Computational interpretation of proofs: Classical realizability

Alexandre Miquel

Semantics of proofs and certified mathematics
PhD school – April 10th, 2014 – CIRM – Luminy

What is classical realizability?

- Complete reformulation of the principles of Kleene realizability to take into account classical reasoning
  - Based on Griffin's discovery about the connection between classical reasoning and control operators (call/cc)

$$\text{call/cc} \ : \ ((A \Rightarrow B) \Rightarrow A) \Rightarrow A \qquad \text{(Peirce's law)}$$

  - Interprets the axiom of dependent choices (DC)
- Initially designed for PA2 (+ DC), but extends to:
  - Higher-order arithmetic (PAω)
  - Zermelo-Fraenkel set theory (ZF)
  - The calculus of constructions with universes (with classical logic in Prop)
- Deep connections with Cohen forcing (3rd lecture)

Plan

Plan

Terms, stacks and processes

- Syntax of the language parameterized by
  - A countable set $\mathcal{K} = \{\text{cc}; \ldots\}$ of instructions, containing at least the instruction cc (call/cc)
  - A countable set Π_0 of stack constants (or stack bottoms)

Terms, stacks and processes

Terms	t, u	$::=$	$x \mid \lambda x\,.\,t \mid tu \mid \kappa \mid \mathsf{k}_\pi$	$(\kappa \in \mathcal{K})$
Stacks	π, π'	$::=$	$\alpha \mid t \cdot \pi$	$(\alpha \in \Pi_0,\ t$ closed$)$
Processes	p, q	$::=$	$t \star \pi$	$(t$ closed$)$

- A λ-calculus with two kinds of constants:
  - Instructions $\kappa \in \mathcal{K}$, including cc
  - Continuation constants k_π, one for every stack π (generated by cc)
- **Notation:** Λ, Π, $\Lambda \star \Pi$ (sets of closed terms / stacks / processes)

Proof-like terms

- **Proof-like term** $\equiv$ Term containing no continuation constant

Proof-like terms $\quad t, u \quad ::= \quad x \quad | \quad \lambda x \,.\, t \quad | \quad tu \quad | \quad \kappa \qquad (\kappa \in \mathcal{K})$

  - **Idea:** All realizers coming from actual proofs are of this form, continuation constants k_π are treated as paraproofs
  - **Notation:** PL $\equiv$ set of closed proof-like terms

- Natural numbers encoded as proof-like terms by:

Krivine numerals $\quad \overline{n} \;\equiv\; \overline{s}^n\,\overline{0} \;\in\; \mathrm{PL} \qquad (n \in \mathbb{N})$

writing $\overline{0} \equiv \lambda xy\,.\,x$ and $\overline{s} \equiv \lambda nxy\,.\,y\,(n\,x\,y)$

  - **Note:** Krivine numerals $\not\equiv$ Church numerals, but β-equivalent

The Krivine Abstract Machine (KAM) (1/2)

- We assume that the set $\Lambda \star \Pi$ comes with a preorder $p \succ p'$ of evaluation satisfying the following rules:

Krivine Abstract Machine (KAM)

Push	$tu \star \pi \succ t \star u \cdot \pi$	
Grab	$\lambda x\,.\,t \star u \cdot \pi \succ t\{x := u\} \star \pi$	
Save	$\mathbf{c} \star u \cdot \pi \succ u \star \mathrm{k}_\pi \cdot \pi$	
Restore	$\mathrm{k}_\pi \star u \cdot \pi' \succ u \star \pi$	
	$\ldots$	$\ldots$

(+ reflexivity & transitivity)

- Evaluation not defined but axiomatized. The preorder $p \succ p'$ is another parameter of the calculus, just like the sets $\mathcal{K}$ and Π_0
- Extensible machinery: can add extra instructions and rules
  (We shall see examples later)

The Krivine Abstract Machine (KAM) (2/2)

- Rules **Push** and **Grab** implement weak head β-reduction (call-by-name strategy):

Push	$tu \star \pi$	$\succ$	$t \star u \cdot \pi$
Grab	$\lambda x\,.\,t \star u \cdot \pi$	$\succ$	$t\{x := u\} \star \pi$

- Rules **Save** and **Restore** implement backtracking:

Save	$\text{œ} \star u \cdot \pi$	$\succ$	$u \star \mathsf{k}_\pi \cdot \pi$
Restore	$\mathsf{k}_\pi \star u \cdot \pi'$	$\succ$	$u \star \pi$

  - Instruction œ creates continuation constants k_π:

    Usage: $\text{œ}\,(\lambda k\,.\,t)$
    Intuition: let k = curr-cont() in t
    Computation: $\text{œ}\,(\lambda k\,.\,t) \star \pi \;\succ\; t\{k := \mathsf{k}_\pi\} \cdot \pi$

  - Continuation constant k_π restores the saved context π

Examples of extra instructions (1/2)

- The instruction quote

$$\mathrm{quote} \star t \cdot u \cdot \pi \quad \succ \quad u \star \overline{\lceil t \rceil} \cdot \pi$$

where $t \mapsto \lceil t \rceil$ is a fixed bijection from Λ to $\mathbb{N}$

  - Useful to realize the axiom of dependent choices [Krivine'03]

- The instruction eq

$$\mathrm{eq} \star t_1 \cdot t_2 \cdot u \cdot v \cdot \pi \quad \succ \quad \begin{cases} u \star \pi & \text{if } t_1 \equiv t_2 \\ v \star \pi & \text{if } t_1 \not\equiv t_2 \end{cases}$$

  - Tests syntactic equality $t_1 \equiv t_2$
  - Can be implemented using quote

- The instruction $\pitchfork$ (fork)

$$\pitchfork \star u \cdot v \cdot \pi \quad \succ \quad \begin{cases} u \star \pi \\ v \star \pi \end{cases}$$

  - Non deterministic choice operator
  - Useful for pedagogy – bad for realizability (collapses to forcing)

Examples of extra instructions (2/2)

- The instruction stop:

$$\mathsf{stop} \star \pi \quad \not\succ$$

Stops execution. Final result returned on the stack π

- The instruction print:

$$\mathsf{print} \star \overline{n} \cdot u \cdot \pi \quad \succ \quad u \star \pi \qquad \text{(formal specification)}$$

and prints integer n on standard output (informal specification)

$\leadsto$ Displays intermediate results without stopping the machine (poor man's side effect)

- The instruction make_coffee:

$$\mathsf{make\_coffee} \star u \cdot \pi \quad \succ \quad u \star \pi \quad + \quad \text{makes coffee}$$

Plan

The language of (minimal) second-order logic

- Second-order logic deals with two kinds of objects:
  - 1st-order objects = individuals (i.e. basic objects of the theory)
  - 2nd-order objects = k-ary relations over individuals

First-order terms and formulas

First-order terms	e, e'	$::=$	$x \mid f(e_1, \ldots, e_k)$
Formulas	A, B	$::=$	$X(e_1, \ldots, e_k) \mid A \Rightarrow B$
			$\mid \forall x\, A \mid \forall X\, A$

- Two kinds of variables
  - 1st-order vars: $x, y, z, \ldots$ (not to be confused with λ-variables!)
  - 2nd-order vars: $X, Y, Z, \ldots$ of all arities $k \geq 0$
- Two kinds of substitution:
  - 1st-order subst.: $e\{x := e_0\}$, $A\{x := e_0\}$ (defined as usual)
  - 2nd-order subst.: $A\{X := P_0\}$, $P\{X := P_0\}$ (postponed)

First-order terms

- Defined from a first order signature Σ (as usual):

First-order terms $\quad e, e' \quad ::= \quad x \quad | \quad f(e_1, \ldots, e_k)$

  - f ranges over k-ary function symbols in Σ

- In what follows we assume that:
  - Each k-ary function symbol f is interpreted in $\mathbb{N}$ by a function

$$f^{\mathbb{N}} \; : \; \mathbb{N}^k \to \mathbb{N}$$

  - The signature Σ contains a function symbol for every primitive recursive function: $\;0,\ s,\ +,\ \times,\ \uparrow,\ \ldots$

- Denotation (in $\mathbb{N}$) of a closed first-order term e written $[\![e]\!]$

Formulas

- Formulas of minimal second-order logic

Formulas $\quad A, B \quad ::= \quad X(e_1, \ldots, e_k) \quad | \quad A \Rightarrow B \quad | \quad \forall x\, A \quad | \quad \forall X\, A$

only based on implication and 1st/2nd-order universal quantification

- Other connectives/quantifiers are defined (second-order encodings)

$$
\begin{array}{rcll}
\bot & \equiv & \forall Z\, Z & \text{(absurdity)} \\
\neg A & \equiv & A \Rightarrow \bot & \text{(negation)} \\
A \wedge B & \equiv & \forall Z\,((A \Rightarrow B \Rightarrow Z) \Rightarrow Z) & \text{(conjunction)} \\
A \vee B & \equiv & \forall Z\,((A \Rightarrow Z) \Rightarrow (B \Rightarrow Z) \Rightarrow Z) & \text{(disjunction)} \\
\exists x\, A(x) & \equiv & \forall Z\,(\forall x\,(A(x) \Rightarrow Z) \Rightarrow Z) & \text{(1st-order } \exists) \\
\exists X\, A(X) & \equiv & \forall Z\,(\forall X\,(A(X) \Rightarrow Z) \Rightarrow Z) & \text{(2nd-order } \exists) \\
e_1 = e_2 & \equiv & \forall Z\,(Z(e_1) \Rightarrow Z(e_2)) & \text{(Leibniz equality)}
\end{array}
$$

Predicates

- 2nd-order variables represent unknown (abstract) relations
- Concrete relations are represented using predicates (syntactic sugar)

Predicates $P, Q \quad ::= \quad \hat{x}_1 \cdots \hat{x}_k\, A$ (of arity k)

- Let $P \equiv \hat{x}_1 \cdots \hat{x}_k\, A$
  - Variables $x_1, \ldots, x_n$ (pairwise $\neq$) are the arguments of P
  - Other free variables of formula A are the parameters of P
  - **Notation:** $FV(P) = FV(A) \setminus \{x_1, \ldots, x_k\}$ (free vars = params)
- Predicates are subject to α-conversion ($\hat{x}_i s$ treated as binders)
- 0-ary predicates are formulas

Predicate application / substitution

- Partial/total application of $P \equiv \hat{x}_1 \cdots \hat{x}_k A$ to $e_1, \ldots, e_\ell$:

$$P(e_1, \ldots, e_\ell) \;\equiv\; \hat{x}_{\ell+1} \cdots \hat{x}_k A\{x_1 := e_1; \ldots; x_\ell := e_\ell\} \qquad (\ell \leq k)$$

where $x_j \notin FV(e_i)$ for $i \in [1..\ell]$, $j \in [\ell+1..k]$)

Result is a $(k-\ell)$-ary predicate, and a formula if $k = \ell$

- Every k-ary 2nd-order variable may be viewed as a predicate:

$$X \;\equiv\; \hat{x}_1 \cdots \hat{x}_k X(x_1, \ldots, x_k)$$

- Second-order substitution (X, P of same arity)

$$(X(e_1, \ldots, e_k))\{X := P\} \;\equiv\; P(e_1, \ldots, e_k)$$

  - In a formula: $A\{X := P\}$
  - In a predicate: $Q\{X := P\}$

Unary predicates as sets

- Unary predicates represent sets of individuals

  Syntactic sugar: $\{x : A\} \equiv \hat{x}A, \qquad e \in P \equiv P(e)$

Example: The set $\mathbb{N}$ of Dedekind numerals

$$\mathbb{N} \equiv \{x : \forall Z\,(0 \in Z \Rightarrow \forall y\,(y \in Z \Rightarrow s(y) \in Z) \Rightarrow x \in Z\}$$

- Relativized quantifications:

$$(\forall x \in P)\,A(x) \equiv \forall x\,(x \in P \Rightarrow A(x))$$

$$(\exists x \in P)\,A(x) \equiv \forall Z\,(\forall x\,(x \in P \Rightarrow A(x) \Rightarrow Z) \Rightarrow Z)$$
$$\Leftrightarrow \exists x\,(x \in P \wedge A(x))$$

- Inclusion and extensional equality:

$$P \subseteq Q \equiv \forall x\,(x \in P \Rightarrow x \in Q)$$
$$P = Q \equiv \forall x\,(x \in P \Leftrightarrow x \in Q)$$

- Set constructors: $P \cup Q \equiv \{x : x \in P \vee x \in Q\}$ (etc.)

A type system for second-order logic (λNK2)

- Use proof-like terms as Curry-style proof terms

  Represent the computational contents of classical proofs

- **Typing judgement:** $\underbrace{x_1 : A_1, \ldots, x_n : A_n}_{\text{typing context } \Gamma} \vdash t : B$

Typing rules

$$\frac{}{\Gamma \vdash x : A}\ (x{:}A)\in\Gamma$$

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

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

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall X\, A}\ X \notin FV(\Gamma) \qquad \frac{\Gamma \vdash t : \forall X\, A}{\Gamma \vdash t : A\{X := P\}}$$

$$\frac{}{\Gamma \vdash \text{cc} : ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$$

Typing examples

- Intuitionistic principes:

$$\begin{array}{rcl}
\mathbf{pair} & \equiv & \lambda xyz\,.\,z\,x\,y \quad : \quad \forall X\,\forall Y\,(X \Rightarrow Y \Rightarrow X \wedge Y) \\
\mathbf{fst} & \equiv & \lambda z\,.\,z\,(\lambda xy\,.\,x) \quad : \quad \forall X\,\forall Y\,(X \wedge Y \Rightarrow X) \\
\mathbf{snd} & \equiv & \lambda z\,.\,z\,(\lambda xy\,.\,y) \quad : \quad \forall X\,\forall Y\,(X \wedge Y \Rightarrow Y) \\
\mathbf{refl} & \equiv & \lambda z\,.\,z \quad : \quad \forall x\,(x = x) \\
\mathbf{trans} & \equiv & \lambda xyz\,.\,y\,(x\,z) \quad : \quad \forall x\,\forall y\,\forall z\,(x = y \Rightarrow y = z \Rightarrow x = z)
\end{array}$$

- Excluded middle, double negation elimination:

$$\begin{array}{rcl}
\mathbf{left} & \equiv & \lambda xuv\,.\,u\,x \quad : \quad \forall X\,\forall Y\,(X \Rightarrow X \vee Y) \\
\mathbf{right} & \equiv & \lambda yuv\,.\,v\,y \quad : \quad \forall X\,\forall Y\,(Y \Rightarrow X \vee Y) \\
\mathbf{EM} & \equiv & \propto(\lambda k\,.\,\mathbf{right}\,(\lambda x\,.\,k\,(\mathbf{left}\,x))) \quad : \quad \forall X\,(X \vee \neg X) \\
\mathbf{DNE} & \equiv & \lambda z\,.\propto(\lambda k\,.\,z\,k) \quad : \quad \forall X\,(\neg\neg X \Rightarrow X)
\end{array}$$

- De Morgan laws:

$$\begin{array}{rcl}
\lambda zy\,.\,z\,(\lambda x\,.\,yx) & : & \exists x\,A(x) \;\Rightarrow\; \neg\forall x\,\neg A(x) \\
\lambda zy\,.\propto(\lambda k\,.\,z\,(\lambda x\,.\,k\,(y\,x))) & : & \neg\forall x\,\neg A(x) \;\Rightarrow\; \exists x\,A(x)
\end{array}$$

Classical second-order logic: PA2

System λNK2 defines provability in classical 2nd-order *logic* (NK2).
For classical 2nd-order *arithmetic* (PA2), add the following axioms:

- Defining equations of primitive recursive functions:

$$\begin{array}{lll} \forall x\,(x+0=x) & \forall x\,\forall y\,(x+s(y)=s(x+y)) & \\ \forall x\,(x\times 0=0) & \forall x\,\forall y\,(x\times s(y)=x\times y+x) & \text{(etc.)} \end{array}$$

- Peano 3rd and 4th axioms:

$$\begin{array}{ll} \text{(P3)} & \forall x\,\forall y\,(s(x)=s(y)\Rightarrow x=y) \\ \text{(P4)} & \forall x\,\neg(s(x)=0) \end{array}$$

- The induction axiom:

$$\begin{array}{rcl} \mathrm{Ind} & \equiv & \forall x\,(x\in\mathbb{N}) \\ & \Leftrightarrow & \forall Z\,[0\in Z\Rightarrow\forall y\,(y\in Z\Rightarrow s(y)\in Z)\Rightarrow\forall x\,(x\in Z)] \end{array}$$

The induction axiom

- **Problem:** The induction axiom is not realizable!

$$\begin{array}{rcl} \mathrm{Ind} & \equiv & \forall x\,(x \in \mathbb{N}) \\ & \Leftrightarrow & \forall Z\,[0 \in Z \Rightarrow \forall y\,(y \in Z \Rightarrow s(y) \in Z) \Rightarrow \forall x\,(x \in Z)] \end{array}$$

- **Solution:** Relativize all 1st-order quantifications to $\mathbb{N}$:

Non relativized		**Relativized**
$\forall x\, A(x)$	$\rightsquigarrow$	$(\forall x \in \mathbb{N})\, A(x)$ $\forall x\,(x{\in}\mathbb{N}{\Rightarrow}A(x))$
$\exists x\, A(x)$ $\forall Z\,(\forall x\,(A(x){\Rightarrow}Z){\Rightarrow}Z)$	$\rightsquigarrow$	$(\exists x \in \mathbb{N})\, A(x)$ $\forall Z\,(\forall x\,(x{\in}\mathbb{N}{\Rightarrow}A(x){\Rightarrow}Z){\Rightarrow}Z)$

Theorem

If $\mathrm{PA2} \vdash A$, then $\mathrm{PA2} - \mathrm{Ind} \vdash A^{\mathbb{N}}$ $(A^{\mathbb{N}} \equiv A$ relativized to $\mathbb{N})$

Computational contents of relativization

- **Intuition:**

$$\begin{array}{rcll} (\forall x \in \mathbb{N})\, A(x) & \equiv & \forall x\,(x \in \mathbb{N} \Rightarrow A(x)) & \\ & \approx & (\Pi x \in \mathrm{nat})\, A(x) & \text{(Coq, Agda)} \end{array}$$

- Recall: $x \in \mathbb{N} \;\equiv\; \forall Z\,[Z(0) \Rightarrow \forall y\,(Z(y) \Rightarrow Z(s(y))) \Rightarrow Z(x)]$

$$\begin{array}{lcll} \overline{0} & \equiv & \lambda zf\,.\,fz & : \; 0 \in \mathbb{N} \\ \overline{s} & \equiv & \lambda nzf\,.\,f\,(n\,z\,f) & : \; (\forall x \in \mathbb{N})\, s(x) \in \mathbb{N} \\ \overline{n} & \equiv & \overline{s}^n\overline{0} & : \; n \in \mathbb{N} \\ \mathrm{plus} & \equiv & \lambda nm\,.\,m\,n\,\overline{s} & : \; (\forall x, y \in \mathbb{N})\, x + y \in \mathbb{N} \\ \mathrm{mult} & \equiv & \lambda nm\,.\,m\,\overline{0}\,(\lambda p\,.\,\mathrm{plus}\,p\,n) & : \; (\forall x, y \in \mathbb{N})\, x \times y \in \mathbb{N} \end{array}$$

(etc.)

Plan

Classical realizability: principles

- **Intuitions:**
  - term = "proof" / stack = "counter-proof"
  - process = "contradiction" (slogan: never trust a classical realizer!)
- Classical realizability model parameterized by a pole $\perp\!\!\!\perp$

  = set of processes closed under anti-evaluation (or saturated)

$$\text{If} \quad p \succ p' \quad \text{and} \quad p' \in \perp\!\!\!\perp, \quad \text{then} \quad p \in \perp\!\!\!\perp$$

- Each formula A is interpreted as two sets:
  - A set of stacks $\|A\|$ (falsity value)
  - A set of terms $|A|$ (truth value)
- Falsity value $\|A\|$ defined by induction on A (negative interpretation)
- Truth value $|A|$ defined by orthogonality:

$$|A| \quad = \quad \|A\|^{\perp\!\!\!\perp} \quad = \quad \{t \in \Lambda \ : \ \forall \pi \in \|A\| \ \ t \star \pi \in \perp\!\!\!\perp\}$$

Architecture of the realizability model

- The realizability model $\mathscr{M}_{\perp\!\!\!\perp}$ is defined from:
  - The full standard model $\mathscr{M}$ of PA2: the ground model
    (but we could take any model $\mathscr{M}$ of PA2 as well)
  - An instance $(\mathcal{K}, \Pi_0, \succ)$ of the λ_c-calculus
  - A saturated set of processes $\perp\!\!\!\perp \subseteq \Lambda \star \Pi$ (the pole)
- Architecture:
  - First-order terms/variables interpreted as natural numbers $n \in \mathbb{N}$
  - Formulas interpreted as falsity values $S \in \mathfrak{P}(\Pi)$
  - k-ary 2nd-order variables (and k-ary predicates) interpreted as falsity functions $F : \mathbb{N}^k \to \mathfrak{P}(\Pi)$.

Formulas with parameters $\quad A, B \quad ::= \quad \cdots \quad | \quad \dot{F}(e_1, \ldots, e_k)$

Add a predicate constant $\dot{F}$ for every falsity function $\dot{F} : \mathbb{N}^k \to \mathfrak{P}(\Pi)$

Interpreting closed formulas with parameters

Let A be a closed formula (with parameters)

- Falsity value $\|A\|$ defined by induction on A:

$$\begin{aligned}
\|\dot{F}(e_1,\ldots,e_n)\| &= F(\llbracket e_1\rrbracket,\ldots,\llbracket e_n\rrbracket)\\
\|A \Rightarrow B\| &= |A|\cdot\|B\| \;=\; \{t\cdot\pi \;:\; t\in|A|,\;\; \pi\in\|B\|\}\\
\|\forall x\; A\| &= \bigcup_{n\in\mathbb{N}} \|A\{x:=n\}\|\\
\|\forall X\; A\| &= \bigcup_{F:\mathbb{N}^k\to\mathfrak{P}(\Pi)} \|A\{X:=\dot{F}\}\|
\end{aligned}$$

- Truth value $|A|$ defined by orthogonality:

$$|A| \;=\; \|A\|^{\perp\!\!\!\perp} \;=\; \{t\in\Lambda \;:\; \forall\pi\in\|A\|\quad t\star\pi\in{\perp\!\!\!\perp}\}$$

The realizability relation

Falsity value $\|A\|$ and truth value $|A|$ depend on the pole $\perp\!\!\!\perp$

$\rightsquigarrow$ write them (sometimes) $\|A\|_{\perp\!\!\!\perp}$ and $|A|_{\perp\!\!\!\perp}$ to recall the dependency

Realizability relations

$$t \Vdash A \quad \equiv \quad t \in |A|_{\perp\!\!\!\perp} \qquad \text{(Realizability w.r.t. } \perp\!\!\!\perp\text{)}$$

$$t \Vvdash A \quad \equiv \quad \forall \perp\!\!\!\perp \;\; t \in |A|_{\perp\!\!\!\perp} \qquad \text{(Universal realizability)}$$

From computation to realizability (1/2)

Fundamental idea: The computational behavior of a term determines the formula(s) it realizes:

Example 1: A closed term t is identity-like if:

$$t \star u \cdot \pi \;\succ\; u \star \pi \qquad\qquad \text{for all } u \in \Lambda,\ \pi \in \Pi$$

Proposition

If t is identity-like, then $t \Vdash \forall X\,(X \Rightarrow X)$

Proof: Exercise! (Remark: converse implication holds – exercise!)

- Examples of identity-like terms:
  - $\lambda x\,.\,x$, $(\lambda x\,.\,x)\,(\lambda x\,.\,x)$, etc.
  - $\lambda x\,.\,\text{cc}\,\lambda k\,.\,x$, $\lambda x\,.\,\text{cc}\,\lambda k\,.\,k\,x$, $\lambda x\,.\,\text{cc}\,\lambda k\,.\,k\,x\,(\delta\,\delta)$, etc.
  - $\lambda x\,.\,\text{marshal}\,x\,\lambda n\,.\,\text{unmarshal}\,n\,(\lambda z\,.\,z)$

From computation to realizability (2/2)

Example 2: Control operators:

$$\begin{array}{lll} \mathfrak{c} \star t \cdot \pi & \succ & t \star \mathsf{k}_\pi \cdot \pi \\ \mathsf{k}_\pi \star t \cdot \pi' & \succ & t \star \pi \end{array}$$

- "Typing" k_π: $\mathsf{k}_\pi \star t \cdot \pi' \succ t \star \pi$

Lemma

If $\pi \in \|A\|$, then $\mathsf{k}_\pi \Vdash A \Rightarrow B$ (B any)

Proof: Exercise

- "Typing" $\mathfrak{c}$: $\mathfrak{c} \star t \cdot \pi \succ t \star \mathsf{k}_\pi \cdot \pi$

Proposition (Peirce's law)

$\mathfrak{c} \Vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$

Proof: Exercise

Adequacy (1/2)

Aim: Prove the theorem of adequacy:

$t : A$ (in the sense of λNK2) implies $t \Vdash A$ (in the sense of realizability)

- Closing typing judgments $x_1 : A_1, \ldots, x_n : A_n \vdash t : A$
  - We close logical objects (1st-order terms, formulas, predicates) using semantic objects (natural numbers, falsity values, falsity functions)
  - We close proof-terms using realizers

Definition (Valuations)

1. A valuation is a function ρ such that:
   - $\rho(x) \in \mathbb{N}$ for each 1st-order variable x
   - $\rho(X) : \mathbb{N}^k \to \mathfrak{P}(\Pi)$ for each 2nd-order variable X of arity k
2. Closure of A with ρ written $A[\rho]$ (formula with parameters)

Adequacy (2/2)

Definition (Adequate judgment, adequate rule)

Given a fixed pole $\perp\!\!\!\perp$:

1. A judgment $x_1 : A_1, \ldots, x_n : A_n \vdash t : A$ is adequate if for every valuation ρ and for all $u_1 \Vdash A_1[\rho], \ldots, u_n \Vdash A_n[\rho]$ we have:

$$t\{x_1 := u_1, \ldots, x_n := u_n\} \Vdash A[\rho]$$

2. A typing rule is adequate if it preserves the property of adequacy (from the premises to the conclusion of the rule)

Theorem

1. All typing rules of λNK2 are adequate
2. All derivable judgments of λNK2 are adequate

Corollary: If $\vdash t : A$ (A closed formula), then $t \Vdash\!\!\Vdash A$

Extending adequacy to subtyping

Definition (Adequate subtyping judgment)

Judgment $A \leq B$ adequate $\quad \equiv \quad \|B[\rho]\| \subseteq \|A[\rho]\|$ (for all valuations ρ)

Remark: Implies: $|A[\rho]| \subseteq |B[\rho]|$ (for all ρ), but strictly stronger

Adequate typing/subtyping rules

$$\frac{}{A \leq A} \qquad \frac{A \leq B \qquad B \leq C}{A \leq C} \qquad \frac{\Gamma \vdash t : A \qquad A \leq B}{\Gamma \vdash t : B}$$

$$\frac{}{\forall x\, A \;\leq\; A\{x := e\}} \qquad \frac{}{\forall X\, A \;\leq\; A\{X := P\}}$$

$$\frac{A \leq B}{A \leq \forall x\, B}\; x \notin FV(A) \qquad \frac{A \leq B}{A \leq \forall X\, B}\; X \notin FV(A) \qquad \frac{A' \leq A \qquad B \leq B'}{A \Rightarrow B \;\leq\; A' \Rightarrow B'}$$

$$\frac{}{\forall x\,(A \Rightarrow B) \;\leq\; A \Rightarrow \forall x\, B}\; x \notin FV(A) \qquad \frac{}{\forall X\,(A \Rightarrow B) \;\leq\; A \Rightarrow \forall X\, B}\; X \notin FV(A)$$

- Example: $\underbrace{\forall X\, \forall Y\, (((X \Rightarrow Y) \Rightarrow X) \Rightarrow X)}_{\text{Peirce's law}} \;\leq\; \underbrace{\forall X\, (\neg\neg X \Rightarrow X)}_{\text{DNE}}$

Anatomy of the model (1/2)

- **Denotation of universal quantification:**

  Falsity value: $\|\forall x\, A\| \;=\; \bigcup_{n\in\mathbb{N}} \|A\{x := n\}\|$ (by definition)

  Truth value: $|\forall x\, A| \;=\; \bigcap_{n\in\mathbb{N}} |A\{x := n\}|$ (by orthogonality)

  (and similarly for 2nd-order universal quantification)

- **Denotation of implication:**

  Falsity value: $\|A \Rightarrow B\| \;=\; |A| \cdot \|B\|$ (by definition)

  Truth value: $|A \Rightarrow B| \;\subseteq\; |A| \to |B|$ (by orthogonality)

  writing $|A| \to |B| \;=\; \{t \in \Lambda \;:\; \forall u \in |A| \;\; tu \in |B|\}$ (realizability arrow)

  1. Converse inclusion does not hold in general, unless $\perp\!\!\!\perp$ closed under **Push**
  2. In all cases: If $t \in |A| \to |B|$, then $\lambda x\,.\,tx \in |A \Rightarrow B|$ (η-expansion)

Anatomy of the model (2/2)

- Falsity value $\|A\|$ and truth value $|A|$ depend on the pole $\perp\!\!\!\perp$
  $\rightsquigarrow$ write them $\|A\|_{\perp\!\!\!\perp}$ and $|A|_{\perp\!\!\!\perp}$ to recall the dependecy
- **Degenerate case:** $\perp\!\!\!\perp = \varnothing$
  - Truth values can take only two values: $\varnothing$ and Λ
  - Classical realizability simply mimics the Tarski interpretation:

Degenerated interpretation

In the case where $\perp\!\!\!\perp = \varnothing$, for every closed formula A:

$$|A| = \begin{cases} \Lambda & \text{if } \mathscr{M} \models A \\ \varnothing & \text{if } \mathscr{M} \not\models A \end{cases}$$

- **Non degenerate cases:** $\perp\!\!\!\perp \neq \varnothing$
  - Every truth value $|A|$ is inhabited:
    If $t_0 \star \pi_0 \in \perp\!\!\!\perp$, then $\mathrm{k}_{\pi_0} t_0 \in |A|$ for all A (paraproof)
  - We shall only consider realizers that are proof-like terms ($\in$ PL)

Provability, universal realizability and truth

- From what precedes:
  1. A provable $\Rightarrow$ A universally realized (by a proof-like term)
  2. A universally realized $\Rightarrow$ A true (in the standard model)

$\leadsto$ Universal realizability: an intermediate notion between provability and truth

- **Beware!**

$$\begin{array}{ccc} \text{Intuitionistic proofs of } A & \subset & \text{Classical proofs of } A \\ \cap & & \cap \\ \text{Intuitionistic realizers of } A & \not\subseteq \; \not\supseteq & \text{Classical realizers of } A \end{array}$$

Plan

The axioms of PA2 (recall)

- Defining equations of primitive recursive functions:

$$\begin{array}{ll} \forall x\,(x+0=x) & \forall x\,\forall y\,(x+s(y)=s(x+y)) \\ \forall x\,(x\times 0=0) & \forall x\,\forall y\,(x\times s(y)=x\times y+x) \qquad \text{(etc.)} \end{array}$$

- Peano 3rd and 4th axioms:

$$\begin{array}{ll} \text{(P3)} & \forall x\,\forall y\,(s(x)=s(y)\Rightarrow x=y) \\ \text{(P4)} & \forall x\,\neg(s(x)=0) \end{array}$$

- The induction axiom:

$$\begin{array}{lll} \text{Ind} & \equiv & \forall x\,(x\in\mathbb{N}) \\ & \Leftrightarrow & \forall Z\,[0\in Z\Rightarrow\forall y\,(y\in Z\Rightarrow s(y)\in Z)\Rightarrow\forall x\,(x\in Z)] \end{array}$$

- **Beware!** Since induction is not realizable (in general), we work in $\mathrm{PA2}^- = \mathrm{PA2} - \mathrm{Ind}$, relativizing all 1st-order $\forall/\exists$ to $\mathbb{N}$

Realizing equalities

- Equality between individuals defined by:

$$e_1 = e_2 \;\equiv\; \forall Z\,(Z(e_1) \Rightarrow Z(e_2)) \qquad \text{(Leibniz equality)}$$

Denotation of Leibniz equality

Given two closed 1st-order terms e_1, e_2 (and a pole $\perp\!\!\!\perp$)

$$\|e_1 = e_2\| \quad = \quad \begin{cases} \|\mathbf{1}\| \;=\; \{t\cdot\pi \;:\; (t\star\pi)\in\perp\!\!\!\perp\} & \text{if } [\![e_1]\!] = [\![e_2]\!] \\ \|\top\Rightarrow\bot\| \;=\; \Lambda\cdot\Pi & \text{if } [\![e_1]\!] \neq [\![e_2]\!] \end{cases}$$

where $\mathbf{1} \equiv \forall Z\,(Z \Rightarrow Z)$ and $\top \equiv \dot{\varnothing}$

- Intuitions:
  - A realizer of a true equality (in the model) behaves as the identity function $\lambda z\,.\,z$
  - A realizer of a false equality (in the model) behaves as a point of backtrack (breakpoint)

Realizing Peano axioms

Corollary 1 (Realizing true equations)

If $\mathscr{M} \models \forall \vec{x}\,(e_1(\vec{x}) = e_2(\vec{x}))$ (truth in the ground model)

then $\lambda z\,.\,z \Vdash \forall \vec{x}\,(e_1(\vec{x}) = e_2(\vec{x}))$ (universal realizability)

Corollary 2

All the defining equations of the arithmetic function symbols $(+,\ \times,\ \uparrow,$ etc.$)$ are realized by $\lambda z\,.\,z$

Corollary 3 (Realizing Peano axioms)

$$\lambda z\,.\,z \Vdash \forall x\,\forall y\,(s(x) = s(y) \Rightarrow x = y)$$
$$\lambda z\,.\,z\,(\lambda w\,.\,w) \Vdash \forall x\,\neg(s(x) = 0)$$

Remark: Corollary 3 generalizes to all the Horn formulas that are true in the ground model (using trivial realizers)

Program extraction

Extracting a program from a proof in PA2

If PA2 $\vdash A$, then there is $t \in \mathrm{PL}$ such that $t \Vdash A^{\mathbb{N}}$

($A^{\mathbb{N}}$ obtained from A by relativizing all 1st-order quantifications to $\mathbb{N}$)

- **In practice:**
  - Only apply the adequacy theorem to the computationally relevant parts of the proof
  - For the computationally irrelevant parts (i.e. Horn formulas), use 'default realizers' $\rightsquigarrow$ realizer optimization
- **Example 1:** $\lambda nmz\,.\,z \Vdash (\forall x, y \in \mathbb{N})\ x + y = y + x$
- **Example 2:** Fermat's last theorem[1]

$$(\forall x, y, z, n \in \mathbb{N})\ (x \geq 1 \Rightarrow y \geq 1 \Rightarrow n \geq 3 \Rightarrow x^n + y^n \neq z^n)$$

1. realized by: $\lambda xyznuvw\,.\,u\,(v\,(w\,\mathbf{I}))$

Plan

Some problems of classical realizability

1. **The specification problem**

   Given a formula A, characterize its universal realizers
   from their computational behavior

   Specifying Peirce's law [Guillermo-Miquel'12]

2. **Witness extraction from classical realizers** (cf below)

3. **Realizability algebras + Cohen forcing** (3rd lecture)

   Realizability algebras: a program to well-order $\mathbb{R}$ [Krivine'11]
   Forcing as a program transformation [Miquel'11]

4. **Models induced by classical realizability**

   What are the interesting formulas that are realized in $\mathscr{M}_{\perp\!\!\!\perp}$
   that are not already true in the ground model $\mathscr{M}$?

   Realizability algebras II: new models of ZF + DC [Krivine'12]

The problem of witness extraction

- **Problem:** Extract a witness from a universal realizer (or a proof)

$$t_0 \ \Vdash\ (\exists x \in \mathbb{N})\, A(x)$$

i.e. some $n \in \mathbb{N}$ such that $A(n)$ is true

- This is not always possible!

$$t_0 \quad \Vdash \quad (\exists x \in \mathbb{N})\,((x = 1 \wedge C) \vee (x = 0 \wedge \neg C))$$

(C = Continuum hypothesis, Goldbach's conjecture, etc.)

- Two possible compromises:
  - Intuitionistic logic: restrict the shape of the realizer t_0 (by only keeping intuitionistic reasoning principles)
  - Classical logic: restrict the shape of the formula $A(x)$ (typically: Δ^0_0-formulas)

Storage operators

- The call-by-value implication:

Formulas $\quad A, B \quad ::= \quad \cdots \quad | \quad \{e\} \Rightarrow A$

Semantics: $\quad \|\{e\} \Rightarrow A\| \;=\; \{\overline{n} \cdot \pi \;:\; n = [\![e]\!],\ \pi \in \|A\|\}$

- From the definition: $\quad e \in \mathbb{N} \Rightarrow A \;\leq\; \{e\} \Rightarrow A$

so that: $\quad \mathbf{I} \;\Vdash\; \forall x\, \forall Z\, ((x \in \mathbb{N} \Rightarrow Z) \Rightarrow (\{x\} \Rightarrow Z))$ (direct implication)

Definition (Storage operator)

A storage operator is a closed proof-like term M such that:

$$M \;\Vdash\; \forall x\, \forall Z\, ((\{x\} \Rightarrow Z) \Rightarrow (x \in \mathbb{N} \Rightarrow Z)) \qquad \text{(converse implication)}$$

- They exist, for instance: $\quad M \;\equiv\; \lambda fn\,.\, n\, f\, (\lambda hx\,.\, h\, (\overline{s}\, x))\, \overline{0}$
- **Conclusion:** $\quad e \in \mathbb{N} \Rightarrow A \quad$ and $\quad \{e\} \Rightarrow A \quad$ interchangeables

Storage operators

- Intuitively, a storage operator

$$M \quad \Vdash \quad \forall x\, \forall Z\, ((\{x\} \Rightarrow Z) \Rightarrow (x \in \mathbb{N} \Rightarrow Z))$$

  is a proof-like term that is intended to be applied to

  - a function f specified only on totally evaluated numerals (i.e. intuitionistic integers)
  - a classical integer $t \Vdash n \in \mathbb{N}$ (n arbitrary)

  and that evaluates (or "smoothes") the classical integer t into a value of the form $\overline{n}$ before passing this value to f

- Alternative point of view:

$$M \quad \Vdash \quad \forall x\, (\{x\} \Rightarrow A(x)) \;\Rightarrow\; (\forall x \in \mathbb{N})\, A(x)$$

  A property that holds for all values (i.e. intuitionistic integers) also holds for all classical integers

Computing with storage operators: an example

- Given a k-ary function f, we let:

$$\begin{array}{lcl} \mathrm{Total}(f) & \equiv & (\forall x_1 \in \mathbb{N}) \cdots (\forall x_k \in \mathbb{N})\, (f(x_1, \ldots, x_k) \in \mathbb{N}) \\ \mathrm{Comput}(f) & \equiv & \forall x_1 \cdots \forall x_k\, \forall Z\ \ [\{x_1\} \Rightarrow \cdots \Rightarrow \{x_k\} \Rightarrow \\ & & \qquad (\{f(x_1, \ldots, x_k)\} \Rightarrow Z) \Rightarrow Z] \end{array}$$

Theorem (Specification of the formula Comput(f))

For all $t \in \Lambda$, the following assertions are equivalent:

1. $t \ \Vdash\ \mathrm{Comput}(f)$
2. t computes f: for all $(n_1, \ldots, n_k) \in \mathbb{N}^k$, $u \in \Lambda$, $\pi \in \Pi$:

$$t \star \overline{n}_1 \cdots \overline{n}_k \cdot u \cdot \pi \quad \succ \quad u \star \overline{f(n_1, \ldots, n_k)} \cdot \pi$$

- Using a storage operator M, we can build proof-like terms such that:

$$\begin{array}{lcl} \xi_k & \Vdash & \mathrm{Total}(f) \quad \Rightarrow \quad \mathrm{Comput}(f) \\ \xi'_k & \Vdash & \mathrm{Comput}(f) \quad \Rightarrow \quad \mathrm{Total}(f) \end{array}$$

The naive extraction method

- A classical realizer $t_0 \Vdash (\exists x \in \mathbb{N})\, A(x)$ always evaluates to a pair witness/justification

Naive extraction

If $t_0 \Vdash (\exists x \in \mathbb{N})\, A(x)$, then there are $n \in \mathbb{N}$ and $u \in \Lambda$ such that

$$t_0 \star M\,(\lambda xy\,.\,\mathrm{stop}\,x\,y) \cdot \pi \quad \succ \quad \mathrm{stop} \star \overline{n} \cdot u \cdot \pi$$

(w.r.t. a particular pole $\perp\!\!\!\perp$)

- But $n \in \mathbb{N}$ might be a false witness because the justification $u \Vdash A(n)$ is cheating! (i.e. u might contain hidden continuations)
- In the case where t_0 comes from an intuitionistic proof, extracted witness $n \in \mathbb{N}$ is always correct

Extraction in the Σ^0_1-case

Extraction in the Σ^0_1-case (+ display intermediate results)

If $t_0 \Vdash (\exists x \in \mathbb{N})\,(f(x) = 0)$, then

$$t_0 \star M\,(\lambda xy\,.\,\mathrm{print}\,x\,y\,(\mathrm{stop}\,x)) \cdot \pi \quad \succ \quad \mathrm{stop} \star \overline{n} \cdot \pi$$

for some $n \in \mathbb{N}$ such that $f(n) = 0$

- Storage operator M used to evaluate 1st component
- 2nd component (y) used as a breakpoint
  (Relies on the particular structure of equality realizers)
- Holds independently from the instruction set
- Supports any representation of numerals
  (one have to implement the storage operator M accordingly)

Extraction in the Σ^0_n-case (1/2)

Definition (conditional refutation)

$r_A \in \Lambda$ is a conditional refutation of the predicate $A(x)$ if

$$\text{For all } n \in \mathbb{N} \text{ such that } \mathscr{M} \not\models A(n): \quad r_A \overline{n} \ \Vdash \ \neg A(n)$$

- Such a conditional refutation can be constructed for every predicate $A(x)$ of 1st-order arithmetic

  This result is a consequence of the:

Theorem [Krivine-Miquey]

For every formula A of 1st-order arithmetic, there exists a proof-like term t_A such that:

$$\text{If} \quad \mathscr{M} \models A \quad \text{then} \quad t_A \ \Vdash \ A$$

Extraction in the Σ^0_n-case (2/2)

The Kamikaze extraction method

Let

1. $t_0 \ \Vdash\ (\exists x \in \mathbb{N})\, A(x)$
2. r_A a conditional refutation of the predicate $A(x)$

Then the process

$$t_0 \star M\,(\lambda xy\,.\,\mathrm{print}\,x\,(r_A\,x\,y)) \cdot \pi$$

displays a correct witness after finitely many evaluation steps

- **Remark:** No correctness invariant is ensured as soon as the (first) correct witness has been displayed!

  After, everything may happen: crash, infinite loop, displaying incorrect witnesses, etc.

Primitive numerals

Extend the machine with the following instructions:

- For every integer $n \in \mathbb{N}$, an instruction $\widehat{n} \in \mathcal{K}$ with no evaluation rule (i.e. inert constant).

  Intuition: $\hat{n} \star \pi \succ$ `segmentation fault`

- An instruction null with the rules:

$$\mathrm{null} \star \widehat{n} \cdot u \cdot v \cdot \pi \quad \succ \quad \begin{cases} u \star \pi & \text{if } n = 0 \\ v \star \pi & \text{otherwise} \end{cases}$$

- Instructions $\check{f}$ with the rules:

$$\check{f} \star \widehat{n}_1 \cdots \widehat{n}_k \cdot u \cdot \pi \quad \succ \quad u \star \widehat{m} \cdot \pi \qquad \text{where } m = f(n_1, \ldots, n_k)$$

  for all the usual arithmetic operations

Primitive numerals (2/2)

- Call-by-value implication, yet another definition:

Formulas $\quad A, B \quad ::= \quad \cdots \quad | \quad [e] \Rightarrow A$

Semantics: $\quad \|[e] \Rightarrow A\| \quad = \quad \{\widehat{n} \cdot \pi \ : \ n = [\![e]\!],\ \pi \in \|A\|\}$

- Redefining the set of natural numbers:
$\mathbb{N}' \ \equiv \ \{x \ : \ \forall Z\,(([e] \Rightarrow Z) \Rightarrow Z)\}$

$$
\begin{array}{lcl}
\mathrm{box} \ \equiv \ \lambda x\,r\,.\,r\,x & \Vdash & \forall x\,([x] \Rightarrow x \in \mathbb{N}') \\
\mathrm{box}\,\widehat{n} & \Vdash & n \in \mathbb{N}' \\
\lambda n\,.\,n\,\lambda x\,.\,\check{s}\,x\,\mathrm{box} & \Vdash & (\forall x \in \mathbb{N}')\,s(x) \in \mathbb{N}' \\
\lambda nm\,.\,n\,\lambda x\,.\,m\,\lambda y\,.\,(\check{+})\,x\,y\,\mathrm{box} & \Vdash & (\forall x, y \in \mathbb{N}')\,x + y \in \mathbb{N}'
\end{array}
$$

$$
\begin{array}{rl}
\mathrm{rec\_cbv} & \equiv \ \lambda z_0 z_s\,.\,\mathbf{Y}\,\lambda xr\,.\,\mathrm{null}\,x\,z_0\,(\mathrm{pred}\,x\,\lambda y\,.\,z_s\,y\,(r\,y)) \\
& \Vdash \ \forall Z\,[Z(0) \ \Rightarrow \ \forall y\,([y] \Rightarrow Z(y) \Rightarrow Z(s(y))) \ \Rightarrow \ \forall x\,([x] \Rightarrow Z(x))]
\end{array}
$$

$$
\begin{array}{rl}
\mathrm{rec} & \equiv \ \lambda z_0 z_s n\,.\,n\,\lambda x\,.\,\mathrm{rec\_cbv}\,z_0\,(\lambda yz\,.\,z_s\,(\mathrm{box}\,y)\,z)\,x \\
& \Vdash \ \forall Z\,[Z(0) \ \Rightarrow \ (\forall y \in \mathbb{N}')\,(Z(y) \Rightarrow Z(s(y))) \ \Rightarrow \ (\forall x \in \mathbb{N}')\,Z(x)]
\end{array}
$$

- **Conclusion:** $\quad \Vdash \ \forall x\,(x \in \mathbb{N}' \ \Leftrightarrow \ x \in \mathbb{N})$

Example: the minimum principle (in Coq)

Minimum principle (for an inhabited type T)

$$(\forall f : T \to \mathbb{N})\ (\exists x : T)\ \underbrace{(\forall y : T)\ \ f(x) \leq f(y)}_{\text{undecidable}}$$

Proof: Reductio ad absurdum + course-by-value induction.

- **Remark:** No intuitionistic proof (oracle)

Corollary

$$(\forall f : \mathbb{N} \to \mathbb{N})\ (\exists x : \mathbb{N})\ \underbrace{f(x) \leq f(2x+1)}_{\text{decidable}}$$

More generally: $(\forall f, g : \mathbb{N} \to \mathbb{N})\ (\exists x : \mathbb{N})\ f(x) \leq f(g(x))$

Proof. Take the point x given by the minimum principle.

Krivine's realizability *vs* the LRS R-translation (1/2)

- Krivine's realizability can seen as the composition of the Lafont-Reus-Streicher (LRS) R-translation with Kleene's realizability

$$\text{CPS} \circ \text{Krivine} \quad = \quad \text{Kleene} \circ \text{LRS}$$ [Oliva-Streicher'08]

The dictionary

Classical realizability	**LRS R-translation**
Pole $\perp\!\!\!\perp$	Return formula R
Falsity value $\|A\|$	Negative translation $A^{\perp}$
$\|A \Rightarrow B\| = \lvert A\rvert \cdot \|B\|$	$(A \Rightarrow B)^{\perp} \equiv A^{LRS} \wedge B^{\perp}$
Truth value $\lvert A\rvert = \|A\|^{\perp\!\!\!\perp}$	$A^{LRS} \equiv A^{\perp} \Rightarrow R$

- Through the CPS translation, Krivine's extraction method in the Σ^0_1-case is exactly Friedman's trick [Miquel'10]

Krivine's realizability *vs* the LRS R-translation (2/2)

Beware of reductionism!

- It only holds for *pure* classical reasoning
  (extra instructions are not taken into account)
- Classical realizers are easier to understand than their CPS-translations (and more efficient...)
- Classical realizability is more than Kleene's realizability composed with the Lafont-Reus-Streicher R-translation!

An image:

$$2H_2 + O_2 \quad \longrightarrow \quad 2H_2O$$

but can we deduce the properties of *water* from the ones of H_2 and O_2?