

Classical realizability as a classifier for non-determinism

Guillaume Geoffroy

I2M, Aix-Marseille Université

July 10, 2018

Krivine's classical realizability



Krivine's classical realizability

Model of computation
(programs t, u, v, \dots) \longrightarrow Realizability theory
(formulas A, B, C, \dots)

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

Krivine's classical realizability

Model of computation
(programs t, u, v, \dots)



Realizability theory
(formulas A, B, C, \dots)

Extension of
pure λ -calculus + control



Extension of
ZF or PA_2



Realizability models
of ZF or PA_2

Krivine's classical realizability

Model of computation
(programs t, u, v, \dots)



Realizability theory
(formulas A, B, C, \dots)

Extension of
pure λ -calculus + control



Forcing poset



Extension of
ZF or PA_2



Realizability models
of ZF or PA_2

Krivine's classical realizability

Model of computation
(programs t, u, v, \dots)

$\xrightarrow{\quad t \Vdash A \quad}$
 $t \text{ realizes } A$

Realizability theory
 $\{ A; \exists t (t \Vdash A) \}$

Extension of
pure λ -calculus + control

$\xrightarrow{\quad}$

Extension of
ZF or PA_2

Behaviour

$t \Vdash A \iff t$ has the *behaviour*
prescribed by A

Behaviour

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

Behaviour

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

$t \Vdash \forall X \forall Y (X \rightarrow Y \rightarrow Y)$

Behaviour

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

$t \Vdash \forall X \forall Y (X \rightarrow Y \rightarrow Y) \iff t \text{ captures the behaviour of its second argument}$

Behaviour

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

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

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

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

Deterministic reductions

test = “does t *eventually* do ...?”

Deterministic reductions

test = “does t eventually do ...?”

$t \overset{\text{~~~~~}}{\longrightarrow} u$
deterministically

Deterministic reductions

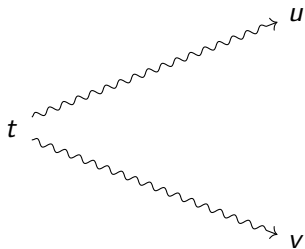
test = “does t eventually do ...?”

$t \overset{\text{~~~~~}}{\sim} u$
deterministically

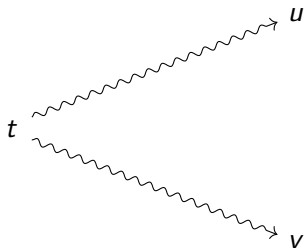


t behaves like u
(t passes all tests that u passes)

Non-deterministic reductions

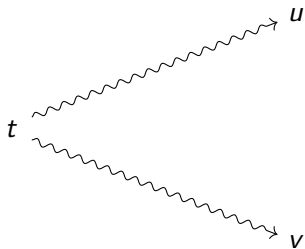


Non-deterministic reductions



test = “*must t eventually do ...?*”

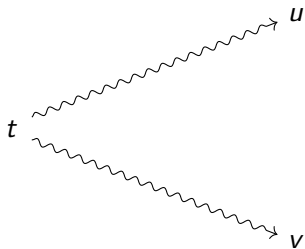
Non-deterministic reductions



test = “*must t eventually do ...?*”

_____ or _____

Non-deterministic reductions

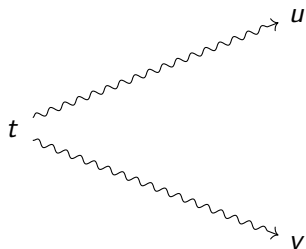


test = “*must* t
eventually do ...?”

———— or ————

test = “*may* t
eventually do ...?”

Non-deterministic reductions

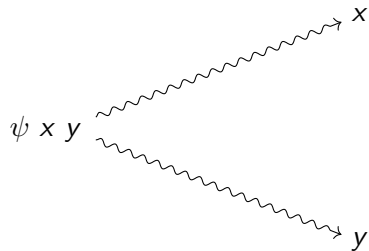


t passes only tests
that u and v pass
(intersection of behaviours)

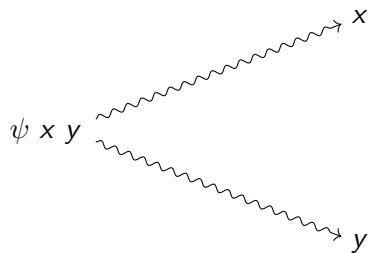
———— or ————

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

Non-deterministic reductions

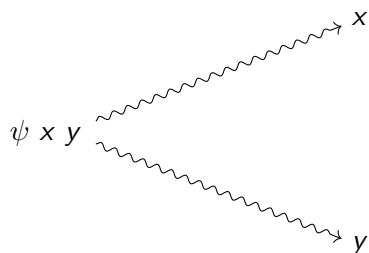


Non-deterministic reductions



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

Non-deterministic reductions

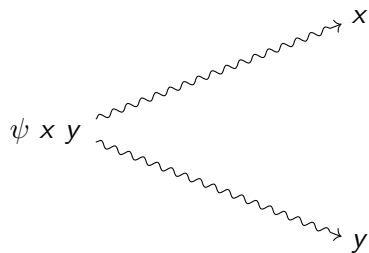


$\psi \times y$ passes only tests
that x and y pass



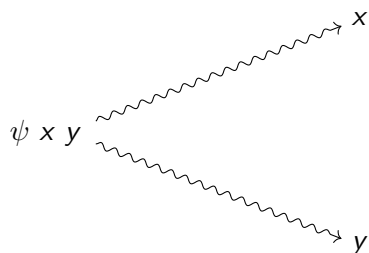
ψ adds nothing to
the realizability theory

Non-deterministic reductions



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

Non-deterministic reductions

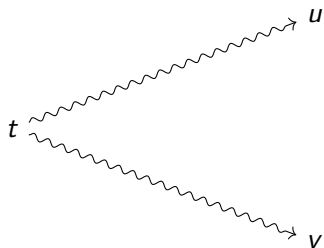


$\psi \ x \ y$ passes all tests
that x or y passes

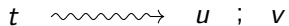


the realizability theory
could be obtained by forcing

Non-deterministic reductions

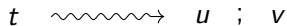
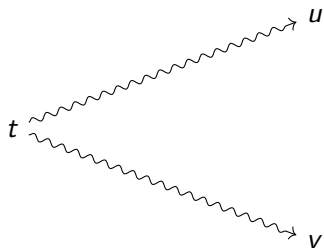


Union of behaviours



Intersection of behaviours

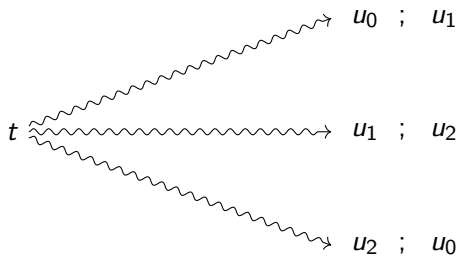
Non-deterministic reductions



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

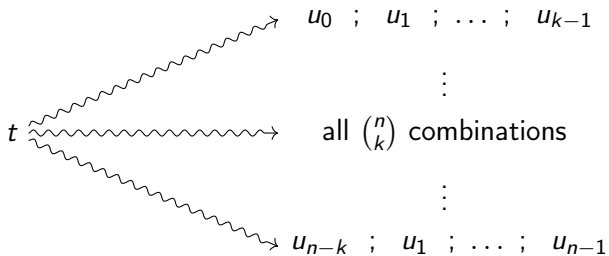
$$\{t\} \succ \{u ; v\}$$

Voting



2-out-of-3 voting

Voting



k-out-of-n voting

The characteristic Boolean algebra \mathbb{B}

First-order formula A
on Boolean Algebras $\xrightarrow{\text{translation}}$ Formula ($\mathbb{B} \models A$) of the
realizability language

The characteristic Boolean algebra $\mathbb{I}2$

First-order formula A
on Boolean Algebras $\xrightarrow{\text{translation}}$ Formula ($\mathbb{I}2 \models A$) of the
realizability language

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

The operation \wedge
is associative

The characteristic Boolean algebra $\mathbb{I}2$

First-order formula A on Boolean Algebras $\xrightarrow{\text{translation}}$ Formula ($\mathbb{I}2 \models A$) of the realizability language

$$\mathbb{I}2 \models \forall x \forall y \forall z \\ x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

The operation \wedge on $\mathbb{I}2$
is associative

The characteristic Boolean algebra $\mathbb{I}2$

First-order formula A
on Boolean Algebras $\xrightarrow{\text{translation}}$ Formula ($\mathbb{I}2 \models A$) of the
realizability language

$$\mathbb{I}2 \models \forall x \forall y \forall z \\ x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

The operation \wedge on $\mathbb{I}2$
is associative

$$\forall x \forall y \\ (x = 0) \vee (x = 1)$$

There are only two elements

The characteristic Boolean algebra $\mathbb{I}2$

First-order formula A
on Boolean Algebras $\xrightarrow{\text{translation}}$ Formula ($\mathbb{I}2 \models A$) of the
realizability language

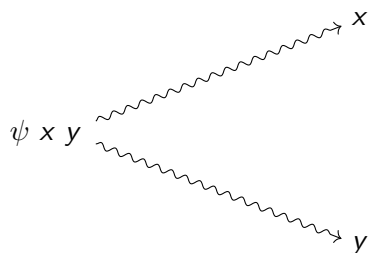
$$\mathbb{I}2 \models \forall x \forall y \forall z \\ x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

The operation \wedge on $\mathbb{I}2$
is associative

$$\mathbb{I}2 \models \forall x \forall y \\ (x = 0) \vee (x = 1)$$

$\mathbb{I}2$ only has two elements

Non-deterministic reductions



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



the realizability theory
could be obtained by forcing

The key idea

$\forall x \in \mathbb{N} A(x)$ \longleftrightarrow Union of behaviours
 $A(0)$ and $A(1)$

$\exists x \in \mathbb{N} A(x)$ \longleftrightarrow Intersection of behaviours
 $A(0)$ and $A(1)$

Equivalences

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

Equivalences

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

There is a program
which simulates k -out-of- n voting \iff “ $\exists 2$ 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 “ $\exists 2$ has less than 2^n elements”
is realized

There is a program
which simulates k -out-of- n voting \iff “ $\exists 2$ 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 $\lceil \frac{m}{m-j} \rceil \leq \lceil \frac{n}{n-k} \rceil$

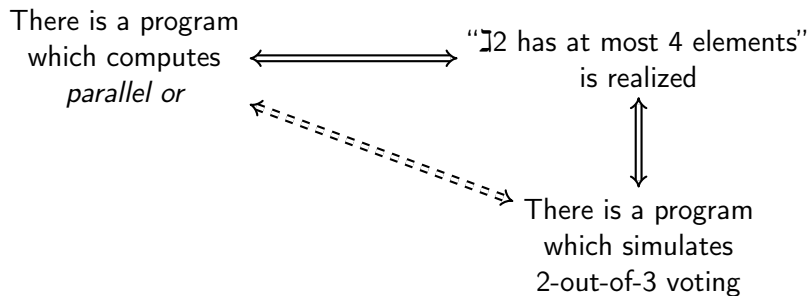
Parallel or

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

Equivalences

There is a program
which computes
parallel or \longleftrightarrow “ $\exists 2$ has at most 4 elements”
is realized

Equivalences



Equivalences

There is a program
which computes
parallel or \longleftrightarrow “ $\exists 2$ has at most 4 elements”
is realized

There is a program
which computes
Gustave's function \longleftrightarrow “ $\exists 2$ has at most 8 elements”
is realized

Equivalences

There is a program
which computes
parallel or \longleftrightarrow “ $\exists 2$ has at most 4 elements”
is realized

There is a program
which computes
Gustave's function \longleftrightarrow “ $\exists 2$ 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 $\mathbb{I}2$
(boolean algebra)

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