a turing-complete choreography calculus

luís cruz-filipe

(joint work with fabrizio montesi)

department of mathematics and computer science
university of southern denmark

amílcar sernadas festschrift
april 23th, 2016

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

choreographies and computation

⇝ trivially turing-complete
(arbitrary computation at each process)

choreographies and computation

- trivially turing-complete
  (arbitrary computation at each process)
- typically geared towards applications
  (many complex primitives)

choreographies and computation

⇝ trivially turing-complete
(arbitrary computation at each process)

⇝ typically geared towards applications
(many complex primitives)

focus communication

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

choreographies and computation

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

$\rightsquigarrow$ typically geared towards applications
(many complex primitives)

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)

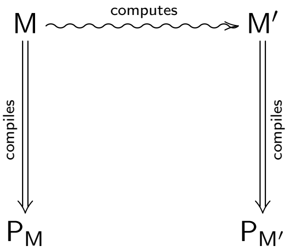
our contribution

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

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

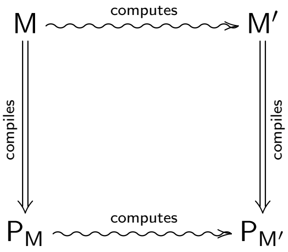
our contribution

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



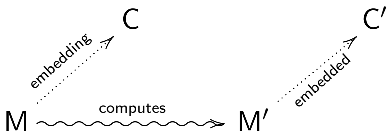
our contribution

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



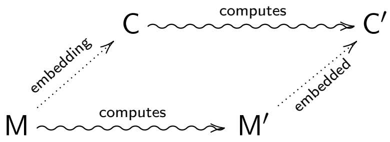
our contribution

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



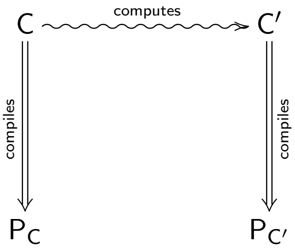
our contribution

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



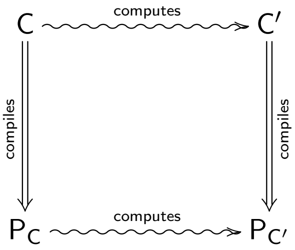
our contribution

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



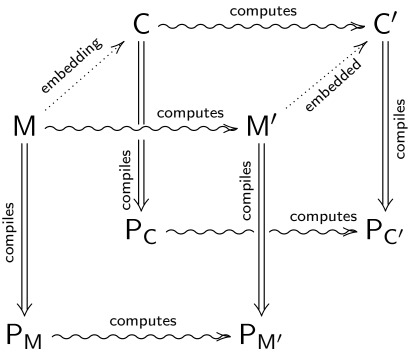
our contribution

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



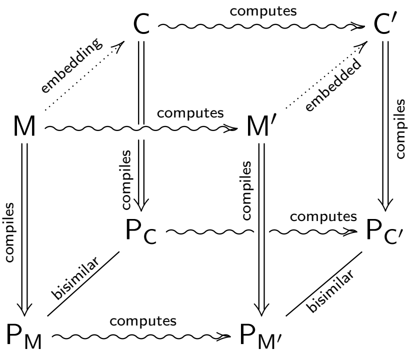
our contribution

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



our contribution

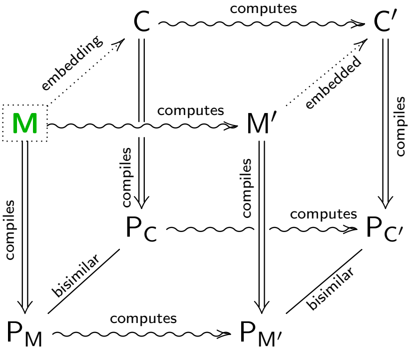
- i/o-based notion of function implementation
- computation by message-passing
- reminiscent of memory models (e.g. urm)



our contribution

focus of this talk:

- minimal choreographies
- their turing completeness



outline

the zoo of communication

communication & computation

practical consequences

typical primitives in choreographies

- termination
- message passing
- label selection
- conditionals
- recursion
- process creation
- channel creation
- channel passing
- role assignment
- ...

typical primitives in choreographies

- termination
- message passing
- label selection
- conditionals
- recursion
- process creation
- channel creation
- channel passing
- role assignment
- ...

typical primitives in choreographies

- termination
- message passing
- label selection
- conditionals
- recursion
- process creation
- channel creation
- channel passing
- role assignment
- ...

minimal choreographies

$$M ::= \quad \mathbf{0} \mid \eta; M \mid \mathsf{if}\ (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2$$
$$\mid \mathsf{def}\ X = M_2\ \mathsf{in}\ M_1 \mid X$$

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

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

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

minimal choreographies

minimal choreographies

$$M ::= \quad \mathbf{0} \mid \eta; M \mid \mathsf{if}\ (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2$$
$$\mid \mathsf{def}\ X = M_2\ \mathsf{in}\ M_1 \mid X$$

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

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

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

urm machine classical model of computation

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

minimal choreographies

minimal choreographies

$$M ::= \mathbf{0} \mid \eta; M \mid \text{if } (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c}) \text{ then } M_1 \text{ else } M_2$$
$$\mid \text{def } X = M_2 \text{ in } M_1 \mid X$$

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

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

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

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

minimal choreographies

minimal choreographies

$$M ::= \mathbf{0} \mid \eta; M \mid \mathsf{if}\ (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2$$
$$\mid \mathsf{def}\ X = M_2\ \mathsf{in}\ M_1 \mid X$$

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

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

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

but...! very different computation model

- no centralized control
- no self-change

minimal choreographies

minimal choreographies

$$M ::= \mathbf{0} \mid \eta; M \mid \text{if } (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c}) \text{ then } M_1 \text{ else } M_2 \mid \text{def } X = M_2 \text{ in } M_1 \mid X$$

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

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

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

on selections

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

implementation of functions

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

implementation of functions

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

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

an example: addition

addition
from p, q *to* r
using t

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

an example: addition

addition
from p, q *to* r
using t

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

$\rightsquigarrow$ 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
using t

def $X =$

  if (r.**c** = q.**c**) then r → p[L]; r → q[L]; r → t[L];
    p.**c** → r; **0**
  else r → p[R]; r → q[R]; r → t[R];
    p.**c** → t; t.(s · **c**) → p;
    r.**c** → t; t.(s · **c**) → r; X
in t.ε → r; X

$\rightsquigarrow$ 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
using t

$\mathsf{def}\ X =$

$\quad \mathsf{if}\ (\mathsf{r}.\mathbf{c} = \mathsf{q}.\mathbf{c})\ \mathsf{then}\ \mathsf{r} \to \mathsf{p}[\mathrm{L}];\ \mathsf{r} \to \mathsf{q}[\mathrm{L}];\ \mathsf{r} \to \mathsf{t}[\mathrm{L}];$

$\qquad \mathsf{p}.\mathbf{c} \to \mathsf{r};\ \mathbf{0}$

$\quad \mathsf{else}\ \mathsf{r} \to \mathsf{p}[\mathrm{R}];\ \mathsf{r} \to \mathsf{q}[\mathrm{R}];\ \mathsf{r} \to \mathsf{t}[\mathrm{R}];$

$\qquad \mathsf{p}.\mathbf{c} \to \mathsf{t};\ \mathsf{t}.(\mathsf{s} \cdot \mathbf{c}) \to \mathsf{p};$

$\qquad \mathsf{r}.\mathbf{c} \to \mathsf{t};\ \mathsf{t}.(\mathsf{s} \cdot \mathbf{c}) \to \mathsf{r};\ X$

$\mathsf{in}\ \mathsf{t}.\varepsilon \to \mathsf{r};\ X$

$\rightsquigarrow$ 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

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

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

⇝ 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 $M \fatsemi M'$ is obtained by replacing $\mathbf{0}$ (in M) by M'

intermezzo: properties of the encoding

$\leadsto$ 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
$M \fatsemi M'$ is obtained by replacing $\mathbf{0}$ (in M) by M'

$\leadsto$ works as expected

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

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

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

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

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

⇝ 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

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

soundness

$$[\![C(f, \tilde{g})]\!]_\ell^{\mathsf{p}_1, \ldots, \mathsf{p}_n \mapsto \mathsf{q}}, \{\mathsf{p}_\mathsf{i} \mapsto \ulcorner x_i \urcorner\} \longrightarrow^* [\![f]\!]_{\ell_{k+1}}^{\mathsf{r}'_1, \ldots, \mathsf{r}'_k \mapsto \mathsf{q}}, \begin{Bmatrix} \mathsf{p}_\mathsf{i} \mapsto \ulcorner x_i \urcorner \\ \mathsf{r}'_\mathsf{j} \mapsto \ulcorner g_j(\tilde{x}) \urcorner \end{Bmatrix}$$

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

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

soundness

$$[\![C(f, \tilde{g})]\!]_{\ell}^{\mathsf{p}_1, \ldots, \mathsf{p}_\mathsf{n} \mapsto \mathsf{q}}, \{\mathsf{p}_\mathsf{i} \mapsto \ulcorner x_i \urcorner\}$$

$$\longrightarrow^* [\![f]\!]_{\ell_{k+1}}^{\mathsf{r}'_1, \ldots, \mathsf{r}'_\mathsf{k} \mapsto \mathsf{q}}, \begin{Bmatrix} \mathsf{p}_\mathsf{i} \mapsto \ulcorner x_i \urcorner \\ \mathsf{r}'_\mathsf{j} \mapsto \ulcorner g_j(\tilde{x}) \urcorner \end{Bmatrix}$$

$$\longrightarrow^* \mathbf{0}, \begin{Bmatrix} \mathsf{p}_\mathsf{i} \mapsto \ulcorner x_i \urcorner \\ \mathsf{r}'_\mathsf{j} \mapsto \ulcorner g_j(\tilde{x}) \urcorner \\ \mathsf{q} \mapsto \ulcorner f(\widetilde{g(\tilde{x})}) \urcorner \end{Bmatrix}$$

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

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

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

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

$$\text{def } T = \text{if } \mathsf{r_c}.\mathbf{c} = \mathsf{p_0}.\mathbf{c} \text{ then } \mathsf{q}'.\mathbf{c} \to \mathsf{q};\ \mathbf{0}$$

$$\text{else } [\![g]\!]_{\ell_g}^{\mathsf{r_c},\mathsf{q}',\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{r_t}} \fatsemi \mathsf{r_t}.\mathbf{c} \to \mathsf{q}';$$

$$\mathsf{r_c}.\mathbf{c} \to \mathsf{r_t};\ \mathsf{r_t}.(\mathsf{s} \cdot \mathbf{c}) \to \mathsf{r_c};\ T$$

$$\text{in } [\![f]\!]_{\ell_f}^{\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{q}'} \fatsemi \mathsf{r_t}.\varepsilon \to \mathsf{r_c};\ T$$

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

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

$$\text{def } T = \text{if } \mathsf{r_c}.\mathbf{c} = \mathsf{p}_0.\mathbf{c} \text{ then } \mathsf{q}'.\mathbf{c} \to \mathsf{q};\ \mathbf{0}$$

$$\text{else } [\![g]\!]_{\ell_g}^{\mathsf{r_c},\mathsf{q}',\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{r_t}} \mathbin{;} \mathsf{r_t}.\mathbf{c} \to \mathsf{q}';$$

$$\mathsf{r_c}.\mathbf{c} \to \mathsf{r_t};\ \mathsf{r_t}.(\mathsf{s} \cdot \mathbf{c}) \to \mathsf{r_c};\ T$$

$$\text{in } [\![f]\!]_{\ell_f}^{\mathsf{p}_1,\ldots,\mathsf{p}_n \mapsto \mathsf{q}'} \mathbin{;} \mathsf{r_t}.\varepsilon \to \mathsf{r_c};\ T$$

soundness

by induction (simple)

partial recursive functions vi/vi

minimization

$$f : \mathbb{N}^{n+1} \to \mathbb{N} \qquad M(f) : \mathbb{N}^n \to \mathbb{N}$$

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

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

$$[\![M(f)]\!]^{\mathsf{p_1,\ldots,p_n \mapsto q}} =$$
$$\text{def } T = [\![f]\!]_{\ell_f}^{\mathsf{p_1,\ldots,p_n,r_c \mapsto q'}} \fatsemi \mathsf{r_c}.\varepsilon \to \mathsf{r_z};$$
$$\text{if } \mathsf{r_z}.\mathbf{c} = \mathsf{q'}.\mathbf{c} \text{ then } \mathsf{r_c}.\mathbf{c} \to \mathsf{q};\ \mathbf{0}$$
$$\text{else } \mathsf{r_c}.\mathbf{c} \to \mathsf{r_z};\ \mathsf{r_z}.(\mathsf{s} \cdot \mathbf{c}) \to \mathsf{r_c};\ T$$
$$\text{in } \mathsf{r_z}.\varepsilon \to \mathsf{r_c};\ T$$

partial recursive functions vi/vi

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

$$[\![M(f)]\!]^{\mathsf{p_1,\ldots,p_n \mapsto q}} =$$
$$\text{def } T = [\![f]\!]_{\ell_f}^{\mathsf{p_1,\ldots,p_n,r_c \mapsto q'}} \fatsemi \mathsf{r_c}.\varepsilon \to \mathsf{r_z};$$
$$\text{if } \mathsf{r_z}.\mathbf{c} = \mathsf{q}'.\mathbf{c} \text{ then } \mathsf{r_c}.\mathbf{c} \to \mathsf{q};\ \mathbf{0}$$
$$\text{else } \mathsf{r_c}.\mathbf{c} \to \mathsf{r_z};\ \mathsf{r_z}.(\mathsf{s} \cdot \mathbf{c}) \to \mathsf{r_c};\ T$$
$$\text{in } \mathsf{r_z}.\varepsilon \to \mathsf{r_c};\ T$$

soundness

by induction (simple)

minimality

minimal choreographies

$$M ::= \mathbf{0} \mid \eta; M \mid \text{if } (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c}) \text{ then } M_1 \text{ else } M_2$$
$$\mid \text{def } X = M_2 \text{ in } M_1 \mid X$$

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

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

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

- 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

minimal choreographies

$$M ::= \mathbf{0} \mid \eta; M \mid \text{if } (\mathsf{p}.\mathbf{c} = \mathsf{q}.\mathbf{c}) \text{ then } M_1 \text{ else } M_2 \mid \text{def } X = M_2 \text{ in } M_1 \mid X$$

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

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

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

- 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 minimal choreographies

what we get

- sound encoding of partial recursive functions as minimal choreographies
- by embedding into other choreography models $\rightsquigarrow$ sound encoding of partial recursive functions in that model

what we get

- sound encoding of partial recursive functions as minimal choreographies
- by embedding into other choreography models $\rightsquigarrow$ sound encoding of partial recursive functions in that model
- by adding necessary selections (deterministically) $\rightsquigarrow$ sound encoding of partial recursive functions as minimal processes

what we get

- sound encoding of partial recursive functions as minimal choreographies
- by embedding into other choreography models $\rightsquigarrow$ sound encoding of partial recursive functions in that model
- by adding necessary selections (deterministically) $\rightsquigarrow$ sound encoding of partial recursive functions as minimal processes
- by adding necessary selections and embedding into other choreography models $\rightsquigarrow$ sound encoding of partial recursive functions in a process model (in particular, π-calculus)

conclusions

- turing completeness of minimal choreographies
- minimal set of primitives
- identifies a deadlock-free, turing-complete fragment of π-calculus
- core language for studying fundamental properties of choreographies

thank you!