Kleene realizability and negative translations

Alexandre Miquel

UNIVERSIDAD DE LA REPUBLICA URUGUAY
FACULTAD DE INGENIERIA
EQUIPO DE LÓGICA UDELAR

July 19th, 20th, 2016 – Piriápolis

Plan

Plan

1. Kleene realizability
2. Gödel-Gentzen negative translation
3. Lafont-Reus-Streicher negative translation

The language of realizers (recall)

Terms of PCF (= λ-calculus + primitive pairs & integers)

$$\textbf{Terms}\quad t,u \;::=\; x \mid \lambda x\,.\,t \mid tu \mid \texttt{pair} \mid \texttt{fst} \mid \texttt{snd} \mid 0 \mid \texttt{S} \mid \texttt{rec}$$

Syntactic worship: Free & bound variables. Renaming. Work up to α-conversion. Set of free variables: $FV(t)$. Capture-avoiding substitution: $t\{x := u\}$

- **Notations:** $\langle t_1, t_2\rangle \;:=\; \texttt{pair}\;t_1\;t_2,\qquad \bar{n} \;:=\; \texttt{S}^n\,0\quad (n \in \mathbb{N})$

Reduction rules

$$(\lambda x\,.\,t)\,u \;\succ\; t\{x := u\}$$

$$\begin{array}{llllll} \texttt{fst}\;\langle t_1, t_2\rangle & \succ & t_1 & \texttt{rec}\;t_0\;t_1\;0 & \succ & t_0 \\ \texttt{snd}\;\langle t_1, t_2\rangle & \succ & t_2 & \texttt{rec}\;t_0\;t_1\;(\texttt{S}\,u) & \succ & t_1\;u\;(\texttt{rec}\;t_0\;t_1\;u) \end{array}$$

- Grand reduction written $t \succ^* u$ (reflexive, transitive, context-closed)

Definition of the relation $t \Vdash A$ (recall)

- **Recall:** For each closed FO-term e, we write $e^{\mathbb{N}}$ its denotation in $\mathbb{N}$

Definition of the realizability relation $t \Vdash A$ (t, A closed)

$$
\begin{array}{lcl}
t \Vdash e_1 = e_2 & \equiv & e_1^{\mathbb{N}} = e_2^{\mathbb{N}} \;\wedge\; t \succ^* 0 \\
t \Vdash \bot & \equiv & \bot \\
t \Vdash \top & \equiv & t \succ^* 0 \\
t \Vdash A \Rightarrow B & \equiv & \forall u\,(u \Vdash A \;\Rightarrow\; tu \Vdash B) \\
t \Vdash A \wedge B & \equiv & \exists t_1\,\exists t_2\,(t \succ^* \langle t_1, t_2\rangle \;\wedge\; t_1 \Vdash A \;\wedge\; t_2 \Vdash B) \\
t \Vdash A \vee B & \equiv & \exists u\,((t \succ^* \langle \bar{0}, u\rangle \;\wedge\; u \Vdash A) \;\vee\; (t \succ^* \langle \bar{1}, u\rangle \;\wedge\; u \Vdash B)) \\
t \Vdash \forall x\, A(x) & \equiv & \forall n\,(t\,\bar{n} \Vdash A(n)) \\
t \Vdash \exists x\, A(x) & \equiv & \exists n\,\exists u\,(t \succ^* \langle \bar{n}, u\rangle \;\wedge\; u \Vdash A(n))
\end{array}
$$

Lemma (closure under anti-evaluation)

If $t \succ^* t'$ and $t' \Vdash A$, then $t \Vdash A$

The main Theorem (recall)

Lemma (Adequacy)

Let $d : (A_1, \dots, A_n \vdash B)$ be a derivation in NJ. Then:

- for all valuations ρ,
- for all realizers $t_1 \Vdash A_1[\rho], \dots, t_n \Vdash A_n[\rho]$,

we have: $d^*[\rho]\{z_1 := t_1, \dots, z_n := t_n\} \Vdash B[\rho]$

Lemma

All axioms of HA are realized

Theorem (Soundness)

If $HA \vdash A$, then $t \Vdash A$ for some closed PCF-term t

Harrop formulas (1/2)

The class of Harrop formulas

Harrop formulas $H \quad ::= \quad e_1 = e_2 \quad | \quad \top \quad | \quad \bot \quad | \quad H_1 \wedge H_2 \quad | \quad A \Rightarrow H \quad | \quad \forall x\, H$

- **Intuition:** Harrop formulas do not contain the two "problematic" constructions $\vee$ and $\exists$, except on the left-hand side of implications
- Therefore, Harrop formulas are classical:

Proposition

For each Harrop formula $H(\vec{x})$:

$$\mathrm{HA} \quad \vdash \quad \forall \vec{x}\, (H(\vec{x}) \;\Leftrightarrow\; \neg\neg H(\vec{x}))$$

Proof. By structural induction on $H(\vec{x})$.

Harrop formulas (2/2)

- To each (possibly open) Harrop formula H, we associate a closed PCF-term t_H that is computationally trivial:

$$\begin{array}{rclcrcl} \tau_H & := & 0 \quad (H \text{ atomic}) & & \tau_{A\Rightarrow H} & = & \lambda\_\,.\,\tau_H \\ \tau_{H_1\land H_2} & = & \langle \tau_{H_1}, \tau_{H_2}\rangle & & \tau_{\forall x\,H} & = & \lambda\_\,.\,\tau_H \end{array}$$

Theorem

For all closed Harrop formulas H:

$$\text{If} \quad H \text{ is realized,} \quad \text{then} \quad \tau_H \Vdash H$$

Moreover, all realizers of H are "computationally equivalent" to τ_H

- **Intuition:** Harrop formulas have computationally irrelevant realizers, that can be replaced by the trivial realizers τ_H
  - Useful for optimizing extracted programs (e.g. Fermat's last theorem)
  - But shows that Harrop formulas are computationally irrelevant

Plan

1. Kleene realizability
2. Gödel-Gentzen negative translation
3. Lafont-Reus-Streicher negative translation

How to cope with classical logic?

- Kleene realizability is definitely incompatible with classical logic:

$$\begin{array}{rcl} & \not\Vdash & \forall x\,(\mathsf{Halt}(x) \vee \neg \mathsf{Halt}(x)) \\ \mathsf{any\_term} & \Vdash & \neg\forall x\,(\mathsf{Halt}(x) \vee \neg \mathsf{Halt}(x)) \end{array}$$

(The same holds for all variants of Kleene realizability)

- Two possible solutions:
  1. Compose Kleene realizability with a negative translation from classical logic (LK) to intuitionistic logic (LJ) (next slide)
  2. Reformulate the principles of realizability to make them compatible with classical logic: Krivine classical realizability (next talk)

The Gödel-Gentzen negative translation

- **Idea:** Turn positive constructions (atomic formulas, $\vee$, $\exists$) into negative constructions ($\bot$, $\neg$, $\Rightarrow$, $\wedge$, $\forall$) using De Morgan laws
- Every formula A is translated into a formula A^G defined by:

$$
\begin{array}{rcl@{\qquad}rcl}
\top^G & := & \top & \bot^G & := & \bot \\
(A \Rightarrow B)^G & := & A^G \Rightarrow B^G & (e_1 = e_2)^G & := & \neg\neg(e_1 = e_2) \\
(A \wedge B)^G & := & A^G \wedge B^G & (A \vee B)^G & := & \neg(\neg A^G \wedge \neg B^G) \\
(\forall x\, A)^G & := & \forall x\, A^G & (\exists x\, A)^G & := & \neg \forall x\, \neg A^G
\end{array}
$$

writing: $\neg A \;:=\; A \Rightarrow \bot$

Theorem (Soundness)

1. $\mathrm{LK} \vdash A^G \Leftrightarrow A$
2. If $\mathrm{PA} \vdash A$, then $\mathrm{HA} \vdash A^G$

Realizing translated formulas

- **Strategy:**
  1. Build a derivation d of A (in PA)
  2. Turn it into a derivation d^G of A^G (in HA)
  3. Turn d^G into a Kleene realizer (program extraction)
- Does not work! Failure comes from:

Proposition (Realizability collapse)

For every closed formula A:

1. A^G is a Harrop formula (computationally irrelevant)
2. Kleene's semantics for A^G mimics Tarski's semantics for A:

$$A^G \text{ is realized} \quad \text{iff} \quad \tau_{A^G} \Vdash A^G \quad \text{iff} \quad \mathbb{N} \models A$$

Proof. By structural induction on A.

- **Conclusion:** Kleene $\circ$ Gödel-Gentzen $=$ Tarski

Friedman's R-translation (called A-translation by Friedman)

- **Principle:** In Gödel-Gentzen translation, replace each occurrence of $\bot$ (absurdity) by a fixed formula R, called the return formula
- Every formula A is translated into a formula A^F defined by:

$$
\begin{array}{rcl@{\qquad}rcl}
\top^F & := & \top & \bot^F & := & R \\
(A \Rightarrow B)^F & := & A^F \Rightarrow B^F & (e_1 = e_2)^F & := & \neg_R \neg_R (e_1 = e_2) \\
(A \wedge B)^F & := & A^F \wedge B^F & (A \vee B)^F & := & \neg_R(\neg_R A^F \wedge \neg_R B^F) \\
(\forall x\, A)^F & := & \forall x\, A^F & (\exists x\, A)^F & := & \neg_R \forall x\, \neg_R A^F
\end{array}
$$

writing: $\neg_R A \;:=\; A \Rightarrow R$

Theorem (Soundness)

If $\mathrm{PA} \vdash A$, then $\mathrm{HA} \vdash A^F$ (independently from the formula R)

Beware! The formulas A and A^F are no more classically equivalent (in general)

Friedman's trick

Theorem (Kreisel-Friedman)

PA conservatively extends HA over Π^0_2-formulas:

If PA $\vdash \forall x \, \exists y \, f(x,y) = 0$, then HA $\vdash \forall x \, \exists y \, f(x,y) = 0$

Proof. Assume that PA $\vdash \forall x \, \exists y \, f(x,y) = 0$. We have:

$$\begin{array}{ll} \text{HA} \vdash \forall x \neg_R \forall y \neg_R \neg_R \neg_R f(x,y) = 0 & \text{(by } R\text{-translation)} \\ \text{HA} \vdash \forall x \neg_R \forall y \neg_R f(x,y) = 0 & \text{(since } \neg_R\neg_R\neg_R \Leftrightarrow \neg_R) \\ \text{HA} \vdash \neg_R \forall y \neg_R f(x_0,y) = 0 & (\forall\text{-elim, } x_0 \text{ fresh}) \\ \text{HA} \vdash \forall y \, (f(x_0,y) = 0 \Rightarrow R) \Rightarrow R & \text{(def. of } \neg_R) \end{array}$$

We now let: $R := \exists y_0 \, f(x_0,y_0) = 0$ (Friedman's trick!) From the def. of R:

$$\text{HA} \vdash \forall y \, (f(x_0,y) = 0 \Rightarrow \exists y_0 \, f(x_0,y_0) = 0) \Rightarrow \exists y_0 \, f(x_0,y_0) = 0$$

But the premise of the above implication is provable

$$\text{HA} \vdash \forall y \, (f(x_0,y) = 0 \Rightarrow \exists y_0 \, f(x_0,y_0) = 0) \qquad (\exists\text{-intro})$$

hence we get

$$\begin{array}{ll} \text{HA} \vdash \exists y_0 \, f(x_0,y_0) = 0 & \text{(modus ponens)} \\ \text{HA} \vdash \forall x_0 \, \exists y_0 \, f(x_0,y_0) = 0 & (\forall\text{-intro}) \end{array}$$

□

Realizing translated formulas, again

- **Strategy:**
  1. Build a derivation d of a Π^0_2-formula A (in PA)
  2. Turn it into a derivation F-trick(d^F) of A (in HA)
  3. Turn F-trick(d^F) into a Kleene realizer of A (program extraction)

- This technique perfectly works in practice. However:
  - The formula A^F is not a Harrop formula (in general), even when A is. **Possible fix:** Introduce specific optimization techniques, e.g.:

    Refined Program Extraction [Berger et al. 2001]

  - The translation $A \mapsto A^F$ completely changes the structure of the underlying proof. **Possible fix:** cf next part

Plan

1. Kleene realizability
2. Gödel-Gentzen negative translation
3. Lafont-Reus-Streicher negative translation

The Lafont-Reus-Streicher negative translation

- **Idea:** Translate each formula A into the (relative) negation of a formula $A^\perp$ already representing the negation of A:

$$A^{LRS} \;:=\; \neg_R A^\perp \;\equiv\; A^\perp \Rightarrow R \qquad (A^\perp \text{ defined by induction on } A)$$

  (Again, this translation is parameterized by a return formula R)

- To every predicate symbol p (source language) we associate a predicate symbol $\bar{p}$ representing its negation (target language)

- Definition of the translations $A \mapsto A^\perp$ and $A \mapsto A^{LRS}$:

$$\begin{array}{rcl@{\qquad}rcl} (p(e_1,\ldots,e_k))^\perp & := & \bar{p}(e_1,\ldots,e_k) & \bot^\perp & := & \top \\ (A \Rightarrow B)^\perp & := & A^{LRS} \wedge B^\perp & (\forall x\, A)^\perp & := & \exists x\, A^\perp \end{array}$$

$$A^{LRS} \;:=\; \neg_R A^\perp \;\equiv\; A^\perp \Rightarrow R$$

Theorem (Soundness)

1. When $R \equiv \bot$: LK $\vdash A^\perp \Leftrightarrow \neg A$ and LK $\vdash A^{LRS} \Leftrightarrow A$
2. If LK $\vdash A$, then LJ $\vdash A^{LRS}$ (independently from the formula R)

Computational interpretation

- **Intuition:** The translated formula $A^\perp$ represents the type of stacks opposing (classical) terms of type A:

$$(A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow B)^\perp \quad \equiv \quad A_1^{LRS} \wedge \cdots \wedge A_n^{LRS} \wedge B^\perp$$

$$(A_1 \to \cdots \to A_n \to B)^\perp \quad \equiv \quad A_1^{LRS} \times \cdots \times A_n^{LRS} \times B^\perp$$

- To analyze the computational contents of the LRS-translation, we need to work across to λ-calculi:

  - A source calculus to represent classical proofs:

    $$\lambda_{\text{source}} \;=\; \lambda_\to + \text{œ} : ((A \to B) \to A) \to A \qquad \text{(Peirce's law)}$$

    (Polymorphic constant œ introduces classical reasoning)

  - An intuitionistic target calculus to represent translated proofs:

    $$\lambda_{\text{target}} \;=\; \lambda_{\to,\times}$$

    (In this calculus, pairs are used to represent stacks)

The source λ-calculus ($\{\bot, \Rightarrow, \forall\}$-fragment of LK)

Syntax

Types	A, B	$::=$	$\bot \mid p(e_1, \ldots, e_k) \mid A \Rightarrow B \mid \forall x\, A$
Proof-terms	t, u	$::=$	$z \mid \lambda z\,.\,t \mid tu \mid \text{cc}$

- Classical logic obtained by introducing an inert constant cc (call/cc) for Peirce's law (taken as an axiom)
- Constructions $\top$, $\wedge$, $\vee$, $\exists$ encoded using De Morgan laws (= full LK)

Typing rules

$$\frac{}{\Gamma \vdash z : A}\ {\scriptstyle (z:A)\in\Gamma} \qquad \frac{}{\Gamma \vdash \text{cc} : ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$$

$$\frac{\Gamma, z : A \vdash t : B}{\Gamma \vdash \lambda z\,.\,t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash tu : B}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall x\, A}\ {\scriptstyle x \notin FV(\Gamma)} \qquad \frac{\Gamma \vdash t : \forall x\, A}{\Gamma \vdash t : A\{x := e\}} \qquad \frac{\Gamma \vdash t : \bot}{\Gamma \vdash t : A}$$

Note: $\forall$ is treated uniformly: $\forall x\, A(x) \approx \bigcap_x A(x)$ (no function argument!)

The target λ-calculus ($\{\top, \Rightarrow, \wedge, \exists\}$-fragment of LJ)

Syntax

Types $A, B ::= \top \mid \bar{p}(e_1, \ldots, e_k) \mid A \Rightarrow B \mid A \wedge B \mid \exists x\, A$

Proof-terms $t, u ::= z \mid \lambda z\,.\,t \mid tu \mid \langle t, u\rangle \mid \mathtt{fst}(t) \mid \mathtt{snd}(t)$

$+$ usual reduction rules for proof-terms

Typing rules

$$\frac{}{\Gamma \vdash z : A}\ {\scriptstyle (z:A)\in\Gamma} \qquad \frac{}{\Gamma \vdash t : \top}\ {\scriptstyle FV(t)\subseteq \mathrm{dom}(\Gamma)}$$

$$\frac{\Gamma, z : A \vdash t : B}{\Gamma \vdash \lambda z\,.\,t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash tu : B}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash t : B}{\Gamma \vdash \langle t, u\rangle : A \wedge B} \qquad \frac{\Gamma \vdash t : A \wedge B}{\Gamma \vdash \mathtt{fst}(t) : A} \qquad \frac{\Gamma \vdash t : A \wedge B}{\Gamma \vdash \mathtt{snd}(t) : B}$$

$$\frac{\Gamma \vdash t : A\{x := e\}}{\Gamma \vdash t : \exists x\, A} \qquad \frac{\Gamma \vdash t : (\exists x\, A) \Rightarrow B}{\Gamma \vdash t : \forall x\,(A \Rightarrow B)}\ {\scriptstyle x\notin FV(B)}$$

Note: $\exists$ treated uniformly: $\exists x\, A(x) \approx \bigcup_x A(x)$ (no witness!)

Remark: Uniform vs non-uniform quantifiers (1/2)

- In the Curry-Howard correspondence (and in realizability), there are two different ways to interpret quantifiers:

Quantifier	**Uniform** (ML/Haskell style)	**Non-uniform** (Type Theory style)
$\forall x\, A(x)$	$\bigcap_{x \in D} A(x)$ (intersection type)	$\prod_{x \in D} A(x)$ (type of dependent functions)
$\exists x\, A(x)$	$\bigcup_{x \in D} A(x)$ (union type)	$\sum_{x \in D} A(x)$ (type of dependent pairs)

- **Remark:** Tarski/Kripke/Heyting/Cohen models do not distinguish the two interpretations: difference only appears in realizability

Remark: Uniform vs non-uniform quantifiers (2/2)

- 1st-, 2nd- and higher-order logic support both interpretations
  (But uniform interpretation is more concise & natural)
- The same holds for impredicative set theories: ZF, IZF_C, IZF_R
- Arithmetic (PA/HA) only supports the non-uniform interpretation
  (due to the induction principle)
- But in all cases, the non-uniform interpretation can be encoded from the uniform interpretation, using a relativization:

$$\text{(non-uniform) } \forall x\, A(x) \quad := \quad \text{(uniform) } \forall x\, (\underbrace{x \in D \Rightarrow A(x)}_{\text{type of functions}})$$

$$\text{(non-uniform) } \exists x\, A(x) \quad := \quad \text{(uniform) } \exists x\, (\underbrace{x \in D \wedge A(x)}_{\text{type of pairs}})$$

- This is why we shall prefer the uniform interpretation (in what follows)

The Lafont-Reus-Streicher logical translation

- The logical translation $A \mapsto A^{LRS}$

$$
\begin{array}{rcl}
(p(e_1,\ldots,e_k))^\perp & := & \bar{p}(e_1,\ldots,e_k) \\
(A \Rightarrow B)^\perp & := & A^{LRS} \wedge B^\perp
\end{array}
\qquad
\begin{array}{rcl}
\perp^\perp & := & \top \\
(\forall x\, A)^\perp & := & \exists x\, A^\perp
\end{array}
$$

$$A^{LRS} \quad := \quad \neg_R A^\perp$$

corresponds to a program transformation on untyped proof terms, called a continuation-passing style (CPS) translation:

$$
\begin{array}{rcl}
(z)^{LRS} & := & \lambda s\,.\,z\,s \\
(\lambda z\,.\,t)^{LRS} & := & \lambda\langle z,s\rangle\,.\,t^{LRS} \\
(tu)^{LRS} & := & \lambda s\,.\,t^{LRS}\,\langle u^{LRS},s\rangle
\end{array}
\qquad
\begin{array}{rcl}
(\mathbf{c})^{LRS} & := & \lambda\langle z,s\rangle\,.\,z\,\langle \mathrm{k}_s,s\rangle \\
\text{where}\ \ \mathrm{k}_s & := & \lambda\langle z,\_\rangle\,.\,z\,s
\end{array}
$$

Note: $\lambda\langle z,s\rangle\,.\,t$ defined as $\lambda z_0\,.\,(\lambda zs\,.\,t)\,(\mathtt{fst}(z_0))\,(\mathtt{snd}(z_0))$

Theorem (Soundness)

If $\Gamma \vdash t : A$ (in the source λ-calculus)

then $\Gamma^{LRS} \vdash t^{LRS} : A^{LRS}$ (in the target λ-calculus)

Towards the Krivine abstract machine

- From the computational behavior of translated proof terms t^{LRS}...

$$
\begin{array}{rcr}
(\lambda z\,.\,t)^{LRS} \;@\; \langle u, s\rangle & \succ & t^{LRS}\{z := u\} \;@\; s \\
(tu)^{LRS} \;@\; s & \succ & t^{LRS} \;@\; \langle u^{LRS}, s\rangle \\
(\text{æ})^{LRS} \;@\; \langle u, s\rangle & \succ & u \;@\; \langle \mathsf{k}_s, s\rangle \\
\mathsf{k}_s \;@\; \langle u, s'\rangle & \succ & u \;@\; s
\end{array}
$$

... we deduce evaluation rules for classical proof terms:

Krivine Abstract Machine (KAM)

Grab	$\lambda z\,.\,t \star u \cdot \pi$	$\succ$	$t\{z := u\} \star \pi$
Push	$tu \star \pi$	$\succ$	$t \star u \cdot \pi$
Save	$\text{æ} \star u \cdot \pi$	$\succ$	$u \star \mathsf{k}_\pi \cdot \pi$
Restore	$\mathsf{k}_\pi \star u \cdot \pi'$	$\succ$	$u \star \pi$

- Reformulating Kleene realizability through the LRS translation (and its CPS), we get Krivine classical realizability (cf next talk)