foundations of choreographies

luís cruz-filipe

(joint work with fabrizio montesi)

department of mathematics and computer science
university of southern denmark

betty meeting
october 6th, 2016

a core choreography calculus

goal develop a minimalistic choreography calculus

- study foundational questions
- obtain general results

a core choreography calculus

goal develop a minimalistic choreography calculus

- study foundational questions
- obtain general results

primitives what characterizes a *choreography* language?

- "alice to bob"-style communication: $A.e \to B$
- label (choice) selection: $A \to B[\ell]$

a core choreography calculus

goal develop a minimalistic choreography calculus

- study foundational questions
- obtain general results

primitives what characterizes a *choreography* language?

- "alice to bob"-style communication: $A.e \to B$
- label (choice) selection: $A \to B[\ell]$

$\rightsquigarrow$ other common choreographic primitives

- process creation
- channel creation and channel passing
- role assignment
- ...

our core model

core choreographies

$$C ::= \quad \mathbf{0} \mid \eta; C \mid \text{if } (\mathsf{p}.* = \mathsf{q}.*) \text{ then } C_1 \text{ else } C_2$$
$$\mid \text{def } X = C_2 \text{ in } C_1 \mid X$$

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

our core model

core choreographies

$$C ::= \mathbf{0} \mid \eta; C \mid \text{if } (\mathsf{p}.* = \mathsf{q}.*) \text{ then } C_1 \text{ else } C_2 \mid \text{def } X = C_2 \text{ in } C_1 \mid X$$

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

inspiration

memory models

- similar to physical memory
- memory cells as processes

our core model

core choreographies

$$C ::= \mathbf{0} \mid \eta; C \mid \text{if } (\mathsf{p}.* = \mathsf{q}.*) \text{ then } C_1 \text{ else } C_2 \mid \text{def } X = C_2 \text{ in } C_1 \mid X$$

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

inspiration memory models

- similar to physical memory
- memory cells as processes

but. . . ! different from classic computation models

- no centralized control
- no self-change

our core model

core choreographies

$$C ::= \mathbf{0} \mid \eta; C \mid \text{if } (\mathsf{p}.* = \mathsf{q}.*) \text{ then } C_1 \text{ else } C_2 \mid \text{def } X = C_2 \text{ in } C_1 \mid X$$

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

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

semantics the transition semantics of CC is standard (using swap relation)

our core model

core choreographies

$$C ::= \mathbf{0} \mid \eta; C \mid \text{if } (\mathsf{p}.* = \mathsf{q}.*) \text{ then } C_1 \text{ else } C_2 \mid \text{def } X = C_2 \text{ in } C_1 \mid X$$

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

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

semantics the transition semantics of CC is standard (using swap relation)

theorem there exists a sound and faithful endpoint projection from CC into a minimal process calculus

concepts and properties

implementation — i/o-based notion of function implementation by a choreography

concurrency — notion of (full) parallel execution

concepts and properties

implementation	i/o-based notion of function implementation by a choreography
concurrency	notion of (full) parallel execution
theorem	unprojectable choreographies can be amended (by inferring label selections to add)

concepts and properties

implementation	i/o-based notion of function implementation by a choreography
concurrency	notion of (full) parallel execution
theorem	unprojectable choreographies can be amended (by inferring label selections to add)
theorem	label selections can be encoded as value communications

concepts and properties

implementation	i/o-based notion of function implementation by a choreography
concurrency	notion of (full) parallel execution
theorem	unprojectable choreographies can be amended (by inferring label selections to add)
theorem	label selections can be encoded as value communications
theorem	CC is turing complete

concepts and properties

implementation	i/o-based notion of function implementation by a choreography
concurrency	notion of (full) parallel execution
theorem	unprojectable choreographies can be amended (by inferring label selections to add)
theorem	label selections can be encoded as value communications
theorem	CC is turing complete
theorem	removing or weakening other primitives from CC breaks Turing completeness

asynchrony

semantically states now include queues of incoming messages

- one queue for each pair of distinct processes
- two-step communication
- also applies to label selection

$\rightsquigarrow$ nicely matches asynchronous semantics at the process level

asynchrony

semantically states now include queues of incoming messages

- one queue for each pair of distinct processes
- two-step communication
- also applies to label selection

$\rightsquigarrow$ nicely matches asynchronous semantics at the process level

syntactically auxiliary processes store messages in transit

- requires ability to spawn processes
- requires name mobility, graph of connections
- allows for synchronous and asynchronous communication

theorem formal correspondence between both models

extraction

problem given a process implementation, can we extract a choreography that describes it?

extraction

problem given a process implementation, can we extract a choreography that describes it?

results algorithm for choreography extraction

- based on abstract execution graphs
- non-deterministic, but well-defined
- able to deal with the asynchronous case

extraction

problem given a process implementation, can we extract a choreography that describes it?

results algorithm for choreography extraction

- based on abstract execution graphs
- non-deterministic, but well-defined
- able to deal with the asynchronous case

$\rightsquigarrow$ to capture interesting asynchronous behaviours we extend CC:

$$\mathsf{p} \triangleright !\mathsf{q}.*; ?\mathsf{q} \mid \mathsf{q} \triangleright !\mathsf{p}.*; ?\mathsf{p}$$

is extracted to

$$\begin{pmatrix} \mathsf{p}.* \to \mathsf{q} \\ \mathsf{q}.* \to \mathsf{p} \end{pmatrix}$$

choreographies in practice

goal write algorithms used in real applications

- PC extends CC with top-level procedures, general sequential composition
- type system for data communications
- synchronous and asynchronous semantics
- limited higher-order features

choreographies in practice

goal write algorithms used in real applications

- PC extends CC with top-level procedures, general sequential composition
- type system for data communications
- synchronous and asynchronous semantics
- limited higher-order features

examples in PC we can write:

- parallel mergesort and quicksort
- gaussian elimination with pipelined communication
- parallel fast fourier transform

thank you!