Classical realizability as a classifier for non-determinism

Guillaume Geoffroy

I2M, Aix-Marseille Université

July 10, 2018

Krivine's classical realizability

$$\begin{array}{ccc} \text{Model of computation} & & \text{Realizability theory} \\ (\text{programs } t, u, v, \ldots) & \longrightarrow & (\text{formulas } A, B, C, \ldots) \end{array}$$

Krivine's classical realizability

$$\begin{array}{ccc} \text{Model of computation} & & \text{Realizability theory} \\ \text{(programs } t, u, v, \ldots) & \longrightarrow & \text{(formulas } A, B, C, \ldots) \end{array}$$

$$\begin{array}{ccc} \text{Extension of} & & \text{Extension of} \\ \text{pure } \lambda\text{-calculus} + \text{control} & \longrightarrow & \text{ZF or PA}_2 \end{array}$$

Krivine's classical realizability

$$\begin{array}{ccc} \text{Model of computation} & & \text{Realizability theory} \\ (\text{programs } t, u, v, \ldots) & \longrightarrow & (\text{formulas } A, B, C, \ldots) \end{array}$$

$$\begin{array}{ccc} \text{Extension of} & & \text{Extension of} \\ \text{pure } \lambda\text{-calculus} + \text{control} & \longrightarrow & \text{ZF or PA}_2 \\ & & \downarrow \\ & & \text{Realizbility models} \\ & & \text{of ZF or PA}_2 \end{array}$$

Krivine's classical realizability

$$
\begin{tikzcd}
\begin{array}{c}\text{Model of computation}\\ \text{(programs } t, u, v, \ldots\text{)}\end{array} \arrow[r] & \begin{array}{c}\text{Realizability theory}\\ \text{(formulas } A, B, C, \ldots\text{)}\end{array}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\begin{array}{c}\text{Extension of}\\ \text{pure } \lambda\text{-calculus} + \text{control}\end{array} \arrow[r] & \begin{array}{c}\text{Extension of}\\ \text{ZF or } \mathrm{PA}_2\end{array} \arrow[d] \\
\text{Forcing poset} \arrow[u] & \begin{array}{c}\text{Realizbility models}\\ \text{of ZF or } \mathrm{PA}_2\end{array}
\end{tikzcd}
$$

Krivine's classical realizability

Model of computation (programs $t, u, v, \ldots$) $\xrightarrow[\substack{t \Vdash A \\ t \text{ realizes } A}]{}$ Realizability theory $\{\ A;\ \exists t\ (t \Vdash A)\ \}$

Extension of pure λ-calculus + control $\longrightarrow$ Extension of ZF or PA_2

Behaviour

$$t \Vdash A \iff \begin{array}{c}t \text{ has the } \textit{behaviour} \\ \text{prescribed by } A\end{array}$$

Behaviour

$$t \Vdash A \iff \begin{array}{c} t \text{ has the } \textit{behaviour} \\ \text{prescribed by } A \end{array} \iff \begin{array}{c} t \text{ passes all tests} \\ \text{required by } A \end{array}$$

Behaviour

$$t \Vdash A \iff \begin{array}{c} t \text{ has the } \textit{behaviour} \\ \text{prescribed by } A \end{array} \iff \begin{array}{c} t \text{ passes all tests} \\ \text{required by } A \end{array}$$

$$t \Vdash \forall X\, \forall Y\, (X \to Y \to Y)$$

Behaviour

$$t \Vdash A \iff \begin{array}{c} t \text{ has the } \textit{behaviour} \\ \text{prescribed by } A \end{array} \iff \begin{array}{c} t \text{ passes all tests} \\ \text{required by } A \end{array}$$

$$t \Vdash \forall X\, \forall Y\, (X \to Y \to Y) \iff \begin{array}{c} t \text{ captures the behaviour} \\ \text{of its second argument} \end{array}$$

Behaviour

$t \Vdash A \iff$ t has the *behaviour* prescribed by A $\iff$ t passes all tests required by A

$t \Vdash \forall X\ \forall Y\ (X \to Y \to Y) \iff$ t captures the behaviour of its second argument

$\Updownarrow$

t behaves like $\lambda x.\lambda y.y$
(true)

$\Updownarrow$

for all u, v, $(t\ u\ v) \rightsquigarrow v$

Deterministic reductions

test = "does t *eventually* do ...?"

Deterministic reductions

test = "does t *eventually* do ...?"

$$t \underset{\text{deterministically}}{\rightsquigarrow} u$$

Deterministic reductions

test = "does t *eventually* do ...?"

$$t \underset{\text{deterministically}}{\rightsquigarrow} u$$

$$\Downarrow$$

t behaves like u

(t passes all tests that u passes)

Non-deterministic reductions

$$\begin{tikzcd}[row sep=small] & u \\ t \arrow[ur, rightsquigarrow] \arrow[dr, rightsquigarrow] & \\ & v \end{tikzcd}$$

Non-deterministic reductions

$$
\begin{tikzcd}
 & u \\
t \arrow[ur, squiggly] \arrow[dr, squiggly] & \\
 & v
\end{tikzcd}
$$

test = "*must t* eventually do . . . ?"

Non-deterministic reductions

$$\begin{tikzcd} & u \\ t \arrow[ur, rightsquigarrow] \arrow[dr, rightsquigarrow] & \\ & v \end{tikzcd}$$

test = "*must t* eventually do . . . ?"

or

Non-deterministic reductions

$$
\begin{tikzcd}[row sep=small]
 & u \\
t \arrow[ur, rightsquigarrow] \arrow[dr, rightsquigarrow] & \\
 & v
\end{tikzcd}
$$

test = "*must t* eventually do . . . ?"

or

test = "*may t* eventually do . . . ?"

Non-deterministic reductions

$$
\begin{tikzcd}
& u \\
t \arrow[ur, rightsquigarrow] \arrow[dr, rightsquigarrow] & \\
& v
\end{tikzcd}
$$

t passes only tests
that *u* and *v* pass
(intersection of behaviours)

or

t passes all tests
that *u* or *v* passes
(union of behaviours)

Non-deterministic reductions

$$\begin{tikzcd} & x \\ \psi\ x\ y \arrow[ur, rightsquigarrow] \arrow[dr, rightsquigarrow] & \\ & y \end{tikzcd}$$

Non-deterministic reductions

$$\psi\ x\ y \rightsquigarrow x$$

$$\psi\ x\ y \rightsquigarrow y$$

$\psi\ x\ y$ passes only tests that x and y pass

Non-deterministic reductions

$$\begin{tikzcd}[column sep=huge]
 & x \\
\psi\ x\ y \arrow[ur, rightsquigarrow] \arrow[dr, rightsquigarrow] & \\
 & y
\end{tikzcd}$$

ψ x y passes only tests
that x and y pass

$$\Downarrow$$

ψ adds nothing to
the realizability theory

Non-deterministic reductions

$$\begin{tikzcd} & x \\ \psi\ x\ y \arrow[ur, rightsquigarrow] \arrow[dr, rightsquigarrow] & \\ & y \end{tikzcd}$$

ψ x y passes all tests that x or y passes

Non-deterministic reductions

$$\begin{tikzcd} & x \\ \psi\ x\ y \arrow[ur, rightsquigarrow] \arrow[dr, rightsquigarrow] & \\ & y \end{tikzcd}$$

ψ x y passes all tests
that x or y passes

$$\Downarrow$$

the realizability theory
could be obtained by forcing

Non-deterministic reductions

$$\begin{tikzcd}[row sep=small] & u \\ t \arrow[ur, rightsquigarrow] \arrow[dr, rightsquigarrow] & \\ & v \end{tikzcd}$$

$$t \rightsquigarrow u \; ; \; v$$

Union of behaviours

Intersection of behaviours

Non-deterministic reductions

$$\begin{tikzcd} & u \\ t \arrow[ur, rightsquigarrow] \arrow[dr, rightsquigarrow] & \\ & v \end{tikzcd}$$

$$t \rightsquigarrow u \;;\; v$$

$$\{t\} \succ \{u\}$$
and $$\{t\} \succ \{v\}$$

$$\{t\} \succ \{u \;;\; v\}$$

Voting

$$\begin{tikzcd}
 & u_0 \;;\; u_1 \\
t \arrow[ur, rightsquigarrow] \arrow[r, rightsquigarrow] \arrow[dr, rightsquigarrow] & u_1 \;;\; u_2 \\
 & u_2 \;;\; u_0
\end{tikzcd}$$

2-out-of-3 voting

Voting

$$\begin{tikzcd}
 & u_0 \;;\; u_1 \;;\; \ldots \;;\; u_{k-1} \\
 & \vdots \\
t \arrow[ruu, rightsquigarrow] \arrow[r, rightsquigarrow] \arrow[rdd, rightsquigarrow] & \text{all } \binom{n}{k} \text{ combinations} \\
 & \vdots \\
 & u_{n-k} \;;\; u_1 \;;\; \ldots \;;\; u_{n-1}
\end{tikzcd}$$

k-out-of-n voting

The caracteristic Boolean algebra $\beth 2$

$$\begin{array}{c}\text{First-order formula } A\\ \text{on Boolean Algebras}\end{array} \xrightarrow[\textit{translation}]{} \begin{array}{c}\text{Formula } (\beth 2 \models A) \text{ of the}\\ \text{realizability language}\end{array}$$

The caracteristic Boolean algebra $\gimel 2$

First-order formula A on Boolean Algebras $\xrightarrow[\textit{translation}]{}$ Formula ($\gimel 2 \models A$) of the realizability language

$$\forall x \; \forall y \; \forall z \quad x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

The operation $\wedge$ is associative

The caracteristic Boolean algebra $\gimel 2$

First-order formula A on Boolean Algebras $\xrightarrow[\textit{translation}]{}$ Formula $(\gimel 2 \models A)$ of the realizability language

$$\gimel 2 \models \ \forall x\, \forall y\, \forall z \quad x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

The operation $\wedge$ on $\gimel 2$ is associative

The caracteristic Boolean algebra $\beth 2$

First-order formula A on Boolean Algebras $\xrightarrow[\textit{translation}]{}$ Formula ($\beth 2 \models A$) of the realizability language

$\beth 2 \models \ \forall x\ \forall y\ \forall z$ $x \wedge (y \wedge z) = (x \wedge y) \wedge z$	The operation $\wedge$ on $\beth 2$ is associative
$\forall x\ \forall y$ $(x = 0) \vee (x = 1)$	There are only two elements

The caracteristic Boolean algebra $\gimel 2$

First-order formula A on Boolean Algebras $\xrightarrow[\textit{translation}]{}$ Formula ($\gimel 2 \models A$) of the realizability language

$$\gimel 2 \models \forall x\, \forall y\, \forall z \quad x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

The operation $\wedge$ on $\gimel 2$ is associative

$$\gimel 2 \models \forall x\, \forall y \quad (x = 0) \vee (x = 1)$$

$\gimel 2$ only has two elements

Non-deterministic reductions

$$
\begin{tikzcd}
 & x \\
\psi\ x\ y \arrow[ur, rightsquigarrow] \arrow[dr, rightsquigarrow] & \\
 & y
\end{tikzcd}
$$

$\psi\ x\ y$ passes all tests
that x or y passes

$$\Downarrow$$

the realizability theory
could be obtained by forcing

The key idea

$\forall x \in \gimel 2 \ A(x) \quad \longleftrightarrow$ Union of behaviours
$A(0)$ and $A(1)$

$\exists x \in \gimel 2 \ A(x) \quad \longleftrightarrow$ Intersection of behaviours
$A(0)$ and $A(1)$

Equivalences

There is a program which simulates $(n-1)$-out-of-n voting $\iff$ "$\beth_2$ has less than 2^n elements" is realized

Equivalences

There is a program
which simulates
$(n-1)$-out-of-n voting
$\iff$
"$\beth 2$ has less than 2^n elements"
is realized

There is a program
which simulates
k-out-of-n voting
$\iff$
"$\beth 2$ has less than $2^{\lceil \frac{n}{n-k} \rceil}$ elements"
is realized

Equivalences

There is a program which simulates $(n-1)$-out-of-n voting $\iff$ "$\beth 2$ has less than 2^n elements" is realized

There is a program which simulates k-out-of-n voting $\iff$ "$\beth 2$ has less than $2^{\lceil \frac{n}{n-k} \rceil}$ elements" is realized

k-out-of-n voting can be simulated with j-out-of-m voting *if and only if* $\lceil \frac{m}{m-j} \rceil \leq \lceil \frac{n}{n-k} \rceil$

Parallel or

	true	false	?
true	true	true	true
false	true	false	?
?	true	?	?

Equivalences

There is a program which computes *parallel or* $\iff$ "$\gimel 2$ has at most 4 elements" is realized

Equivalences

$$\begin{tikzcd}
\begin{array}{c}\text{There is a program}\\ \text{which computes}\\ \textit{parallel or}\end{array} \arrow[r, Leftrightarrow] \arrow[dr, dashed, Leftrightarrow] & \begin{array}{c}\text{“}\gimel\text{2 has at most 4 elements”}\\ \text{is realized}\end{array} \arrow[d, Leftrightarrow] \\
 & \begin{array}{c}\text{There is a program}\\ \text{which simulates}\\ \text{2-out-of-3 voting}\end{array}
\end{tikzcd}$$

Equivalences

There is a program which computes *parallel or*	$\Longleftrightarrow$	"$\gimel 2$ has at most 4 elements" is realized
There is a program which computes *Gustave's function*	$\Longleftrightarrow$	"$\gimel 2$ has at most 8 elements" is realized

Equivalences

There is a program which computes *parallel or*	$\Longleftrightarrow$	"$\gimel 2$ has at most 4 elements" is realized
There is a program which computes *Gustave's function*	$\Longleftrightarrow$	"$\gimel 2$ has at most 8 elements" is realized

Gustave's function can be simulated with *parallel or*, but not the converse

Conclusion

Non-deterministic behaviour (unknown structure) $\longleftrightarrow$ Information on $\beth 2$ (boolean algebra)

What about non-classical settings?
(pure λ-calculus, PCF, *etc.*)