Classical realizability and forcing
Part 1: Introduction

Alexandre Miquel

Logic Colloquium (LC'14)
Vienna Summer of Logic – July 17th, 2014 – Vienna

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 an control operators (call/cc)

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

- Initially designed for PA2, but extends to:
  - Higher-order arithmetic (PAω)
  - Zermelo-Fraenkel set theory (ZF)
  - Interprets the Axiom of Dependent Choices (DC)

- Deep connections with Cohen forcing (cf Part 2)

$\leadsto$ can be used to define new models of PA2/ZF (cf Part 3)

Different notions of models

- **Tarski models:** $[\![A]\!] \in \{0;1\}$
  - Interprets classical provability (correctness/completeness)
- **Intuitionistic realizability:** $[\![A]\!] \in \mathfrak{P}(\Lambda)$ [Kleene 45]
  - Interprets intuitionistic proofs
  - Independence results in intuitionistic theories
  - Definitely incompatible with classical logic
- **Cohen forcing:** $[\![A]\!] \in \mathfrak{P}(C)$ [Cohen 63]
  - Independence results, in classical theories
    (Negation of continuum hypothesis, Solovay's axiom, etc.)
- **Classical realizability:** $[\![A]\!] \in \mathfrak{P}(\Lambda_c)$ [Krivine 94, 03]
  - Interprets classical proofs
  - Generalizes Tarski models... and forcing!

The Brouwer-Heyting-Kolmogorov (BHK) semantics

- **Philosophical input:** the meaning of a proposition A is the set $[\![A]\!]$ of evidences that A holds:

$$[\![A \Rightarrow B]\!] = [\![A]\!] \to [\![B]\!] \qquad \text{('computable' functions)}$$

$$[\![A \wedge B]\!] = [\![A]\!] \times [\![B]\!] \qquad \text{(Cartesian product)}$$

$$[\![A \vee B]\!] = [\![A]\!] + [\![B]\!] \qquad \text{(Disjoint union)}$$

$$[\![(\forall x \in \mathbb{N})\, A(x)]\!] = \prod_{n \in \mathbb{N}} [\![A(n)]\!] \qquad \text{(Dependent product)}$$

$$[\![(\exists x \in \mathbb{N})\, A(x)]\!] = \sum_{n \in \mathbb{N}} [\![A(n)]\!] \qquad \text{(Dependent sum)}$$

- **Typical example:** $(\forall x \in \mathbb{N})(\exists y \in \mathbb{N})\ A(x, y)$

From philosophy to mathematics

The BHK philosophical interpretation of propositions can be given a formal (i.e. mathematical) contents: the theory of realizability

$$t \Vdash A$$

- Which notion of an evidence?
  - Gödel codes of recursive functions [Kleene 45]
  - λ-terms
  - Elements of an arbitrary PCA
- For which theory?
  - Heyting Arithmetic (HA)
  - Second/higher-order Heyting Arithmetic (HA2/HAω)
  - Intuitionistic Zermelo-Fraenkel Set theory (IZF) [Myhill-Friedman 73, McCarty 84]

Building realizers from proofs

Theorem

From a derivation d of A, one can effectively extract a realizer $d^* \in [\![A]\!]$

- Works in most intuitionistic theories: HA, HA2, HAω, IZF, etc.
- **Technically:** Read each deduction rule as a typing rule and build the program d^* accordingly:

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

- Relies on the property of

Adequacy: If $\vdash t : A$, then $t \Vdash A$ (i.e. $t \in [\![A]\!]$)

- Axioms are realized separately

Why BHK is incompatible with classical logic

- A simple argument:
  - The following proposition is classically provable:

$$(\forall x \in \mathbb{N})\,(\mathrm{Halt}(x) \vee \neg\mathrm{Halt}(x))$$

  - But a realizer would solve the halting problem!
  - **Remark:** Incompatibility due to the restriction to computable functions. Vanishes if we introduce non computable realizers (oracles)

- Another argument, in 2nd-order logic:
  - The negation of excluded middle is realizable!

$$\neg\forall X\,(X \vee \neg X)$$

  - **Reason:** 2nd-order $\forall$ commutes with $\vee$ in Kleene's realizability

A technical defect of BHK semantics

- In BHK semantics, we have $[\![\bot]\!] = \varnothing$. Hence:

$$[\![\neg A]\!] \;=\; [\![A]\!] \to \varnothing \;=\; \begin{cases} \varnothing & \text{if } [\![A]\!] \neq \varnothing \\ \Lambda & \text{if } [\![A]\!] = \varnothing \end{cases}$$

- **Consequences:**
  - Negated formulas have no computational contents
  - On the fragment formed by all negated formulas, BHK semantics degenerates to a 2-valued model $\rightsquigarrow$ Tarski semantics
  - BHK semantics not suited to be used with $\neg\neg$-translation

How to cope with classical logic?

1. Keep BHK semantics, but compose it with Friedman's translation

$$[\![A]\!]^* \;:=\; [\![A^{\neg\neg R}]\!]$$

   - Translation $A \mapsto A^{\neg\neg R}$ parameterized by a return formula R, uses relative negation $\neg_R A \;:=\; A \Rightarrow R$ instead of negation
   - Useful for witness extraction for Σ^0_1/Π^0_2-formulas (Friedman's trick)
   - Alters the computational meaning of proofs / typing rules

2. Hard-wire Friedman's translation in the semantics

   $\rightsquigarrow$ get a new semantics: Krivine's semantics

Realizability vs. Boolean/Heyting-valued models

- In Boolean/Heyting-valued models, conjunction is interpreted as meet/intersection...
  ... so that universal quantification amounts to an infinitary intersection

- But in proof theory, universal quantification is very different from an infinitary conjunction:

$$\frac{\vdash A(x)}{\vdash \forall x\, A(x)} \qquad \frac{\vdash A(0) \quad \vdash A(1) \quad \cdots \quad \vdash A(41)}{\vdash A(0) \wedge A(1) \wedge \cdots \wedge A(41)}$$

- In intuitionistic/classical realizability
  - Conjunction is interpreted as a Cartesian product
  - Universal quantification (over predicates or sets) is interpreted as an infinitary intersection ($\neq$ infinitary conjunction)

The missing link

	$\wedge = \forall = \cap$	$\wedge = \times, \forall = \cap$
Int. logic	Heyting-valued models	Int. realizability
Class. logic	Boolean-valued models (Cohen forcing)	Class. realizability

A cardinals' heresy in classical realizability (teaser)

- Recall that in ZFC (= ZF + AC), cardinals are well-ordered
  (Since they are represented by ordinals, thanks to Zermelo's Lemma)
- In *Realizability algebras II: new models of ZF + DC* (2012), Krivine presents a classical realizability model of ZF + DC (the model of threads) in which we can find:

(1) An infinite set $S \subseteq \mathbb{R}$ which is not equipotent with $S \times S$

(2) Two infinite sets $S_1, S_2 \subseteq \mathbb{R}$ such that there is no surjection in either direction (and thus no injection in either direction)

(3) An infinite sequence $(S_q)_{q \in \mathbb{Q}}$ of infinite subsets of $\mathbb{R}$ indexed by $\mathbb{Q}$ whose cardinals are strictly increasing (dense ordering!)

If ZF is consistent, then so is ZF + DC + (1) + (2) + (3)

- In Part 3, we shall rephrase Krivine's result in 2nd-order logic

Plan

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

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

- Two kinds of variables
  - 1st-order vars: $x, y, z, \ldots$
  - 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 at least a function symbol for every primitive recursive function (0, s, $+$, $-$, $\times$, $/$, mod, $\uparrow$, $\ldots$), each of them being interpreted the standard way

- 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)

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

Predicates

- Concrete relations are represented using predicates (syntactic sugar)

Predicates $P, Q \quad ::= \quad \hat{x}_1 \cdots \hat{x}_k A_0$ (of arity k)

Definition (Predicate application and 2nd-order substitution)

1. $P(e_1, \ldots, e_k)$ is the formula defined by

$$P(e_1, \ldots, e_k) \;\equiv\; A_0\{x_1 := e_1, \ldots, x_k := e_k\}$$

where $P \equiv \hat{x}_1 \cdots \hat{x}_k A_0$, and where $e_1, \ldots, e_k$ are k first-order terms

2. 2nd-order substitution $A\{X := P\}$ (where X and P are of the same arity k) consists to replace in the formula A every atomic sub-formula of the form

$$X(e_1, \ldots, e_k) \qquad \text{by the formula} \qquad P(e_1, \ldots, e_k)$$

- **Note:** Every k-ary 2nd-order variable X can be seen as a predicate:

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

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:

$$\begin{array}{rcl}(\forall x \in P)\,A(x) & \equiv & \forall x\,(x \in P \Rightarrow A(x)) \\ (\exists x \in P)\,A(x) & \equiv & \forall Z\,(\forall x\,(x \in P \Rightarrow A(x) \Rightarrow Z) \Rightarrow Z) \\ & \Leftrightarrow & \exists x\,(x \in P \wedge A(x))\end{array}$$

- Inclusion and extensional equality:

$$\begin{array}{rcl}P \subseteq Q & \equiv & \forall x\,(x \in P \Rightarrow x \in Q) \\ P = Q & \equiv & \forall x\,(x \in P \Leftrightarrow x \in Q)\end{array}$$

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

A type system for second-order logic (λNK2)

- Represent the computational contents of classical proofs using Curry-style proof terms, with call/cc for classical logic:

$$t, u \quad ::= \quad x \quad | \quad \lambda x\,.\,t \quad | \quad tu \quad | \quad \text{cc}$$

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

Typing rules

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

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

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

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

From the derivation to the proof term

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{\overline{[\forall x\,(B(x)\Rightarrow C(x))]}^{\,g}}{B(x)\Rightarrow C(x)}
          \qquad
          \dfrac{
            \dfrac{\overline{[\forall x\,(A(x)\Rightarrow B(x))]}^{\,f}}{A(x)\Rightarrow B(x)}
            \qquad
            \overline{[A(x)]}^{\,u}
          }{B(x)}\,@
        }{C(x)}\,@
      }{A(x)\Rightarrow C(x)}\,\lambda u
    }{\forall x\,(A(x)\Rightarrow C(x))}
  }{\forall x\,(B(x)\Rightarrow C(x)) \;\Rightarrow\; \forall x\,(A(x)\Rightarrow C(x))}\,\lambda g
}{\forall x\,(A(x)\Rightarrow B(x)) \;\Rightarrow\; \forall x\,(B(x)\Rightarrow C(x)) \;\Rightarrow\; \forall x\,(A(x)\Rightarrow C(x))}\,\lambda f
$$

$$\lambda f\,.\,\lambda g\,.\,\lambda u\,.\,g\,(f\;u)$$

Typing examples

- Intuitionistic principles:

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

- Excluded middle, double negation elimination:

$$\begin{array}{rcl} \textbf{left} & \equiv & \lambda xuv\,.\,u\,x & : & \forall X\,\forall Y\,(X \Rightarrow X \vee Y) \\ \textbf{right} & \equiv & \lambda yuv\,.\,v\,y & : & \forall X\,\forall Y\,(Y \Rightarrow X \vee Y) \end{array}$$

$$\textbf{EM} \;\equiv\; \text{œ}\,(\lambda k\,.\,\textbf{right}\,(\lambda x\,.\,k\,(\textbf{left}\,x))) \quad : \quad \forall X\,(X \vee \neg X)$$

$$\textbf{DNE} \;\equiv\; \lambda z\,.\,\text{œ}\,(\lambda k\,.\,z\,k) \quad : \quad \forall X\,(\neg\neg X \Rightarrow X)$$

- De Morgan laws:

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

Plan

Adding axioms

- Defining equations of all 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:

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

Definition of Second-Order Logic (in this tutorial)

SOL	=	System NK2
	+	Defining equations (of prim. rec. functions)
	+	Peano axioms (P3) and (P4)

- **Remark:** No induction axiom!

Induction

- **Problem:** Induction axiom is not realizable!

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

- **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 $\text{PA2} \vdash A$, then $\text{PA2} - \text{Ind} \vdash A^{\mathbb{N}}$ ($A^{\mathbb{N}} = A$ relativized to $\mathbb{N}$)

Plan

Terms, stacks and processes

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

Terms, stacks and processes

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

- A λ-calculus with two kinds of constants:
  - Instructions $\kappa \in \mathcal{K}$, including æ
  - Continuation constants k_π, one for every stack π (generated by æ)
- **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	$\star$	π	$\succ$	t	$\star$	$u \cdot \pi$
Grab	$\lambda x\,.\,t$	$\star$	$u \cdot \pi$	$\succ$	$t\{x := u\}$	$\star$	π
Save	$\mathtt{c\!c}$	$\star$	$u \cdot \pi$	$\succ$	u	$\star$	$\mathrm{k}_\pi \cdot \pi$
Restore	k_π	$\star$	$u \cdot \pi'$	$\succ$	u	$\star$	π
		$\ldots$				$\ldots$	

(+ 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:

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

  - Example: $$(\lambda xy\,.\,t)\,u\,v \star \pi \;\succ\; \lambda xy\,.\,t \star u \cdot v \cdot \pi \;\succ\; t\{x := u\}\{y := v\} \star \pi$$

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

Save	$\text{c\!c} \star u \cdot \pi$	$\succ$	$u \star \mathrm{k}_\pi \cdot \pi$
Restore	$\mathrm{k}_\pi \star u \cdot \pi'$	$\succ$	$u \star \pi$

  - Instruction $\text{c\!c}$ creates continuation constants k_π.
    Most often used in the pattern
    $$\text{c\!c}\,(\lambda k\,.\,t) \star \pi \quad \succ \quad \cdots \quad \succ \quad t\{k := \mathrm{k}_\pi\} \star \pi$$
  - Continuation constant k_π restores the saved context π

Example of extra instructions

- 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 (DC) [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)