A model of replicated asymmetric state machines

Roland Kuhn @ Actyx
Daniela Marottoli @ UBA  Hernán Melgratti @ UBA
Emilio Tuosto @ GSSI

School of Informatics, Leicester

October 22, 2021

Research partly supported by the EU H2020 RISE programme under the Marie Skłodowska-Curie grant agreement No 778233
Take-away message

Behavioural specs for the (existing) Actyx industrial platform to develop applications for factory automation
Take-away message

Behavioural specs for the (existing) Actyx industrial platform to develop applications for factory automation

With respect to the state-of-the-art, our models

- feature
  - pub-subscribe (instead of point-to-point)
  - generalised choices
  - arbitrary (and variable) number of instances
Take-away message

Behavioural specs for the (existing) Actyx industrial platform to develop applications for factory automation

With respect to the state-of-the-art, our models

- feature
  - pub-subscribe (instead of point-to-point)
  - generalised choices
  - arbitrary (and variable) number of instances

- focus on
  - availability (instead of consensus)
  - eventual-consistency (instead of eg. message loss, deadlock, session fidelity, etc.)
Plan

- Challenges in factory automation
- Our behavioural specifications
- Model-driven realisation
- Concluding remarks
– Prelude –

[ Factory automation ]
A highly collaborative environment

People + Real-time controllers + IT systems and networks:

- work divided among many autonomous production cells
- efficiency is determined by designing and controlling the flow of resource and information
- local failures must be tolerated for brief time periods
Industrial scenarios

A highly collaborative environment
People + Real-time controllers + IT systems and networks:
work divided among many autonomous production cells
efficiency is determined by designing and controlling the
flow of resource and information

local failures must be tolerated for brief time periods

Execution model

\[
\text{machine/operator/forklift/...} \rightarrow \text{Local twin (state machine)}
\]

- twins are replicated where needed
- events have unique IDs and
  - record facts (e.g., from sensors) or
  - decisions (e.g., from an operator)
  - spread asynchronously
- events are kept in local logs used to compute the state
  of the twin
- logs are replicated and merged
Challenges

Specify application-level protocols where decisions

- don’t require consensus
Specify application-level protocols where decisions
- don’t require consensus
- are based on stale local states
Challenges

Specify application-level protocols where decisions

- don’t require **consensus**
- are based on **stale local states**
- yet, **collaboration** has to be successful
– Behavioural Specifications –

[ Decide locally, agree globally...eventually ]
Syntax

Global specs

\[ G ::= c_1 @ R_1 \langle l_1 \rangle \cdot G_1 + \cdots + c_n @ R_n \langle l_n \rangle \cdot G_n \]

\[ G = \text{publish} @ A \langle p \rangle \cdot G' \]
\[ G' = \text{bid} @ B \langle b \rangle \cdot G' \]
\[ + \]
\[ \text{select} @ A \langle s \rangle \cdot \text{finish} @ A \langle f \rangle \cdot 0 \]

Local Specs

\[ M ::= \kappa \cdot [t_1 ? M_1 \& \cdots \& t_n ? M_n] \]

\[ M_A = \{\text{publish} / p\} \cdot [p ? M_A'] \]
\[ M_A' = \{\text{select} / s\} \cdot [b ? M_A' \& s ? \{\text{finish} / f\} \cdot f ? 0] \]
\[ M_B = p ? M_B' \]
\[ M_B' = \{\text{bid} / b\} \cdot [b ? M_B' \& s ? f ? 0] \]
Syntax

Global specs

\[ G ::= c_1 \circ R_1 \langle l_1 \rangle . G_1 + \cdots + c_n \circ R_n \langle l_n \rangle . G_n \]

\[ G = \text{publish} \circ A \langle p \rangle . G' \]
\[ G' = \text{bid} \circ B \langle b \rangle . G' \]
\[ + \text{select} \circ A \langle s \rangle . \text{finish} \circ A \langle f \rangle . 0 \]

Local Specs

\[ M ::= \kappa \cdot \begin{bmatrix} t_1 \cdot M_1 & \cdots & t_n \cdot M_n \end{bmatrix} \]

\[ M_A = \{ \text{publish} / p \} \cdot \begin{bmatrix} p \cdot M_A' \end{bmatrix} \]
\[ M_A' = \{ \text{select} / s \} \cdot \begin{bmatrix} b \cdot M_A' & s \cdot \text{finish} / f \cdot f \cdot 0 \end{bmatrix} \]
\[ M_B = p \cdot M_B' \]
\[ M_B' = \{ \text{bid} / b \} \cdot \begin{bmatrix} b \cdot M_B' & s \cdot f \cdot 0 \end{bmatrix} \]
Semantics of specifications

### Global (protocol spec)

$$\delta(G, l) \xrightarrow{c/l} G' \quad \vdash l': l \quad l' \text{ fresh} \quad \frac{\vdash l': l}{(G, l) \xrightarrow{c/l} (G, l \cdot l')}$$

where

$l$ is an (idealised) global/shared log

$$\cdots + c_i @ R_i(l_i). G_i + \cdots \xrightarrow{c_i/l_i} G_i$$

$$\delta(G, \epsilon) = G$$

$$\delta(G, l) = \begin{cases} \delta(G', l \cdot l') & \text{if } G \xrightarrow{c/l} G', l \neq \epsilon, \vdash l': l \\ \bot & \text{otherwise} \end{cases}$$

### Local (components' spec)

$$\delta(M, l) = M' \quad M' \xleftarrow{c/l} \vdash l': l \quad l' \text{ fresh}$$

where

$l$ is the local log accessible to $M$

$$M' \xleftarrow{c/l} \iff c/l \text{ enabled at } M'$$

$$\delta(M, \epsilon) = M$$

$$\delta(M, e \cdot l) = \begin{cases} \delta(M_j, l) & \text{if } \vdash e : t_j, \\ M = \kappa \cdot \ldots & t_j ? M_j & \ldots & \ldots \end{cases}$$
Syntax of systems

Events are univocally associated to the machines generating them.

Machines with local logs & a global one

\[ S = (M_1, l_1) | \ldots | (M_n, l_n) | l \]

(Recall: the global log is an optical illusion)

Well-formedness

\[ (M_1, l_1) | \ldots | (M_n, l_n) | l \] is well-formed if

\[ l = l_1 \cup \ldots \cup l_n \]

and for all \( i \), \( l_i \subseteq l \)
Syntax of systems

Events are univocally associated to the machines generating them.

Machines with local logs & a global one

\[ S = (M_1, l_1) \mid \ldots \mid (M_n, l_n) \mid / \]

(Recall: the global log is an optical illusion)

Well-formedness

\((M_1, l_1) \mid \ldots \mid (M_n, l_n) \mid /\) is well-formed if

\[ l = l_1 \cup \ldots \cup l_n \quad \text{and} \quad \text{for all } i, \ l_i \subseteq l \]

where

\[ l_i \subseteq l \iff l_i = \ldots = l_i \]

\[ l_i = \ldots = l_i \]

\[ e_1 \]

\[ e_{i,1} \]

\[ \vdots \]

\[ e_{i,n} \]

\[ \vdots \]

\[ = l \]

\[ e_m \]
Events are univocally associated to the machines generating them.

**Machines with local logs & a global one**

\[ S = (M_1, l_1) \mid \ldots \mid (M_n, l_n) \mid / \]

(Recall: the global log is an optical illusion)

**Well-formedness**

\( (M_1, l_1) \mid \ldots \mid (M_n, l_n) \mid / \) is well-formed if

\[ / = l_1 \cup \ldots \cup l_n \quad \text{and} \quad \text{for all } i, \ l_i \sqsubseteq / \]

where

\[ l_i \sqsubseteq / \iff l_i = \ldots = l \]

formally, there is an order-preserving
Syntax of systems

Events are univocally associated to the machines generating them.

Machines with local logs & a global one

\[ S = (M_1, l_1) \mid \ldots \mid (M_n, l_n) \mid / \]

(Recall: the global log is an optical illusion)

Well-formedness

\( (M_1, /_1) \mid \ldots \mid (M_n, /_n) \mid / \) is well-formed if

\[ / = /_1 \cup \ldots \cup /_n \quad \text{and} \quad \text{for all } i, \; /_i \sqsubseteq / \]

where

\[ /_i \sqsubseteq / \iff /_i = \ldots = / \]

formally, there is an order-preserving and
Syntax of systems

Events are univocally associated to the machines generating them.

**Machines with local logs & a global one**

\[ S = (M_1, l_1) | \ldots | (M_n, l_n) | / \]

(Recall: the global log is an optical illusion)

**Well-formedness**

\[ (M_1, l_1) | \ldots | (M_n, l_n) | / \] is well-formed if

\[ l = l_1 \cup \ldots \cup l_n \]

and for all \( i \), \( l_i \sqsubseteq l \)

where

\[ l_i \sqsubseteq l \iff l_i = \{ e_{i,1}, \ldots, e_{i,n} \} \]

formally, there is an order-preserving and downward-total morphism from \( l_i \) into \( l \)
Specs describe how to “produce/consume” events

- **Global**: how/when roles produce events
- **Local**: how/when instances consume events
  - “skipping” irrelevant events
Linking semantics of specs & systems

- Specs describe how to “produce/consume” events
  - **Global**: how/when roles produce events
  - **Local**: how/when instances consume events
    - “skipping” irrelevant events
- Communication is **non-deterministic**

The semantics of systems consists of two phases:
Linking semantics of specs & systems

- Specs describe how to “produce/consume” events
  - **Global**: how/when roles produce events
  - **Local**: how/when instances consume events
    - “skipping” irrelevant events
- Communication is **non-deterministic**

The semantics of systems consists of two phases:

**First: Events’ generation**

The local log of a machine is extended with the fresh events generated by the machine.
Linking semantics of specs & systems

- Specs describe how to “produce/consume” events
  - **Global**: how/when roles produce events
  - **Local**: how/when instances consume events
    - “skipping” irrelevant events
- Communication is **non-deterministic**

The semantics of systems consists of two phases:

**First: Events’ generation**
The local log of a machine is extended with the fresh events generated by the machine

**Second: Events’ propagation**
Emitted events propagate asynchronously & non-deterministically
Semantics of systems: formally

**[Local]**

\[
i \in \text{dom } S \quad S(i) = (M, l_i) \quad (M, l_i) \xrightarrow{c/1} (M, l'_i) \quad l' \in l \otimes l'_i
\]

\[
(S, l) \xrightarrow{c/1} (S[i \mapsto (M, l'_i)], l')
\]

where

\[
l_1 \otimes l_2 = \{ l \mid l \subseteq l_1 \cup l_2 \land l_1 \sqsubseteq l \land l_2 \sqsubseteq l \}\]

**[Prop]**

\[
i \in \text{dom } S \quad S(i) = (M, l_i) \quad l_i \sqsubseteq l' \sqsubseteq l \quad l_i \subset l'
\]

\[
(S, l) \xrightarrow{\tau} (S[i \mapsto (M, l')], l)
\]
If eat_honey /ёёё, then, by Local...
Events generation

If eat_honey / 😊😊 then, by [LOCAL]
Events generation

If eat_honey /😊😊 then, by [LOCAL]

1

3

2

1

2

3

1

2

3

1
Events generation

If eat_honey /olución / então, by [LOCAL]
Events generation

If eat honey /😊😊 then, by [LOCAL]

1

2

3
Events generation

If eat honey /😊 😊 then, by [LOCAL]

1

3

2

1

3

2
Propagation

By rule [PROP]
By rule [PROP]
By rule [PROP]
Propagation

By rule [PROP]
Properties of our semantics

**Theorem: Well-Formedness preservation**

If

\[ S \text{ is well-formed and } S \xrightarrow{[\text{Local}]/[\text{Prop}]} S' \]

then

\[ S' \text{ is well-formed} \]

**Theorem: Eventual Consistency**

If

\[ S = (M_1, l_1) \mid \ldots \mid (M_n, l_n) \mid l \text{ is well-formed} \]

then

\[ S \xrightarrow{\tau^*} (M_1, l) \mid \ldots \mid (M_n, l) \mid l \]
– From models to implementations –

[ On realisation ]
Exercise on realisation

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

A trivial protocol

Take

\[ G = \text{prepare}_{\mathcal{P}}(\text{piece}).\text{pack}_{\mathcal{A}}(\text{product}).0 \]

Do the following machines realise \( G \)?

\[ M_P = \{ \text{prepare} / \text{piece} \}.0 \quad \text{and} \quad M_A = \text{piece}?\{ \text{pack} / \text{product} \}.0 \]
Exercise on realisation

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

A trivial protocol

Take

\[ G = \text{prepare}@P\langle \text{piece} \rangle \cdot \text{pack}@A\langle \text{product} \rangle \cdot 0 \]

Do the following machines realise \( G \)?

\[ M_P = \{ \text{prepare} / \text{piece} \} \cdot 0 \quad \text{and} \quad M_A = \text{piece}? \{ \text{pack} / \text{product} \} \cdot 0 \]

Do they do that “correctly”?
Choreographies provide a principled development approach

Natural support for choreographic design
Choreographies provide a principled development approach

Natural support for choreographic design
Choreographies provide a principled development approach

**Natural support for choreographic design**
Choreographies provide a principled development approach

Natural support for choreographic design
Choreographies provide a principled development approach

Natural support for choreographic design
Choreographies provide a principled development approach

Natural support for choreographic design

[Diagram showing the flow of components and viewpoints, illustrating synchronization and asynchronization.]
Choreographies provide a principled development approach

Natural support for choreographic design

Behavioural types & distributed applications
Realisation...choreographically

### Well-formedness of global specs

Each guard, say \( l_i \), should be

- **causal consistent**
  - each selector in (the continuation of) \( l_i \) reacts to \( l_i \)
  - each role involved in the continuation of \( l_i \) cannot react to more events on \( l_i \) than selectors on the branch

- **determined**
  - each role in the continuation of \( l_i \) reacts to \( l_i[0] \)
  - selectors in the continuation of \( l_i \) react to the same set of event types in \( l_i \)

- **confusion-free**
  - guards of different branches start with distinct event types
  - an event type cannot occur in more than one guard
A glimpse of top-down design

Definition (Projection)

Given a global type $G$ and a subscription $\sigma$, the projection of $G$ over a role $R$ with respect to $\sigma$, written $G \downarrow^\sigma_R$, is co-inductively defined as follows:

$$
\left( \sum_{i \in I} c_i \odot R_i \langle l_i \rangle \cdot G_i \right) \downarrow^\sigma_R = \kappa \cdot \left[ \&_{i \in I} \left( c_i \odot R_i \langle l_i \rangle \cdot G_i \right) \downarrow^\sigma_R \right]
$$

where $\kappa = \{ (c_i, l_i) \mid i \in I \land R_i = R \}$ and

$$(c_i \odot R_i \langle l_i \rangle \cdot G_i) \downarrow^\sigma_R = \begin{cases} 
\text{filter}(l_i, \sigma(R)) \; ? \; (G_i \downarrow^\sigma_R) & \text{if filter}(l_i, \sigma(R)) \neq \epsilon \\
0 & \text{otherwise}
\end{cases}$$
Wrapping up
Conclusions

We have a useful basis to take design decisions
- a formal model ensuring relevant properties
- projectable global specs
- a prototype implementation for top-down engineering
Conclusions

We have a useful basis to take design decisions
- a formal model ensuring relevant properties
- projectable global specs
- a prototype implementation for top-down engineering

We would like to have
- our models as reference documentation for Actyx’s developers
- “minimal” subscriptions
- tools / develop behavioral typing / inference (ie going bottom-up)
- compensations (hence causality tracking) / active monitoring?
- failure handling (in event propagation)
Thank you!