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.)

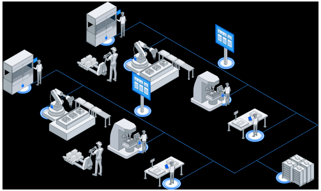
Plan

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

– Prelude –

[Factory automation]

Industrial scenarios



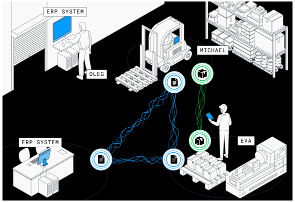
(courtesy of Actyx)

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



(courtesy of Actyx)

Execution model

machine/operator/forklift/... $\mapsto$ 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

Challenges

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 –

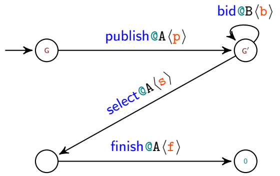
[Decide locally, agree globally…eventually]

Syntax

Global specs

$$G \ ::= \ \mathtt{c}_1 @ \mathtt{R}_1 \langle \mathtt{l}_1 \rangle \,.\, G_1 + \cdots + \mathtt{c}_n @ \mathtt{R}_n \langle \mathtt{l}_n \rangle \,.\, G_n$$

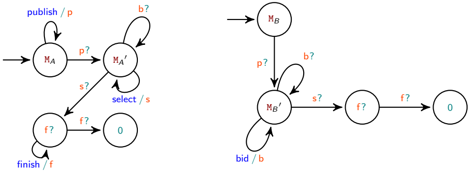
$$\begin{aligned} G &= \mathsf{publish} @ \mathtt{A} \langle \mathtt{p} \rangle \,.\, G' \\ G' &= \mathsf{bid} @ \mathtt{B} \langle \mathtt{b} \rangle \,.\, G' \\ &\quad + \\ &\quad \mathsf{select} @ \mathtt{A} \langle \mathtt{s} \rangle \,.\, \mathsf{finish} @ \mathtt{A} \langle \mathtt{f} \rangle \,.\, 0 \end{aligned}$$



Local Specs

$$\mathtt{M} \ ::= \ \kappa \cdot [\mathtt{t}_1 ?\, \mathtt{M}_1 \ \& \ \cdots \ \& \ \mathtt{t}_n ?\, \mathtt{M}_n]$$

$$\begin{aligned} \mathtt{M}_A &= \{\mathsf{publish} \,/\, \mathsf{p}\} \cdot [\mathsf{p} ?\, \mathtt{M}_A{}'] \\ \mathtt{M}_A{}' &= \{\mathsf{select} \,/\, \mathsf{s}\} \cdot [\mathsf{b} ?\, \mathtt{M}_A{}' \ \& \ \mathsf{s} ?\, \{\mathsf{finish} \,/\, \mathsf{f}\} \cdot \mathsf{f} ?\, 0] \\ \mathtt{M}_B &= \mathsf{p} ?\, \mathtt{M}_B{}' \\ \mathtt{M}_B{}' &= \{\mathsf{bid} \,/\, \mathsf{b}\} \cdot [\mathsf{b} ?\, \mathtt{M}_B{}' \ \& \ \mathsf{s} ?\, \mathsf{f} ?\, 0] \end{aligned}$$



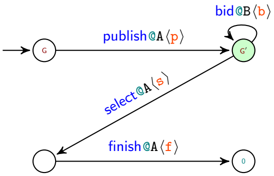
Syntax

Global specs

$$\mathtt{G} \;::=\; \mathsf{c}_1 @ \mathtt{R}_1 \langle \mathtt{l}_1 \rangle \,.\, \mathsf{G}_1 + \cdots + \mathsf{c}_n @ \mathtt{R}_n \langle \mathtt{l}_n \rangle \,.\, \mathsf{G}_n$$

$$\mathsf{G} = \mathsf{publish} @ \mathtt{A} \langle \mathsf{p} \rangle \,.\, \mathsf{G}'$$

$$\mathsf{G}' = \mathsf{bid} @ \mathtt{B} \langle \mathsf{b} \rangle \,.\, \mathsf{G}' \;+\; \mathsf{select} @ \mathtt{A} \langle \mathsf{s} \rangle \,.\, \mathsf{finish} @ \mathtt{A} \langle \mathtt{f} \rangle \,.\, 0$$



Local Specs

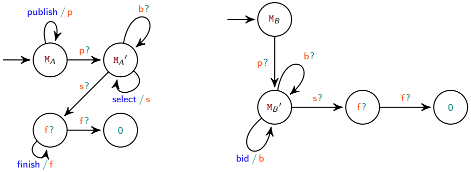
$$\mathtt{M} \;::=\; \kappa \cdot [\mathtt{t}_1 ? \mathtt{M}_1 \;\&\; \cdots \;\&\; \mathtt{t}_n ? \mathtt{M}_n]$$

$$\mathtt{M}_A = \{\mathsf{publish} \,/\, \mathsf{p}\} \cdot [\mathsf{p} ? \mathtt{M}_A{}']$$

$$\mathtt{M}_A{}' = \{\mathsf{select} \,/\, \mathsf{s}\} \cdot [\mathsf{b} ? \mathtt{M}_A{}' \;\&\; \mathsf{s} ? \{\mathsf{finish} \,/\, \mathsf{f}\} \cdot \mathsf{f} ? 0]$$

$$\mathtt{M}_B = \mathsf{p} ? \mathtt{M}_B{}'$$

$$\mathtt{M}_B{}' = \{\mathsf{bid} \,/\, \mathsf{b}\} \cdot [\mathsf{b} ? \mathtt{M}_B{}' \;\&\; \mathsf{s} ? \mathsf{f} ? 0]$$



Semantics of specifications

Global (protocol spec)

$$\frac{\delta(\mathsf{G}, l) \xrightarrow{\mathtt{c}\,/\,\mathtt{l}} \mathsf{G}' \qquad \vdash l' : \mathtt{l} \qquad l' \textit{ fresh}}{(\mathsf{G}, l) \xrightarrow{\mathtt{c}\,/\,\mathtt{l}} (\mathsf{G}, l \cdot l')}$$

Local (components' spec)

$$\frac{\delta(\mathtt{M}, l) = \mathtt{M}' \qquad \mathtt{M}' \downarrow_{\mathtt{c}\,/\,\mathtt{l}} \qquad \vdash l' : \mathtt{l} \qquad l' \text{ fresh}}{(\mathtt{M}, l) \xrightarrow{\mathtt{c}\,/\,\mathtt{l}} (\mathtt{M}, l \cdot l')}$$

where

l is an (idealised) global/shared log

$$\cdots + \mathtt{c}_i @ \mathrm{R}_i \langle \mathtt{l}_i \rangle \,.\, \mathsf{G}_i + \cdots \quad \xrightarrow{\mathtt{c}_i\,/\,\mathtt{l}_i} \quad \mathsf{G}_i$$

$$\delta(\mathsf{G}, \epsilon) = \mathsf{G}$$

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

where

l is the local log accessible to $\mathtt{M}$

$$\mathtt{M}' \downarrow_{\mathtt{c}\,/\,\mathtt{l}} \iff \mathtt{c}\,/\,\mathtt{l} \text{ enabled at } \mathtt{M}'$$

$$\delta(\mathtt{M}, \epsilon) = \mathtt{M}$$

$$\delta(\mathtt{M}, e \cdot l) = \begin{cases} \delta(\mathtt{M}_j, l) & \text{if } \ \vdash e : \mathtt{t}_j, \\ & \quad \mathtt{M} = \kappa \cdot [\ldots \ \& \ \mathtt{t}_j ?\, \mathtt{M}_j \ \& \ \ldots] \\ \delta(\mathtt{M}, l) & \text{otherwise} \end{cases}$$

Syntax of systems

Events are univocally associated to the machines generating them.

Machines with local logs & a global one

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

(Recall: the global log is an optical illusion)

Well-formedness

$(\mathtt{M}_1, l_1) \mid \ldots \mid (\mathtt{M}_n, l_n) \mid l$ is well-formed if

$l = l_1 \cup \cdots \cup l_n$ and for all i, $l_i \sqsubseteq l$

Syntax of systems

Events are univocally associated to the machines generating them.

Machines with local logs & a global one

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

(Recall: the global log is an optical illusion)

Well-formedness

$(\mathtt{M}_1, l_1) \mid \ldots \mid (\mathtt{M}_n, l_n) \mid l$ is well-formed if

$$l = l_1 \cup \cdots \cup l_n \qquad \text{and} \qquad \text{for all } i,\ l_i \sqsubseteq l$$

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

Syntax of systems

Events are univocally associated to the machines generating them.

Machines with local logs & a global one

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

(Recall: the global log is an optical illusion)

Well-formedness

$(\mathtt{M}_1, l_1) \mid \ldots \mid (\mathtt{M}_n, l_n) \mid l$ is well-formed if

$l = l_1 \cup \cdots \cup l_n$ and for all i, $l_i \sqsubseteq l$

where $l_i \sqsubseteq l \iff$

$$l_i = \begin{array}{ccc} & & e_1 \\ e_{i,1} & \longrightarrow & \\ \vdots & & \vdots \\ e_{i,n} & \longrightarrow & \\ & & e_m \end{array} = l$$

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

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

(Recall: the global log is an optical illusion)

Well-formedness

$(\mathtt{M}_1, l_1) \mid \ldots \mid (\mathtt{M}_n, l_n) \mid l$ is well-formed if

$$l = l_1 \cup \cdots \cup l_n \qquad \text{and} \qquad \text{for all } i,\ l_i \sqsubseteq l$$

where $\quad l_i \sqsubseteq l \iff$

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

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

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

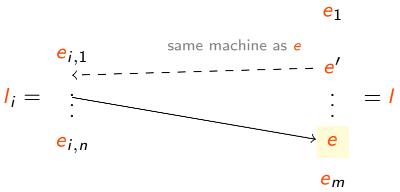
(Recall: the global log is an optical illusion)

Well-formedness

$(\mathtt{M}_1, l_1) \mid \ldots \mid (\mathtt{M}_n, l_n) \mid l$ is well-formed if

$l = l_1 \cup \cdots \cup l_n$ and for all i, $l_i \sqsubseteq l$

where $l_i \sqsubseteq l \iff$



formally, there is an order-preserving and downward-total morphism from l_i into l

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

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]

$$\frac{i \in \operatorname{dom} \mathtt{S} \qquad \mathtt{S}(i) = (\mathtt{M}, l_i) \qquad (\mathtt{M}, l_i) \xrightarrow{\mathtt{c}\,/\,\mathtt{l}} (\mathtt{M}, l_i') \qquad l' \in l \bowtie l_i'}{(\mathtt{S}, l) \xrightarrow{\mathtt{c}\,/\,\mathtt{l}} (\mathtt{S}[i \mapsto (\mathtt{M}, l_i')], l')}$$

where

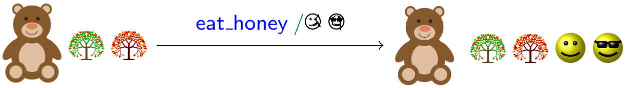
$$l_1 \bowtie l_2 = \{ l \mid l \subseteq l_1 \cup l_2 \ \wedge \ l_1 \sqsubseteq l \ \wedge \ l_2 \sqsubseteq l \}$$

[Prop]

$$\frac{i \in \operatorname{dom} \mathtt{S} \qquad \mathtt{S}(i) = (\mathtt{M}, l_i) \qquad l_i \sqsubseteq l' \sqsubseteq l \qquad l_i \subset l'}{(\mathtt{S}, l) \xrightarrow{\tau} (\mathtt{S}[i \mapsto (\mathtt{M}, l')], l)}$$

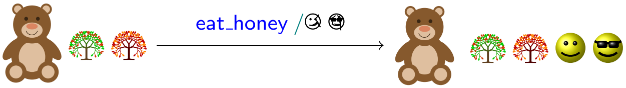
Events generation

If

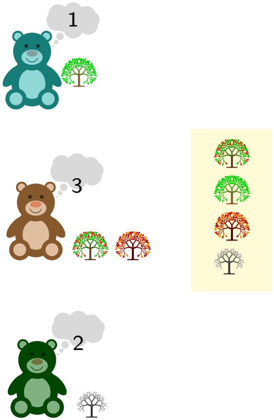


Events generation

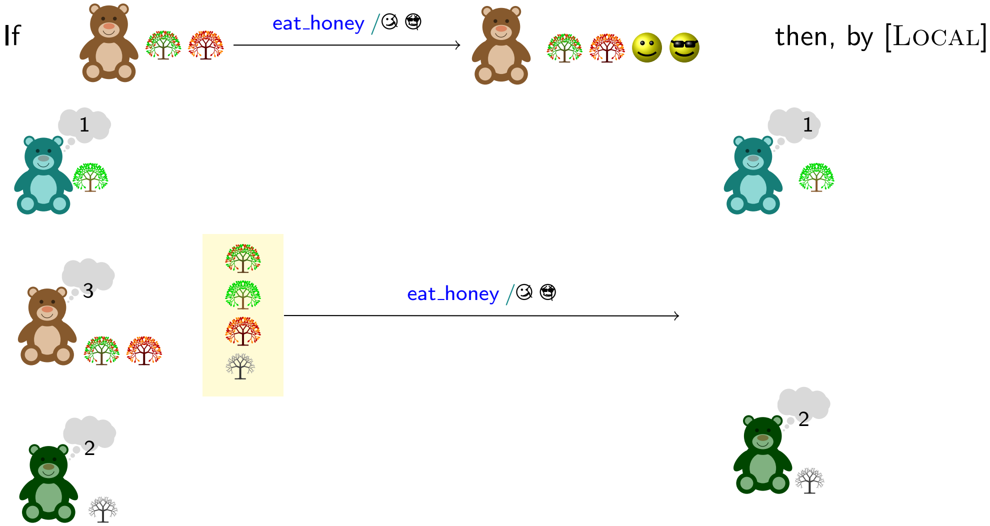
If



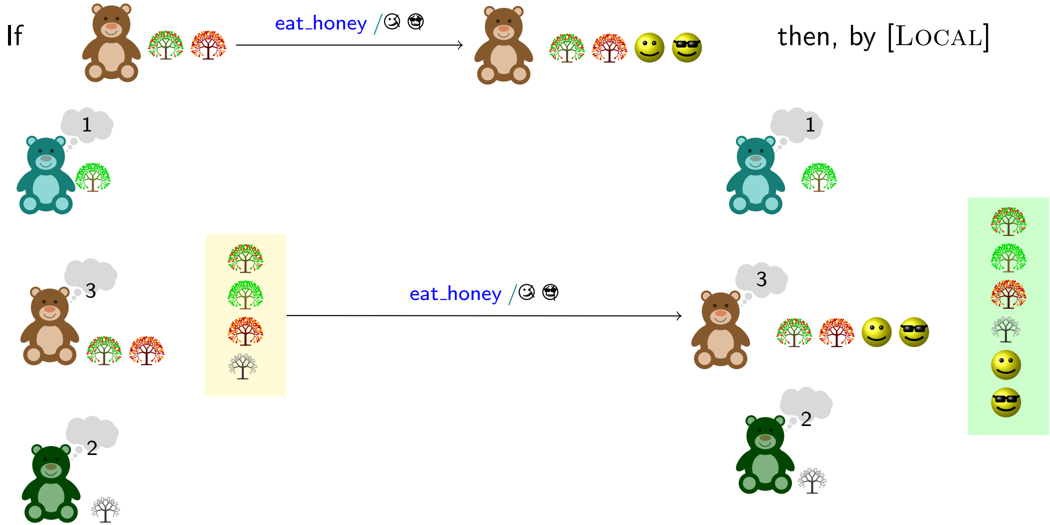
then, by [Local]



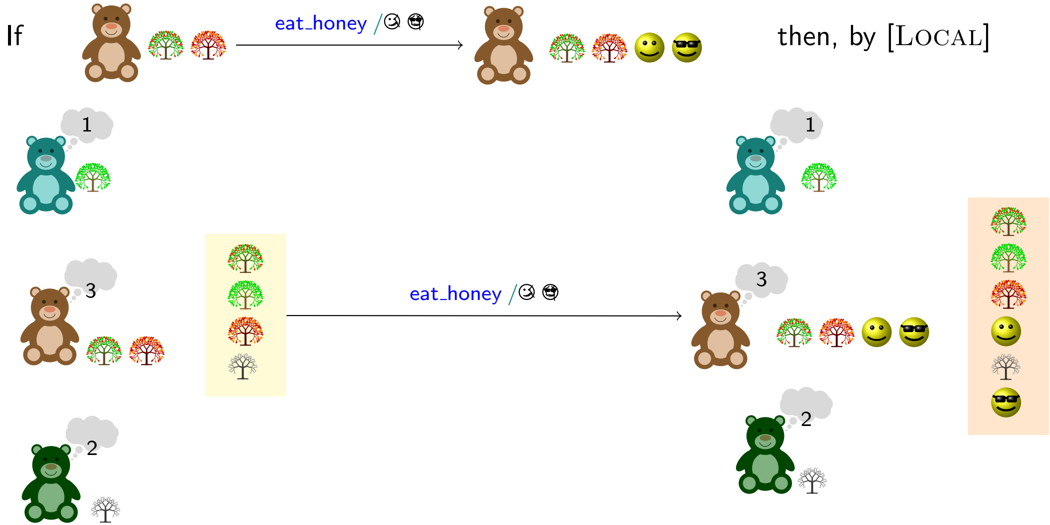
Events generation



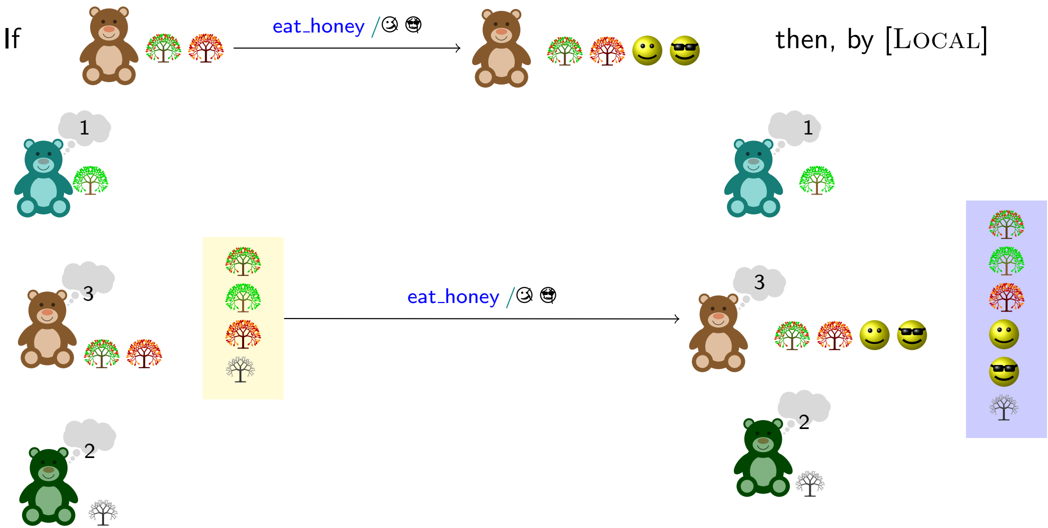
Events generation



Events generation

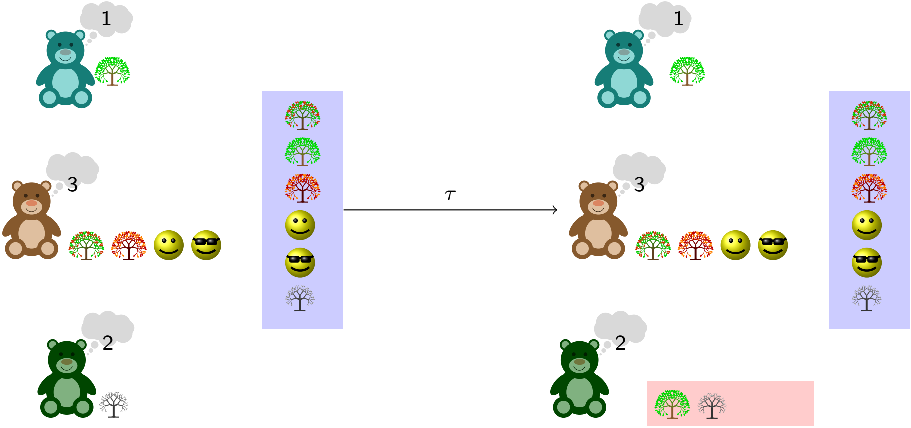


Events generation



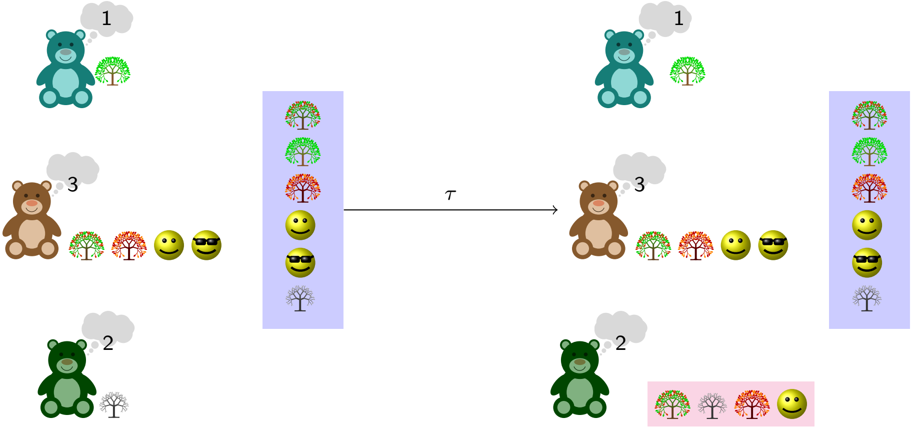
Propagation

By rule [PROP]



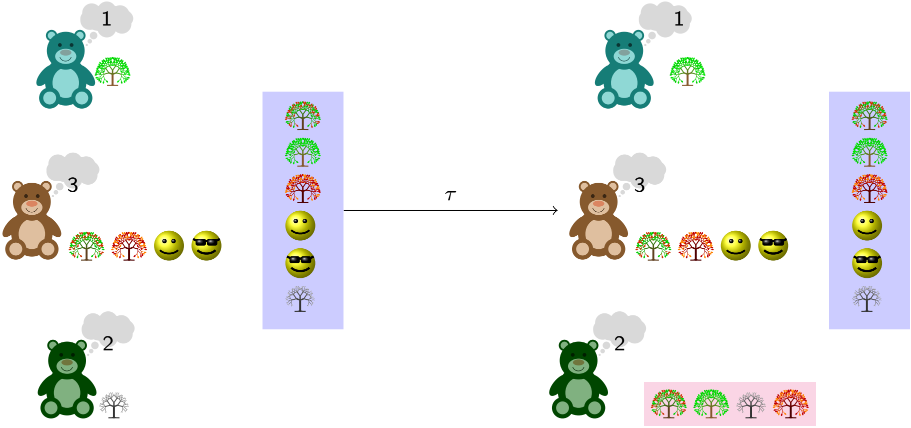
Propagation

By rule [PROP]



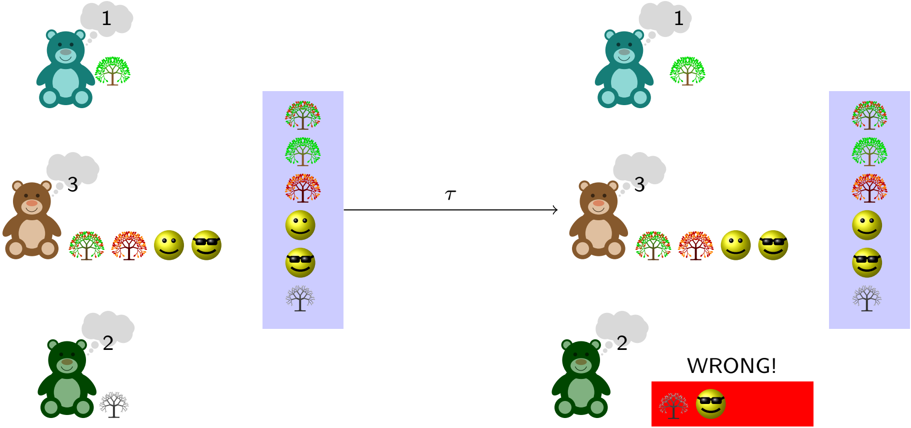
Propagation

By rule [PROP]



Propagation

By rule [PROP]



Properties of our semantics

Theorem: Well-Formedness preservation

If

$\mathtt{S}$ is well-formed and $\mathtt{S} \xrightarrow{[Local]/[Prop]}^{\star} \mathtt{S}'$

then

$\mathtt{S}'$ is well-formed

Theorem: Eventual Consistency

If

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

then

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

– 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

$$\mathsf{G} = \mathsf{prepare@}\mathtt{P}\langle\mathtt{piece}\rangle\,.\,\mathsf{pack@}\mathtt{A}\langle\mathtt{product}\rangle\,.\,\mathsf{0}$$

Do the following machines realise G?

$$\mathtt{M}_P = \{\mathsf{prepare} \,/\, \mathsf{piece}\}\cdot\mathsf{0} \qquad \text{and} \qquad \mathtt{M}_A = \mathsf{piece?}\,\{\mathsf{pack} \,/\, \mathsf{product}\}\cdot\mathsf{0}$$

Exercise on realisation

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

A trivial protocol

Take

$$\mathsf{G} = \mathsf{prepare}@\mathtt{P}\langle\mathtt{piece}\rangle \,.\, \mathsf{pack}@\mathtt{A}\langle\mathtt{product}\rangle \,.\, \mathbf{0}$$

Do the following machines realise G?

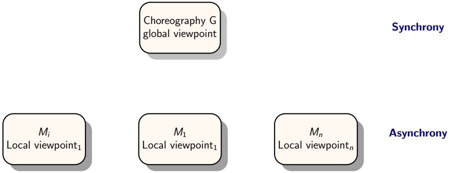
$$\mathtt{M}_P = \{\mathsf{prepare} \,/\, \mathsf{piece}\}\cdot\mathbf{0} \qquad \text{and} \qquad \mathtt{M}_A = \mathsf{piece}?\,\{\mathsf{pack} \,/\, \mathsf{product}\}\cdot\mathbf{0}$$

Do they do that "correctly"?

Behavioural types & distributed applications

Choreographies provide a principled development approach

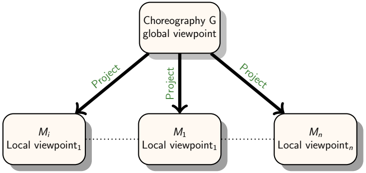
Natural support for choreographic design



Behavioural types & distributed applications

Choreographies provide a principled development approach

Natural support for choreographic design



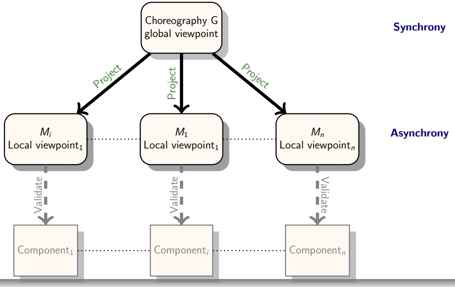
Synchrony

Asynchrony

Behavioural types & distributed applications

Choreographies provide a principled development approach

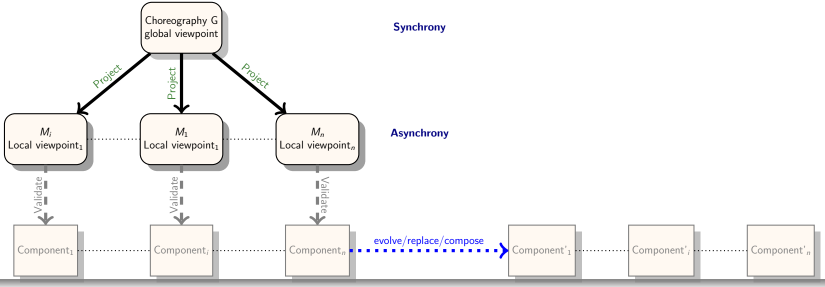
Natural support for choreographic design



Behavioural types & distributed applications

Choreographies provide a principled development approach

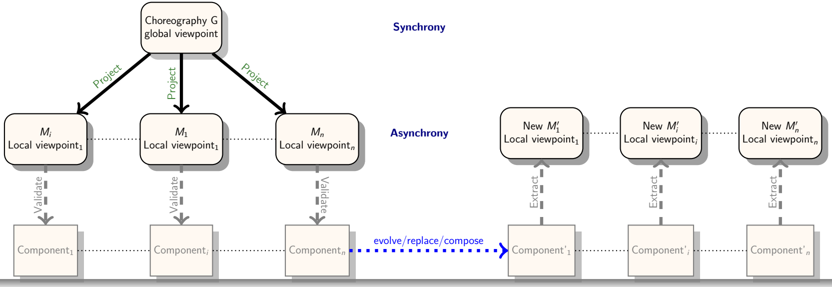
Natural support for choreographic design



Behavioural types & distributed applications

Choreographies provide a principled development approach

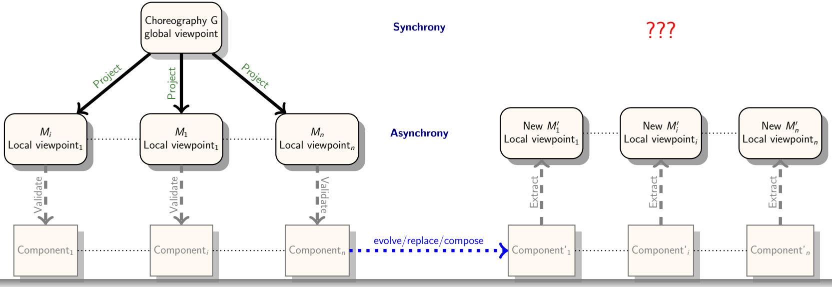
Natural support for choreographic design



Behavioural types & distributed applications

Choreographies provide a principled development approach

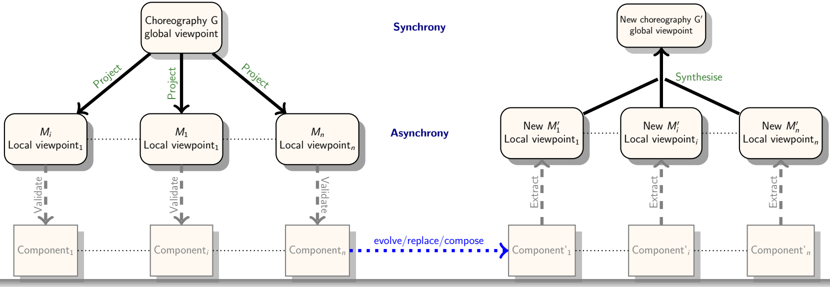
Natural support for choreographic design



Behavioural types & distributed applications

Choreographies provide a principled development approach

Natural support for choreographic design



Realisation...choreographically

Well-formedness of global specs

Each guard, say $\mathtt{l}_i$, should be

- causal consistent
  - each selector in (the continuation of) $\mathtt{l}_i$ reacts to $\mathtt{l}_i$
  - each role involved in the continuation of $\mathtt{l}_i$ cannot react to more events on $\mathtt{l}_i$ than selectors on the branch
- determined
  - each role in the continuation of $\mathtt{l}_i$ reacts to $\mathtt{l}_i[0]$
  - selectors in the continuation of $\mathtt{l}_i$ react to the same set of event types in $\mathtt{l}_i$
- 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 σ, the *projection of* G *over a role* $\mathtt{R}$ *with respect to* σ, written $\mathsf{G}\downarrow^{\sigma}_{\mathtt{R}}$, is co-inductively defined as follows:

$$\Big(\sum_{i\in I} \mathsf{c}_i @ \mathtt{R}_i\langle \mathtt{l}_i\rangle \,.\, \mathsf{G}_i\Big)\downarrow^{\sigma}_{\mathtt{R}} \quad = \quad \kappa\cdot\Big[\&_{i\in I}\,(\mathsf{c}_i @ \mathtt{R}_i\langle \mathtt{l}_i\rangle \,.\, \mathsf{G}_i)\downarrow^{\sigma}_{\mathtt{R}}\Big]$$

where $\kappa = \{(\mathsf{c}_i, \mathtt{l}_i) \mid i \in I \wedge \mathtt{R}_i = \mathtt{R}\}$ and

$$(\mathsf{c}_i @ \mathtt{R}_i\langle \mathtt{l}_i\rangle \,.\, \mathsf{G}_i)\downarrow^{\sigma}_{\mathtt{R}} = \begin{cases} \mathsf{filter}(\mathtt{l}_i, \sigma(\mathtt{R}))\,?\,(\mathsf{G}_i\downarrow^{\sigma}_{\mathtt{R}}) & \text{if } \mathsf{filter}(\mathtt{l}_i, \sigma(\mathtt{R})) \neq \epsilon \\ 0 & \text{otherwise} \end{cases}$$

– 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!