Formulae-as-Types for an Involutive Negation

Guillaume Munch-Maccagnoni

LIPN, Université Paris 13

Joint meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science

(CSL-LICS 2014)

July 18th 2014

Constructive interpretations of classical logic

Proposition (Joyal)

Any Cartesian closed category $\mathscr{C}$ with an object 0 satisfying a natural isomorphism $0^{0^A} \simeq A$ is a boolean algebra (= does not distinguish proofs).

Not easy to see which hypotheses of CCCs we should relinquish.

- $\neg\neg A$ retract of A and $\perp$ not initial
  Call-by-name λC calculus
- Symmetric monoidal instead of Cartesian
  Multiplicative Linear Logic
- Composition not always associative
  Evaluation order defined by polarities (Here)

Constructive interpretations of classical logic

Two steps

- Gödel-Gentzen $\neg\neg$-translation
  \+ Friedman-Dragalin's A-*translation*
  Π^0_2-*conservativity of Peano Arithmetic over Heyting Arithmetic*
- Gödel-Gentzen $\neg\neg$-translation
  \+ Gödel's Dialectica interpretation
  Interpretation of the axiom of dependent choice using bar recursion (Spector)
- Cut-elimination in Girard's variant *LC* of Gentzen's *LK*
  \+ analysis of cut-free proofs
  Sequent calculus satisfying $A = \neg\neg A$
- CPS translation + passing the identity continuation
  Translations for control operators (Griffin, Murthy) in a certain relationship with Gödel-Gentzen $\neg\neg$*-translations (Lafont-Reus-Streicher, Laurent)*

Constructive interpretations of classical logic

Example

Kuroda translation (1951) / Call-by-value CPS translation

$$
\begin{aligned}
X(t_1, \ldots, t_n)^* &\stackrel{\text{def}}{=} X(t_1, \ldots, t_n) \\
(P \vee Q)^* &\stackrel{\text{def}}{=} P^* \vee Q^* \\
(P \wedge Q)^* &\stackrel{\text{def}}{=} P^* \wedge Q^* \\
(\exists x\, P)^* &\stackrel{\text{def}}{=} \exists x\, P^* \\
(P \to Q)^* &\stackrel{\text{def}}{=} \neg(P^* \wedge \neg Q^*) \\
(\forall x\, P)^* &\stackrel{\text{def}}{=} \neg\exists x\, \neg P^*
\end{aligned}
$$

Proposition

- *If $P \vdash Q$ classically then $P^* \vdash \neg\neg Q^*$ intuitionistically*
- *If $P \vdash Q$ classically then $P \vdash Q$ intuitionistically when P and Q are* purely positive *(transform an intuitionistic derivation of $P \vdash \neg\neg Q$ into one of $P \vdash (Q \to Q) \to Q$).*

Constructive interpretations of classical logic

Direct interpretations

- Gentzen's sequent calculus refined by Girard and Danos, Joinet and Schellinx
  = ¬¬-*translation* + *A-translation*
- Formulae-as-types, λ calculi with control operators: Griffin ($\lambda\mathcal{C}$); Parigot ($\lambda\mu$); Curien and Herbelin ($\bar{\lambda}\mu\tilde{\mu}$)
  = ¬¬-*translation* + *A-translation*
- Krivine's classical realisability
  = ¬¬-*translation* + *A-translation* + *modified realisability* (+ *Cohen's Forcing*)

And others (Selinger, Coquand, Avigad, Aschieri-Berardi-de'Liguoro...)

Constructive interpretations of classical logic

The $\lambda\mathcal{C}$ calculus

- The control operator $\mathcal{C}$ can be typed with $\neg\neg P \to P$ (Griffin)
- The most convenient way of reducing terms is with abstract machines (Krivine, Curien-Herbelin)
  The call-by-name machine of Reus and Streicher:

$$
\begin{array}{ccc}
\langle t\, u \parallel \pi \rangle & \succ_n & \langle t \parallel u{\cdot}\pi \rangle \\
\langle \lambda x.t \parallel u{\cdot}\pi \rangle & \succ_n & \langle t[u/x] \parallel \pi \rangle \\
\langle \mathcal{C} \parallel t{\cdot}\pi \rangle & \succ_n & \langle t \parallel \mathsf{k}_\pi{\cdot}\mathsf{stop} \rangle \\
\langle \mathsf{k}_\pi \parallel t{\cdot}\pi' \rangle & \succ_n & \langle t \parallel \pi \rangle
\end{array}
$$

- Amounts to studying at once the translations and the target.

Expressive vs. fine-grained interpretations

- Expressiveness
  ex. *"Is it possible to realise the formula A?"*
  - Cut-elimination
  - Witness extraction
  - Consistency
- Understanding the fine details
  ex. *"Is there a behaviour common to all realisers of A?"*
  - In particular: type isomorphisms
    (thus: Equational theory with η laws)
  - Rewriting theory
  - Böhm theorem
  - Is there a canonical interpretation for classical logic?

Expressive vs. fine-grained interpretations

Does the $\lambda\mathcal{C}$ calculus give a fine-grained interpretation ?

- Example:

  $$(\neg\forall \mathrm{x} \in \mathbb{N}\ A) \to \exists \mathrm{y} \in \mathbb{N}\ \neg A$$

  has a proof with the following skeleton:

  $$\lambda xy.(\mathcal{C}\,\lambda k.(x\,\lambda e.(\mathcal{C}\,\lambda l.(k\,(y\,e\,l)))))$$

- Reasoning by contrapositive is non-trivial and counter-intuitive
  (Yet *e.g.* Krivine realises the axiom of dependent choice via its contrapositive)

Expressive vs. fine-grained interpretations

Does the $\lambda\mathcal{C}$ calculus give a fine-grained interpretation ?

Realising $(\neg\forall x \in \mathbb{N}\ A) \to \exists y \in \mathbb{N}\ \neg A$ should be as simple as:

1. Evaluating the argument until a stack of the form $n \cdot \pi$ appears where n is an integer
2. Return the pair (n, k_π) where k_π is the continuation of type $\neg A$

This is more or less what happens in the $\lambda\ell$ calculus

Expressive vs. fine-grained interpretations

The $\lambda\ell$ calculus

The $\lambda\ell$ calculus is both:

- A term syntax for classical natural deduction that satisfies:

$$A \simeq \neg\neg A \quad , \quad \neg\forall x(A \to B) \simeq \exists x(A \wedge \neg B) \ \dots$$

  (*i.e.* reasoning by contrapositive)
- A Curry-style λ calculus with a delimited control operator (ℓ) that implements the fact that *captured stacks, contrarily to continuations, can be inspected*

L calculi

1) Solving equations on abstract machines

- The λ calculus is universal in the sense that it represents combinators abstractly by their reduction rules. Ex:

$$\mathsf{S}xyz > xz(yz)$$

  1. One can prove $\mathsf{S} \simeq_{\beta\eta} \lambda xyz.xz(yz)$
  2. $\mathsf{S} = \lambda xyz.xz(yz)$ is a solution

- Similarly, *L calculi* (here Curien-Herbelin's $\bar{\lambda}\mu\tilde{\mu}_T$) are universal because they extend the above principle to abstract machines — the transitions rules on the left are *solved* on the right using the μ binder:

$$\begin{array}{rrl}
t\,u: & \pi \mapsto \langle t \parallel u\cdot\pi\rangle & t\,u \stackrel{\text{def}}{=} \mu\alpha.\langle t \parallel u\cdot\alpha\rangle \\
\mathsf{k}_\pi: & t\cdot\pi' \mapsto \langle t \parallel \pi\rangle & \mathsf{k}_e \stackrel{\text{def}}{=} \lambda x.\mu\alpha.\langle x \parallel e\rangle \\
C: & u\cdot\pi \mapsto \langle u \parallel \mathsf{k}_\pi\cdot\mathsf{stop}\rangle & C \stackrel{\text{def}}{=} \lambda x.\mu\alpha.\langle x \parallel \mathsf{k}_\alpha\cdot\mathsf{stop}\rangle
\end{array}$$

Caution: μ has nothing to do with least fixed-points.

L calculi

2) Correspondence with sequent calculus

$$
\dfrac{\Gamma \vdash t : A \to B \mid \Delta \qquad \dfrac{\Gamma' \vdash u : A \mid \Delta' \qquad \dfrac{}{\mid \alpha : B \vdash \alpha : B}\,(\mathrm{ax}\vdash)}{\Gamma' \mid u{\cdot}\alpha : A \to B \vdash \alpha : B, \Delta'}\,(\to\vdash)}{\dfrac{\langle t \parallel u{\cdot}\alpha\rangle : (\Gamma, \Gamma' \vdash \alpha : B, \Delta, \Delta')}{\Gamma, \Gamma' \vdash \underbrace{\mu\alpha.\langle t \parallel u{\cdot}\alpha\rangle}_{=t\,u} : B \mid \Delta, \Delta'}\,(\vdash\mu)}\,(\mathrm{cut})
$$

L calculi

3) Delimited control interprets ⊥: logic side

- Units (e.g. ⊥) are problematic when combining Curry-style + extensionality
  Restrict the β and η laws vs still have enough isomorphisms involving ⊥
- Dynamically-scoped variable $\widehat{\mathtt{tp}}$:

$$\frac{c : (\Gamma \vdash \Delta)}{\Gamma \vdash \mu\widehat{\mathtt{tp}}.c : \bot \mid \Delta}\ (\vdash \bot) \qquad \frac{}{\Gamma \mid \widehat{\mathtt{tp}} : \bot \vdash \Delta}\ (\bot \vdash)$$

We do have:

$$\mu\widehat{\mathtt{tp}}.\langle t \parallel \widehat{\mathtt{tp}}\rangle \simeq t$$
$$\langle \mu\widehat{\mathtt{tp}}.c \parallel \widehat{\mathtt{tp}}\rangle \simeq c$$

but no longer $\langle \mu\alpha.c \parallel \widehat{\mathtt{tp}}\rangle \rhd c[\widehat{\mathtt{tp}}/\alpha]$

L calculi

3) Delimited control interprets $\bot$: programming side

- Auxiliary stack of stacks:

$$\begin{array}{ll} \text{push:} & \langle \mu\widehat{\mathrm{tp}}.c \parallel \pi_1 \rangle \{\pi_2, \dots, \pi_n\} \rhd c\{\pi_1, \pi_2, \dots, \pi_n\} \\ \text{pop:} & \langle t \parallel \widehat{\mathrm{tp}} \rangle \{\pi_1, \pi_2, \dots, \pi_n\} \rhd \langle t \parallel \pi_1 \rangle \{\pi_2, \dots, \pi_n\} \end{array}$$

- The μ binder does not capture the auxiliary stack:

$$\langle \mu\alpha.c \parallel \pi \rangle \{\sigma\} \rhd c[\pi/\alpha]\{\sigma\}$$

"Delimited control"
(Felleisen, Danvy-Filinski — in logic:
Ariola-Herbelin-Sabry, Herbelin-Ghilezan)

Polarisation and focalisation

4) Polarisation

What Giving a formal status to the polarities of connectives

Why Reconcile β-reductions with η-expansions

- For negative connectives, η-expansion delays evaluation. E.g. for $\to$:

$$tu \qquad vs \qquad \lambda x.tux$$

- For positive connectives, η-expansion forces evaluation. E.g. for $\vee$:

$$u \qquad vs \qquad \mathsf{match}\ u\ \mathsf{with}\ (l(x).l(x) \mid r(x).r(x))$$

How Variables and terms have a polarity that determines the local reduction strategy

- Terms of a negative type like $\to$ are called by name
- Terms of a positive type like $\exists$ are called by value

(Girard, Danos-Joinet-Schellinx, also my 2009 paper at CSL)

Polarisation and focalisation

4) Polarisation

- Composition is not associative but reminiscent of Loday's duplicial algebras

$$(h \bullet g) \bullet f = h \bullet (g \bullet f)$$
$$(h \circ g) \circ f = h \circ (g \circ f)$$
$$(h \bullet g) \circ f = h \bullet (g \circ f)$$
$$(h \circ g) \bullet f \neq h \circ (g \bullet f) \text{ in general}$$

M.-M. Models of a non-associative composition. In A. Muscholl, editor, *FoSSaCS*, volume 8412 of *LNCS*, pages 397–412. Springer, 2014

- Introduced by Girard in order to give a meaning to $A = \neg\neg A$ in classical sequent calculus (the logic LC)
  In LC, negation is defined by duality and is therefore not given as a connective. Negation inverts the polarity.

Polarisation and focalisation

5) Captured contexts are not continuations

The main insight of *LC* is, to me, the idea that the introduction rules of negation, taken as a connective, hide cuts (*focalisation*)

$$\dfrac{\dfrac{\Gamma, N \overset{\pi}{\vdash} \Delta}{\Gamma \vdash \neg N, \Delta} \qquad \Gamma', \neg N \overset{\pi'}{\vdash} \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \quad \triangleright \quad \dfrac{\dfrac{\dfrac{\dfrac{}{N \vdash N}}{\vdash \neg N, N} \qquad \Gamma', \neg N \overset{\pi'}{\vdash} \Delta'}{\Gamma' \vdash N, \Delta'} \qquad \Gamma, N \overset{\pi}{\vdash} \Delta}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

$$\dfrac{\Gamma' \overset{\pi'}{\vdash} \neg P, \Delta' \qquad \dfrac{\Gamma \overset{\pi}{\vdash} P, \Delta}{\Gamma, \neg P \vdash \Delta}}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \quad \triangleright \quad \dfrac{\Gamma \overset{\pi}{\vdash} P, \Delta \qquad \dfrac{\Gamma' \overset{\pi'}{\vdash} \neg P, \Delta' \qquad \dfrac{\dfrac{}{P \vdash P}}{P, \neg P \vdash}}{\Gamma', P \vdash \Delta'}}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

Polarisation and focalisation

5) Captured contexts are not continuations

- We show that Girard's *LC* is related to the idea in programming of having high-level access to the components of the contexts captured by control operators
- The type of captured contexts is therefore different from the type of continuations $A \to \bot$. Continuations are functions, and the contents of functions cannot be accessed in an immediate way
- It is obvious in "real-world" programming languages such as C (getcontext) or Smalltalk (thisContext) that captured contexts are positive objects that can be inspected. Clements' thesis theorises having high-level access to the components of the contexts.

Polarisation and focalisation

5) Captured contexts are not continuations

One more motivation:

- Krivine simplifies reasoning in the λC calculus, by allowing certain *pseudo-types* in the left-hand side of implications.
- For technical reasons, an essential pseudo-type in Krivine's work is the set $\{\mathsf{k}_\pi \mid \pi \in X\}$. This also amounts to distinguishing a positive type of captured stacks from the type of continuations $X \to \bot$.
- The difference is, we will do so in a direct manner, making such types first class, in the sense that we define their meaning also when they are on the right-hand side of implications.

Polarisation and focalisation

Summary of the method

$\lambda\ell$ calculus Classical natural deduction	(quasi-proofs)
↓ *solving equations*	
$L_{\mathrm{pol},\widehat{\mathrm{tp}}}$ Sequent calculus	(confluence, equational theory)
↓ *CPS translation*	
λ calculus with surjective pairs Intuitionistic logic	(coherence, strong normalisation)

The $\lambda\ell$ calculus

- We introduce the positive type $\sim A$ of *inspectable stacks*, which is distinct from the negative type $A \to \bot$ of continuations
- We define negation in function of the polarity with:

$$\neg P \overset{\text{def}}{=} P \to \bot \qquad , \qquad \neg N \overset{\text{def}}{=} \sim N$$

  (defining negation in function of the polarity is reminiscent of Danos, Joinet and Schellinx)
- In the $\lambda\ell$ calculus we have the following isomorphisms:

$$\begin{array}{lcl} P & \cong & \sim(P \to \bot) \\ N & \cong & (\sim N) \to \bot \\ \sim\forall x(A \to B) & \cong & \exists x(A \land \sim B) \end{array}$$

The $\lambda\ell$ calculus

- The values that inhabit the type $\sim A$ are of the form $[\pi]$ where π is a context of the abstract machine
- We introduce combinators that let us access the contents of these inspectable stacks

$$D_{\to} : (\sim(A \to B)) \to (A \land \sim B)$$
$$D_{\forall} : (\sim\forall x\, N) \to \exists x \sim N$$
$$D_{\bot} : \bot \to A \to A$$

$$\langle D_{\to} \parallel [V\cdot\pi_1]\cdot\pi_2\rangle \succ \langle (V, [\pi_1]) \parallel \pi_2\rangle$$
$$\langle D_{\forall} \parallel [\pi_1]\cdot\pi_2\rangle \succ \langle [\pi_1] \parallel \pi_2\rangle$$
$$\langle D_{\bot} \parallel [\pi_{\ominus}]\cdot t\cdot\pi'\rangle \succ \langle t \parallel \pi'\rangle\{\pi_{\ominus}\}$$

The $\lambda\ell$ calculus

Example

We derive $D_{\forall\rightarrow} : (\sim\forall \mathrm{x}(A \rightarrow B)) \rightarrow \exists \mathrm{x}(A \wedge \sim B)$ as follows:

$$D_{\forall\rightarrow} \stackrel{\text{def}}{=} \lambda x^{+}.\mathsf{let}\ y^{+}\ \mathsf{be}\ D_{\forall}\ x^{+}\ \mathsf{in}\ D_{\rightarrow}\ y^{+}$$

$D_{\forall\rightarrow}$ reduces as follows:

$$\langle D_{\forall\rightarrow} \parallel [V\cdot\pi]\cdot\pi_{+}\rangle\{\sigma\} \succ^{*} \langle (V, [\pi]) \parallel \pi_{+}\rangle\{\sigma\}$$

i.e. in pattern-matching notation:

$$\begin{aligned} D_{\forall\rightarrow} &\simeq \lambda[x\cdot\alpha].(x, [\alpha]) \\ &\stackrel{\text{def}}{=} \lambda[\gamma].\mu\beta.\langle \lambda x.\mu\alpha.\langle (x, [\alpha]) \parallel \beta\rangle \parallel \gamma\rangle \end{aligned}$$

(compare to the term of the λC calculus $\lambda xy.(C\,\lambda k.(x\,\lambda e.(C\,\lambda l.(k\,(y\,e\,l))))))$

The $\lambda\ell$ calculus

A captured stack $[\pi]$ can be re-installed as the context of another term t by the constant send:

$$\langle \mathsf{send} \parallel [\pi] \cdot t \cdot \pi' \rangle \{\sigma\} \succ \langle t \parallel \pi \rangle \{\pi', \sigma\}$$

In other words, the constant send converts a captured stack into a continuation:

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

The $\lambda\ell$ calculus

The operator responsible for the apparition of inspectable stacks is ℓ:

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

This operation is formally described by introducing the j_π operator (analogous to the k_π of λC).
The operator ℓ saves with j the context π in which ℓ is applied:

$$\langle \ell \parallel t \cdot \pi \rangle \{\pi', \sigma\} \succ \langle t \parallel \mathsf{j}_\pi \cdot \pi' \rangle \{\sigma\}$$

Once the operator j_π comes in head position, it captures the stack and restores the context π:

$$\langle \mathsf{j}_\pi \parallel \pi' \rangle \{\sigma\} \succ \langle [\pi'] \parallel \pi \rangle \{\sigma\}$$

The $\lambda\ell$ calculus

Contributions in details

- Natural deduction, hence a language of untyped realisers (quasi-proofs), at the same time a delimited control calculus that implements high-level access to stacks
- An L calculus provides a confluent cut-elimination and an equational theory ($L_{\mathrm{pol},\hat{\mathfrak{tp}}}$ inspired by Curien and Herbelin's $\bar{\lambda}\mu\tilde{\mu}$)
- CPS translations for $\lambda\ell$ and $L_{\mathrm{pol},\hat{\mathfrak{tp}}}$ simulate reductions and preserve equivalences (hence strong normalisation of typed terms and coherence)
- Subsumes call-by-value and call-by-name $\lambda\mu$ calculus as well as De Groote-Saurin's $\Lambda\mu$ calculus and variants of the $\mathrm{shift}_0/\mathrm{reset}_0$ operators
- A direct computational interpretation of polarities being adapted for non-classical Call-by-Push-Value models

Thank you