foundational questions in choreographic programming

luís cruz-filipe

(joint work with fabrizio montesi)

department of mathematics and computer science
university of southern denmark

logic and computation seminar, ist
november 25th, 2016

outline

background

asychrony, semantically

choreography extraction

core choreographies

previously

- minimal primitives for turing completeness
- captures the “essence” of choreographies
- framework to study foundational questions

core choreographies

previously

- minimal primitives for turing completeness
- captures the "essence" of choreographies
- framework to study foundational questions

in this work study some foundational questions

- asynchronous communication
- extraction from implementations

core choreographies (i/ii)

choreographies

- global view of the system
- directed communication (from alice to bob)
- deadlock-free by design
- compilable to process calculi

core choreographies (i/ii)

choreographies

- global view of the system
- directed communication (from alice to bob)
- deadlock-free by design
- compilable to process calculi

syntax

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

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

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

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

core choreographies (ii/ii)

semantics

$$\frac{v = e[\sigma(\mathsf{p})/*]}{\mathsf{p}.e \to \mathsf{q}; C, \sigma \to C, \sigma[\mathsf{q} \mapsto v]}$$

$$\frac{}{\mathsf{p} \to \mathsf{q}[l]; C, \sigma \to C, \sigma}$$

$$\frac{i = 1 \text{ if } \sigma(\mathsf{p}) = \sigma(\mathsf{q}),\ \ i = 2 \text{ else}}{\text{if}\,(\mathsf{p}.* = \mathsf{q}.*)\,\text{then}\ C_1\ \text{else}\ C_2, \sigma \to C_i, \sigma}$$

$$\frac{C_1, \sigma \to C_1', \sigma'}{\text{def}\ X = C_2\ \text{in}\ C_1, \sigma \to \text{def}\ X = C_2\ \text{in}\ C_1', \sigma'}$$

$$\frac{C_1 \preceq C_1' \qquad C_1', \sigma \to C_2', \sigma' \qquad C_2' \preceq C_2}{C_1, \sigma \to C_2, \sigma'}$$

(last rule says that
e.g. $\mathsf{p}.e \to \mathsf{q}; \mathsf{r}.e' \to \mathsf{s}, \sigma \to \mathsf{p}.e \to \mathsf{q}, \sigma'$)

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

$$\mathsf{p}.e \to \mathsf{q}; \mathsf{p}.e' \to \mathsf{r}$$

projects to

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

outline

the problem

goal represent asynchronous communication in choreographies

$\rightsquigarrow$ at the process level, this is easy:

- no synchronization on communications
- processes have queues of incoming messages

the solution

syntax

extend choreographies with runtime terms:

$$\mathsf{p}.e \rightarrow^{x} \bullet_{\mathsf{q}} \qquad \bullet_{\mathsf{p}} \rightarrow^{x} \mathsf{q} \qquad \bullet_{\mathsf{p}} \rightarrow^{v} \mathsf{q}$$

(and likewise for selections)

- variables are used exactly twice (in matching pairs)
- they store track messages in transit
- $\bullet_{\mathsf{p}} \rightarrow^{x} \mathsf{q}$ denotes a message that has not been sent yet
- $\bullet_{\mathsf{p}} \rightarrow^{v} \mathsf{q}$ denotes a message sent by p but not received by q

semantics

formally we replace rules for communication with the following ones:

$$\frac{}{\mathsf{p}.e \rightarrow \mathsf{q} \preceq \mathsf{p}.e \rightarrow^{x} \bullet_{\mathsf{q}} ; \bullet_{\mathsf{p}} \rightarrow^{x} \mathsf{q}}$$

$$\frac{v = e[\sigma(\mathsf{p})/*]}{\mathsf{p}.e \rightarrow^{x} \bullet_{\mathsf{q}} ; C, \sigma \rightarrow C[v/x], \sigma}$$

$$\frac{}{\bullet_{\mathsf{p}} \rightarrow^{v} \mathsf{q}; C, \sigma \rightarrow C, \sigma[\mathsf{q} \mapsto v]}$$

an example

$$
\begin{aligned}
&\mathsf{p}.e \to \mathsf{q}; \mathsf{p}.e' \to \mathsf{r} \\
&\qquad \preceq \mathsf{p}.e \to^{x} \bullet_{\mathsf{q}}; \bullet_{\mathsf{p}} \to^{x} \mathsf{q}; \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r}
\end{aligned}
$$

an example

$$\begin{aligned}
&\mathsf{p}.e \rightarrow \mathsf{q}; \mathsf{p}.e' \rightarrow \mathsf{r} \\
&\qquad \preceq \mathsf{p}.e \rightarrow^{x} \bullet_{\mathsf{q}}; \bullet_{\mathsf{p}} \rightarrow^{x} \mathsf{q}; \mathsf{p}.e' \rightarrow^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \rightarrow^{y} \mathsf{r} \\
&\qquad \rightarrow \bullet_{\mathsf{p}} \rightarrow^{v} \mathsf{q}; \mathsf{p}.e' \rightarrow^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \rightarrow^{y} \mathsf{r}
\end{aligned}$$

an example

$$\begin{aligned}
&\mathsf{p}.e \to \mathsf{q}; \mathsf{p}.e' \to \mathsf{r} \\
&\qquad \preceq \mathsf{p}.e \to^{x} \bullet_{\mathsf{q}}; \bullet_{\mathsf{p}} \to^{x} \mathsf{q}; \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r} \\
&\qquad \to \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r} \\
&\qquad \preceq \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r}
\end{aligned}$$

an example

$$
\begin{aligned}
&\mathsf{p}.e \to \mathsf{q}; \mathsf{p}.e' \to \mathsf{r}\\
&\qquad \preceq \mathsf{p}.e \to^{x} \bullet_{\mathsf{q}}; \bullet_{\mathsf{p}} \to^{x} \mathsf{q}; \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r}\\
&\qquad \to \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r}\\
&\qquad \preceq \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r}\\
&\qquad \to \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \bullet_{\mathsf{p}} \to^{v'} \mathsf{r}
\end{aligned}
$$

an example

$$
\begin{aligned}
\mathsf{p}.e \to \mathsf{q}; \mathsf{p}.e' \to \mathsf{r} \\
&\preceq \mathsf{p}.e \to^{x} \bullet_{\mathsf{q}}; \bullet_{\mathsf{p}} \to^{x} \mathsf{q}; \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r} \\
&\to \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r} \\
&\preceq \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r} \\
&\to \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \bullet_{\mathsf{p}} \to^{v'} \mathsf{r} \\
&\preceq \bullet_{\mathsf{p}} \to^{v'} \mathsf{r}; \bullet_{\mathsf{p}} \to^{v} \mathsf{q}
\end{aligned}
$$

an example

$$
\begin{aligned}
&\mathsf{p}.e \to \mathsf{q}; \mathsf{p}.e' \to \mathsf{r} \\
&\qquad \preceq \mathsf{p}.e \to^{x} \bullet_{\mathsf{q}}; \bullet_{\mathsf{p}} \to^{x} \mathsf{q}; \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r} \\
&\qquad \to \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r} \\
&\qquad \preceq \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r} \\
&\qquad \to \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \bullet_{\mathsf{p}} \to^{v'} \mathsf{r} \\
&\qquad \preceq \bullet_{\mathsf{p}} \to^{v'} \mathsf{r}; \bullet_{\mathsf{p}} \to^{v} \mathsf{q} \\
&\qquad \to \bullet_{\mathsf{p}} \to^{v} \mathsf{q}
\end{aligned}
$$

an example

$$
\begin{aligned}
&\mathsf{p}.e \to \mathsf{q}; \mathsf{p}.e' \to \mathsf{r} \\
&\qquad \preceq \mathsf{p}.e \to^{x} \bullet_{\mathsf{q}}; \bullet_{\mathsf{p}} \to^{x} \mathsf{q}; \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r} \\
&\qquad \to \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r} \\
&\qquad \preceq \mathsf{p}.e' \to^{y} \bullet_{\mathsf{r}}; \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \bullet_{\mathsf{p}} \to^{y} \mathsf{r} \\
&\qquad \to \bullet_{\mathsf{p}} \to^{v} \mathsf{q}; \bullet_{\mathsf{p}} \to^{v'} \mathsf{r} \\
&\qquad \preceq \bullet_{\mathsf{p}} \to^{v'} \mathsf{r}; \bullet_{\mathsf{p}} \to^{v} \mathsf{q} \\
&\qquad \to \bullet_{\mathsf{p}} \to^{v} \mathsf{q}
\end{aligned}
$$

projection we can still project to process calculus, but bisimulation only holds for well-formed choreographies (runtime terms are at the head)

outline

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 $\mathtt{yes}$, we can build C bisimilar to N
- we return $\mathtt{yes}$ 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

$$
\begin{tikzcd}[column sep=tiny]
& \mathsf{p} \triangleright \mathsf{q}!e; \mathsf{r}!e' \mid \mathsf{q} \triangleright \mathsf{p}? \mid \mathsf{r} \triangleright \mathsf{if}\ * = \mathsf{p}.*\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ \mathsf{q}? \arrow[d, "\mathsf{p}.e \to \mathsf{q}"] & \\
& \mathsf{p} \triangleright \mathsf{r}!e' \mid \mathsf{q} \triangleright \mathbf{0} \mid \mathsf{r} \triangleright \mathsf{if}\ * = \mathsf{p}.*\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ \mathsf{q}? \arrow[dl, "\mathsf{p}.e' \to \mathsf{r}.\mathsf{then}"'] \arrow[dr, "\mathsf{p}.e' \to \mathsf{r}.\mathsf{else}"] & \\
\mathsf{p} \triangleright \mathbf{0} \mid \mathsf{q} \triangleright \mathbf{0} \mid \mathsf{r} \triangleright \mathbf{0} & & \mathsf{p} \triangleright \mathbf{0} \mid \mathsf{q} \triangleright \mathbf{0} \mid \mathsf{r} \triangleright \mathsf{q}?
\end{tikzcd}
$$

our approach

idea

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

$$
\begin{tikzcd}[column sep=tiny]
& \mathsf{p} \triangleright \mathsf{q}!e; \mathsf{r}!e' \mid \mathsf{q} \triangleright \mathsf{p}? \mid \mathsf{r} \triangleright \mathsf{if}\ * = \mathsf{p}.* \ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ \mathsf{q}? \arrow[d, "\mathsf{p}.e \to \mathsf{q}"] & \\
& \mathsf{p} \triangleright \mathsf{r}!e' \mid \mathsf{q} \triangleright \mathbf{0} \mid \mathsf{r} \triangleright \mathsf{if}\ * = \mathsf{p}.* \ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ \mathsf{q}? \arrow[dl, "\mathsf{p}.e' \to \mathsf{r}.\mathsf{then}"'] \arrow[dr, "\mathsf{p}.e' \to \mathsf{r}.\mathsf{else}"] & \\
\mathsf{p} \triangleright \mathbf{0} \mid \mathsf{q} \triangleright \mathbf{0} \mid \mathsf{r} \triangleright \mathbf{0} & & \mathsf{p} \triangleright \mathbf{0} \mid \mathsf{q} \triangleright \mathbf{0} \mid \mathsf{r} \triangleright \mathsf{q}?
\end{tikzcd}
$$

extracted choreography

$\mathsf{p}.e \to \mathsf{q}; \mathsf{if}\ \mathsf{r}.* = \mathsf{p}.* \ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ \mathbf{1}$
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

$$\begin{array}{l} \;\; \mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; X\ \mathsf{in}\ X \\ \mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y \end{array}$$

introducing recursion

the problem

consider the following networks

- $$\begin{aligned} &\mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; X \ \mathsf{in}\ X \\ \mid\ &\mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y \ \mathsf{in}\ Y \end{aligned}$$
- $$\begin{aligned} &\mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; X \ \mathsf{in}\ \mathsf{q}!e; X \\ \mid\ &\mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y \ \mathsf{in}\ Y \end{aligned}$$

introducing recursion

the problem

consider the following networks

- $\quad \mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; X\ \mathsf{in}\ X$
  $\mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y$
- $\quad \mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; X\ \mathsf{in}\ \mathsf{q}!e; X$
  $\mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y$
- $\quad \mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; \mathsf{q}!e; X\ \mathsf{in}\ X$
  $\mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y$

introducing recursion

the problem

consider the following networks

- $$\begin{aligned} &\mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; X\ \mathsf{in}\ X \\ \mid\ &\mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y \end{aligned}$$
- $$\begin{aligned} &\mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; X\ \mathsf{in}\ \mathsf{q}!e; X \\ \mid\ &\mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y \end{aligned}$$
- $$\begin{aligned} &\mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; \mathsf{q}!e; X\ \mathsf{in}\ X \\ \mid\ &\mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y \end{aligned}$$
- $$\begin{aligned} &\mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; \mathsf{q}!e; X\ \mathsf{in}\ \mathsf{q}!e; X \\ \mid\ &\mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; \mathsf{p}?; Y\ \mathsf{in}\ Y \end{aligned}$$

introducing recursion

the problem consider the following networks

- $\begin{array}{l} \ \ \mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; X\ \mathsf{in}\ X \\ \mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y \end{array}$
- $\begin{array}{l} \ \ \mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; X\ \mathsf{in}\ \mathsf{q}!e; X \\ \mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y \end{array}$
- $\begin{array}{l} \ \ \mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; \mathsf{q}!e; X\ \mathsf{in}\ X \\ \mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y \end{array}$
- $\begin{array}{l} \ \ \mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; \mathsf{q}!e; X\ \mathsf{in}\ \mathsf{q}!e; X \\ \mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; \mathsf{p}?; Y\ \mathsf{in}\ Y \end{array}$

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

- $\mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; X\ \mathsf{in}\ X \mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y \quad \rightsquigarrow \mathsf{def}\ X = \mathsf{p}.e \to \mathsf{q}\ \mathsf{in}\ X$
- $\mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; X\ \mathsf{in}\ \mathsf{q}!e; X \mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y \quad \rightsquigarrow \mathsf{def}\ X = \mathsf{p}.e \to \mathsf{q}\ \mathsf{in}\ \mathsf{p}.e \to \mathsf{q}; X$
- $\mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; \mathsf{q}!e; X\ \mathsf{in}\ X \mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y\ \mathsf{in}\ Y \quad \rightsquigarrow \mathsf{def}\ X = \mathsf{p}.e \to \mathsf{q}; \mathsf{p}.e \to \mathsf{q}\ \mathsf{in}\ X$
- $\mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!e; \mathsf{q}!e; X\ \mathsf{in}\ \mathsf{q}!e; X \mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; \mathsf{p}?; Y\ \mathsf{in}\ Y \rightsquigarrow \mathsf{def}\ X = \mathsf{p}.e \to \mathsf{q}; \mathsf{p}.e \to \mathsf{q}\ \mathsf{in}\ X$

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

the idea do the same as before, but allow loops

fairness and starvation

problems

not all loops are equal...

$$\begin{aligned} & \mathsf{p} \triangleright \mathsf{def}\ X = \mathsf{q}!{*}; X \ \mathsf{in}\ X \mid \mathsf{q} \triangleright \mathsf{def}\ Y = \mathsf{p}?; Y \ \mathsf{in}\ Y \\ \mid\ & \mathsf{r} \triangleright \mathsf{def}\ Z = \mathsf{s}!{*}; Z \ \mathsf{in}\ Z \mid \mathsf{s} \triangleright \mathsf{def}\ W = \mathsf{r}?; W \ \mathsf{in}\ W \end{aligned}$$

fairness and starvation

problems not all loops are equal...

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

solution annotate procedure calls

fairness and starvation

problems not all loops are equal...

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

solution annotate procedure calls

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

fairness and starvation

problems not all loops are equal...

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

solution annotate procedure calls

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

solution no finite behaviour in loops (except deadlocks)

fairness and starvation

problems not all loops are equal...

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

solution annotate procedure calls

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

solution no finite behaviour in loops (except deadlocks)

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

fairness and starvation

problems not all loops are equal...

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

solution annotate procedure calls

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

solution no finite behaviour in loops (except deadlocks)

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

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 $O\left(n \times e^{2n/e}\right)$
- works for synchronous and asynchronous semantics
- can be extended in the asynchronous case

conclusions

- showed how to model asynchronous communication in choreographies
- construction holds in “every” model
- showed how to extract choreographies from implementations
- complexity is lower bound for “all” languages

thank you!