Computational interpretation of classical forcing

Lionel Rieg

Collège de France

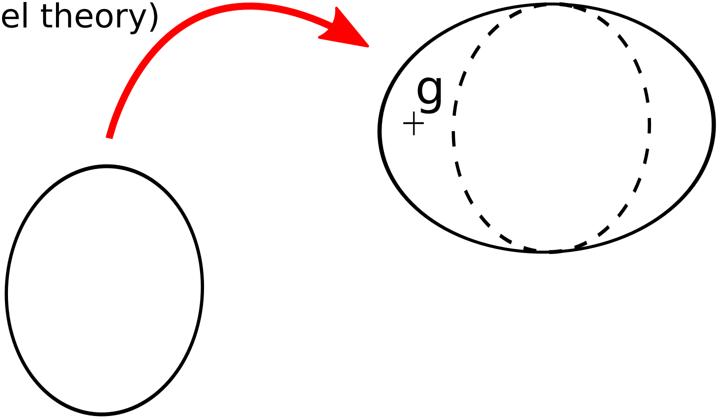
July 22nd, 2016

The question

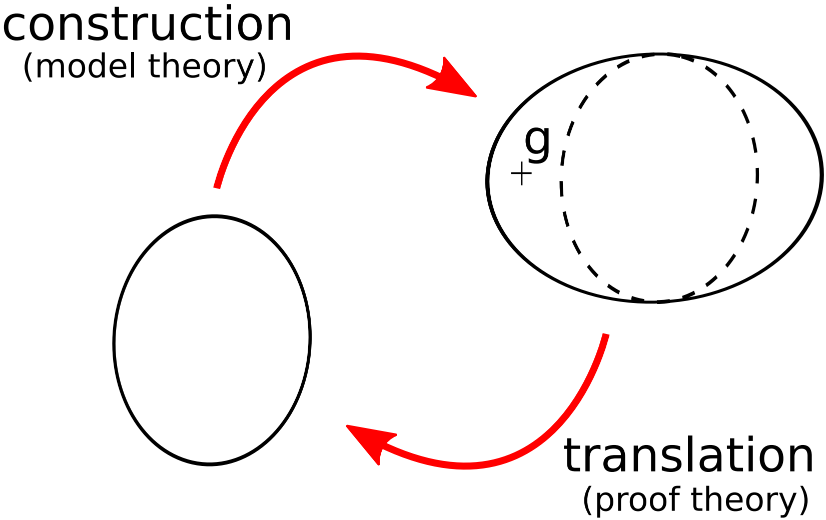
Logic	**Programs**
$\neg\neg$-translation	CPS translation
$\rightsquigarrow$ formula $\bot$	$\rightsquigarrow$ return type
Forcing $\rightsquigarrow$ forcing conditions $\rightsquigarrow$ forcing transformation	???

Forcing in one drawing

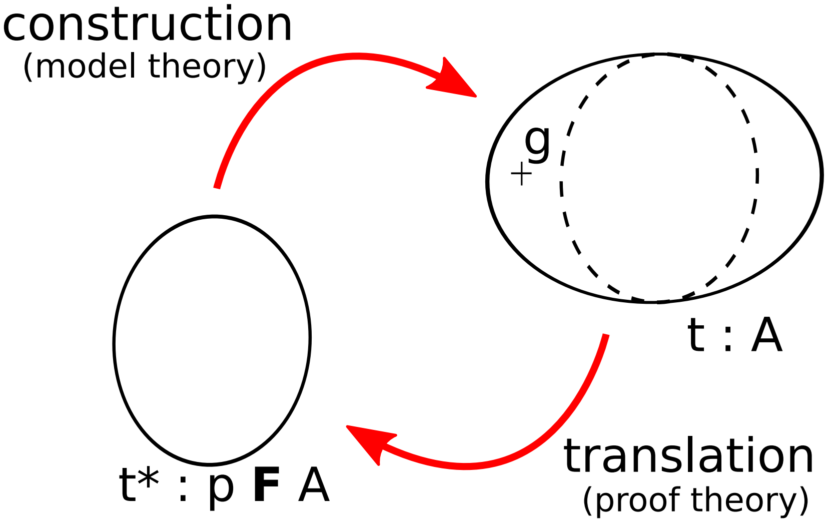
construction
(model theory)



Forcing in one drawing



Forcing in one drawing



Outline

1. Formal proof system: PAω^+
2. Forcing in PAω^+
3. An example of computation by forcing

PAω^+: syntax

Sorts

$$\tau, \sigma \quad := \quad \iota \quad | \quad o \quad | \quad \tau \to \sigma$$

Expressions

$$\begin{array}{rcll} M, N, A, B & := & x^\tau \ | \ \lambda x^\tau.\, M \ | \ M\,N & \lambda\text{-calculus} \\ & | & 0 \ | \ S \ | \ \mathrm{rec}_\tau & \text{arithmetic} \\ & | & A \Rightarrow B \ | \ \forall x^\tau.\, A & \text{minimal logic} \end{array}$$

Proof-terms

$$t, u \quad := \quad x \quad | \quad \lambda x.\, t \quad | \quad t\,u \quad | \quad \mathrm{callcc}$$

PAω^+: Logical connectives

Second-order encodings:

$$
\begin{array}{lcl}
\bot & := & \forall Z.\, Z \\
\neg A & := & A \Rightarrow \bot \\
A \wedge B & := & \forall Z.\, (A \Rightarrow B \Rightarrow Z) \Rightarrow Z \\
A \vee B & := & \forall Z.\, (A \Rightarrow Z) \Rightarrow (B \Rightarrow Z) \Rightarrow Z \\
\exists x.\, A & := & \forall Z.\, (\forall x.\, A \Rightarrow Z) \Rightarrow Z \\
e_1 = e_2 & := & \forall Z.\, Z\, e_1 \Rightarrow Z\, e_2
\end{array}
$$

Notations: $x \in P := P(x)$ $\quad$ $\forall x \in P.\, A := \forall x.\, x \in P \Rightarrow A$

$\exists x \in P.\, A := \exists x.\, x \in P \wedge A$

PAω^+: syntax

Sorts

$$\tau, \sigma \quad := \quad \iota \quad | \quad o \quad | \quad \tau \to \sigma$$

Expressions

$$\begin{array}{rcl} M, N, A, B & := & x^\tau \quad | \quad \lambda x^\tau.\, M \quad | \quad M\, N \\ & | & 0 \quad | \quad S \quad | \quad \mathrm{rec}_\tau \\ & | & A \Rightarrow B \quad | \quad \forall x^\tau.\, A \quad | \quad M \doteq_\tau N \hookrightarrow A \end{array}$$

Proof-terms

$$t, u \quad := \quad x \quad | \quad \lambda x.\, t \quad | \quad t\, u \quad | \quad \mathrm{callcc}$$

$$M \doteq_\tau N \hookrightarrow A \iff M = N \Rightarrow A$$

+ some congruence on formulas

PAω^+: proof system

$$\text{Axiom}\ \frac{}{\mathcal{E}; \Gamma, x : A \vdash x : A} \qquad \frac{}{\mathcal{E}; \Gamma \vdash \text{callcc} : ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}\ \text{Peirce}$$

$$\text{Congruence}\ \frac{\mathcal{E}; \Gamma \vdash t : A}{\mathcal{E}; \Gamma \vdash t : A'}\ A \approx_{\mathcal{E}} A'$$

$$\Rightarrow_i \frac{\mathcal{E}; \Gamma, x : A \vdash t : B}{\mathcal{E}; \Gamma \vdash \lambda x.\, t : A \Rightarrow B} \qquad \frac{\mathcal{E}; \Gamma \vdash t : A \Rightarrow B \quad \mathcal{E}; \Gamma \vdash u : A}{\mathcal{E}; \Gamma \vdash t\, u : B} \Rightarrow_e$$

$$\forall_i \frac{\mathcal{E}; \Gamma \vdash t : A}{\mathcal{E}; \Gamma \vdash t : \forall x^\tau.\, A}\ x \notin FV(\Gamma, \mathcal{E}) \qquad \frac{\mathcal{E}; \Gamma \vdash t : \forall x^\tau.\, A}{\mathcal{E}; \Gamma \vdash t : A[N^\tau / x^\tau]}\ \forall_e$$

$$\hookrightarrow_i \frac{\mathcal{E}, M = N; \Gamma \vdash t : A}{\mathcal{E}; \Gamma \vdash t : M \doteq_\tau N \hookrightarrow A} \qquad \frac{\mathcal{E}; \Gamma \vdash t : M \doteq_\tau M \hookrightarrow A}{\mathcal{E}; \Gamma \vdash t : A} \hookrightarrow_e$$

Classical realizability semantics

- Different from intuitionistic realizability
  - intuitionistic: limits proofs, full extraction
  - classical: full proofs, limits extraction

Classical realizability semantics

- Different from intuitionistic realizability
  - intuitionistic: limits proofs, full extraction
  - classical: full proofs, limits extraction
- The KAM (Krivine's Abstract Machine)
  Stack machine for λ-calculus + callcc

Classical realizability semantics

- Different from intuitionistic realizability
  - intuitionistic: limits proofs, full extraction
  - classical: full proofs, limits extraction
- The KAM (Krivine's Abstract Machine)
  Stack machine for λ-calculus + callcc
- Realizability interpretation
  - Based on a pole $\perp\!\!\!\perp$ (set of processes of the KAM)
  - Propositions interpreted by stacks (refutations)
  - Realizers defined by orthogonality: $|A| := [\![A]\!]^{\perp\!\!\!\perp}$

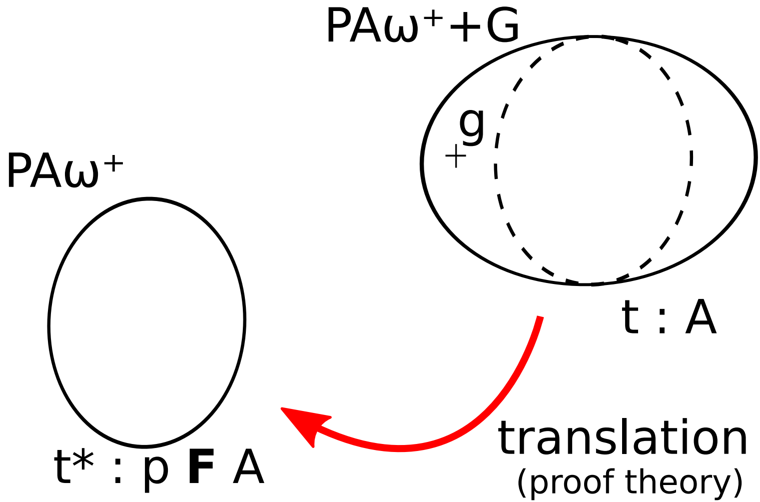
Classical realizability semantics

- Different from intuitionistic realizability
  - intuitionistic: limits proofs, full extraction
  - classical: full proofs, limits extraction
- The KAM (Krivine's Abstract Machine)
  Stack machine for λ-calculus + callcc
- Realizability interpretation
  - Based on a pole $\perp\!\!\!\perp$ (set of processes of the KAM)
  - Propositions interpreted by stacks (refutations)
  - Realizers defined by orthogonality: $|A| := [\![A]\!]^{\perp\!\!\!\perp}$
- Results:
  - Adequacy: $\vdash t : A$ implies $t \Vdash A$
  - Logical consistency: when $\perp\!\!\!\perp = \emptyset$, Tarski model
  - Simple methods to extract witnesses for Σ^0_1 formulas

Outline

1. Formal proof system: PAω^+
2. Forcing in PAω^+
3. An example of computation by forcing

Forcing: overall idea



Forcing: input

Definition (Forcing structure)

A forcing structure is given by

- a sort κ of forcing conditions
- a predicate $C^{\kappa \to o}$ of well-formed conditions ($p \in C$ written $C[p]$)
- a product operation $\cdot$ on forcing conditions
- a maximal condition 1
- a bunch of proof terms $\alpha_0, \ldots, \alpha_8$

G = generic filter on the set of forcing conditions
= "approximations of g"

$g = \bigcup G$

Forcing: input (example)

Example (Forcing structure)

The forcing structure to add a single Cohen real

- $\kappa := \iota$ (finite relations between $\mathbb{N}$ and Bool)
- $C[p] :=$ "p is functional" $\qquad (p : \mathbb{N} \rightharpoonup \text{Bool})$
- $p \cdot q := p \cup q$
- $1 := \varnothing$
- $\alpha_0, \ldots, \alpha_8$

$G :=$ pair-wise compatible finite functions from $\mathbb{N}$ to Bool
$\quad =$ "approximations of g"

$g = \bigcup G \qquad$ (a full function from $\mathbb{N}$ to Bool)

Forcing: output

3 translations $(\_)^*$:

Forcing: output

3 translations $(\_)^*$:

- on kinds:

$$\iota^* := \iota \qquad o^* := \kappa \to o \qquad (\sigma \to \tau)^* := \sigma^* \to \tau^*$$

Forcing: output

3 translations $(\_)^*$:

- on kinds:

$$\iota^* := \iota \qquad o^* := \kappa \to o \qquad (\sigma \to \tau)^* := \sigma^* \to \tau^*$$

- on expressions:
  - $(A \Rightarrow B)^*\, p := \forall q \forall r.\, p \doteq q \cdot r \hookrightarrow (\forall s.\, C[q \cdot s] \Rightarrow A^*\, s) \Rightarrow B^*\, r$
  - merely propagates through other constructions

Forcing: output

3 translations $(\_)^*$:

- on kinds:

$$\iota^* := \iota \qquad o^* := \kappa \to o \qquad (\sigma \to \tau)^* := \sigma^* \to \tau^*$$

- on expressions:
  - $(A \Rightarrow B)^*\, p := \forall q \forall r.\, p \doteq q \cdot r \hookrightarrow (\forall s.\, C[q \cdot s] \Rightarrow A^*\, s) \Rightarrow B^*\, r$
  - merely propagates through other constructions

The forcing transformation: $p \mathrel{F} A := \forall r.\, C[p \cdot r] \Rightarrow A^*\, r$

Forcing: output

3 translations $(\_)^*$:

- on kinds:

$$\iota^* := \iota \qquad o^* := \kappa \to o \qquad (\sigma \to \tau)^* := \sigma^* \to \tau^*$$

- on expressions:
  - $(A \Rightarrow B)^*\, p := \forall q \forall r.\, p \doteq q \cdot r \hookrightarrow (\forall s.\, C[q \cdot s] \Rightarrow A^*\, s) \Rightarrow B^*\, r$
  - merely propagates through other constructions

  The forcing transformation: $p \mathrel{F} A := \forall r.\, C[p \cdot r] \Rightarrow A^*\, r$

- on proof terms:

$$\begin{array}{ll} x^* & := x \\ (t\,u)^* & := \gamma_3\, t^*\, u^* \\ (\lambda x.\, t)^* & := \gamma_1 (\lambda x.\, t^*[(\beta_3 y)/y][(\beta_4 x)/x]) \qquad y \neq x \\ \text{callcc}^* & := \lambda cx.\ \text{callcc}\,(\lambda k.\, x\,(\alpha_{14}\, c)\,(\gamma_4\, k)) \end{array}$$

The KFAM: regular mode

Like the KAM

terms	t, u	$:=$	$x \mid \lambda x.\, t \mid t\, u \mid \text{callcc} \mid \ldots$
environments	e	$:=$	$\emptyset \mid e, x \leftarrow c$
closures	c	$:=$	$t[e] \mid k_\pi$
stacks	π	$:=$	$\alpha \mid c \cdot \pi$
processes	p	$:=$	$c \star \pi$

Skip	$x[e, y \leftarrow c]$	$\star$	π	$\succ$	$x[e]$	$\star$	π
Access	$x[e, x \leftarrow c]$	$\star$	π	$\succ$	c	$\star$	π
Push	$(t\, u)[e]$	$\star$	π	$\succ$	$t[e]$	$\star$	$u[e] \cdot \pi$
Grab	$(\lambda x.\, t)[e]$	$\star$	$c \cdot \pi$	$\succ$	$t[e, x \leftarrow c]$	$\star$	π
Save	$\text{callcc}[e]$	$\star$	$c \cdot \pi$	$\succ$	c	$\star$	$k_\pi \cdot \pi$
Restore	$k_{\pi'}$	$\star$	$c \cdot \pi$	$\succ$	c	$\star$	π'

The KFAM: regular mode

Like the KAM + forcing

terms	t, u	$:=$	$x \mid \lambda x.\, t \mid t\, u \mid \text{callcc} \mid \ldots$
environments	e	$:=$	$\varnothing \mid e, x \leftarrow c$
closures	c	$:=$	$t[e] \mid k_\pi \mid t[e]^* \mid k_\pi^*$
stacks	π	$:=$	$\alpha \mid c \cdot \pi$
processes	p	$:=$	$c \star \pi$

Skip	$x[e, y \leftarrow c]$	$\star\ \pi$	$\succ$	$x[e]$	$\star$	π
Access	$x[e, x \leftarrow c]$	$\star\ \pi$	$\succ$	c	$\star$	π
Push	$(t\, u)[e]$	$\star\ \pi$	$\succ$	$t[e]$	$\star$	$u[e] \cdot \pi$
Grab	$(\lambda x.\, t)[e]$	$\star\ c \cdot \pi$	$\succ$	$t[e, x \leftarrow c]$	$\star$	π
Save	$\text{callcc}[e]$	$\star\ c \cdot \pi$	$\succ$	c	$\star$	$k_\pi \cdot \pi$
Restore	$k_{\pi'}$	$\star\ c \cdot \pi$	$\succ$	c	$\star$	π'

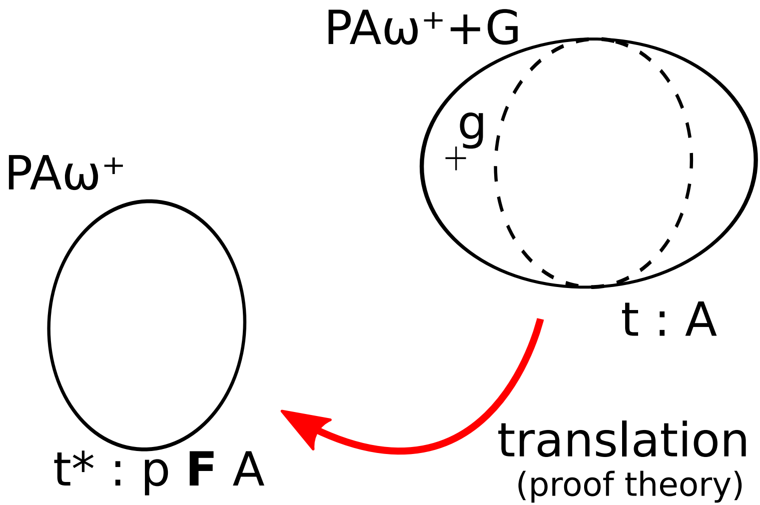
The KFAM: evaluation

Skip	$x[e, y \leftarrow c]$	$\star$	π	$\succ$	$x[e]$	$\star$	π
Access	$x[e, x \leftarrow c]$	$\star$	π	$\succ$	c	$\star$	π
Push	$(t\,u)[e]$	$\star$	π	$\succ$	$t[e]$	$\star$	$u[e] \cdot \pi$
Grab	$(\lambda x.\, t)[e]$	$\star$	$c \cdot \pi$	$\succ$	$t[e, x \leftarrow c]$	$\star$	π
Save	$\mathrm{callcc}[e]$	$\star$	$c \cdot \pi$	$\succ$	c	$\star$	$k_\pi \cdot \pi$
Restore	$k_{\pi'}$	$\star$	$c \cdot \pi$	$\succ$	c	$\star$	π'

$\Uparrow \qquad \Downarrow$

Skip*	$x[e, y \leftarrow c]^*$	$\star$	$f\cdot\ \pi$	$\succ$	$x[e]^*$	$\star$	$\alpha_9\ f\cdot$	π
Access*	$x[e, x \leftarrow c]^*$	$\star$	$f\cdot\ \pi$	$\succ$	c	$\star$	$\alpha_{10}\ f\cdot$	π
Push*	$(t\,u)[e]^*$	$\star$	$f\cdot\ \pi$	$\succ$	$t[e]^*$	$\star$	$\alpha_{11}\ f\cdot$	$u[e]^* \cdot \pi$
Grab*	$(\lambda x.\, t)[e]^*$	$\star$	$f\cdot c \cdot \pi$	$\succ$	$t[e, x \leftarrow c]^*$	$\star$	$\alpha_6\ f\cdot$	π
Save*	callcc^*	$\star$	$f\cdot c \cdot \pi$	$\succ$	c	$\star$	$\alpha_{14}\ f\cdot$	$k^*_\pi \cdot \pi$
Restore*	$k^*_{\pi'}$	$\star$	$f\cdot c \cdot \pi$	$\succ$	c	$\star$	$\alpha_{15}\ f\cdot$	π'

Forcing: overall idea



Forcing: extension to the generic filter

Restriction: C is invariant by forcing (arithmetical)

Forcing: extension to the generic filter

Restriction: C is invariant by forcing (arithmetical)

PAω^+ + G	$\longrightarrow$	Forcing translation	$\longrightarrow$	PAω^+
A				$p \, F \, A$
$t : A$				$t^* : p \, F \, A$
$q \in G$				??

Forcing: extension to the generic filter

Restriction: C is invariant by forcing (arithmetical)

PA$\omega^+ + G$	$\longrightarrow$	Forcing translation	$\longrightarrow$	PAω^+
A				$p \mathrel{F} A$
$t : A$				$t^* : p \mathrel{F} A$
$q \in G$				$p \leqslant q$

$$p \mathrel{F} q \in G \quad \equiv \quad p \leqslant q \quad := \quad \forall r.\, C[p \cdot r] \Rightarrow C[q \cdot r]$$

Forcing: extension to the generic filter

Restriction: C is invariant by forcing (arithmetical)

PAω^+ + G	$\longrightarrow$	Forcing translation	$\longrightarrow$	PAω^+
A				$p \Vdash A$
$t : A$				$t^* : p \Vdash A$
$q \in G$				$p \leqslant q$

$$p \Vdash q \in G \quad \equiv \quad p \leqslant q \quad := \quad \forall r.\, C[p \cdot r] \Rightarrow C[q \cdot r]$$

Nice properties of G in the forcing universe:

- non empty $\quad 1 \in G$
- subset of C $\quad \forall p \in G.\, C[p]$
- filter $\quad \forall p \forall q.\, (p \cdot q) \in G \Rightarrow p \in G$
  $\quad \forall p \in G.\, \forall q \in G.\, (p \cdot q) \in G$
- genericity $\quad \ldots$

We need to prove that they are forced

forcing/kernel modes

Forcing usage : the big picture

We want to prove $\dfrac{A_1 \quad \dots \quad A_n}{A}$.

Base universe | Forcing universe

Forcing usage : the big picture

We want to prove $\dfrac{A_1 \qquad \ldots \qquad A_n}{A}$.

Base universe	Forcing universe
1. Build the forcing structure	

Forcing usage : the big picture

We want to prove $\dfrac{A_1 \qquad \ldots \qquad A_n}{A}$.

Base universe

1. Build the forcing structure
2. Assume the premises $x_1 \ldots x_n$

Forcing universe

Forcing usage : the big picture

We want to prove $\dfrac{A_1 \quad \dots \quad A_n}{A}$.

Base universe

1. Build the forcing structure
2. Assume the premises $x_1 \dots x_n$

Forcing universe

3. Lift the premises $x_1 \dots x_n$

Forcing usage : the big picture

We want to prove $\dfrac{A_1 \qquad \dots \qquad A_n}{A}$.

Base universe

1. Build the forcing structure
2. Assume the premises $x_1 \dots x_n$

Forcing universe

3. Lift the premises $x_1 \dots x_n$
4. Make the proof (using g/G)
   $t(x_1, \dots, x_n) : A$

Forcing usage : the big picture

We want to prove $\dfrac{A_1 \quad \dots \quad A_n}{A}$.

Base universe

1. Build the forcing structure
2. Assume the premises $x_1 \dots x_n$

Forcing universe

3. Lift the premises $x_1 \dots x_n$
4. Make the proof (using g/G) $t(x_1, \dots, x_n) : A$

Base universe

5. Use the forcing translation $t^*(x_1^*, \dots, x_n^*) : 1 \ F \ A$

Forcing usage : the big picture

We want to prove $\dfrac{A_1 \quad \dots \quad A_n}{A}$.

Base universe

1. Build the forcing structure
2. Assume the premises $x_1 \dots x_n$

Forcing universe

3. Lift the premises $x_1 \dots x_n$
4. Make the proof (using g/G) $t(x_1, \dots, x_n) : A$

Base universe

5. Use the forcing translation $t^*(x_1^*, \dots, x_n^*) : 1 \; F \; A$
6. Remove forcing $w\, t^*(x_1^*, \dots, x_n^*) : A$

Forcing usage : the big picture

We want to prove $\dfrac{A_1 \quad \dots \quad A_n}{A}$.

Base universe

1. Build the forcing structure
2. Assume the premises $x_1 \dots x_n$

Forcing universe

3. Lift the premises $x_1 \dots x_n$
4. Make the proof (using g/G) $t(x_1, \dots, x_n) : A$

Base universe

5. Use the forcing translation $t^*(x_1^*, \dots, x_n^*) : 1 \; F \; A$
6. Remove forcing $w \, t^*(x_1^*, \dots, x_n^*) : A$
7. Extract a witness (classical realizability)

Outline

1. Formal proof system: PAω^+
2. Forcing in PAω^+
3. An example of computation by forcing

Disjunction property and Herbrand's theorem

Disjunction property (intuitionistic logic)

If $\exists\vec{x}.\,F(\vec{x})$ is provable,
then there exists a closed term $\vec{t}$
such that $F(\vec{t})$ is provable.

Herbrand's theorem (classical logic)

If $\exists\vec{x}.\,F(\vec{x})$ is provable,
then there exists closed terms $\vec{t}_1, \ldots, \vec{t}_k$
such that $F(\vec{t}_1) \vee \ldots \vee F(\vec{t}_k)$ is provable.

Disjunction property and Herbrand's theorem

Disjunction property (intuitionistic logic)

If $\exists \vec{x}.\, F(\vec{x})$ is provable,
then there exists a closed term $\vec{t}$
such that $F(\vec{t})$ is provable.

Herbrand's theorem (classical logic)

If $\exists \vec{x}.\, F(\vec{x})$ is provable,
then there exists closed terms $\vec{t}_1, \ldots, \vec{t}_k$
such that $F(\vec{t}_1) \vee \ldots \vee F(\vec{t}_k)$ is provable.

To which model correspond each witness?

Herbrand trees

Definition (Herbrand tree)

A *Herbrand tree* is a finite binary tree such that:

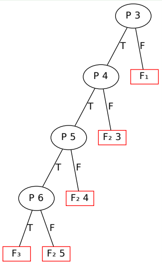
- inner nodes = atomic formulas
- leaves = witnesses $\vec{t}$

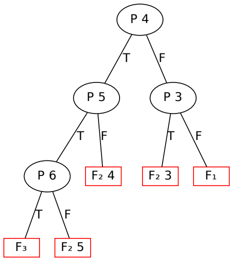
branch = partial valuation

Example

$F\,n := F_1 \vee F_2 \vee F_3$

- $F_1 := \neg P\,3$
- $F_2 := P\,n \wedge \neg P\,(n+1)$
- $F_3 := P\,6$





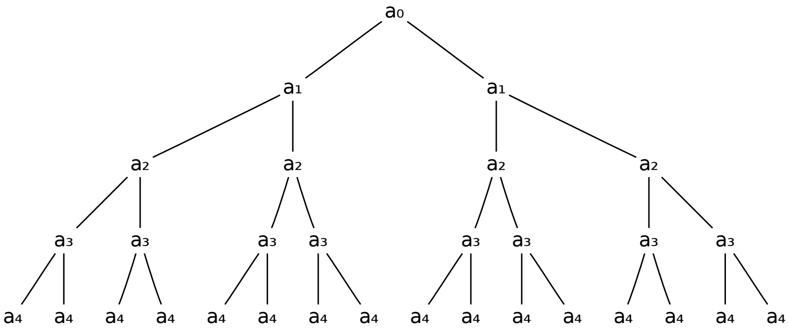
Build Herbrand trees by a proof of Herbrand's theorem

If $\exists\vec{x}.\, F(\vec{x})$ is provable,
then there exists closed terms $\vec{t}_1, \ldots, \vec{t}_k$
such that $F(\vec{t}_1) \vee \ldots \vee F(\vec{t}_k)$ is provable.

Let us fix an enumeration $(a_i)_{i\in\mathbb{N}}$ of the atoms.
(atoms = closed atomic formulas)

Build Herbrand trees by a proof of Herbrand's theorem

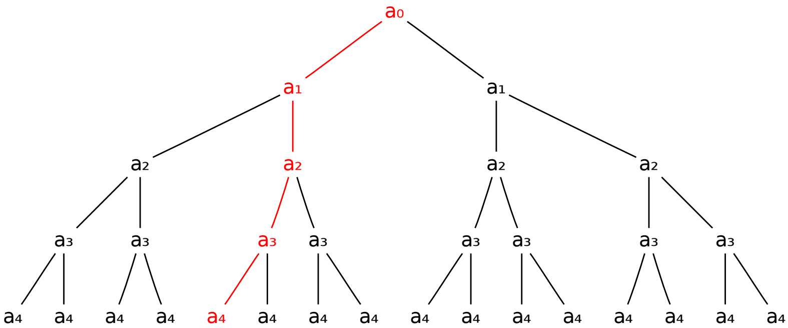
If $\exists \vec{x}.\, F(\vec{x})$ is provable,
then there exists closed terms $\vec{t}_1, \ldots, \vec{t}_k$
such that $F(\vec{t}_1) \vee \ldots \vee F(\vec{t}_k)$ is provable.



consider the atom-enumerating complete infinite tree

Build Herbrand trees by a proof of Herbrand's theorem

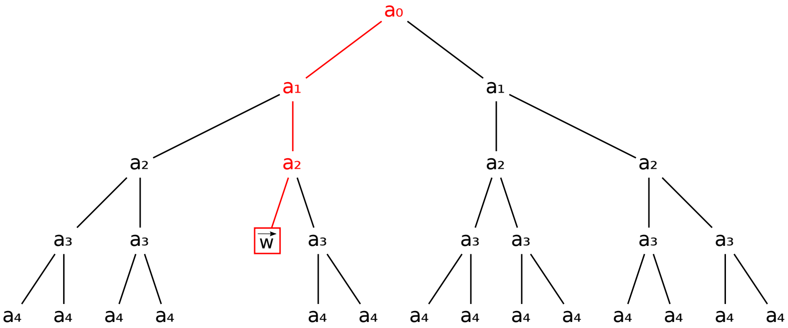
If $\exists\vec{x}.\, F(\vec{x})$ is provable,
then there exists closed terms $\vec{t}_1, \ldots, \vec{t}_k$
such that $F(\vec{t}_1) \vee \ldots \vee F(\vec{t}_k)$ is provable.



pick any infinite branch

Build Herbrand trees by a proof of Herbrand's theorem

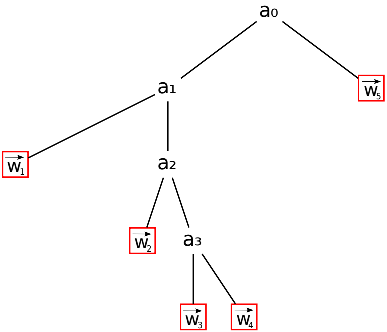
If $\exists \vec{x}.\, F(\vec{x})$ is provable,
then there exists closed terms $\vec{t}_1, \ldots, \vec{t}_k$
such that $F(\vec{t}_1) \vee \ldots \vee F(\vec{t}_k)$ is provable.



by hypothesis (and $F(\vec{w})$ finite), we can cut it at finite depth

Build Herbrand trees by a proof of Herbrand's theorem

If $\exists\vec{x}.\, F(\vec{x})$ is provable,
then there exists closed terms $\vec{t}_1, \ldots, \vec{t}_k$
such that $F(\vec{t}_1) \vee \ldots \vee F(\vec{t}_k)$ is provable.



conclude using the fan theorem

The interest of forcing here

- forcing takes care of the tree structure
  only perform the proof on the generic branch
- no need to give *a priori* an order on atoms

g is here a generic model *i.e.* a generic branch

The interest of forcing here

- forcing takes care of the tree structure
  only perform the proof on the generic branch
- no need to give *a priori* an order on atoms

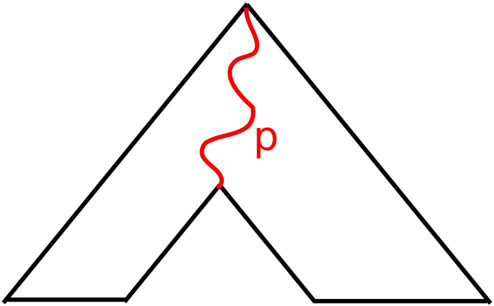
g is here a generic model *i.e.* a generic branch

Our forcing structure: 1 specific Cohen real

$$\begin{aligned} \text{forcing conditions} &:= \text{finite functions from atoms to bool} \\ \kappa &:= \iota \\ C[p] &:= (p : \mathrm{Atom} \rightharpoonup \mathrm{Bool}) \wedge k \\ p \cdot q &:= p \cup q \\ 1 &:= \varnothing \\ G &= \text{pairwise compatible conditions} \end{aligned}$$

The computational content of forcing conditions

$$C[p] := p : \text{Atom} \rightharpoonup \text{Bool} \wedge k$$



Key ingredients of the forcing proof

1. Forcing structure:
   $\leadsto$ contains the Herbrand tree under construction

4. Proof in the forcing universe:
   - uses only one model: g
   - uses the (classical) proof of $\exists \vec{x}.\, F(\vec{x})$
   - uses the axioms about g: specifically the genericity axiom

Key ingredients of the forcing proof

1. Forcing structure:
   $\leadsto$ contains the Herbrand tree under construction

4. Proof in the forcing universe:
   - uses only one model: g
   - uses the (classical) proof of $\exists \vec{x}.\, F(\vec{x})$
   - uses the axioms about g: specifically the genericity axiom
     $\leadsto$ actually a weaker form: the totality of g

$$\text{(A)} \qquad \forall a \in \text{Atom}.\, \exists q \in G.\, \exists b \in \text{Bool}.\, q(a) = b$$

5. Realize the axiom A

The totality axiom

Used instead of genericity

$p \Vdash \forall a \in \text{Atom}.\, \exists q \in G.\, \exists b \in \text{Bool}.\, q(a) = b$

The totality axiom

Used instead of genericity

$p \ F \ \forall a \in \text{Atom}. \exists q \in G. \exists b \in \text{Bool}. \, q(a) = b$

2 cases:

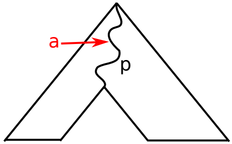
The totality axiom

Used instead of genericity

$p \Vdash \forall a \in \text{Atom}. \exists q \in G. \exists b \in \text{Bool}. q(a) = b$

2 cases:

- $a \in p$: answer b as in p



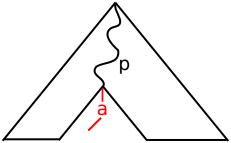
The totality axiom

Used instead of genericity

$p \mathrel{F} \forall a \in \text{Atom}.\, \exists q \in G.\, \exists b \in \text{Bool}.\, q(a) = b$

2 cases:

- $a \in p$: answer b as in p
- $a \notin p$: we try both true and false



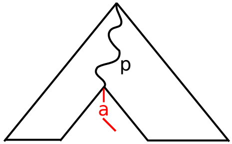
The totality axiom

Used instead of genericity

$p \; F \; \forall a \in \text{Atom}.\, \exists q \in G.\, \exists b \in \text{Bool}.\, q(a) = b$

2 cases:

- $a \in p$: answer b as in p
- $a \notin p$: we try both true and false



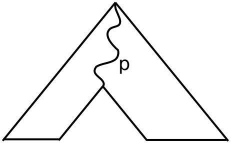
The totality axiom

Used instead of genericity

$p\ F\ \forall a \in \mathrm{Atom}.\, \exists q \in G.\, \exists b \in \mathrm{Bool}.\, q(a) = b$

2 cases:

- $a \in p$: answer b as in p
- $a \notin p$: we try both true and false



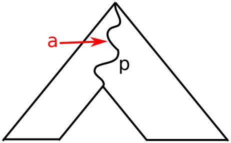
The program realizing the totality axiom

$$
\begin{array}{l}
\lambda caf.\ \mathrm{let}\ p, t := \alpha\, c\ \mathrm{in} \\
\quad \mathrm{if}\ \ \mathrm{Tot}_{\mathrm{test}}\ a'\ \mathrm{true}\ p\ \mathrm{then}\ f\,(\alpha\, c)\ \mathrm{I}\ \mathrm{true}^*\ \ \mathrm{I}^*\ \ \mathrm{else} \\
\quad \mathrm{if}\ \ \mathrm{Tot}_{\mathrm{test}}\ a'\ \mathrm{false}\ p\ \mathrm{then}\ f\,(\alpha\, c)\ \mathrm{I}\ \mathrm{false}^*\ \ \mathrm{I}^*\ \ \mathrm{else} \\
\quad f\,\langle \mathrm{Up}_{\mathrm{FVal}}\,((a')^+ \cup p), \lambda u. \\
\quad\quad f\,\langle \mathrm{Up}_{\mathrm{FVal}}\,((a')^- \cup p), \lambda v. \\
\quad\quad\quad t\,(\mathrm{merge}\, a'\, u\, v)\rangle\ \mathrm{I}\ \mathrm{false}^*\ \ \mathrm{I}^*\rangle\ \mathrm{I}\ \mathrm{true}^*\ \ \mathrm{I}^*
\end{array}
$$

The totality axiom

Used instead of genericity

$p\ F\ \forall a \in \text{Atom}.\ \exists q \in G.\ \exists b \in \text{Bool}.\ q(a) = b$



2 cases:

- $a \in p$: answer b as in p
- $a \notin p$: we try both true and false

The program realizing the totality axiom

$\lambda caf.$ let $p, t := \alpha\, c$ in
if $\text{Tot}_{\text{test}}\ a'$ true p then $f\,(\alpha\, c)$ I true* I* else
if $\text{Tot}_{\text{test}}\ a'$ false p then $f\,(\alpha\, c)$ I false* I* else
$f\,\langle \text{Up}_{\text{FVal}}\,((a')^{+} \cup p), \lambda u.$
$f\,\langle \text{Up}_{\text{FVal}}\,((a')^{-} \cup p), \lambda v.$
$t\,(\text{merge}\ a'\ u\ v)\rangle$ I false* I*$\rangle$ I true* I*

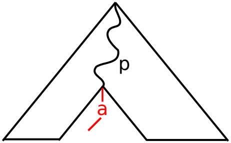
The totality axiom

Used instead of genericity

$p \; F \; \forall a \in \text{Atom}.\, \exists q \in G.\, \exists b \in \text{Bool}.\, q(a) = b$

2 cases:

- $a \in p$: answer b as in p
- $a \notin p$: we try both true and false



The program realizing the totality axiom

$$
\begin{array}{l}
\lambda caf.\ \text{let } p, t := \alpha\, c \text{ in} \\
\quad \text{if } \text{Tot}_{\text{test}}\ a'\ \text{true}\ p \text{ then } f\,(\alpha\, c)\ \text{I}\ \text{true}^*\ \text{I}^*\ \text{ else} \\
\quad \text{if } \text{Tot}_{\text{test}}\ a'\ \text{false}\ p \text{ then } f\,(\alpha\, c)\ \text{I}\ \text{false}^*\ \text{I}^*\ \text{ else} \\
\quad f\,\langle \text{Up}_{\text{FVal}}\,((a')^+ \cup p), \lambda u. \\
\qquad f\,\langle \text{Up}_{\text{FVal}}\,((a')^- \cup p), \lambda v. \\
\qquad\quad t\,(\text{merge}\ a'\ u\ v)\rangle\ \text{I}\ \text{false}^*\ \text{I}^*\rangle\ \text{I}\ \text{true}^*\ \text{I}^*
\end{array}
$$

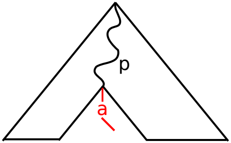
The totality axiom

Used instead of genericity

$p \; F \; \forall a \in \text{Atom}.\, \exists q \in G.\, \exists b \in \text{Bool}.\, q(a) = b$

2 cases:

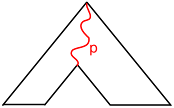
- $a \in p$: answer b as in p
- $a \notin p$: we try both true and false



The program realizing the totality axiom

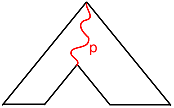
$$
\begin{array}{l}
\lambda caf.\ \text{let } p, t := \alpha\, c \text{ in} \\
\quad \text{if } \ \text{Tot}_{\text{test}}\ a'\ \text{true}\ p \text{ then } f\,(\alpha\, c)\ \text{I}\ \text{true}^*\ \ \text{I}^*\ \text{ else} \\
\quad \text{if } \ \text{Tot}_{\text{test}}\ a'\ \text{false}\ p \text{ then } f\,(\alpha\, c)\ \text{I}\ \text{false}^*\ \ \text{I}^*\ \text{ else} \\
\quad f\,\langle \text{Up}_{\text{FVal}}\,((a')^+ \cup p), \lambda u. \\
\quad\quad f\,\langle \text{Up}_{\text{FVal}}\,((a')^- \cup p), \lambda v. \\
\quad\quad\quad t\,(\text{merge}\ a'\ u\ v)\rangle\ \text{I}\ \text{false}^*\ \ \text{I}^*\rangle\ \text{I}\ \text{true}^*\ \ \text{I}^*
\end{array}
$$

More insight on the computational content



- Realizer of $C[p]$: zipper with hole
- Proof in the forcing universe
  - gives a user-level program
    $\rightsquigarrow$ no direct access to the forcing condition
  - access to the tree is provided by the axioms on G (mostly A)
- Realizer of A performs the extension of the tree + querrying
  No erasing of the tree (even with backtrack in the forcing proof)
- G is a "moving set" and g a "moving branch"

More insight on the computational content



- Realizer of $C[p]$: zipper with hole
- Proof in the forcing universe
  - gives a user-level program
    $\rightsquigarrow$ no direct access to the forcing condition
  - access to the tree is provided by the axioms on G (mostly A)
- Realizer of A performs the extension of the tree + querrying
  No erasing of the tree (even with backtrack in the forcing proof)
- G is a “moving set” and g a “moving branch”

We can put datatypes inside $C[p]$

Conclusion

- Practical method for extracting proofs using forcing
- Extend Curry-Howard correspondence

Logic	**Programs**
forcing transformation	add a memory cell
forcing conditions	value of the memory cell
axioms on G	instructions on the memory cell
new object g	"meaning" of the memory cell

- One example (Herbrand) where forcing "=" tree library
- More generally: forcing performs an abstraction barrier
- Very efficient: datatypes

Conclusion

- Practical method for extracting proofs using forcing
- Extend Curry-Howard correspondence

Logic	**Programs**
forcing transformation	add a memory cell
forcing conditions	value of the memory cell
axioms on G	instructions on the memory cell
new object g	"meaning" of the memory cell

- One example (Herbrand) where forcing "=" tree library
- More generally: forcing performs an abstraction barrier
- Very efficient: datatypes

The End

Conclusion

- Practical method for extracting proofs using forcing
- Extend Curry-Howard correspondence

Logic	**Programs**
forcing transformation	add a memory cell
forcing conditions	value of the memory cell
axioms on G	instructions on the memory cell
new object g	“meaning” of the memory cell

- One example (Herbrand) where forcing “=” tree library
- More generally: forcing performs an abstraction barrier
- Very efficient: datatypes

The End

PAω^+: congruence

Reflexivity, symmetry, transitivity and base case

$$\frac{}{M \approx_{\mathcal{E}} M} \qquad \frac{M \approx_{\mathcal{E}} N}{N \approx_{\mathcal{E}} M} \qquad \frac{M \approx_{\mathcal{E}} N \qquad N \approx_{\mathcal{E}} P}{M \approx_{\mathcal{E}} P} \qquad \frac{}{M \approx_{\mathcal{E}} N}\ (M = N) \in \mathcal{E}$$

Context closure

...

$\beta\eta\iota$-conversion

$$\frac{}{(\lambda x^\tau.\, M)\, N^\tau \approx_{\mathcal{E}} M[N^\tau/x^\tau]} \qquad \frac{}{\lambda x.\, M\, x \approx_{\mathcal{E}} M}\ x \notin FV(M)$$

$$\frac{}{\mathrm{rec}_\tau\, M\, N\, 0 \approx_{\mathcal{E}} M} \qquad \frac{}{\mathrm{rec}_\tau\, M\, N\, (S\, P) \approx_{\mathcal{E}} N\, P\, (\mathrm{rec}_\tau\, M\, N\, P)}$$

Semantically equivalent propositions

$$\frac{}{\forall x^\tau \forall y^\sigma.\, A \approx_{\mathcal{E}} \forall y^\sigma \forall x^\tau.\, A} \qquad \frac{}{\forall x^\tau.\, A \approx_{\mathcal{E}} A}\ x \notin FV(A)$$

$$\frac{}{A \Rightarrow \forall x^\tau.\, B \approx_{\mathcal{E}} \forall x^\tau.\, A \Rightarrow B}\ x \notin FV(A)$$

$$\frac{}{M \doteq M \hookrightarrow A \approx_{\mathcal{E}} A} \qquad \frac{}{M \doteq N \hookrightarrow A \approx_{\mathcal{E}} N \doteq M \hookrightarrow A}$$

$$\frac{}{M \doteq N \hookrightarrow P \doteq Q \hookrightarrow A \approx_{\mathcal{E}} P \doteq Q \hookrightarrow M \doteq N \hookrightarrow A}$$

$$\frac{}{A \Rightarrow M \doteq N \hookrightarrow B \approx_{\mathcal{E}} M \doteq N \hookrightarrow A \Rightarrow B}$$

$$\frac{}{\forall x^\tau.\, M \doteq N \hookrightarrow A \approx_{\mathcal{E}} M \doteq N \hookrightarrow \forall x^\tau.\, A}\ x \notin FV(M, N)$$

Classical realizability interpretation

Sorts

$$\llbracket \iota \rrbracket := \mathbb{N}$$
$$\llbracket o \rrbracket := \mathcal{P}(\Pi)$$
$$\llbracket \sigma \to \tau \rrbracket := \llbracket \tau \rrbracket^{\llbracket \sigma \rrbracket}$$

Terms

$$\llbracket x^\tau \rrbracket_\rho := \rho(x)$$
$$\llbracket \lambda x.\, M \rrbracket_\rho := v \mapsto \llbracket M \rrbracket_{\rho, x^\tau \leftarrow v}$$
$$\llbracket MN \rrbracket_\rho := \llbracket M \rrbracket_\rho \, \llbracket N \rrbracket_\rho$$
$$\llbracket 0 \rrbracket_\rho := 0$$
$$\llbracket S \rrbracket_\rho := n \mapsto n + 1$$
$$\llbracket \mathrm{rec}_\tau \rrbracket_\rho := \mathrm{rec}_{\llbracket \tau \rrbracket}$$
$$\llbracket A \Rightarrow B \rrbracket_\rho := \left\{ t \cdot \pi \;\middle|\; t \in |A|_\rho \wedge \pi \in \|B\|_\rho \right\}$$
$$\llbracket \forall x^\tau . A \rrbracket_\rho := \bigcup_{v \in \llbracket \tau \rrbracket} \llbracket A \rrbracket_{\rho, x^\tau \leftarrow v}$$
$$\llbracket M \doteq_\tau N \hookrightarrow A \rrbracket_\rho := \begin{cases} \llbracket A \rrbracket_\rho & \text{if } \llbracket M \rrbracket_\rho = \llbracket N \rrbracket_\rho \\ \varnothing & \text{otherwise} \end{cases}$$

Truth values

$$|A|_\rho := \left\{ t \in \Lambda \;\middle|\; \forall \pi \in \llbracket A \rrbracket_\rho \,.\, t \star \pi \in \perp\!\!\!\perp \right\}$$