

the paths to choreography extraction

luís cruz-filipe

(joint work with kim s. larsen and fabrizio montesi)

department of mathematics and computer science
university of southern denmark

fossacs 2017
april 27th, 2017

outline

background

*choreography
extraction*

choreographic programming

context

choreographies

- high-level descriptions of communicating systems
- directed communication (from alice to bob)
- automatic compilation to process calculi
- good theoretical properties

choreographic programming

context

choreographies

- high-level descriptions of communicating systems
- directed communication (from alice to bob)
- automatic compilation to process calculi
- good theoretical properties

previously

core choreographies

- minimal primitives for turing completeness
- captures the “essence” of choreographies

choreographic programming

context

choreographies

- high-level descriptions of communicating systems
- directed communication (from alice to bob)
- automatic compilation to process calculi
- good theoretical properties

previously

core choreographies

- minimal primitives for turing completeness
- captures the “essence” of choreographies

in this work

inverting compilation

- extraction from implementations

core choreographies (i/ii)

syntax

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

$\eta ::= p.e \rightarrow q \mid p \rightarrow q[l]$

$l ::= \text{labels (at least two distinct)}$

$e ::= \text{some set of expressions}$

core choreographies (ii/ii)

semantics

$$\frac{v = e[\sigma(p)/*]}{p.e \rightarrow q; C, \sigma \longrightarrow C, \sigma[q \mapsto v]}$$
$$\frac{}{p \rightarrow q[l]; C, \sigma \longrightarrow C, \sigma}$$
$$\frac{i = 1 \text{ if } \sigma(p) = \sigma(q), \ i = 2 \text{ else}}{\text{if } (p.* = q.*) \text{ then } C_1 \text{ else } C_2, \sigma \longrightarrow C_i, \sigma}$$
$$\frac{C_1, \sigma \longrightarrow C'_1, \sigma'}{\text{def } X = C_2 \text{ in } C_1, \sigma \longrightarrow \text{def } X = C_2 \text{ in } C'_1, \sigma'}$$
$$\frac{C_1 \preceq C'_1 \quad C'_1, \sigma \longrightarrow C'_2, \sigma' \quad C'_2 \preceq C_2}{C_1, \sigma \longrightarrow C_2, \sigma'}$$

(last rule says that

e.g. $p.e \rightarrow q; r.e' \rightarrow s, \sigma \longrightarrow p.e \rightarrow q, \sigma'$)

stateful processes

target language

a process calculus with the corresponding primitives:

- send to/receive from a process
- offer a choice to/select an option from a process
- conditional
- recursive definition

epp

the endpoint projection of a choreography is a process term that implements the corresponding choreography

example

the choreography

$$p.e \rightarrow q; p.e' \rightarrow r$$

projects to

$$p \triangleright q!e; r!e' \mid q \triangleright p? \mid r \triangleright p?$$

outline

background

*choreography
extraction*

the problem

questions

given a process network N :

- is there a choreography C with the same behaviour (bisimilarity)?
- in the affirmative case, can we construct C from N ?

the problem

questions

given a process network N :

- is there a choreography C with the same behaviour (bisimilarity)?
- in the affirmative case, can we construct C from N ?

answer

no

- undecidability results prevent perfect solution
- ... but can we solve this for a large enough set of N ?

the problem

questions

given a process network N :

- is there a choreography C with the same behaviour (bisimilarity)?
- in the affirmative case, can we construct C from N ?

answer

no

- undecidability results prevent perfect solution
- ... but can we solve this for a large enough set of N ?

new goal

given a process network N :

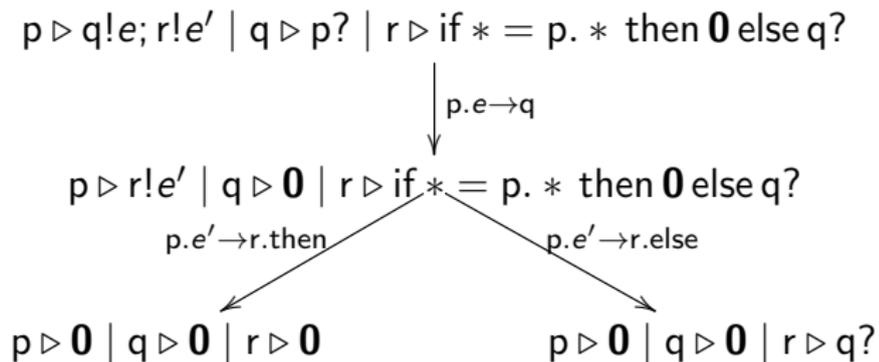
- if we return yes, we can build C bisimilar to N
- we return yes as much as possible

our approach

idea symbolic execution of N
(abstracting from values, two cases in conditionals)
each “path” corresponds to a choreography

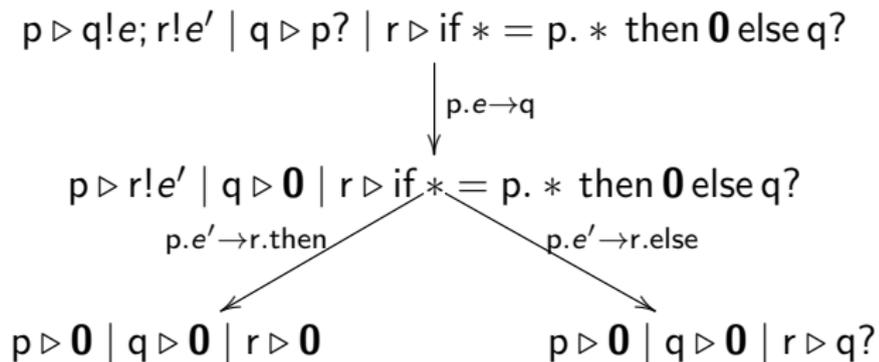
our approach

idea symbolic execution of N
(abstracting from values, two cases in conditionals)
each “path” corresponds to a choreography



our approach

idea symbolic execution of N
(abstracting from values, two cases in conditionals)
each “path” corresponds to a choreography



*extracted
choreography*

$p.e \rightarrow q; \text{if } r.* = p. * \text{ then } \mathbf{0} \text{ else } \mathbf{1}$
where $\mathbf{1}$ stands for deadlock (equivalent to $\mathbf{0}$)

properties (finite case)

- always terminates
- identifies potential problems by **1**
- bisimilarity *always* holds!
- non-deterministic (up to structural equivalence)
- is sound and (almost) complete
(deadlocks may occur in dead code)

introducing recursion

the problem

consider the following networks

- $p \triangleright \text{def } X = q!*; X \text{ in } X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
- $p \triangleright \text{def } X = q!*; X \text{ in } q!*; X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
- $p \triangleright \text{def } X = q!*; q!*; X \text{ in } X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
- $p \triangleright \text{def } X = q!*; q!*; X \text{ in } q!*; X$
| $q \triangleright \text{def } Y = p?; p?; Y \text{ in } Y$

introducing recursion

the problem

consider the following networks

- $p \triangleright \text{def } X = q!*; X \text{ in } X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
- $p \triangleright \text{def } X = q!*; X \text{ in } q!*; X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
- $p \triangleright \text{def } X = q!*; q!*; X \text{ in } X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
- $p \triangleright \text{def } X = q!*; q!*; X \text{ in } q!*; X$
| $q \triangleright \text{def } Y = p?; p?; Y \text{ in } Y$

main intuition

we do not care what the recursive definitions at the processes say!

the idea

do the same as before, but allow loops

introducing recursion

the problem

consider the following networks

- $p \triangleright \text{def } X = q!*; X \text{ in } X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y \quad \rightsquigarrow \text{def } X = p.e \rightarrow q; X \text{ in } X$
- $p \triangleright \text{def } X = q!*; X \text{ in } q!*; X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
- $p \triangleright \text{def } X = q!*; q!*; X \text{ in } X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
- $p \triangleright \text{def } X = q!*; q!*; X \text{ in } q!*; X$
| $q \triangleright \text{def } Y = p?; p?; Y \text{ in } Y$

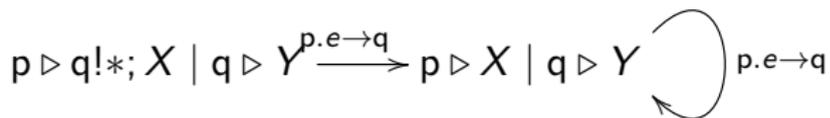


introducing recursion

the problem

consider the following networks

- $p \triangleright \text{def } X = q!^*; X \text{ in } X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y \quad \rightsquigarrow \text{def } X = p.e \rightarrow q; X \text{ in } X$
- $p \triangleright \text{def } X = q!^*; X \text{ in } q!^*; X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
 $\rightsquigarrow \text{def } X = p.e \rightarrow q; X \text{ in } p.e \rightarrow q; X$
- $p \triangleright \text{def } X = q!^*; q!^*; X \text{ in } X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
- $p \triangleright \text{def } X = q!^*; q!^*; X \text{ in } q!^*; X$
| $q \triangleright \text{def } Y = p?; p?; Y \text{ in } Y$

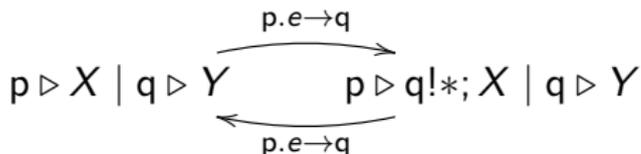


introducing recursion

the problem

consider the following networks

- $p \triangleright \text{def } X = q!*; X \text{ in } X$
 $| q \triangleright \text{def } Y = p?; Y \text{ in } Y \quad \rightsquigarrow \text{def } X = p.e \rightarrow q; X \text{ in } X$
- $p \triangleright \text{def } X = q!*; X \text{ in } q!*; X$
 $| q \triangleright \text{def } Y = p?; Y \text{ in } Y$
 $\rightsquigarrow \text{def } X = p.e \rightarrow q; X \text{ in } p.e \rightarrow q; X$
- $p \triangleright \text{def } X = q!*; q!*; X \text{ in } X$
 $| q \triangleright \text{def } Y = p?; Y \text{ in } Y$
 $\rightsquigarrow \text{def } X = p.e \rightarrow q; p.e \rightarrow q; X \text{ in } X$
- $p \triangleright \text{def } X = q!*; q!*; X \text{ in } q!*; X$
 $| q \triangleright \text{def } Y = p?; p?; Y \text{ in } Y$

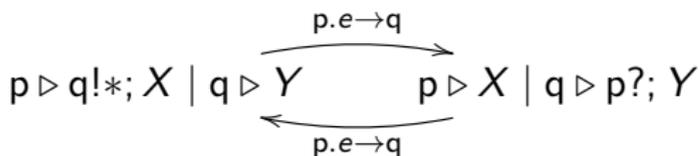


introducing recursion

the problem

consider the following networks

- $p \triangleright \text{def } X = q!*; X \text{ in } X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y \quad \rightsquigarrow \text{def } X = p.e \rightarrow q; X \text{ in } X$
- $p \triangleright \text{def } X = q!*; X \text{ in } q!*; X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
 $\rightsquigarrow \text{def } X = p.e \rightarrow q; X \text{ in } p.e \rightarrow q; X$
- $p \triangleright \text{def } X = q!*; q!*; X \text{ in } X$
| $q \triangleright \text{def } Y = p?; Y \text{ in } Y$
 $\rightsquigarrow \text{def } X = p.e \rightarrow q; p.e \rightarrow q; X \text{ in } X$
- $p \triangleright \text{def } X = q!*; q!*; X \text{ in } q!*; X$
| $q \triangleright \text{def } Y = p?; p?; Y \text{ in } Y$
 $\rightsquigarrow \text{def } X = p.e \rightarrow q; p.e \rightarrow q; X \text{ in } X$



fairness and starvation

problems not all loops are equal...

$$p \triangleright \text{def } X = q!*; X \text{ in } X \mid q \triangleright \text{def } Y = p?; Y \text{ in } Y \\ \mid r \triangleright \text{def } Z = s!*; Z \text{ in } Z \mid s \triangleright \text{def } W = r?; W \text{ in } W$$

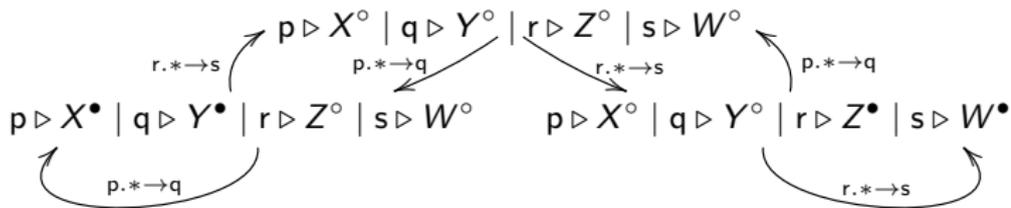

extracts to $\text{def } X = p.* \rightarrow q; X \text{ in } X$ or $\text{def } X = r.* \rightarrow s; X \text{ in } X$

fairness and starvation

problems not all loops are equal...

$$p \triangleright \text{def } X = q!*; X \text{ in } X \mid q \triangleright \text{def } Y = p?; Y \text{ in } Y \\ \mid r \triangleright \text{def } Z = s!*; Z \text{ in } Z \mid s \triangleright \text{def } W = r?; W \text{ in } W$$

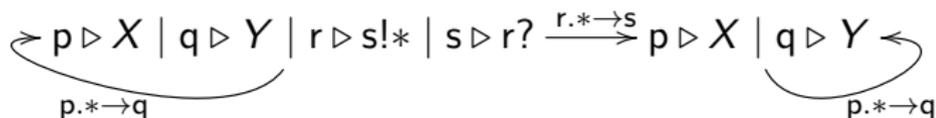
solution annotate procedure calls



fairness and starvation

problems

not all loops are equal...

$$p \triangleright \text{def } X = q!*; X \text{ in } X \mid q \triangleright \text{def } Y = p?; Y \text{ in } Y \\ \mid r \triangleright s!* \mid s \triangleright r?$$


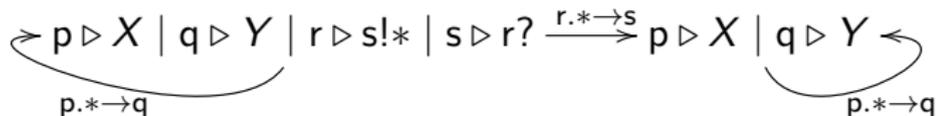
extracts to

$$\text{def } X = p.* \rightarrow q; X \text{ in } X \text{ or} \\ \text{def } X = p.* \rightarrow q; X \text{ in } r.* \rightarrow s; X$$

fairness and starvation

problems

not all loops are equal...

$$p \triangleright \text{def } X = q!*; X \text{ in } X \mid q \triangleright \text{def } Y = p?; Y \text{ in } Y \\ \mid r \triangleright s!* \mid s \triangleright r?$$


extracts to

$$\text{def } X = p.* \rightarrow q; X \text{ in } X \text{ or} \\ \text{def } X = p.* \rightarrow q; X \text{ in } r.* \rightarrow s; X$$

solution

no finite behaviour in loops (except deadlocks)

fairness and starvation

problems not all loops are equal. . .

$$\begin{aligned} & p \triangleright \text{def } X = q!*; X \text{ in } X \mid q \triangleright \text{def } Y = p?; Y \text{ in } Y \\ & \mid r \triangleright \text{def } Z = q!*; Z \text{ in } Z \end{aligned}$$
$$p \triangleright X^\circ \mid q \triangleright Y^\circ \mid r \triangleright Z^\circ \xrightarrow{p.* \rightarrow q} p \triangleright X^\bullet \mid q \triangleright Y^\bullet \mid r \triangleright Z^\circ \xleftarrow{p.* \rightarrow q}$$

oops not extractable (but r is deadlocked)

fairness and starvation

problems

not all loops are equal. . .

$$\begin{aligned} & p \triangleright \text{def } X = q!*; X \text{ in } X \mid q \triangleright \text{def } Y = p?; Y \text{ in } Y \\ & \mid r \triangleright \text{def } Z = q!*; Z \text{ in } Z \end{aligned}$$
$$p \triangleright X^\circ \mid q \triangleright Y^\circ \mid r \triangleright Z^\circ \xrightarrow{p.* \rightarrow q} p \triangleright X^\bullet \mid q \triangleright Y^\bullet \mid r \triangleright Z^\circ \xrightarrow{p.* \rightarrow q}$$

oops

not extractable (but r is deadlocked)

in general

some networks are not extractable

results

- if symbolic execution does not generate a node from which some process is always deadlocked, then N is extractable
- if C is extracted from N , then C and N are bisimilar (C may contain deadlocks)
- extraction terminates in time $O(n \times e^{2n/e})$
- works for synchronous semantics
- can be adapted to/extended in the asynchronous case

conclusions & future directions

- showed how to extract choreographies from implementations
- significant improvement in complexity wrt previous work
- prototype implementation nearly ready
- extension to process spawning in progress

thank you!