

Scalar System F for Linear-Algebraic λ -Calculus

Towards a Quantum Physical Logic

Pablo Arrighi and Alejandro Díaz-Caro

{pablo.arrighi, alejandro.diaz-caro}@imag.fr
 Université de Grenoble, LIG

April 9th, 2009. VI Quantum Physics and Logic. Oxford University

- ▶ 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.

Linear-Algebraic λ -Calculus²

Higher-order computation

$t ::= x \mid \lambda x. t \mid (t t)$

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

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

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

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

Linear algebra

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

▶ 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 $\mathbf{u} (\mathbf{v} + \mathbf{w}) \rightarrow (\mathbf{u} \mathbf{v}) + (\mathbf{u} \mathbf{w})$. (***)

Linear-Algebraic λ -Calculus

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.

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.

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).

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!

$\mathbf{t} - \mathbf{t} \rightarrow \mathbf{0}$ 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 $\mathbf{u} (\mathbf{v} + \mathbf{w}) \rightarrow (\mathbf{u} \mathbf{v}) + (\mathbf{u} \mathbf{w})$.

The scalar type system (I)

Types grammar:

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

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

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

The scalar type system (II)

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

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

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

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

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

Strong normalisation

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

Also, let use the following notation:

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

$\perp^{\natural} = T$ 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)

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

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

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.

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.

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”.

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.

Summary of conclusions and future work

- ▶ *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 amplituds rather than probabilities.