a core model for choreographic programming

luís cruz-filipe

(joint work with fabrizio montesi)

department of mathematics and computer science
university of southern denmark

facs'16, besançon, france
october 21st, 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?

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)

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

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

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

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

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

minimal choreographies

minimal choreographies

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

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

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

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

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

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

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

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

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

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 / loop

minimal choreographies

minimal choreographies

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

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

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

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

but...! very different computation model

- no centralized control
- no self-change

minimal choreographies

minimal choreographies

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

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

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

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

on selections

- not needed for computational completeness
- useful 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 $p_1, \dots, p_n$ 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

$\mathsf{def}\ X =$

$\quad \mathsf{if}\ (\mathsf{r}.* = \mathsf{q}.*)\ \mathsf{then}$

$\quad\quad \mathsf{p}.* \rightarrow \mathsf{r};\ \mathbf{0}$

$\quad \mathsf{else}$

$\quad\quad \mathsf{p}.* \rightarrow \mathsf{t};\ \mathsf{t}.(\mathsf{s} \cdot *) \rightarrow \mathsf{p};$

$\quad\quad \mathsf{r}.* \rightarrow \mathsf{t};\ \mathsf{t}.(\mathsf{s} \cdot *) \rightarrow \mathsf{r};\ X$

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

an example: addition

addition from p, q *to* r *using* t

def $X =$

if (r.$*$ = q.$*$) then

p.$* \to$ r; $\mathbf{0}$

else

p.$* \to$ t; t.(s $\cdot *$) $\to$ p;

r.$* \to$ t; t.(s $\cdot *$) $\to$ r; X

in t.$\varepsilon \to$ r; X

$\leadsto$ 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
using t

def X =

  if (r.$*$ = q.$*$) then r $\to$ p[L]; r $\to$ q[L]; r $\to$ t[L];

    p.$*$ $\to$ r; $\mathbf{0}$

  else r $\to$ p[R]; r $\to$ q[R]; r $\to$ t[R];

    p.$*$ $\to$ t; t.(s $\cdot$ $*$) $\to$ p;

    r.$*$ $\to$ t; t.(s $\cdot$ $*$) $\to$ r; X

in t.ε $\to$ 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

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

$\rightsquigarrow$ 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

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

partial recursive functions i/vi

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

implementation

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

soundness

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

partial recursive functions ii/vi

zero

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

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

soundness

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

inductive cases (omitted)

three recursive constructions (see paper)

- composition
- recursion
- minimization

$\rightsquigarrow$ composition executes in parallel

minimality

minimal choreographies

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

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

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

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

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

minimality

minimal choreographies

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

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

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

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

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

minimality

minimal choreographies

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

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

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

$$e ::= \varepsilon \mid * \mid \textsf{s} \cdot *$$

- selections can be encoded as communications (but. . .)

outline

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!