

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.)

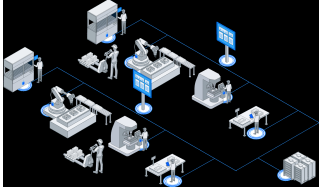
Plan

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

– Prelude –

[Factory automation]

Industrial scenarios



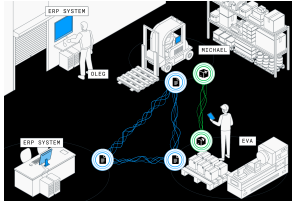
(courtesy of Actyx)

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]

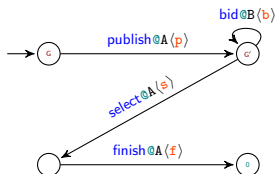
Syntax

Global specs

$$G ::= c_1 @ R_1 \langle l_1 \rangle . G_1 + \dots + c_n @ R_n \langle l_n \rangle . G_n$$

$$G = \text{publish} @ A \langle p \rangle . G'$$

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



Local Specs

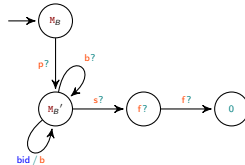
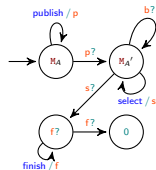
$$M ::= \kappa . [t_1 ? M_1 \ \& \ \dots \ \& \ t_n ? M_n]$$

$$M_A = \{\text{publish} / p\} . [p ? M_A']$$

$$M_A' = \{\text{select} / s\} . [b ? M_A' \ \& \ s ? \{\text{finish} / f\} . f ? 0]$$

$$M_B = p ? M_B'$$

$$M_B' = \{\text{bid} / b\} . [b ? M_B' \ \& \ s ? f ? 0]$$



Syntax

Global specs

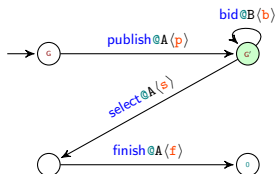
$$G ::= c_1 @ R_1 \langle l_1 \rangle . G_1 + \dots + c_n @ R_n \langle l_n \rangle . G_n$$

$$G = \text{publish} @ A \langle p \rangle . G'$$

$$G' = \text{bid} @ B \langle b \rangle . G'$$

+

$$\text{select} @ A \langle s \rangle . \text{finish} @ A \langle f \rangle . 0$$



Local Specs

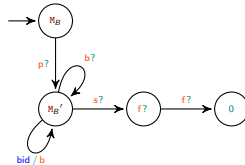
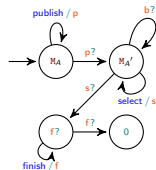
$$M ::= \kappa . [t_1 ? M_1 \ \& \ \dots \ \& \ t_n ? M_n]$$

$$M_A = \{\text{publish} / p\} . [p ? M_A']$$

$$M_A' = \{\text{select} / s\} . [b ? M_A' \ \& \ s ? \{\text{finish} / f\} . f ? 0]$$

$$M_B = p ? M_B'$$

$$M_B' = \{\text{bid} / b\} . [b ? M_B' \ \& \ s ? f ? 0]$$



Semantics of specifications

Global (protocol spec)

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

where

l is an (idealised) global/shared log

$$\dots + c_i @ R_i \langle l_i \rangle . G_i + \dots \xrightarrow{c_i / l_i} G_i$$

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

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

Local (components' spec)

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

where

l is the local log accessible to M

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

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

$$\delta(M, e \cdot l) = \begin{cases} \delta(M_j, l) & \text{if } \vdash e : t_j, \\ & M = \kappa[\dots \& t_j?M_j \& \dots] \\ \delta(M, l) & \text{otherwise} \end{cases}$$

Syntax of systems

Events are univocally associated to the machines generating them.

Machines with local logs & a global one

$$S = (M_1, I_1) \mid \dots \mid (M_n, I_n) \mid I$$

(Recall: the global log is an optical illusion)

Well-formedness

$(M_1, I_1) \mid \dots \mid (M_n, I_n) \mid I$ is well-formed if

$$I = I_1 \cup \dots \cup I_n \quad \text{and} \quad \text{for all } i, I_i \sqsubseteq I$$

Syntax of systems

Events are univocally associated to the machines generating them.

Machines with local logs & a global one

$$S = (M_1, I_1) \mid \dots \mid (M_n, I_n) \mid I$$

(Recall: the global log is an optical illusion)

Well-formedness

$(M_1, I_1) \mid \dots \mid (M_n, I_n) \mid I$ is well-formed if

$$I = I_1 \cup \dots \cup I_n \quad \text{and} \quad \text{for all } i, I_i \subseteq I$$

where

$$I_i \subseteq I \iff I_i = \begin{matrix} e_{i,1} \\ \vdots \\ e_{i,n} \end{matrix} \quad \begin{matrix} e_1 \\ \vdots \\ e_m \end{matrix} = I$$

Syntax of systems

Events are univocally associated to the machines generating them.

Machines with local logs & a global one

$$S = (M_1, l_1) \mid \dots \mid (M_n, l_n) \mid l$$

(Recall: the global log is an optical illusion)

Well-formedness

$(M_1, l_1) \mid \dots \mid (M_n, l_n) \mid l$ is well-formed if

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

where

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

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

$$S = (M_1, I_1) \mid \dots \mid (M_n, I_n) \mid I$$

(Recall: the global log is an optical illusion)

Well-formedness

$(M_1, I_1) \mid \dots \mid (M_n, I_n) \mid I$ is well-formed if

$$I = I_1 \cup \dots \cup I_n \quad \text{and} \quad \text{for all } i, I_i \sqsubseteq I$$

where

$$I_i \sqsubseteq I \iff I_i = \begin{matrix} e_{i,1} \\ \vdots \\ e_{i,n} \end{matrix} \rightarrow \begin{matrix} \vdots \\ e \\ e_m \end{matrix} = I$$

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

$$S = (M_1, I_1) \mid \dots \mid (M_n, I_n) \mid I$$

(Recall: the global log is an optical illusion)

Well-formedness

$(M_1, I_1) \mid \dots \mid (M_n, I_n) \mid I$ is **well-formed** if

$$I = I_1 \cup \dots \cup I_n \quad \text{and} \quad \text{for all } i, I_i \sqsubseteq I$$

where

$$I_i \sqsubseteq I \iff I_i = \begin{array}{c} e_{i,1} \\ \vdots \\ e_{i,n} \end{array} \begin{array}{c} \xleftarrow{\text{same machine as } e} e' \\ \vdots \\ \rightarrow e \end{array} = I$$

formally, there is an **order-preserving** and **downward-total morphism** from I_i into I

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]

$$\frac{i \in \text{dom } \mathbf{S} \quad \mathbf{S}(i) = (\mathbf{M}, l_i) \quad (\mathbf{M}, l_i) \xrightarrow{c/l} (\mathbf{M}, l'_i) \quad l' \in l \bowtie l'_i}{(\mathbf{S}, l) \xrightarrow{c/l} (\mathbf{S}[i \mapsto (\mathbf{M}, l'_i)], l')}$$

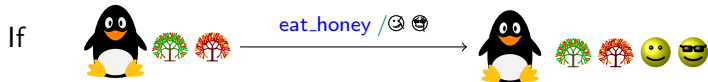
where

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

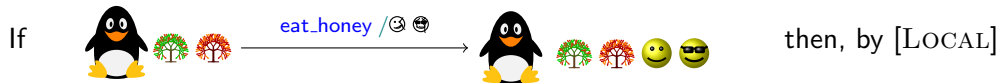
[PROP]

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

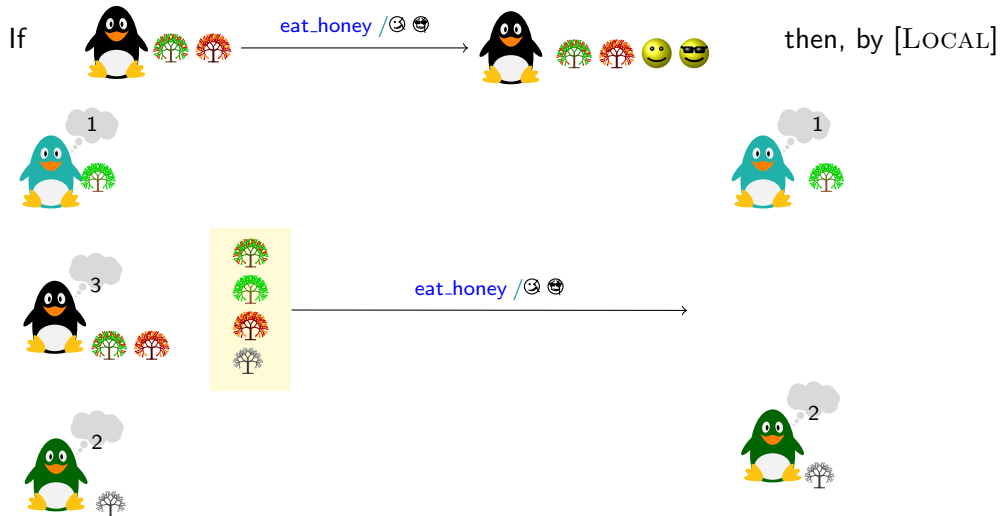
Events generation



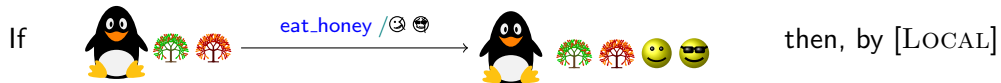
Events generation



Events generation



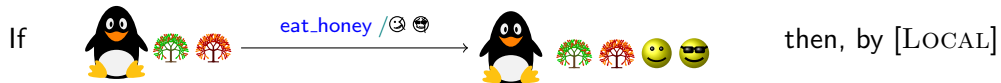
Events generation



$\xrightarrow{\text{eat_honey} / \text{😄} \text{😋}}$



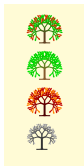
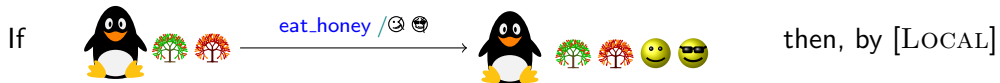
Events generation



$\xrightarrow{\text{eat_honey} / \text{😄} \text{😄}}$



Events generation

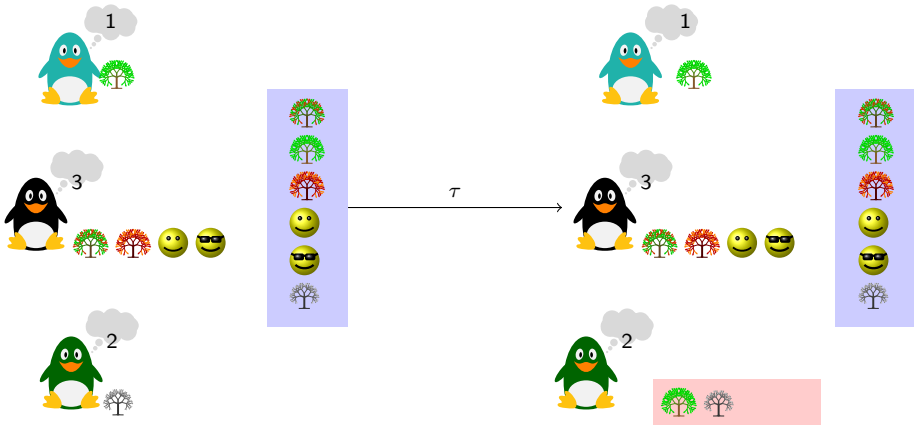


$\xrightarrow{\text{eat_honey} / \text{😄} \text{😄}}$



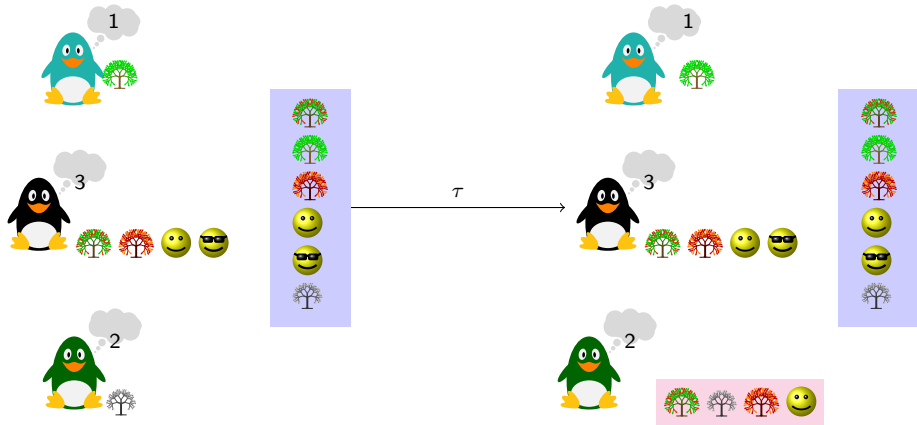
Propagation

By rule [PROP]



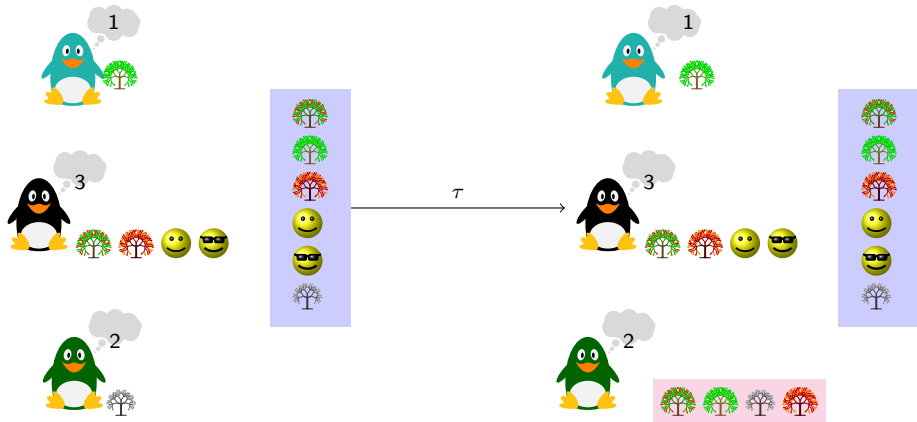
Propagation

By rule [PROP]



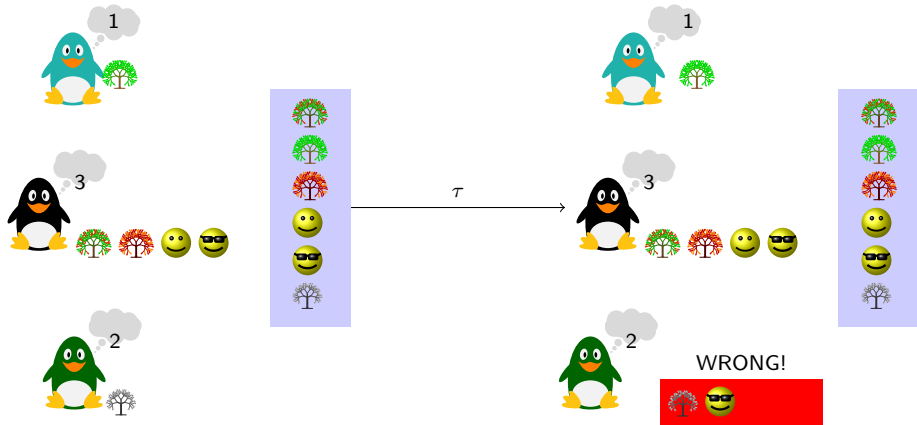
Propagation

By rule [PROP]



Propagation

By rule [PROP]



Properties of our semantics

Theorem: Well-Formedness preservation

If

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

then

S' is well-formed

Theorem: Eventual Consistency

If

$S = (M_1, I_1) \mid \dots \mid (M_n, I_n) \mid I$ is well-formed

then

$S \xrightarrow{\tau}^* (M_1, I) \mid \dots \mid (M_n, I) \mid I$

- 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

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

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

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

Do the following machines realise G ?

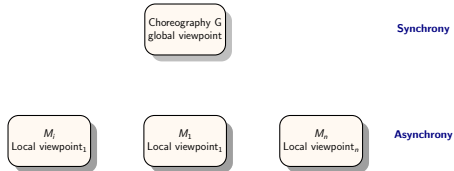
$$M_P = \{\text{prepare} / \text{piece}\} . 0 \quad \text{and} \quad M_A = \text{piece?} \{\text{pack} / \text{product}\} . 0$$

Do they do that “**correctly**”?

Behavioural types & distributed applications

Choreographies provide a principled development approach

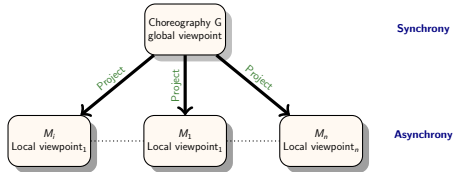
Natural support for choreographic design



Behavioural types & distributed applications

Choreographies provide a principled development approach

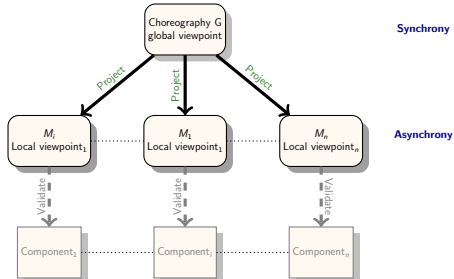
Natural support for choreographic design



Behavioural types & distributed applications

Choreographies provide a principled development approach

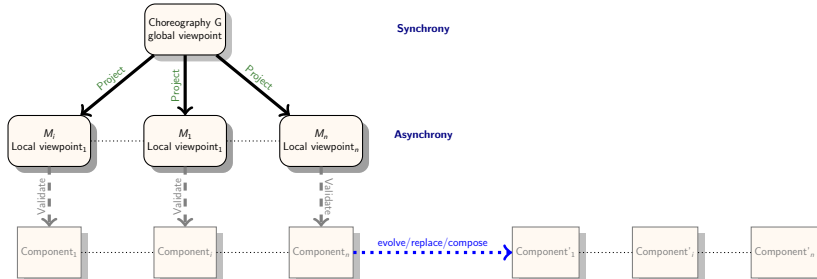
Natural support for choreographic design



Behavioural types & distributed applications

Choreographies provide a principled development approach

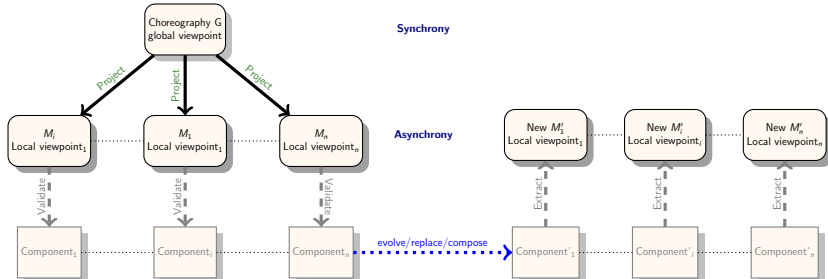
Natural support for choreographic design



Behavioural types & distributed applications

Choreographies provide a principled development approach

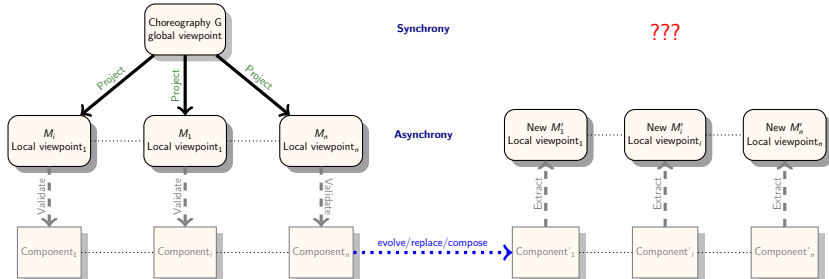
Natural support for choreographic design



Behavioural types & distributed applications

Choreographies provide a principled development approach

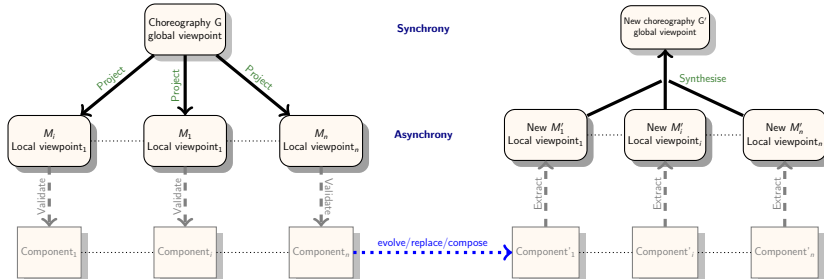
Natural support for choreographic design



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 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 σ , the *projection of G over a role R with respect to σ* , written $G \downarrow_R^\sigma$, is co-inductively defined as follows:

$$\left(\sum_{i \in I} c_i @ R_i \langle \mathbf{1}_i \rangle . G_i \right) \downarrow_R^\sigma = \kappa \cdot \left[\&_{i \in I} (c_i @ R_i \langle \mathbf{1}_i \rangle . G_i) \downarrow_R^\sigma \right]$$

where $\kappa = \{ (c_i, \mathbf{1}_i) \mid i \in I \wedge R_i = R \}$ and

$$(c_i @ R_i \langle \mathbf{1}_i \rangle . G_i) \downarrow_R^\sigma = \begin{cases} \text{filter}(\mathbf{1}_i, \sigma(R)) ? (G_i \downarrow_R^\sigma) & \text{if } \text{filter}(\mathbf{1}_i, \sigma(R)) \neq \epsilon \\ 0 & \text{otherwise} \end{cases}$$

– 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!