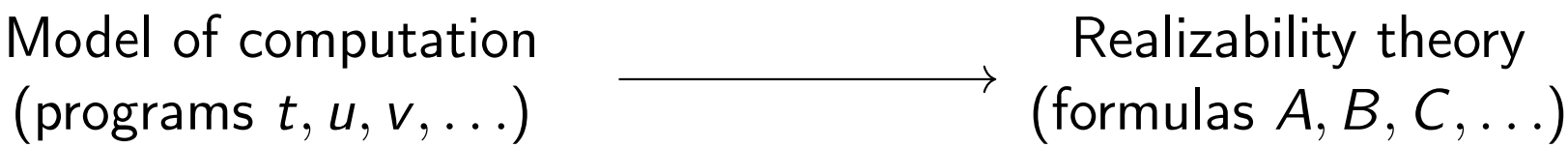
Classical realizability as a classifier for non-determinism

Guillaume Geoffroy

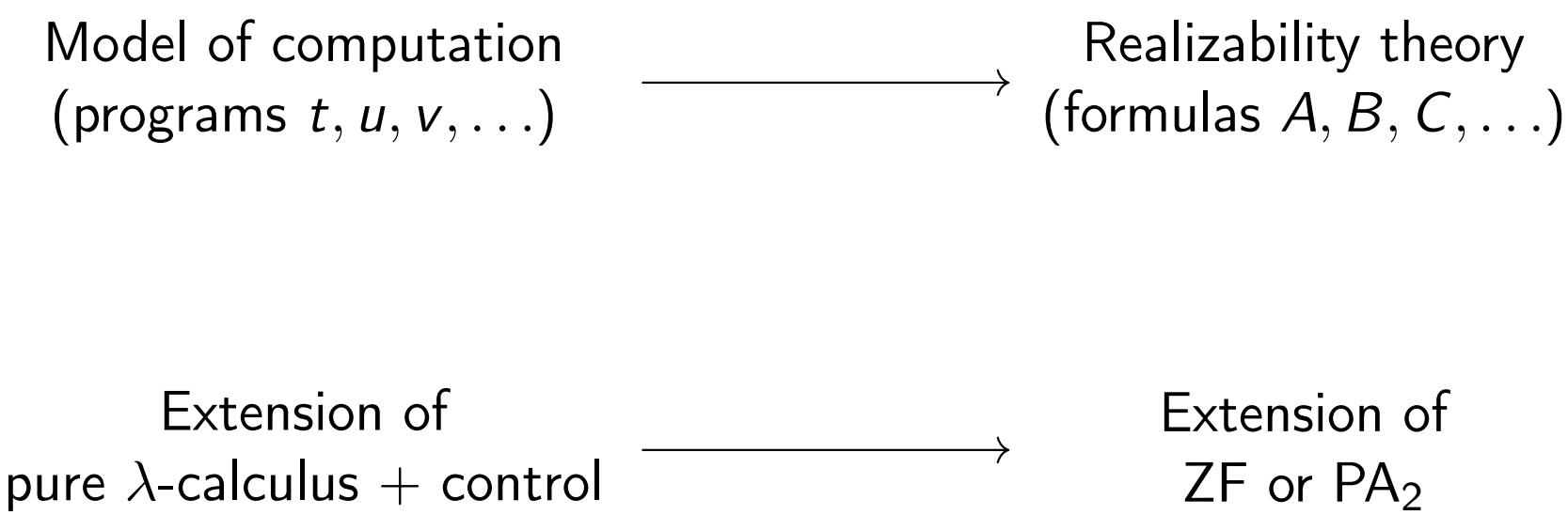
I2M, Aix-Marseille Université

July 10, 2018

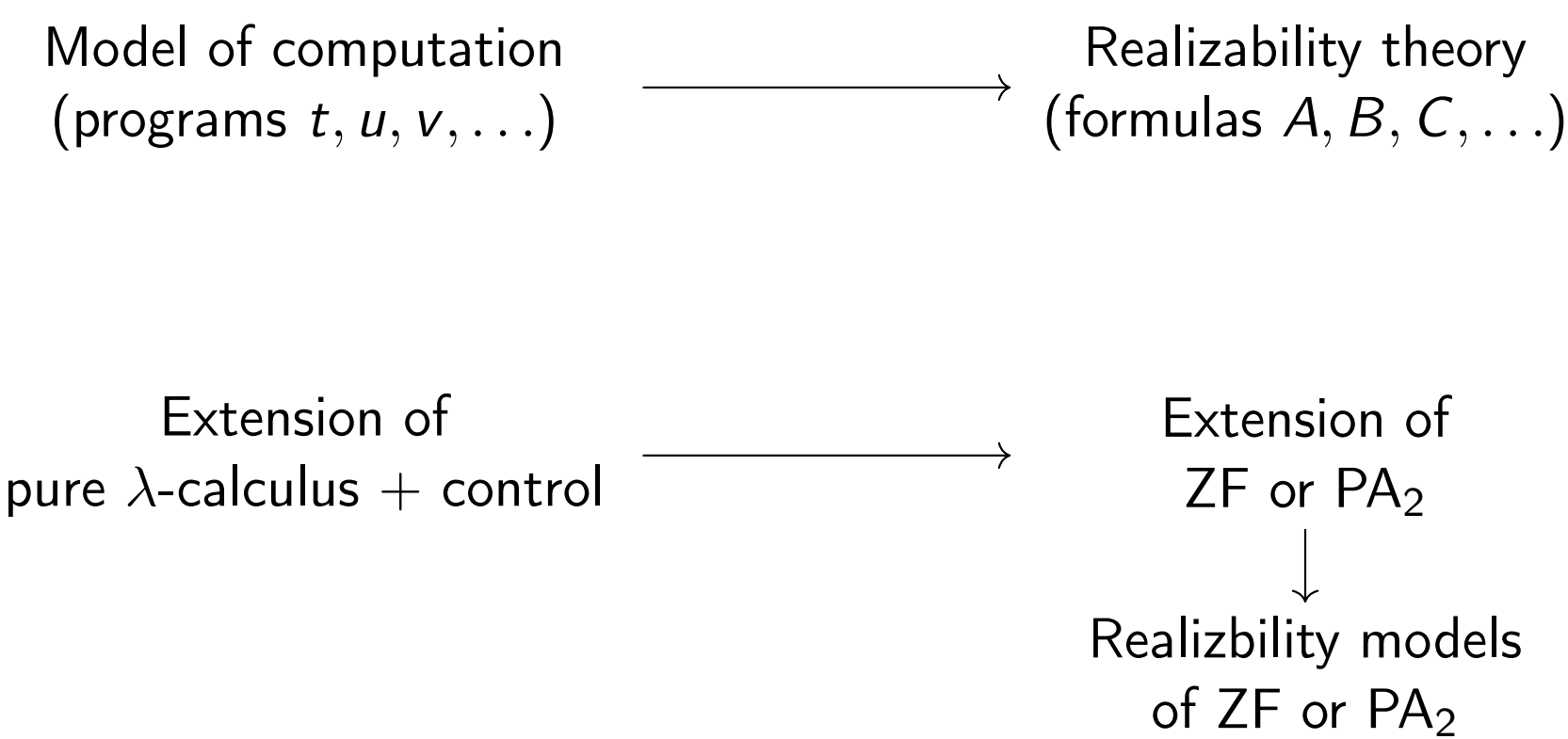
Krivine's classical realizability



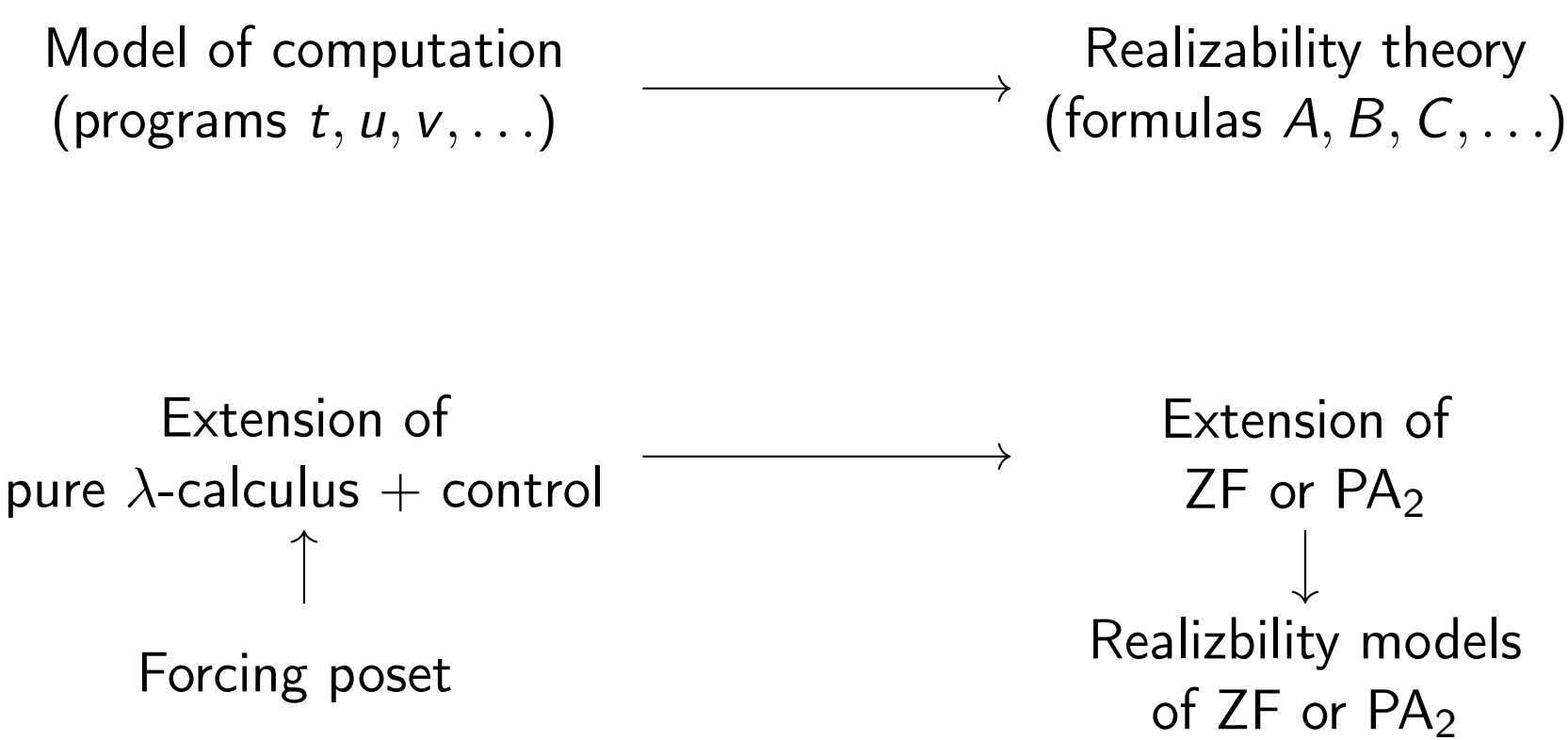
Krivine's classical realizability



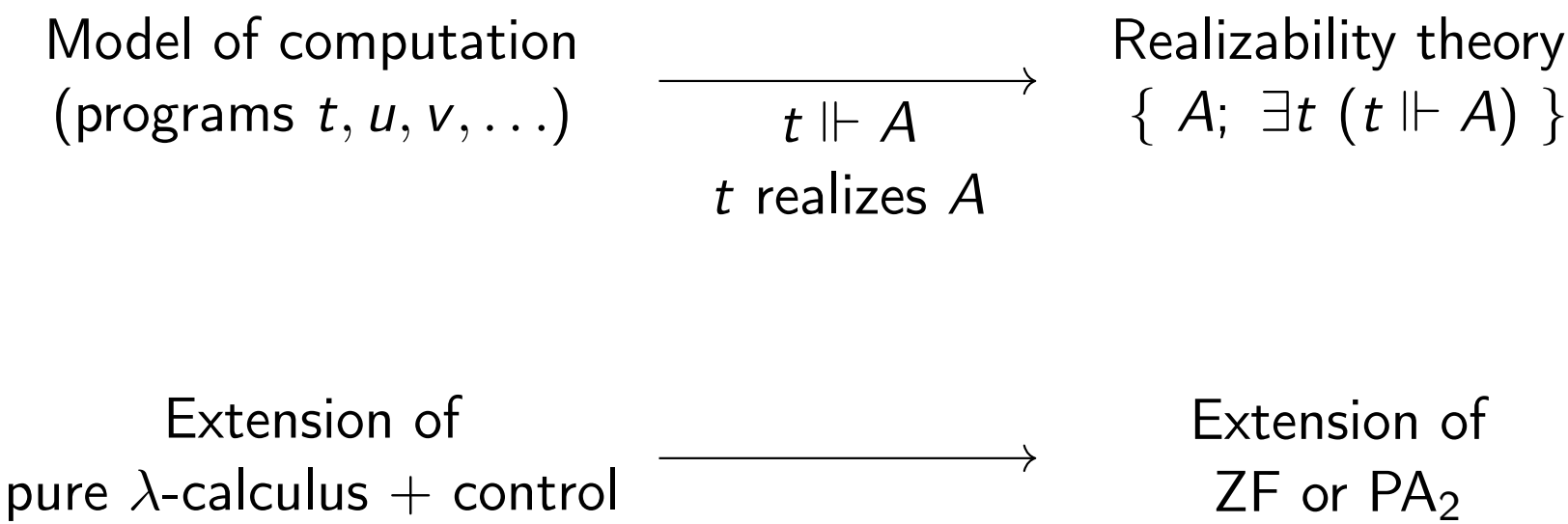
Krivine's classical realizability



Krivine's classical realizability



Krivine's classical realizability



Behaviour

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

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

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

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

Behaviour

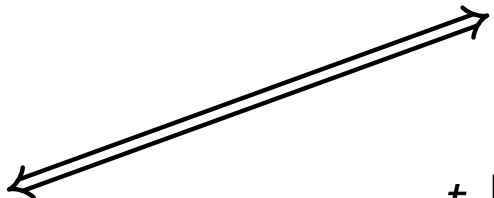
$$t \Vdash A \iff t \text{ has the } \textit{behaviour} \text{ prescribed by } A \iff t \text{ passes all tests required by } A$$

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

Behaviour

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

$t \Vdash \forall X\ \forall Y\ (X \to Y \to Y) \iff$ 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 . . . ?"

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

Deterministic reductions

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

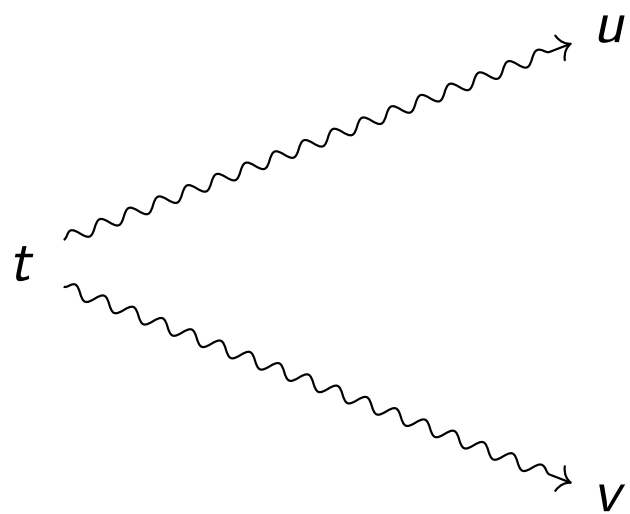
$t \overset{\text{deterministically}}{\rightsquigarrow} u$

$\Downarrow$

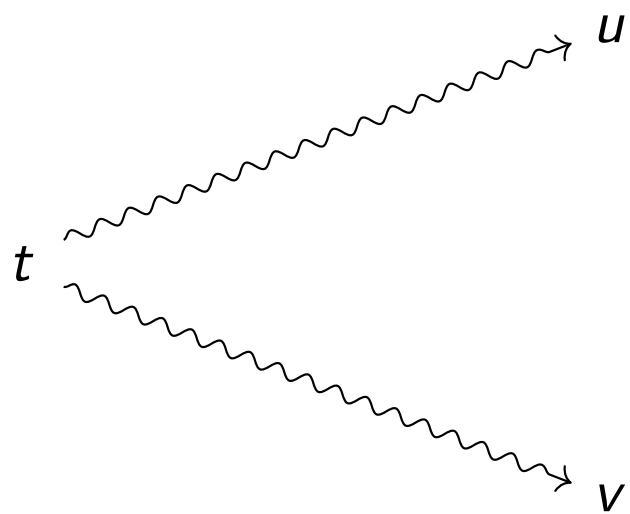
t behaves like u

(t passes all tests that u passes)

Non-deterministic reductions

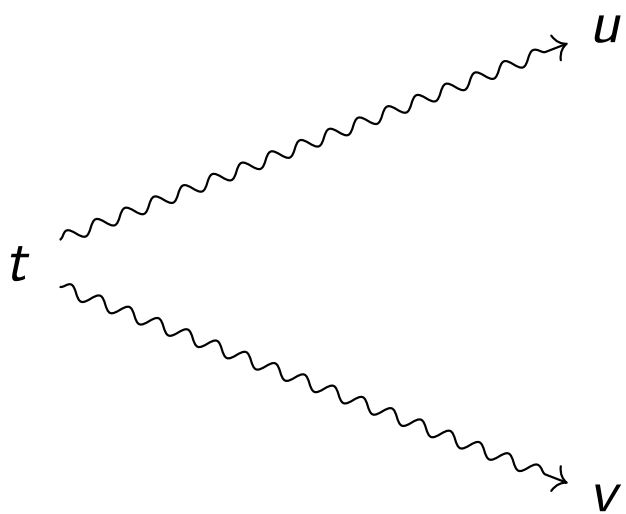


Non-deterministic reductions



test = “*must t* eventually do . . . ?”

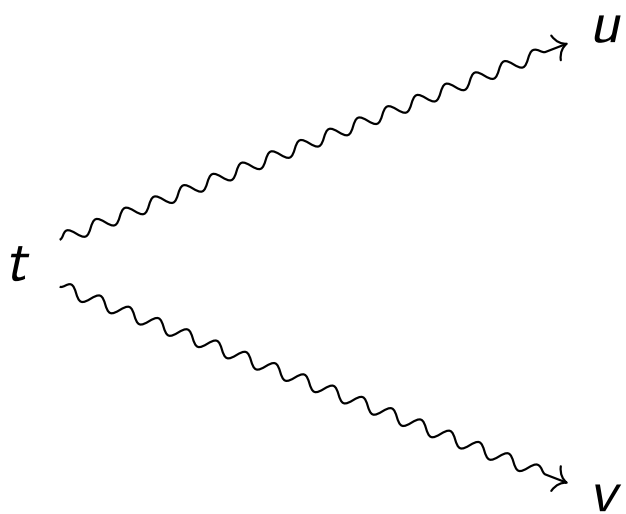
Non-deterministic reductions



test = “*must t*
eventually do . . . ?”

or

Non-deterministic reductions

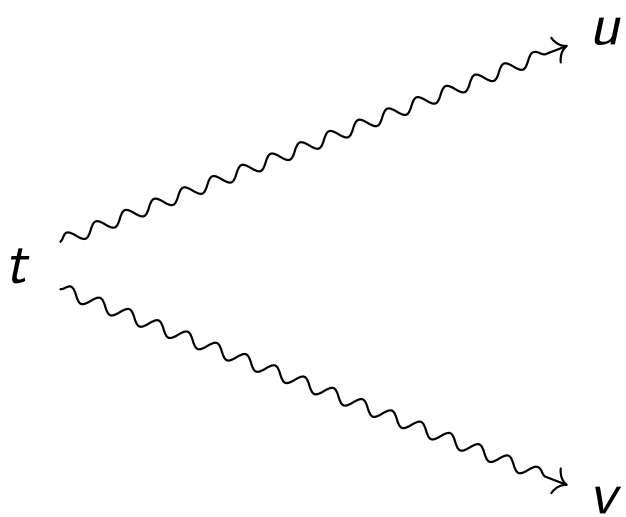


test = “*must t*
eventually do . . . ?”

——— *or* ———

test = “*may t*
eventually do . . . ?”

Non-deterministic reductions

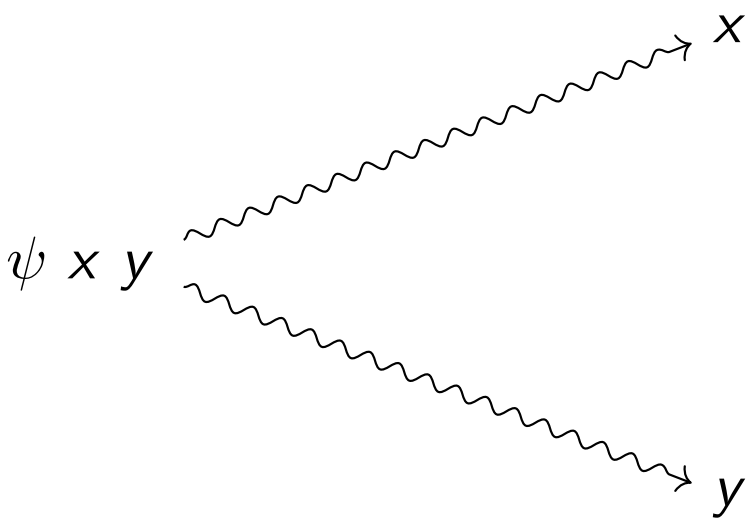


t passes only tests
that u and v pass
(intersection of behaviours)

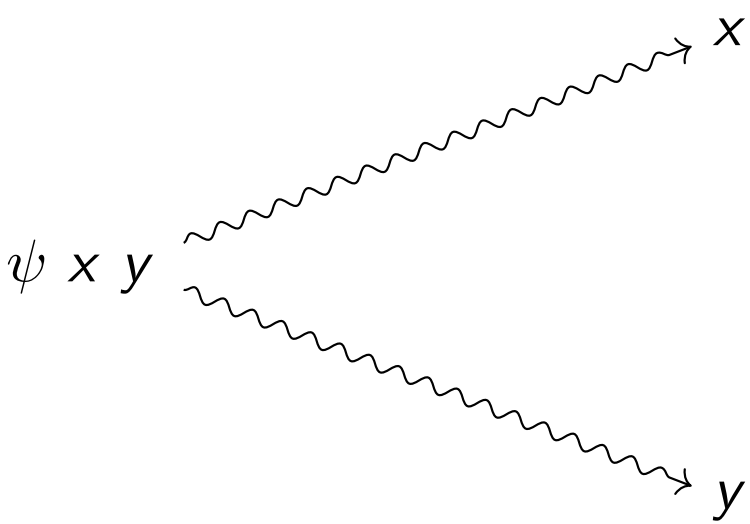
or

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

Non-deterministic reductions

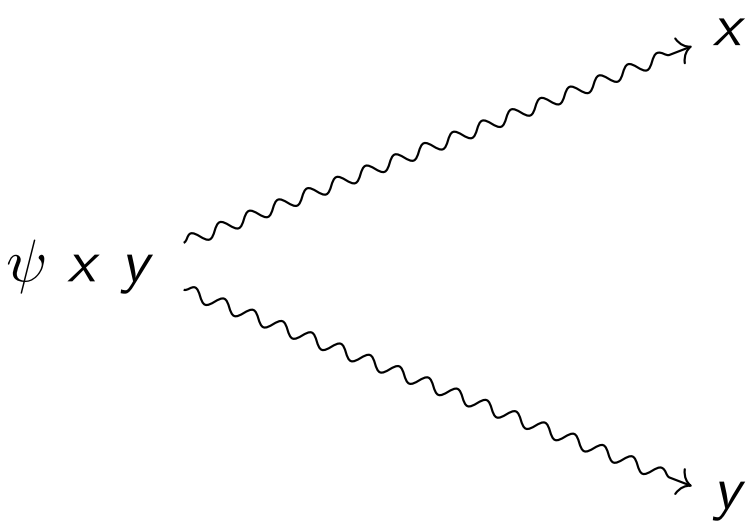


Non-deterministic reductions



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

Non-deterministic reductions

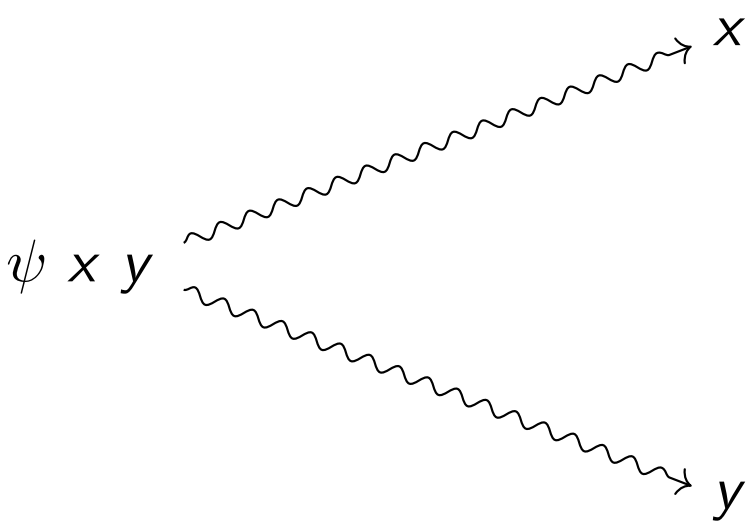


$\psi\ x\ y$ passes only tests
that x and y pass

$\Downarrow$

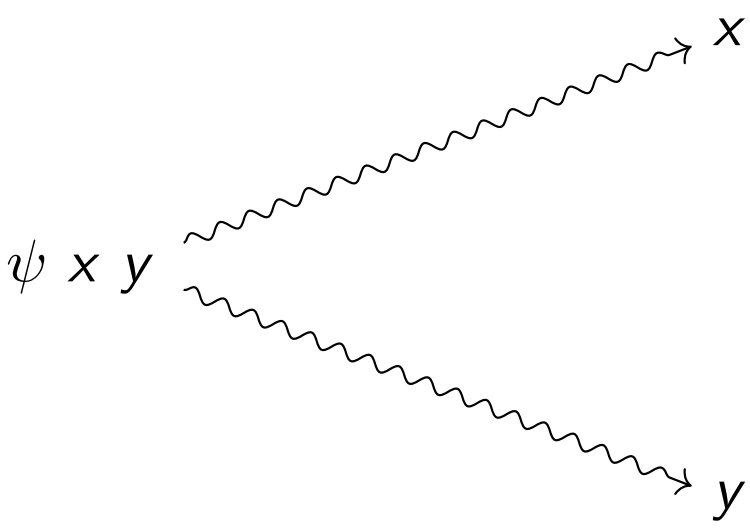
ψ adds nothing to
the realizability theory

Non-deterministic reductions



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

Non-deterministic reductions

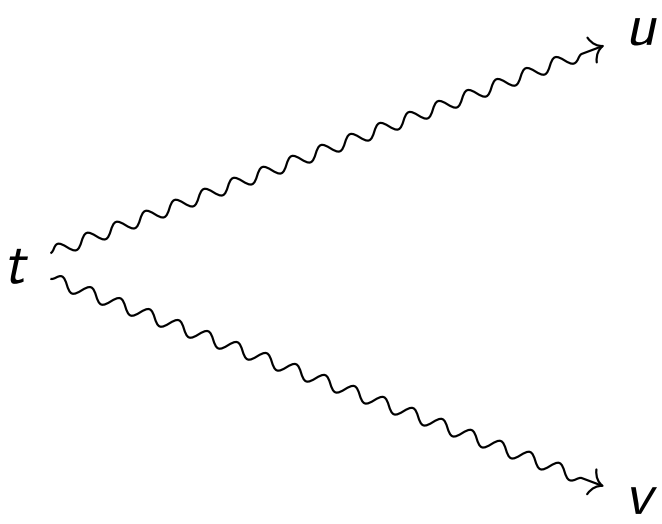


$\psi\ x\ y$ passes all tests
that x or y passes

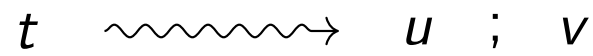
$\Downarrow$

the realizability theory
could be obtained by forcing

Non-deterministic reductions

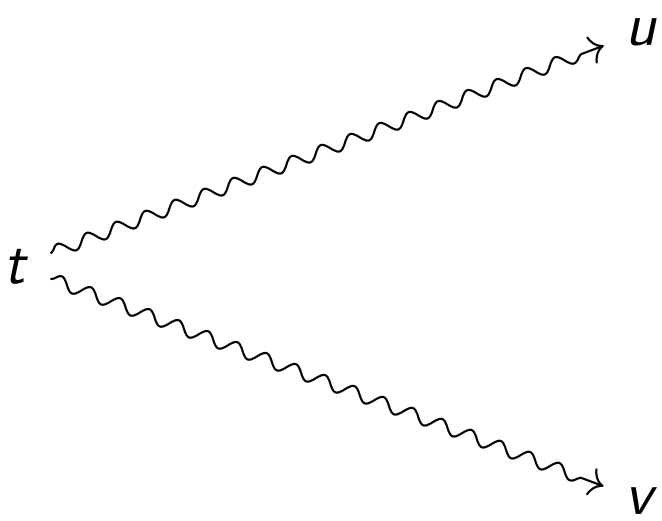


Union of behaviours



Intersection of behaviours

Non-deterministic reductions

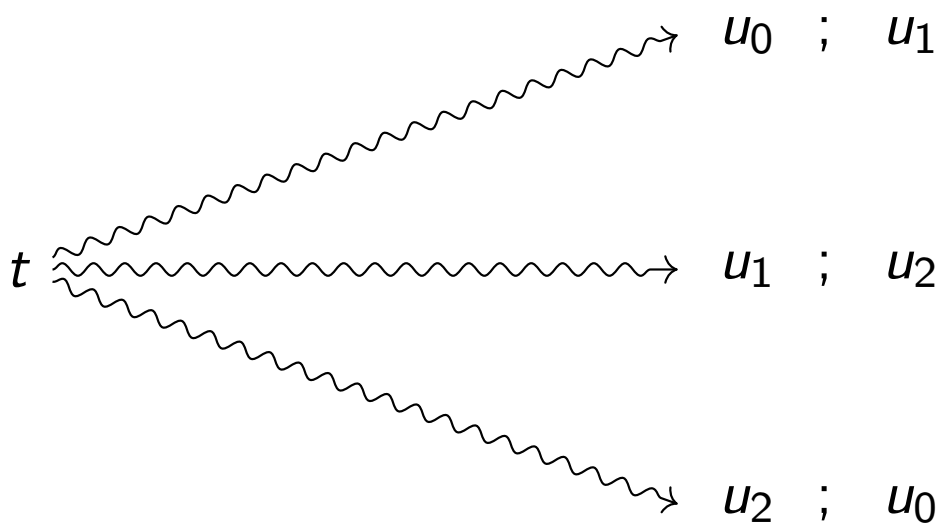


$$t \rightsquigarrow u \;;\; v$$

$$\begin{array}{rl} & \{t\} \succ \{u\} \\ \text{and} & \{t\} \succ \{v\} \end{array}$$

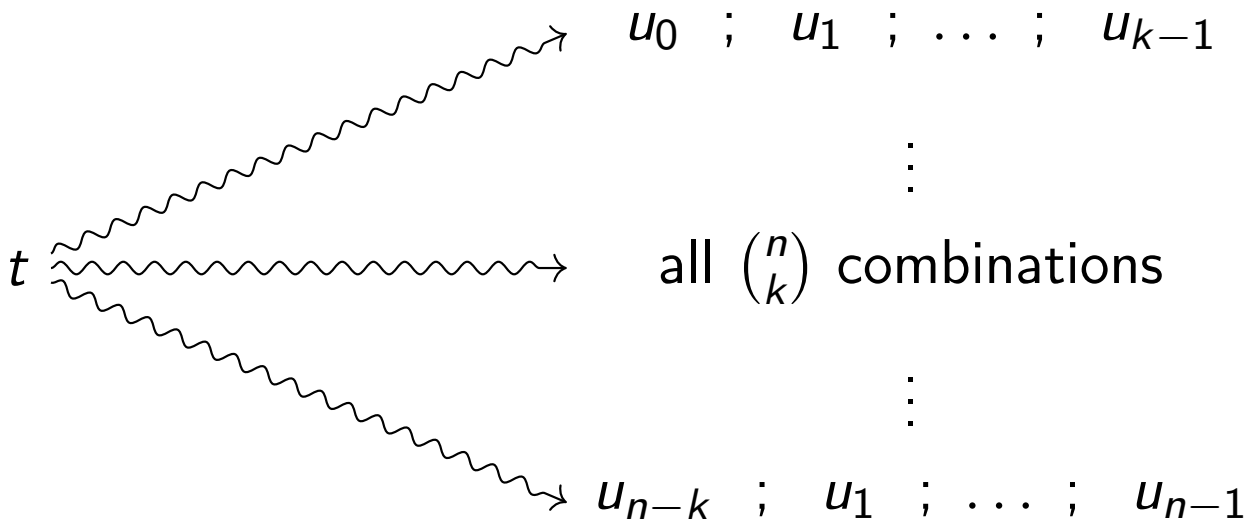
$$\{t\} \succ \{u \;;\; v\}$$

Voting



2-out-of-3 voting

Voting



k-out-of-n voting

The caracteristic Boolean algebra $\daleth 2$

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

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

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

The operation $\wedge$ 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

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 $\beth 2$

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

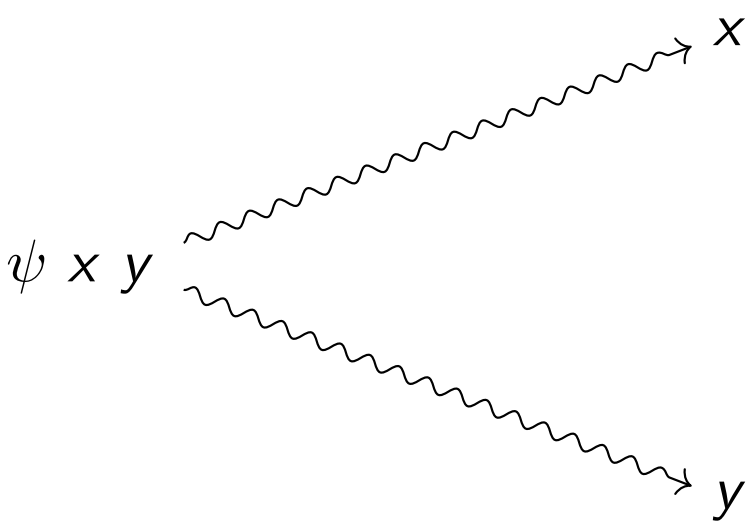
$$\beth 2 \models \ \forall x\ \forall y\ \forall z$$
$$x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

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

$$\beth 2 \models \ \forall x\ \forall y$$
$$(x = 0) \vee (x = 1)$$

$\beth 2$ only has two elements

Non-deterministic reductions



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

$\Downarrow$

the realizability theory
could be obtained by forcing

The key idea

$\forall x \in \mathbb{2} \; A(x) \quad \longleftrightarrow \quad$ Union of behaviours $A(0)$ and $A(1)$

$\exists x \in \mathbb{2} \; A(x) \quad \longleftrightarrow \quad$ Intersection of behaviours $A(0)$ and $A(1)$

Equivalences

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

Equivalences

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

There is a program which simulates k-out-of-n voting $\Longleftrightarrow$ "$\beth_2$ 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 $\Longleftrightarrow$ "$\exists 2$ has less than 2^n elements" is realized

There is a program which simulates k-out-of-n voting $\Longleftrightarrow$ "$\exists 2$ 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	?	?

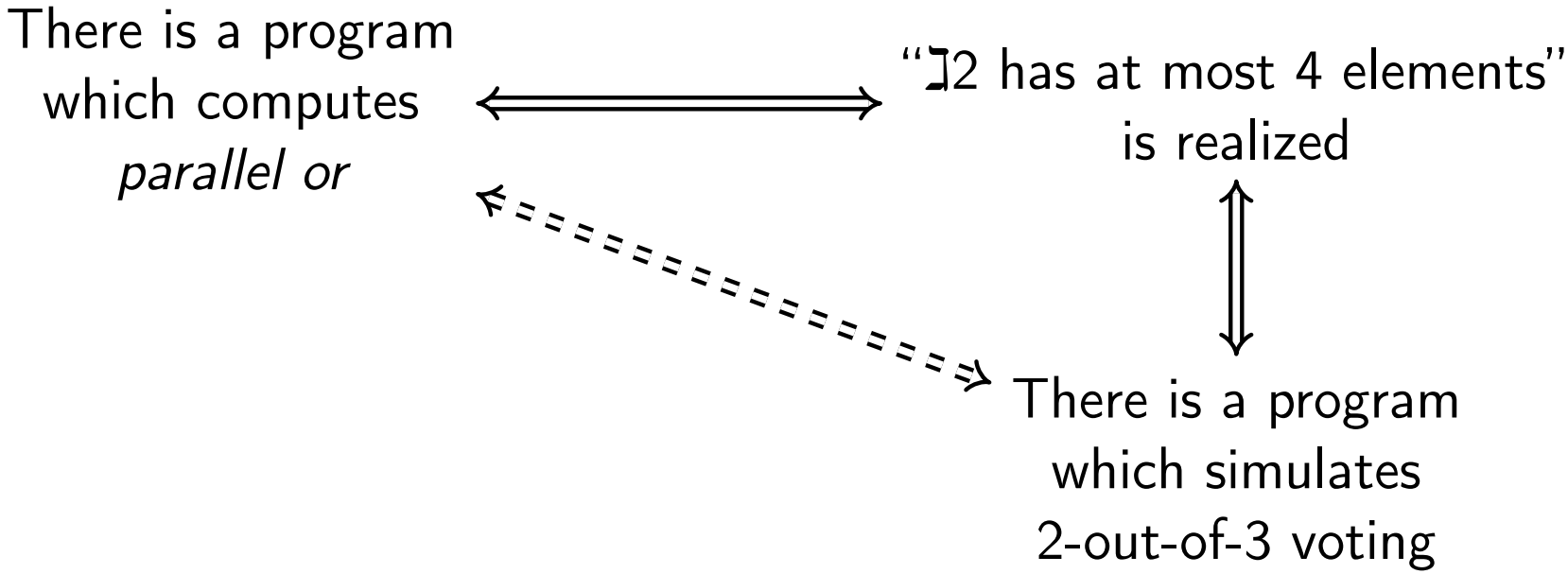
Equivalences

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 $\beth_2$ (boolean algebra)

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