connectors meet choreographies

luís cruz-filipe

(with farhad arbab, sung-shik jongmans and fabrizio montesi)

department of mathematics and computer science
university of southern denmark

talks@di
march 31st, 2017

a motivating example

two-buyer protocol

alice wants to buy a book from a seller

- alice sends the title to the seller
- the seller replies to alice and her bank with the price
- alice tells her bank how much she wants to pay
- the bank checks whether alice has enough funds
  - in the affirmative case, the bank confirms and the seller sends the book to alice
  - otherwise, the bank rejects the transaction

outline

three paradigms for concurrency

cho-reo-graphies

conclusions

process calculi

π-calculus the canonical model

- low-level model of communication
- local implementations of concurrent processes
- many interesting fragments are undecidable

process calculi

π-calculus the canonical model

- low-level model of communication
- local implementations of concurrent processes
- many interesting fragments are undecidable

communication

- based on channels and/or sessions
- synchronous/asynchronous
- one-to-one, one-to-many, many-to-one
- . . . but typically homogeneous

choreographic programming

origin theory follows practice

- inspired by common practice
- global, high-level views of systems
- formally similar to types for π-calculus

choreographic programming

origin theory follows practice

- inspired by common practice
- global, high-level views of systems
- formally similar to types for π-calculus

choreographies a different programming paradigm

- directed communication (from alice to bob)
- deadlock-freedom by design
- correct compilation to process calculi

choreographic programming

origin theory follows practice

- inspired by common practice
- global, high-level views of systems
- formally similar to types for π-calculus

choreographies a different programming paradigm

- directed communication (from alice to bob)
- deadlock-freedom by design
- correct compilation to process calculi

communication

- abstract formulation, implemented as before
- same variants, same considerations

exogenous coordination

focus separation of concerns

- focus on communication structures
- processes communicate only through ports
- message flow is defined by a communication medium

exogenous coordination

focus separation of concerns

- focus on communication structures
- processes communicate only through ports
- message flow is defined by a communication medium

communication media the focus of the paradigm:
how do messages flow between ports?

- a common language: Reo
- 30+ different semantics
- in this work: communication automata

our goal

$\rightsquigarrow$ combine choreographies with connectors

our goal

$\rightsquigarrow$ combine choreographies with connectors

- keep separation of concerns
- choreographies interact with connectors

our goal

$\rightsquigarrow$ combine choreographies with connectors

- keep separation of concerns
- choreographies interact with connectors
- . . . but they are independent
- notion of compatibility
- two-flavored semantics

outline

three paradigms for concurrency

cho-reo-graphies

conclusions

a minimalist model

- proof of principle
- turing completeness
- good decidability properties
- allows us to understand the typical problems

syntax

the calculus

$$
\begin{array}{rll}
C ::= & \mathbf{0} & \text{(termination)} \\
& \mid \widetilde{\eta}\ \mathsf{thru}\ \gamma;\ C & \text{(communication)} \\
& \mid \mathsf{if}\ (\mathsf{p}.e)\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 & \text{(choice)} \\
& \mid \mathsf{def}\ X = C_2\ \mathsf{in}\ C_1 & \text{(recursion)} \\
& \mid X & \text{(call)} \\
\\
\eta ::= & \mathsf{p}.e \to \mathsf{q} & \text{(value)} \\
& \mid \mathsf{p} \to \mathsf{q}[\ell] & \text{(label)}
\end{array}
$$

$\ell \in$ a non-singleton finite set

$e \in$ a suitable language

semantics (i/iii)

processes standard choreography semantics

- state assigns a value to each process
- transition semantics
- communications ruled by external parameter
- runtime terms for incomplete communications
- swapping of independent actions

semantics (i/iii)

processes standard choreography semantics

- state assigns a value to each process
- transition semantics
- communications ruled by external parameter
- runtime terms for incomplete communications
- swapping of independent actions

connectors constraint automata define possible communications

- one automaton per communication medium
- internal state partially replicated in runtime choreographies

semantics (ii/iii)

$\rightsquigarrow$ the communication rule

$$\frac{\emptyset \neq P \subseteq \mathsf{ports}(\tilde{\eta}) \quad \mathcal{A}(\gamma)_1 \xrightarrow{P,\phi}_{\gamma} s' \qquad \mathcal{A}(\gamma)_2 \xrightarrow{\sigma,\phi}_{\gamma} \mu' \qquad \phi *^{\mu} \tilde{\eta}}{\tilde{\eta}\,\mathsf{thru}\,\gamma; C, \sigma, \mathcal{A} \rightsquigarrow_{\mathcal{G}} P(\tilde{\eta})\,\mathsf{thru}\,\gamma; C, \phi(\sigma), \mathcal{A}[\gamma \mapsto \langle s', \mu' \rangle]}$$

- $\mathsf{ports}(\tilde{\eta})$ is the set of ports derived from $\tilde{\eta}$
- $\phi(\sigma)$ denotes the result of updating σ according to the interactions that were completed
- $P(\tilde{\eta})$ contains the unfinished/unexecuted communications in $\tilde{\eta}$
- $\phi *^{\mu} \tilde{\eta}$ expresses that ϕ and $\tilde{\eta}$ agree on the messages in transit

semantics (iii/iii)

an example

$\mathcal{G}(\gamma)$ allows p and r to send simultaneously to q and s, who then can receive the messages in any order

semantics (iii/iii)

an example

$\mathcal{G}(\gamma)$ allows p and r to send simultaneously to q and s, who then can receive the messages in any order

possible reduction path

η :	$\mathsf{p}.1 \to \mathsf{q}$ $\mathsf{r}.2 \to \mathsf{s}$			
σ :	$\mathsf{q} \mapsto 0$ $\mathsf{s} \mapsto 0$			
s :	s_0			
μ :	μ_0			

semantics (iii/iii)

an example

$\mathcal{G}(\gamma)$ allows p and r to send simultaneously to q and s, who then can receive the messages in any order

possible reduction path

η :	$\mathsf{p}.1 \to \mathsf{q}$ $\mathsf{r}.2 \to \mathsf{s}$	$1 \to \mathsf{q}$ $2 \to \mathsf{s}$		
σ :	$\mathsf{q} \mapsto 0$ $\mathsf{s} \mapsto 0$	$\mathsf{q} \mapsto 0$ $\mathsf{s} \mapsto 0$		
s :	s_0	s_1		
μ :	μ_0	$m_{\mathsf{pq}} \mapsto 1$ $m_{\mathsf{rs}} \mapsto 2$		

semantics (iii/iii)

an example

$\mathcal{G}(\gamma)$ allows p and r to send simultaneously to q and s, who then can receive the messages in any order

possible reduction path

η :	$\mathsf{p}.1 \rightarrow \mathsf{q}$ $\mathsf{r}.2 \rightarrow \mathsf{s}$	$1 \rightarrow \mathsf{q}$ $2 \rightarrow \mathsf{s}$	$1 \rightarrow \mathsf{q}$
σ :	$\mathsf{q} \mapsto 0$ $\mathsf{s} \mapsto 0$	$\mathsf{q} \mapsto 0$ $\mathsf{s} \mapsto 0$	$\mathsf{q} \mapsto 0$ $\mathsf{s} \mapsto 2$
s :	s_0	s_1	s_2
μ :	μ_0	$m_{\mathsf{pq}} \mapsto 1$ $m_{\mathsf{rs}} \mapsto 2$	$m_{\mathsf{pq}} \mapsto 1$ $m_{\mathsf{rs}} \mapsto 2$

semantics (iii/iii)

an example

$\mathcal{G}(\gamma)$ allows p and r to send simultaneously to q and s, who then can receive the messages in any order

possible reduction path

η :	$\mathsf{p}.1 \to \mathsf{q}$ $\mathsf{r}.2 \to \mathsf{s}$	$1 \to \mathsf{q}$ $2 \to \mathsf{s}$	$1 \to \mathsf{q}$	$\mathbf{0}$
σ :	$\mathsf{q} \mapsto 0$ $\mathsf{s} \mapsto 0$	$\mathsf{q} \mapsto 0$ $\mathsf{s} \mapsto 0$	$\mathsf{q} \mapsto 0$ $\mathsf{s} \mapsto 2$	$\mathsf{q} \mapsto 1$ $\mathsf{s} \mapsto 2$
s :	s_0	s_1	s_2	s_0
μ :	μ_0	$m_{\mathsf{pq}} \mapsto 1$ $m_{\mathsf{rs}} \mapsto 2$	$m_{\mathsf{pq}} \mapsto 1$ $m_{\mathsf{rs}} \mapsto 2$	$m_{\mathsf{pq}} \mapsto 1$ $m_{\mathsf{rs}} \mapsto 2$

the swap relation

we can permute independent communications:

- on different connectors and different processes

$$\frac{\mathsf{pn}(\tilde{\eta}) \cap \mathsf{pn}(\widetilde{\eta'}) = \emptyset \qquad \gamma \neq \gamma'}{\left(\tilde{\eta}\,\mathsf{thru}\,\gamma; \widetilde{\eta'}\,\mathsf{thru}\,\gamma'\right) \equiv \left(\widetilde{\eta'}\,\mathsf{thru}\,\gamma'; \tilde{\eta}\,\mathsf{thru}\,\gamma\right)} \ \lfloor \mathsf{C}|\mathsf{Eta\text{-}Eta} \rceil$$

the swap relation

we can permute independent communications:

- on different connectors and different processes

$$\frac{\mathsf{pn}(\tilde{\eta}) \cap \mathsf{pn}(\widetilde{\eta'}) = \emptyset \qquad \gamma \neq \gamma'}{\left(\tilde{\eta}\,\mathsf{thru}\,\gamma; \widetilde{\eta'}\,\mathsf{thru}\,\gamma'\right) \equiv \left(\widetilde{\eta'}\,\mathsf{thru}\,\gamma'; \tilde{\eta}\,\mathsf{thru}\,\gamma\right)} \quad \lfloor \mathsf{C|Eta\text{-}Eta} \rceil$$

- on the same connector

$$\frac{\mathsf{pn}(\widetilde{\eta_1}) \cap \mathsf{pn}(\widetilde{\eta_2}) = \emptyset}{(\widetilde{\eta_1}\,\mathsf{thru}\,\gamma; \widetilde{\eta_2}\,\mathsf{thru}\,\gamma) \equiv (\widetilde{\eta_1} \cup \widetilde{\eta_2})\ \mathsf{thru}\,\gamma} \quad \lfloor \mathsf{C|Eta\text{-}Split} \rceil$$

the swap relation

we can permute independent communications:

- on different connectors and different processes

$$\frac{\mathsf{pn}(\tilde{\eta}) \cap \mathsf{pn}(\widetilde{\eta'}) = \emptyset \qquad \gamma \neq \gamma'}{\left(\tilde{\eta}\,\mathsf{thru}\,\gamma; \widetilde{\eta'}\,\mathsf{thru}\,\gamma'\right) \equiv \left(\widetilde{\eta'}\,\mathsf{thru}\,\gamma'; \tilde{\eta}\,\mathsf{thru}\,\gamma\right)} \ \lfloor \mathsf{C|Eta\text{-}Eta} \rceil$$

- on the same connector

$$\frac{\mathsf{pn}(\widetilde{\eta_1}) \cap \mathsf{pn}(\widetilde{\eta_2}) = \emptyset}{(\widetilde{\eta_1}\,\mathsf{thru}\,\gamma; \widetilde{\eta_2}\,\mathsf{thru}\,\gamma) \equiv (\widetilde{\eta_1} \cup \widetilde{\eta_2})\ \mathsf{thru}\,\gamma} \ \lfloor \mathsf{C|Eta\text{-}Split} \rceil$$

- combining these, we obtain interleaving

deadlock freedom

possible problem communication rule can fail to be applicable

- with reasonable assumptions, detecting this is undecidable

deadlock freedom

possible problem communication rule can fail to be applicable

- with reasonable assumptions, detecting this is undecidable

solution more restrictive compatibility relation

- ignore swap
- symbolic execution
- always consider both branches in choices
- require recursive calls to be uniform

deadlock freedom

possible problem communication rule can fail to be applicable

- with reasonable assumptions, detecting this is undecidable

solution more restrictive compatibility relation

- ignore swap
- symbolic execution
- always consider both branches in choices
- require recursive calls to be uniform

$\rightsquigarrow$ decidable, reasonable assumptions in practice

incomplete

$$\mathsf{def}\, X = \mathsf{p}.e \to \mathsf{q}\,\mathsf{thru}\,\gamma; \mathsf{r}.e \to \mathsf{s}\,\mathsf{thru}\,\gamma; X\,\mathsf{in}\,X$$

where $\mathcal{G}(\gamma)$ allows (only) alternating one communication from p to q with two communications from r to s

projection

as usual in choreography languages, we can project our choreographies to process implementations

target language a variant of π-calculus

- primitives "input from port p" and "output to port p" (rather than e.g. channels)
- semantics uses a set of connectors over the actual ports
- similar rule for communication

projection

as usual in choreography languages, we can project our choreographies to process implementations

target language a variant of π-calculus

- primitives "input from port p" and "output to port p" (rather than e.g. channels)
- semantics uses a set of connectors over the actual ports
- similar rule for communication

projection built as standard in choreography calculi
actions are split in their local components

properties operational correspondence (up-to bisimulation) between choreographies and their projections
$\rightsquigarrow$ deadlock-freedom by construction

outline

three paradigms for concurrency

cho-reo-graphies

conclusions

conclusions & future work

results

- a unifying model integrating choreographic programming and exogenous coordination
- inherits the good properties of both paradigms

what's next?

- relaxing the requirements on the constraint automata to obtain non-determinism
- allow choreographies to underspecify communications to model open-ended systems
- similar combination with multiparty session types

thank you!