Quantum computing, non-determinism, probabilistic systems... and the logic behind

Alejandro Díaz-Caro
Université Paris-Ouest Nanterre La Défense

Séminaire Modal'X
January 31, 2013

Outline

A brief and fast introduction to **quantum computing**

A brief and fast introduction to λ-**calculus**

A brief and fast introduction to **typed** λ-**calculus**

How does it relates to **intuitionistic logic**?
(a word on the Curry-Howard correspondence)

What I am doing
Algebraic calculi and **vectorial** typing
Non-determinism, a simpler problem

Work-in-progress
From non-determinism to *probabilities*

Outline

A brief and fast introduction to **quantum computing**

A brief and fast introduction to λ-**calculus**

A brief and fast introduction to **typed** λ-**calculus**

How does it relates to **intuitionistic logic**?
(a word on the Curry-Howard correspondence)

What I am doing
Algebraic calculi and **vectorial** typing
Non-determinism, a simpler problem

Work-in-progress
From non-determinism to *probabilities*

A brief and fast introduction to quantum computing

A bit of history

Richard Feynman's quote (1982)

I'm not happy with all the analyses that go with just the classical theory, because **nature isn't classical**, and if you want to make a **simulation of nature**, you'd better make it **quantum mechanical**, and by golly it's a wonderful problem, because it doesn't look so easy.

A brief and fast introduction to quantum computing

A bit of history

Richard Feynman's quote (1982)

I'm not happy with all the analyses that go with just the classical theory, because **nature isn't classical**, and if you want to make a **simulation of nature**, you'd better make it **quantum mechanical**, and by golly it's a wonderful problem, because it doesn't look so easy.

1985	David Deutsch	$\rightarrow$ 1st. model of Quantum Turing Machine
1993	Charles Benett *et.al.*	$\rightarrow$ Teleportation algorithm
1994	Peter Shor	$\rightarrow$ Fast factorisation algorithm
1996	Lov Grover	$\rightarrow$ Fast search algorithm
1998	Isaac Chuang *et.al.*	$\rightarrow$ 1st (1 qubit) quantum computer
2001	Lieven Vandersypen *et.al.*	$\rightarrow$ 7 qubit quantum computer

A brief and fast introduction to quantum computing

A bit of history

Richard Feynman's quote (1982)

I'm not happy with all the analyses that go with just the classical theory, because **nature isn't classical**, and if you want to make a **simulation of nature**, you'd better make it **quantum mechanical**, and by golly it's a wonderful problem, because it doesn't look so easy.

1985 David Deutsch → 1st. model of Quantum Turing Machine
1993 Charles Benett *et.al.* → Teleportation algorithm
1994 Peter Shor → Fast factorisation algorithm
1996 Lov Grover → Fast search algorithm
1998 Isaac Chuang *et.al.* → 1st (1 qubit) quantum computer
2001 Lieven Vandersypen *et.al.* → 7 qubit quantum computer
. . . factorising the number 15

⋮

A *physics free* introduction to quantum computing

Quantum vs. Classic, side by side

- **Classic computing** **Bit**: $0, 1$
  Quantum computing **Qubit**: Normalised vector from $\mathbb{C}^2$. We choose the basis $\{|0\rangle, |1\rangle\}$ where $|0\rangle = (0, 1)^T$ and $|1\rangle = (1, 0)^T$

A *physics free* introduction to quantum computing

Quantum vs. Classic, side by side

- **Classic computing Bit**: $0, 1$
  Quantum computing Qubit: Normalised vector from $\mathbb{C}^2$. We choose the basis $\{|0\rangle, |1\rangle\}$ where $|0\rangle = (0, 1)^T$ and $|1\rangle = (1, 0)^T$
- **Classic computing 2-bits system**: one from $\{00, 01, 10, 00\}$
  Quantum computing 2-qubits system: One from the tensor product of the two corresponding spaces. A canonical base for it is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, where $|xy\rangle = |x\rangle \otimes |y\rangle$.

A *physics free* introduction to quantum computing

Quantum vs. Classic, side by side

- **Classic computing** **Bit**: $0, 1$
  Quantum computing **Qubit**: Normalised vector from $\mathbb{C}^2$. We choose the basis $\{|0\rangle, |1\rangle\}$ where $|0\rangle = (0,1)^T$ and $|1\rangle = (1,0)^T$
- **Classic computing** **2-bits system**: one from $\{00, 01, 10, 00\}$
  Quantum computing **2-qubits system**: One from the tensor product of the two corresponding spaces. A canonical base for it is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, where $|xy\rangle = |x\rangle \otimes |y\rangle$.
- **Classic computing** **Reading data**: No problem
  Quantum computing **Measuring the system**: Measurement of $\alpha|0\rangle + \beta|1\rangle$ returns a **bit**, and the system collapses to
  - $|0\rangle$ if 0 was measured, with probability $p_0 = |\alpha|^2$
  - $|1\rangle$ if 1 was measured, with probability $p_1 = |\beta|^2$.

A *physics free* introduction to quantum computing

Cont.

- **Classic computing computation**: Logic gates {NOT, AND, etc...}
  Quantum computing operations: Unitary matrices ($U^\dagger U = I$)

Example:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

We can also combine them:
$(H \otimes I)$ = Apply H to the first qubit, and identity to the second

A *physics free* introduction to quantum computing

Cont.

- **Classic computing computation**: Logic gates {NOT, AND, etc...}
  Quantum computing operations: Unitary matrices ($U^\dagger U = I$)

Example:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

We can also combine them:
$(H \otimes I)$ = Apply H to the first qubit, and identity to the second

- **No-cloning theorem** "There is no universal cloning machine"
  i.e. $\nexists U$ s.t. $U|\phi\psi\rangle = |\psi\psi\rangle$ for an arbitrary qubit $|\psi\rangle$

A *physics free* introduction to quantum computing

Cont.

- **Classic computing computation**: Logic gates {NOT, AND, etc...}
  Quantum computing operations: Unitary matrices ($U^\dagger U = I$)

Example:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

We can also combine them:
$(H \otimes I)$ = Apply H to the first qubit, and identity to the second

- **No-cloning theorem** "There is no universal cloning machine"
  i.e. $\nexists U$ s.t. $U|\phi\psi\rangle = |\psi\psi\rangle$ for an arbitrary qubit $|\psi\rangle$

- **Entanglement** $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ is a 2-qubit system but cannot be written as $|\psi\rangle \otimes |\phi\rangle$...
  Consequence: Measuring the first qubit... both collapse!

A *physics free* introduction to quantum computing

Example: Deutsh's algorithm

Objective: To know if a 1-bit function $f : B \to B$ is constant or not.

Oracle: $U_f|x, y\rangle = |x, y \oplus f(x)\rangle$ where $\oplus$ is the addition modulo 2.

$$\boxed{U_f|x, 0\rangle = |x, f(x)\rangle}$$

A *physics free* introduction to quantum computing

Example: Deutsh's algorithm

Objective: To know if a 1-bit function $f : B \to B$ is constant or not.
Oracle: $U_f|x, y\rangle = |x, y \oplus f(x)\rangle$ where $\oplus$ is the addition modulo 2.

$$\boxed{U_f|x, 0\rangle = |x, f(x)\rangle}$$

First attempt:

$$U_f(H \otimes I)|00\rangle = U_f\left(\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)\right) = \underbrace{\frac{1}{\sqrt{2}}(|0, f(0)\rangle + |1, f(1)\rangle)}_{\text{Superposition of results}}$$

Only one call to U_f... but how to *read* the results?

A *physics free* introduction to quantum computing

Example: Deutsh's algorithm

Objective: To know if a 1-bit function $f : B \to B$ is constant or not.
Oracle: $U_f|x, y\rangle = |x, y \oplus f(x)\rangle$ where $\oplus$ is the addition modulo 2.

$$\boxed{U_f|x, 0\rangle = |x, f(x)\rangle}$$

First attempt:

$$U_f(H \otimes I)|00\rangle \;=\; U_f\left(\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)\right) \;=\; \underbrace{\frac{1}{\sqrt{2}}(|0, f(0)\rangle + |1, f(1)\rangle)}_{\text{Superposition of results}}$$

Only one call to U_f... but how to *read* the results?

Deutsch's algorithm:

$$(H \otimes I)U_f(H \otimes H)|00\rangle \;= \cdots = \begin{cases} \pm|0\rangle|-\rangle & \text{if } f(0) = f(1) \\ \pm|1\rangle|-\rangle & \text{if } f(0) \neq f(1) \end{cases}$$

with $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

Measuring the first qubit, we have the answer...
with only one call to the oracle

Outline

A brief and fast introduction to **quantum computing**

A brief and fast introduction to λ-**calculus**

A brief and fast introduction to **typed** λ-**calculus**

How does it relates to **intuitionistic logic**?
(a word on the Curry-Howard correspondence)

What I am doing
Algebraic calculi and **vectorial** typing
Non-determinism, a simpler problem

Work-in-progress
From non-determinism to *probabilities*

A brief and fast introduction to λ-calculus

History and intuitions

Introduced in 1936 by Alonzo Church (Alan Turing's doctoral adviser)

Motivation: Investigating the *foundations of mathematics*
(in particular, the concept of recursion)

A brief and fast introduction to λ-calculus

History and intuitions

Introduced in 1936 by Alonzo Church (Alan Turing's doctoral adviser)

Motivation: Investigating the *foundations of mathematics* (in particular, the concept of recursion)

Why we still use it

- Recursive functions are fundamental within computer science
- It is the simplest model to study properties of computation

A brief and fast introduction to λ-calculus

History and intuitions

Introduced in 1936 by Alonzo Church (Alan Turing's doctoral adviser)

Motivation: Investigating the *foundations of mathematics* (in particular, the concept of recursion)

Why we still use it

- Recursive functions are fundamental within computer science
- It is the simplest model to study properties of computation

Two fundamental simplifications

- Anonymity of functions:
  Example: $sqsum(x, y) = x^2 + y^2$
  is written anonymously as $(x, y) \mapsto x^2 + y^2$

No names needed

A brief and fast introduction to λ-calculus

History and intuitions

Introduced in 1936 by Alonzo Church (Alan Turing's doctoral adviser)

Motivation: Investigating the *foundations of mathematics*
(in particular, the concept of recursion)

Why we still use it

- Recursive functions are fundamental within computer science
- It is the simplest model to study properties of computation

Two fundamental simplifications

- Anonymity of functions:
  Example: $sqsum(x, y) = x^2 + y^2$
  is written anonymously as $(x, y) \mapsto x^2 + y^2$

  No names needed

- All the functions are in a single variable:
  Example: $(x, y) \mapsto x^2 + y^2$
  is written as $x \mapsto (y \mapsto x^2 + y^2)$

  A 2-vars function is a 1-var function, returning a 1-var function, which does the calculation

A brief and fast introduction to λ-calculus

Formalisation

Language of terms (a grammar)

$$M, N \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MN$$

- A variable $x \in \mathit{Vars}$ is a λ-term
- If M is a term, and x is a variable, $\lambda x.M$ is a term $(x \mapsto M)$
- If M and N are two terms, MN is a term (application)

These are the only possible terms.

A brief and fast introduction to λ-calculus

Formalisation

Language of terms (a grammar)

$$M, N \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MN$$

- A variable $x \in \mathit{Vars}$ is a λ-term
- If M is a term, and x is a variable, $\lambda x.M$ is a term $(x \mapsto M)$
- If M and N are two terms, MN is a term (application)

These are the only possible terms.

A rewrite rule (β-reduction)

$$(\lambda x.M)N \to M[x := N]$$

A brief and fast introduction to λ-calculus

Formalisation

Language of terms (a grammar)

$$M, N \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MN$$

- A variable $x \in \mathit{Vars}$ is a λ-term
- If M is a term, and x is a variable, $\lambda x.M$ is a term $(x \mapsto M)$
- If M and N are two terms, MN is a term (application)

These are the only possible terms.

A rewrite rule (β-reduction)

$$(\lambda x.M)N \to M[x := N]$$

Example: Let $x^2 + 1$ be a λ-term (with some encoding)

$$f(x) = x^2 + 1 \qquad \text{is written} \qquad \lambda x.x^2 + 1$$

A brief and fast introduction to λ-calculus

Formalisation

Language of terms (a grammar)

$$M, N \quad ::= \quad x \quad | \quad \lambda x.M \quad | \quad MN$$

- A variable $x \in \mathit{Vars}$ is a λ-term
- If M is a term, and x is a variable, $\lambda x.M$ is a term $(x \mapsto M)$
- If M and N are two terms, MN is a term (application)

These are the only possible terms.

A rewrite rule (β-reduction)

$$(\lambda x.M)N \to M[x := N]$$

Example: Let $x^2 + 1$ be a λ-term (with some encoding)

$$f(x) = x^2 + 1 \qquad \text{is written} \qquad \lambda x.x^2 + 1$$

$f(N)$ is written $(\lambda x.x^2 + 1)N$ which β-reduces to

$$(x^2 + 1)[x := N] \quad = \quad N^2 + 1$$

A brief and fast introduction to λ-calculus

Normal form

Not every computation ends well...

Consider $\lambda x.xx$

(the function that takes an argument, and applies it to itself)

A brief and fast introduction to λ-calculus

Normal form

Not every computation ends well...

Consider $\lambda x.xx$

(the function that takes an argument, and applies it to itself)

$$\Omega = (\lambda x.xx)(\lambda x.xx)$$

A brief and fast introduction to λ-calculus

Normal form

Not every computation ends well...

Consider $\lambda x.xx$

(the function that takes an argument, and applies it to itself)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \;\rightarrow\; xx[x := \lambda x.xx]$$

A brief and fast introduction to λ-calculus

Normal form

Not every computation ends well...

Consider $\lambda x.xx$
(the function that takes an argument, and applies it to itself)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \;\rightarrow\; xx[x := \lambda x.xx] \;=\; (\lambda x.xx)(\lambda x.xx) = \Omega$$

$$\text{So} \quad \Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \cdots$$

A brief and fast introduction to λ-calculus

Normal form

Not every computation ends well...

Consider $\lambda x.xx$
(the function that takes an argument, and applies it to itself)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \;\rightarrow\; xx[x := \lambda x.xx] \;=\; (\lambda x.xx)(\lambda x.xx) = \Omega$$

$$\text{So} \quad \Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \cdots$$

Normalisation

M is in normal form, if it does not rewrite e.g. $\lambda x.x$

A brief and fast introduction to λ-calculus

Normal form

Not every computation ends well...

Consider $\lambda x.xx$
(the function that takes an argument, and applies it to itself)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \;\rightarrow\; xx[x := \lambda x.xx] \;=\; (\lambda x.xx)(\lambda x.xx) = \Omega$$

$$\text{So} \quad \Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \cdots$$

Normalisation

M is in normal form, if it does not rewrite	e.g.	$\lambda x.x$
M is normalising if it *can* end	e.g.	$(\lambda x.\lambda y.y)\Omega$

A brief and fast introduction to λ-calculus

Normal form

Not every computation ends well...

Consider $\lambda x.xx$
(the function that takes an argument, and applies it to itself)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \;\rightarrow\; xx[x := \lambda x.xx] \;=\; (\lambda x.xx)(\lambda x.xx) = \Omega$$

$$\text{So} \quad \Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \cdots$$

Normalisation

M is in normal form, if it does not rewrite	e.g.	$\lambda x.x$
M is normalising if it *can* end	e.g.	$(\lambda x.\lambda y.y)\Omega$
M is strongly normalising if it always ends	e.g.	$(\lambda x.x)(\lambda x.x)$

A brief and fast introduction to λ-calculus

Normal form

Not every computation ends well...

Consider $\lambda x.xx$
(the function that takes an argument, and applies it to itself)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \;\rightarrow\; xx[x := \lambda x.xx] \;=\; (\lambda x.xx)(\lambda x.xx) = \Omega$$

$$\text{So} \quad \Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \cdots$$

Normalisation

M is in normal form, if it does not rewrite	e.g.	$\lambda x.x$
M is normalising if it *can* end	e.g.	$(\lambda x.\lambda y.y)\Omega$
M is strongly normalising if it always ends	e.g.	$(\lambda x.x)(\lambda x.x)$

How can we know if a λ-term is (strongly) normalising?

Outline

A brief and fast introduction to **quantum computing**

A brief and fast introduction to λ-**calculus**

A brief and fast introduction to **typed λ-calculus**

How does it relates to **intuitionistic logic**?
(a word on the Curry-Howard correspondence)

What I am doing
Algebraic calculi and **vectorial** typing
Non-determinism, a simpler problem

Work-in-progress
From non-determinism to *probabilities*

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

Terms	M, N	$::=$	x	$\mid$	$\lambda x.M$	$\mid$ MN
Types	τ, σ	$::=$	c	$\mid$	$\tau \Rightarrow \sigma$	

- c is a *base type*
- $\tau \Rightarrow \sigma$ is the functional type

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

$$\begin{array}{lrcl} \textbf{Terms} & M, N & ::= & x \mid \lambda x.M \mid MN \\ \textbf{Types} & \tau, \sigma & ::= & c \mid \tau \Rightarrow \sigma \end{array}$$

- c is a *base type*
- $\tau \Rightarrow \sigma$ is the functional type

Context: set of typed variables $\Gamma = x_1^{\tau_1}, \ldots, x_n^{\tau_n}$

$\Gamma \vdash M : \tau$ "M has type τ in context Γ"

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

Terms	M, N	$::=$	x	$\mid$	$\lambda x.M$	$\mid$ MN
Types	τ, σ	$::=$	c	$\mid$	$\tau \Rightarrow \sigma$	

- c is a *base type*
- $\tau \Rightarrow \sigma$ is the functional type

Context: set of typed variables $\Gamma = x_1^{\tau_1}, \ldots, x_n^{\tau_n}$

$\Gamma \vdash M : \tau$ "M has type τ in context Γ"

Typing rules

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

$$\begin{array}{lrcl} \textbf{Terms} & M, N & ::= & x \mid \lambda x.M \mid MN \\ \textbf{Types} & \tau, \sigma & ::= & c \mid \tau \Rightarrow \sigma \end{array}$$

- c is a *base type*
- $\tau \Rightarrow \sigma$ is the functional type

Context: set of typed variables $\Gamma = x_1^{\tau_1}, \ldots, x_n^{\tau_n}$

$\Gamma \vdash M : \tau$ "M has type τ in context Γ"

Typing rules

$$\frac{}{\Gamma, x^\tau \vdash x : \tau}\ ax$$

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

Terms	M, N	$::=$	x	$\mid$	$\lambda x.M$	$\mid$ MN
Types	τ, σ	$::=$	c	$\mid$	$\tau \Rightarrow \sigma$	

- c is a *base type*
- $\tau \Rightarrow \sigma$ is the functional type

Context: set of typed variables $\quad \Gamma = x_1^{\tau_1}, \ldots, x_n^{\tau_n}$

$\Gamma \vdash M : \tau \quad$ "M has type τ in context Γ"

Typing rules

$$\frac{}{\Gamma, x^\tau \vdash x : \tau}\, ax \qquad \frac{\Gamma, x^\tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \Rightarrow \sigma} \Rightarrow_I$$

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

Terms	M, N	$::=$	x	$\mid$	$\lambda x.M$	$\mid$ MN
Types	τ, σ	$::=$	c	$\mid$	$\tau \Rightarrow \sigma$	

- c is a *base type*
- $\tau \Rightarrow \sigma$ is the functional type

Context: set of typed variables $\quad \Gamma = x_1^{\tau_1}, \ldots, x_n^{\tau_n}$

$\Gamma \vdash M : \tau \quad$ "M has type τ in context Γ"

Typing rules

$$\frac{}{\Gamma, x^\tau \vdash x : \tau}\, ax \qquad \frac{\Gamma, x^\tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \Rightarrow \sigma} \Rightarrow_I \qquad \frac{\Gamma \vdash M : \tau \Rightarrow \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma} \Rightarrow_E$$

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

$$\begin{array}{lrcl} \textbf{Terms} & M, N & ::= & x \mid \lambda x.M \mid MN \\ \textbf{Types} & \tau, \sigma & ::= & c \mid \tau \Rightarrow \sigma \end{array}$$

- c is a *base type*
- $\tau \Rightarrow \sigma$ is the functional type

Context: set of typed variables $\Gamma = x_1^{\tau_1}, \ldots, x_n^{\tau_n}$

$\Gamma \vdash M : \tau$ "M has type τ in context Γ"

Typing rules

$$\frac{}{\Gamma, x^\tau \vdash x : \tau}\, ax \qquad \frac{\Gamma, x^\tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \Rightarrow \sigma} \Rightarrow_I \qquad \frac{\Gamma \vdash M : \tau \Rightarrow \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma} \Rightarrow_E$$

Example of type derivation

$$\frac{}{x^\tau \vdash x : \tau}\, ax$$

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

$$\begin{array}{lrcl} \textbf{Terms} & M, N & ::= & x \mid \lambda x.M \mid MN \\ \textbf{Types} & \tau, \sigma & ::= & c \mid \tau \Rightarrow \sigma \end{array}$$

- c is a *base type*
- $\tau \Rightarrow \sigma$ is the functional type

Context: set of typed variables $\quad \Gamma = x_1^{\tau_1}, \ldots, x_n^{\tau_n}$

$\Gamma \vdash M : \tau \quad$ "M has type τ in context Γ"

Typing rules

$$\frac{}{\Gamma, x^\tau \vdash x : \tau}\, ax \qquad \frac{\Gamma, x^\tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \Rightarrow \sigma} \Rightarrow_I \qquad \frac{\Gamma \vdash M : \tau \Rightarrow \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma} \Rightarrow_E$$

Example of type derivation

$$\dfrac{\dfrac{}{x^\tau \vdash x : \tau}\, ax}{\vdash \lambda x.x : \tau \Rightarrow \tau} \Rightarrow_I$$

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

$$\begin{array}{lrcl} \textbf{Terms} & M, N & ::= & x \mid \lambda x.M \mid MN \\ \textbf{Types} & \tau, \sigma & ::= & c \mid \tau \Rightarrow \sigma \end{array}$$

- c is a *base type*
- $\tau \Rightarrow \sigma$ is the functional type

Context: set of typed variables $\Gamma = x_1^{\tau_1}, \ldots, x_n^{\tau_n}$

$\Gamma \vdash M : \tau$ "M has type τ in context Γ"

Typing rules

$$\frac{}{\Gamma, x^\tau \vdash x : \tau}\, ax \qquad \frac{\Gamma, x^\tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \Rightarrow \sigma} \Rightarrow_I \qquad \frac{\Gamma \vdash M : \tau \Rightarrow \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma} \Rightarrow_E$$

Example of type derivation

$$\dfrac{\dfrac{}{x^{\tau \Rightarrow \tau} \vdash x : \tau \Rightarrow \tau}\, ax}{\vdash \lambda x.x : (\tau \Rightarrow \tau) \Rightarrow (\tau \Rightarrow \tau)} \Rightarrow_I \qquad \dfrac{\dfrac{}{x^\tau \vdash x : \tau}\, ax}{\vdash \lambda x.x : \tau \Rightarrow \tau} \Rightarrow_I$$

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

$$\begin{array}{lrcl} \textbf{Terms} & M, N & ::= & x \mid \lambda x.M \mid MN \\ \textbf{Types} & \tau, \sigma & ::= & c \mid \tau \Rightarrow \sigma \end{array}$$

- c is a *base type*
- $\tau \Rightarrow \sigma$ is the functional type

Context: set of typed variables $\quad \Gamma = x_1^{\tau_1}, \ldots, x_n^{\tau_n}$

$\Gamma \vdash M : \tau \quad$ "M has type τ in context Γ"

Typing rules

$$\frac{}{\Gamma, x^\tau \vdash x : \tau}\, ax \qquad \frac{\Gamma, x^\tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \Rightarrow \sigma}\, \Rightarrow_I \qquad \frac{\Gamma \vdash M : \tau \Rightarrow \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma}\, \Rightarrow_E$$

Example of type derivation

$$\dfrac{\dfrac{\dfrac{}{x^{\tau \Rightarrow \tau} \vdash x : \tau \Rightarrow \tau}\, ax}{\vdash \lambda x.x : (\tau \Rightarrow \tau) \Rightarrow (\tau \Rightarrow \tau)}\, \Rightarrow_I \qquad \dfrac{\dfrac{}{x^\tau \vdash x : \tau}\, ax}{\vdash \lambda x.x : \tau \Rightarrow \tau}\, \Rightarrow_I}{\vdash (\lambda x.x)(\lambda x.x) : \tau \Rightarrow \tau}\, \Rightarrow_E$$

A brief and fast introduction to typed λ-calculus

Simply types

Static way of classify λ-terms (i.e. without reducing it)

Terms	M, N	$::=$	x	$\mid$	$\lambda x.M$	$\mid$ MN
Types	τ, σ	$::=$	c	$\mid$	$\tau \Rightarrow \sigma$	

- c is a *base type*
- $\tau \Rightarrow \sigma$ is the functional type

Context: set of typed variables $\quad \Gamma = x_1^{\tau_1}, \ldots, x_n^{\tau_n}$

$\Gamma \vdash M : \tau \quad$ "M has type τ in context Γ"

Typing rules

$$\frac{}{\Gamma, x^\tau \vdash x : \tau}\, ax \qquad \frac{\Gamma, x^\tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \Rightarrow \sigma} \Rightarrow_I \qquad \frac{\Gamma \vdash M : \tau \Rightarrow \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma} \Rightarrow_E$$

Example of type derivation

$$\dfrac{\dfrac{\dfrac{}{x^{\tau \Rightarrow \tau} \vdash x : \tau \Rightarrow \tau}\, ax}{\vdash \lambda x.x : (\tau \Rightarrow \tau) \Rightarrow (\tau \Rightarrow \tau)} \Rightarrow_I \qquad \dfrac{\dfrac{}{x^\tau \vdash x : \tau}\, ax}{\vdash \lambda x.x : \tau \Rightarrow \tau} \Rightarrow_I}{\vdash (\lambda x.x)(\lambda x.x) : \tau \Rightarrow \tau} \Rightarrow_E$$

Verification: $(\lambda x.x)(\lambda x.x)$ rewrites to $\lambda x.x$ (of type $\tau \Rightarrow \tau$)

A brief and fast introduction to typed λ-calculus

Normalisation

Ω does not have a type in this theory

Ω does not have a type in this theory

Moreover...

Theorem (Strong normalisation)

If M has a type, M is strongly normalising

Ω does not have a type in this theory

Moreover...

Theorem (Strong normalisation)

If M has a type, M is strongly normalising

Slogan *"Well-typed programs cannot go wrong"* — `[R. Milner'78]`

Outline

A brief and fast introduction to **quantum computing**

A brief and fast introduction to λ-**calculus**

A brief and fast introduction to **typed** λ-**calculus**

How does it relates to **intuitionistic logic**?
(a word on the Curry-Howard correspondence)

What I am doing
Algebraic calculi and **vectorial** typing
Non-determinism, a simpler problem

Work-in-progress
From non-determinism to *probabilities*

How does it relates to intuitionistic logics?

A word on the Curry-Howard correspondence

Classical logic: a well-formed statement assumed true or false

How does it relates to intuitionistic logics?

A word on the Curry-Howard correspondence

Classical logic: a well-formed statement assumed true or false

Intuitionistic logic: a statement is true (false) if there is a constructive proof that it is true (false)

Law of excluded middle is not an axiom! (and cannot be proved neither) in intuitionistic logic

How does it relates to intuitionistic logics?

A word on the Curry-Howard correspondence

Classical logic: a well-formed statement assumed true or false

Intuitionistic logic: a statement is true (false) if there is a constructive proof that it is true (false)

Law of excluded middle is not an axiom! (and cannot be proved neither) in intuitionistic logic

Minimal intuitionistic logic

$$\frac{}{\Gamma, \tau \vdash \tau}\, ax \qquad \frac{\Gamma, \tau \vdash \sigma}{\Gamma \vdash \tau \Rightarrow \sigma} \Rightarrow_I \qquad \frac{\Gamma \vdash \tau \Rightarrow \sigma \qquad \Gamma \vdash \tau}{\Gamma \vdash \sigma} \Rightarrow_E$$

How does it relates to intuitionistic logics?

A word on the Curry-Howard correspondence

Classical logic: a well-formed statement assumed true or false

Intuitionistic logic: a statement is true (false) if there is a constructive proof that it is true (false)

Law of excluded middle is not an axiom! (and cannot be proved neither) in intuitionistic logic

Minimal intuitionistic logic

$$\frac{}{\Gamma, \tau \vdash \tau}\, ax \qquad \frac{\Gamma, \tau \vdash \sigma}{\Gamma \vdash \tau \Rightarrow \sigma} \Rightarrow_I \qquad \frac{\Gamma \vdash \tau \Rightarrow \sigma \qquad \Gamma \vdash \tau}{\Gamma \vdash \sigma} \Rightarrow_E$$

Typing rules

$$\frac{}{\Gamma, x^\tau \vdash x : \tau}\, ax \qquad \frac{\Gamma, x^\tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \Rightarrow \sigma} \Rightarrow_I \qquad \frac{\Gamma \vdash M : \tau \Rightarrow \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma} \Rightarrow_E$$

How does it relates to intuitionistic logics?

A word on the Curry-Howard correspondence

Classical logic: a well-formed statement assumed true or false

Intuitionistic logic: a statement is true (false) if there is a constructive proof that it is true (false)

Law of excluded middle is not an axiom!
(and cannot be proved neither) in intuitionistic logic

Minimal intuitionistic logic

$$\frac{}{\Gamma, \tau \vdash \tau}\, ax \qquad \frac{\Gamma, \tau \vdash \sigma}{\Gamma \vdash \tau \Rightarrow \sigma} \Rightarrow_I \qquad \frac{\Gamma \vdash \tau \Rightarrow \sigma \qquad \Gamma \vdash \tau}{\Gamma \vdash \sigma} \Rightarrow_E$$

Typing rules

$$\frac{}{\Gamma, x^\tau \vdash x : \tau}\, ax \qquad \frac{\Gamma, x^\tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \Rightarrow \sigma} \Rightarrow_I \qquad \frac{\Gamma \vdash M : \tau \Rightarrow \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma} \Rightarrow_E$$

The λ-term is the proof of the statement

Proofs... are programs!

Haskell Curry and William Howard, between 1934 and 1969

How does it relates to intuitionistic logics?

A word on the Curry-Howard correspondence

Classical logic: a well-formed statement assumed true or false

Intuitionistic logic: a statement is true (false) if there is a constructive proof that it is true (false)

Law of excluded middle is not an axiom! (and cannot be proved neither) in intuitionistic logic

Minimal intuitionistic logic

$$\frac{}{\Gamma, \tau \vdash \tau}\, ax \qquad \frac{\Gamma, \tau \vdash \sigma}{\Gamma \vdash \tau \Rightarrow \sigma}\, \Rightarrow_I \qquad \frac{\Gamma \vdash \tau \Rightarrow \sigma \qquad \Gamma \vdash \tau}{\Gamma \vdash \sigma}\, \Rightarrow_E$$

Typing rules

$$\frac{}{\Gamma, x^\tau \vdash x : \tau}\, ax \qquad \frac{\Gamma, x^\tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \Rightarrow \sigma}\, \Rightarrow_I \qquad \frac{\Gamma \vdash M : \tau \Rightarrow \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma}\, \Rightarrow_E$$

The λ-term is the proof of the statement

Proofs... are programs!

Haskell Curry and William Howard, between 1934 and 1969

More complex logics corresponds to more complex typing systems

Outline

A brief and fast introduction to **quantum computing**

A brief and fast introduction to λ-**calculus**

A brief and fast introduction to **typed** λ-**calculus**

How does it relates to **intuitionistic logic**?
(a word on the Curry-Howard correspondence)

What I am doing
Algebraic calculi and **vectorial** typing
Non-determinism, a simpler problem

Work-in-progress
From non-determinism to *probabilities*

Untyped algebraic extensions

Two origins:

- Vaux'09 (from Linear Logic)
- Arrighi,Dowek'08 (for Quantum computing)

Equivalent formalisms [Díaz-Caro,Perdrix,Tasson,Valiron'10]

Untyped algebraic extensions

Two origins:

- Vaux'09 (from Linear Logic)
- Arrighi,Dowek'08 (for Quantum computing)

Equivalent formalisms [Díaz-Caro,Perdrix,Tasson,Valiron'10]

$M, N ::= x \mid \lambda x.M \mid MN \mid M + N \mid \alpha.M \mid 0 \qquad \alpha \in (\mathcal{S}, +, \times)$, a ring.

Untyped algebraic extensions

Two origins:

- Vaux'09 (from Linear Logic)
- Arrighi,Dowek'08 (for Quantum computing)

Equivalent formalisms [Díaz-Caro,Perdrix,Tasson,Valiron'10]

$M, N ::= x \mid \lambda x.M \mid MN \mid M + N \mid \alpha.M \mid 0 \qquad \alpha \in (\mathcal{S}, +, \times)$, a ring.

$$\beta\text{-reduction: } (\lambda x.M)N \to M[x := N]$$

"Algebraic" reductions:

$$\alpha.M + \beta.N \to (\alpha + \beta).M,$$
$$\alpha.\beta.M \to (\alpha \times \beta).M,$$
$$M(N_1 + N_2) \to MN_1 + MN_2,$$
$$(M_1 + M_2)N \to M_1N + M_2N,$$
$$\dots$$

(oriented version of the axioms of vectorial spaces)

Untyped algebraic extensions

Two origins:

- Vaux'09 (from Linear Logic)
- Arrighi,Dowek'08 (for Quantum computing)

Equivalent formalisms [Díaz-Caro,Perdrix,Tasson,Valiron'10]

$M, N ::= x \mid \lambda x.M \mid MN \mid M + N \mid \alpha.M \mid 0 \qquad \alpha \in (\mathcal{S}, +, \times)$, a ring.

β-reduction: $(\lambda x.M)N \to M[x := N]$

"Algebraic" reductions:

$$\alpha.M + \beta.N \to (\alpha + \beta).M,$$
$$\alpha.\beta.M \to (\alpha \times \beta).M,$$
$$M(N_1 + N_2) \to MN_1 + MN_2,$$
$$(M_1 + M_2)N \to M_1N + M_2N,$$
$$\dots$$

(oriented version of the axioms of vectorial spaces)

Vectorial space of values

$\mathcal{B} = \{ \text{ vars. and abs. } \}$

Space of values $::= \mathrm{Span}(\mathcal{B})$

Value == result of the computation, if it ends

Example: simple encoding of quantum computing

Two base vectors: $|0\rangle = \lambda x.\lambda y.x$

$|1\rangle = \lambda x.\lambda y.y$

Example: simple encoding of quantum computing

Two base vectors: $|0\rangle = \lambda x.\lambda y.x$, $|1\rangle = \lambda x.\lambda y.y$

We want a linear map H s.t.

$$H|0\rangle \to \overbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}^{|+\rangle}$$

$$H|1\rangle \to \underbrace{\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)}_{|-\rangle}$$

Example: simple encoding of quantum computing

Two base vectors: $|0\rangle = \lambda x.\lambda y.x$, $|1\rangle = \lambda x.\lambda y.y$

We want a linear map H s.t.

$$H|0\rangle \to \overbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}^{|+\rangle}$$

$$H|1\rangle \to \underbrace{\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)}_{|-\rangle}$$

$$\boxed{H \;:=\; \lambda x.\,\{x\;\;[|+\rangle]\;\;[|-\rangle]\}}$$

Example: simple encoding of quantum computing

Two base vectors: $|0\rangle = \lambda x.\lambda y.x$, $|1\rangle = \lambda x.\lambda y.y$

We want a linear map H s.t.

$$H|0\rangle \to \overbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}^{|+\rangle}$$
$$H|1\rangle \to \underbrace{\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)}_{|-\rangle}$$

$$\boxed{H \;:=\; \lambda x.\,\{x\;\;[|+\rangle]\;\;[|-\rangle]\}}$$

$$H|+\rangle = H(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) \;\to\; \frac{1}{\sqrt{2}}(H|0\rangle + H|1\rangle) \;\to\; \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle)$$

Example: simple encoding of quantum computing

Two base vectors: $|0\rangle = \lambda x.\lambda y.x$, $|1\rangle = \lambda x.\lambda y.y$

We want a linear map H s.t.

$$H|0\rangle \to \overbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}^{|+\rangle}$$
$$H|1\rangle \to \underbrace{\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)}_{|-\rangle}$$

$$\boxed{H \;:=\; \lambda x.\,\{x\ \ [|+\rangle]\ \ [|-\rangle]\}}$$

$$H|+\rangle = H(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) \;\to\; \frac{1}{\sqrt{2}}(H|0\rangle + H|1\rangle) \;\to\; \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle)$$

$$= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) \;\to\; \frac{1}{\sqrt{2}}(\sqrt{2}|0\rangle) \;\to\; |0\rangle$$

My thesis: *On vectorial typing* [Díaz-Caro'11]

In one slide

A type system capturing the "vectorial" structure of terms

. . . to check for properties of probabilistic processes

. . . to check for properties of quantum processes

. . . or whatever application needing the structure of the vector

My thesis: *On vectorial typing* [Díaz-Caro'11]

In one slide

A type system capturing the "vectorial" structure of terms

. . . to check for properties of probabilistic processes

. . . to check for properties of quantum processes

. . . or whatever application needing the structure of the vector

Most important property of *Vectorial*

Theorem

If $\Gamma \vdash M : \sum_i \alpha_i.\tau_i$ *then* $M \downarrow = \sum_i \alpha_i.B_i$ *where* $\Gamma \vdash B_i : \tau_i$

If $M \downarrow = \sum_i \alpha_i.B_i$ *then* $\Gamma \vdash M : \sum_i \alpha_i.\tau_i + 0.\sigma$, *where* $\Gamma \vdash B_i : \tau_i$

My thesis: *On vectorial typing* [Díaz-Caro'11]

In one slide

A type system capturing the "vectorial" structure of terms
. . . to check for properties of probabilistic processes
. . . to check for properties of quantum processes
. . . or whatever application needing the structure of the vector

Most important property of *Vectorial*

Theorem

If $\Gamma \vdash M : \sum_i \alpha_i.\tau_i$ *then* $M \downarrow = \sum_i \alpha_i.B_i$ *where* $\Gamma \vdash B_i : \tau_i$
If $M \downarrow = \sum_i \alpha_i.B_i$ *then* $\Gamma \vdash M : \sum_i \alpha_i.\tau_i + 0.\sigma$, *where* $\Gamma \vdash B_i : \tau_i$

Long-term goal: a Curry-Howard approach to define a quantum logic from a quantum programming language

Simplifying the problem: non-determinism

$$M, N ::= x \mid \lambda x.M \mid MN \mid M + N$$

$$M + N \to M \qquad\qquad M + N \to N$$

Simplifying the problem: non-determinism

$$M, N \ ::= \ x \mid \lambda x.M \mid MN \mid M + N$$

$$M + N \to M \qquad\qquad M + N \to N$$

- Restricting to Linear Logic: Highly informative *quantitative* version of strong normalisation [Díaz-Caro,Manzonetto,Pagani'13]

Simplifying the problem: non-determinism

$$M, N ::= x \mid \lambda x.M \mid MN \mid M + N$$

$$M + N \to M \qquad\qquad M + N \to N$$

- Restricting to Linear Logic: Highly informative *quantitative* version of strong normalisation [Díaz-Caro,Manzonetto,Pagani'13]

- Full calculus: 2nd order propositional logic [Díaz-Caro,Petit'12]

Simplifying the problem: non-determinism

$$M, N ::= x \mid \lambda x.M \mid MN \mid M + N$$

$$M + N \to M \qquad\qquad M + N \to N$$

- Restricting to Linear Logic: Highly informative *quantitative* version of strong normalisation [Díaz-Caro,Manzonetto,Pagani'13]
- Full calculus: 2nd order propositional logic [Díaz-Caro,Petit'12]
- Naturally arise by considering some isomorphisms between propositions [Díaz-Caro,Dowek'12]

e.g.

$$\tau \wedge \sigma \equiv \sigma \wedge \tau$$
$$\tau \Rightarrow (\sigma_1 \wedge \sigma_2) \equiv (\tau \Rightarrow \sigma_1) \wedge (\tau \Rightarrow \sigma_2)$$

Outline

A brief and fast introduction to **quantum computing**

A brief and fast introduction to λ-**calculus**

A brief and fast introduction to **typed** λ-**calculus**

How does it relates to **intuitionistic logic**?
(a word on the Curry-Howard correspondence)

What I am doing
Algebraic calculi and **vectorial** typing
Non-determinism, a simpler problem

Work-in-progress
From non-determinism to *probabilities*

From non-determinism to probabilities

Work-in-progress (in collaboration with G. Dowek)

Premise: The algebraic calculus is too complex

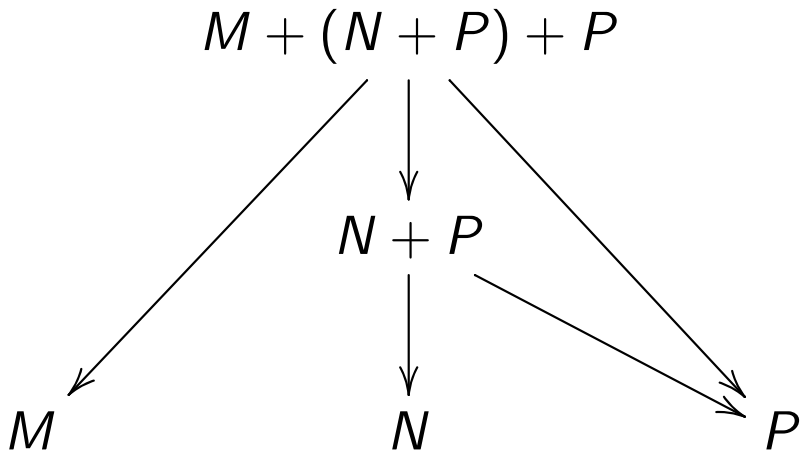
Do we really need it?

From non-determinism to probabilities

Work-in-progress (in collaboration with G. Dowek)

Premise: The algebraic calculus is too complex

Do we really need it?

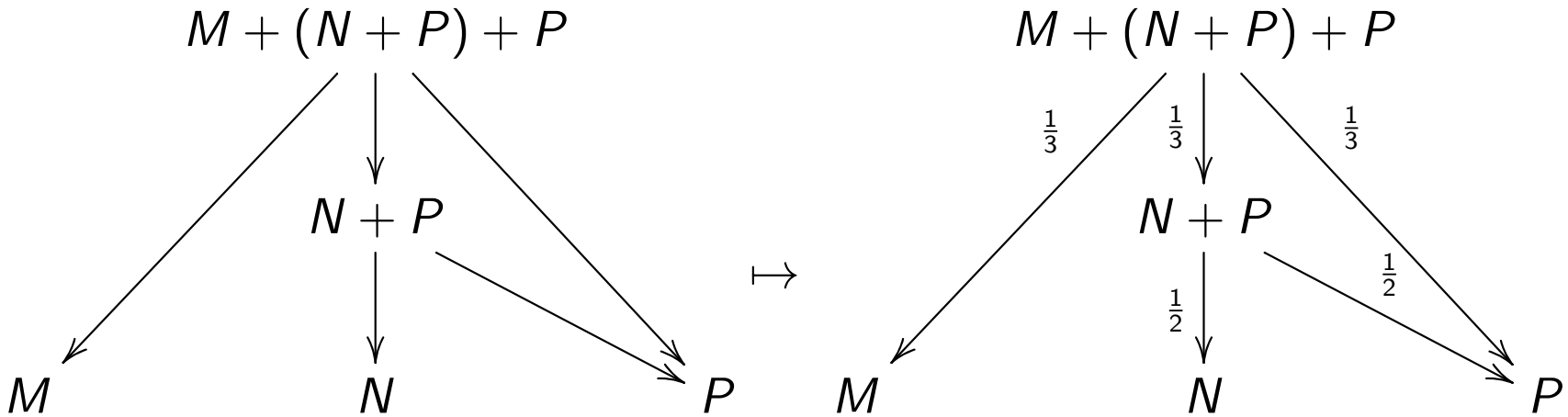


From non-determinism to probabilities

Work-in-progress (in collaboration with G. Dowek)

Premise: The algebraic calculus is too complex

Do we really need it?



$$\sim \quad \frac{1}{3}M + \frac{1}{6}N + \frac{1}{2}P$$

From non-determinism to probabilities

Generalising for any non-deterministic abstract rewrite system

Definition (Degree)

$\rho(M) = \sharp\{N \mid M \to N\}$

Definition (Oracle)

$f(M) = N$ if $M \to N$ (if $\rho(M) = n$ there are n oracles)

$\Omega =$ set of all the oracles

From non-determinism to probabilities

Generalising for any non-deterministic abstract rewrite system

Definition (Degree)

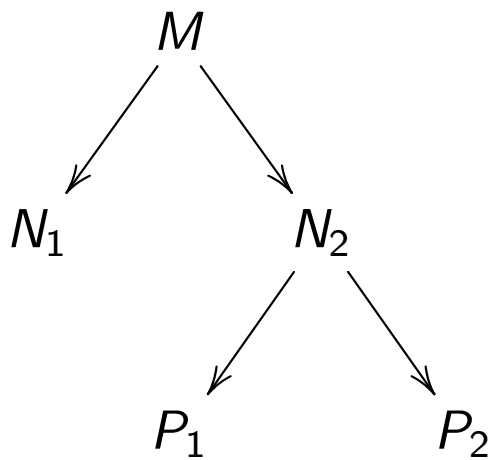
$\rho(M) = \sharp\{N \mid M \to N\}$

Definition (Oracle)

$f(M) = N$ if $M \to N$ (if $\rho(M) = n$ there are n oracles)

$\Omega =$ set of all the oracles

E.g. Rewrite system



$\Omega = \{f, g, h, i\}$, with

$f(M) = N_1$ $\quad$ $g(M) = N_1$
$f(N_2) = P_1$ $\quad$ $g(N_2) = P_2$

$h(M) = N_2$ $\quad$ $i(M) = N_2$
$h(N_2) = P_1$ $\quad$ $i(N_2) = P_2$

From non-determinism to probabilities

Generalising for any non-deterministic abstract rewrite system (cont.)

Definition (Box)

$B \subseteq \Omega$ of the form

$$B = \{f \mid f(M_1) = N_1, \ldots, f(M_n) = N_n\}$$

Definition (Probability function)

Let $B = \{f \mid f(M_1) = N_1, \ldots, f(M_n) = N_n\}$

$$m(B) = \frac{1}{\rho(M_1) \times \cdots \times \rho(M_n)}$$

$$P(S) = \inf\{\sum_{B \in \mathcal{C}} m(B) \mid \mathcal{C} \text{ is a countable family of boxes s.t. } S \subseteq \bigcup_{B \in \mathcal{C}} B\}$$

$$\mathcal{A} = \{A \subseteq \Omega \mid A \text{ is Lebesgue measurable}\}$$

From non-determinism to probabilities

Generalising for any non-deterministic abstract rewrite system (cont.)

Theorem

$(\Omega, \mathcal{A}, P)$ *is a probability space*

- Ω is the set of all possible oracles
- $\mathcal{A}$ is the set of events (Lebesgue measurable subsets of Ω)
- P is the probability function

Proof.

We show that it satisfies the Kolmogorov axioms. □

Summarising

The long-term aim is to define a quantum *computational* logic

Summarising

The long-term aim is to define a quantum *computational* logic

We have

- A λ-calculus extension able to express quantum programs
- A complex type system characterising the structure of the vectors
- A strong relation between non-determinism and second order propositional logic
- A restricted non-deterministic model related to linear logic
- An easy way to move from non-determinism to probabilities, without changing the model

Summarising

The long-term aim is to define a quantum *computational* logic

We have

- A λ-calculus extension able to express quantum programs
- A complex type system characterising the structure of the vectors
- A strong relation between non-determinism and second order propositional logic
- A restricted non-deterministic model related to linear logic
- An easy way to move from non-determinism to probabilities, without changing the model

We need

- To move from probabilities to *quantum*, without loosing the connections to logic
  - No-cloning (partially solved `[Arrighi,Dowek'08]`)
  - Measurement: we need to check for orthogonality
    $\alpha.M + \beta.N \to M$ with prob. $|\alpha|^2$, if $M \perp N$