but defining a notion of quasi-proofs (or programs) was not immediate due to the perfect symmetry. Yet quasi-proofs are essential in classical realisability to obtain models in the sense of model theory.

Involutive negation

Idea 1

We distinguish the type $\sim A$ of captured stacks from the type $A \to \bot$ of continuations

- From Girard and Danos-Joinet-Schellinx: the rules of negation hide cuts, in particular the following one:

$$\frac{\Gamma, N \vdash \Delta}{\Gamma \vdash \neg N, \Delta}$$

- From Krivine: distinguishing the pseudo-type $\mathcal{X}^- \stackrel{\text{def}}{=} \{\mathsf{k}_\pi \mid \pi \in \mathcal{X}\}$ from the type of continuations.
- From Felleisen-Clements: high-level access to captured stacks.

Involutive negation

Idea 2

Delimited control provides a constructive interpretation for $\bot$ in an untyped setting.

(Inspired from Herbelin et al.)

Involutive negation

The $\lambda\ell$ calculus

$$\ell : (A \to \bot) \to {\sim}A$$
$$\mathsf{send} : {\sim}A \to A \to \bot$$

ℓ refines C and j refines k.

$$\langle \ell \| t_\ominus \cdot e_+ \rangle [\pi_\ominus, \sigma] \rhd_R \langle t_\ominus \| \mathsf{j}_{e_+} \cdot \pi_\ominus \rangle [\sigma]$$
$$\langle \mathsf{j}_{\pi_+} \| \pi \rangle [\sigma] \rhd_R \langle [\pi] \| \pi_+ \rangle [\sigma]$$
$$\langle \mathsf{send} \| [\pi_\varepsilon] \cdot t_\varepsilon \cdot \pi'_\ominus \rangle [\sigma] \rhd_R \langle t_\varepsilon \| \pi_\varepsilon \rangle [\pi'_\ominus, \sigma]$$

+ constants that can access the contents of captured stacks $[\pi]$.

Involutive negation

Method

- Extensional reasoning, necessary for proving isomorphisms, is provided on $\lambda\ell$ by the calculus $\mathbf{L}_{\mathrm{pol},\widehat{\mathrm{tp}}^{\ominus}}$.
- The $\lambda\ell$ calculus is then a subset of terms of $\mathbf{L}_{\mathrm{pol},\widehat{\mathrm{tp}}^{\ominus}}$ with no free co-variables which is closed under composition.

Characterisation of linearity and thunkability

A context e is linear if and only if one of the following two equivalent properties holds:

1. for all c, t, q, q' one has $\langle t \,\|\, \tilde{\mu}q'.\langle \mu q.c \,\|\, e\rangle\rangle \simeq_{\mathrm{RE}_p} \langle \mu q.\langle t \,\|\, \tilde{\mu}q'.c\rangle \,\|\, e\rangle$,
2. for all c, α one has $\langle \mu\alpha.c \,\|\, e\rangle \simeq_{\mathrm{RE}_p} c[e/\alpha]$.

A term t is thunkable if and only if one of the following two equivalent properties holds:

1. for all c, e, q, q' one has $\langle \mu q'.\langle t \,\|\, \tilde{\mu}q.c\rangle \,\|\, e\rangle \simeq_{\mathrm{RE}_p} \langle t \,\|\, \tilde{\mu}q.\langle \mu q'.c \,\|\, e\rangle\rangle$,
2. for all c, x, $\langle t \,\|\, \tilde{\mu}x.c\rangle \simeq_{\mathrm{RE}_p} c[t/x]$.

Characterisation of linearity and thunkability

Semantic notion of stoup

- We have the following stoup-like property:

  Let c be a command. The set:

  $$\left\{\alpha^+ \,\middle|\, \mu\alpha^+.c \text{ is thunkable}\right\} \cup \left\{x^\ominus \,\middle|\, \tilde{\mu}x^\ominus.c \text{ is linear}\right\}$$

  has at most one element.

- With this property we define a semantic notion of stoup that encompasses the stoup of Girard's **LC**, which itself is hard to manipulate formally. By *semantic* we mean that the notion is attached to the properties rather than to the form.

Characterisation of linearity and thunkability

Remarks

- We were not satisfied with the restrictions called *A-normal form / monadic normal form / η restriction*. This is because substitution has to be defined for all terms and contexts.
- The notion cannot be reduced to a notion of value as developed through continuation calculi. For instance any term which reduces to a value in a referentially transparent way is thunkable.

Perspectives

Connectives in terms of duploids **L** calculi only use simple β and η rules. This suggests that connectives should be characterised with a suitable notion of universal property over duploids. (Ongoing work with Fiore and Curien)

Orthogonality-based logical relations Proving syntactic properties of systems, inspired by normalisation-by-evaluation. A conceptual treatment of renaming is expected.

Proof theory of arithmetic It appears that many aspects of the works of Krivine could be simplified with an adequate treatment of polarities.

Perspectives

Polarised intuitionistic logic Liang and Miller uncovered an intuitionistic system where the two negations $\sim$ and $\cdot \to \bot$ coexist. It is similar to our classical natural deduction in which the classical principle $(A \to \bot) \to \sim A$ is removed.

Operational models of programming languages The **L** calculi do not account yet for linearity-related optimisations described in CPS translations.

Rewriting theory The rewriting theory of extensional syntaxes with constructors is a recent and active field of study.

Thank you