Expressive Power and Algorithms

Guillaume Bonfante[1] Pierre Boudes[2] Jean-Yves Moyen[2]
Jean-Yves.Moyen@lipn.univ-paris13.fr

[1]Université de Lorraine
École des Mines de Nancy

[2]Université Paris 13

Partly supported by the ANR project Complice (ANR-08-BLANC-0211-01)

November 2013

Introduction

Motivations

- Are “complicated” programming constructions really useful? We’re still Turing-complete without them.
  High-order? Non-determinism? Co-arity? Multiple tapes? Large alphabets? . . .

Motivations

- Are "complicated" programming constructions really useful? We're still Turing-complete without them.
  High-order? Non-determinism? Co-arity? Multiple tapes? Large alphabets? . . .
- Which is "the best" ICC system for PTIME?
  MPO+QI? DLAL? *mwp*-polynomials? NSI? . . .
- How to even compare them?

Magic Trick!

Explaining the Trick

- bead = program
- color = computed function
- small bead = a given syntactical criterion

Explaining the Trick

- bead = program
- color = computed function
- small bead = a given syntactical criterion

Each set of “programs” computes the same set of “functions”. But the first one has more things (small black beads) and only the filter can reveal it.

Explaining the Trick

- bead = program
- color = computed function
- small bead = a given syntactical criterion

Each set of "programs" computes the same set of "functions". But the first one has more things (small black beads) and only the filter can reveal it.

There is no color- and size-preserving function from the first set to the second.

Life without Cons

First order functional programs are Turing-complete. Why using high-order?

- Libraries

Life without Cons

First order functional programs are Turing-complete. Why using high-order?

- ~~Libraries~~
- Easier to write, more elegant programs.

Life without Cons

First order functional programs are Turing-complete. Why using high-order?

- ~~Libraries~~
- Easier to write, more elegant programs.

Jones looked at “CONS free” (read only) programs.

- First order CONS free: PTIME
- Second order CONS free: EXPTIME
- High order CONS free: ELEM

Life without Cons

First order functional programs are Turing-complete. Why using high-order?

- ~~Libraries~~
- Easier to write, more elegant programs.

Jones looked at "CONS free" (read only) programs.

- First order CONS free: PTIME
- Second order CONS free: EXPTIME
- High order CONS free: ELEM

High order is "more expressive" than first order, but Turing-completeness hide this.

Expressive Power

Compilation

- Programming language: $\mathcal{P}$ $\supset L, M$ (syntactic)

L

M

Compilation

- Programming language: $\mathcal{P}$ $\supset L, M$ (syntactic)
- Computable functions: $\mathcal{C}$

L

M

Compilation

- Programming language: $\mathcal{P}$ $\supset L, M$ (syntactic)
- Computable functions: $\mathcal{C}$
- Semantics: $\mathcal{P} \xrightarrow{[\![\bullet]\!]} \mathcal{C}$

L

M

Compilation

- Programming language: $\mathcal{P}$ $\supset L, M$ (syntactic)
- Computable functions: $\mathcal{C}$
- Semantics: $\mathcal{P} \xrightarrow{[\![\bullet]\!]} \mathcal{C}$ $[\![L]\!] = A = B = [\![M]\!]$

$$\begin{tikzcd}
L \arrow[r, two heads, "{[\![\bullet]\!]}"] & A \arrow[d, "id"] \\
M \arrow[r, two heads, "{[\![\bullet]\!]}"] & B
\end{tikzcd}$$

Compilation

- Programming language: $\mathcal{P}$ $\supset L, M$ (syntactic)
- Computable functions: $\mathcal{C}$
- Semantics: $\mathcal{P} \xrightarrow{[\![\bullet]\!]} \mathcal{C}$

$$\boxed{[\![L]\!] = A} = \boxed{B = [\![M]\!]}$$

$$\begin{tikzcd}
L \arrow[r, two heads, "{[\![\bullet]\!]}"] & A \arrow[d, "id"] \\
M \arrow[r, two heads, "{[\![\bullet]\!]}"] & B
\end{tikzcd}$$

Hard to prove

Compilation

- Programming language: $\mathcal{P}$ $\supset L, M$ (syntactic)
- Computable functions: $\mathcal{C}$
- Semantics: $\mathcal{P} \xrightarrow{[\![\bullet]\!]} \mathcal{C}$ $[\![L]\!] = A = B = [\![M]\!]$
- Compilation $\varphi : L \to M$, $[\![\varphi(\mathtt{p})]\!] = [\![\mathtt{p}]\!]$ Semantics preserving

$$
\begin{tikzcd}
L \arrow[r, two heads, "{[\![\bullet]\!]}"] \arrow[d, "\varphi"] & A \arrow[d, "id"] \\
M \arrow[r, two heads, "{[\![\bullet]\!]}"] & B
\end{tikzcd}
$$

Filtering

Hypothesis

- $[\![L]\!] = A = B = [\![M]\!]$

Filtering

Hypothesis

- $[\![L]\!] = A = B = [\![M]\!]$
- "Filter": set of programs $F \subset \mathcal{P}$ (syntactic criterion)
- Filtering: $L' = L \cap F$, $M' = M \cap F$

Filtering

Hypothesis

- $[\![L]\!] = A = B = [\![M]\!]$
- "Filter": set of programs $F \subset \mathcal{P}$ (syntactic criterion)
- Filtering: $L' = L \cap F$, $M' = M \cap F$
- Breaking semantic equality: $[\![L']\!] = A' \supsetneq B' = [\![M']\!]$

Filtering

Hypothesis

- $[\![L]\!] = A = B = [\![M]\!]$
- "Filter": set of programs $F \subset \mathcal{P}$ (syntactic criterion)
- Filtering: $L' = L \cap F$, $M' = M \cap F$
- Breaking semantic equality: $[\![L']\!] = A' \supsetneq B' = [\![M']\!]$

Conclusion

There is no filter-preserving compilation: $\mathtt{p} \in F \Rightarrow \varphi(\mathtt{p}) \in F$

Conversely, if there exists a filter preserving compilation, then $A' = B'$.

Expressiveness

$$\begin{tikzcd}
& L' \arrow[rr, two heads, "\llbracket\bullet\rrbracket"] \arrow[dl, hook', "\iota"] \arrow[dd, dotted, "\varphi"] & & A' \arrow[dl, hook', "\iota"] \arrow[dd, "id"'] \\
L \arrow[rr, two heads, "\llbracket\bullet\rrbracket"] \arrow[dd, "\varphi"'] & & A \arrow[dd, "id"] & \\
& M' \arrow[rr, dotted, two heads, "\llbracket\bullet\rrbracket"] \arrow[dl, dotted, hook', "\iota"] & & B' \arrow[dl, hook', "\iota"] \\
M \arrow[rr, two heads, "\llbracket\bullet\rrbracket"] & & B &
\end{tikzcd}$$

$\mathtt{p} \in F \Rightarrow \varphi(\mathtt{p}) \in F$

$B' \subsetneq A'$

Expressiveness

$$\begin{tikzcd}
L \arrow[r, two heads, "\llbracket \bullet \rrbracket"] \arrow[d, "\varphi"] & A \arrow[d, "id"] \\
M \arrow[r, two heads, "\llbracket \bullet \rrbracket"] & B
\end{tikzcd}$$

$\mathtt{p} \in F \Rightarrow \varphi(\mathtt{p}) \in F$

$B' \subsetneq A'$

Compilation

Expressiveness

$$\begin{tikzcd}
 & L' \arrow[dl, hook', "\iota"'] \arrow[dd, dotted, "\varphi"] \\
L \arrow[dd, "\varphi"] & \\
 & M' \arrow[dl, hook', dotted, "\iota"'] \\
M &
\end{tikzcd}$$

$\mathtt{p} \in F \Rightarrow \varphi(\mathtt{p}) \in F$

$B' \subsetneq A'$

Compilation

$\mathtt{p} \in L' = L \cap F$
$\Rightarrow \varphi(\mathtt{p}) \in M \cap F = M'$

Expressiveness

$$\begin{tikzcd} & \mathrm{L'} \arrow[rr, two heads, "\llbracket\bullet\rrbracket"] \arrow[dl, hook, "\iota"] & & \mathrm{A'} \arrow[dl, hook, "\iota"] \\ \mathrm{L} \arrow[rr, two heads, "\llbracket\bullet\rrbracket"] & & \mathrm{A} & \end{tikzcd}$$

$$\begin{tikzcd} & \mathrm{M'} \arrow[rr, two heads, dotted, "\llbracket\bullet\rrbracket"] \arrow[dl, hook, dotted, "\iota"] & & \mathrm{B'} \arrow[dl, hook, "\iota"] \\ \mathrm{M} \arrow[rr, two heads, "\llbracket\bullet\rrbracket"] & & \mathrm{B} & \end{tikzcd}$$

$\mathtt{p} \in F \Rightarrow \varphi(\mathtt{p}) \in F$

$B' \subsetneq A'$

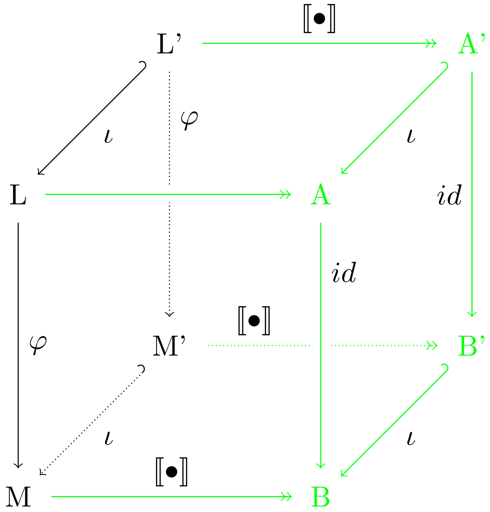
Compilation

$\mathtt{p} \in L' = L \cap F$

$\Rightarrow \varphi(\mathtt{p}) \in M \cap F = M'$

Canonical injection

Expressiveness



$\mathtt{p} \in F \Rightarrow \varphi(\mathtt{p}) \in F$

$B' \subsetneq A'$

Compilation

$\mathtt{p} \in L' = L \cap F$

$\Rightarrow \varphi(\mathtt{p}) \in M \cap F = M'$

Canonical injection

$f \in A'$

$\mathtt{p} \in L', [\![\mathtt{p}]\!] = f$

$\mathtt{q} = \varphi(\mathtt{p}) \in M', [\![\mathtt{q}]\!] = f$

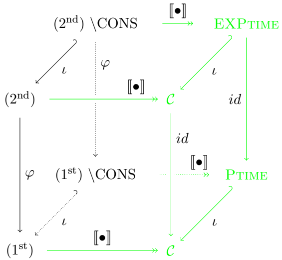
$f \in B'$

High Order

$$
\begin{tikzcd}
& (2^{\mathrm{nd}})\ \backslash\mathrm{CONS} \arrow[ld, hook', "\iota"] \arrow[dd, dotted, "\varphi"] \arrow[rr, two heads, "\llbracket\bullet\rrbracket"] & & \textsc{EXPTime} \arrow[ld, hook', "\iota"'] \arrow[dd, "id"] \\
(2^{\mathrm{nd}}) \arrow[rr, two heads, "\llbracket\bullet\rrbracket"] \arrow[dd, "\varphi"'] & & \mathcal{C} \arrow[dd, "id"] & \\
& (1^{\mathrm{st}})\ \backslash\mathrm{CONS} \arrow[ld, dotted, hook', "\iota"] \arrow[rr, dotted, two heads, "\llbracket\bullet\rrbracket"] & & \textsc{PTime} \arrow[ld, hook', "\iota"] \\
(1^{\mathrm{st}}) \arrow[rr, two heads, "\llbracket\bullet\rrbracket"] & & \mathcal{C} &
\end{tikzcd}
$$

$\mathcal{P}$ = high order TRS
$L = (2^{\mathrm{nd}})$ $M = (1^{\mathrm{st}})$ $F = \backslash\mathrm{CONS}$

High Order



$\mathcal{P}$ = high order TRS

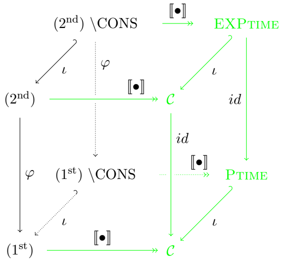
$L = (2^{\text{nd}})$ $M = (1^{\text{st}})$ $F = \backslash\text{CONS}$

$[\![(2^{\text{nd}})]\!] = \mathcal{C} = [\![(1^{\text{st}})]\!]$

$[\![(2^{\text{nd}})\backslash\text{CONS}]\!] = \text{EXPTIME}$

$[\![(1^{\text{st}})\backslash\text{CONS}]\!] = \text{PTIME}$

High Order

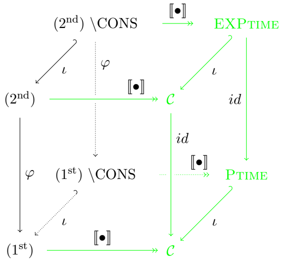


$\mathcal{P}$ = high order TRS
$L = (2^{nd})$ $M = (1^{st})$ $F = \setminus$CONS

$[\![(2^{nd})]\!] = \mathcal{C} = [\![(1^{st})]\!]$
$[\![(2^{nd})\setminus\text{CONS}]\!] = $ EXPTIME
$[\![(1^{st})\setminus\text{CONS}]\!] = $ PTIME

EXPTIME $\neq$ PTIME, hence there exists no compilation from (2^{nd}) to (1^{st}) preserving $\setminus$CONS.

High Order



$\mathcal{P}$ = high order TRS
$L = (2^{\text{nd}})$ $M = (1^{\text{st}})$ $F = \backslash\text{CONS}$

$[\![(2^{\text{nd}})]\!] = \mathcal{C} = [\![(1^{\text{st}})]\!]$
$[\![(2^{\text{nd}})\backslash\text{CONS}]\!] = \text{EXPTIME}$
$[\![(1^{\text{st}})\backslash\text{CONS}]\!] = \text{PTIME}$

$\text{EXPTIME} \neq \text{PTIME}$, hence there exists no compilation from (2^{nd}) to (1^{st}) preserving \CONS.

High order programs are more expressive than first order ones.

Non-Determinism

$$
\begin{tikzcd}
 & (\mathrm{MPO{+}ND}) \cap (\mathrm{QI}) \arrow[rr, two heads, "\llbracket \bullet \rrbracket"] \arrow[dl, hook', "\iota"] \arrow[dd, dotted, "\varphi"] & & \textsc{Pspace} \arrow[dl, hook', "\iota"] \arrow[dd, "id"] \\
(\mathrm{MPO{+}ND}) \arrow[rr, two heads, "\llbracket \bullet \rrbracket"] \arrow[dd, "\varphi"] & & \mathrm{PR} \arrow[dd, "id"] & \\
 & (\mathrm{MPO}) \cap (\mathrm{QI}) \arrow[rr, dotted, two heads, "\llbracket \bullet \rrbracket"] \arrow[dl, dotted, hook', "\iota"] & & \textsc{Ptime} \arrow[dl, hook', "\iota"] \\
(\mathrm{MPO}) \arrow[rr, two heads, "\llbracket \bullet \rrbracket"] & & \mathrm{PR} &
\end{tikzcd}
$$

$\mathcal{P} = \text{ND first order TRS}$

$L = (\mathrm{MPO{+}ND}) \qquad M = (\mathrm{MPO})$

$F = (\mathrm{QI})$

Non-Determinism

$$\begin{tikzcd}
 & (\mathrm{MPO{+}ND}) \cap (\mathrm{QI}) \arrow[ld, hook, "\iota"'] \arrow[dd, dotted, "\varphi"] \arrow[rr, two heads, "\llbracket \bullet \rrbracket"] & & \textsc{Pspace} \arrow[ld, hook, "\iota"'] \arrow[dd, "id"] \\
(\mathrm{MPO{+}ND}) \arrow[rr, two heads, "\llbracket \bullet \rrbracket"] \arrow[dd, "\varphi"'] & & \mathrm{PR} \arrow[dd, "id"] & \\
 & (\mathrm{MPO}) \cap (\mathrm{QI}) \arrow[ld, hook, dotted, "\iota"'] \arrow[rr, two heads, dotted, "\llbracket \bullet \rrbracket"] & & \textsc{Ptime} \arrow[ld, hook, "\iota"'] \\
(\mathrm{MPO}) \arrow[rr, two heads, "\llbracket \bullet \rrbracket"] & & \mathrm{PR} &
\end{tikzcd}$$

$\mathcal{P} = \text{ND first order TRS}$
$L = (\mathrm{MPO{+}ND})$ $\quad M = (\mathrm{MPO})$
$F = (\mathrm{QI})$

$\llbracket (\mathrm{MPO{+}ND}) \rrbracket = PR = \llbracket (\mathrm{MPO}) \rrbracket$
$\llbracket (\mathrm{MPO{+}ND}) \cap (\mathrm{QI}) \rrbracket = \textsc{Pspace}$
$\llbracket (\mathrm{MPO}) \cap (\mathrm{QI}) \rrbracket = \textsc{Ptime}$

Non-Determinism

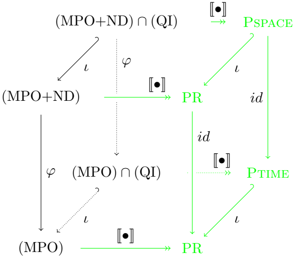
$$
\begin{tikzcd}
& (\mathrm{MPO{+}ND}) \cap (\mathrm{QI}) \arrow[rr, two heads, "{[\![\bullet]\!]}"] \arrow[dl, hook', "\iota"] \arrow[dd, dotted, "\varphi"] & & \textsc{Pspace} \arrow[dl, hook', "\iota"] \arrow[dd, "id"] \\
(\mathrm{MPO{+}ND}) \arrow[rr, two heads, "{[\![\bullet]\!]}"] \arrow[dd, "\varphi"] & & \mathrm{PR} \arrow[dd, "id"] & \\
& (\mathrm{MPO}) \cap (\mathrm{QI}) \arrow[rr, dotted, two heads, "{[\![\bullet]\!]}"] \arrow[dl, dotted, hook', "\iota"] & & \textsc{Ptime} \arrow[dl, hook', "\iota"] \\
(\mathrm{MPO}) \arrow[rr, two heads, "{[\![\bullet]\!]}"] & & \mathrm{PR} &
\end{tikzcd}
$$

$\mathcal{P}$ = ND first order TRS
$L = (\mathrm{MPO{+}ND}) \qquad M = (\mathrm{MPO})$
$F = (\mathrm{QI})$

$[\![(\mathrm{MPO{+}ND})]\!] = PR = [\![(\mathrm{MPO})]\!]$
$[\![(\mathrm{MPO{+}ND}) \cap (\mathrm{QI})]\!] = \textsc{Pspace}$
$[\![(\mathrm{MPO}) \cap (\mathrm{QI})]\!] = \textsc{Ptime}$

$\textsc{Pspace} \neq \textsc{Ptime}$ (maybe), hence there exists no compilation from (MPO+ND) to (MPO) preserving (QI).

Non-Determinism



$\mathcal{P}$ = ND first order TRS

L = (MPO+ND) $\qquad M$ = (MPO)

F = (QI)

$[\![\text{(MPO+ND)}]\!] = PR = [\![\text{(MPO)}]\!]$

$[\![\text{(MPO+ND)} \cap \text{(QI)}]\!] = \textsc{Pspace}$

$[\![\text{(MPO)} \cap \text{(QI)}]\!] = \textsc{Ptime}$

$\textsc{Pspace} \neq \textsc{Ptime}$ (maybe), hence there exists no compilation from (MPO+ND) to (MPO) preserving (QI).

Non-deterministic programs are more expressive than first order ones.

Algorithms

What is an Algorithm?

- Solid (mathematical) theory of functions.
- No good theory of algorithms
  (Gurevich's thesis: ASM $\equiv$ algorithm).

What is an Algorithm?

- Solid (mathematical) theory of functions.
- No good theory of algorithms
(Gurevich's thesis: ASM $\equiv$ algorithm).

What's sure:

- Two programs may implement the same algorithm.
- Two algorithms may compute the same function.

What is an Algorithm?

- Solid (mathematical) theory of functions.
- No good theory of algorithms
  (Gurevich's thesis: ASM $\equiv$ algorithm).

What's sure:

- Two programs may implement the same algorithm.
- Two algorithms may compute the same function.

Algorithms lay somewhere between programs and functions.

$$\begin{tikzcd} \mathcal{P} \arrow[r, two heads, "|\bullet|"] \arrow[rr, bend right, two heads, "[\![\bullet]\!]"'] & \mathcal{Z} \arrow[r, two heads, "[\bullet]"] & \mathcal{C} \end{tikzcd}$$

$$[|\mathtt{p}|] = [\![\mathtt{p}]\!]$$

The Algorithmic Level

- Programming language: $\mathcal{P}$ $\supset L, M, F$ (syntactic)

The Algorithmic Level

- Programming language: $\mathcal{P}$ $\supset L, M, F$ (syntactic)
- Algorithms (?): $\mathcal{Z}$ Computable functions: $\mathcal{C}$
- Semantics: $\mathcal{P} \xrightarrow{|\bullet|} \mathcal{Z} \xrightarrow{[\bullet]} \mathcal{C}$

  Algorithms respect filter: $\mathtt{p} \in F, \mathtt{q} \notin F \Rightarrow |\mathtt{p}| \neq |\mathtt{q}|$

The Algorithmic Level

- Programming language: $\mathcal{P} \supset L, M, F$ (syntactic)
- Algorithms (?): $\mathcal{Z}$ Computable functions: $\mathcal{C}$
- Semantics: $\mathcal{P} \xrightarrow{|\bullet|} \mathcal{Z} \xrightarrow{[\bullet]} \mathcal{C}$

  Algorithms respect filter: $\mathtt{p} \in F, \mathtt{q} \notin F \Rightarrow |\mathtt{p}| \neq |\mathtt{q}|$
- $\boxed{[\![L]\!] = A} = \boxed{B = [\![M]\!]}$

The Algorithmic Level

- Programming language: $\mathcal{P} \quad \supset L, M, F$ (syntactic)
- Algorithms (?): $\mathcal{Z}$ Computable functions: $\mathcal{C}$
- Semantics: $\mathcal{P} \xrightarrow{|\bullet|} \mathcal{Z} \xrightarrow{[\bullet]} \mathcal{C}$

  Algorithms respect filter: $\mathtt{p} \in F, \mathtt{q} \notin F \Rightarrow |\mathtt{p}| \neq |\mathtt{q}|$
- $[\![L]\!] = A = B = [\![M]\!]$
- Filtering: $L' = L \cap F$, $M' = M \cap F$
- Breaking semantic equality: $[\![L']\!] = A' \supsetneq B' = [\![M']\!]$

Transforming algorithms

Hypothesis

- $[\![L]\!] = A = B = [\![M]\!]$
- $[\![L']\!] = A' \neq B' = [\![M']\!]$
- Algorithms respect filter: $\mathtt{p} \in F, \mathtt{q} \notin F \Rightarrow |\mathtt{p}| \neq |\mathtt{q}|$
- $|L| = X \quad |M| = Y$

Transforming algorithms

Hypothesis

- $[\![L]\!] = A = B = [\![M]\!]$
- $[\![L']\!] = A' \neq B' = [\![M']\!]$
- Algorithms respect filter: $\mathtt{p} \in F, \mathtt{q} \notin F \Rightarrow |\mathtt{p}| \neq |\mathtt{q}|$
- $|L| = X \qquad |M| = Y$

Algorithm transformation: $\tau : X \to Y$

Conclusion

There is no filter-preserving transformation of algorithms:
$x \in |F| \Rightarrow \tau(x) \in |F|$

Transforming algorithms

Hypothesis

- $[\![L]\!] = A = B = [\![M]\!]$
- $[\![L']\!] = A' \neq B' = [\![M']\!]$
- Algorithms respect filter: $\mathtt{p} \in F, \mathtt{q} \notin F \Rightarrow |\mathtt{p}| \neq |\mathtt{q}|$
- $|L| = X \qquad |M| = Y$

Algorithm transformation: $\tau : X \to Y$

Conclusion

There is no filter-preserving transformation of algorithms: $x \in |F| \Rightarrow \tau(x) \in |F|$

Conversely, if there exists a filter preserving transformation (incl. identity), then $A' = B'$.

Magic Trick Again

- Suppose that I have written something (an “algorithm”) inside each bead (text? formula? drawing? . . .)

Magic Trick Again

- Suppose that I have written something (an "algorithm") inside each bead (text? formula? drawing? ...)
- It's written in white inside black beads, and in black inside white beads. Hence beads of different colours (computing different functions) necessarily have different algorithms.
- I claim that beads of different size have different algorithms (respecting the filter).

Magic Trick Again

- Suppose that I have written something (an "algorithm") inside each bead (text? formula? drawing? ...)
- It's written in white inside black beads, and in black inside white beads. Hence beads of different colours (computing different functions) necessarily have different algorithms.
- I claim that beads of different size have different algorithms (respecting the filter).

[Filtering]

- There are things written in beads of the first set that are not written in beads of the second (those in small black beads).

The big picture

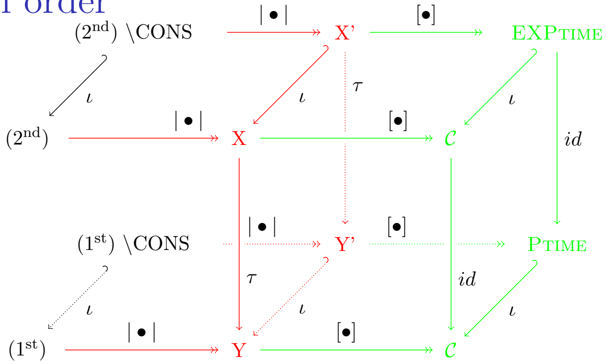
$$
\begin{tikzcd}
 & L' \arrow[rr, two heads, "\lvert \bullet \rvert"] \arrow[dl, hook, "\iota"] & & X' \arrow[rr, two heads, "{[\bullet]}"] \arrow[dl, hook, "\iota"] \arrow[dd, dotted, "\tau"] & & A' \arrow[dl, hook, "\iota"] \arrow[dd, "id"] \\
L \arrow[rr, two heads, "\lvert \bullet \rvert"] & & X \arrow[rr, two heads, "{[\bullet]}"] \arrow[dd, "\tau"] & & A \arrow[dd, "id"] & \\
 & M' \arrow[rr, dotted, two heads, "\lvert \bullet \rvert"] \arrow[dl, dotted, hook, "\iota"] & & Y' \arrow[rr, dotted, two heads, "{[\bullet]}"] \arrow[dl, dotted, hook, "\iota"] & & B' \arrow[dl, hook, "\iota"] \\
M \arrow[rr, two heads, "\lvert \bullet \rvert"] & & Y \arrow[rr, two heads, "{[\bullet]}"] & & B &
\end{tikzcd}
$$

High order

$$
\begin{tikzcd}
& (2^{\mathrm{nd}}) \setminus \mathrm{CONS} \arrow[rr, two heads, "|\bullet|"] \arrow[dl, hook, "\iota"] & & \mathrm{X'} \arrow[rr, two heads, "[\bullet]"] \arrow[dl, hook, "\iota"] \arrow[dd, dotted, "\tau"] & & \textsc{ExpTime} \arrow[dl, hook, "\iota"] \arrow[dd, "id"] \\
(2^{\mathrm{nd}}) \arrow[rr, two heads, "|\bullet|"] & & \mathrm{X} \arrow[rr, two heads, "[\bullet]"] \arrow[dd, "\tau"] & & \mathcal{C} \arrow[dd, "id"] & \\
& (1^{\mathrm{st}}) \setminus \mathrm{CONS} \arrow[rr, dotted, two heads, "|\bullet|"] \arrow[dl, dotted, hook, "\iota"] & & \mathrm{Y'} \arrow[rr, dotted, two heads, "[\bullet]"] \arrow[dl, dotted, hook, "\iota"] & & \textsc{PTime} \arrow[dl, hook, "\iota"] \\
(1^{\mathrm{st}}) \arrow[rr, two heads, "|\bullet|"] & & \mathrm{Y} \arrow[rr, two heads, "[\bullet]"] & & \mathcal{C} &
\end{tikzcd}
$$

Algorithms respect cons-free.

High order



Algorithms respect cons-free.

There is no transformation from high order algorithms to first order algorithm *preserving cons-free.*

There are cons-free high-order algorithms with no first order counterpart.

High order

$$\begin{tikzcd}
& (2^{\text{nd}})\setminus\text{CONS} \arrow[rr, two heads, "|\bullet|"] \arrow[dl, hook, "\iota"] & & \text{X'} \arrow[rr, two heads, "[\bullet]"] \arrow[dl, hook, "\iota"] \arrow[dd, dotted, "\tau"] & & \text{EXP}\textsc{time} \arrow[dl, hook, "\iota"] \arrow[dd, "id"] \\
(2^{\text{nd}}) \arrow[rr, two heads, "|\bullet|"] & & \text{X} \arrow[rr, two heads, "[\bullet]"] \arrow[dd, "\tau"] & & \mathcal{C} \arrow[dd, "id"] & \\
& (1^{\text{st}})\setminus\text{CONS} \arrow[rr, dotted, two heads, "|\bullet|"] \arrow[dl, dotted, hook, "\iota"] & & \text{Y'} \arrow[rr, dotted, two heads, "[\bullet]"] \arrow[dl, dotted, hook, "\iota"] & & \textsc{Ptime} \arrow[dl, hook, "\iota"] \\
(1^{\text{st}}) \arrow[rr, two heads, "|\bullet|"] & & \text{Y} \arrow[rr, two heads, "[\bullet]"] & & \mathcal{C} &
\end{tikzcd}$$

Algorithms respect cons-free.

There is no transformation from high order algorithms to first order algorithm *preserving cons-free.*

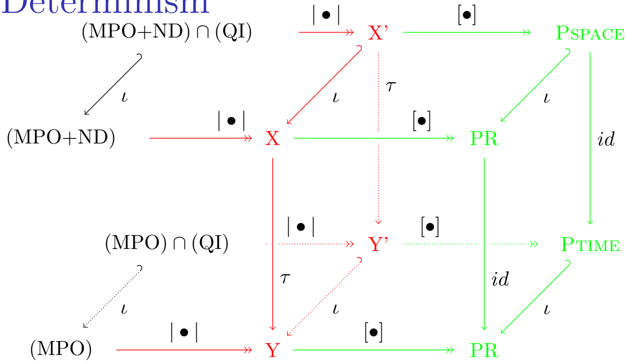
There are cons-free high-order algorithms with no first order counterpart.

Non Determinism

$$\begin{tikzcd}
 & (\mathrm{MPO+ND}) \cap (\mathrm{QI}) \arrow[rr, two heads, "|\bullet|"] \arrow[dl, hook', "\iota"] & & \mathrm{X'} \arrow[rr, two heads, "{[\bullet]}"] \arrow[dl, hook', "\iota"] \arrow[dd, dotted, "\tau"] & & \textsc{Pspace} \arrow[dl, hook', "\iota"] \arrow[dd, "id"] \\
(\mathrm{MPO+ND}) \arrow[rr, two heads, "|\bullet|"] & & \mathrm{X} \arrow[rr, two heads, "{[\bullet]}"] \arrow[dd, "\tau"] & & \mathrm{PR} \arrow[dd, "id"] & \\
 & (\mathrm{MPO}) \cap (\mathrm{QI}) \arrow[rr, dotted, two heads, "|\bullet|"] \arrow[dl, dotted, hook', "\iota"] & & \mathrm{Y'} \arrow[rr, dotted, two heads, "{[\bullet]}"] \arrow[dl, dotted, hook', "\iota"] & & \textsc{Ptime} \arrow[dl, hook', "\iota"] \\
(\mathrm{MPO}) \arrow[rr, two heads, "|\bullet|"] & & \mathrm{Y} \arrow[rr, two heads, "{[\bullet]}"] & & \mathrm{PR} &
\end{tikzcd}$$

Algorithms respect QI.

Non Determinism



Algorithms respect QI.

There is no transformation from non-deterministic algorithms to first order algorithm *preserving Quasi-Interpretations*.
There are Non-deterministic algorithms admitting a Quasi-Interpretation with no deterministic counterpart.

Non Determinism

$$
\begin{tikzcd}
 & (\mathrm{MPO+ND}) \cap (\mathrm{QI}) \arrow[rr, two heads, "|\bullet|"] \arrow[dl, hook, "\iota"] & & \mathrm{X'} \arrow[rr, two heads, "{[\bullet]}"] \arrow[dl, hook, "\iota"] \arrow[dd, dotted, "\tau"] & & \textsc{Pspace} \arrow[dl, hook, "\iota"] \arrow[dd, "id"] \\
(\mathrm{MPO+ND}) \arrow[rr, two heads, "|\bullet|"] & & \mathrm{X} \arrow[rr, two heads, "{[\bullet]}"] \arrow[dd, "\tau"] & & \mathrm{PR} \arrow[dd, "id"] & \\
 & (\mathrm{MPO}) \cap (\mathrm{QI}) \arrow[rr, dotted, two heads, "|\bullet|"] \arrow[dl, dotted, hook, "\iota"] & & \mathrm{Y'} \arrow[rr, dotted, two heads, "{[\bullet]}"] \arrow[dl, dotted, hook, "\iota"] & & \textsc{Ptime} \arrow[dl, hook, "\iota"] \\
(\mathrm{MPO}) \arrow[rr, two heads, "|\bullet|"] & & \mathrm{Y} \arrow[rr, two heads, "{[\bullet]}"] & & \mathrm{PR} &
\end{tikzcd}
$$

Algorithms respect QI.

There is no transformation from non-deterministic algorithms to first order algorithm *preserving Quasi-Interpretations*.

There are Non-deterministic algorithms admitting a Quasi-Interpretation with no deterministic counterpart.

Future Works

Other Examples

- The framework is generic and could be applied to many cases.
- But we still need to find a good filter and prove:
  $[\![L]\!] = [\![M]\!]$
  $[\![L']\!] \neq [\![M']\!]$

Other Examples

- The framework is generic and could be applied to many cases.
- But we still need to find a good filter and prove:
  $[\![L]\!] = [\![M]\!]$
  $[\![L']\!] \neq [\![M']\!]$
- Fortunately, we have 20 years of ICC with lot of results of this kind to use!
- Linear Logic? Imperative programs? ...

Cross-language comparison

Can we compare systems for imperative programs with Linear Logic systems?

Cross-language comparison

Can we compare systems for imperative programs with Linear Logic systems?

- The key to the proof is that the filter is “the same” in both cases.
- When dealing with algorithms, if we have two filters F and G, we need $|F| = |G|$
  Especially, $[\![F]\!] = [\![G]\!]$ is not sufficient.

Cross-language comparison

Can we compare systems for imperative programs with Linear Logic systems?

- The key to the proof is that the filter is "the same" in both cases.
- When dealing with algorithms, if we have two filters F and G, we need $|F| = |G|$
  Especially, $[\![F]\!] = [\![G]\!]$ is not sufficient.
- But without knowing what an algorithm is, we cannot prove that.

We have to find another way to circumvent this.

Comparing Systems for PTIME

	(1^{st}) \CONS	Interp.	MPO+QI

Comparing Systems for PTIME

	(1^{st}) \CONS	Interp.	MPO+QI
	PTIME	PTIME	PTIME

- The three ICC systems capture the same functions.
- Experiments suggest that it's easier to write MPO+QI programs than (1^{st}) \CONS ones.

Comparing Systems for PTIME

	$(1^{st}) \setminus$CONS	Interp.	MPO+QI
Deterministic	PTIME	PTIME	PTIME
Non-deterministic	PTIME	NPTIME	PSPACE

- The three ICC systems capture the same functions.
- Experiments suggest that it's easier to write MPO+QI programs than $(1^{st}) \setminus$CONS ones.
- Adding Non-determinism reveals a jump in expressiveness.

Comparing Systems for PTIME

	(1^{st}) \CONS	Interp.	MPO+QI
Deterministic	PTIME	PTIME	PTIME
Non-deterministic	PTIME	NPTIME	PSPACE

- The three ICC systems capture the same functions.
- Experiments suggest that it's easier to write MPO+QI programs than (1^{st}) \CONS ones.
- Adding Non-determinism reveals a jump in expressiveness.
- There was more "things" (algorithms) in MPO+QI but we need the Non-determinism to make them appear.

Chemistry

A chemical version of this:

Conclusion

Conclusion

- Generic framework.
- Can tell something about algorithms, under "reasonable" assumptions (or at least we can argue about the assumptions).
- Reuse 20 years of ICC to get more results.
- Can still be extended for more interesting results?
- A step toward a theory of algorithms?