Polarities and classical constructiveness

Guillaume Munch-Maccagnoni

LIPN, Université Paris 13

Semantics of proofs and certified mathematics
Institut Henri Poincaré thematic trimester
June 4th 2014
(slides as of June 5th)

Overview

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.

Proof.

- The co-Cartesian structure is obtained by the duality 0^-.
- One has $\mathscr{C}(0 \times 0, 0) \simeq \mathscr{C}(0, 0^0)$ and $0^0 \simeq I$ is terminal thus one has $\pi_1 = \pi_2 : 0 \times 0 \to 0$.
- Thus for any $f, g : A \to 0$ one has $f = g$ because the two projections of $\langle f, g \rangle : A \to 0 \times 0$ are equal.
- Thus $\mathscr{C}(A \times 0^B, 0)$ contains at most one morphism, yet we have: $\mathscr{C}(A, B) \simeq \mathscr{C}(A, 0^{0^B}) \simeq \mathscr{C}(A \times 0^B, 0)$ ∎

Moral: Not easy to see which hypotheses we should relinquish.

Overview

Indirect interpretations

- Gödel-Gentzen ¬¬-translation + Friedman-Dragalin's A-*translation*
  Π^0_2*-conservativity of Peano Arithmetic over Heyting Arithmetic*
- Gödel-Gentzen ¬¬-translation + Gödel's Dialectica interpretation
  Interpretation of the axiom of dependent choice using bar recursion
- Translations into Girard's linear logic
  Denotational semantics satisfying $A = \neg\neg A$

Overview

Example

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

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 $P \vdash (Q \to Q) \to Q$).*

Overview

Direct interpretations

- Gentzen's sequent calculus *(consistency of arithmetic)*, notably refined by Girard and Danos, Joinet and Schellinx
- Games (Gentzen; Novikoff; Coquand)
- Formulae-as-types, λ calculi with control operators: Griffin ($\lambda\mathcal{C}$); Parigot ($\lambda\mu$); Curien and Herbelin ($\bar{\lambda}\mu\tilde{\mu}$)
- Categorical interpretations of double-negation translations: Selinger; Hofmann and Streicher.
- Avigad's classical realisability
- Krivine's classical realisability
- Aschieri, Berardi and de' Liguoro's interactive realisability

and more

Direct interpretations

The $\lambda\mathcal{C}$ calculus

- Griffin showed that the control operator $\mathcal{C}$ has type $\neg\neg P \to P$
- The most convenient way of reducing terms is with abstract machines
- The call-by-name machine of Reus and Streicher:

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

Direct interpretations

The meaning of Π^0_2-conservativity

- In the presence of side effects such as control, programs of certain types (functions, etc.) are opaque at runtime.
- But programs of type $P \vdash Q$ are still algorithms when P and Q are purely positive.

In other words, we do not assume that the behaviour of proofs has to be referentially transparent. Thus a proof of $A \vee \neg A$ needs not provide a decision procedure for A.

Direct interpretations

Lots of relationships

- Continuation-passing-style (CPS) translations that implement control operators are ¬¬-translations *(Murthy)* in a certain relationship with Gödel-Gentzen ¬¬-translations *(Lafont, Reus and Streicher and Laurent)*
- Girard's classical sequent calculus = refined ¬¬-translation + A-translation *(Murthy)*
- Avigad's classical realisability = ¬¬-translation + A-translation + modified realisability *(Avigad)*
- Interactive realisability = A-translation + modified realisability *(Aschieri and Berardi)*
- Krivine's classical realisability = ¬¬-translation + A-translation + modified realisability (+ Cohen's Forcing) *(Oliva and Streicher)*
- (2014) Formulae-as-types for Gödel's Dialectica interpretation *(Pédrot)*

Direct interpretations

Lots of relationships

- Coherent picture emerges
- Understanding the translations is at least as important as understanding the intuitionistic target
- Direct interpretations amount to studying both translations and their target at the same time

Expressive vs. fine-grained interpretations

- Expressiveness
  ex. *"Is it possible to realise the formula A?"*
  - Cut-elimination
  - Witness extraction
  - Consistency

  (easier)

- 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

Here: Classical natural deduction that satisfies:

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

(*i.e.* reasoning by contrapositive)
with a clear constructive (*i.e.* programming) content:
the $\lambda\ell$ calculus, where ℓ is a control operator that we introduce

Guillaume Munch-Maccagnoni. Formulae-as-types for an involutive negation. In *Proceedings of the 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. To appear

Expressive vs. fine-grained interpretations

The $\lambda\mathcal{C}$ calculus is not fine-grained enough

- Cartesian closed (i.e. call-by-name λ calculus)
- A is a retract of $\neg\neg A$
- 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

The $\lambda\mathcal{C}$ calculus is not fine-grained enough

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
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

Formulae-as-types for an involutive negation

Polarisation

- Give a formal status to the polarities of connectives
  Goal: reconcile β-reductions with η-expansions
- For negative connectives, η-expansion delays evaluation. E.g. for $\to$:

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

  Consequently, terms of a negative type are called by name
- For positive connectives, η-expansion forces evaluation. E.g. for $\vee$:

  $$E[u] \qquad vs \qquad \text{match}\ u\ \text{with}\ (l(x).E[l(x)] \mid r(x).E[r(x)])$$

  Consequently, terms of a positive type are called by value

Formulae-as-types for an involutive negation

Polarisation

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

Formulae-as-types for an involutive negation

Polarisation

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

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

Formulae-as-types for an involutive negation

Captured contexts are not continuations

- We show that Girard's logic 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. 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 that captured contexts are more primitive than continuations

Formulae-as-types for an involutive negation

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.

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:

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

(defining negation in function of the polarity is reminiscent of Danos, Joinet and Schellinx)

- In the $\lambda\ell$ calculus we have the following isomorphisms:

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

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

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

The $\lambda\ell$ calculus

Example

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

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

In pattern-matching notation, $D_{\forall\rightarrow}$ is the function:

$$\lambda[x \cdot \alpha].(x, [\alpha])$$

(compare to the term of the λC calculus)

The $\lambda\ell$ calculus

A captured stack $[\pi]$ can be re-installed as the context of another term t by the constant send[1]:

$$\langle \mathsf{send} \parallel [\pi]\cdot t\cdot \pi' \rangle \succ_p \langle t \parallel \pi \rangle$$

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

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

[1]For didactic reasons, the present versions of send and ℓ (next slide) are undelimited variants of the operators from the article.

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:

$$\langle \ell \parallel t \cdot \pi \rangle \succ_p \langle t \parallel \mathsf{j}_\pi \cdot \mathsf{stop} \rangle$$

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

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

The $\lambda\ell$ calculus

Contributions in details

- A language of untyped realisers (quasi-proofs)
- The issue of $\bot$ in an untyped setting is solved with control delimiters (inspired by Ariola, Herbelin and Sabry; Herbelin and Ghilezan)
- $\lambda\ell$ is provided with an equational theory by embedding into a sequent calculus whose cut-elimination is confluent ($\mathsf{L}_{\mathrm{pol},\hat{\mathrm{tp}}}$ inspired by Curien and Herbelin's $\bar{\lambda}\mu\tilde{\mu}$)
- Double-negation translations for $\lambda\ell$ and $\mathsf{L}_{\mathrm{pol},\hat{\mathrm{tp}}}$ simulate reductions and preserve equivalences (hence strong normalisation of typed terms and coherence)
- A direct computational interpretation of polarities, which can be adapted for non-classical Call-by-Push-Value models
- Contains De Groote-Saurin's $\Lambda\mu$ and variants of the the $\mathrm{shift}_0/\mathrm{reset}_0$ operators

The $\lambda\ell$ calculus

Contributions in details

The catch is: we give up associativity of composition when the middle map is from positive to negative (*duploids*, see the second part)

Thank you