a turing-complete choreography calculus

luís cruz-filipe

(joint work with fabrizio montesi)

department of mathematics and computer science
university of southern denmark

labmag seminar
july 21th, 2015

outline

the zoo of communication

communication & computation

practical consequences

models of communicating systems

process calculi

π-calculus and its variants

- low-level modeling of communication
- too technical for many purposes
- many interesting fragments are undecidable

models of communicating systems

process calculi

π-calculus and its variants

- low-level modeling of communication
- too technical for many purposes
- many interesting fragments are undecidable

choreographies

- global view of the system
- directed communication (from alice to bob)
- deadlock-free by design
- compilable to process calculi

models of communicating systems

process calculi

π-calculus and its variants

- low-level modeling of communication
- too technical for many purposes
- many interesting fragments are undecidable

choreographies

- global view of the system
- directed communication (from alice to bob)
- deadlock-free by design
- compilable to process calculi

actor systems

- even more abstract
- avoid "implementation details" (channels, sessions)

computational expressiveness

$\rightsquigarrow$ trivially turing-complete
(arbitrary computation at each process)

computational expressiveness

$\rightsquigarrow$ trivially turing-complete
(arbitrary computation at each process)

focus: communication

- reduce local computation to a minimum
- reduce system primitives to a minimum
- how far can we go?

computational expressiveness

$\rightsquigarrow$ trivially turing-complete
(arbitrary computation at each process)

focus: communication

- reduce local computation to a minimum
- reduce system primitives to a minimum
- how far can we go?

π-calculus direct encoding of λ-calculus is unsatisfactory

- counter-intuitive notion of computation
- data and programs at the same level

our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)

our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)

A

our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)

$$\mathsf{A} \xrightarrow{\text{computes}} \mathsf{A}'$$

our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)

$$
\begin{tikzcd}
\mathsf{A} \arrow[rr, rightsquigarrow, "\text{computes}"] \arrow[d, Rightarrow, "\text{compiles}"'] & & \mathsf{A}' \arrow[d, Rightarrow, "\text{compiles}"] \\
\mathsf{P}_{\mathsf{A}} & & \mathsf{P}_{\mathsf{A}'}
\end{tikzcd}
$$

our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)

$$\begin{tikzcd}
\mathsf{A} \arrow[r, squiggly, "\text{computes}"] \arrow[d, Rightarrow, "\text{compiles}"'] & \mathsf{A}' \arrow[d, Rightarrow, "\text{compiles}"] \\
\mathsf{P}_{\mathsf{A}} \arrow[r, squiggly, "\text{computes}"] & \mathsf{P}_{\mathsf{A}'}
\end{tikzcd}$$

our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)

$$\mathsf{A} \overset{\text{computes}}{\rightsquigarrow} \mathsf{A}'$$

our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)

$$
\begin{tikzcd}
 & C & & C' \\
A \arrow[ur, dotted, "\text{translates}"] \arrow[rr, squiggly, "\text{computes}"] & & A' \arrow[ur, dotted, "\text{translates}"'] &
\end{tikzcd}
$$

our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)

$$
\begin{tikzcd}
 & C \arrow[rr, squiggly, "\text{computes}"] & & C' \\
A \arrow[ur, dotted, "\text{translates}"] \arrow[rr, squiggly, "\text{computes}"] & & A' \arrow[ur, dotted, "\text{translates}"'] &
\end{tikzcd}
$$

our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)

$$\mathsf{C} \overset{\text{computes}}{\rightsquigarrow} \mathsf{C}'$$

our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)

$$
\begin{tikzcd}
\mathsf{C} \arrow[rr, squiggly, "\text{computes}"] \arrow[d, Rightarrow, "\text{compiles}"'] & & \mathsf{C}' \arrow[d, Rightarrow, "\text{compiles}"] \\
\mathsf{P}_{\mathsf{C}} & & \mathsf{P}_{\mathsf{C}'}
\end{tikzcd}
$$

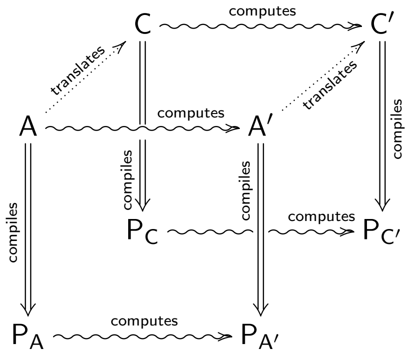
our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)

$$\begin{tikzcd}
\mathsf{C} \arrow[r, rightsquigarrow, "\text{computes}"] \arrow[d, Rightarrow, "\text{compiles}"'] & \mathsf{C}' \arrow[d, Rightarrow, "\text{compiles}"] \\
\mathsf{P}_{\mathsf{C}} \arrow[r, rightsquigarrow, "\text{computes}"] & \mathsf{P}_{\mathsf{C}'}
\end{tikzcd}$$

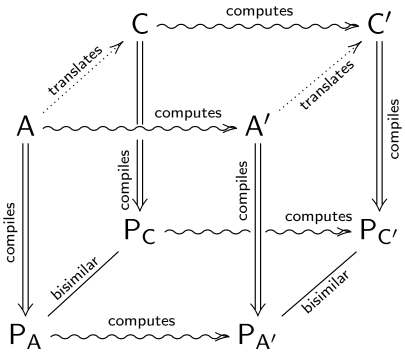
our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)



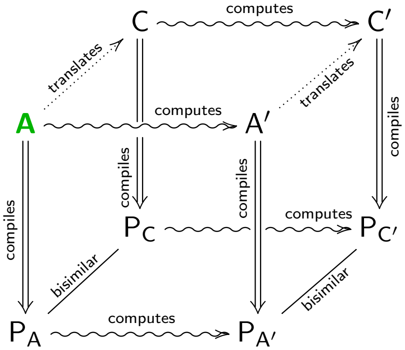
our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)



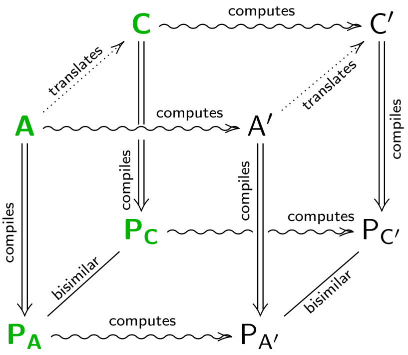
our contribution

- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)



our contribution

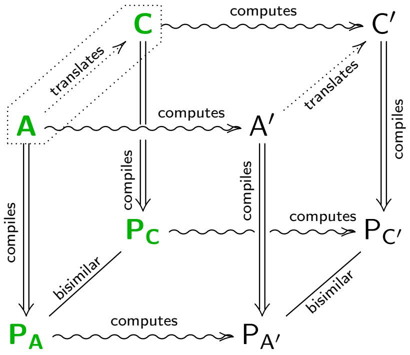
- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)



our contribution

focus of this talk:

- turing-completeness of actor choreographies
- the embedding into channel choreographies



outline

from channels to actors...

channel choreographies

$$
\begin{aligned}
C ::= \quad & \mathbf{0} \mid \eta; C \mid (\nu r)C \\
& \mid \text{if } \mathsf{p}.(e = e') \text{ then } C_1 \text{ else } C_2 \\
& \mid \text{def } X(\tilde{D}) = C_2 \text{ in } C_1 \mid X\langle \tilde{E} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}[\mathsf{A}].e \to \mathsf{q}[\mathsf{B}].x : k \\
& \mid \mathsf{p}[\mathsf{A}] \to \mathsf{q}[\mathsf{B}] : k[l] \\
& \mid \mathsf{p}[\mathsf{A}] \to \mathsf{q}[\mathsf{B}] : k\langle k'[\mathsf{C}] \rangle \\
& \mid \widetilde{\mathsf{p}[\mathsf{A}]} \text{ start } \widetilde{\mathsf{q}[\mathsf{B}]} : a(k)
\end{aligned}
$$

l ::= infinite set of labels

e ::= expressions over some language

from channels to actors...

channel choreographies

$$C ::= \mathbf{0} \mid \eta; C \mid (\nu r) C$$
$$\mid \text{if } \mathsf{p}.(e = e') \text{ then } C_1 \text{ else } C_2$$
$$\mid \text{def } X(\tilde{D}) = C_2 \text{ in } C_1 \mid X\langle \tilde{E} \rangle$$

$$\eta ::= \mathsf{p}[\mathsf{A}].e \to \mathsf{q}[\mathsf{B}].x : k$$
$$\mid \mathsf{p}[\mathsf{A}] \to \mathsf{q}[\mathsf{B}] : k[l]$$
$$\mid \mathsf{p}[\mathsf{A}] \to \mathsf{q}[\mathsf{B}] : k\langle k'[\mathsf{C}] \rangle$$
$$\mid \widetilde{\mathsf{p}[\mathsf{A}]} \text{ start } \widetilde{\mathsf{q}[\mathsf{B}]} : a(k)$$

$l ::=$ infinite set of labels

$e ::=$ expressions over some language

$\leadsto$ fresh names are cool, but irrelevant

from channels to actors...

channel choreographies

$$
\begin{aligned}
C ::= \quad & \mathbf{0} \mid \eta; C \\
& \mid \mathsf{if}\ \mathsf{p}.(e = e')\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \\
& \mid \mathsf{def}\ X(\tilde{D}) = C_2\ \mathsf{in}\ C_1 \mid X\langle \tilde{E} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}[\mathsf{A}].e \rightarrow \mathsf{q}[\mathsf{B}].x : k \\
& \mid \mathsf{p}[\mathsf{A}] \rightarrow \mathsf{q}[\mathsf{B}] : k[l] \\
& \mid \mathsf{p}[\mathsf{A}] \rightarrow \mathsf{q}[\mathsf{B}] : k\langle k'[\mathsf{C}] \rangle
\end{aligned}
$$

l ::= infinite set of labels

e ::= expressions over some language

from channels to actors...

channel choreographies

$$
\begin{aligned}
C ::= \quad & \mathbf{0} \mid \eta; C \\
& \mid \text{if } \mathsf{p}.(e = e') \text{ then } C_1 \text{ else } C_2 \\
& \mid \text{def } X(\tilde{D}) = C_2 \text{ in } C_1 \mid X\langle \tilde{E} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}[\mathsf{A}].e \to \mathsf{q}[\mathsf{B}].x : k \\
& \mid \mathsf{p}[\mathsf{A}] \to \mathsf{q}[\mathsf{B}] : k[l] \\
& \mid \mathsf{p}[\mathsf{A}] \to \mathsf{q}[\mathsf{B}] : k\langle k'[\mathsf{C}] \rangle
\end{aligned}
$$

l ::= infinite set of labels

e ::= expressions over some language

$\rightsquigarrow$ role passing important in practice, but not needed

from channels to actors...

channel choreographies

$$
\begin{aligned}
C ::= \quad & \mathbf{0} \mid \eta; C \\
& \mid \mathsf{if}\ \mathsf{p}.(e = e')\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \\
& \mid \mathsf{def}\ X(\tilde{D}) = C_2\ \mathsf{in}\ C_1 \mid X\langle \tilde{E} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}[\mathsf{A}].e \to \mathsf{q}[\mathsf{B}].x : k \\
& \mid \mathsf{p}[\mathsf{A}] \to \mathsf{q}[\mathsf{B}] : k[l]
\end{aligned}
$$

l ::= infinite set of labels

e ::= expressions over some language

from channels to actors. . .

channel choreographies

$$C ::= \quad \mathbf{0} \mid \eta; C \mid \text{if } \mathsf{p}.(e = e') \text{ then } C_1 \text{ else } C_2 \mid \text{def } X(\tilde{D}) = C_2 \text{ in } C_1 \mid X\langle \tilde{E} \rangle$$

$$\eta ::= \quad \mathsf{p}[\mathsf{A}].e \to \mathsf{q}[\mathsf{B}].x : k \mid \mathsf{p}[\mathsf{A}] \to \mathsf{q}[\mathsf{B}] : k[l]$$

l ::= infinite set of labels

e ::= expressions over some language

$\rightsquigarrow$ . . . but now roles are irrelevant

from channels to actors...

channel choreographies

$$
\begin{aligned}
C ::= \quad & \mathbf{0} \mid \eta; C \\
& \mid \mathsf{if}\ \mathsf{p}.(e = e')\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \\
& \mid \mathsf{def}\ X(\tilde{D}) = C_2\ \mathsf{in}\ C_1 \mid X\langle \tilde{E} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}.e \to \mathsf{q}.x : k \\
& \mid \mathsf{p} \to \mathsf{q} : k[l]
\end{aligned}
$$

l ::= infinite set of labels

e ::= expressions over some language

from channels to actors...

channel choreographies

$$
\begin{aligned}
C ::= \quad & \mathbf{0} \mid \eta; C \\
& \mid \text{if } \mathsf{p}.(e = e') \text{ then } C_1 \text{ else } C_2 \\
& \mid \text{def } X(\tilde{D}) = C_2 \text{ in } C_1 \mid X\langle\tilde{E}\rangle
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}.e \to \mathsf{q}.x : k \\
& \mid \mathsf{p} \to \mathsf{q} : k[l]
\end{aligned}
$$

l ::= infinite set of labels

e ::= expressions over some language

$\rightsquigarrow$ communication can take place over only one channel

from channels to actors...

channel choreographies

$$
\begin{aligned}
C ::= \quad & \mathbf{0} \mid \eta; C \\
& \mid \mathsf{if}\ \mathsf{p}.(e = e')\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \\
& \mid \mathsf{def}\ X(\tilde{D}) = C_2\ \mathsf{in}\ C_1 \mid X\langle \tilde{E} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}.e \to \mathsf{q}.x \\
& \mid \mathsf{p} \to \mathsf{q}[l]
\end{aligned}
$$

l ::= infinite set of labels

e ::= expressions over some language

from channels to actors...

actor choreographies

$$
\begin{aligned}
A ::= \quad & \mathbf{0} \mid \eta; A \\
& \mid \mathsf{if}\ \mathsf{p}.(e = e')\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2 \\
& \mid \mathsf{def}\ X(\tilde{D}) = A_2\ \mathsf{in}\ A_1 \mid X\langle \tilde{E} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}.e \to \mathsf{q}.x \\
& \mid \mathsf{p} \to \mathsf{q}[l]
\end{aligned}
$$

l ::= infinite set of labels

e ::= expressions over some language

from channels to actors...

actor choreographies

$$
\begin{aligned}
A ::= \quad & \mathbf{0} \mid \eta; A \\
& \mid \text{if } \mathsf{p}.(e = e') \text{ then } A_1 \text{ else } A_2 \\
& \mid \text{def } X(\tilde{D}) = A_2 \text{ in } A_1 \mid X\langle \tilde{E} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}.e \to \mathsf{q}.x \\
& \mid \mathsf{p} \to \mathsf{q}[l]
\end{aligned}
$$

l ::= infinite set of labels

e ::= expressions over some language

$\rightsquigarrow$ parameters suddenly do very little

from channels to actors...

actor choreographies

$$A ::= \mathbf{0} \mid \eta; A \mid \text{if } \mathsf{p}.(e = e') \text{ then } A_1 \text{ else } A_2 \mid \text{def } X = A_2 \text{ in } A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q}.x \mid \mathsf{p} \to \mathsf{q}[l]$$

$l ::=$ infinite set of labels

$e ::=$ expressions over some language

from channels to actors...

actor choreographies

$$
\begin{aligned}
A ::= \quad & \mathbf{0} \mid \eta; A \\
& \mid \text{if } \mathsf{p}.(e = e') \text{ then } A_1 \text{ else } A_2 \\
& \mid \text{def } X = A_2 \text{ in } A_1 \mid X
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}.e \to \mathsf{q}.x \\
& \mid \mathsf{p} \to \mathsf{q}[l]
\end{aligned}
$$

l ::= infinite set of labels

e ::= expressions over some language

$\leadsto$ only two labels, only one memory cell...

from channels to actors...

actor choreographies

$$A ::= \mathbf{0} \mid \eta; A$$
$$\mid \text{if } \mathsf{p}.(e = e') \text{ then } A_1 \text{ else } A_2$$
$$\mid \text{def } X = A_2 \text{ in } A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q}$$
$$\mid \mathsf{p} \to \mathsf{q}[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

e ::= expressions over some language

from channels to actors...

actor choreographies

$$A ::= \mathbf{0} \mid \eta; A \mid \mathsf{if}\ \mathsf{p}.(e = e')\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2 \mid \mathsf{def}\ X = A_2\ \mathsf{in}\ A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q} \mid \mathsf{p} \to \mathsf{q}[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$e ::=$ expressions over some language

$\rightsquigarrow$ ...and minimal set of expressions...

from channels to actors...

actor choreographies

$$A ::= \mathbf{0} \mid \eta; A \mid \text{if } \mathsf{p}.(e = e') \text{ then } A_1 \text{ else } A_2 \mid \text{def } X = A_2 \text{ in } A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q} \mid \mathsf{p} \to \mathsf{q}[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

$\rightsquigarrow$ ... requiring a different conditional

from channels to actors...

actor choreographies

$$A ::= \mathbf{0} \mid \eta; A \mid \mathsf{if}\ (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2 \mid \mathsf{def}\ X = A_2\ \mathsf{in}\ A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q} \mid \mathsf{p} \to \mathsf{q}[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

. . . and back again

actor choreographies

$$
\begin{aligned}
A ::= \quad & \mathbf{0} \mid \eta; A \\
& \mid \mathsf{if}\ (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2 \\
& \mid \mathsf{def}\ X = A_2\ \mathsf{in}\ A_1 \mid X
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}.e \to \mathsf{q} \\
& \mid \mathsf{p} \to \mathsf{q}[l]
\end{aligned}
$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

... and back again

actor choreographies

$$A ::= \mathbf{0} \mid \eta; A$$
$$\mid \text{if } (\mathsf{p}[\mathsf{p}].\mathbf{c} = \mathsf{q}[\mathsf{q}].\mathbf{c}) \text{ then } A_1 \text{ else } A_2$$
$$\mid \text{def } X = A_2 \text{ in } A_1 \mid X$$

$$\eta ::= \mathsf{p}[\mathsf{p}].e \to \mathsf{q}[\mathsf{q}] : k$$
$$\mid \mathsf{p}[\mathsf{p}] \to \mathsf{q}[\mathsf{q}] : k[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

$\rightsquigarrow$ reintroduce roles and (one) channel *actor choreographies*

$$
\begin{aligned}
A ::= \quad & \mathbf{0} \mid \eta; A \\
& \mid \mathsf{if}\ (\mathsf{p}[\mathsf{p}].\mathbf{c} = \mathsf{q}[\mathsf{q}].\mathbf{c})\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2 \\
& \mid \mathsf{def}\ X = A_2\ \mathsf{in}\ A_1 \mid X
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}[\mathsf{p}].e \to \mathsf{q}[\mathsf{q}] : k \\
& \mid \mathsf{p}[\mathsf{p}] \to \mathsf{q}[\mathsf{q}] : k[l]
\end{aligned}
$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

. . . and back again

actor choreographies

$$A ::= \mathbf{0} \mid \eta; A \mid \mathsf{q}[\mathsf{q}].x \to \mathsf{p}[\mathsf{p}].y : k; \text{if } \mathsf{p}[\mathsf{p}].(x = y) \text{ then } A_1 \text{ else } A_2 \mid \text{def } X = A_2 \text{ in } A_1 \mid X$$

$$\eta ::= \mathsf{p}[\mathsf{p}].e \to \mathsf{q}[\mathsf{q}].x : k \mid \mathsf{p}[\mathsf{p}] \to \mathsf{q}[\mathsf{q}] : k[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid x \mid \mathsf{s} \cdot x$$

$\rightsquigarrow$ one variable for content, another for testing

... and back again

actor choreographies

$$\begin{aligned} A ::= \quad & \mathbf{0} \mid \eta; A \\ & \mid \mathsf{q}[\mathsf{q}].x \to \mathsf{p}[\mathsf{p}].y : k; \text{if } \mathsf{p}[\mathsf{p}].(x = y) \text{ then } A_1 \text{ else } A_2 \\ & \mid \text{def } X = A_2 \text{ in } A_1 \mid X \end{aligned}$$

$$\begin{aligned} \eta ::= \quad & \mathsf{p}[\mathsf{p}].e \to \mathsf{q}[\mathsf{q}].x : k \\ & \mid \mathsf{p}[\mathsf{p}] \to \mathsf{q}[\mathsf{q}] : k[l] \end{aligned}$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid x \mid \mathsf{s} \cdot x$$

... and back again

actor choreographies

$$
\begin{aligned}
A ::= \quad & \mathbf{0} \mid \eta; A \\
& \mid \mathsf{q}[\mathsf{q}].x \to \mathsf{p}[\mathsf{p}].y : k; \text{if } \mathsf{p}[\mathsf{p}].(x = y) \text{ then } A_1 \text{ else } A_2 \\
& \mid \text{def } X(\tilde{D}) = A_2 \text{ in } A_1 \mid X\langle \tilde{E} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= \quad & \mathsf{p}[\mathsf{p}].e \to \mathsf{q}[\mathsf{q}].x : k \\
& \mid \mathsf{p}[\mathsf{p}] \to \mathsf{q}[\mathsf{q}] : k[l]
\end{aligned}
$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid x \mid \mathsf{s} \cdot x$$

$\rightsquigarrow$ extensively annotate recursive definitions (trivial)

...and back again

$$
\begin{aligned}
A ::= {} & \mathbf{0} \mid \eta; A \\
& \mid \mathsf{q}[\mathsf{q}].x \to \mathsf{p}[\mathsf{p}].y : k; \mathsf{if}\ \mathsf{p}[\mathsf{p}].(x = y)\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2 \\
& \mid \mathsf{def}\ X(\tilde{D}) = A_2\ \mathsf{in}\ A_1 \mid X\langle \tilde{E} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\eta ::= {} & \mathsf{p}[\mathsf{p}].e \to \mathsf{q}[\mathsf{q}].x : k \\
& \mid \mathsf{p}[\mathsf{p}] \to \mathsf{q}[\mathsf{q}] : k[l]
\end{aligned}
$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid x \mid \mathsf{s} \cdot x$$

actor choreographies

$$A ::= \quad \mathbf{0} \mid \eta; A \mid \mathsf{if}\ (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2$$
$$\mid \mathsf{def}\ X = A_2\ \mathsf{in}\ A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q} \mid \mathsf{p} \to \mathsf{q}[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

actor choreographies

actor choreographies

$$A ::= \quad \mathbf{0} \mid \eta; A \mid \mathsf{if}\ (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2$$
$$\mid \mathsf{def}\ X = A_2\ \mathsf{in}\ A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q} \mid \mathsf{p} \to \mathsf{q}[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

urm machine classical model of computation

- similar to physical memory
- memory cells store natural numbers
- memory operations: zero, successor, copy
- jump-on-equal

actor choreographies

actor choreographies

$$A ::= \mathbf{0} \mid \eta; A \mid \text{if } (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c}) \text{ then } A_1 \text{ else } A_2$$
$$\mid \text{def } X = A_2 \text{ in } A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q} \mid \mathsf{p} \to \mathsf{q}[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

urm machine

classical model of computation

- similar to physical memory
- memory cells store natural numbers $\rightsquigarrow$ processes
- memory operations: zero, successor, copy
- jump-on-equal $\rightsquigarrow$ conditional

actor choreographies

actor choreographies

$$A ::= \mathbf{0} \mid \eta; A \mid \mathsf{if}\ (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2$$
$$\mid \mathsf{def}\ X = A_2\ \mathsf{in}\ A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q} \mid \mathsf{p} \to \mathsf{q}[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

but...! very different computation model

- no centralized control
- no self-change

actor choreographies

actor choreographies

$$A ::= \mathbf{0} \mid \eta; A \mid \mathsf{if}\ (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2 \mid \mathsf{def}\ X = A_2\ \mathsf{in}\ A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q} \mid \mathsf{p} \to \mathsf{q}[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

on selections

- not needed for computational completeness
- essential for projectability (e.g. to π-calculus)
- known algorithms for inferring selections

implementation

state a *state* of an actor choreography is a mapping from the set of process names to the set of values

implementation

state a *state* of an actor choreography is a mapping from the set of process names to the set of values

implementation choreography A implements $f : \mathbb{N}^n \to \mathbb{N}$ with inputs $p_1, \ldots, p_n$ and output q if:
for every σ such that $\sigma(p_i) = \ulcorner x_i \urcorner$,

- if $f(\tilde{x})$ is defined, then $A, \sigma \to^* \mathbf{0}, \sigma'$ and $\sigma'(q) = \ulcorner f(\tilde{x}) \urcorner$
- if $f(\tilde{x})$ is not defined, then $A, \sigma \not\to^* \mathbf{0}$ (diverges)

an example: addition

addition from p, q *to* r

$$
\begin{array}{l}
\mathsf{def}\ X = \\
\quad \mathsf{if}\ (\mathsf{r}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then} \\
\qquad \mathsf{p}.\mathbf{c} \to \mathsf{r};\ \mathbf{0} \\
\quad \mathsf{else} \\
\qquad \mathsf{p}.\mathbf{c} \to \mathsf{t};\ \mathsf{t}.\mathsf{s}\cdot\mathbf{c} \to \mathsf{p};\ \mathsf{r}.\mathbf{c} \to \mathsf{t};\ \mathsf{t}.\mathsf{s}\cdot\mathbf{c} \to \mathsf{r};\ X \\
\mathsf{in}\ \mathsf{t}.\varepsilon \to \mathsf{r};\ X
\end{array}
$$

an example: addition

addition from p, q *to* r

$$
\begin{array}{l}
\mathsf{def}\ X = \\
\qquad \mathsf{if}\ (\mathsf{r}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then} \\
\qquad\qquad \mathsf{p}.\mathbf{c} \to \mathsf{r};\ \mathbf{0} \\
\qquad \mathsf{else} \\
\qquad\qquad \mathsf{p}.\mathbf{c} \to \mathsf{t};\ \mathsf{t}.\mathsf{s} \cdot \mathbf{c} \to \mathsf{p};\ \mathsf{r}.\mathbf{c} \to \mathsf{t};\ \mathsf{t}.\mathsf{s} \cdot \mathbf{c} \to \mathsf{r};\ X \\
\mathsf{in}\ \mathsf{t}.\varepsilon \to \mathsf{r};\ X
\end{array}
$$

$\rightsquigarrow$ does not compile!

- projection of p does not know whether to send a message to r or t
- projection of t does not know whether to wait for a message or terminate

an example: addition

addition from p, q *to* r

$$
\begin{aligned}
&\mathsf{def}\ X = \\
&\qquad \mathsf{if}\ (\mathsf{r}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ \mathsf{r} \to \mathsf{p}[\mathrm{L}];\ \mathsf{r} \to \mathsf{q}[\mathrm{L}];\ \mathsf{r} \to \mathsf{t}[\mathrm{L}]; \\
&\qquad\qquad \mathsf{p}.\mathbf{c} \to \mathsf{r};\ \mathbf{0} \\
&\qquad \mathsf{else}\ \mathsf{r} \to \mathsf{p}[\mathrm{R}];\ \mathsf{r} \to \mathsf{q}[\mathrm{R}];\ \mathsf{r} \to \mathsf{t}[\mathrm{R}]; \\
&\qquad\qquad \mathsf{p}.\mathbf{c} \to \mathsf{t};\ \mathsf{t}.\mathsf{s} \cdot \mathbf{c} \to \mathsf{p};\ \mathsf{r}.\mathbf{c} \to \mathsf{t};\ \mathsf{t}.\mathsf{s} \cdot \mathbf{c} \to \mathsf{r};\ X \\
&\mathsf{in}\ \mathsf{t}.\varepsilon \to \mathsf{r};\ X
\end{aligned}
$$

$\leadsto$ does not compile!

- projection of p does not know whether to send a message to r or t
- projection of t does not know whether to wait for a message or terminate

an example: addition

addition from
p, q *to* r

$$
\begin{array}{l}
\mathsf{def}\ X = \\
\qquad \mathsf{if}\ (\mathsf{r}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ \mathsf{r} \to \mathsf{p}[\mathrm{L}];\ \mathsf{r} \to \mathsf{q}[\mathrm{L}];\ \mathsf{r} \to \mathsf{t}[\mathrm{L}]; \\
\qquad\qquad \mathsf{p}.\mathbf{c} \to \mathsf{r};\ \mathbf{0} \\
\qquad \mathsf{else}\ \mathsf{r} \to \mathsf{p}[\mathrm{R}];\ \mathsf{r} \to \mathsf{q}[\mathrm{R}];\ \mathsf{r} \to \mathsf{t}[\mathrm{R}]; \\
\qquad\qquad \mathsf{p}.\mathbf{c} \to \mathsf{t};\ \mathsf{t}.\mathsf{s} \cdot \mathbf{c} \to \mathsf{p};\ \mathsf{r}.\mathbf{c} \to \mathsf{t};\ \mathsf{t}.\mathsf{s} \cdot \mathbf{c} \to \mathsf{r};\ X \\
\mathsf{in}\ \mathsf{t}.\varepsilon \to \mathsf{r};\ X
\end{array}
$$

$\rightsquigarrow$ compiles!

- projections of p and t wait for notification from r
- projection of q also needs to be notified

partial recursive functions i/vi

successor $S : \mathbb{N} \to \mathbb{N}$ such that $S(x) = x + 1$ for all x

partial recursive functions i/vi

successor $S : \mathbb{N} \to \mathbb{N}$ such that $S(x) = x + 1$ for all x

implementation

$$[\![S]\!]^{\mathsf{p} \mapsto \mathsf{q}} = \mathsf{p}.(\mathsf{s} \cdot \mathbf{c}) \to \mathsf{q}$$

partial recursive functions i/vi

successor $S : \mathbb{N} \to \mathbb{N}$ such that $S(x) = x + 1$ for all x

implementation

$$[\![S]\!]^{\mathsf{p} \mapsto \mathsf{q}} = \mathsf{p}.(\mathsf{s} \cdot \mathbf{c}) \to \mathsf{q}$$

soundness

$$\mathsf{p}.(\mathsf{s} \cdot \mathbf{c}) \to \mathsf{q}, \{\mathsf{p} \mapsto \ulcorner x \urcorner\} \longrightarrow \mathbf{0}, \begin{Bmatrix} \mathsf{p} \mapsto \ulcorner x \urcorner \\ \mathsf{q} \mapsto \ulcorner x + 1 \urcorner \end{Bmatrix}$$

partial recursive functions ii/vi

zero $Z : \mathbb{N} \to \mathbb{N}$ such that $S(x) = 0$ for all x

implementation

$$[\![Z]\!]^{\mathsf{p} \mapsto \mathsf{q}} = \mathsf{p}.\varepsilon \to \mathsf{q}$$

soundness

$$\mathsf{p}.\varepsilon \to \mathsf{q}, \{\mathsf{p} \mapsto \ulcorner x \urcorner\} \longrightarrow \mathbf{0}, \begin{Bmatrix} \mathsf{p} \mapsto \ulcorner x \urcorner \\ \mathsf{q} \mapsto \ulcorner 0 \urcorner \end{Bmatrix}$$

partial recursive functions iii/vi

projections $P^n_m : \mathbb{N} \to \mathbb{N}$ such that $P^n_m(x_1, \ldots, x_n) = x_m$ for all $\tilde{x}$

implementation

$$[\![P^n_m]\!]^{\mathsf{p_1,\ldots,p_n \mapsto q}} = \mathsf{p_m}.\mathbf{c} \to \mathsf{q}$$

soundness

$$\mathsf{p_m}.\mathbf{c} \to \mathsf{q}, \{\mathsf{p_i} \mapsto \ulcorner x_i \urcorner\} \longrightarrow \mathbf{0}, \begin{Bmatrix} \mathsf{p_i} \mapsto \ulcorner x_i \urcorner \\ \mathsf{q} \mapsto \ulcorner x_m \urcorner \end{Bmatrix}$$

intermezzo: properties of the encoding

$\rightsquigarrow$ properties we use in inductive constructions

- execution preserves contents of input processes
- all choreographies have exactly one exit point (occurrence of $\mathbf{0}$)

intermezzo: properties of the encoding

$\rightsquigarrow$ properties we use in inductive constructions

- execution preserves contents of input processes
- all choreographies have exactly one exit point (occurrence of $\mathbf{0}$)

sequential composition for processes with only one exit point
$A \fatsemi A'$ is obtained by replacing $\mathbf{0}$ (in A) by A'

intermezzo: properties of the encoding

$\rightsquigarrow$ properties we use in inductive constructions

- execution preserves contents of input processes
- all choreographies have exactly one exit point (occurrence of $\mathbf{0}$)

sequential composition for processes with only one exit point $A \fatsemi A'$ is obtained by replacing $\mathbf{0}$ (in A) by A'

$\rightsquigarrow$ works as expected

- if $A, \sigma \to^* \mathbf{0}, \sigma'$ and $A', \sigma' \to^* \mathbf{0}, \sigma''$, then $A \fatsemi A', \sigma \to^* \mathbf{0}, \sigma''$
- if $A, \sigma \to^* \mathbf{0}, \sigma'$ and A', σ' diverges, then $A \fatsemi A', \sigma$ diverges
- if A, σ diverges, then $A \fatsemi A', \sigma \to \mathbf{0}, \sigma''$ diverges

partial recursive functions iv/vi

composition

$g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$ $\qquad C(f, \tilde{g}) : \mathbb{N}^n \to \mathbb{N}$

$f : \mathbb{N}^k \to \mathbb{N}$ $\qquad \tilde{x} \mapsto f(g_1(\tilde{x}), \ldots, g_k(\tilde{x}))$

partial recursive functions iv/vi

composition

$$g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N} \qquad C(f, \tilde{g}) : \mathbb{N}^n \to \mathbb{N}$$

$$f : \mathbb{N}^k \to \mathbb{N} \qquad \tilde{x} \mapsto f(g_1(\tilde{x}), \ldots, g_k(\tilde{x}))$$

implementation

$$[\![C(f, \tilde{g})]\!]_{\ell}^{\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{q}} = [\![g_1]\!]_{\ell_1}^{\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{r}'_1} \mathbin{;} \ldots \mathbin{;} [\![g_k]\!]_{\ell_k}^{\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{r}'_k} \mathbin{;} [\![f]\!]_{\ell_{k+1}}^{\mathsf{r}'_1,\ldots,\mathsf{r}'_k \mapsto \mathsf{q}}$$

partial recursive functions iv/vi

composition

$$g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N} \qquad C(f, \tilde{g}) : \mathbb{N}^n \to \mathbb{N}$$
$$f : \mathbb{N}^k \to \mathbb{N} \qquad \tilde{x} \mapsto f(g_1(\tilde{x}), \ldots, g_k(\tilde{x}))$$

implementation

$$[\![C(f, \tilde{g})]\!]_{\ell}^{\mathsf{p}_1, \ldots, \mathsf{p}_n \mapsto \mathsf{q}} = [\![g_1]\!]_{\ell_1}^{\mathsf{p}_1, \ldots, \mathsf{p}_n \mapsto \mathsf{r}'_1} \fatsemi \ldots \fatsemi [\![g_k]\!]_{\ell_k}^{\mathsf{p}_1, \ldots, \mathsf{p}_n \mapsto \mathsf{r}'_k} \fatsemi [\![f]\!]_{\ell_{k+1}}^{\mathsf{r}'_1, \ldots, \mathsf{r}'_k \mapsto \mathsf{q}}$$

$\rightsquigarrow$ r'_{i} are auxiliary processes numbered from ℓ: $\mathsf{r}'_{\mathsf{i}} = \mathsf{r}_{\ell+\mathsf{i}-1}$
in recursive calls we increment the counter:
$\ell_{i+1} = \ell_i + \pi(g_i)$

partial recursive functions iv/vi

composition

$$g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N} \qquad C(f, \tilde{g}) : \mathbb{N}^n \to \mathbb{N}$$
$$f : \mathbb{N}^k \to \mathbb{N} \qquad \tilde{x} \mapsto f(g_1(\tilde{x}), \ldots, g_k(\tilde{x}))$$

implementation

$$[\![C(f,\tilde{g})]\!]_\ell^{\mathsf{p_1,\ldots,p_n \mapsto q}} = [\![g_1]\!]_{\ell_1}^{\mathsf{p_1,\ldots,p_n \mapsto r'_1}} \mathbin{;} \ldots \mathbin{;} [\![g_k]\!]_{\ell_k}^{\mathsf{p_1,\ldots,p_n \mapsto r'_k}} \mathbin{;} [\![f]\!]_{\ell_{k+1}}^{\mathsf{r'_1,\ldots,r'_k \mapsto q}}$$

soundness

$$[\![C(f,\tilde{g})]\!]_\ell^{\mathsf{p_1,\ldots,p_n \mapsto q}}, \{\mathsf{p_i} \mapsto \ulcorner x_i \urcorner\} \longrightarrow^* [\![f]\!]_{\ell_{k+1}}^{\mathsf{r'_1,\ldots,r'_k \mapsto q}}, \left\{ \begin{matrix} \mathsf{p_i} \mapsto \ulcorner x_i \urcorner \\ \mathsf{r'_j} \mapsto \ulcorner g_j(\tilde{x}) \urcorner \end{matrix} \right\}$$

partial recursive functions iv/vi

composition

$$g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N} \qquad C(f, \tilde{g}) : \mathbb{N}^n \to \mathbb{N}$$
$$f : \mathbb{N}^k \to \mathbb{N} \qquad \tilde{x} \mapsto f(g_1(\tilde{x}), \ldots, g_k(\tilde{x}))$$

implementation

$$\llbracket C(f,\tilde{g}) \rrbracket_{\ell}^{\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{q}} = \llbracket g_1 \rrbracket_{\ell_1}^{\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{r}'_1} \fatsemi \ldots \fatsemi \llbracket g_k \rrbracket_{\ell_k}^{\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{r}'_k} \fatsemi \llbracket f \rrbracket_{\ell_{k+1}}^{\mathsf{r}'_1,\ldots,\mathsf{r}'_k \mapsto \mathsf{q}}$$

soundness

$$\begin{aligned} & \llbracket C(f,\tilde{g}) \rrbracket_{\ell}^{\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{q}}, \{\mathsf{p}_\mathsf{i} \mapsto \ulcorner x_i \urcorner\} \\ \longrightarrow^* \; & \llbracket f \rrbracket_{\ell_{k+1}}^{\mathsf{r}'_1,\ldots,\mathsf{r}'_k \mapsto \mathsf{q}}, \left\{ \begin{array}{l} \mathsf{p}_\mathsf{i} \mapsto \ulcorner x_i \urcorner \\ \mathsf{r}'_\mathsf{j} \mapsto \ulcorner g_j(\tilde{x}) \urcorner \end{array} \right\} \\ \longrightarrow^* \; & \mathbf{0}, \left\{ \begin{array}{l} \mathsf{p}_\mathsf{i} \mapsto \ulcorner x_i \urcorner \\ \mathsf{r}'_\mathsf{j} \mapsto \ulcorner g_j(\tilde{x}) \urcorner \\ \mathsf{q} \mapsto \ulcorner f(\widetilde{g(\tilde{x})}) \urcorner \end{array} \right\} \end{aligned}$$

partial recursive functions iv/vi

composition

$$g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N} \qquad C(f, \tilde{g}) : \mathbb{N}^n \to \mathbb{N}$$
$$f : \mathbb{N}^k \to \mathbb{N} \qquad \tilde{x} \mapsto f(g_1(\tilde{x}), \ldots, g_k(\tilde{x}))$$

implementation

$$[\![C(f, \tilde{g})]\!]_\ell^{\mathsf{p}_1, \ldots, \mathsf{p}_\mathsf{n} \mapsto \mathsf{q}} = [\![g_1]\!]_{\ell_1}^{\mathsf{p}_1, \ldots, \mathsf{p}_\mathsf{n} \mapsto \mathsf{r}'_1} \fatsemi \ldots \fatsemi [\![g_k]\!]_{\ell_k}^{\mathsf{p}_1, \ldots, \mathsf{p}_\mathsf{n} \mapsto \mathsf{r}'_\mathsf{k}} \fatsemi [\![f]\!]_{\ell_{k+1}}^{\mathsf{r}'_1, \ldots, \mathsf{r}'_\mathsf{k} \mapsto \mathsf{q}}$$

soundness

if $g_j(\tilde{x})$ is undefined the corresponding step diverges and likewise for $f(\widetilde{g(\tilde{x})})$

recursion

$$f : \mathbb{N}^n \to \mathbb{N} \qquad g : \mathbb{N}^{n+2} \to \mathbb{N}$$

$$h = R(f, g) : \mathbb{N}^{n+1} \to \mathbb{N}$$

$$\tilde{x} \mapsto \begin{cases} f(\vec{x}) & x_0 = 0 \\ g(k, h(k, \tilde{x}), \tilde{x}) & x_0 = k + 1 \end{cases}$$

partial recursive functions v/vi

recursion

$$f : \mathbb{N}^n \to \mathbb{N} \qquad g : \mathbb{N}^{n+2} \to \mathbb{N}$$

$$h = R(f,g) : \mathbb{N}^{n+1} \to \mathbb{N}$$

$$\tilde{x} \mapsto \begin{cases} f(\vec{x}) & x_0 = 0 \\ g(k, h(k,\tilde{x}), \tilde{x}) & x_0 = k+1 \end{cases}$$

implementation

$$
\begin{aligned}
&[\![h]\!]^{\mathsf{p_0,\ldots,p_n \mapsto q}} = \\
&\quad \mathsf{def}\ T = \mathsf{if}\ \mathsf{r_c}.\mathbf{c} = \mathsf{p_0}.\mathbf{c}\ \mathsf{then}\ \mathsf{q}'.\mathbf{c} \to \mathsf{q};\ \mathbf{0} \\
&\qquad\qquad\qquad \mathsf{else}\ [\![g]\!]_{\ell_g}^{\mathsf{r_c,q',p_1,\ldots,p_n \mapsto r_t}} \fatsemi \mathsf{r_t}.\mathbf{c} \to \mathsf{q}'; \\
&\qquad\qquad\qquad\qquad \mathsf{r_c}.\mathbf{c} \to \mathsf{r_t};\ \mathsf{r_t}.(\mathsf{s}\cdot\mathbf{c}) \to \mathsf{r_c};\ T \\
&\quad \mathsf{in}\ [\![f]\!]_{\ell_f}^{\mathsf{p_1,\ldots,p_n \mapsto q'}} \fatsemi \mathsf{r_t}.\varepsilon \to \mathsf{r_c};\ T
\end{aligned}
$$

partial recursive functions v/vi

recursion

$$f : \mathbb{N}^n \to \mathbb{N} \qquad g : \mathbb{N}^{n+2} \to \mathbb{N}$$

$$h = R(f,g) : \mathbb{N}^{n+1} \to \mathbb{N}$$

$$\tilde{x} \mapsto \begin{cases} f(\vec{x}) & x_0 = 0 \\ g(k, h(k,\tilde{x}), \tilde{x}) & x_0 = k+1 \end{cases}$$

implementation

$$[\![h]\!]^{\mathsf{p}_0,\ldots,\mathsf{p}_n \mapsto \mathsf{q}} =$$

$$\begin{aligned} &\text{def } T = \text{if } \mathsf{r}_\mathsf{c}.\mathbf{c} = \mathsf{p}_0.\mathbf{c} \text{ then } \mathsf{q}'.\mathbf{c} \to \mathsf{q};\ \mathbf{0} \\ &\qquad\qquad \text{else } [\![g]\!]_{\ell_g}^{\mathsf{r}_\mathsf{c},\mathsf{q}',\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{r}_\mathsf{t}} \fatsemi \mathsf{r}_\mathsf{t}.\mathbf{c} \to \mathsf{q}'; \\ &\qquad\qquad\qquad \mathsf{r}_\mathsf{c}.\mathbf{c} \to \mathsf{r}_\mathsf{t};\ \mathsf{r}_\mathsf{t}.(\mathsf{s}\cdot\mathbf{c}) \to \mathsf{r}_\mathsf{c};\ T \\ &\text{in } [\![f]\!]_{\ell_f}^{\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{q}'} \fatsemi \mathsf{r}_\mathsf{t}.\varepsilon \to \mathsf{r}_\mathsf{c};\ T \end{aligned}$$

soundness by induction (simple)

minimization

$f : \mathbb{N}^{n+1} \to \mathbb{N}$

$M(f) : \mathbb{N}^n \to \mathbb{N}$

$\tilde{x} \mapsto \mu y . f(\vec{x}, y) = 0$

minimization

$$f : \mathbb{N}^{n+1} \to \mathbb{N} \qquad M(f) : \mathbb{N}^n \to \mathbb{N}$$
$$\tilde{x} \mapsto \mu y . f(\vec{x}, y) = 0$$

implementation

$$[\![M(f)]\!]^{\mathsf{p}_1,\ldots,\mathsf{p}_\mathsf{n} \mapsto \mathsf{q}} =$$
$$\mathsf{def}\ T = [\![f]\!]_{\ell_f}^{\mathsf{p}_1,\ldots,\mathsf{p}_\mathsf{n},\mathsf{r}_\mathsf{c} \mapsto \mathsf{q}'} \mathbin{⨟} \mathsf{r}_\mathsf{c}.\varepsilon \to \mathsf{r}_\mathsf{z};$$
$$\mathsf{if}\ \mathsf{r}_\mathsf{z}.\mathbf{c} = \mathsf{q}'.\mathbf{c}\ \mathsf{then}\ \mathsf{r}_\mathsf{c}.\mathbf{c} \to \mathsf{q};\ \mathbf{0}$$
$$\mathsf{else}\ \mathsf{r}_\mathsf{c}.\mathbf{c} \to \mathsf{r}_\mathsf{z};\ \mathsf{r}_\mathsf{z}.(\mathsf{s} \cdot \mathbf{c}) \to \mathsf{r}_\mathsf{c};\ T$$
$$\mathsf{in}\ \mathsf{r}_\mathsf{z}.\varepsilon \to \mathsf{r}_\mathsf{c};\ T$$

minimization

$$f : \mathbb{N}^{n+1} \to \mathbb{N} \qquad M(f) : \mathbb{N}^n \to \mathbb{N}$$
$$\tilde{x} \mapsto \mu y. f(\vec{x}, y) = 0$$

implementation

$$\begin{aligned}
&[\![M(f)]\!]^{\mathsf{p}_1,\ldots,\mathsf{p}_\mathsf{n} \mapsto \mathsf{q}} = \\
&\quad \mathsf{def}\ T = [\![f]\!]_{\ell_f}^{\mathsf{p}_1,\ldots,\mathsf{p}_\mathsf{n},\mathsf{r_c} \mapsto \mathsf{q}'} \fatsemi \mathsf{r_c}.\varepsilon \to \mathsf{r_z}; \\
&\qquad\quad \mathsf{if}\ \mathsf{r_z}.\mathbf{c} = \mathsf{q}'.\mathbf{c}\ \mathsf{then}\ \mathsf{r_c}.\mathbf{c} \to \mathsf{q};\ \mathbf{0} \\
&\qquad\quad \mathsf{else}\ \mathsf{r_c}.\mathbf{c} \to \mathsf{r_z};\ \mathsf{r_z}.(\mathsf{s} \cdot \mathbf{c}) \to \mathsf{r_c};\ T \\
&\quad \mathsf{in}\ \mathsf{r_z}.\varepsilon \to \mathsf{r_c};\ T
\end{aligned}$$

soundness

by induction (simple)

minimality

actor choreographies

$$A ::= \quad \mathbf{0} \mid \eta; A \mid \mathsf{if}\ (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2$$
$$\mid \mathsf{def}\ X = A_2\ \mathsf{in}\ A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q} \mid \mathsf{p} \to \mathsf{q}[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

- no exit points $\rightsquigarrow$ nothing terminates
- no communication $\rightsquigarrow$ no output
- less expressions $\rightsquigarrow$ cannot compute base cases
- no selection $\rightsquigarrow$ not everything is projectable
- no conditions $\rightsquigarrow$ termination is decidable
- no recursion $\rightsquigarrow$ everything terminates

minimality

actor choreographies

$$A ::= \mathbf{0} \mid \eta; A \mid \mathsf{if}\ (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2 \mid \mathsf{def}\ X = A_2\ \mathsf{in}\ A_1 \mid X$$

$$\eta ::= \mathsf{p}.e \to \mathsf{q} \mid \mathsf{p} \to \mathsf{q}[l]$$

$$l ::= \mathrm{L} \mid \mathrm{R}$$

$$e ::= \varepsilon \mid \mathbf{c} \mid \mathsf{s} \cdot \mathbf{c}$$

- only zero-testing $\rightsquigarrow$ termination is decidable (skipping proof. . .)
- only (arbitrary) constant-testing $\rightsquigarrow$ termination is decidable

outline

the zoo of communication

communication & computation

practical consequences

what we get

- sound encoding of partial recursive functions as actor choreographies

what we get

- sound encoding of partial recursive functions as actor choreographies
- by embedding into channel choreographies $\rightsquigarrow$ sound encoding of partial recursive functions as channel choreographies

what we get

- sound encoding of partial recursive functions as actor choreographies
- by embedding into channel choreographies $\rightsquigarrow$ sound encoding of partial recursive functions as channel choreographies
- by adding necessary selections (deterministically) $\rightsquigarrow$ sound encoding of partial recursive functions as actor processes

what we get

- sound encoding of partial recursive functions as actor choreographies
- by embedding into channel choreographies $\rightsquigarrow$ sound encoding of partial recursive functions as channel choreographies
- by adding necessary selections (deterministically) $\rightsquigarrow$ sound encoding of partial recursive functions as actor processes
- by adding necessary selections and embedding into channel choreographies $\rightsquigarrow$ sound encoding of partial recursive functions as channel processes (π-calculus)

making it more beautiful

additional primitives give more structure

- generation of fresh names “hides” auxiliary processes

making it more beautiful

additional primitives give more structure

- generation of fresh names "hides" auxiliary processes

improving the embedding

- state is encoded as a substitution
- ignoring state: functional process
  (needs a context to set up inputs)

making it more beautiful

additional primitives give more structure

- generation of fresh names "hides" auxiliary processes

improving the embedding

- state is encoded as a substitution
- ignoring state: functional process
(needs a context to set up inputs)

operational proof of completeness for π-calculus

- by slight tweaking: process that "waits" for parallel components with input and output

conclusions

- turing-completeness of actor choreographies
- minimal set of primitives
- identifies a deadlock-free, turing-complete fragment of π-calculus

thank you!