Choreography Automata

Franco Barbanera\textsuperscript{1}, Ivan Lanese\textsuperscript{2}, Emilio Tuosto\textsuperscript{3}

\textsuperscript{1} University of Catania
\textsuperscript{2} University of Bologna/INRIA
\textsuperscript{3} GSSI/University of Leicester

Coordination@DisCoTec 2020 - Malta, June 2020
Good ideas are recyclable

If you have a bunch of dancers...
Good ideas are recyclable

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.
The recycling

preparation

Whole

Synch/Async model

studio and reflection

Infrastructure

Interaction

Process

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

*projection* is an operation producing the local view from the global one.
The choreographic approach:
A lighthouse on the Formal Verification road

- specification languages: WS-CDL, BPMN, ...
- choreographies for microservices;
- experimental choreographic languages: Chor
- etc.
Bounty Hunters Attention!

WANTED

A simple, clear and widely-agreed upon,

Theoretical Choreography Model

$5,000 REWARD!

Notify nearest law enforcement agency
Which abstraction for processes?
Which abstraction for processes?
A machine $M_A$ can send $msg_1$ to machine $M_B$; asynchronously; through the directed buffered FIFO channel $AB$

Then, either $msg_2$ or $msg_3$ can be received from $M_B$; through channel $BA$;

and so on....
Communicating Finite State Machines (CFSMs)

A formalism for the description and the analysis of distributed systems.

A machine $M_A$

- $M_A$ can send msg1 to machine $M_B$; asynchronously; through the directed buffered FIFO channel AB
- Then, either msg2 or msg3 can be received from $M_B$; through channel BA;
- and so on....
Communicating Finite State Machines (CFSMs)

A formalism for the description and the analysis of distributed systems.

A machine $M_A$

- $M_A$ can send $msg_1$ to machine $M_B$; \textit{asynchronously}; \textit{through the directed buffered FIFO channel AB}
- Then, either $msg_2$ or $msg_3$ can be received from $M_B$; \textit{through channel BA};
- and so on....
Communicating Finite State Machines (CFSMs)

A formalism for the description and the analysis of distributed systems.

A machine $M_A$

- $M_A$ can send msg1 to machine $M_B$; **asynchronously**; through the directed buffered FIFO channel $AB$
- Then, either msg2 or msg3 can be received from $M_B$; through channel $BA$;
- and so on....
Communicating Finite State Machines (CFSMs)

A formalism for the description and the analysis of distributed systems.

A machine $M_A$

- $M_A$ can send msg1 to machine $M_B$; **asynchronously**; through the directed buffered FIFO channel AB
- Then, either msg2 or msg3 can be received from $M_B$; through channel BA;
- and so on....
Systems of CFSMs

A system of CFSMs:

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

- $P$ is the set of roles (participants) of $S$, and
- for each $p \in P$, $M_p = (Q_p, q_{0p}, A, \delta_p)$ is a CFSM.

A configuration of $S$:

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

- $\vec{q} = (q_p)_{p \in P}$ the overall state of the system where $q_p \in Q_p$ the current state of machine $M_p$
- $\vec{w} = (w_{pq})_{pq \in \text{Chan}}$ with $w_{pq} \in A^*$. the current contents of channels

The initial configuration of $S$ is $s_0 = (\vec{q}_0, \vec{e})$ with $\vec{q}_0 = (q_{0p})_{p \in P}$. 
Systems of CFSMs

A system of CFSMs:

\[ S = (M_p)_{p \in P} \]

- \( P \) is the set of roles (participants) of \( S \), and
- for each \( p \in P \), \( M_p = (Q_p, q_{0p}, \Lambda, \delta_p) \) is a CFSM.

A configuration of \( S \):

\[ s = (\vec{q}, \vec{w}) \]

- \( \vec{q} = (q_p)_{p \in P} \) the overall state of the system
  where \( q_p \in Q_p \) the current state of machine \( M_p \)
- \( \vec{w} = (w_{pq})_{pq \in \text{Chan}} \) with \( w_{pq} \in \Lambda^* \) the current contents of channels

The initial configuration of \( S \) is \( s_0 = (\vec{q}_0, \vec{e}) \) with \( \vec{q}_0 = (q_{0p})_{p \in P} \).
Systems of CFSMs

A system of CFSMs:

\[ S = (M_p)_{p \in P} \]

- \( P \) is the set of roles (participants) of \( S \), and
- for each \( p \in P \), \( M_p = (Q_p, q_{0p}, \Delta, \delta_p) \) is a CFSM.

A configuration of \( S \):

\[ s = (\vec{q}, \vec{w}) \]

- \( \vec{q} = (q_p)_{p \in P} \) the overall state of the system
  where \( q_p \in Q_p \) the current state of machine \( M_p \)
- \( \vec{w} = (w_{pq})_{pq \in \text{Chan}} \) with \( w_{pq} \in A^* \). the current contents of channels

The initial configuration of \( S \) is \( s_0 = (\vec{q}_0, \vec{e}) \) with \( \vec{q}_0 = (q_{0p})_{p \in P} \).
Systems of CFSMs

A system of CFSMs:

\[ S = (M_p)_{p \in P} \]

- \( P \) is the set of roles (participants) of \( S \), and
- for each \( p \in P \), \( M_p = (Q_p, q_{0p}, \Delta, \delta_p) \) is a CFSM.

A configuration of \( S \):

\[ s = (\vec{q}, \vec{w}) \]

- \( \vec{q} = (q_p)_{p \in P} \) the overall state of the system
  where \( q_p \in Q_p \) the current state of machine \( M_p \)
- \( \vec{w} = (w_{pq})_{pq \in \text{Chan}} \) with \( w_{pq} \in \mathbb{A}^* \) the current contents of channels

The initial configuration of \( S \) is \( s_0 = (\vec{q}_0, \vec{0}) \) with \( \vec{q}_0 = (q_{0p})_{p \in P} \).
Systems of CFSMs

A system of CFSMs:

\[ S = (M_p)_{p \in P} \]

- \( P \) is the set of roles (participants) of \( S \), and
- for each \( p \in P \), \( M_p = (Q_p, q_{0p}, A, \delta_p) \) is a CFSM.

A configuration of \( S \):

\[ s = (\vec{q}, \vec{w}) \]

- \( \vec{q} = (q_p)_{p \in P} \) the overall state of the system
  where \( q_p \in Q_p \) the current state of machine \( M_p \)
- \( \vec{w} = (w_{pq})_{pq \in \text{Chan}} \) with \( w_{pq} \in A^* \). the current contents of channels

The initial configuration of \( S \) is \( s_0 = (\vec{q}_0, \vec{e}) \) with \( \vec{q}_0 = (q_{0p})_{p \in P} \).
A system of CFSMs:

\[ S = (M_p)_{p \in P} \]

- \( P \) is the set of roles (participants) of \( S \), and
- for each \( p \in P \), \( M_p = (Q_p, q_{0p}, A, \delta_p) \) is a CFSM.

A configuration of \( S \):

\[ s = (\vec{q}, \vec{w}) \]

- \( \vec{q} = (q_p)_{p \in P} \) \textit{the overall state of the system}
  where \( q_p \in Q_p \) \textit{the current state of machine} \( M_p \)
- \( \vec{w} = (w_{pq})_{pq \in \text{Chan}} \) with \( w_{pq} \in A^* \) \textit{the current contents of channels}

The initial configuration of \( S \) is \( s_0 = (\vec{q}_0, \vec{e}) \) with \( \vec{q}_0 = (q_{0p})_{p \in P} \).
Systems of CFSMs

A system of CFSMs:

\[ S = (M_p)_{p \in P} \]

- \( P \) is the set of roles (participants) of \( S \), and
- for each \( p \in P \), \( M_p = (Q_p, q_{0p}, A, \delta_p) \) is a CFSM.

A configuration of \( S \):

\[ s = (\vec{q}, \vec{w}) \]

- \( \vec{q} = (q_p)_{p \in P} \) the overall state of the system
  - where \( q_p \in Q_p \) the current state of machine \( M_p \)
- \( \vec{w} = (w_{pq})_{pq \in \text{Chan}} \) with \( w_{pq} \in \Delta^* \) the current contents of channels

The initial configuration of \( S \) is \( s_0 = (\vec{q}_0, \vec{e}) \) with \( \vec{q}_0 = (q_{0p})_{p \in P} \).
System transitions:

\[(q, w) \xrightarrow{\text{AB!msg}} (q', w')\]

- \((q_A, \text{AB!msg}, q'_A) \in \delta_A\)
- \(\forall p \neq A.\ q'_p = q_p\)
- \(w'_A = w_A \cdot \text{msg}\) and \(\forall pr \neq \text{AB}.\ w'_pr = w_pr\)

Similarly for

\[(q, w) \xrightarrow{\text{AB?msg}} (q', w')\]
Synchronous communications

It is easy to equip CFSMs also with a synchronous communications.
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
Choreographies for CFSMs systems: Which description formalism?

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

Choreography Automata
Choreography Automata through an Example
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

\[
C \rightarrow S: \text{req} \\
S \rightarrow C: \text{res} \\
C \rightarrow S: \text{ok} \\
S \rightarrow L: \text{cnt} \\
C \rightarrow S: \text{ref} \\
S \rightarrow C: \text{res} \\
S \rightarrow C: \text{noRef} \\
C \rightarrow S: \text{bye} \\
S \rightarrow L: \text{bye} \\
L = \frac{19}{40}
\]
Projection

\[
\begin{align*}
C \rightarrow S &: \text{req} \\
S \rightarrow C &: \text{res} \\
S \rightarrow L &: \text{cnt} \\
C \rightarrow S &: \text{ok} \\
C \rightarrow S &: \text{ref} \\
S \rightarrow C &: \text{res} \\
S \rightarrow C &: \text{noRef} \\
S \rightarrow L &: \text{bye} \\
C \rightarrow S &: \text{bye} \\
L &= \frac{19}{40}
\end{align*}
\]
Projection
Projection

\[
\text{SL?cnt} \rightarrow \text{SL?bye}
\]
Projection

\[
\begin{align*}
C \rightarrow S &: \text{req} \\
S \rightarrow C &: \text{res} \\
S \rightarrow L &: \text{cnt} \\
C \rightarrow S &: \text{ref} \\
C \rightarrow S &: \text{ok} \\
S \rightarrow C &: \text{noRef} \\
C \rightarrow S &: \text{bye} \\
S \rightarrow L &: \text{bye} \\
S \rightarrow C &: \text{res}
\end{align*}
\]
Projection

\[
\begin{align*}
&C \to S: \text{req} \quad S \to C: \text{res} \quad S \to L: \text{cnt} \\
&C \to S: \text{ok} \\
&S \to C: \text{noRef} \\
&C \to S: \text{ref} \\
&S \to C: \text{res} \\
&S \to L: \text{bye} \\
&C \to S: \text{bye}
\end{align*}
\]
The behaviour of the system of CFSMs perfectly match the overall behaviour described by the choreography automata:

- The system is **Live**, i.e. if a machine is willing to perform some actions, the system can evolve so that one eventually is done.

- The system is **Deadlock-Free** i.e. it will never get stuck (the system does 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 progress).
The behaviour of the system of CFSMs perfectly match the overall behaviour described by the choreography automata:

- The system is **Live**, i.e. if a machine is willing to perform some actions, the system can evolve so that one eventually is done.
- The system is **Deadlock-Free** i.e. it will never get stuck (the system does 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 progress)
The behaviour of the system of CFSMs perfectly match the overall behaviour described by the choreography automata:

- **The system is Live**, i.e. if a machine is willing to perform some actions, the system can evolve so that one eventually is done.
- **The system is Deadlock-Free** i.e. it will never get stuck (the system does 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 progress).
Projection

The behaviour of the system of CFSMs perfectly match the overall behaviour described by the choreography automata:

- The system is **Live**, *i.e.* *if a machine is willing to perform some actions, the system can evolve so that one eventually is done*
- The system is **Deadlock-Free** *i.e.* *it will never get stuck (the system does 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 progress)*
The behaviour of the system of CFSMs perfectly match the overall behaviour described by the choreography automata:

- The system is **Live**, *i.e.* *if a machine is willing to perform some actions, the system can evolve so that one eventually is done*

- The system is **Deadlock-Free** *i.e.* *it will never get stuck (the system does 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 progress)*
The behaviour of the system of CFSMs perfectly match the overall behaviour described by the choreography automata:

- The system is **Live**, i.e. *if a machine is willing to perform some actions, the system can evolve so that one eventually is done*
- The system is **Deadlock-Free** i.e. *it will never get stuck (the system does 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 progress)*
The behaviour of the system of CFSMs perfectly match the overall behaviour described by the choreography automata:

- The system is **Live**, *i.e. if a machine is willing to perform some actions, the system can evolve so that one eventually is done*

- The system is **Deadlock-Free** *i.e. it will never get stuck (the system does 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 progress)*
The behaviour of the system of CFSMs perfectly match the overall behaviour described by the choreography automata:

- The system is **Live**, *i.e.* if a machine is willing to perform some actions, the system can evolve so that one eventually is done.
- The system is **Deadlock-Free** *i.e.* it will never get stuck (the system does 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 progress).
Projection

Both for **Synchronous** and **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, \((CA|_A)_A \in \mathcal{P}\), 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 participants are eventually made aware of the choices made;
- parallelism of independent interactions must be made explicit by interleaving them.
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, \((CA|_A)_{A \in \mathcal{P}}\), 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 participants are eventually made aware of the choices made;
- parallelism of independent interactions must be made explicit by interleaving them.
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, \((CA|_A)_{A \in \mathcal{P}}\), 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 participants are eventually made aware of the choices made;
- parallelism of independent interactions must be made explicit by interleaving them.
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, 
\((CA|_A)_A\in\mathcal{P}\), 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 participants are eventually made aware of the choices made;
- parallelism of independent interactions must be made explicit by interleaving them.
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, \((CA|_A)_{A \in P}\), 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 participants are eventually made aware of the choices made;
▶ parallelism of independent interactions must be made explicit by interleaving them
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, \((CA|_A)_{A \in \mathcal{P}}\), 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 participants are eventually made aware of the choices made;
- parallelism of independent interactions must be made explicit by interleaving them.
A promising future development

Usually choreographic models are good for the description of closed systems. What about open systems? The “participants as interfaces” approach to choreography for open systems.

One of the main motivations to develop a choreography model based on automata was to have a formalism enabling to internally describe a composition mechanism of global specifications (preserving well-formedness)
A promising future development

Usually choreographic models are good for the description of closed systems. What about open systems? The “participants as interfaces” approach to choreography for open systems.

One of the main motivations to develop a choreography model based on automata was to have a formalism enabling to internally describe a composition mechanism of global specifications (preserving well-formedness)
A promising future development

Usually choreographic models are good for the description of \textit{closed} systems. What about \textit{open} systems? The “participants as interfaces” approach to choreography for open systems.

One of the main motivations to develop a choreography model based on automata was to have a formalism enabling to internally describe a composition mechanism of global specifications (preserving well-formedness)
A promising future development

Usually choreographic models are good for the description of closed systems. What about open systems? The “participants as interfaces” approach to choreography for open systems.

One of the main motivations to develop a choreography model based on automata was to have a formalism enabling to internally describe a composition mechanism of global specifications (preserving well-formedness)
A promising future development

Usually choreographic models are good for the description of closed systems. What about open systems? The “participants as interfaces” approach to choreography for open systems.

One of the main motivations to develop a choreography model based on automata was to have a formalism enabling to internally describe a composition mechanism of global specifications (preserving well-formedness)