Choreography Automata

Franco Barbanera[1], Ivan Lanese[2], Emilio Tuosto[3]

[1] University of Catania
[2] University of Bologna/INRIA
[3] GSSI/University of Leicester

February 16, 2021

The general idea

If you have a bunch of dancers...

The general idea

If you have a bunch of dancers...

....would you like to end up with this....

or with THIS?

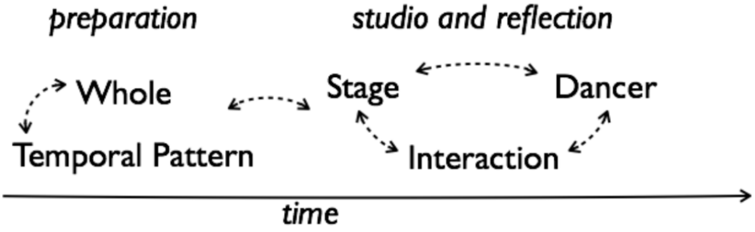
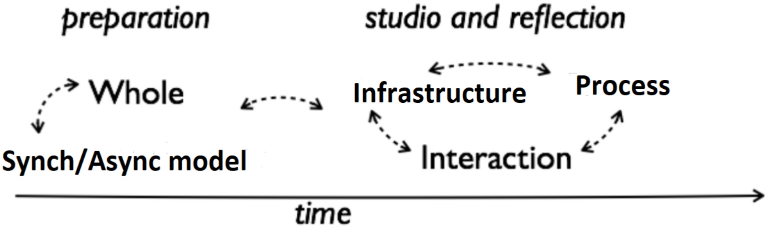


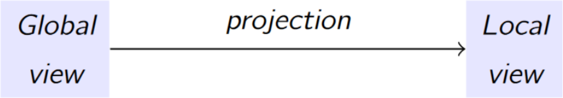
Figure 10. Focal points along the creative phases.

Same approach, different context



Choreographic development of distributed (message-passing) systems:

- EXPLOITS GLOBAL & LOCAL SPECIFICATIONS
  coexistence of two distinct but related views of a system: the *global* and the *local* views.

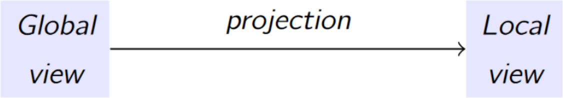


- SUPPORTS CORRECTNESS-BY-CONSTRUCTION
  "*projection*" : an operation producing the local view from the global one

Choreographic development of distributed (message-passing) systems:

- EXPLOITS GLOBAL & LOCAL SPECIFICATIONS

  coexistence of two distinct but related views of a system: the *global* and the *local* views.



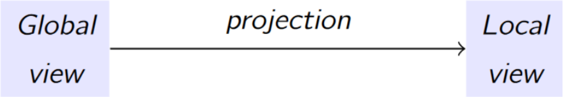
- SUPPORTS CORRECTNESS-BY-CONSTRUCTION

  "*projection*" : an operation producing the local view from the global one

Choreographic development of distributed (message-passing) systems:

- EXPLOITS GLOBAL & LOCAL SPECIFICATIONS

  coexistence of two distinct but related views of a system: the *global* and the *local* views.

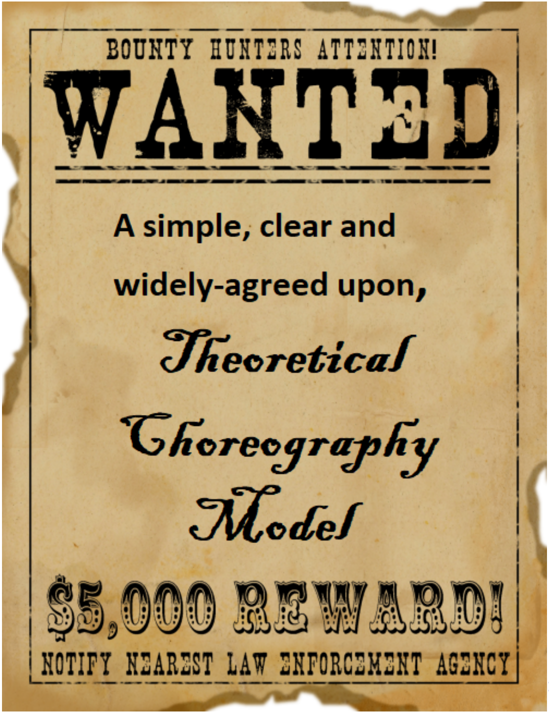


- SUPPORTS CORRECTNESS-BY-CONSTRUCTION

  "*projection*" : an operation producing the local view from the global one

The choreographic approach: A lighthouse on the Formal Verification ocean

- specification languages: WS-CDL, BPMN, ...
- choreographies for microservices;
- experimental choreographic languages: Chor, AIOCJ, ...
- etc.

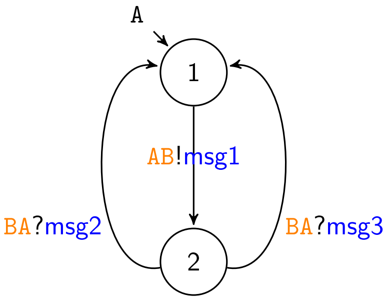

BOUNTY HUNTERS ATTENTION!
WANTED
A simple, clear and
widely-agreed upon,
Theoretical
Choreography
Model
$5,000 REWARD!
NOTIFY NEAREST LAW ENFORCEMENT AGENCY

Which setting?

- NON channel-based
- "Basically" actor-based.

Which abstraction for processes?

Which abstraction for processes?



Communicating Finite State Machines (CFSMs)

An **automata-based** formalism for the description and the analysis of distributed systems.

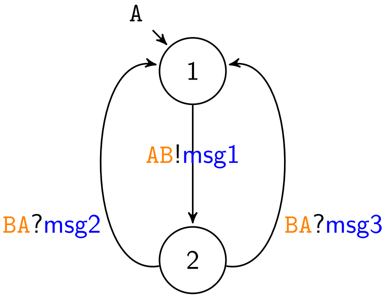
A machine $M_{\mathtt{A}}$

- $M_{\mathtt{A}}$ can send msg1 to machine $M_{\mathtt{B}}$;
  asynchronously; *through the directed buffered FIFO channel* AB
- Then, either msg2 or msg3 can be received from $M_{\mathtt{B}}$;
  through channel BA;
- and so on....

Communicating Finite State Machines (CFSMs)

An **automata-based** formalism for the description and the analysis of distributed systems.

A machine $M_{\mathtt{A}}$

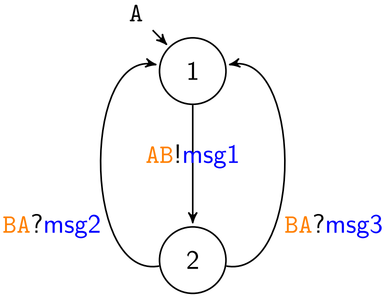


- $M_{\mathtt{A}}$ can send msg1 to machine $M_{\mathtt{B}}$;
  asynchronously; *through the directed buffered FIFO channel* AB
- Then, either msg2 or msg3 can be received from $M_{\mathtt{B}}$;
  through channel BA;
- and so on....

Communicating Finite State Machines (CFSMs)

An **automata-based** formalism for the description and the analysis of distributed systems.

A machine $M_{\mathtt{A}}$

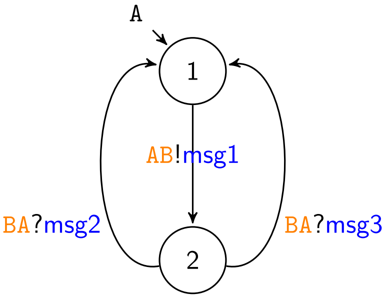


- $M_{\mathtt{A}}$ can send msg1 to machine $M_{\mathtt{B}}$;
  asynchronously; *through the directed buffered FIFO channel* AB
- Then, either msg2 or msg3 can be received from $M_{\mathtt{B}}$;
  through channel BA;
- and so on....

Communicating Finite State Machines (CFSMs)

An **automata-based** formalism for the description and the analysis of distributed systems.

A machine $M_{\mathtt{A}}$

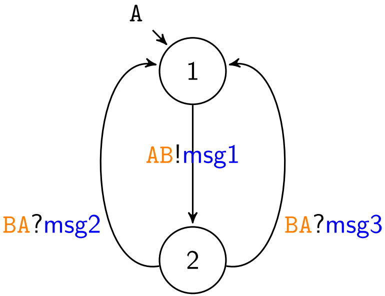


- $M_{\mathtt{A}}$ can send msg1 to machine $M_{\mathtt{B}}$;
  asynchronously; *through the directed buffered FIFO channel* AB
- Then, either msg2 or msg3 can be received from $M_{\mathtt{B}}$;
  through channel BA;
- and so on....

Communicating Finite State Machines (CFSMs)

An **automata-based** formalism for the description and the analysis of distributed systems.

A machine $M_{\mathtt{A}}$



- $M_{\mathtt{A}}$ can send msg1 to machine $M_{\mathtt{B}}$;
  asynchronously; *through the directed buffered FIFO channel* AB
- Then, either msg2 or msg3 can be received from $M_{\mathtt{B}}$;
  through channel BA;
- and so on....

Systems of CFSMs

A *system* of CFSMs:

$$S = (M_{\mathrm{p}})_{\mathrm{p} \in \mathrm{P}}$$

- P is the set of *roles* (participants) of S, and
- for each $\mathrm{p} \in \mathrm{P}$, $M_{\mathrm{p}} = (Q_{\mathrm{p}}, q_{0\mathrm{p}}, \mathbb{A}, \delta_{\mathrm{p}})$ is a CFSM.

A *configuration* of S:

$$s = (\vec{q}, \vec{w})$$

- $\vec{q} = (q_{\mathrm{p}})_{\mathrm{p} \in \mathrm{P}}$ *the overall state of the system*
  where $q_{\mathrm{p}} \in Q_{\mathrm{p}}$ *the current state of machine* M_{p}
- $\vec{w} = (w_{\mathrm{pq}})_{\mathrm{pq} \in \mathsf{Chan}}$ with $w_{\mathrm{pq}} \in \mathbb{A}^*$. *the current contents of channels*

The initial configuration of S is $s_0 = (\vec{q_0}, \vec{\varepsilon})$ with $\vec{q_0} = (q_{0_{\mathrm{p}}})_{\mathrm{p} \in \mathrm{P}}$.

Systems of CFSMs

A *system* of CFSMs:

$$S = (M_{\mathrm{p}})_{\mathrm{p} \in \mathsf{P}}$$

- P is the set of *roles* (participants) of S, and
- for each $\mathrm{p} \in \mathsf{P}$, $M_{\mathrm{p}} = (Q_{\mathrm{p}}, q_{0\mathrm{p}}, \mathbb{A}, \delta_{\mathrm{p}})$ is a CFSM.

A *configuration* of S:

$$s = (\vec{q}, \vec{w})$$

- $\vec{q} = (q_{\mathrm{p}})_{\mathrm{p} \in \mathsf{P}}$ *the overall state of the system*
  where $q_{\mathrm{p}} \in Q_{\mathrm{p}}$ *the current state of machine* M_{p}
- $\vec{w} = (w_{\mathrm{pq}})_{\mathrm{pq} \in \mathsf{Chan}}$ with $w_{\mathrm{pq}} \in \mathbb{A}^*$. *the current contents of channels*

The initial configuration of S is $s_0 = (\vec{q_0}, \vec{\varepsilon})$ with $\vec{q_0} = (q_{0_{\mathrm{p}}})_{\mathrm{p} \in \mathsf{P}}$.

Systems of CFSMs

A *system* of CFSMs:

$$S = (M_{\mathrm{p}})_{\mathrm{p} \in \mathsf{P}}$$

- P is the set of *roles* (participants) of S, and
- for each $\mathrm{p} \in \mathsf{P}$, $M_{\mathrm{p}} = (Q_{\mathrm{p}}, q_{0\mathrm{p}}, \mathbb{A}, \delta_{\mathrm{p}})$ is a CFSM.

A *configuration* of S:

$$s = (\vec{q}, \vec{w})$$

- $\vec{q} = (q_{\mathrm{p}})_{\mathrm{p} \in \mathsf{P}}$ *the overall state of the system*
  where $q_{\mathrm{p}} \in Q_{\mathrm{p}}$ *the current state of machine M_{p}*
- $\vec{w} = (w_{\mathrm{pq}})_{\mathrm{pq} \in \mathsf{Chan}}$ with $w_{\mathrm{pq}} \in \mathbb{A}^*$. *the current contents of channels*

The initial configuration of S is $s_0 = (\vec{q_0}, \vec{\varepsilon})$ with $\vec{q_0} = (q_{0_{\mathrm{p}}})_{\mathrm{p} \in \mathsf{P}}$.

Systems of CFSMs

A *system* of CFSMs:

$$S = (M_{\mathrm{p}})_{\mathrm{p} \in \mathsf{P}}$$

- P is the set of *roles* (participants) of S, and
- for each $\mathrm{p} \in \mathsf{P}$, $M_{\mathrm{p}} = (Q_{\mathrm{p}}, q_{0\mathrm{p}}, \mathbb{A}, \delta_{\mathrm{p}})$ is a CFSM.

A *configuration* of S:

$$s = (\vec{q}, \vec{w})$$

- $\vec{q} = (q_{\mathrm{p}})_{\mathrm{p} \in \mathsf{P}}$ *the overall state of the system*
  where $q_{\mathrm{p}} \in Q_{\mathrm{p}}$ *the current state of machine M_{p}*
- $\vec{w} = (w_{\mathrm{pq}})_{\mathrm{pq} \in \mathsf{Chan}}$ with $w_{\mathrm{pq}} \in \mathbb{A}^*$. *the current contents of channels*

The initial configuration of S is $s_0 = (\vec{q_0}, \vec{\varepsilon})$ with $\vec{q_0} = (q_{0_{\mathrm{p}}})_{\mathrm{p} \in \mathsf{P}}$.

Systems of CFSMs

A *system* of CFSMs:

$$S = (M_{\mathrm{p}})_{\mathrm{p} \in \mathsf{P}}$$

- P is the set of *roles* (participants) of S, and
- for each $\mathrm{p} \in \mathsf{P}$, $M_{\mathrm{p}} = (Q_{\mathrm{p}}, q_{0\mathrm{p}}, \mathbb{A}, \delta_{\mathrm{p}})$ is a CFSM.

A *configuration* of S:

$$s = (\vec{q}, \vec{w})$$

- $\vec{q} = (q_{\mathrm{p}})_{\mathrm{p} \in \mathsf{P}}$ *the overall state of the system*
  where $q_{\mathrm{p}} \in Q_{\mathrm{p}}$ *the current state of machine M_{p}*

- $\vec{w} = (w_{\mathrm{pq}})_{\mathrm{pq} \in \mathsf{Chan}}$ with $w_{\mathrm{pq}} \in \mathbb{A}^*$. *the current contents of channels*

The initial configuration of S is $s_0 = (\vec{q_0}, \vec{\varepsilon})$ with $\vec{q_0} = (q_{0_{\mathrm{p}}})_{\mathrm{p} \in \mathsf{P}}$.

Systems of CFSMs

A *system* of CFSMs:

$$S = (M_{\mathrm{p}})_{\mathrm{p} \in \mathrm{P}}$$

- P is the set of *roles* (participants) of S, and
- for each $\mathrm{p} \in \mathrm{P}$, $M_{\mathrm{p}} = (Q_{\mathrm{p}}, q_{0\mathrm{p}}, \mathbb{A}, \delta_{\mathrm{p}})$ is a CFSM.

A *configuration* of S:

$$s = (\vec{q}, \vec{w})$$

- $\vec{q} = (q_{\mathrm{p}})_{\mathrm{p} \in \mathrm{P}}$ *the overall state of the system*
  where $q_{\mathrm{p}} \in Q_{\mathrm{p}}$ *the current state of machine* M_{p}
- $\vec{w} = (w_{\mathrm{pq}})_{\mathrm{pq} \in \mathsf{Chan}}$ with $w_{\mathrm{pq}} \in \mathbb{A}^*$. *the current contents of channels*

The initial configuration of S is $s_0 = (\vec{q_0}, \vec{\varepsilon})$ with $\vec{q_0} = (q_{0_{\mathrm{p}}})_{\mathrm{p} \in \mathrm{P}}$.

Systems of CFSMs

A *system* of CFSMs:

$$S = (M_{\mathrm{p}})_{\mathrm{p} \in \mathsf{P}}$$

- P is the set of *roles* (participants) of S, and
- for each $\mathrm{p} \in \mathsf{P}$, $M_{\mathrm{p}} = (Q_{\mathrm{p}}, q_{0\mathrm{p}}, \mathbb{A}, \delta_{\mathrm{p}})$ is a CFSM.

A *configuration* of S:

$$s = (\vec{q}, \vec{w})$$

- $\vec{q} = (q_{\mathrm{p}})_{\mathrm{p} \in \mathsf{P}}$ *the overall state of the system*
  where $q_{\mathrm{p}} \in Q_{\mathrm{p}}$ *the current state of machine* M_{p}
- $\vec{w} = (w_{\mathrm{pq}})_{\mathrm{pq} \in \mathsf{Chan}}$ with $w_{\mathrm{pq}} \in \mathbb{A}^*$. *the current contents of channels*

The initial configuration of S is $s_0 = (\vec{q_0}, \vec{\varepsilon})$ with $\vec{q_0} = (q_{0_{\mathrm{p}}})_{\mathrm{p} \in \mathsf{P}}$.

System transitions:

$$(\vec{q}, w) \xrightarrow{\mathtt{AB}!msg} (q', w')$$

- In the machine $M_{\mathtt{A}}$:

$$\begin{tikzcd} q_{\mathtt{A}} \arrow[r, bend left, "\mathtt{AB}!\mathsf{msg}"] & q'_{\mathtt{A}} \end{tikzcd}$$

- and the message msg is buffered in the channel from $\mathtt{A}$ to $\mathtt{B}$, that is $w'_{\mathtt{AB}} = w_{\mathtt{AB}} \cdot \mathsf{msg}$ and $\forall \mathtt{pr} \neq \mathtt{AB}.\ w'_{\mathtt{pr}} = w_{\mathtt{pr}}$

System transitions:

$$(\vec{q}, w) \xrightarrow{\mathtt{AB}!msg} (q', w')$$

- In the machine $M_{\mathtt{A}}$:

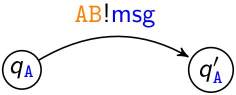
$$\begin{tikzcd} q_{\mathtt{A}} \arrow[r, bend left, "\mathtt{AB}!\mathsf{msg}"] & q'_{\mathtt{A}} \end{tikzcd}$$

- and the message msg is buffered in the channel from $\mathtt{A}$ to $\mathtt{B}$, that is $w'_{\mathtt{AB}} = w_{\mathtt{AB}} \cdot \mathsf{msg}$ and $\forall \mathtt{pr} \neq \mathtt{AB}.\ w'_{\mathtt{pr}} = w_{\mathtt{pr}}$

System transitions:

$$(\vec{q}, w) \xrightarrow{\texttt{AB}!msg} (q', w')$$

- In the machine $M_{\texttt{A}}$:



- and the message msg is buffered in the channel from A to B, that is $w'_{\texttt{AB}} = w_{\texttt{AB}} \cdot \mathsf{msg}$ and $\forall \mathtt{pr} \neq \mathtt{AB}.\ w'_{\mathtt{pr}} = w_{\mathtt{pr}}$

System transitions:

$$(\vec{q}, w) \xrightarrow{\mathtt{AB}!msg} (q', w')$$

- In the machine $M_{\mathtt{A}}$: $$q_{\mathtt{A}} \xrightarrow{\mathtt{AB}!\mathsf{msg}} q'_{\mathtt{A}}$$
- and the message msg is buffered in the channel from $\mathtt{A}$ to $\mathtt{B}$, that is $w'_{\mathtt{AB}} = w_{\mathtt{AB}} \cdot \mathsf{msg}$ and $\forall \mathtt{pr} \neq \mathtt{AB}.\ w'_{\mathtt{pr}} = w_{\mathtt{pr}}$

System transitions:

$$(\vec{q}, w) \xrightarrow{\mathtt{BA}?msg} (q', w')$$

- In the machine $M_{\mathtt{A}}$:

- and the message msg (if present) is popped from top of the buffered channel BA,
  that is $w_{\mathtt{AB}} = \mathsf{msg}\cdot w'_{\mathtt{AB}}$ and $\forall \mathtt{pr} \neq \mathtt{AB}.\ w'_{\mathtt{pr}} = w_{\mathtt{pr}}$

System transitions:

$$(\vec{q}, w) \xrightarrow{\texttt{BA}?msg} (q', w')$$

- In the machine $M_{\texttt{A}}$:

- and the message msg (if present) is popped from top of the buffered channel BA,
  that is $w_{\texttt{AB}} = \text{msg}\cdot w'_{\texttt{AB}}$ and $\forall \texttt{pr} \neq \texttt{AB}.\ w'_{\texttt{pr}} = w_{\texttt{pr}}$

System transitions:

$$(\vec{q}, w) \xrightarrow{\mathtt{BA}?msg} (q', w')$$

- In the machine $M_{\mathtt{A}}$:

$$q_{\mathtt{A}} \xrightarrow{\mathtt{BA}?\mathsf{msg}} q'_{\mathtt{A}}$$

- and the message msg (if present) is popped from top of the buffered channel $\mathtt{BA}$,
  that is $w_{\mathtt{AB}} = \mathsf{msg} \cdot w'_{\mathtt{AB}}$ and $\forall \mathtt{pr} \neq \mathtt{AB}.\ w'_{\mathtt{pr}} = w_{\mathtt{pr}}$

System transitions:

$$(\vec{q}, w) \xrightarrow{\mathtt{BA}?msg} (q', w')$$

- In the machine $M_{\mathtt{A}}$:

$$q_{\mathtt{A}} \xrightarrow{\mathtt{BA}?\mathsf{msg}} q'_{\mathtt{A}}$$

- and the message msg (if present) is popped from top of the buffered channel $\mathtt{BA}$,
  that is $w_{\mathtt{AB}} = \mathsf{msg} \cdot w'_{\mathtt{AB}}$ and $\forall \mathtt{pr} \neq \mathtt{AB}.\ w'_{\mathtt{pr}} = w_{\mathtt{pr}}$

Synchronous communications model for CFSMs

A *configuration* of S: $\vec{q} = (q_{\mathrm{p}})_{\mathrm{p} \in \mathrm{P}}$
where $q_{\mathrm{p}} \in Q_{\mathrm{p}}$ *is the current state of* M_{p}

System transitions: $\vec{q} \xrightarrow{\mathtt{B} \longrightarrow \mathtt{A} \colon \mathsf{msg}} \vec{q'}$
whenever

In the machine $M_{\mathtt{A}}$: $q_{\mathtt{A}} \xrightarrow{\mathtt{BA}?\mathsf{msg}} q'_{\mathtt{A}}$

In the machine $M_{\mathtt{B}}$: $q_{\mathtt{A}} \xrightarrow{\mathtt{BA}!\mathsf{msg}} q'_{\mathtt{A}}$

In all other machines $M_{\mathtt{X}}$ ($\mathtt{X} \neq \mathtt{A}, \mathtt{B}$): $q_{\mathtt{X}} = q'_{\mathtt{X}}$

Synchronous communications model for CFSMs

A *configuration* of S: $\vec{q} = (q_{\mathrm{p}})_{\mathrm{p} \in \mathrm{P}}$
where $q_{\mathrm{p}} \in Q_{\mathrm{p}}$ *is the current state of* M_{p}

System transitions: $\vec{q} \xrightarrow{\mathtt{B} \longrightarrow \mathtt{A}:\ \mathsf{msg}} \vec{q}'$
whenever

In the machine $M_{\mathtt{A}}$:

$$q_{\mathtt{A}} \xrightarrow{\mathtt{BA}?\mathsf{msg}} q'_{\mathtt{A}}$$

In the machine $M_{\mathtt{B}}$:

$$q_{\mathtt{A}} \xrightarrow{\mathtt{BA}!\mathsf{msg}} q'_{\mathtt{A}}$$

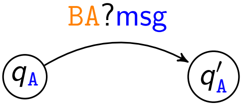
In all other machines $M_{\mathtt{X}}$ $(\mathtt{X} \neq \mathtt{A}, \mathtt{B})$: $q_{\mathtt{X}} = q'_{\mathtt{X}}$

Synchronous communications model for CFSMs

A *configuration* of S: $\vec{q} = (q_{\mathrm{p}})_{\mathrm{p} \in \mathrm{P}}$
where $q_{\mathrm{p}} \in Q_{\mathrm{p}}$ *is the current state of* M_{p}

System transitions: $\vec{q} \xrightarrow{\mathtt{B} \longrightarrow \mathtt{A}:\ \mathsf{msg}} \vec{q'}$
whenever

In the machine $M_{\mathtt{A}}$:



In the machine M_{B}:

In all other machines $M_{\mathtt{X}}$ ($\mathtt{X} \neq \mathtt{A}, \mathtt{B}$): $q_{\mathtt{X}} = {q'}_{\mathtt{X}}$

Synchronous communications model for CFSMs

A *configuration* of S: $\vec{q} = (q_{\mathrm{p}})_{\mathrm{p} \in \mathrm{P}}$

where $q_{\mathrm{p}} \in Q_{\mathrm{p}}$ *is the current state of* M_{p}

System transitions: $\vec{q} \xrightarrow{\mathtt{B} \longrightarrow \mathtt{A}:\ \mathsf{msg}} \vec{q}'$

whenever

In the machine $M_{\mathtt{A}}$:

$$\begin{tikzcd} q_{\mathtt{A}} \arrow[r, bend left, "\mathtt{BA}?\mathsf{msg}"] & q'_{\mathtt{A}} \end{tikzcd}$$

In the machine $M_{\mathtt{B}}$:

$$\begin{tikzcd} q_{\mathtt{A}} \arrow[r, bend left, "\mathtt{BA}!\mathsf{msg}"] & q'_{\mathtt{A}} \end{tikzcd}$$

In all other machines $M_{\mathtt{X}}$ $(\mathtt{X} \neq \mathtt{A}, \mathtt{B})$: $q_{\mathtt{X}} = q'_{\mathtt{X}}$

Synchronous communications model for CFSMs

A *configuration* of S: $\vec{q} = (q_{\mathtt{p}})_{\mathtt{p} \in \mathsf{P}}$

where $q_{\mathtt{p}} \in Q_{\mathtt{p}}$ *is the current state of* $M_{\mathtt{p}}$

System transitions: $\vec{q} \xrightarrow{\mathtt{B} \longrightarrow \mathtt{A} \colon \mathsf{msg}} \vec{q'}$

whenever

In the machine $M_{\mathtt{A}}$:

$$q_{\mathtt{A}} \xrightarrow{\mathtt{BA}?\mathsf{msg}} q'_{\mathtt{A}}$$

In the machine $M_{\mathtt{B}}$:

$$q_{\mathtt{A}} \xrightarrow{\mathtt{BA}!\mathsf{msg}} q'_{\mathtt{A}}$$

In all other machines $M_{\mathtt{X}}$ ($\mathtt{X} \neq \mathtt{A}, \mathtt{B}$): $q_{\mathtt{X}} = q'_{\mathtt{X}}$

Relevant properties of systems

- **Liveness:**
  whenever a machine is willing to perform some actions, the system can evolve so that one of those actions is eventually done
- **Deadlock-Freedom:**
  in case the system get stuck, no machine is in a state with an outgoing transition (the system do progress)
- **Lock-Freedom:**
  if a machine can perform some actions, sooner or later it will do one (any single machine does progress)

We restrict the attention to *fair* runs of systems.

Relevant properties of systems

- **Liveness:**
  whenever a machine is willing to perform some actions, the system can evolve so that one of those actions is eventually done
- **Deadlock-Freedom:**
  in case the system get stuck, no machine is in a state with an outgoing transition (the system do progress)
- **Lock-Freedom:**
  if a machine can perform some actions, sooner or later it will do one (any single machine does progress)

We restrict the attention to *fair* runs of systems.

Relevant properties of systems

- **Liveness:**
  whenever a machine is willing to perform some actions, the system can evolve so that one of those actions is eventually done
- **Deadlock-Freedom:**
  in case the system get stuck, no machine is in a state with an outgoing transition (the system do progress)
- **Lock-Freedom:**
  if a machine can perform some actions, sooner or later it will do one (any single machine does progress)

We restrict the attention to *fair* runs of systems.

Relevant properties of systems

- **Liveness:**
  whenever a machine is willing to perform some actions, the system can evolve so that one of those actions is eventually done
- **Deadlock-Freedom:**
  in case the system get stuck, no machine is in a state with an outgoing transition (the system do progress)
- **Lock-Freedom:**
  if a machine can perform some actions, sooner or later it will do one (any single machine does progress)

We restrict the attention to *fair* runs of systems.

Relevant properties of systems

- **Liveness:**
  whenever a machine is willing to perform some actions, the system can evolve so that one of those actions is eventually done
- **Deadlock-Freedom:**
  in case the system get stuck, no machine is in a state with an outgoing transition (the system do progress)
- **Lock-Freedom:**
  if a machine can perform some actions, sooner or later it will do one (any single machine does progress)

We restrict the attention to *fair* runs of systems.

Relevant properties of systems

- **Liveness:**
  whenever a machine is willing to perform some actions, the system can evolve so that one of those actions is eventually done
- **Deadlock-Freedom:**
  in case the system get stuck, no machine is in a state with an outgoing transition (the system do progress)
- **Lock-Freedom:**
  if a machine can perform some actions, sooner or later it will do one (any single machine does progress)

We restrict the attention to *fair* runs of systems.

Relevant properties of systems

- **Liveness:**
  whenever a machine is willing to perform some actions, the system can evolve so that one of those actions is eventually done
- **Deadlock-Freedom:**
  in case the system get stuck, no machine is in a state with an outgoing transition (the system do progress)
- **Lock-Freedom:**
  if a machine can perform some actions, sooner or later it will do one (any single machine does progress)

We restrict the attention to *fair* runs of systems.

Relevant properties of systems

- **Liveness:**
  whenever a machine is willing to perform some actions, the system can evolve so that one of those actions is eventually done
- **Deadlock-Freedom:**
  in case the system get stuck, no machine is in a state with an outgoing transition (the system do progress)
- **Lock-Freedom:**
  if a machine can perform some actions, sooner or later it will do one (any single machine does progress)

We restrict the attention to *fair* runs of systems.

Choreographies for CFSMs systems: Which description formalism?

It takes a thief to catch a thief... so

Choreography Automata

Choreographies for CFSMs systems: Which description formalism?

It takes a thief to catch a thief... so

Choreography Automata

Choreographies for CFSMs systems: Which description formalism?

It takes a thief to catch a thief... so

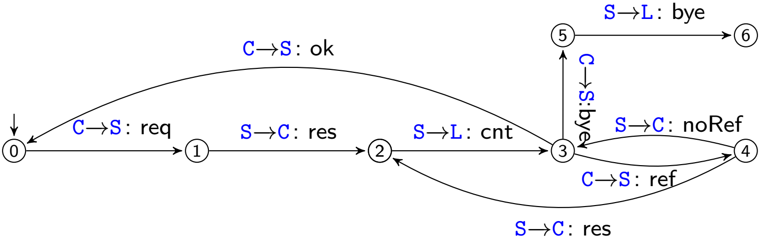
Choreography Automata

Choreographies for CFSMs systems: Which description formalism?

It takes a thief to catch a thief... so

Choreography Automata

Choreography Automata through an Example



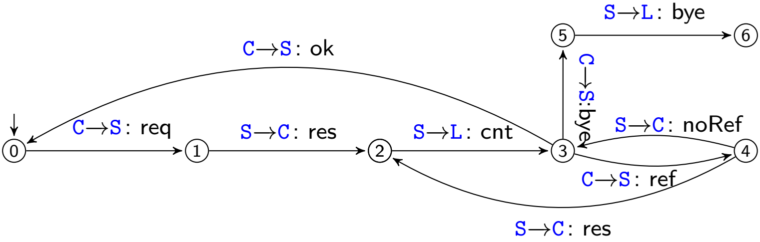
Which sequences of interactions are represented?

* A finite state automaton where all states are final.

Moreover we take all infinite words whose all finite prefixes are accepted.

(prefix-closed and continuous sets of interaction sequences).

Choreography Automata through an Example



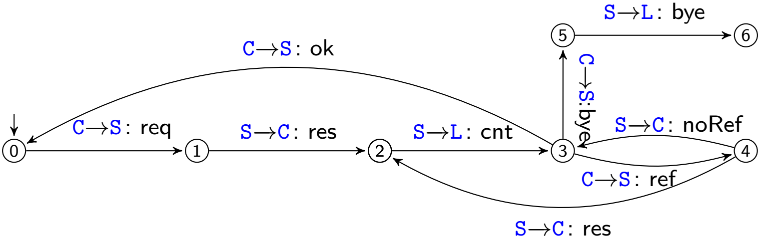
Which sequences of interactions are represented?

* A finite state automaton where all states are final.

Moreover we take all infinite words whose all finite prefixes are accepted.

(prefix-closed and continuous sets of interaction sequences).

Choreography Automata through an Example



Which sequences of interactions are represented?

* A finite state automaton where all states are final.

Moreover we take all infinite words whose all finite prefixes are accepted.

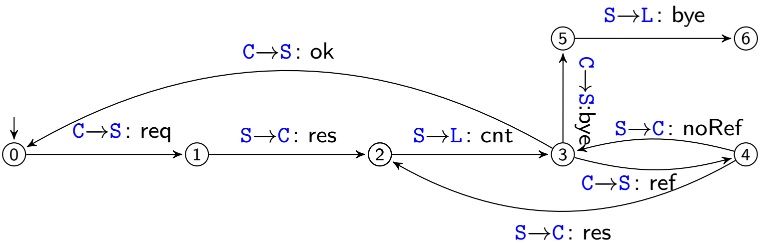
(prefix-closed and continuous sets of interaction sequences).

An apparent resemblance

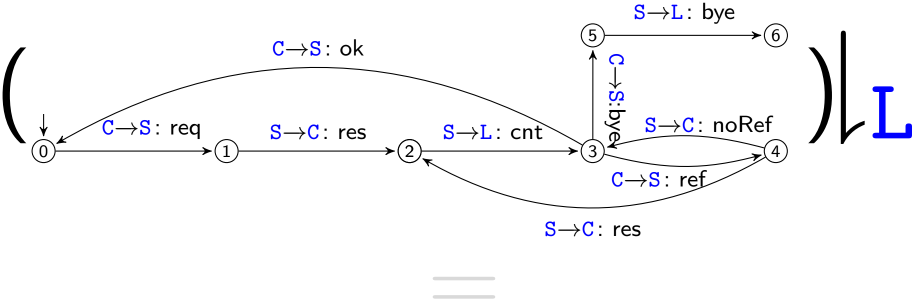
Choreography Automata **vs.** Conversation Protocols
(by Bultan et al.)

They look alike, but actually their semantics and underlying communication models do differ.
(a thorough comparison in the Related Works section of the paper)

Choreography Automata through an Example

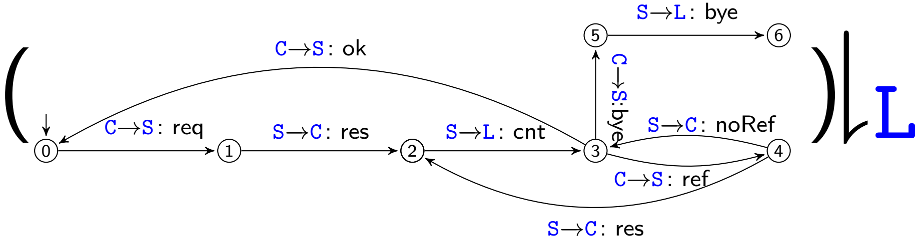


Projection



C→S: ok
S→L: bye
C→S: req
S→C: res
S→L: cnt
C→S:bye
S→C: noRef
C→S: ref
S→C: res
0
1
2
3
4
5
6
L

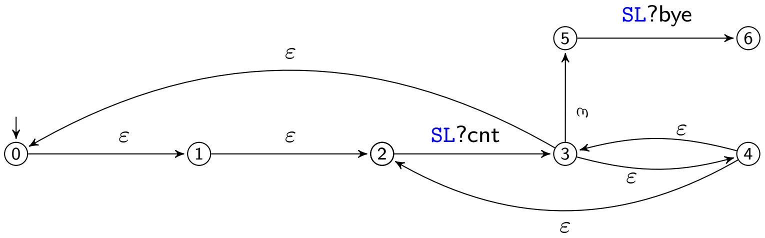
Projection



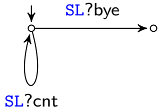
S→L: bye
C→S: ok
C→S:bye
C→S: req
S→C: res
S→L: cnt
S→C: noRef
C→S: ref
S→C: res
0
1
2
3
4
5
6
L

$=$

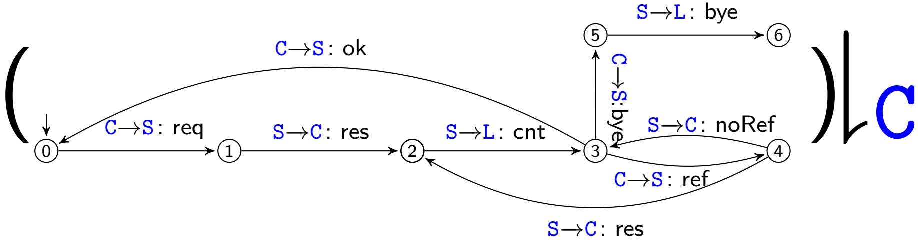
Projection



Projection

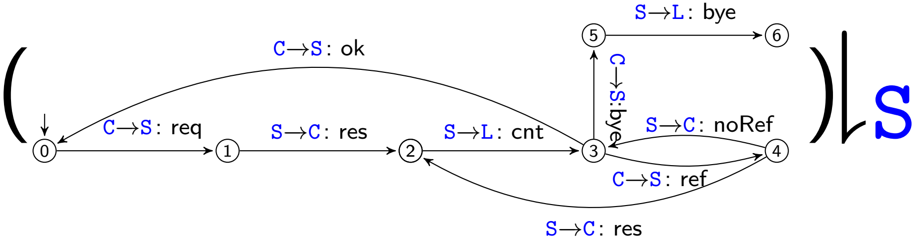


Projection



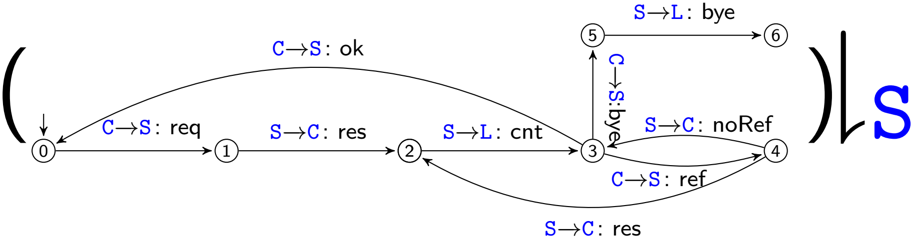
S→L: bye
C→S: ok
C→S:bye
S→C: noRef
C→S: req
S→C: res
S→L: cnt
C→S: ref
S→C: res
0
1
2
3
4
5
6
C

Projection



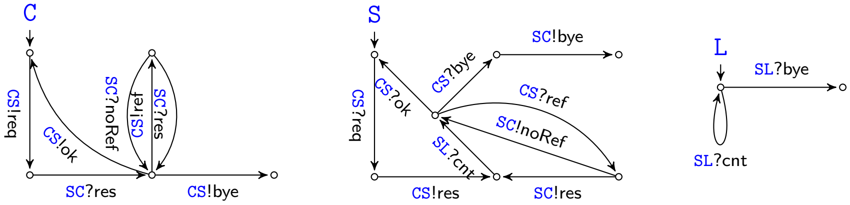
C→S: ok
S→L: bye
C→S:bye
C→S: req
S→C: res
S→L: cnt
S→C: noRef
C→S: ref
S→C: res
0
1
2
3
4
5
6
S

Projection



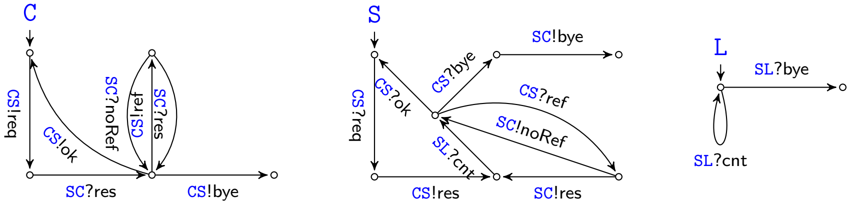
C→S: ok
S→L: bye
0
C→S: req
1
S→C: res
2
S→L: cnt
3
C→S:bye
5
6
S→C: noRef
4
C→S: ref
S→C: res
S

Projection



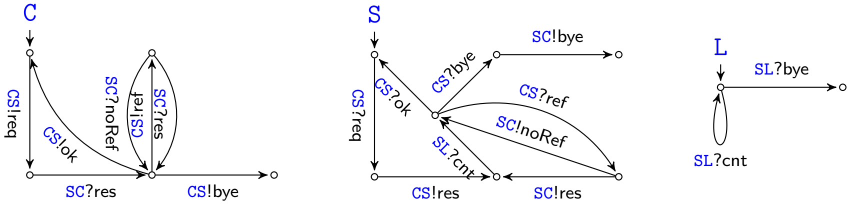
- The behaviour of the system of CFSMs perfectly matches the overall behaviour described by the choreography automata:
- The system is **Live**, *i.e. if a machine wants to perform some actions, the system can evolve so that one of them eventually is done*
- The system is **Deadlock-Free** *i.e. it will never get stuck (the system does eventually progress)*
- The system is **Lock-Free** *i.e. if a machine can perform some actions, sooner or later it will do one (any single machine does eventually progress)*

Projection



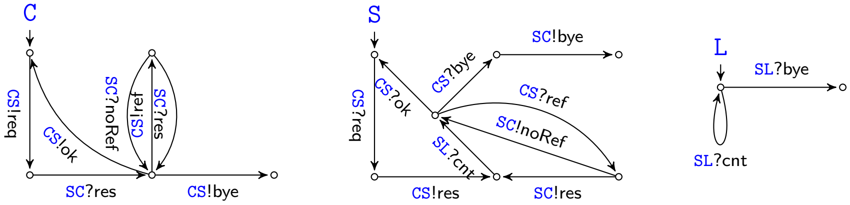
- The behaviour of the system of CFSMs perfectly matches the overall behaviour described by the choreography automata:
- The system is **Live**, *i.e. if a machine wants to perform some actions, the system can evolve so that one of them eventually is done*
- The system is **Deadlock-Free** *i.e. it will never get stuck (the system does eventually progress)*
- The system is **Lock-Free** *i.e. if a machine can perform some actions, sooner or later it will do one (any single machine does eventually progress)*

Projection



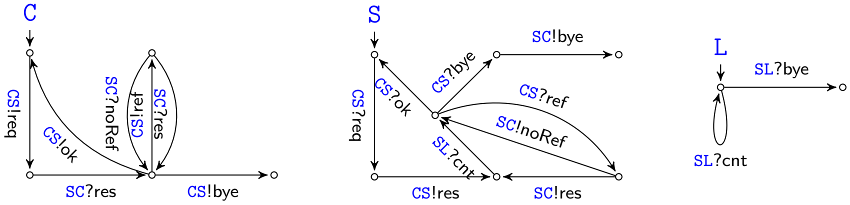
- The behaviour of the system of CFSMs perfectly matches the overall behaviour described by the choreography automata:
- The system is **Live**, *i.e. if a machine wants to perform some actions, the system can evolve so that one of them eventually is done*
- The system is **Deadlock-Free** *i.e. it will never get stuck (the system does eventually progress)*
- The system is **Lock-Free** *i.e. if a machine can perform some actions, sooner or later it will do one (any single machine does eventually progress)*

Projection



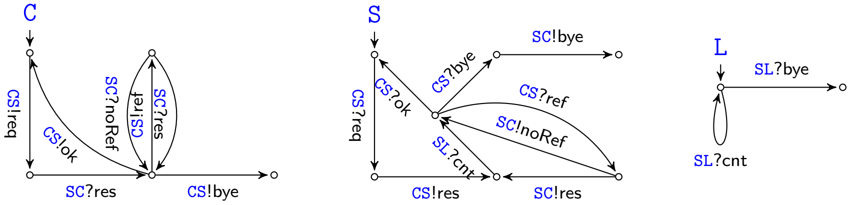
- The behaviour of the system of CFSMs perfectly matches the overall behaviour described by the choreography automata:
- The system is **Live**, *i.e. if a machine wants to perform some actions, the system can evolve so that one of them eventually is done*
- The system is **Deadlock-Free** *i.e. it will never get stuck (the system does eventually progress)*
- The system is **Lock-Free** *i.e. if a machine can perform some actions, sooner or later it will do one (any single machine does eventually progress)*

Projection



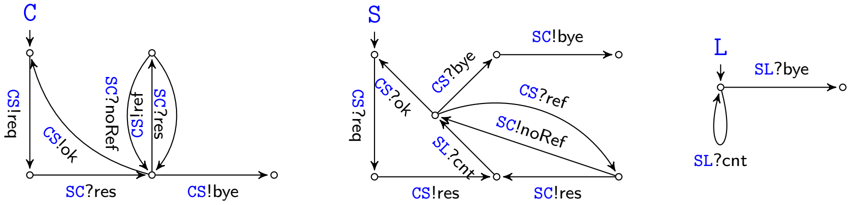
- The behaviour of the system of CFSMs perfectly matches the overall behaviour described by the choreography automata:
- The system is **Live**, *i.e. if a machine wants to perform some actions, the system can evolve so that one of them eventually is done*
- The system is **Deadlock-Free** *i.e. it will never get stuck (the system does eventually progress)*
- The system is **Lock-Free** *i.e. if a machine can perform some actions, sooner or later it will do one (any single machine does eventually progress)*

Projection



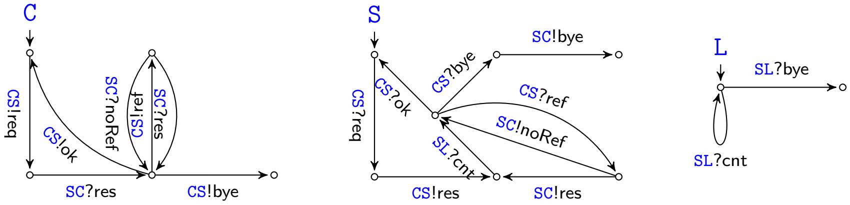
- The behaviour of the system of CFSMs perfectly matches the overall behaviour described by the choreography automata:
- The system is **Live**, *i.e. if a machine wants to perform some actions, the system can evolve so that one of them eventually is done*
- The system is **Deadlock-Free** *i.e. it will never get stuck (the system does eventually progress)*
- The system is **Lock-Free** *i.e. if a machine can perform some actions, sooner or later it will do one (any single machine does eventually progress)*

Projection



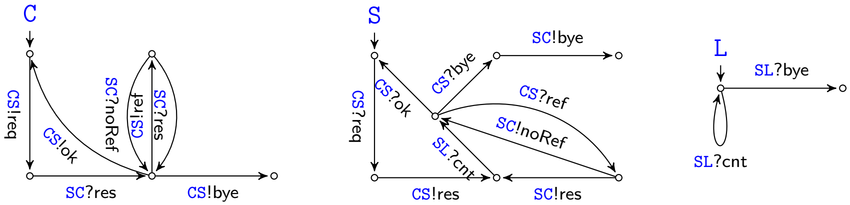
- The behaviour of the system of CFSMs perfectly matches the overall behaviour described by the choreography automata:
- The system is **Live**, *i.e. if a machine wants to perform some actions, the system can evolve so that one of them eventually is done*
- The system is **Deadlock-Free** *i.e. it will never get stuck (the system does eventually progress)*
- The system is **Lock-Free**
  i.e. if a machine can perform some actions, sooner or later it will do one (any single machine does eventually progress)

Projection



- The behaviour of the system of CFSMs perfectly matches the overall behaviour described by the choreography automata:
- The system is **Live**, *i.e. if a machine wants to perform some actions, the system can evolve so that one of them eventually is done*
- The system is **Deadlock-Free** *i.e. it will never get stuck (the system does eventually progress)*
- The system is **Lock-Free** *i.e. if a machine can perform some actions, sooner or later it will do one (any single machine does eventually progress)*

Projection



These good properties do hold in case of either **Synchronous** or **Asynchronous** communications

There ain't no such thing as a free lunch

Only the projections of *well-behaved* Choreography Automata are *well-behaved*.

Theorem

Given a well-formed c-automaton CA, the system obtained by projection, $(CA\lfloor_{\mathtt{A}})_{\mathtt{A}\in\mathcal{P}}$, is live, lock-free, and deadlock-free both for synchronous and asynchronous communications.

Definition (Well-formedness)

A c-automaton CA is well-formed if (roughly)

- when there is a choice, a single participant decides;
- all the partecipants are made aware of the choices affecting their expected behaviour ;
- parallelism of independent interactions must be made explicit by interleaving them

Slight changes between the synchronous and the asynchronous cases.

There ain't no such thing as a free lunch

Only the projections of *well-behaved* Choreography Automata are *well-behaved*.

Theorem

Given a well-formed c-automaton CA, the system obtained by projection, $(CA\downharpoonright_{\mathtt{A}})_{\mathtt{A}\in\mathcal{P}}$, is live, lock-free, and deadlock-free both for synchronous and asynchronous communications.

Definition (Well-formedness)

A c-automaton CA is well-formed if (roughly)

- when there is a choice, a single participant decides;
- all the partecipants are made aware of the choices affecting their expected behaviour ;
- parallelism of independent interactions must be made explicit by interleaving them

Slight changes between the synchronous and the asynchronous cases.

There ain't no such thing as a free lunch

Only the projections of *well-behaved* Choreography Automata are *well-behaved*.

Theorem

Given a well-formed c-automaton CA, the system obtained by projection, $(CA\lfloor_{\mathtt{A}})_{\mathtt{A}\in\mathcal{P}}$, is live, lock-free, and deadlock-free both for synchronous and asynchronous communications.

Definition (Well-formedness)

A c-automaton CA is well-formed if (roughly)

- when there is a choice, a single participant decides;
- all the partecipants are made aware of the choices affecting their expected behaviour ;
- parallelism of independent interactions must be made explicit by interleaving them

Slight changes between the synchronous and the asynchronous cases.

There ain't no such thing as a free lunch

Only the projections of *well-behaved* Choreography Automata are *well-behaved*.

Theorem

Given a well-formed c-automaton CA, the system obtained by projection, $(CA\lfloor_{\mathtt{A}})_{\mathtt{A}\in\mathcal{P}}$, is live, lock-free, and deadlock-free both for synchronous and asynchronous communications.

Definition (Well-formedness)

A c-automaton CA is well-formed if (roughly)

- when there is a choice, a single participant decides;
- all the partecipants are made aware of the choices affecting their expected behaviour ;
- parallelism of independent interactions must be made explicit by interleaving them

Slight changes between the synchronous and the asynchronous cases.

There ain't no such thing as a free lunch

Only the projections of *well-behaved* Choreography Automata are *well-behaved*.

Theorem

Given a well-formed c-automaton CA, the system obtained by projection, $(CA\lfloor_{\mathtt{A}})_{\mathtt{A}\in\mathcal{P}}$, *is live, lock-free, and deadlock-free both for synchronous and asynchronous communications.*

Definition (Well-formedness)

A c-automaton CA is well-formed if (roughly)

- when there is a choice, a single participant decides;
- all the partecipants are made aware of the choices affecting their expected behaviour ;
- parallelism of independent interactions must be made explicit by interleaving them

Slight changes between the synchronous and the asynchronous cases.

There ain't no such thing as a free lunch

Only the projections of *well-behaved* Choreography Automata are *well-behaved*.

Theorem

Given a well-formed c-automaton CA, the system obtained by projection, $(CA\lfloor_{\mathtt{A}})_{\mathtt{A}\in\mathcal{P}}$, is live, lock-free, and deadlock-free both for synchronous and asynchronous communications.

Definition (Well-formedness)

A c-automaton CA is well-formed if (roughly)

- when there is a choice, a single participant decides;
- all the partecipants are made aware of the choices affecting their expected behaviour ;
- parallelism of independent interactions must be made explicit by interleaving them

Slight changes between the synchronous and the asynchronous cases.

There ain't no such thing as a free lunch

Only the projections of *well-behaved* Choreography Automata are *well-behaved*.

Theorem

Given a well-formed c-automaton CA, the system obtained by projection, $(CA\lfloor_{\mathtt{A}})_{\mathtt{A}\in\mathcal{P}}$*, is live, lock-free, and deadlock-free both for synchronous and asynchronous communications.*

Definition (Well-formedness)

A c-automaton CA is well-formed if (roughly)

- when there is a choice, a single participant decides;
- all the partecipants are made aware of the choices affecting their expected behaviour ;
- parallelism of independent interactions must be made explicit by interleaving them

Slight changes between the synchronous and the asynchronous cases.

There ain't no such thing as a free lunch

Only the projections of *well-behaved* Choreography Automata are *well-behaved*.

Theorem

Given a well-formed c-automaton CA, the system obtained by projection, $(CA\lfloor_{\mathtt{A}})_{\mathtt{A}\in\mathcal{P}}$, is live, lock-free, and deadlock-free both for synchronous and asynchronous communications.

Definition (Well-formedness)

A c-automaton CA is well-formed if (roughly)

- when there is a choice, a single participant decides;
- all the partecipants are made aware of the choices affecting their expected behaviour ;
- parallelism of independent interactions must be made explicit by interleaving them

Slight changes between the synchronous and the asynchronous cases.

Well-formedness = Well-sequenced + Well-branched

Definition (Well-sequencedness (synchronous))

A c-automaton is *well-sequenced* if for each two consecutive transitions $q \xrightarrow{\mathtt{A} \longrightarrow \mathtt{B}:\ \mathsf{m}} q' \xrightarrow{\mathtt{C} \longrightarrow \mathtt{D}:\ \mathsf{n}} q''$ either

- they share a participant, that is $\{\mathtt{A}, \mathtt{B}\} \cap \{\mathtt{C}, \mathtt{D}\} \neq \emptyset$, or
- they are concurrent, i.e. there is q''' such that $q \xrightarrow{\mathtt{C} \longrightarrow \mathtt{D}:\ \mathsf{n}} q''' \xrightarrow{\mathtt{A} \longrightarrow \mathtt{B}:\ \mathsf{m}} q''$.

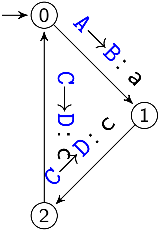
Well-formedness = Well-sequenced + Well-branched

Definition (Well-sequencedness (synchronous))

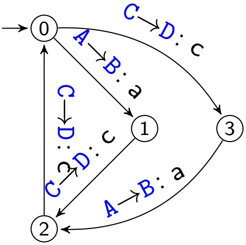
A c-automaton is *well-sequenced* if for each two consecutive transitions $q \xrightarrow{\mathtt{A}\rightarrow\mathtt{B}:\,\mathsf{m}} q' \xrightarrow{\mathtt{C}\rightarrow\mathtt{D}:\,\mathsf{n}} q''$ either

- they share a participant, that is $\{\mathtt{A}, \mathtt{B}\} \cap \{\mathtt{C}, \mathtt{D}\} \neq \emptyset$, or
- they are concurrent, i.e. there is q''' such that $q \xrightarrow{\mathtt{C}\rightarrow\mathtt{D}:\,\mathsf{n}} q''' \xrightarrow{\mathtt{A}\rightarrow\mathtt{B}:\,\mathsf{m}} q''$.

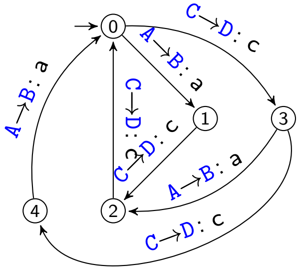
Not all c-automata can be “completed” to well-sequenced ones.



(a)



(b)



(c)

Well-formedness = Well-sequenced + Well-branched

Definition (Well-branchedness (synch and asynch))

A c-automaton is *well-branched* if for each state q in and $\mathtt{A} \in \mathcal{P}$ sender in a transition from q, all of the following conditions must hold:

(1) all transitions from q involving $\mathtt{A}$, have sender $\mathtt{A}$;

(2) for each transition t from q whose sender is not $\mathtt{A}$ and each transition t' from q whose sender is $\mathtt{A}$, t and t' are concurrent

(3) for each q-span (σ, σ') where $\mathtt{A}$ chooses at and each participant $\mathtt{B} \neq \mathtt{A} \in \mathcal{P}$, the first pair of different labels on the runs σB and $\sigma' B$ (if any) is of the form $(CB?m, DB?n)$ with $\mathtt{C} \neq \mathtt{D}$ or m $\neq$ n.

We dub $\mathtt{A}$ a *selector at q*.

Well-formedness = Well-sequenced + Well-branched

Definition (Well-branchedness (synch and asynch))

A c-automaton is *well-branched* if for each state q in and $\mathtt{A} \in \mathcal{P}$ sender in a transition from q, all of the following conditions must hold:

(1) all transitions from q involving $\mathtt{A}$, have sender $\mathtt{A}$;

(2) for each transition t from q whose sender is not $\mathtt{A}$ and each transition t' from q whose sender is $\mathtt{A}$, t and t' are concurrent

(3) for each q-span (σ, σ') where $\mathtt{A}$ chooses at and each participant $\mathtt{B} \neq \mathtt{A} \in \mathcal{P}$, the first pair of different labels on the runs σB and $\sigma' B$ (if any) is of the form $(CB?m, DB?n)$ with $\mathtt{C} \neq \mathtt{D}$ or m $\neq$ n.

We dub $\mathtt{A}$ a *selector at q*.

Future work: Towards choreographic models for open systems

Usually choreographic models are good for the description of **closed** systems. What about **open** systems?

A starting point:
The "participants as interfaces" approach to open (**i.e. composable**) systems of (asynchronous) CFSMs

Barbanera, de'Liguoro, Hennicker
CONNECTING OPEN SYSTEMS OF COMMUNICATING FINITE STATE MACHINES (JLAMP)

Future work: Towards choreographic models for open systems

Usually choreographic models are good for the description of **closed** systems. What about **open** systems?

A starting point:
The "participants as interfaces" approach to open (**i.e. composable**) systems of (asynchronous) CFSMs

Barbanera, de'Liguoro, Hennicker
Connecting open systems of communicating finite state machines (JLAMP)

Future work: Towards choreographic models for open systems

Usually choreographic models are good for the description of **closed** systems. What about **open** systems?

A starting point:
The "participants as interfaces" approach to open (**i.e. composable**) systems of (asynchronous) CFSMs

Barbanera, de'Liguoro, Hennicker
CONNECTING OPEN SYSTEMS OF COMMUNICATING FINITE STATE MACHINES (JLAMP)

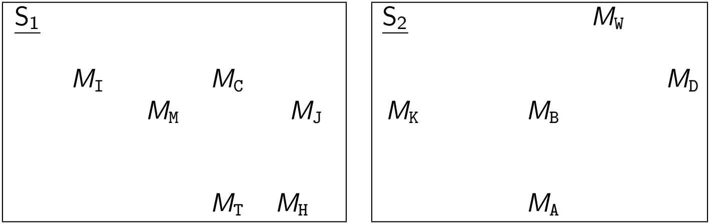
Future work: Towards choreographic models for open systems

Usually choreographic models are good for the description of **closed** systems. What about **open** systems?

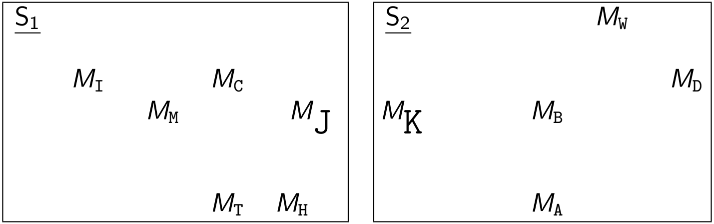
A starting point:
The "participants as interfaces" approach to open (**i.e. composable**) systems of (asynchronous) CFSMs

Barbanera, de'Liguoro, Hennicker
Connecting open systems of communicating finite state machines (JLAMP)

The "participants as interfaces" approach to open systems

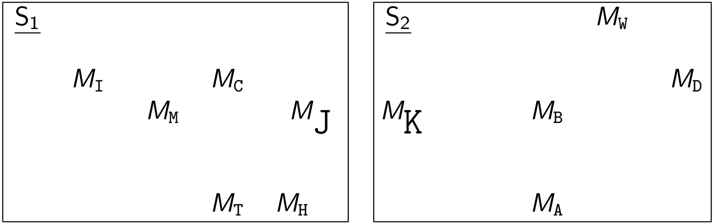


The “participants as interfaces” approach to open systems



ANY participant can be looked at as an interface.

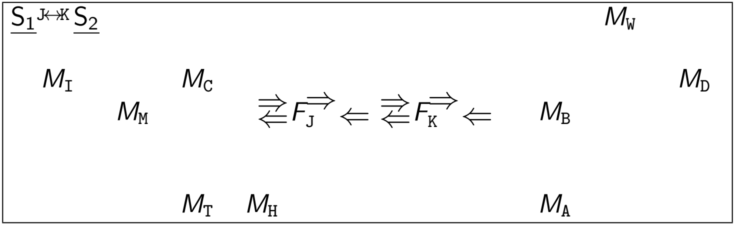
The “participants as interfaces” approach to open systems



ANY participant can be looked at as an interface.

If J and K are “compatible”
the two open systems can be connected via `J` and `K`
by simply replacing them by “forwarders”.

The “participants as interfaces” approach to open systems



ANY participant can be looked at as an interface.

If J and K are “compatible”
the two open systems can be connected via $\mathtt{J}$ and $\mathtt{K}$
by simply replacing them by “forwarders”.
* Good properties of the systems are preseved by composition.*

The “participants as interfaces” approach to open systems

A preliminary investigation of the “participants as interfaces” approach to open systems of **synchronous** CFSMs

Barbanera, Lanese, Tuosto
Composing Communicating Systems, Synchronously ISoLA 2020

The “participants as interfaces” approach to open systems

A preliminary investigation of the “participants as interfaces” approach to open systems of **synchronous** CFSMs

Barbanera, Lanese, Tuosto
Composing Communicating Systems, Synchronously ISoLA 2020

Future work: Towards choreographic models for open systems

A first step (**done**): Using Global Types to internally describe the "participants as interfaces" composition mechanism on global specifications (preserving well-formedness)

Barbanera, Dezani, Lanese, Tuosto
COMPOSITION AND DECOMPOSITION OF MULTIPARTY SESSIONS
(JLAMP)

The second step (to do):
Extending the approach to Coreography automata.

Future work: Towards choreographic models for open systems

A first step (**done**): Using Global Types to internally describe the "participants as interfaces" composition mechanism on global specifications (preserving well-formedness)

Barbanera, Dezani, Lanese, Tuosto
COMPOSITION AND DECOMPOSITION OF MULTIPARTY SESSIONS
(JLAMP)

The second step (to do):
Extending the approach to Coreography automata.

Future work: Towards choreographic models for open systems

A first step (**done**): Using Global Types to internally describe the "participants as interfaces" composition mechanism on global specifications (preserving well-formedness)

Barbanera, Dezani, Lanese, Tuosto
COMPOSITION AND DECOMPOSITION OF MULTIPARTY SESSIONS (JLAMP)

The second step (to do):
Extending the approach to Coreography automata.