Quantitative (polarised) classical realizability

OPLSS 2011

Aloïs Brunel

LIPN - Université Paris 13

Classical realizability has been introduced by J-L. Krivine to :

- Investigate extensions of the CH correspondence to *ZF* + axioms
- Build **strange** models of *ZF* set theory
- Study the computational meaning of Cohen's forcing

Classical realizability has been introduced by J-L. Krivine to :

- Investigate extensions of the CH correspondence to *ZF* + axioms
- Build **strange** models of *ZF* set theory
- Study the computational meaning of Cohen's forcing

But it can be used to :

- Prove computational properties of typed programs

Classical realizability has been introduced by J-L. Krivine to :

- Investigate extensions of the CH correspondence to *ZF* + axioms
- Build **strange** models of *ZF* set theory
- Study the computational meaning of Cohen's forcing

But it can be used to :

- Prove **quantitative** computational properties of typed programs

  e.g : soundness of light linear logic w.r.t **PTIME**

Classical realizability has been introduced by J-L. Krivine to :

- Investigate extensions of the CH correspondence to *ZF* + axioms
- Build **strange** models of *ZF* set theory
- Study the computational meaning of Cohen's forcing

But it can be used to :

- Prove **quantitative** computational properties of typed programs

  e.g : soundness of light linear logic w.r.t **PTIME**

In fact, here we use polarized classical realizability (**G. Munch**).

Interaction and orthogonality

A usual notion in mathematics : **orthogonality**.

In computer science

Define the behavior of a "program" by observing its **interaction** with "environments".

We say that $t \perp e$ if $\langle t|e \rangle$ is a **good** process.

e.g : termination, non-termination, etc.

If X is a set of programs, then its **orthogonal** is :

$$X^{\perp} = \{\ e \mid \forall t \in X, t \perp e\ \}$$

$X^{\perp\perp}$ is the set of programs that behave like those in X.

Main idea : types A are **behaviors** (i.e : sets X s.t $X = X^{\perp\perp}$).

$\longrightarrow$ Membership tested **interactively**.

Examples

- Interaction between programs : Krivine's **classical realizability**, Girard's **ludics**
- Interaction between cliques in a graph : **coherence spaces**
- Interaction between operators in a Von Neumann algebra : **Geometry of Interaction**

Proving computational properties :

- $t \perp e \equiv \langle t|e\rangle$ terminates
- $|A \Rightarrow B| = \{\ u.e \mid u \in |A| \wedge e \in |B|^{\perp}\ \}^{\perp}$

Proving computational properties :

- $t \perp e \equiv \langle t|e\rangle$ terminates
- $|A \Rightarrow B| = \{\ u.e \mid u \in |A| \wedge e \in |B|^{\perp}\ \}^{\perp}$

Adequacy lemma

If $\vdash t : A$ then $t \in |A|$.

Proving quantitative computational properties :

- $(t, p)\perp(e, q) \equiv \langle t|e\rangle$ terminates in less than $p + q$ steps
- $|A \Rightarrow B| = \{ (u.e, q + r) \mid (u, q) \in |A| \wedge (e, r) \in |B|^{\perp} \}^{\perp}$

Adequacy lemma

If $\vdash t : A$ then there exists some p such that $(t, p) \in |A|$.

Proving quantitative computational properties :

- $(t, p) \perp (e, q) \equiv \langle t | e \rangle$ terminates in less than $p + q$ steps
- $|A \Rightarrow B| = \{ (u.e, q + r) \mid (u, q) \in |A| \wedge (e, r) \in |B|^{\perp} \}^{\perp}$

Adequacy lemma

If $\vdash t : A$ then there exists some p such that $(t, p) \in |A|$.

Usually $\mathbb{N}$ is not sufficient, we use a **resource monoid**

(Dal Lago & Hofmann)

We have :

- A biorthogonality-based framework to prove quantitative properties
- Soundness proofs of various type systems w.r.t complexity classes
- A uniform treatment of CBN and CBV
- A formal relation with forcing (in the style of **[Miquel, LICS 2011]**)

Compiler correctness

In progress work with M. Gaboardi, G. Jaber and N. Tabareau.

Based on :

- Compiler correctness using classical realizability

  [Jaber & Tabareau, LOLA 2010]

- Linear dependent types for capturing complexity

  [Dal Lago & Gaboardi, LICS 2011]

- Quantitative realizability

$\longrightarrow$ prove complexity preservation property for a compiler.

Conclusion

Things to do :

- Extend to side-effects (see **[Madet & Amadio, TLCA 2011]**)
- Apply to quantitative properties different from resource consumption

I'm also interested in other topics like :

- Dependent types and proof-nets
- Formalizing (some) complex analysis in Coq