Algebra and Coalgebra in the Light Affine Lambda Calculus

Marco Gaboardi
Suny Buffalo

Romain Péchoux
Université de Lorraine

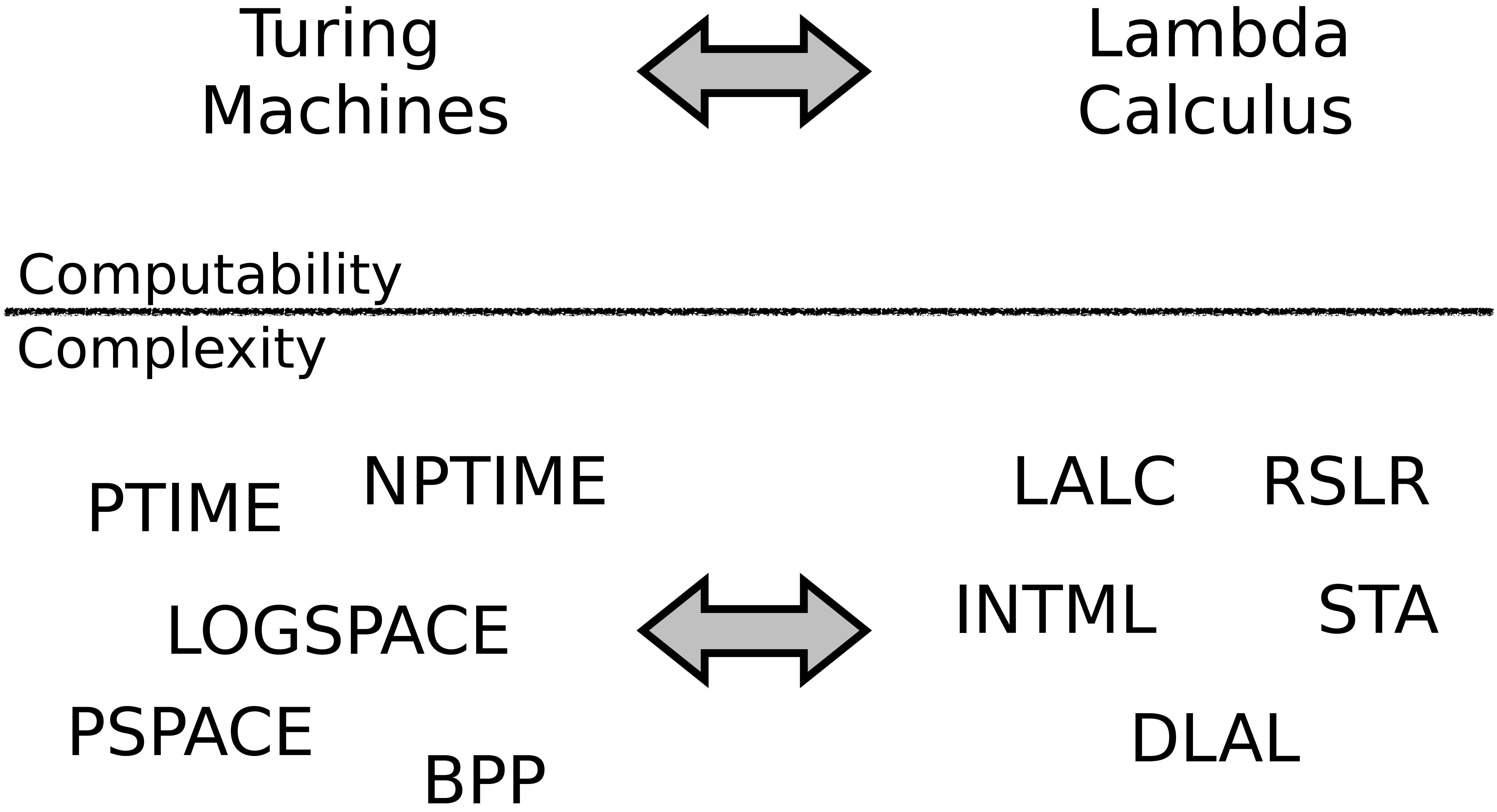
Computability

Complexity

Turing Machines ⟺ Lambda Calculus

Computability

Complexity



Turing Machines

Lambda Calculus

Computability

Complexity

PTIME

NPTIME

LOGSPACE

PSPACE

BPP

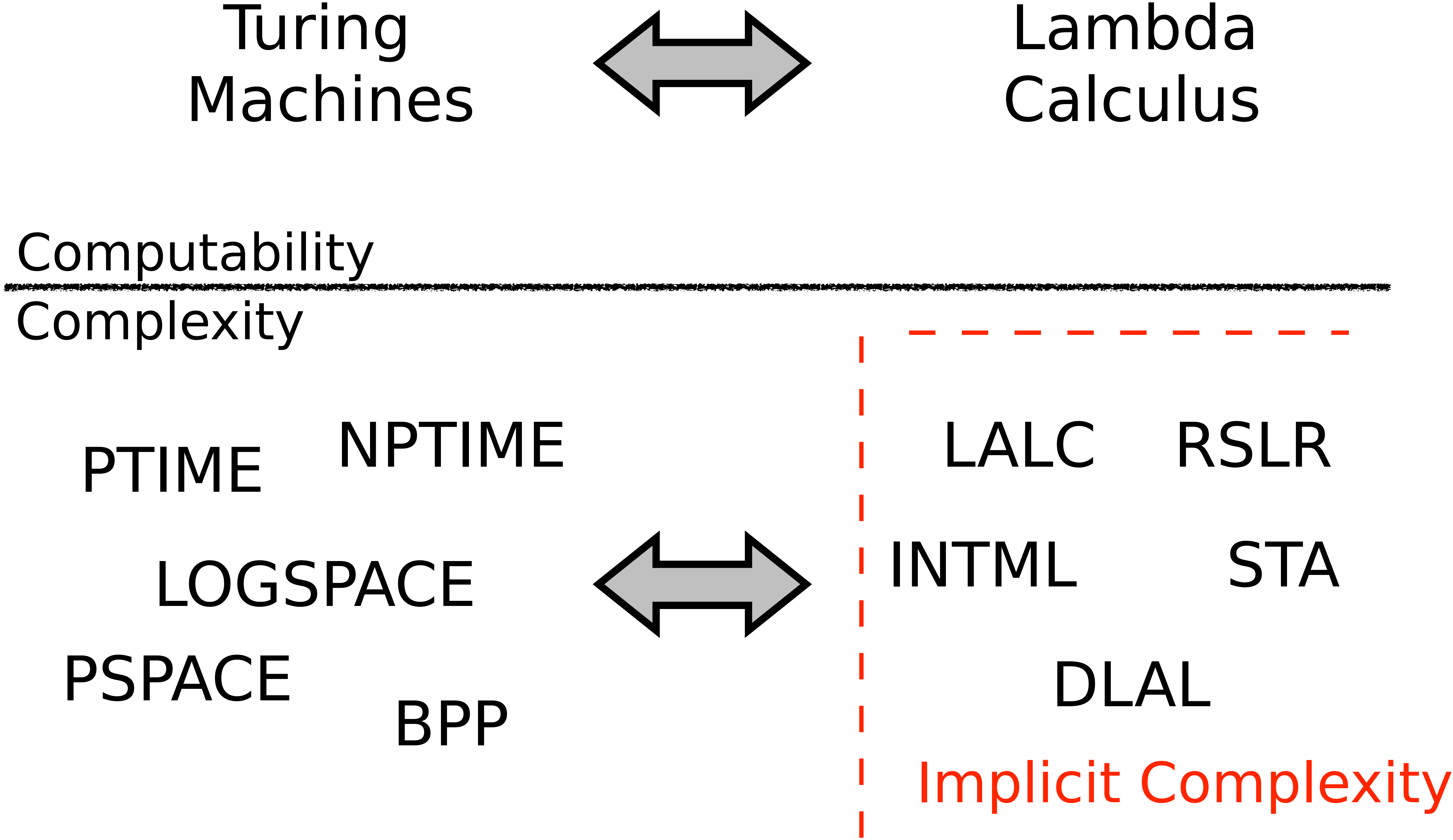
LALC

RSLR

INTML

STA

DLAL



Turing Machines

Lambda Calculus

Computability

Complexity

PTIME

NPTIME

LOGSPACE

PSPACE

BPP

LALC

RSLR

INTML

STA

DLAL

Implicit Complexity

Turing Machines

Lambda Calculus

Computability

Complexity

NPTIME
PTIME
LOGSPACE
PSPACE
BPP

LALC
RSLR
INTML
STA
DLAL

(Light Affine Lambda Calculus)

LALC

⇕

PTIME

Where these languages could help?

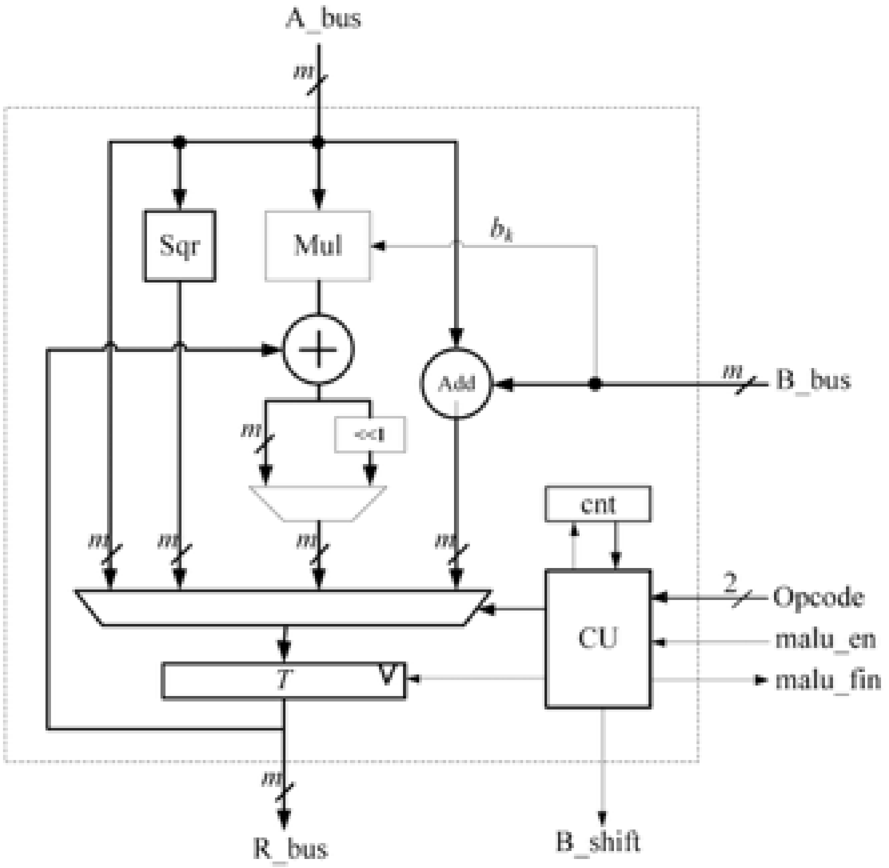
Where these languages could help?

Resource
Analysis

Where these languages could help?

Resource Analysis

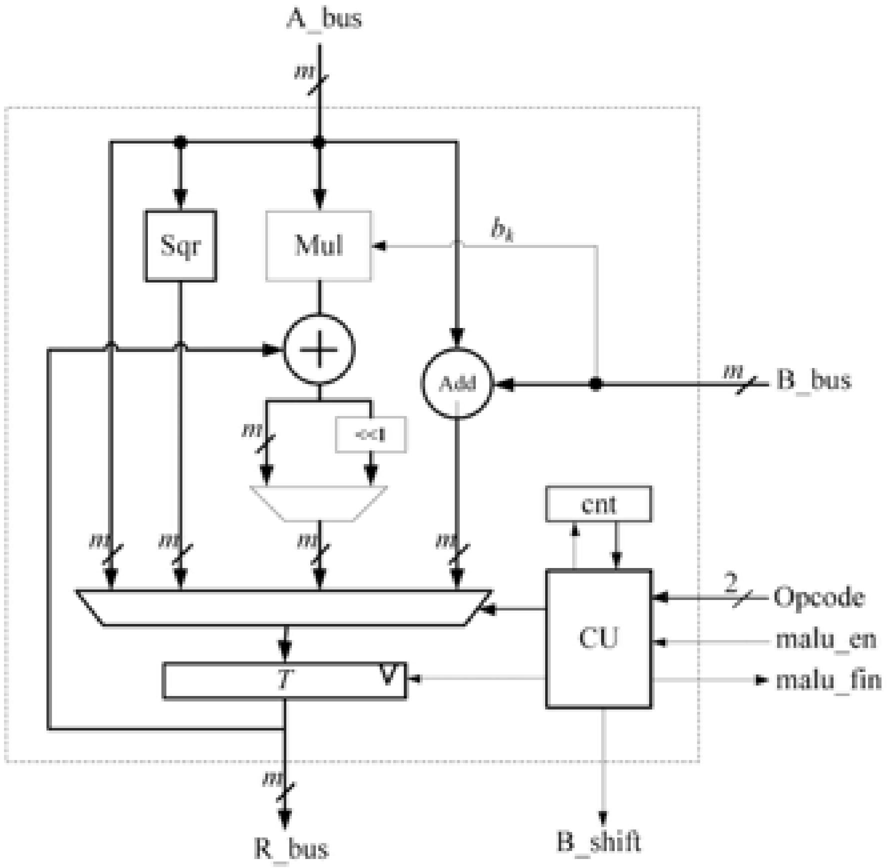
Efficient arithmetic implementation



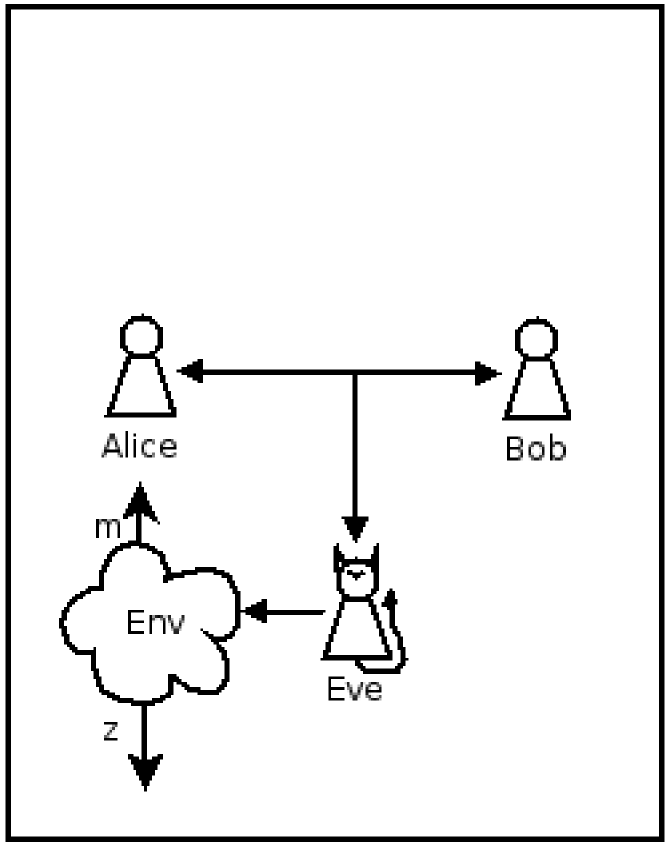
Where these languages could help?

Resource Analysis

Efficient arithmetic implementation



Computational Indistinguishability



PTIME ⟺ LALC

Soundness: every LALC program can be run in polynomial time.

Completeness every PTIME Turing Machine can be expressed in LALC.

PTIME ⇔ LALC

Soundness: every LALC program can be run in polynomial time.

Completeness every PTIME Turing Machine can be expressed in LALC.

Expressivity?

Our research question:

Can we express Algebra and Coalgebra in LALC ?

Our contribution

Our contribution

1 - Weak notions of Algebras and Coalgebras can be encoded in LALC.

Our contribution

1 - Weak notions of Algebras and Coalgebras can be encoded in LALC.

2 - Data types:

- ✓ Inductive types
- ✗ Coinductive types

Our contribution

1 - Weak notions of Algebras and Coalgebras can be encoded in LALC.

2 - Data types:

✓ Inductive types
✗ Coinductive types

3 - LALC restrictions can be relaxed to achieve more expressivity for coinductive types.

Light Affine Lambda Calculus - LALC

LALC ⊂ Linear (Affine) System F

Light Affine Lambda Calculus - LALC

LALC ⊂ Linear (Affine) System F

Main Idea

$$A \multimap B$$

$$A \to B \quad =$$

$$!A \multimap \S B$$

Light Affine Lambda Calculus - LALC

LALC ⊂ Linear (Affine) System F

Main Idea

$$A \to B \quad = \quad \begin{array}{c} A \multimap B \\ \\ !A \multimap \S B \end{array}$$

A non iterative function

-

A function using its argument only once

Light Affine Lambda Calculus - LALC

LALC ⊂ Linear (Affine) System F

Main Idea

$$A \multimap B$$

$$A \to B \quad =$$

$$!A \multimap \S B$$

an iterative function

-

! needed for duplication

§ placeholder witnessing duplication

Iterators in LALC

$$IT : \forall a. !(a \multimap a) \multimap \S(a \multimap a)$$

Iterators in LALC

$$\mathrm{IT} : \forall a.\,!(a \multimap a) \multimap \S(a \multimap a)$$

✓ IT A (step: $A \multimap A$)

Iterators in LALC

$\mathrm{IT} : \forall a.\,!(a \multimap a) \multimap \S(a \multimap a)$

✓ $\mathrm{IT}\;\; A\;(\text{step: } A \multimap A)$

✗ $\mathrm{IT}\;\; A\;(\text{step: } !A \multimap \S A)$

Algebras and Coalgeb

$$\begin{tikzcd} FA \arrow[r, "f_A"] & A \end{tikzcd}$$

Algebra

$$\begin{tikzcd} B \arrow[r, "g_B"] & FB \end{tikzcd}$$

Coalgebra

Algebras and Coalgeb

$$\begin{tikzcd} FA \arrow[r, "f_A"] & A \end{tikzcd}$$

Algebra

$$\begin{tikzcd} B \arrow[r, "g_B"] & FB \end{tikzcd}$$

Coalgebra

$$\begin{tikzcd} F\mu X.FX \arrow[r, "in"] \arrow[d, "Fh"'] & \mu X.FX \arrow[d, "h"] \\ FA \arrow[r, "f_A"] & A \end{tikzcd}$$

Weak Initial Algebra

$\forall A, \exists\ h$

Algebras and Coalgeb

$$\begin{tikzcd}
FA \arrow[r, "f_A"] & A
\end{tikzcd}$$

Algebra

$$\begin{tikzcd}
B \arrow[r, "g_B"] & FB
\end{tikzcd}$$

Coalgebra

$$\begin{tikzcd}
F\mu X.FX \arrow[r, "in"] \arrow[d, "Fh"'] & \mu X.FX \arrow[d, "h"] \\
FA \arrow[r, "f_A"] & A
\end{tikzcd}$$

Weak Initial Algebra

$\forall A, \exists h$

$$\begin{tikzcd}
B \arrow[r, "g_B"] \arrow[d, "h"'] & FB \arrow[d, "Fh"] \\
\nu X.FX \arrow[r, "out"] & F\nu X.FX
\end{tikzcd}$$

Weak Final Coalgebra

$\forall B, \exists h$

Examples

	Initial Algebra	Final Coalgebra
$F(-) = 1 + (-)$	N	$N \cup \{\infty\}$
$F(-) = 1 + A \times (-)$	A^*	A^∞
$F(-) = 1 + A \times (-) \times (-)$	$T^*(A)$	$T^\infty(A)$

(co)algebras in System *F*

Theorem [Reynolds, Plotkin, Geuvers, ...]:

Given F expressible in the polymorphic LC:

- there exists a weakly initial F-Algebra
- there exists a weakly final F-Coalgebra

(co)algebras in System *F*

Theorem [Reynolds, Plotkin, Geuvers, ...]:
Given F expressible in the polymorphic LC:

- there exists a weakly initial F-Algebra
- there exists a weakly final F-Coalgebra

Wraith-Wadler encoding:

$$\mu a.Fa = \forall a.(Fa \to a) \to a$$

$$\nu a.Fa = \exists a.(a \to Fa) * a$$

(co)algebras in System *F*

Theorem [Reynolds, Plotkin, Geuvers, ...]:
Given F expressible in the polymorphic LC:

- there exists a weakly initial F-Algebra
- there exists a weakly final F-Coalgebra

Wraith-Wadler encoding:

F-algebra!

$$\mu a.Fa = \forall a.(\boxed{Fa \to a}) \to a$$

$$\nu a.Fa = \exists a.(a \to Fa) * a$$

(co)algebras in System *F*

Theorem [Reynolds, Plotkin, Geuvers, ...]:

Given F expressible in the polymorphic LC:

- there exists a weakly initial F-Algebra
- there exists a weakly final F-Coalgebra

Wraith-Wadler encoding:

F-algebra!

$$\mu a.Fa = \forall a.(\boxed{Fa \to a}) \to a$$

$$\nu a.Fa = \exists a.(\boxed{a \to Fa}) * a$$

F-coalgebra!

Wraith-Wadler encoding

$$\begin{tikzcd}
F(\mu a.Fa) \arrow[r, "in"] \arrow[d, "F\ (fold\ f)"'] & \mu a.Fa \arrow[d, "fold\ f"] \\
FB \arrow[r, "f"] & B
\end{tikzcd}$$

in : F(μa.Fa) → μa.Fa

in = λs.λk.k (F (fold k) s)

fold : ∀b. (Fb → b) → μa.Fa → b

fold = λf.λt. t f

μa.Fa = ∀a.(Fa→a)→a

Wraith-Wadler encoding in LALC

$\mu a.Fa = \forall a.\ (Fa \to a) \to a$

$\text{in} : F(\mu a.Fa) \to \mu a.Fa$

$\text{in} = \lambda s.\lambda k.\ k\ (F\ (\text{fold}\ k)\ s)$

$\text{fold} : \forall b.\ (Fb \to b) \to \mu a.Fa \to b$

$\text{fold} = \lambda f.\lambda t.\ t\ f$

Wraith-Wadler encoding in LALC

$$\mu a.Fa = \forall a.\ (Fa \to a) \to a$$

k need to be duplicated!

$$\mathrm{in} : F(\mu a.Fa) \to \mu a.Fa$$

$$\mathrm{in} = \lambda s.\lambda k.\ k\ (F\ (\mathrm{fold}\ k)\ s)$$

$$\mathrm{fold} : \forall b.\ (Fb \to b) \to \mu a.Fa \to b$$

$$\mathrm{fold} = \lambda f.\lambda t.\ t\ f$$

Wraith-Wadler encoding in LALC

$$\mu a.Fa = \forall a.!(Fa \multimap a) \multimap \S a$$

Let’s change the type!

$$\mathrm{in} : \ F(\mu a.Fa) \multimap \mu a.Fa$$
$$\mathrm{in} = \lambda s.\lambda k.\S k \ (F \ (\mathrm{fold} \ !k) \ s)$$

$$\mathrm{fold} : \forall b.!(Fb \multimap b) \multimap \mu a.Fa \multimap \S b$$
$$\mathrm{fold} = \lambda f.\lambda t. \ t \ !f$$

Wraith-Wadler encoding in LALC

$$\begin{tikzcd} F(\mu a.Fa) \arrow[r, "\mathrm{in}"] \arrow[d, "F\ (\mathrm{fold}\ !f)"'] & \mu a.Fa \arrow[d, "\mathrm{fold}\ !f"] \\ F\S B \arrow[r, "???"] & \S B \end{tikzcd}$$

$\mathrm{in} : \ F(\mu a.Fa) \multimap \mu a.Fa$

$\mathrm{fold} : \forall b.!(Fb \multimap b) \multimap \mu a.Fa \multimap \S b$

Weakly-Initial algebra under §

$$\begin{tikzcd}
F(\mu a.Fa) \arrow[rr, "in"] \arrow[d, "F\ (fold\ !f)"'] & & \mu a.Fa \arrow[d, "fold\ !f"] \\
F\S B \arrow[r, "L_F"] & \S FB \arrow[r, "\S f"] & \S B
\end{tikzcd}$$

$in :\ F(\mu a.Fa) \multimap \mu a.Fa$

$L_F : \forall b.\ F\S b \multimap \S Fb$

$fold : \forall b.!(Fb \multimap b) \multimap \mu a.Fa \multimap \S b$

Weakly-Initial algebra under §

$$\begin{tikzcd}
F(\mu a.Fa) \arrow[rr, "in"] \arrow[d, "F\ (fold\ !f)"'] & & \mu a.Fa \arrow[d, "fold\ !f"] \\
F\S B \arrow[r, "L_F"] & \S FB \arrow[r, "\S f"] & \S B
\end{tikzcd}$$

Left distributivity

$in :\ F(\mu a.Fa) \multimap \mu a.Fa$

$L_F : \forall b.\ F\S b \multimap \S Fb$

$fold : \forall b.!(Fb \multimap b) \multimap \mu a.Fa \multimap \S b$

Weakly-Final coalgebra under §

$$\begin{tikzcd}
\nu a.Fa \arrow[rr, "\mathrm{out}"] & & F(\nu a.Fa) \arrow[d, "F(\mathrm{unfold}\ !f)"] \\
\S B \arrow[u, "\mathrm{unfold}\ !f"] \arrow[r, "\S f"] & \S FB \arrow[r, "R_F"] & F\S B
\end{tikzcd}$$

$\mathrm{out} : \nu a.Fa \multimap F(\nu a.Fa)$

$R_F : \forall b.\ \S Fb \multimap F\S b$

$\mathrm{unfold} : \forall b.!(b \multimap Fb) \multimap \S b \multimap \nu a.Fa$

$\nu a.Fa = \exists a.!(a \multimap Fa) \otimes \S a$

Weakly-Final coalgebra under §

$$\begin{tikzcd}
\nu a.Fa \arrow[rr, "\mathrm{out}"] & & F(\nu a.Fa) \arrow[d, "F(\mathrm{unfold}\ !f)"] \\
\S B \arrow[u, "\mathrm{unfold}\ !f"] \arrow[r, "\S f"] & \S FB \arrow[r, "R_F"] & F\S B
\end{tikzcd}$$

Right distributivity

$\mathrm{out} : \nu a.Fa \multimap F(\nu a.Fa)$

$R_F : \forall b.\ \S Fb \multimap F\S b$

$\mathrm{unfold} : \forall b.!(b \multimap Fb) \multimap \S b \multimap \nu a.Fa$

$\nu a.Fa = \exists a.!(a \multimap Fa) \otimes \S a$

Expressivity?

- Which Functor satisfies $L_F : \forall b.\ F\S b \multimap \S F b$?

$$F(-) = 1 \oplus -$$

$$F(-) = 1 \oplus A \otimes -$$

$$F(-) = 1 \oplus A \otimes - \otimes -$$

Expressivity?

- Which Functor satisfies $L_F : \forall b.\ F\S b \multimap \S F b$?

✓ $F(-) = 1 \oplus -$

✓ $F(-) = 1 \oplus A \otimes -$

✓ $F(-) = 1 \oplus A \otimes - \otimes -$

provided $\vdash A \multimap \S A$

Expressivity?

- Which Functor satisfies R_F: $\forall b.\ \S F b \multimap F \S b$

$$F(-) = 1 \oplus -$$

$$F(-) = 1 \oplus A \otimes -$$

$$F(-) = 1 \oplus A \otimes - \otimes -$$

Expressivity?

- Which Functor satisfies R_F: $\forall b.\ \S F b \multimap F \S b$

✗ $F(-) = 1 \oplus -$

✗ $F(-) = 1 \oplus A \otimes -$

✗ $F(-) = 1 \oplus A \otimes - \otimes -$

Where is the problem?

The modality § does not commute with the other type constructions!

Where is the problem?

The modality § does not commute with the other type constructions!

Solution: make § to commute

Adding terms for distributivity

We can add to LALC the following terms:

$\mathrm{dist}_{\oplus} : \S(A \oplus B) \multimap \S A \oplus \S B$

$\mathrm{dist}_{\otimes} : \S(A \otimes B) \multimap \S A \otimes \S B$

$\mathrm{dist}_{\oplus}\ \S(\mathrm{inj}\ t) \rightarrow \mathrm{inj}\ \S t$

$\mathrm{dist}_{\otimes}\ \S(<t_1,t_2>) \rightarrow <\S t_1,\S t_2>$

Adding terms for distributivity

We can add to LALC the following terms:

$\mathrm{dist}_{\oplus} : \S(A \oplus B) \multimap \S A \oplus \S B$

$\mathrm{dist}_{\otimes} : \S(A \otimes B) \multimap \S A \otimes \S B$

$\mathrm{dist}_{\oplus}\ \S(\mathrm{inj}\ t) \rightarrow \mathrm{inj}\ \S t$

$\mathrm{dist}_{\otimes}\ \S(<t_1,t_2>) \rightarrow <\S t_1,\S t_2>$

They require the evaluation of terms inside a §

Problem: this breaks polynomial time soundness.

Adding terms for distributivity

We can add to LALC the following terms:

$\mathrm{dist}_{\oplus} : \S(A \oplus B) \multimap \S A \oplus \S B$

$\mathrm{dist}_{\otimes} : \S(A \otimes B) \multimap \S A \otimes \S B$

$\mathrm{dist}_{\oplus}\ \S(\mathrm{inj}\ t) \rightarrow \mathrm{inj}\ \S t$

$\mathrm{dist}_{\otimes}\ \S(<t_1,t_2>) \rightarrow <\S t_1,\S t_2>$

They require the evaluation of terms inside a §

Problem: this breaks polynomial time soundness.

New (quite technical) proof in the paper!

Expressivity?

- Which Functor satisfies R_F: $\forall b.\ \S F b \multimap F \S b$

$$F(-) = 1 \oplus -$$

$$F(-) = 1 \oplus A \otimes -$$

$$F(-) = 1 \oplus A \otimes - \otimes -$$

Expressivity?

- Which Functor satisfies R_F: $\forall b.\ \S F b \multimap F \S b$

✓ $F(-) = 1 \oplus -$

✓ $F(-) = 1 \oplus A \otimes -$

✓ $F(-) = 1 \oplus A \otimes - \otimes -$

provided $\vdash \S A \multimap A$

Expressivity?

- Which Functor satisfies R_F: $\forall b.\ \S F b \multimap F \S b$

✓ $F(-) = 1 \oplus -$

✓ $F(-) = 1 \oplus A \otimes -$

✓ $F(-) = 1 \oplus A \otimes - \otimes -$

provided $\vdash \S A \multimap A$

We can encode streams and trees at every finite type

Take out?

- Algebras and Coalgebras encodings make sense also for polynomial time languages,
- Due to the restrictive nature of languages for implicit complexity their definitions can be a bit more tricky,
- The expressivity may still depend on the restrictions of the language.