Automata for choreographies

Emilio Tuosto

Department of Computer Science

DI/NOVA LINCS

July 27, 2022

Take-home message

- Choreographies for communicating systems
- Two automata-based models
  - point-to-point communications
  - event-notification coordination
- An emerging pattern
  - fix a communication model
  - find suitable global and local specs
  - define well-formedness
  - get correct realisations by projection

– Act I –

Formal Choreographies, informally

(joint work with Roberto Guanciale)

Top-down model-driven development

Choreography = Global spec + Local spec

Quoting W3C:

"[...] a contract [...] of the common ordering conditions and constraints under which messages are exchanged [...] from a global viewpoint [...]
Each party can then use the global definition to build and test solutions [...]
global specification is in turn realised by combination of the resulting local systems"

Top-down model-driven development

Choreography = Global spec + Local spec

Quoting W3C:

"[...] a contract [...] of the common ordering conditions and constraints under which messages are exchanged [...] from a global viewpoint [...]
Each party can then use the global definition to build and test solutions [...]
global specification is in turn realised by combination of the resulting local systems"

Choreography G
global viewpoint

well-formedness

M_i
Local viewpoint$_1$

M_1
Local viewpoint$_1$

M_n
Local viewpoint$_n$

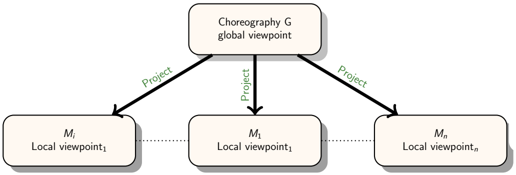
specs,not code

Top-down model-driven development

Choreography = Global spec + Local spec

Quoting W3C:

"[...] a contract [...] of the common ordering conditions and constraints under which messages are exchanged [...] from a global viewpoint [...]
Each party can then use the global definition to build and test solutions [...]
global specification is in turn realised by combination of the resulting local systems"



well-formedness

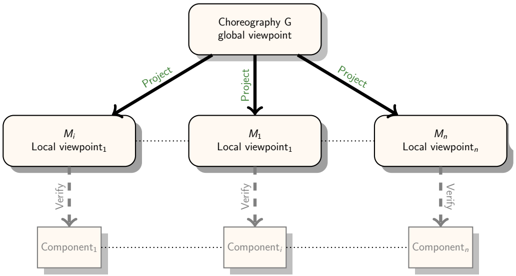
specs,not code

Top-down model-driven development

Choreography = Global spec + Local spec

Quoting W3C:

"[...] a contract *[...] of the common* ordering conditions and constraints *under which* messages *are exchanged [...] from a* global viewpoint *[...]*
Each party can then use the global definition to build and test solutions [...]
global specification is in turn realised by combination of the resulting local systems"



well-formedness

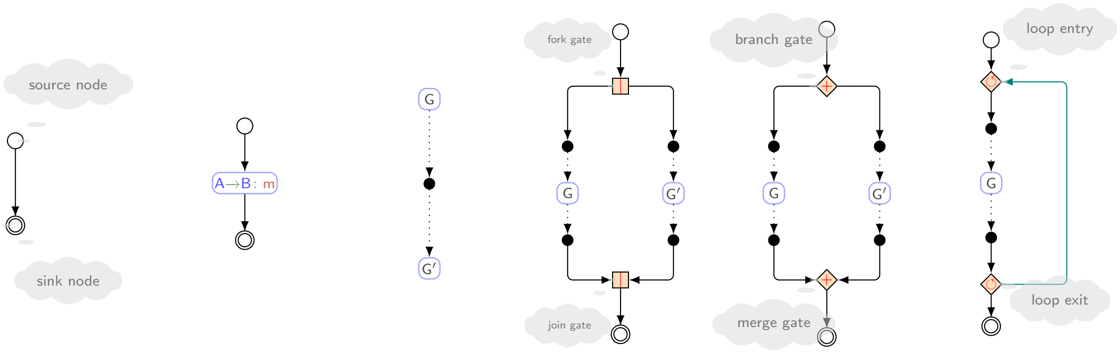
specs,not code

A model of global specs

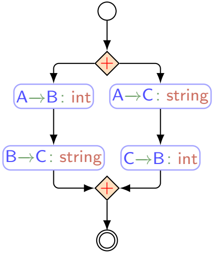
$$
\begin{array}{lcll}
G,\ G' & ::= & (o) & \text{empty} \\
& | & A \to B : m & \text{interaction} \\
& | & G ; G' & \text{sequential} \\
& | & G \mid G' & \text{parallel} \\
& | & G + G' & \text{branch} \\
& | & G^{\star} & \text{iteration}
\end{array}
$$

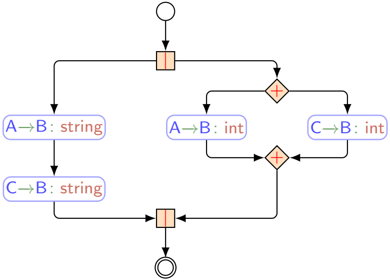
A model of global specs

$G,\ G'$	$::=$	(o)	empty
	$\mid$	$A \to B : m$	interaction
	$\mid$	$G; G'$	sequential
	$\mid$	$G \mid G'$	parallel
	$\mid$	$G + G'$	branch
	$\mid$	$G^\star$	iteration



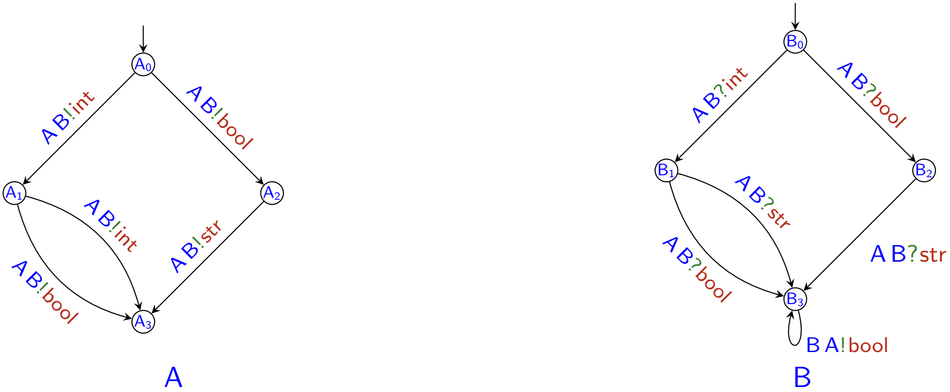
Some examples





A model of local specs

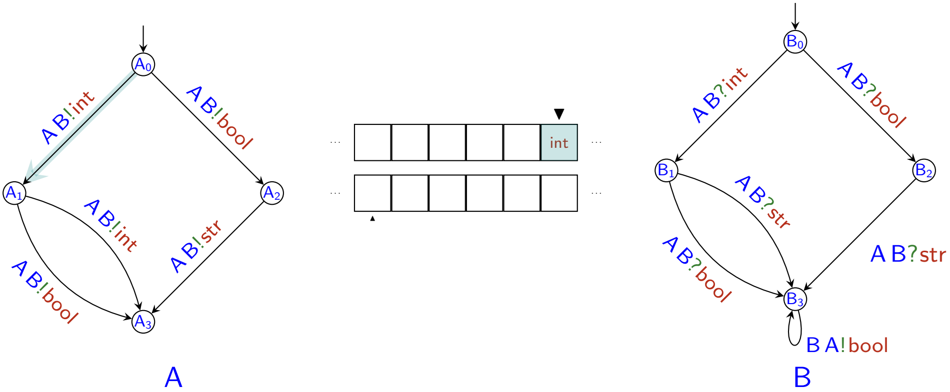
Communicating Finite-State Machines [Brand&Zafiropulo 1983]



Global specs can be projected (i.e., compiled) on CFSMs

A model of local specs

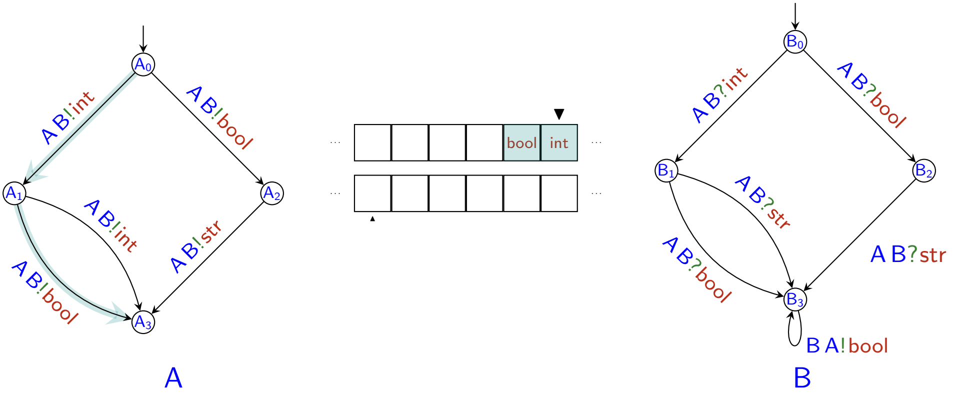
Communicating Finite-State Machines [Brand&Zafiropulo 1983]



Global specs can be projected (i.e., compiled) on CFSMs

A model of local specs

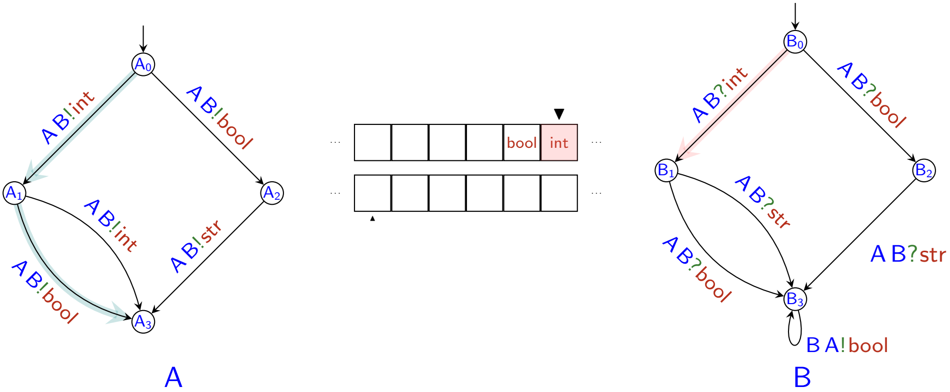
Communicating Finite-State Machines [Brand&Zafiropulo 1983]



Global specs can be projected (i.e., compiled) on CFSMs

A model of local specs

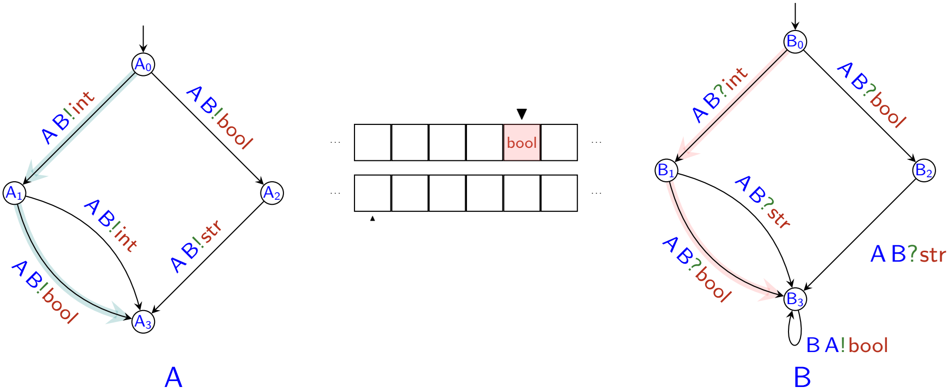
Communicating Finite-State Machines [Brand&Zafiropulo 1983]



Global specs can be projected (i.e., compiled) on CFSMs

A model of local specs

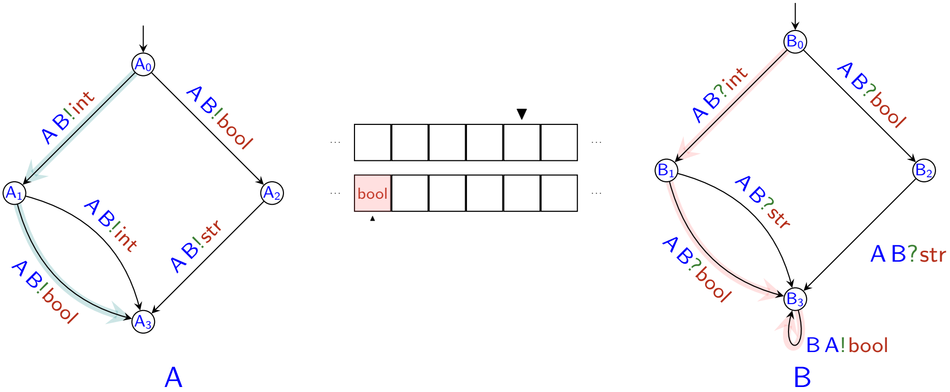
Communicating Finite-State Machines [Brand&Zafiropulo 1983]



Global specs can be projected (i.e., compiled) on CFSMs

A model of local specs

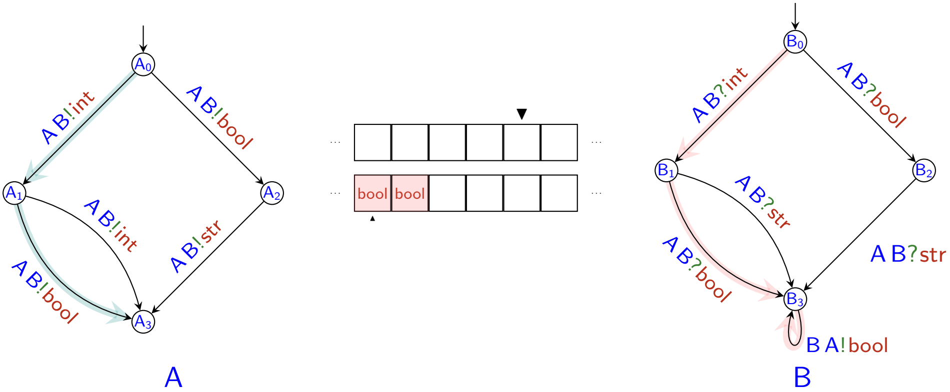
Communicating Finite-State Machines [Brand&Zafiropulo 1983]



Global specs can be projected (i.e., compiled) on CFSMs

A model of local specs

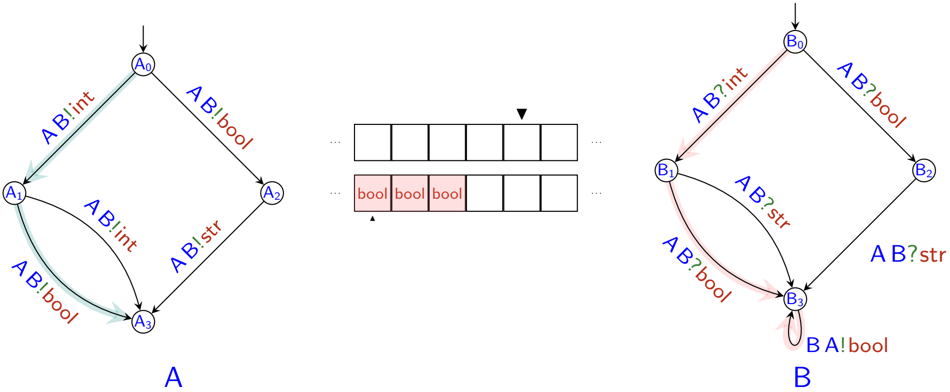
Communicating Finite-State Machines [Brand&Zafiropulo 1983]



Global specs can be projected (i.e., compiled) on CFSMs

A model of local specs

Communicating Finite-State Machines [Brand&Zafiropulo 1983]



Global specs can be projected (i.e., compiled) on CFSMs

An obvious (fundamental) question

Given a global specification, is it realisable distributively?

An obvious (fundamental) question

Given a global specification, is it realisable distributively?

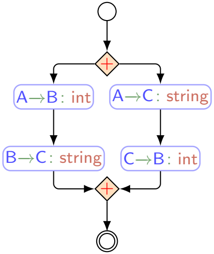
Put simply...

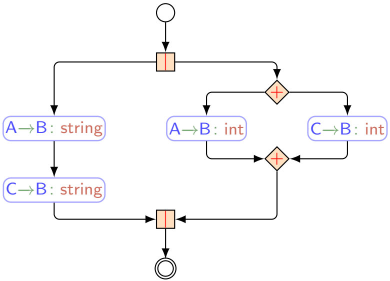
A global spec G is realizable if there is a deadlock-free[a] system of CFSMs whose traces "have some relation with" G.

[a] A system S is *deadlock-free* if none of its reachable configurations s is a deadlock, that is $s \not\rightarrow$ and either some buffers are not empty or some CFSMs have transitions from their state in s.

Class test

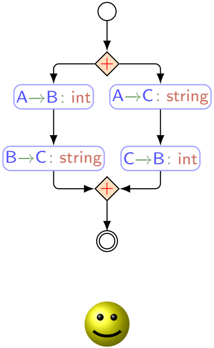
Revisiting our examples

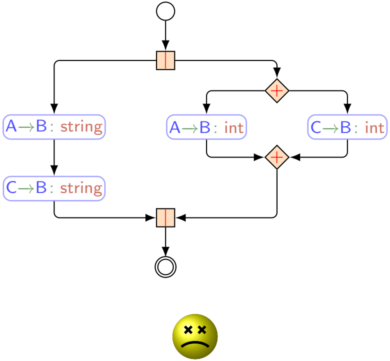




Class test

Revisiting our examples





A (main) source of problems: Well-branchedness

Distributed consensus

A distributed choice $G_1 + G_2 + \cdots$ is well-branched if

- there is one active participant
- any non-active participant is passive

A (main) source of problems: Well-branchedness

Distributed consensus

A distributed choice $G_1 + G_2 + \cdots$ is well-branched if

- there is one active participant
- any non-active participant is passive

Def. A is active when it locally decides which branch to take in a choice

Def. B is passive when

- either B behaves uniformly in each branch
- or B "unambiguously understands" which branch A opted for from some inputs

A (main) source of problems: Well-branchedness

Distributed consensus

A distributed choice $G_1 + G_2 + \cdots$ is well-branched if

- there is one active participant
- any non-active participant is passive

Def. A is active when it locally decides which branch to take in a choice

Def. B is passive when

- either B behaves uniformly in each branch
- or B "unambiguously understands" which branch A opted for from some inputs

Well-branchedness

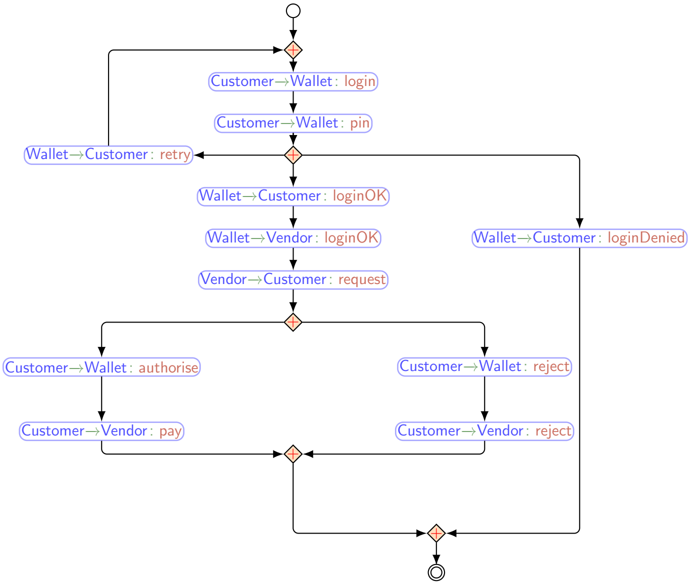
When the above holds true for each choice, the choreography is well-branched. This enables correctness-by-design.

– Act II –

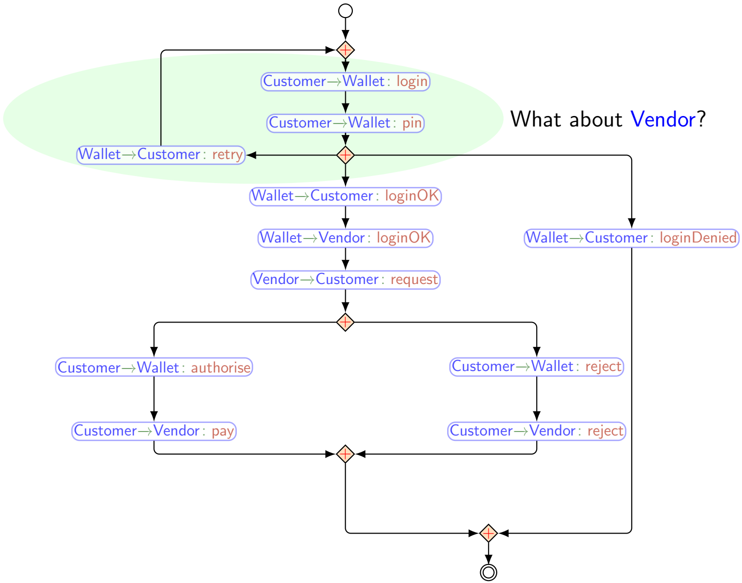
Choreography Automata

(joint work with Franco Barbanera, Ivan Lanese)

The online-wallet protocol

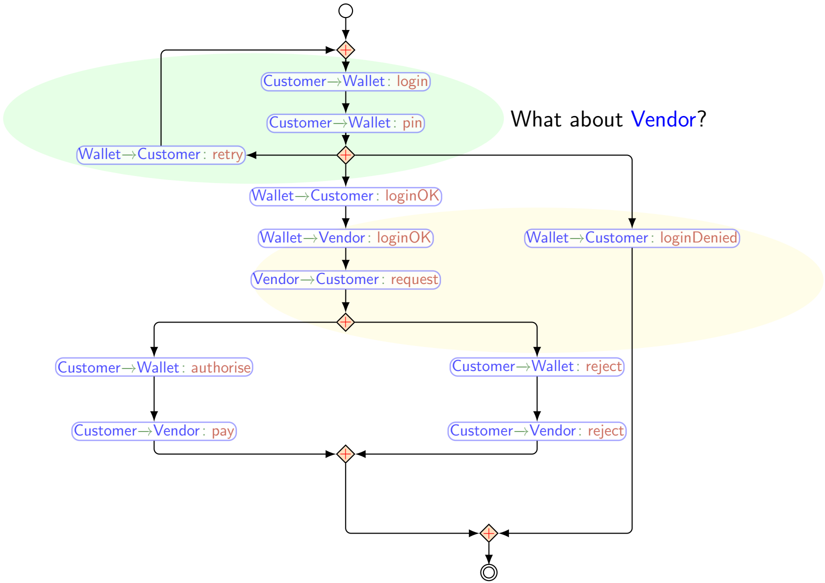


The online-wallet protocol ...some modelling problems



What about Vendor?

The online-wallet protocol ...some modelling problems

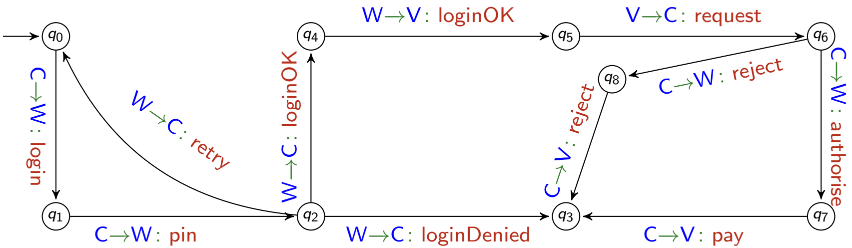


What about Vendor?

What about payloads?

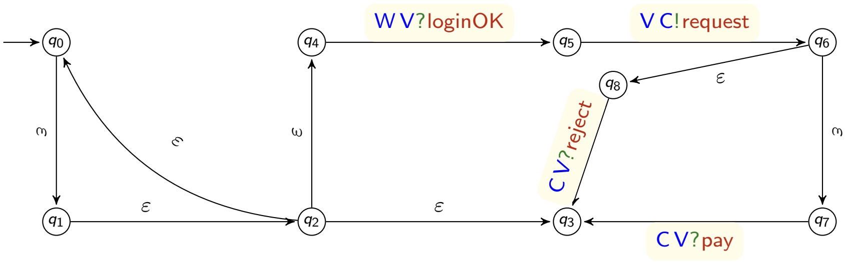
Our global & local specs

Choreography automata: Interaction, globally



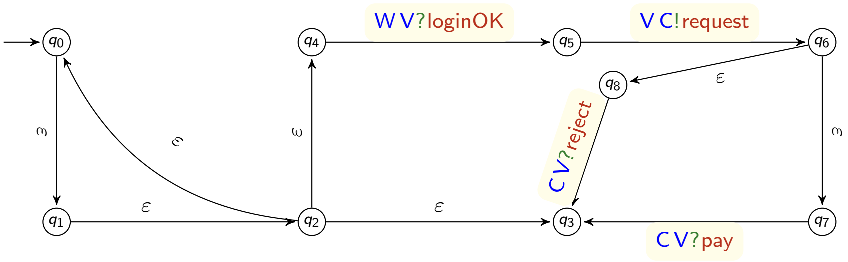
Our global & local specs

Intermediate automata: from interactions to communications

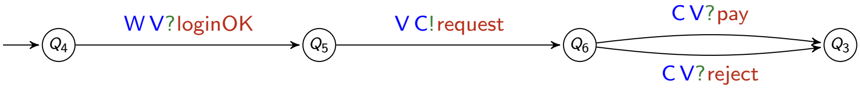


Our global & local specs

Intermediate automata: from interactions to communications



CFSMs locally: determinise the intermediate automaton



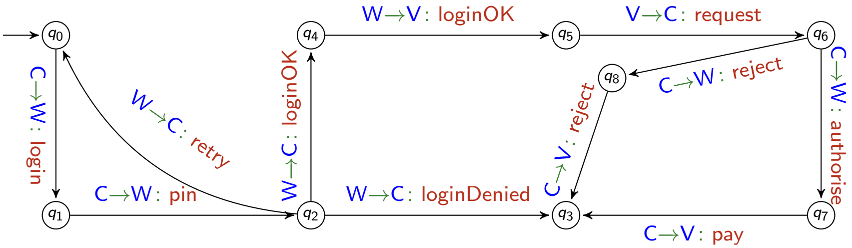
Projections preserve semantics

Theorem. Choreography automata are bisimilar to their projections

$\Longrightarrow$ traces equivalence

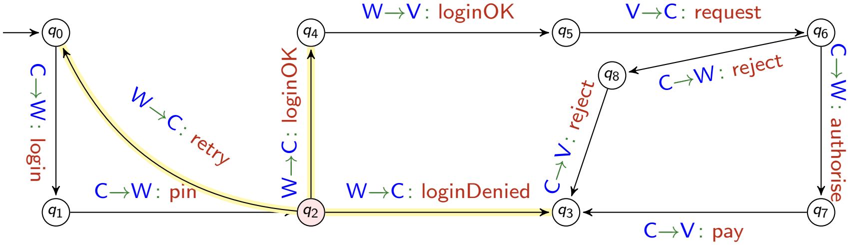
Flexibility by example

Selective participation in OLW



Flexibility by example

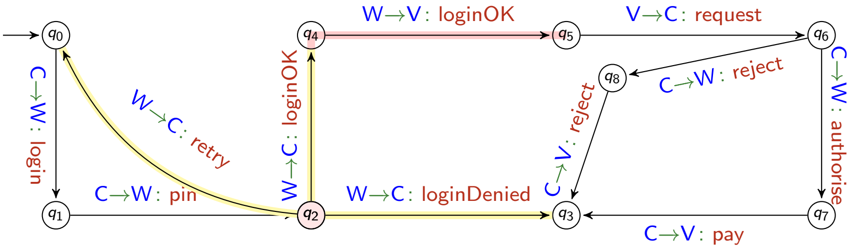
Selective participation in OLW



- at q_2 Wallet and Customer aware from the very beginning

Flexibility by example

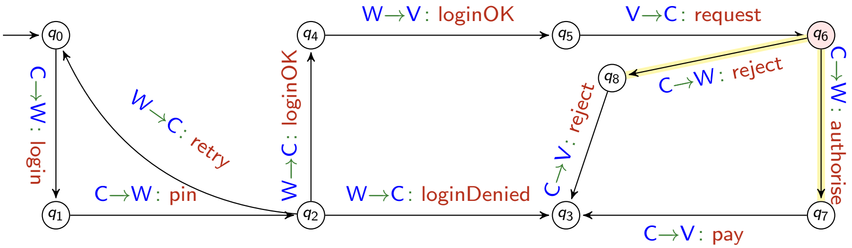
Selective participation in OLW



- at q_2 Wallet and Customer aware from the very beginning
  - Vendor involved on one branch only, but that's fine: Wallet is aware

Flexibility by example

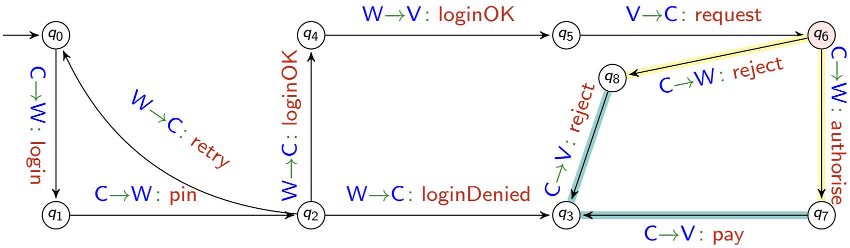
Selective participation in OLW



- at q_2 Wallet and Customer aware from the very beginning
  - Vendor involved on one branch only, but that's fine: Wallet is aware
- at q_6 Wallet and Customer aware from the very beginning

Flexibility by example

Selective participation in OLW



- at q_2 Wallet and Customer aware from the very beginning
  - Vendor involved on one branch only, but that's fine: Wallet is aware
- at q_6 Wallet and Customer aware from the very beginning
  - Vendor eventually informed by Customer on each branch

Theorems

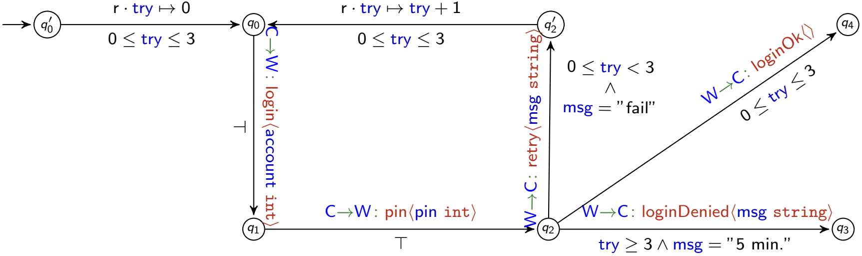
Correctness by construction

Theorem. Projections of well-formed choreography automata are deadlock-free

Theorem. Projections of well-formed choreography automata are lock-free

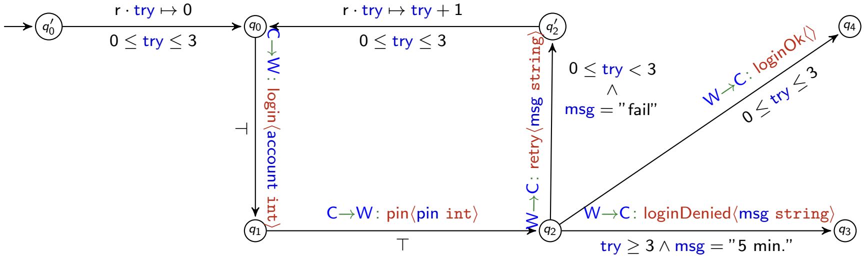
DbC vs. choreography automata

Asserting (an excerpt of) OLW



DbC vs. choreography automata

Asserting (an excerpt of) OLW



Consistency

- history senesitiveness: in $q \xrightarrow[A]{\lambda} q'$, A predicates on *known* variables
- temporal satisfiability: the conjunction of the predicates on a path is satisfiable
- well-formedness of the underlying choreography automaton

Theorems

Projections are a bit more complicated than for choreography automata

On consistent asserted choreography automata

Theorem. Asserted choreography automata are weakly bisimilar to their projections

$\Longrightarrow$ trace equivalence

Theorem. Projections of WF choreography automata are deadlock-free

Theorems

Projections are a bit more complicated than for choreography automata

On consistent asserted choreography automata

Theorem. Asserted choreography automata are weakly bisimilar to their projections

$\Longrightarrow$ trace equivalence

Theorem. Projections of WF choreography automata are deadlock-free

And more...cf. [ECOOP 2022]

A tool chain for

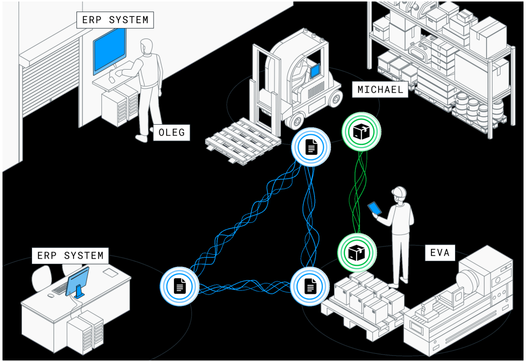
- validating finitary Scribble protocols via choreography automata
- TypeScript web programming via API generation

– Act III –

Local-first!

(joint work with Daniela Marottoli, Hernán Melgratti, Roland Kuhn)

A completely different setting

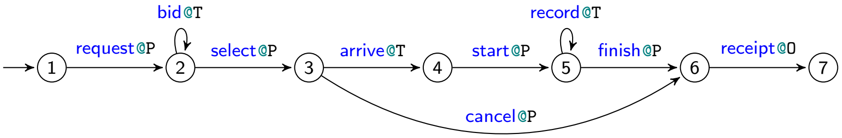


Desiderata

- different features
  - arbitrary (and variable) number of instances
  - local-first principle!
    - As rock climbers say: "Don't Be Afraid To Fail. Be Afraid Not To Try."
  - pub-sub (instead of point-to-point)
- different properties
  - progress despite unavailability
    $\Longrightarrow$ inconsistent views
  - eventual-consistency instead of "old" properties (eg. session fidelity)

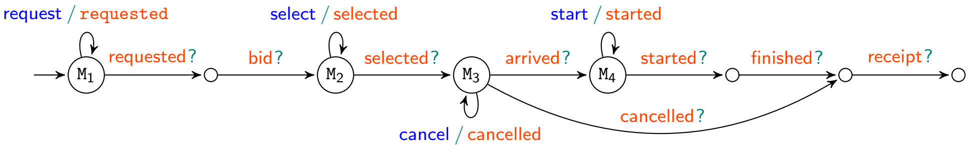
Swarm protocols and machines by example

Global



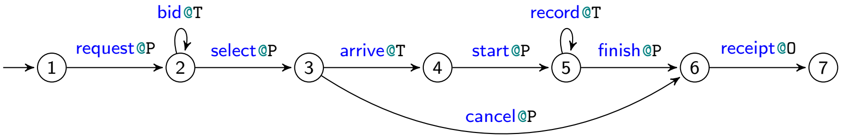
(log types omitted for readability)

Local (projected)



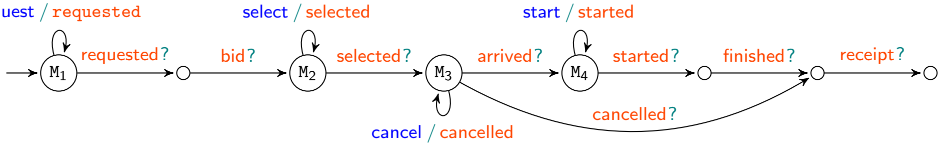
Swarm protocols and machines by example

Global



(log types omitted for readability)

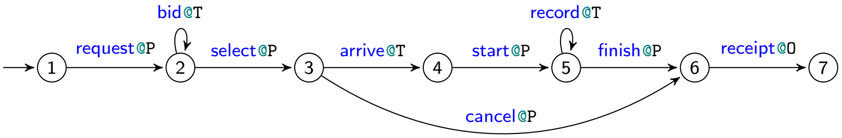
Local (projected)



local log: $r_1 \cdot r_2 \cdot b$

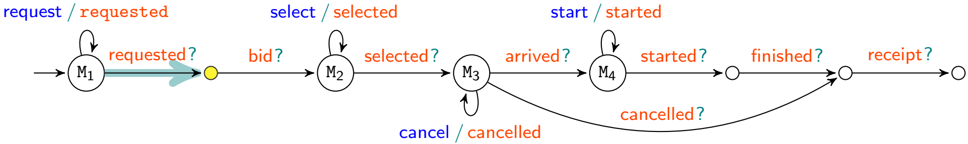
Swarm protocols and machines by example

Global



(log types omitted for readability)

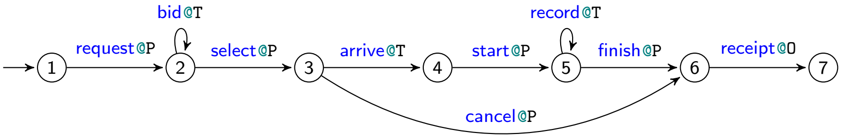
Local (projected)



local log: $\cdot\ r_2 \cdot b$

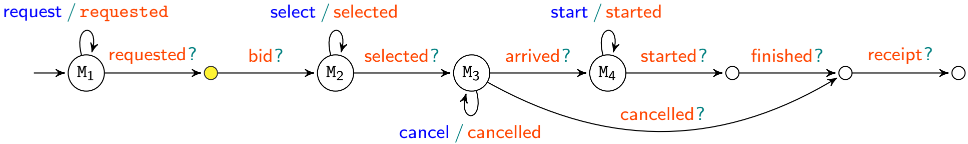
Swarm protocols and machines by example

Global



(log types omitted for readability)

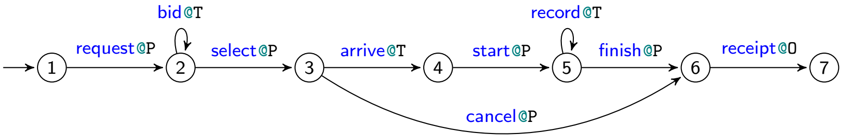
Local (projected)



local log: · · *b*

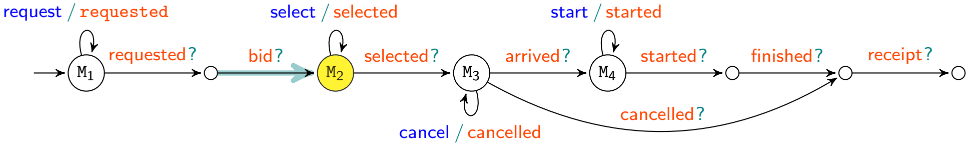
Swarm protocols and machines by example

Global



(log types omitted for readability)

Local (projected)



local log: · · and now select is enabled

Semantics, intuitively

- Types "produce/consume" events
  - swarm protocols: how/when roles produce events
  - machines: how/when instances consume events "skipping" the ones irrelevant to them
- Deterministic types only
  - swarm protocols: log types of branches have no common non-trivial prefixes and command/role pairs are pairwise distinct
  - machines: event types of branches are pairwise distinct
- Non-deterministic events' propagation

Swarms

Machines, local logs, and global log (...a mirage)

Events are univocally associated to the machines generating them.
Def. swarm = global log + map from unique identities to pairs machines/local logs

$$(\mathtt{S}, l) = (\mathtt{M}_1, l_1) \mid \ldots \mid (\mathtt{M}_n, l_n) \mid l$$

such that $l_i \sqsubseteq l$ where, $\quad l_i \sqsubseteq l \iff l_i = \begin{matrix} e_{i,1} \\ \vdots \\ e_{i,n} \end{matrix} \qquad \begin{matrix} e_1 \\ \\ \vdots \\ \\ e_m \end{matrix} = l$

i.e., there is an order-preserving and downward-total morphism from l_i into l on events of a same machine.

Swarms

Machines, local logs, and global log (...a mirage)

Events are univocally associated to the machines generating them.

Def. swarm = global log + map from unique identities to pairs machines/local logs

$$(\mathtt{S}, l) = (\mathtt{M}_1, l_1) \mid \ldots \mid (\mathtt{M}_n, l_n) \mid l$$

such that $l_i \sqsubseteq l$ where,

$$l_i \sqsubseteq l \iff l_i = \begin{tikzcd}[row sep=small] & e_1 \\ e_{i,1} \arrow[r] & {} \\ \vdots & \vdots \\ e_{i,n} \arrow[r] & {} \\ & e_m \end{tikzcd} = l$$

i.e., there is an order-preserving and downward-total morphism from l_i into l on events of a same machine.

Machines, local logs, and global log (...a mirage)

Events are univocally associated to the machines generating them.

Def. swarm = global log + map from unique identities to pairs machines/local logs

$$(\mathtt{S}, l) = (\mathtt{M}_1, l_1) \mid \ldots \mid (\mathtt{M}_n, l_n) \mid l$$

such that $l_i \sqsubseteq l$ where,

$$l_i \sqsubseteq l \iff l_i = \begin{array}{c} e_{i,1} \\ \vdots \\ e_{i,n} \end{array} \qquad \begin{array}{c} e_1 \\ \vdots \\ \boxed{e} \\ e_m \end{array} = l$$

i.e., there is an order-preserving and downward-total morphism from l_i into l on events of a same machine.

Swarms

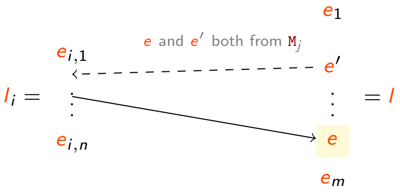
Machines, local logs, and global log (...a mirage)

Events are univocally associated to the machines generating them.

Def. swarm = global log + map from unique identities to pairs machines/local logs

$$(\mathtt{S}, l) = (\mathtt{M}_1, l_1) \mid \ldots \mid (\mathtt{M}_n, l_n) \mid l$$

such that $l_i \sqsubseteq l$ where, $\quad l_i \sqsubseteq l \iff$



i.e., there is an order-preserving and downward-total morphism from l_i into l on events of a same machine.

Swarms' semantics...intuitively

- Events' generation
  The local log of a machine is extended with the fresh events generated by (the execution of a command on) the machine
- Events' propagation
  Emitted events propagate asynchronously & non-deterministically

Swarms' semantics: formally

[Local]

$$\frac{\mathtt{S} : i \mapsto (\mathtt{M}, l) \qquad (\mathtt{M}, l) \xrightarrow{\mathtt{c}\,/\,\mathtt{l}} (\mathtt{M}, l') \qquad l'' \in l \bowtie \hat{l}}{(\mathtt{S}, \hat{l}) \xrightarrow{\mathtt{c}\,/\,\mathtt{l}} (\mathtt{S}[i \mapsto (\mathtt{M}, l')], l'')}$$

where $\qquad l_1 \bowtie l_2 = \{ l \mid l \subseteq l_1 \cup l_2 \ \wedge \ l_1 \sqsubseteq l \ \wedge \ l_2 \sqsubseteq l \}$

[Prop]

$$\frac{\mathtt{S} : i \mapsto (\mathtt{M}, l) \qquad l \sqsubseteq l' \sqsubseteq \hat{l} \qquad l \subset l'}{(\mathtt{S}, \hat{l}) \xrightarrow{\tau} (\mathtt{S}[i \mapsto (\mathtt{M}, l')], \hat{l})}$$

Properties of our semantics

Coherence

A swarm $(\mathtt{M}_1, I_1) \mid \ldots \mid (\mathtt{M}_n, I_n) \mid I$ is coherent if

$$\text{for all } i,\ I_i \sqsubseteq I \qquad \text{and} \qquad I = \bigcup_{i \in \underline{n}} I_i$$

Coherence preservation

[LOCAL] & [PROP] preserve coherence

Eventual Consistency

If

$$\mathtt{S} = (\mathtt{M}_1, I_1) \mid \ldots \mid (\mathtt{M}_n, I_n) \mid I \text{ is coherent}$$

then

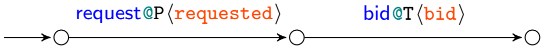
$$\mathtt{S} \xrightarrow{\tau}^{\star} (\mathtt{M}_1, I) \mid \ldots \mid (\mathtt{M}_n, I) \mid I$$

Realisation

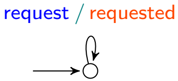
It is hard to get it right (even without multi-instances or choices!)

A trivial protocol

Take the swarm protocol



Are



and

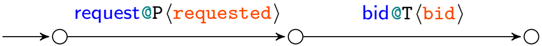
a realisation?

Realisation

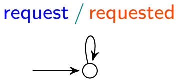
It is hard to get it right (even without multi-instances or choices!)

A trivial protocol

Take the swarm protocol



Are



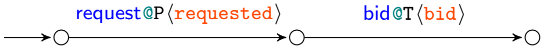
and

a (correct) realisation?

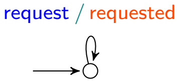
It is hard to get it right (even without multi-instances or choices!)

A trivial protocol

Take the swarm protocol



Are



and

a (correct) realisation? What does that actually mean?

Ideas

Not so simple

A swarm correctly realises a swarm protocol if it generates only logs that the protocol can generate.

That's impossible due to events' skipping at local level but not at the global one.

Ideas

Not so simple

A swarm correctly realises a swarm protocol if it generates only logs that the protocol can generate.

That's impossible due to events' skipping at local level but not at the global one.

A weaker condition

A swarm correctly realises a swarm protocol if it generates only logs that are admissible with some that the protocol can generate.

A log is admissible for a swarm protocol when its restriction to the events processed by the active machines is equivalent to a log of the protocol.

Realisation by projection

Well-formedness of swarm protocols

Each log type `l` of a branch should be

- causal consistent
  - each selector in (the continuation of) `l` reacts to `l`
  - each role involved in the continuation of `l` cannot react to more events on `l` than selectors on the branch
- determined
  - each role in the continuation of `l` reacts to `l`[0]
- confusion-free
  - an event type cannot occur in more than one branch

– Epilogue –

[. . .]

Summing up

Automata models for choreography

Advantages

- increased flexibility
- good basis for (enhanced) tool support
- good also for practitioners

Plans

- weakening well-formedness conditions
- studying more complex communication models (eg non-atomic propagation of events)

Thank you!