A type system for the vectorial aspects of the linear-algebraic lambda-calculus

Pablo Arrighi[1,2] Alejandro Díaz-Caro[1] Benoît Valiron[3,4]

[1]Université de Grenoble, LIG, France

[2]École Normale Supérieure de Lyon, LIP, France

[3]Université de Paris-Nord, LIPN, France

[4]University of Pennsylvania, USA

7^{th} DCM • July 3, 2011 • Zurich, Switzerland

$$M, N ::= x \mid \lambda x.M \mid (M)N \mid M + N \mid \alpha.M \mid 0$$

Beta reduction:

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

"Algebraic" reductions:

$$\alpha.M + \beta.M \to (\alpha + \beta).M,$$
$$(M)(N_1 + N_2) \to (M)N_1 + (M)N_2,$$
$$\dots$$
$$\dots$$
$$\dots$$

(oriented version of the axioms of vectorial spaces)

Two origins:

- Differential λ-calculus: capturing linearity *à la* Linear Logic

$\to$ *Removing the differential operator*: Algebraic λ-calculus (λ_{alg}) [Vaux'09]

- Quantum computing: superposition of programs

$\to$ *Linearity as in algebra*: Linear-algebraic λ-calculus (λ_{lin})

[Arrighi,Dowek'08]

$$M, N ::= x \mid \lambda x.M \mid (M)N \mid M + N \mid \alpha.M \mid 0$$

Beta reduction:

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

“Algebraic” reductions:

$$\alpha.M + \beta.M \to (\alpha + \beta).M,$$
$$(M)(N_1 + N_2) \to (M)N_1 + (M)N_2,$$
$$\ldots$$
$$\ldots$$
$$\ldots$$

(oriented version of the axioms of vectorial spaces)

	λ_{alg}	λ_{lin}
Origin	Linear Logic	Quantum computing
Strategy	Call-by-name	Call-by-value
Algebraic part	Equalities	Rewrite system

An infinite dimensional vectorial space of values

$$\mathcal{B} = \{M_i : M_i \text{ is a variable or abstraction }\}$$

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

(Now we should call λ_{lin}'s strategy: "call-by-base")

Why would it be interesting?

- Several theories using the concept of linear-combination of terms
  quantum, probabilistic, non-deterministic models, ...
- "Why would vector spaces be an interesting theory?"
  Many applications and moreover, interesting by itself!

Aim of the current work:

A type system capturing the "vectorial" structure of terms

... to check for probability distributions

... or "quantumness" of the term

... or whatever application needing the structure of the vector in normal form

... a Curry-Howard approach to defining Fuzzy/Quantum/Probabilistic logics from Fuzzy/Quantum/Probabilistic programming languages.

The *Scalar* Type System [Arrighi,Díaz-Caro'09]

A polymorphic type system *tracking scalars*:

$$\frac{\Gamma \vdash M : T}{\Gamma \vdash \alpha.M : \alpha.T}$$

- Barycentric restrictions
- Characterises the "amount" of terms

The *Additive* Type System [Díaz-Caro,Petit'10]

A polymorphic type system *with sums*:

$$\frac{\Gamma \vdash M : T \qquad \Gamma \vdash N : R}{\Gamma \vdash M + N : T + R}$$

- Sums $\sim$ Assoc., comm. pairs
- distributive w.r.t. application

Can we combine them?

The *Vectorial* Type System

Types:

$$\begin{array}{rcl} T, R, S & := & U \mid T + R \mid \alpha.T \\ U, V, W & := & X \mid U \to T \mid \forall X.U \end{array}$$

(U, V, W reflect the basis terms)

Equivalences:

$$\begin{array}{rcl} 1.T & \equiv & T \\ \alpha.(\beta.T) & \equiv & (\alpha \times \beta).T \\ \alpha.T + \alpha.R & \equiv & \alpha.(T + R) \\ \alpha.T + \beta.T & \equiv & (\alpha + \beta).T \\ T + R & \equiv & R + T \\ T + (R + S) & \equiv & (T + R) + S \end{array}$$

(reflect the vectorial spaces axioms)

The factorisation rule problem

$$\frac{\Gamma \vdash M : T \qquad \Gamma \vdash M : T'}{\Gamma \vdash \alpha.M + \beta.M : \alpha.T + \beta.T'}$$

- However, $\alpha.M + \beta.M \to (\alpha + \beta).M$
- In general $\alpha.T + \beta.T' \neq (\alpha + \beta).T \neq (\alpha + \beta).T'$

(and since we are working in System F, there is no principal types neither)

Several possible solutions:

- Remove factorisation rule (Done. SR and SN both work)
  - $+$ in scalars not used anymore. Scalars $\Rightarrow$ Monoid
  - It works!... but it is no so expressive ("vectorial" structure lost)

Several possible solutions:

- Remove factorisation rule (Done. SR and SN both work)
  - $+$ in scalars not used anymore. Scalars $\Rightarrow$ Monoid
  - It works!... but it is no so expressive ("vectorial" structure lost)

- Add several typing rules to allow typing $(\alpha + \beta).M$ with $\alpha.T + \beta.T'$
  - As soon as we add one, we have to add many to make it work
  - Too complex and inelegant (subject reduction by axiom)

Several possible solutions:

- Remove factorisation rule (Done. SR and SN both work)
  - $+$ in scalars not used anymore. Scalars $\Rightarrow$ Monoid
  - It works!... but it is no so expressive ("vectorial" structure lost)

- Add several typing rules to allow typing $(\alpha + \beta).M$ with $\alpha.T + \beta.T'$
  - As soon as we add one, we have to add many to make it work
  - Too complex and inelegant (subject reduction by axiom)

- Church style
  - Seems to be the natural solution
  - Big complexity with polymorphism and distributivity

Several possible solutions:

- Remove factorisation rule (Done. SR and SN both work)
  - $+$ in scalars not used anymore. Scalars $\Rightarrow$ Monoid
  - It works!... but it is no so expressive ("vectorial" structure lost)
- Add several typing rules to allow typing $(\alpha + \beta).M$ with $\alpha.T + \beta.T'$
  - As soon as we add one, we have to add many to make it work
  - Too complex and inelegant (subject reduction by axiom)
- Church style
  - Seems to be the natural solution
  - Big complexity with polymorphism and distributivity
- Weak subject reduction (this work)
  - What is the best we can get in Curry style?

Typing rules

$$\frac{}{\Gamma, x : U \vdash x : U}\ ax \qquad \frac{\Gamma \vdash M : T}{\Gamma \vdash 0 : 0.T}\ 0_I \qquad \frac{\Gamma \vdash M : T}{\Gamma \vdash \alpha.M : \alpha.T}\ \alpha_I$$

$$\frac{\Gamma \vdash M : \sum_{i=1}^{n} \alpha_i.\forall \vec{X}.(U \to T_i) \qquad \Gamma \vdash N : \sum_{j=1}^{m} \beta_j.V_j \qquad {\scriptstyle \forall V_j, \exists \vec{W}_j / U[\vec{W}_j/\vec{X}] = V_j}}{\Gamma \vdash (M)N : \sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_i \times \beta_j.T_i[\vec{W}_j/\vec{X}]}\ \to_E$$

$$\frac{\Gamma, x : U \vdash M : T}{\Gamma \vdash \lambda x.M : U \to T}\ \to_I \qquad \frac{\Gamma \vdash M : T \qquad \Gamma \vdash N : R}{\Gamma \vdash M + N : T + R}\ +_I$$

$$\frac{\Gamma \vdash M : U \qquad {\scriptstyle X \notin FV(\Gamma)}}{\Gamma \vdash M : \forall X.U}\ \forall_I \qquad \frac{\Gamma \vdash M : \forall X.U}{\Gamma \vdash M : U[V/X]}\ \forall_E$$

$$(\alpha+\beta).T \sqsubseteq \alpha.T + \beta.T' \qquad \text{if } \exists M \ / \ \Gamma \vdash M : T \text{ and } \Gamma \vdash M : T'$$

(and its contextual closure)

$$(\alpha+\beta).T \sqsubseteq \alpha.T + \beta.T' \qquad \text{if } \exists M \ / \ \Gamma \vdash M : T \text{ and } \Gamma \vdash M : T'$$

(and its contextual closure)

Theorem (A weak subject reduction)

If $\Gamma \vdash M : T$ and $M \to_R N$, then

- *if R is not a factorisation rule: $\Gamma \vdash N : T$*
- *if R is a factorisation rule: $\exists S \sqsubseteq T \ / \ \Gamma \vdash N : S$*

$$(\alpha+\beta).T \sqsubseteq \alpha.T+\beta.T' \qquad \text{if } \exists M \;/\; \Gamma\vdash M:T \text{ and } \Gamma\vdash M:T'$$
(and its contextual closure)

Theorem (A weak subject reduction)

If $\Gamma \vdash M : T$ and $M \to_R N$, then

- *if R is not a factorisation rule: $\Gamma \vdash N : T$*
- *if R is a factorisation rule: $\exists S \sqsubseteq T \;/\; \Gamma \vdash N : S$*

How weak?

Let $M \to N$,

Subject reduction

$\Gamma \vdash M : T \Rightarrow \Gamma \vdash N : T$

Subtyping

$\Gamma \vdash M : T \Rightarrow \Gamma \vdash N : S$, but $S \leq T$, so $\Gamma \vdash N : T$

Our theorem

$\Gamma \vdash M : T \Rightarrow \Gamma \vdash N : S$, and $S \sqsubseteq T$

Confluence and Strong normalisation

In the original untyped setting: "confluence by restrictions":

$$Y_B = (\lambda x.(B + (x)x))\lambda x.(B + (x)x)$$

$$Y_B \to B + Y_B \to B + B + Y_B \to \dots$$

Confluence and Strong normalisation

In the original untyped setting: "confluence by restrictions":

$$Y_B = (\lambda x.(B + (x)x))\lambda x.(B + (x)x)$$

$$Y_B \to B + Y_B \to B + B + Y_B \to \dots$$

$$\begin{array}{ccccc} Y_B + (-1).Y_B & \longrightarrow & (1-1).Y_B & \longrightarrow^* & 0 \\ \downarrow & & & & \\ B + Y_B + (-1).Y_B & & & & \\ \downarrow_* & & & & \\ B & & & & \end{array}$$

Solution in the untyped setting:
$\alpha.M + \beta.M \to (\alpha + \beta).M$
only if M is closed-normal

Confluence and Strong normalisation

In the original untyped setting: "confluence by restrictions":

$$Y_B = (\lambda x.(B + (x)x))\lambda x.(B + (x)x)$$

$$Y_B \to B + Y_B \to B + B + Y_B \to \dots$$

$$\begin{array}{ccccc} Y_B + (-1).Y_B & \longrightarrow & (1-1).Y_B & \longrightarrow^* & 0 \\ \downarrow & & & & \\ B + Y_B + (-1).Y_B & & & & \\ \downarrow_* & & & & \\ B & & & & \end{array}$$

Solution in the untyped setting:
$\alpha.M + \beta.M \to (\alpha + \beta).M$
only if M is closed-normal

In the typed setting: **Strong normalisation solves the problem**

Theorem (Strong normalisation)

$\Gamma \vdash M : T \Rightarrow M$ *strongly normalising.*

Proof.

Reducibility candidates method.

Main difficulty: Show that

$$\{M_i\}_i \text{ strongly normalizing} \Rightarrow \sum_i \alpha_i.M_i \text{ strongly normalizing}$$

Done by using a measurement on terms decreasing on algebraic rewrites. □

Theorem (Confluence)

$$\forall M \;/\; \Gamma \vdash M : T \qquad \begin{array}{l} M \to^* N_1 \\ M \to^* N_2 \end{array} \Rightarrow \exists L \textit{ such that } \begin{array}{l} N_1 \to^* L \\ N_2 \to^* L \end{array}$$

Proof.

1) **local confluence**: $\begin{array}{l} M \to N_1 \\ M \to N_2 \end{array} \Rightarrow \exists L \text{ such that } \begin{array}{l} N_1 \to^* L \\ N_2 \to^* L \end{array}$

- Algebraic fragment: Coq proof
- Beta-reduction: Straightforward extension
- Commutation: Induction

2) Local confluence + Strong normalisation $\Rightarrow$ Confluence [TeReSe'03] □

Expressing matrices and vectors

Two base vectors: $\begin{array}{l}\textbf{true} = \lambda x.\lambda y.x \\ \textbf{false} = \lambda x.\lambda y.y\end{array}$

Expressing matrices and vectors

Two base vectors: $\begin{array}{l} \textbf{true} = \lambda x.\lambda y.x \\ \textbf{false} = \lambda x.\lambda y.y \end{array}$

Their types: $\begin{array}{l} \mathbb{T} = \forall XY.X \to Y \to X \\ \mathbb{F} = \forall XY.X \to Y \to Y \end{array}$

$$\vdash \alpha.\textbf{true} + \beta.\textbf{false} : \alpha.\mathbb{T} + \beta.\mathbb{F}$$

Expressing matrices and vectors

Two base vectors: $$\begin{aligned}\textbf{true} &= \lambda x.\lambda y.x \\ \textbf{false} &= \lambda x.\lambda y.y\end{aligned}$$

Their types: $$\begin{aligned}\mathbb{T} &= \forall XY.X \to Y \to X \\ \mathbb{F} &= \forall XY.X \to Y \to Y\end{aligned}$$

$$\vdash \alpha.\textbf{true} + \beta.\textbf{false} : \alpha.\mathbb{T} + \beta.\mathbb{F}$$

Linear map **U** s.t. $$\begin{aligned}(\mathbf{U})\textbf{true} &= a.\textbf{true} + b.\textbf{false} \\ (\mathbf{U})\textbf{false} &= c.\textbf{true} + d.\textbf{false}\end{aligned}$$

Expressing matrices and vectors

Two base vectors: $\begin{array}{l}\textbf{true} = \lambda x.\lambda y.x \\ \textbf{false} = \lambda x.\lambda y.y\end{array}$

Their types: $\begin{array}{l}\mathbb{T} = \forall XY.X \to Y \to X \\ \mathbb{F} = \forall XY.X \to Y \to Y\end{array}$

$$\vdash \alpha.\textbf{true} + \beta.\textbf{false} : \alpha.\mathbb{T} + \beta.\mathbb{F}$$

Linear map $\mathbf{U}$ s.t. $\begin{array}{l}(\mathbf{U})\textbf{true} = a.\textbf{true} + b.\textbf{false} \\ (\mathbf{U})\textbf{false} = c.\textbf{true} + d.\textbf{false}\end{array}$

$$\mathbf{U} := \lambda x.\{((x)[a.\textbf{true} + b.\textbf{false}])[c.\textbf{true} + d.\textbf{false}]\}$$

$$\left|\ \text{with} \quad \begin{array}{r l}[M] & := \lambda z.M \\ \{M\} & := (M)\_ \\ \{[M]\} & \to M\end{array}\right.$$

$$\vdash \mathbf{U} : \forall X.((I \to (a.\mathbb{T} + b.\mathbb{F})) \to (I \to (c.\mathbb{T} + d.\mathbb{F})) \to X) \to X$$

Contributions

- Scalar $\cup$ Additive ("AC, distributive pairs")
  $\Rightarrow$ linear-combination of types
- The typing gives the information of
  "how much the scalars sums" in the normal form
- Weak SR
  $\Rightarrow$ Church style captures better the vectorial structure
- Strong normalisation
  $\Rightarrow$ Confluence without restrictions
- Representation of matrices and vectors