

From a scalar type system to a vectorial type system

– Work in progress –

Alejandro Díaz-Caro

Joint work with Pablo Arrighi

Université de Grenoble
Laboratoire d'Informatique de Grenoble

May 4th, 2009. LIX (Paris)

Roadmap

Motivation

Linear-Algebraic λ -Calculus

System F

The scalar type system

Work in progress

Future work

Motivation

- ▶ Quantum Logic¹ was developed *ad hoc* before quantum computing (no clear relation with quantum programs).

¹Birkhoff, G. and J. von Neumann, *The logic of quantum mechanics*, Annals of Mathematics **37** (1936), pp. 823–843.

Motivation

- ▶ Quantum Logic¹ was developed *ad hoc* before quantum computing (no clear relation with quantum programs).
- ▶ There is a need for a logic that could aid us to isolating the reasoning behind some quantum algorithms.

¹Birkhoff, G. and J. von Neumann, *The logic of quantum mechanics*, Annals of Mathematics **37** (1936), pp. 823–843.

Motivation

- ▶ Quantum Logic¹ was developed *ad hoc* before quantum computing (no clear relation with quantum programs).
- ▶ There is a need for a logic that could aid us to isolating the reasoning behind some quantum algorithms.
- ▶ Usually the reasoning behind a program can be made to arise via a formally-defined logic from the study of type systems (Curry-Howard correspondence).

¹Birkhoff, G. and J. von Neumann, *The logic of quantum mechanics*, Annals of Mathematics **37** (1936), pp. 823–843.

- ▶ Quantum Logic¹ was developed *ad hoc* before quantum computing (no clear relation with quantum programs).
- ▶ There is a need for a logic that could aid us to isolating the reasoning behind some quantum algorithms.
- ▶ Usually the reasoning behind a program can be made to arise via a formally-defined logic from the study of type systems (Curry-Howard correspondence).

This work is a first step towards a formally-defined quantum physical logic arising via the Curry-Howard correspondence

¹Birkhoff, G. and J. von Neumann, *The logic of quantum mechanics*, Annals of Mathematics **37** (1936), pp. 823–843.

Roadmap

Motivation

Linear-Algebraic λ -Calculus

The language

Why the restrictions

System F

The scalar type system

Work in progress

Future work

Linear-Algebraic λ -Calculus²

The language

Higher-order computation

$\mathbf{t} ::= x \mid \lambda x. \mathbf{t} \mid (\mathbf{t} \mathbf{t})$ |

²Arrighi, P. and G. Dowek. *Linear-algebraic λ -calculus: higher-order, encodings and confluence*. Lecture Notes in Computer Science (RTA'08), **5117** (2008), pp. 17–31.

Linear-Algebraic λ -Calculus²

The language

Higher-order computation

$\mathbf{t} ::= x \mid \lambda x. \mathbf{t} \mid (\mathbf{t} \mathbf{t})$

Linear algebra

$\mathbf{t} + \mathbf{t} \mid \alpha. \mathbf{t} \mid \mathbf{0}$

²Arrighi, P. and G. Dowek. *Linear-algebraic λ -calculus: higher-order, encodings and confluence*. Lecture Notes in Computer Science (RTA'08), **5117** (2008), pp. 17–31.

Linear-Algebraic λ -Calculus²

The language

Higher-order computation

$\mathbf{t} ::= x \mid \lambda x. \mathbf{t} \mid (\mathbf{t} \mathbf{t})$ |

▶ $\lambda x. \mathbf{t} \mathbf{b} \rightarrow \mathbf{t}[\mathbf{b}/x] (*)$

(*) \mathbf{b} an abstraction or a variable.

Linear algebra

$\mathbf{t} + \mathbf{t} \mid \alpha. \mathbf{t} \mid \mathbf{0}$

²Arrighi, P. and G. Dowek. *Linear-algebraic λ -calculus: higher-order, encodings and confluence*. Lecture Notes in Computer Science (RTA'08), **5117** (2008), pp. 17–31.

Linear-Algebraic λ -Calculus²

The language

Higher-order computation

$\mathbf{t} ::= x \mid \lambda x. \mathbf{t} \mid (\mathbf{t} \mathbf{t})$ |

▶ $\lambda x. \mathbf{t} \mathbf{b} \rightarrow \mathbf{t}[\mathbf{b}/x] (*)$

(*) \mathbf{b} an abstraction or a variable.

(**) \mathbf{u} closed normal.

(***) \mathbf{u} and $\mathbf{u} + \mathbf{v}$ closed normal.

Linear algebra

$\mathbf{t} + \mathbf{t} \mid \alpha. \mathbf{t} \mid \mathbf{0}$

▶ Elementary rules such as
 $\mathbf{u} + \mathbf{0} \rightarrow \mathbf{u}$ and

$\alpha. (\mathbf{u} + \mathbf{v}) \rightarrow \alpha. \mathbf{u} + \alpha. \mathbf{v}.$

▶ Factorisation rules such as
 $\alpha. \mathbf{u} + \beta. \mathbf{u} \rightarrow (\alpha + \beta). \mathbf{u}.$ (**)

▶ Application rules such as
 $\mathbf{u} (\mathbf{v} + \mathbf{w}) \rightarrow (\mathbf{u} \mathbf{v}) + (\mathbf{u} \mathbf{w}).$
(***)

²Arrighi, P. and G. Dowek. *Linear-algebraic λ -calculus: higher-order, encodings and confluence*. Lecture Notes in Computer Science (RTA'08), 5117 (2008), pp. 17–31.

Linear-Algebraic λ -Calculus

Why the restrictions

Untyped λ -calculus + linear algebra $\Rightarrow \infty$

Linear-Algebraic λ -Calculus

Why the restrictions

Untyped λ -calculus + linear algebra $\Rightarrow \infty$

$$\mathbf{Yb} \equiv \lambda x.(\mathbf{b} + (x \ x)) \ \lambda x.(\mathbf{b} + (x \ x))$$

$$\mathbf{Yb} \rightarrow \mathbf{b} + \mathbf{Yb}$$

Linear-Algebraic λ -Calculus

Why the restrictions

Untyped λ -calculus + linear algebra $\Rightarrow \infty$

$$\mathbf{Yb} \equiv \lambda x.(\mathbf{b} + (x x)) \lambda x.(\mathbf{b} + (x x))$$

$$\mathbf{Yb} \rightarrow \mathbf{b} + \mathbf{Yb}$$

But whoever says infinity says trouble says...

Linear-Algebraic λ -Calculus

Why the restrictions

Untyped λ -calculus + linear algebra $\Rightarrow \infty$

$$\mathbf{Yb} \equiv \lambda x.(\mathbf{b} + (x\ x)) \lambda x.(\mathbf{b} + (x\ x))$$

$$\mathbf{Yb} \rightarrow \mathbf{b} + \mathbf{Yb}$$

But whoever says infinity says trouble says... indefinite forms.

Linear-Algebraic λ -Calculus

Why the restrictions

Untyped λ -calculus + linear algebra $\Rightarrow \infty$

$$\mathbf{Yb} \equiv \lambda x.(\mathbf{b} + (x x)) \lambda x.(\mathbf{b} + (x x))$$

$$\mathbf{Yb} \rightarrow \mathbf{b} + \mathbf{Yb}$$

But whoever says infinity says trouble says... indefinite forms.

$$\mathbf{Yb} - \mathbf{Yb} \rightarrow \mathbf{b} + \mathbf{Yb} - \mathbf{Yb} \rightarrow \mathbf{b}$$

\downarrow^*

$\mathbf{0}$

Linear-Algebraic λ -Calculus

Why the restrictions

Untyped λ -calculus + linear algebra $\Rightarrow \infty$

$$\mathbf{Yb} \equiv \lambda x.(\mathbf{b} + (x x)) \lambda x.(\mathbf{b} + (x x))$$

$$\mathbf{Yb} \rightarrow \mathbf{b} + \mathbf{Yb}$$

But whoever says infinity says trouble says... indefinite forms.

$$\mathbf{Yb} - \mathbf{Yb} \rightarrow \mathbf{b} + \mathbf{Yb} - \mathbf{Yb} \rightarrow \mathbf{b}$$

\downarrow^*

$$\mathbf{0}$$

High school teacher says we must restrict factorization rules to **finite vectors** \rightarrow *i.e.* closed-normal forms.

Roadmap

Motivation

Linear-Algebraic λ -Calculus

System F

Straightforward extension of System F ($\lambda 2^a$)

The scalar type system

Work in progress

Future work

System F

Straightforward extension of System F ($\lambda 2^{la}$)

System F rules plus simple rules to type algebraic terms

$$\frac{}{\Gamma \vdash \mathbf{0} : A} \text{ax}_0 \quad \frac{\Gamma \vdash \mathbf{u} : A \quad \Gamma \vdash \mathbf{v} : A}{\Gamma \vdash \mathbf{u} + \mathbf{v} : A} +I \quad \frac{\Gamma \vdash \mathbf{t} : A}{\Gamma \vdash \alpha.\mathbf{t} : A} \alpha I$$

System F

Straightforward extension of System F ($\lambda 2^{la}$)

System F rules plus simple rules to type algebraic terms

$$\frac{}{\Gamma \vdash \mathbf{0} : A} \text{ax}_0 \quad \frac{\Gamma \vdash \mathbf{u} : A \quad \Gamma \vdash \mathbf{v} : A}{\Gamma \vdash \mathbf{u} + \mathbf{v} : A} +I \quad \frac{\Gamma \vdash \mathbf{t} : A}{\Gamma \vdash \alpha.\mathbf{t} : A} \alpha I$$

Theorem (Strong normalization)

$\Gamma \vdash \mathbf{t} : T \Rightarrow \mathbf{t}$ is strongly normalising.

Proof. Extension of Barendregt's proof (Barendregt, H.P., "Lambda calculi with types", Handbook of Logic in Computer Science **2**, Clarendon Press, Oxford, 1992).

System F

Linear-Algebraic λ -Calculus with $\lambda 2^{la}$

Higher-order computation

$\mathbf{t} ::= x \mid \lambda x. \mathbf{t} \mid (\mathbf{t} \mathbf{t})$ |

▶ $\lambda x. \mathbf{t} \mathbf{b} \rightarrow \mathbf{t}[\mathbf{b}/x](*)$

(*) \mathbf{b} an abstraction or a variable.

Linear algebra

$\mathbf{t} + \mathbf{t} \mid \alpha. \mathbf{t} \mid \mathbf{0}$

- ▶ Elementary rules such as $\mathbf{u} + \mathbf{0} \rightarrow \mathbf{u}$ and $\alpha.(\mathbf{u} + \mathbf{v}) \rightarrow \alpha. \mathbf{u} + \alpha. \mathbf{v}$.
- ▶ Factorisation rules such as $\alpha. \mathbf{u} + \beta. \mathbf{u} \rightarrow (\alpha + \beta). \mathbf{u}$.
- ▶ Application rules such as $\mathbf{u} (\mathbf{v} + \mathbf{w}) \rightarrow (\mathbf{u} \mathbf{v}) + (\mathbf{u} \mathbf{w})$.

System F

Linear-Algebraic λ -Calculus with $\lambda 2^{la}$

Higher-order computation

$\mathbf{t} ::= x \mid \lambda x. \mathbf{t} \mid (\mathbf{t} \mathbf{t})$ |

▶ $\lambda x. \mathbf{t} \mathbf{b} \rightarrow \mathbf{t}[\mathbf{b}/x](*)$

(*) \mathbf{b} an abstraction or a variable.

Every typable term is strong normalizing

Linear algebra

$\mathbf{t} + \mathbf{t} \mid \alpha. \mathbf{t} \mid \mathbf{0}$

- ▶ Elementary rules such as $\mathbf{u} + \mathbf{0} \rightarrow \mathbf{u}$ and $\alpha.(\mathbf{u} + \mathbf{v}) \rightarrow \alpha. \mathbf{u} + \alpha. \mathbf{v}$.
- ▶ Factorisation rules such as $\alpha. \mathbf{u} + \beta. \mathbf{u} \rightarrow (\alpha + \beta). \mathbf{u}$.
- ▶ Application rules such as $\mathbf{u} (\mathbf{v} + \mathbf{w}) \rightarrow (\mathbf{u} \mathbf{v}) + (\mathbf{u} \mathbf{w})$.

System F

Linear-Algebraic λ -Calculus with $\lambda 2^{la}$

Higher-order computation

$\mathbf{t} ::= x \mid \lambda x. \mathbf{t} \mid (\mathbf{t} \mathbf{t})$ |

▶ $\lambda x. \mathbf{t} \mathbf{b} \rightarrow \mathbf{t}[\mathbf{b}/x] (*)$

(*) \mathbf{b} an abstraction or a variable.

Every typable term is strong normalizing

Hence \mathbf{Yb} is **no typable!**

Linear algebra

$\mathbf{t} + \mathbf{t} \mid \alpha. \mathbf{t} \mid \mathbf{0}$

- ▶ Elementary rules such as $\mathbf{u} + \mathbf{0} \rightarrow \mathbf{u}$ and $\alpha. (\mathbf{u} + \mathbf{v}) \rightarrow \alpha. \mathbf{u} + \alpha. \mathbf{v}$.
- ▶ Factorisation rules such as $\alpha. \mathbf{u} + \beta. \mathbf{u} \rightarrow (\alpha + \beta). \mathbf{u}$.
- ▶ Application rules such as $\mathbf{u} (\mathbf{v} + \mathbf{w}) \rightarrow (\mathbf{u} \mathbf{v}) + (\mathbf{u} \mathbf{w})$.

System F

Linear-Algebraic λ -Calculus with $\lambda 2^{la}$

Higher-order computation

$\mathbf{t} ::= x \mid \lambda x. \mathbf{t} \mid (\mathbf{t} \mathbf{t})$ |

▶ $\lambda x. \mathbf{t} \mathbf{b} \rightarrow \mathbf{t}[\mathbf{b}/x] (*)$

(*) \mathbf{b} an abstraction or a variable.

Every typable term is strong normalizing

Hence $\mathbf{Y} \mathbf{b}$ is **no typable!**

$\mathbf{t} - \mathbf{t} \rightarrow \mathbf{0}$ always, so it is not necessary to reduce \mathbf{t} first. **we can remove the closed-normal restrictions!**

Linear algebra

$\mathbf{t} + \mathbf{t} \mid \alpha. \mathbf{t} \mid \mathbf{0}$

- ▶ Elementary rules such as $\mathbf{u} + \mathbf{0} \rightarrow \mathbf{u}$ and $\alpha. (\mathbf{u} + \mathbf{v}) \rightarrow \alpha. \mathbf{u} + \alpha. \mathbf{v}$.
- ▶ Factorisation rules such as $\alpha. \mathbf{u} + \beta. \mathbf{u} \rightarrow (\alpha + \beta). \mathbf{u}$.
- ▶ Application rules such as $\mathbf{u} (\mathbf{v} + \mathbf{w}) \rightarrow (\mathbf{u} \mathbf{v}) + (\mathbf{u} \mathbf{w})$.

Roadmap

Motivation

Linear-Algebraic λ -Calculus

System F

The scalar type system

- Grammar

- Strong normalisation

- Subject reduction

- Probabilistic type system

- Logical content: No-cloning theorem

- Summary of conclusions and future work

Work in progress

The *scalar* type system

Grammar

Types grammar:

$$\mathcal{T} = \mathcal{U} \mid \forall X. \mathcal{T} \mid \alpha. \mathcal{T} \mid \bar{0},$$

$$\mathcal{U} = X \mid \mathcal{U} \rightarrow \mathcal{T} \mid \forall X. \mathcal{U}$$

where $\alpha \in \mathcal{S}$ and $(\mathcal{S}, +, \times)$ is a commutative ring.

¹Arrighi, P. and A. Díaz-Caro. *Scalar system F for linear-algebraic λ -calculus: Towards a quantum physical logic*. In Proceedings of 6th QPL, pp. 206–215, Oxford, UK, 2009.

Type inference rules

$$\frac{}{\Gamma, x: U \vdash x: U} \text{ax}[U]$$

$$\frac{\Gamma \vdash \mathbf{u}: U \rightarrow T \quad \Gamma \vdash \mathbf{v}: U}{\Gamma \vdash (\mathbf{u} \ \mathbf{v}): T} \rightarrow E$$

$$\frac{\Gamma, x: U \vdash \mathbf{t}: T}{\Gamma \vdash \lambda x \ \mathbf{t}: U \rightarrow T} \rightarrow I[U]$$

$$\frac{\Gamma \vdash \mathbf{u}: \forall X. T}{\Gamma \vdash \mathbf{u}: T[U/X]} \forall E[X := U]$$

$$\frac{\Gamma \vdash \mathbf{u}: T}{\Gamma \vdash \mathbf{u}: \forall X. T} \forall I[X] \text{ with } X \notin FV(\Gamma)$$

Type inference rules

$$\frac{}{\Gamma, x: U \vdash x: U} ax[U]$$

$$\frac{\Gamma \vdash \mathbf{u}: U \rightarrow T \quad \Gamma \vdash \mathbf{v}: U}{\Gamma \vdash (\mathbf{u} \ \mathbf{v}): T} \rightarrow E \quad \frac{\Gamma, x: U \vdash \mathbf{t}: T}{\Gamma \vdash \lambda x \ \mathbf{t}: U \rightarrow T} \rightarrow I[U]$$

$$\frac{\Gamma \vdash \mathbf{u}: \forall X. T}{\Gamma \vdash \mathbf{u}: T[U/X]} \forall E[X := U] \quad \frac{\Gamma \vdash \mathbf{u}: T}{\Gamma \vdash \mathbf{u}: \forall X. T} \forall I[X] \text{ with } X \notin FV(\Gamma)$$

$$\frac{}{\Gamma \vdash \mathbf{0}: T} ax_0 \quad \frac{\Gamma \vdash \mathbf{u}: T \quad \Gamma \vdash \mathbf{v}: T}{\Gamma \vdash \mathbf{u} + \mathbf{v}: T} +I \quad \frac{\Gamma \vdash \mathbf{u}: T}{\Gamma \vdash \alpha. \mathbf{u}: T} \alpha I$$

Type inference rules

$$\frac{}{\Gamma, x: U \vdash x: U} \text{ax}[U]$$

$$\frac{\Gamma \vdash \mathbf{u}: \alpha.(U \rightarrow T) \quad \Gamma \vdash \mathbf{v}: \beta.U}{\Gamma \vdash (\mathbf{u} \mathbf{v}): (\alpha \times \beta).T} \rightarrow E \quad \frac{\Gamma, x: U \vdash \mathbf{t}: T}{\Gamma \vdash \lambda x \mathbf{t}: U \rightarrow T} \rightarrow I[U]$$

$$\frac{\Gamma \vdash \mathbf{u}: \forall X.T}{\Gamma \vdash \mathbf{u}: T[U/X]} \forall E[X := U] \quad \frac{\Gamma \vdash \mathbf{u}: T}{\Gamma \vdash \mathbf{u}: \forall X.T} \forall I[X] \text{ with } X \notin FV(\Gamma)$$

$$\frac{}{\Gamma \vdash \mathbf{0}: \bar{0}} \text{ax}_{\bar{0}} \quad \frac{\Gamma \vdash \mathbf{u}: \alpha.T \quad \Gamma \vdash \mathbf{v}: \beta.T}{\Gamma \vdash \mathbf{u} + \mathbf{v}: (\alpha + \beta).T} +I \quad \frac{\Gamma \vdash \mathbf{u}: T}{\Gamma \vdash \alpha.\mathbf{u}: \alpha.T} sI[\alpha]$$

Where $U \in \mathcal{U}$.

The *scalar* type system

Strong normalisation

Let $(\cdot)^{\sharp}$ be a map from $\mathcal{T} \setminus \{\bar{0}\}$ to $\mathbb{T}(\lambda 2^{/a})$.

The *scalar* type system

Strong normalisation

Let $(\cdot)^{\natural}$ be a map from $\mathcal{T} \setminus \{\bar{0}\}$ to $\mathbb{T}(\lambda 2^{/a})$.

Also, let use the following notation:

$$\Gamma^{\natural} = \{(x : T^{\natural}) \mid (x : T) \in \Gamma\}$$

$$\bar{0}^{\natural} = T \text{ for whatever type } T \in \mathbb{T}(\lambda 2^{/a}).$$

The *scalar* type system

Strong normalisation

Let $(\cdot)^{\natural}$ be a map from $\mathcal{T} \setminus \{\bar{0}\}$ to $\mathbb{T}(\lambda 2^{/a})$.

Also, let us use the following notation:

$$\Gamma^{\natural} = \{(x : T^{\natural}) \mid (x : T) \in \Gamma\}$$

$$\bar{0}^{\natural} = T \text{ for whatever type } T \in \mathbb{T}(\lambda 2^{/a}).$$

Lemma (Correspondence with $\lambda 2^{/a}$)

$$\Gamma \vdash \mathbf{t} : T \Rightarrow \Gamma^{\natural} \vdash_{\lambda 2^{/a}} \mathbf{t} : T^{\natural}.$$

The *scalar* type system

Strong normalisation

Let $(\cdot)^{\natural}$ be a map from $\mathcal{T} \setminus \{\bar{0}\}$ to $\mathbb{T}(\lambda 2^{/a})$.

Also, let use the following notation:

$$\Gamma^{\natural} = \{(x : T^{\natural}) \mid (x : T) \in \Gamma\}$$

$$\bar{0}^{\natural} = T \text{ for whatever type } T \in \mathbb{T}(\lambda 2^{/a}).$$

Lemma (Correspondence with $\lambda 2^{/a}$)

$$\Gamma \vdash \mathbf{t} : T \Rightarrow \Gamma^{\natural} \vdash_{\lambda 2^{/a}} \mathbf{t} : T^{\natural}.$$

Theorem (Strong normalisation)

$$\Gamma \vdash \mathbf{t} : T \Rightarrow \mathbf{t} \text{ is strongly normalising.}$$

Proof. By previous lemma $\Gamma^{\natural} \vdash_{\lambda 2^{/a}} \mathbf{t} : T^{\natural}$, then \mathbf{t} is strong normalising.

The *scalar* type system

Subject reduction

Theorem (Subject Reduction)

Let $t \rightarrow^* t'$. Then $\Gamma \vdash t : T \Rightarrow \Gamma \vdash t' : T$

The *scalar* type system

Subject reduction

Theorem (Subject Reduction)

Let $t \rightarrow^* t'$. Then $\Gamma \vdash t : T \Rightarrow \Gamma \vdash t' : T$

Proof. There are 27 auxiliary lemmas to make the proof.

The *scalar* type system

Probabilistic type system

Conditional functions → same type on each branch.

²Di Pierro, A., C. Hanking and H. Wiklicky, *Probabilistic λ -calculus and quantitative program analysis*, Journal of Logic and Computation **15** (2005), pp. 159–179.

The *scalar* type system

Probabilistic type system

Conditional functions \rightarrow same type on each branch.

By restricting the scalars to positive reals \rightarrow **probabilistic type system**.²

²Di Pierro, A., C. Hanking and H. Wiklicky, *Probabilistic λ -calculus and quantitative program analysis*, Journal of Logic and Computation **15** (2005), pp. 159–179.

The *scalar* type system

Probabilistic type system

Conditional functions \rightarrow same type on each branch.

By restricting the scalars to positive reals \rightarrow **probabilistic type system**.²

For example, one can type functions such as

$$\lambda x \{x [\frac{1}{2}.(\text{true} + \text{false})] [\frac{1}{4}.\text{true} + \frac{3}{4}.\text{false}]\} : \mathcal{B} \rightarrow \mathcal{B}$$

with the type system serving as a guarantee that the function conserves probabilities summing to one.

²Di Pierro, A., C. Hanking and H. Wiklicky, *Probabilistic λ -calculus and quantitative program analysis*, Journal of Logic and Computation **15** (2005), pp. 159–179.

The *scalar* type system

Logical content: No-cloning theorem

Let Π a tree of typing rules and think of Π as a function from lists of sequents to proofs

The *scalar* type system

Logical content: No-cloning theorem

Let Π a tree of typing rules and think of Π as a function from lists of sequents to proofs

Theorem (No-cloning)

$\nexists \Pi$ such that $\forall A, \Pi(\Gamma \vdash A)$ has as conclusion $\Delta \vdash A \otimes A$.

Remark: Think of Π as a “universal machine”. Then this theorem says “there is not a universal cloning machine”.

The *scalar* type system

Summary of conclusions for scalar type system

- ▶ *Scalar* type system \rightarrow probabilistic type system guaranteeing probabilistic functions to be well defined.

The *scalar* type system

Summary of conclusions for scalar type system

- ▶ *Scalar* type system \rightarrow probabilistic type system guaranteeing probabilistic functions to be well defined.
- ▶ Strong normalization theorem \rightarrow most restrictions can be lifted in the reduction rules.

The *scalar* type system

Summary of conclusions for scalar type system

- ▶ *Scalar* type system \rightarrow probabilistic type system guaranteeing probabilistic functions to be well defined.
- ▶ Strong normalization theorem \rightarrow most restrictions can be lifted in the reduction rules.
- ▶ No-cloning theorem \rightarrow thinking in terms of *machines* rather than linear-logic *resources*.

The *scalar* type system

Summary of conclusions for scalar type system

- ▶ *Scalar* type system → probabilistic type system guaranteeing probabilistic functions to be well defined.
- ▶ Strong normalization theorem → most restrictions can be lifted in the reduction rules.
- ▶ No-cloning theorem → thinking in terms of *machines* rather than linear-logic *resources*.
- ▶ This is the first step towards a future **vectorial** type system.
 - ▶ Scalar type system → **magnitude** and **signs** for type vectors.
 - ▶ Future system → **direction**, (*i.e.* addition and orthogonality of types).
Then it would be possible to use amplitudes rather than probabilities.

Roadmap

Motivation

Linear-Algebraic λ -Calculus

System F

The scalar type system

Work in progress

- The additive type system

- The vectorial type system

- Orthogonality

- The vectorial[^] type system

Future work

Work in progress

The *additive* type system: Introduction

- ▶ We can think the Scalar Logic, if we takes natural numbers, as an addition between equal types, just taking in account the amount of each type.

Work in progress

The *additive* type system: Introduction

- ▶ We can think the Scalar Logic, if we takes natural numbers, as an addition between equal types, just taking in account the amount of each type.
- ▶ The following step is to figure out how addition behaves adding diferent types.

Work in progress

The *additive* type system: Introduction

- ▶ We can think the Scalar Logic, if we takes natural numbers, as an addition between equal types, just taking in account the amount of each type.
- ▶ The following step is to figure out how addition behaves adding diferent types.
- ▶ As we care just about addition between diferent types, this type system is for just a part of Linear-Algebraic Lambda Calculus, a simplified language with sums but no scalars.

Work in progress

The *additive* type system: Introduction

- ▶ We can think the Scalar Logic, if we takes natural numbers, as an addition between equal types, just taking in account the amount of each type.
- ▶ The following step is to figure out how addition behaves adding diferent types.
- ▶ As we care just about addition between diferent types, this type system is for just a part of Linear-Algebraic Lambda Calculus, a simplified language with sums but no scalars.
- ▶ Then, the reduction rules is pruned (all the rules to deal with scalars are removed).

Work in progress

The *additive* type system: Grammar

Grammar

The set of types is defined by the following abstract grammar:

$$\mathcal{T} = \mathcal{U} \mid \forall X. \mathcal{T} \mid \mathcal{T} + \mathcal{T} \mid \bar{0}$$

$$\mathcal{U} = X \mid \mathcal{U} \rightarrow \mathcal{T} \mid \forall X. \mathcal{U}$$

Work in progress

The *additive* type system: Grammar

Grammar

The set of types is defined by the following abstract grammar:

$$\mathcal{T} = \mathcal{U} \mid \forall X. \mathcal{T} \mid \mathcal{T} + \mathcal{T} \mid \bar{0}$$

$$\mathcal{U} = X \mid \mathcal{U} \rightarrow \mathcal{T} \mid \forall X. \mathcal{U}$$

Notation

$$\sum_{i=1}^0 \mathcal{T} \equiv \bar{0}$$

$$\sum_{i=1}^n \mathcal{T}_i \equiv \mathcal{T}_n + \sum_{i=1}^{n-1} \mathcal{T}_i$$

Work in progress

The *additive* type system: Type Inference Rules

Let $U \in \mathcal{U}$

$$\frac{}{\Gamma, x: U \vdash x: U} ax \qquad \frac{}{\Gamma \vdash \mathbf{0}: \bar{0}} ax_{\bar{0}}$$

$$\frac{\Gamma \vdash \mathbf{u}: \sum_{i=1}^{\alpha} (U \rightarrow T_i) \quad \Gamma \vdash \mathbf{v}: \sum_{j=1}^{\beta} U}{\Gamma \vdash (\mathbf{u} \mathbf{v}): \sum_{i=1}^{\alpha} \sum_{j=1}^{\beta} T_i} \rightarrow E$$

$$\frac{\Gamma, x: U \vdash \mathbf{t}: T}{\Gamma \vdash \lambda x \mathbf{t}: U \rightarrow T} \rightarrow I \qquad \frac{\Gamma \vdash \mathbf{u}: A \quad \Gamma \vdash \mathbf{v}: B}{\Gamma \vdash \mathbf{u} + \mathbf{v}: A + B} +I$$

$$\frac{\Gamma \vdash \mathbf{u}: \forall X. B}{\Gamma \vdash \mathbf{u}: B[U/X]} \forall E \qquad \frac{\Gamma \vdash \mathbf{u}: B}{\Gamma \vdash \mathbf{u}: \forall X. B} \forall I \text{ with } X \notin FV(\Gamma)$$

Work in progress

The *additive* type system: Subject reduction

Theorem (Subject reduction)

Let $t \rightarrow^* t'$. Then $\Gamma \vdash t : T \Rightarrow \Gamma \vdash t' : T$

Work in progress

The *additive* type system: Subject reduction

Theorem (Subject reduction)

Let $t \rightarrow^* t'$. Then $\Gamma \vdash t : T \Rightarrow \Gamma \vdash t' : T$

Proof. There are 9 auxiliary lemmas to make the proof.

It seems to be possible to map the additive logic into Intuitionistic Multiplicative Exponential Linear Logic (IMELL)³:

$$\begin{aligned}T + S &\mapsto T \otimes S \\ \bar{0} &\mapsto 1 \\ X &\mapsto X \\ U \rightarrow T &\mapsto !U \rightarrow T \\ \mathcal{U} \vdash T &\mapsto !\mathcal{U} \vdash T\end{aligned}$$

The grammar would be

$$\begin{aligned}\mathcal{T} &::= \mathcal{U} \mid \mathcal{T} \otimes \mathcal{T} \mid 1 \\ \mathcal{U} &::= X \mid !\mathcal{U} \rightarrow \mathcal{T}\end{aligned}$$

³Thanks to Olivier Laurent (ENS-Lyon) who first realize about this!

Work in progress

The *vectorial* type system: Grammar

Grammar

The set of types is defined by the following abstract grammar:

$$\mathcal{T} = \mathcal{U} \mid \forall X. \mathcal{T} \mid \alpha. \mathcal{T} \mid \mathcal{T} + \mathcal{T} \mid \bar{0}$$

$$\mathcal{U} = X \mid \mathcal{U} \rightarrow \mathcal{T} \mid \forall X. \mathcal{U}$$

where $\alpha \in \mathcal{S}$ and $(\mathcal{S}, +, \times)$ is commutative ring.

Work in progress

The *vectorial* type system: Type Inference Rules

Let $U \in \mathcal{U}$

$$\frac{}{\Gamma, x: U \vdash x: U} ax \quad \frac{}{\Gamma \vdash \mathbf{0}: \bar{0}} ax_{\bar{0}} \quad \frac{\Gamma, x: U \vdash \mathbf{t}: B}{\Gamma \vdash \lambda x \mathbf{t}: U \rightarrow B} \rightarrow I$$

$$\frac{\Gamma \vdash \mathbf{u}: \sum_{i=1}^n \alpha_i. U \rightarrow T_i \quad \Gamma \vdash \mathbf{v}: \sum_{j=1}^m \beta_j. U}{\Gamma \vdash (\mathbf{u} \mathbf{v}): \sum_{i=1}^n \sum_{j=1}^m \alpha_i \times \beta_j. T_i} \rightarrow E$$

$$\frac{\Gamma \vdash \mathbf{u}: \forall X. B}{\Gamma \vdash \mathbf{u}: B[U/X]} \forall E$$

$$\frac{\Gamma \vdash \mathbf{u}: B}{\Gamma \vdash \mathbf{u}: \forall X. B} \forall I \text{ with } X \notin FV(\Gamma)$$

$$\frac{\Gamma \vdash \mathbf{u}: A \quad \Gamma \vdash \mathbf{v}: B}{\Gamma \vdash \mathbf{u} + \mathbf{v}: A + B} +I$$

$$\frac{\Gamma \vdash \mathbf{u}: A}{\Gamma \vdash \alpha. \mathbf{u}: \alpha. A} \alpha I$$

Work in progress

The *vectorial* type system: Subject reduction

Theorem (Subject reduction)

Let $t \rightarrow^* t'$. Then $\Gamma \vdash t : T \Rightarrow \Gamma \vdash t' : T$

Work in progress

The *vectorial* type system: Subject reduction

Theorem (Subject reduction)

Let $t \rightarrow^* t'$. Then $\Gamma \vdash t : T \Rightarrow \Gamma \vdash t' : T$

Proof. There are 15 auxiliary lemmas to make the proof.

Work in progress

Orthogonality (I)

For quantum computing we need vectors with **norm one**.

Work in progress

Orthogonality (I)

For quantum computing we need vectors with **norm one**.

Example

$$|\psi\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

$$\| |\psi\rangle \| = \left| \frac{1}{\sqrt{2}} \right|^2 + \left| \frac{1}{\sqrt{2}} \right|^2 = 1$$

Work in progress

Orthogonality (I)

For quantum computing we need vectors with **norm one**.

- We write vectors as a linear combination of the base elements:

Example

$$|\psi\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

$$\| |\psi\rangle \| = \left| \frac{1}{\sqrt{2}} \right|^2 + \left| \frac{1}{\sqrt{2}} \right|^2 = 1$$

$$|\psi\rangle = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Work in progress

Orthogonality (I)

For quantum computing we need vectors with **norm one**.

- We write vectors as a linear combination of the base elements:

Example

$$|\psi\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

$$\| |\psi\rangle \| = \left| \frac{1}{\sqrt{2}} \right|^2 + \left| \frac{1}{\sqrt{2}} \right|^2 = 1$$

$$|\psi\rangle = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

- What we need to check is orthogonality, so we can define the basis of our **infinite dimensional vectorial space of terms**.

Work in progress

Orthogonality (II)

Definition (Terms inner product)

Let $\mathbf{t}, \mathbf{s}, \mathbf{r}$ be any term, and let \mathbf{u}, \mathbf{v} be terms in normal form which are not sum of terms nor a scalar times a term. We define the function 'delta' as follows:

$$\delta_{\mathbf{t}, \mathbf{s}} = \begin{cases} 0 & \text{if } \mathbf{t} \neq \mathbf{s} \vee \mathbf{t} = \mathbf{0} \vee \mathbf{s} = \mathbf{0} \\ 1 & \text{in any other case} \end{cases}$$

Then, we define the inner product between terms recursively by

$$\langle \mathbf{u} | \mathbf{v} \rangle = \langle \mathbf{v} | \mathbf{u} \rangle = \delta_{\mathbf{u}, \mathbf{v}}$$

$$\langle \mathbf{t} | \mathbf{s} \rangle = \overline{\langle \mathbf{s} | \mathbf{t} \rangle} = \langle \mathbf{t} \downarrow | \mathbf{s} \downarrow \rangle$$

$$\langle \alpha \cdot \mathbf{t} + \beta \cdot \mathbf{s} | \mathbf{r} \rangle = \alpha \times \langle \mathbf{t} | \mathbf{r} \rangle + \beta \times \langle \mathbf{s} | \mathbf{r} \rangle$$

Work in progress

Orthogonality (II)

Definition (Terms inner product)

Let $\mathbf{t}, \mathbf{s}, \mathbf{r}$ be any term, and let \mathbf{u}, \mathbf{v} be terms in normal form which are not sum of terms nor a scalar times a term. We define the function 'delta' as follows:

$$\delta_{\mathbf{t}, \mathbf{s}} = \begin{cases} 0 & \text{if } \mathbf{t} \neq \mathbf{s} \vee \mathbf{t} = \mathbf{0} \vee \mathbf{s} = \mathbf{0} \\ 1 & \text{in any other case} \end{cases}$$

Then, we define the inner product between terms recursively by

$$\langle \mathbf{u} | \mathbf{v} \rangle = \langle \mathbf{v} | \mathbf{u} \rangle = \delta_{\mathbf{u}, \mathbf{v}}$$

$$\langle \mathbf{t} | \mathbf{s} \rangle = \overline{\langle \mathbf{s} | \mathbf{t} \rangle} = \langle \mathbf{t} \downarrow | \mathbf{s} \downarrow \rangle$$

$$\langle \alpha \cdot \mathbf{t} + \beta \cdot \mathbf{s} | \mathbf{r} \rangle = \alpha \times \langle \mathbf{t} | \mathbf{r} \rangle + \beta \times \langle \mathbf{s} | \mathbf{r} \rangle$$

Definition (Terms orthogonality)

Let \mathbf{t}, \mathbf{s} be terms vectors. We say \mathbf{t} is orthogonal to \mathbf{s} , noted by $\mathbf{t} \perp \mathbf{s}$, if and only if $\langle \mathbf{t} | \mathbf{s} \rangle = 0$.

Work in progress

Orthogonality (III)

Let $\Gamma \vdash \mathbf{u} : T$ and $\Gamma \vdash \mathbf{v} : S$.

Does $\mathbf{u} \perp \mathbf{v} \Leftrightarrow T \perp S$?

(for some definition of orthogonality between types)

Work in progress

Orthogonality (III)

Let $\Gamma \vdash \mathbf{u} : T$ and $\Gamma \vdash \mathbf{v} : S$.

Does $\mathbf{u} \perp \mathbf{v} \Leftrightarrow T \perp S$?

(for some definition of orthogonality between types)

► $\mathbf{u} \perp \mathbf{v} \not\Rightarrow T \perp S$

Work in progress

Orthogonality (III)

Let $\Gamma \vdash \mathbf{u} : T$ and $\Gamma \vdash \mathbf{v} : S$.

Does $\mathbf{u} \perp \mathbf{v} \Leftrightarrow T \perp S$?

(for some definition of orthogonality between types)

► $\mathbf{u} \perp \mathbf{v} \not\Leftrightarrow T \perp S$

Counterexample

$\vdash \text{true} : \text{Bool}$

$\vdash \text{false} : \text{Bool}$

Work in progress

Orthogonality (III)

Let $\Gamma \vdash \mathbf{u} : T$ and $\Gamma \vdash \mathbf{v} : S$.

Does $\mathbf{u} \perp \mathbf{v} \Leftrightarrow T \perp S$?

(for some definition of orthogonality between types)

▶ $\mathbf{u} \perp \mathbf{v} \not\Leftrightarrow T \perp S$

▶ $T \perp S \not\Leftrightarrow \mathbf{u} \perp \mathbf{v}$

Counterexample

$\vdash \text{true} : \text{Bool}$

$\vdash \text{false} : \text{Bool}$

Work in progress

Orthogonality (III)

Let $\Gamma \vdash \mathbf{u} : T$ and $\Gamma \vdash \mathbf{v} : S$.

Does $\mathbf{u} \perp \mathbf{v} \Leftrightarrow T \perp S$?

(for some definition of orthogonality between types)

► $\mathbf{u} \perp \mathbf{v} \not\Leftrightarrow T \perp S$

Counterexample

$\vdash \text{true} : \text{Bool}$

$\vdash \text{false} : \text{Bool}$

► $T \perp S \not\Leftrightarrow \mathbf{u} \perp \mathbf{v}$

Counterexample

$y : W \vdash \lambda x y : U \rightarrow W$

$y : W \vdash \lambda x y : V \rightarrow W$

Work in progress

Orthogonality (III)

Let $\Gamma \vdash \mathbf{u} : T$ and $\Gamma \vdash \mathbf{v} : S$.

Does $\mathbf{u} \perp \mathbf{v} \Leftrightarrow T \perp S$?

(for some definition of orthogonality between types)

▶ $\mathbf{u} \perp \mathbf{v} \not\Rightarrow T \perp S$

▶ $T \perp S \not\Rightarrow \mathbf{u} \perp \mathbf{v}$

Counterexample

$\vdash \text{true} : \text{Bool}$

$\vdash \text{false} : \text{Bool}$

Counterexample

$y : W \vdash \lambda x y : U \rightarrow W$

$y : W \vdash \lambda x y : V \rightarrow W$

However, using Church-style it is possible to prove

$T \perp S \Rightarrow \mathbf{u} \perp \mathbf{v}$

Work in progress

The *vectorial*[^] type system: Introduction (I)

More requirements coming from quantum theory:

We need conditional abstractions returning orthogonal vectors

Work in progress

The *vectorial*[^] type system: Introduction (I)

More requirements coming from quantum theory:

We need conditional abstractions returning orthogonal vectors

Example

$$H = \lambda x \{x [\frac{1}{\sqrt{2}}.\text{true} + \frac{1}{\sqrt{2}}.\text{false}] [\frac{1}{\sqrt{2}}.\text{true} - \frac{1}{\sqrt{2}}.\text{false}]\}$$

Work in progress

The *vectorial*[^] type system: Introduction (I)

More requirements coming from quantum theory:

We need conditional abstractions returning orthogonal vectors

Example

$$H = \lambda x \{x [\frac{1}{\sqrt{2}}.\text{true} + \frac{1}{\sqrt{2}}.\text{false}] [\frac{1}{\sqrt{2}}.\text{true} - \frac{1}{\sqrt{2}}.\text{false}]\}$$

Problem: H has no type:

- ▶ one branch has type $\frac{1}{\sqrt{2}}.\mathbb{T} + \frac{1}{\sqrt{2}}.\mathbb{F}$
- ▶ the other branch has type $\frac{1}{\sqrt{2}}.\mathbb{T} - \frac{1}{\sqrt{2}}.\mathbb{F}$.

Work in progress

The *vectorial*[^] type system: Introduction (II)

$$H = \lambda x \{x [\frac{1}{\sqrt{2}}.\text{true} + \frac{1}{\sqrt{2}}.\text{false}] [\frac{1}{\sqrt{2}}.\text{true} - \frac{1}{\sqrt{2}}.\text{false}]\}$$

Idea!

- ▶ H has type $\mathbb{T} \rightarrow \frac{1}{\sqrt{2}}.\mathbb{T} + \frac{1}{\sqrt{2}}.\mathbb{F}$
- ▶ H also has type $\mathbb{F} \rightarrow \frac{1}{\sqrt{2}}.\mathbb{T} - \frac{1}{\sqrt{2}}.\mathbb{F}$

Work in progress

The *vectorial*[^] type system: Introduction (II)

$$H = \lambda x \{x [\frac{1}{\sqrt{2}}.\text{true} + \frac{1}{\sqrt{2}}.\text{false}] [\frac{1}{\sqrt{2}}.\text{true} - \frac{1}{\sqrt{2}}.\text{false}]\}$$

Idea!

▶ H has type $\mathbb{T} \rightarrow \frac{1}{\sqrt{2}}.\mathbb{T} + \frac{1}{\sqrt{2}}.\mathbb{F}$

▶ H also has type $\mathbb{F} \rightarrow \frac{1}{\sqrt{2}}.\mathbb{T} - \frac{1}{\sqrt{2}}.\mathbb{F}$

Then, we can modify our type system to give to H the type

$$\mathbb{T} \rightarrow \frac{1}{\sqrt{2}}.\mathbb{T} + \frac{1}{\sqrt{2}}.\mathbb{F}$$

\wedge

$$\mathbb{F} \rightarrow \frac{1}{\sqrt{2}}.\mathbb{T} - \frac{1}{\sqrt{2}}.\mathbb{F}$$

Work in progress

The *vectorial*[^] type system: Grammar (I)

- ▶ The first step is to make a *vectorial type system à la Church with \wedge symbols*.

Work in progress

The *vectorial*[^] type system: Grammar (I)

- ▶ The first step is to make a **vectorial type system à la Church with \wedge symbols.**

Grammar

$$\mathbf{t} ::= x^U \mid \bigwedge_{i=1}^n \{ [U_i] \rightarrow T_i \} . \lambda x \mathbf{t} \mid (\mathbf{t} \mathbf{t}) \mid \mathbf{0} \mid \alpha . \mathbf{t} \mid \mathbf{t} + \mathbf{t} \mid \wedge X . \mathbf{t} \mid \mathbf{t}[U]$$

where X is a type variable, $U, U_i \in \mathcal{U}$ and $T_i \in \mathcal{T}$ from the following type grammar:

$$\mathcal{T} = \mathcal{U} \mid \forall X . \mathcal{T} \mid \alpha . \mathcal{T} \mid \mathcal{T} + \mathcal{T} \mid \bar{0}$$

$$\mathcal{U} = X \mid \mathcal{U} \rightarrow \mathcal{T} \mid \forall X . \mathcal{U} \mid \bigwedge_{i=1}^n \mathcal{U}_i$$

where $\alpha \in \mathcal{S} \subseteq \mathbb{C}$ and $(\mathcal{S}, +, \times)$ form a ring.

Work in progress

The *vectorial*[^] type system: Grammar (II)

Symplifying the notations:

Work in progress

The *vectorial*[^] type system: Grammar (II)

Simplifying the notations:

$$\blacktriangleright \bigwedge_{i=1}^n \{ [U_i] \rightarrow T_i \}. \lambda x \mathbf{t} \quad \rightsquigarrow \quad \lambda x \bigwedge_{i=1}^n [U_i] \rightarrow T_i \mathbf{t}.$$

Work in progress

The *vectorial*[^] type system: Grammar (II)

Simplifying the notations:

- ▶ $\bigwedge_{i=1}^n \{[U_i] \rightarrow T_i\} . \lambda x \mathbf{t} \rightsquigarrow \lambda x^{i=1} \bigwedge [U_i] \rightarrow T_i \mathbf{t}.$
- ▶ So, the terms grammar is:

$$\mathbf{t} ::= x^U \mid \lambda x^{i=1} \bigwedge [U_i] \rightarrow T_i \mathbf{t} \mid (\mathbf{t} \mathbf{t}) \mid \mathbf{0} \mid \alpha . \mathbf{t} \mid \mathbf{t} + \mathbf{t} \mid \Lambda X . \mathbf{t} \mid \mathbf{t}[U]$$

Work in progress

The *vectorial*[^] type system: Grammar (II)

Simplifying the notations:

- ▶ $\bigwedge_{i=1}^n \{[U_i] \rightarrow T_i\} . \lambda x \mathbf{t} \rightsquigarrow \lambda x^{i=1} \bigwedge [U_i] \rightarrow T_i \mathbf{t}.$
- ▶ So, the terms grammar is:

$$\mathbf{t} ::= x^U \mid \lambda x^{i=1} \bigwedge [U_i] \rightarrow T_i \mathbf{t} \mid (\mathbf{t} \mathbf{t}) \mid \mathbf{0} \mid \alpha . \mathbf{t} \mid \mathbf{t} + \mathbf{t} \mid \Lambda X . \mathbf{t} \mid \mathbf{t}[U]$$

- ▶ In the type inference rules, instead of

$$\Gamma \vdash \lambda x^{i=1} \bigwedge [U_i] \rightarrow T_i \mathbf{t} : \bigwedge_{i=1}^n U_i \rightarrow T_i$$

we can just write

$$\Gamma \vdash \lambda x \mathbf{t} : \bigwedge_{i=1}^n [U_i] \rightarrow T_i$$

Work in progress

The *vectorial*[^] type system: Type inference rules (I)

$$\frac{}{\Gamma, x: U \vdash x: U} \text{ax} \quad \frac{}{\Gamma \vdash \mathbf{0}: \bar{0}} \text{ax}_{\bar{0}} \quad \frac{\Gamma, x: U \vdash \mathbf{t}: T}{\Gamma \vdash \lambda x \mathbf{t}: [U] \rightarrow T} \rightarrow I$$

$$\frac{\Gamma \vdash \mathbf{u}: \sum_{i=1}^n \alpha_i \cdot \bigwedge_{j=1}^m U_j \rightarrow T_{ij} \quad \Gamma \vdash \mathbf{v}: \sum_{j=1}^{m'} \beta_j \cdot U_j}{\Gamma \vdash (\mathbf{u} \mathbf{v}): \sum_{i=1}^n \sum_{j=1}^{m'} \alpha_i \times \beta_j \cdot T_{ij}} \rightarrow E \text{ with } m' \leq m$$

$$\frac{\Gamma \vdash \mathbf{u}: \forall X. T}{\Gamma \vdash \mathbf{u}[U]: T[U/X]} \forall E \quad \frac{\Gamma \vdash \mathbf{u}: T}{\Gamma \vdash \Lambda X. \mathbf{u}: \forall X. T} \forall I \text{ with } X \notin FV(\Gamma)$$

$$\frac{\Gamma \vdash \mathbf{u}: A \quad \Gamma \vdash \mathbf{v}: B}{\Gamma \vdash \mathbf{u} + \mathbf{v}: A + B} +I \quad \frac{\Gamma \vdash \mathbf{u}: T}{\Gamma \vdash \alpha. \mathbf{u}: \alpha. T} \alpha I$$

Continues...

Work in progress

The *vectorial*[^] type system: Type inference rules (II)

$$\frac{\Gamma \vdash \lambda x \mathbf{t}: \bigwedge_{i=1}^n ([U_i] \rightarrow T_i) \quad \Gamma \vdash \lambda x \mathbf{t}: \bigwedge_{j=1}^m ([V_j] \rightarrow T'_j)}{\Gamma \vdash \lambda x \mathbf{t}: \bigwedge_{i=1}^n ([U_i] \rightarrow T_i) \wedge \bigwedge_{j=1}^m ([V_j] \rightarrow T'_j)} \wedge I$$

Work in progress

The vectorial^\wedge type system: Subject reduction

Theorem (Subject reduction)

Let $t \rightarrow^* t'$. Then $\Gamma \vdash t : T \Rightarrow \Gamma \vdash t' : T$

Work in progress

The *vectorial*[^] type system: Subject reduction

Theorem (Subject reduction)

Let $t \rightarrow^ t'$. Then $\Gamma \vdash t : T \Rightarrow \Gamma \vdash t' : T$*

Proof. There are 15 auxiliary lemmas to make the proof. (we need to prove 3 of them yet)

Roadmap

Motivation

Linear-Algebraic λ -Calculus

System F

The scalar type system

Work in progress

Future work

The quantum type system

Furute work

The *quantum* type system

- ▶ Once $vectorial^\wedge$ have been proved, we need to add restrictions to the type inference rules asking for orthogonality.

Furute work

The *quantum* type system

- ▶ Once *vectorial*[^] have been proved, we need to add restrictions to the type inference rules asking for orthogonality.
- ▶ Also it is needed to prove unitarity of the abstractions

Furute work

The *quantum* type system

- ▶ Once *vectorial*[^] have been proved, we need to add restrictions to the type inference rules asking for orthogonality.
- ▶ Also it is needed to prove unitarity of the abstractions
- ▶ Then, we have to interpret the logic behind this type system