Open Problems in Choreographic Development of Message-Passing Applications

Emilio Tuosto @ GSSI

ASYDE 2020 September 15, 2020

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

Take-home message

This talk in 1 slide

Take-home message

This talk in 1 slide

Choreographic development of distributed (message-passing) systems

- exploits global & local specifications
- supports correctness-by-construction
- facilitates SDLC

Take-home message

This talk in 1 slide

Choreographic development of distributed (message-passing) systems

- exploits global & local specifications
- supports correctness-by-construction
- facilitates SDLC

Nonetheless, choreographies

- lack support for modularity/compositionality
- to be complemented by testing
- generalisations to (more abstract) coordination paradigm
- ...

Take-home message

This talk in 1 slide

Choreographic development of distributed (message-passing) systems

- exploits global & local specifications
- supports correctness-by-construction
- facilitates SDLC

Nonetheless, choreographies

- lack support for modularity/compositionality
- to be complemented by testing
- generalisations to (more abstract) coordination paradigm
- ...

Goal

Generate interest and/or criticisms & possibly collaborations

– Prologue –

[Choreographies, informally]

What do I mean by "choreography"?

Choreography $=$ Global spec $+$ Local spec

Model-driven development...by nature

Choreography = Global spec + Local spec

Model-driven development...by nature

Choreography
=
Global spec
+
Local spec

Quoting W3C:

"Using the Web Services Choreography specification, a contract containing a global definition of the common ordering conditions and constraints under which messages are exchanged, is produced that describes, from a global viewpoint [...] observable behaviour of all the parties involved. Each party can then use the global definition to build and test solutions that conform to it. The global specification is in turn realised by combination of the resulting local systems [...]"

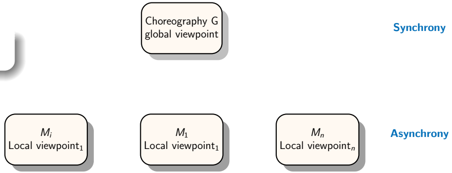
Model-driven development...by nature

Choreography = Global spec + Local spec

Quoting W3C:

"Using the Web Services Choreography specification, a contract containing a global definition of the common ordering conditions and constraints under which messages are exchanged, is produced that describes, from a global viewpoint [...] observable behaviour of all the parties involved. Each party can then use the global definition to build and test solutions that conform to it. The global specification is in turn realised by combination of the resulting local systems [...]"

specs, not code



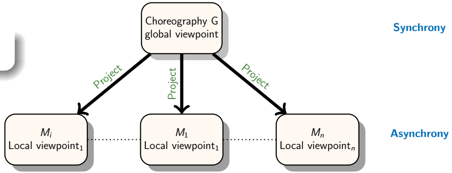
Model-driven development...by nature

Choreography = Global spec + Local spec

Quoting W3C:

"Using the Web Services Choreography specification, a contract containing a global definition of the common ordering conditions and constraints under which messages are exchanged, is produced that describes, from a global viewpoint [...] observable behaviour of all the parties involved. Each party can then use the global definition to build and test solutions that conform to it. The global specification is in turn realised by combination of the resulting local systems [...]"

specs, not code



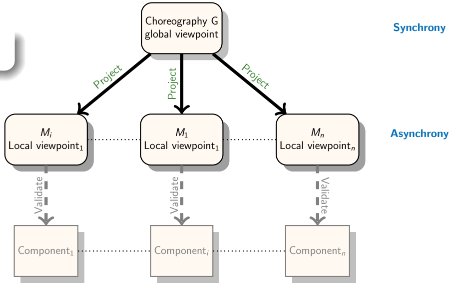
Model-driven development...by nature

Choreography = Global spec + Local spec

Quoting W3C:

"Using the Web Services Choreography specification, a contract containing a global definition of the common ordering conditions and constraints under which messages are exchanged, is produced that describes, from a global viewpoint [...] observable behaviour of all the parties involved. Each party can then use the global definition to build and test solutions that conform to it. The global specification is in turn realised by combination of the resulting local systems [...]"

specs, not code



Some advantages

A (possibly) useful equation

Distribution $=$ Local computation $+$ Communication

Some advantages

A (possibly) useful equation

Distribution = Local computation + Communication

- Separation of concerns

Some advantages

A (possibly) useful equation

$$\text{Distribution} = \text{Local computation} + \text{Communication}$$

- Separation of concerns
- Global specs support program comprehension

Some advantages

A (possibly) useful equation

Distribution = Local computation + Communication

- Separation of concerns
- Global specs support program comprehension
- "Distributed / DevOP-ish" development
  - Projections yield specs of local components
  - Developers can "test" each component against the local spec
  - **if** cond(global artefact) **then** *behave*(*projection*(global artefact))

Some advantages

A (possibly) useful equation

Distribution = Local computation + Communication

- Separation of concerns
- Global specs support program comprehension
- "Distributed / DevOP-ish" development
  - Projections yield specs of local components
  - Developers can "test" each component against the local spec
  - **if** cond(global artefact) **then** *behave*(*projection*(global artefact))
- No centralisation / full distribution / Scalability
- ...

Some drawbacks

A (possibly) painful equation

Distribution = Local computation + Communication

Some drawbacks

A (possibly) painful equation

Distribution $=$ Local computation $+$ Communication

- The "right" global spec could be difficult to be found

Some drawbacks ☹

A (possibly) painful equation

$$\text{Distribution} = \text{Local computation} + \text{Communication}$$

- The “right” global spec could be difficult to be found
- Message-passing: “unusual” paradigm & asynchrony $\implies$ complexity

Some drawbacks ☹

A (possibly) painful equation

Distribution = Local computation + Communication

- The "right" global spec could be difficult to be found
- Message-passing: "unusual" paradigm & asynchrony $\Longrightarrow$ complexity
- "coordination = communication" / distributed consensus / dependencies

Some drawbacks

A (possibly) painful equation

Distribution = Local computation + Communication

- The "right" global spec could be difficult to be found
- Message-passing: "unusual" paradigm & asynchrony $\Longrightarrow$ complexity
- "coordination = communication" / distributed consensus / dependencies
  - local decisions (sometimes) need to be propagated
  - global state "scattered" across components
  - hence, components (must!) have a partial view of the global state

Some drawbacks

A (possibly) painful equation

Distribution $=$ Local computation $+$ Communication

- The "right" global spec could be difficult to be found
- Message-passing: "unusual" paradigm & asynchrony $\Longrightarrow$ complexity
- "coordination $=$ communication" / distributed consensus / dependencies
  - local decisions (sometimes) need to be propagated
  - global state "scattered" across components
  - hence, components (must!) have a partial view of the global state
- Expose behaviour

Some drawbacks

A (possibly) painful equation

Distribution = Local computation + Communication

- The "right" global spec could be difficult to be found
- Message-passing: "unusual" paradigm & asynchrony $\Longrightarrow$ complexity
- "coordination = communication" / distributed consensus / dependencies
  - local decisions (sometimes) need to be propagated
  - global state "scattered" across components
  - hence, components (must!) have a partial view of the global state
- Expose behaviour
- Code reuse maybe problematic

Some drawbacks

A (possibly) painful equation

Distribution = Local computation + Communication

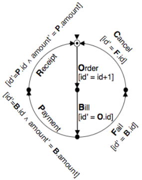
- The "right" global spec could be difficult to be found
- Message-passing: "unusual" paradigm & asynchrony $\Longrightarrow$ complexity
- "coordination = communication" / distributed consensus / dependencies
  - local decisions (sometimes) need to be propagated
  - global state "scattered" across components
  - hence, components (must!) have a partial view of the global state
- Expose behaviour
- Code reuse maybe problematic
- ... (wait for the last part)

– Act I –

[A bird-eye view of choregraphic design]

Global specs & Local specs

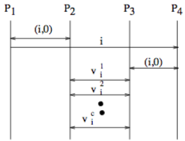
There're many[1] ("formal") models...



Conversation protocols [Bultan et al.]

$$G' \stackrel{\text{def}}{=} \mu \mathbf{t}. \begin{pmatrix} \mathtt{DP} \to \mathtt{K}\colon d\,\langle \mathsf{bool} \rangle. \\ \mathtt{KP} \to \mathtt{K}\colon k\,\langle \mathsf{bool} \rangle. \\ \mathtt{K} \to \mathtt{C}\colon c\,\langle \mathsf{bool} \rangle.\ \mathbf{t} \end{pmatrix}$$

Global Types [Honda et al.]



MSC [Alur et al.]

...

Reminder

I'm not advertising: my goal is to generate interest, criticisms & possibly collaborations

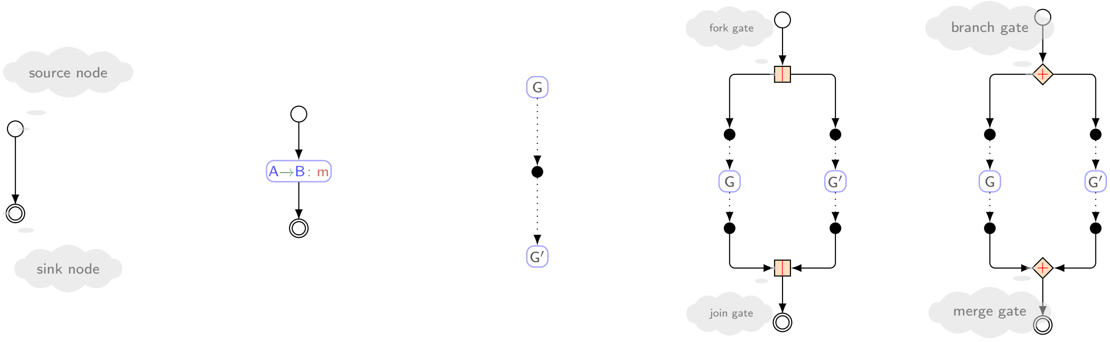
[1] No systematic comparative study yet

Choosing a model of global specs

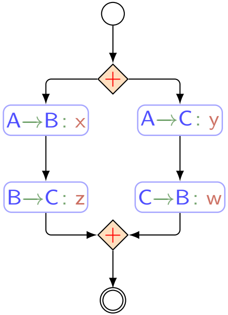
$$
\begin{array}{llll}
\mathsf{G} & ::= & (\mathsf{o}) & \text{empty} \\
 & | & \mathsf{A} \rightarrow \mathsf{B} : \mathsf{m} & \text{interaction} \\
 & | & \mathsf{G} ; \mathsf{G} & \text{sequential} \\
 & | & \mathsf{G} \mid \mathsf{G} & \text{parallel} \\
 & | & \texttt{sel}\ \{\mathsf{G} + \cdots + \mathsf{G}\} & \text{branch}
\end{array}
$$

Choosing a model of global specs

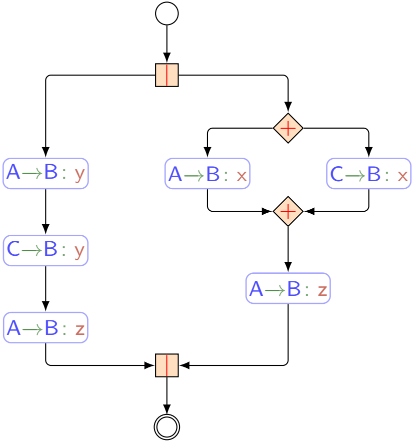
$$
\begin{array}{lcll}
\mathsf{G} & ::= & (\mathsf{o}) & \text{empty} \\
& | & \mathsf{A} \to \mathsf{B} : \mathsf{m} & \text{interaction} \\
& | & \mathsf{G} ; \mathsf{G} & \text{sequential} \\
& | & \mathsf{G} \mid \mathsf{G} & \text{parallel} \\
& | & \texttt{sel}\ \{\mathsf{G} + \cdots + \mathsf{G}\} & \text{branch}
\end{array}
$$



Some examples



$$A \to B : x; B \to C : z$$
$$+$$
$$A \to C : y; C \to B : w$$



$$A \to B : y; C \to B : y; A \to B : z$$
$$|$$
$$A \to B : x + C \to B : x; A \to B : z$$

Setting-up a communication model

We're going to review some results about a specific communication model

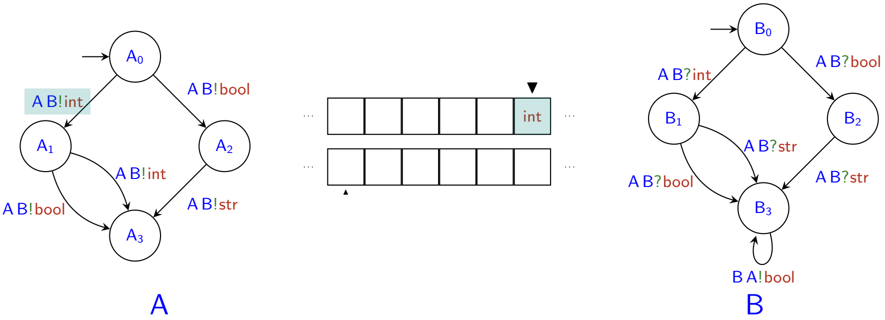
- channel-based
- asynchronous (most often)
- point-to-point

Setting-up a communication model

We're going to review some results about a specific communication model

- channel-based
- asynchronous (most often)
- point-to-point

Communicating Finite-State Machines



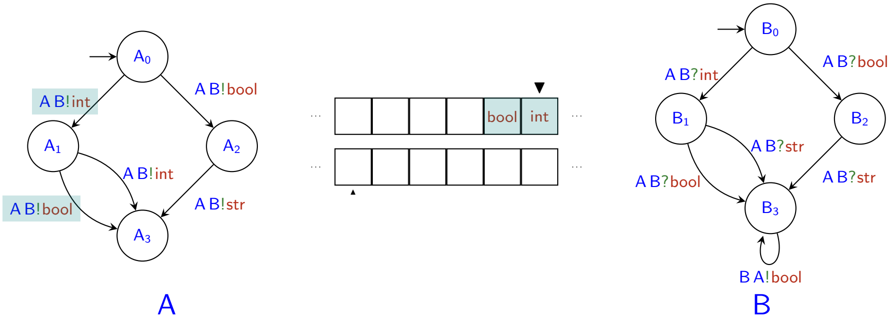
Global specs can be projected (i.e., compiled) on CFSMs

Setting-up a communication model

We're going to review some results about a specific communication model

- channel-based
- asynchronous (most often)
- point-to-point

Communicating Finite-State Machines



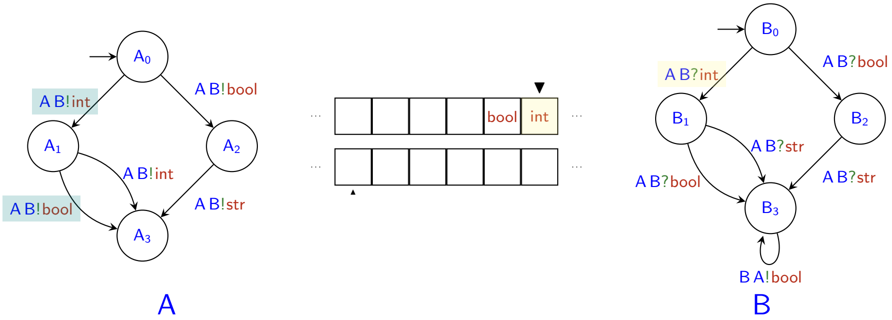
Global specs can be projected (i.e., compiled) on CFSMs

Setting-up a communication model

We're going to review some results about a specific communication model

- channel-based
- asynchronous (most often)
- point-to-point

Communicating Finite-State Machines



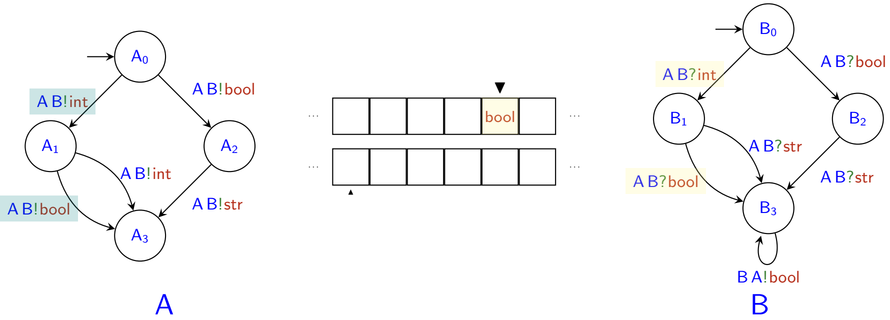
Global specs can be projected (i.e., compiled) on CFSMs

Setting-up a communication model

We're going to review some results about a specific communication model

- channel-based
- asynchronous (most often)
- point-to-point

Communicating Finite-State Machines



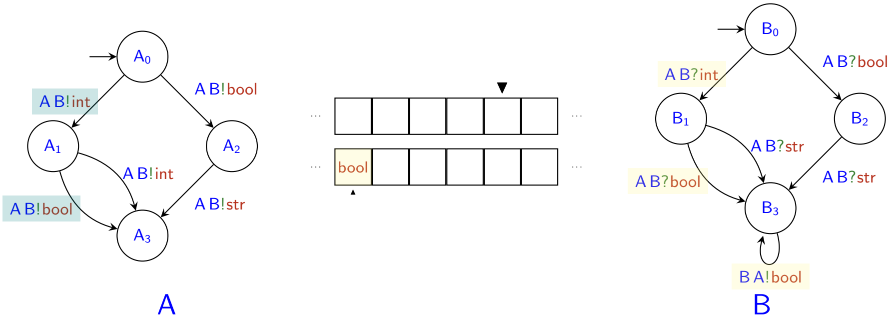
Global specs can be projected (i.e., compiled) on CFSMs

Setting-up a communication model

We're going to review some results about a specific communication model

- channel-based
- asynchronous (most often)
- point-to-point

Communicating Finite-State Machines



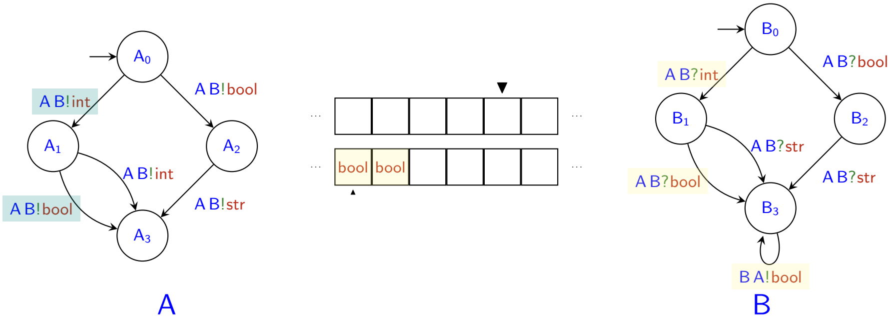
Global specs can be projected (i.e., compiled) on CFSMs

Setting-up a communication model

We're going to review some results about a specific communication model

- channel-based
- asynchronous (most often)
- point-to-point

Communicating Finite-State Machines



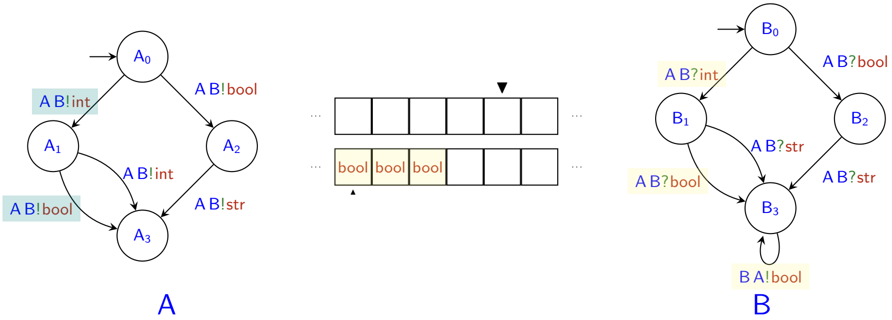
Global specs can be projected (i.e., compiled) on CFSMs

Setting-up a communication model

We're going to review some results about a specific communication model

- channel-based
- asynchronous (most often)
- point-to-point

Communicating Finite-State Machines



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

An obvious (fundamental) question

Given a global specification, is it realisable distributively?

Examples

Not all specs can be “faithfully” executed distributively...

Examples

Not all specs can be "faithfully" executed distributively...

Trivial non-realisability

$$\mathsf{A}\ \mathsf{B}?\mathsf{m} \longrightarrow \mathsf{B}\ \mathsf{C}?\mathsf{n}$$

Examples

Not all specs can be "faithfully" executed distributively...

Trivial non-realisability

A B?m ⟶ B C?n

Communicating systems "start" with outputs!

Examples

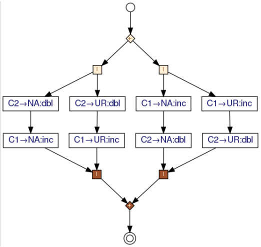
Not all specs can be "faithfully" executed distributively...

Trivial non-realisability

A B?m ⟶ B C?n

Communicating systems "start" with outputs!

Non-trivial non-realisability



[Alur et al. 2003]

Realisability

Put simply...

A global spec G is realizable if there is a deadlock-free[a] communicating system whose language "has 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.

Realisability

Put simply...

A global spec G is realizable if there is a deadlock-free[a] communicating system whose language "has 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.

A recipe for theorems

1. Define projections and the semantics of global and local specs
2. Show the global spec G is well-formed (for some def of well-formedness)
3. Show that G and its projections have a "suitable" relation

Realisability

Put simply...

A global spec G is realizable if there is a deadlock-free[a] communicating system whose language "has 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.

A recipe for theorems

1. Define projections and the semantics of global and local specs
2. Show the global spec G is well-formed (for some def of well-formedness)
3. Show that G and its projections have a "suitable" relation

Some instances

- G well-formed iff $\mathcal{L}(G)$ closed; then usual projections yield a language included in $\mathcal{L}(G)$ [GT19]
- G whole-spectrum iff G cannot drop mandatory beh.; then projections cover G [BMT20]
- G well-asserted iff G temporal satisfiable & history sensitive; then projections simulates G [BHTY10]

A (main) source of problems: Well-formedness (intuitively)

Distributed consensus

In a distributed choice $\mathsf{G}_1 + \mathsf{G}_2 + \cdots$

- there should be one active participant
- any non-active participant should be passive decides which branch to take in a choice

A (main) source of problems: Well-formedness (intuitively)

Distributed consensus

In a distributed choice $G_1 + G_2 + \cdots$

- there should be one active participant
- any non-active participant should be passive decides which branch to take in a choice

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 through the information received on each branch

A (main) source of problems: Well-formedness (intuitively)

Distributed consensus

In a distributed choice $G_1 + G_2 + \cdots$

- there should be one active participant
- any non-active participant should be passive decides which branch to take in a choice

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 through the information received on each branch

Well-branchedness

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

Class test

Figure out the graphical structure[2] of the following terms and for each of them say which one is well-branched

- $G_1 = \mathsf{A} \to \mathsf{B} : \mathsf{int} + \mathsf{A} \to \mathsf{B} : \mathsf{str}$
- $G_2 = \mathsf{A} \to \mathsf{B} : \mathsf{int} + \mathsf{A} \to \mathsf{C} : \mathsf{str}$
- $G_3 = \begin{pmatrix} \mathsf{A} \to \mathsf{C} : \mathsf{int}; \mathsf{A} \to \mathsf{B} : \mathsf{bool} \\ + \\ \mathsf{A} \to \mathsf{C} : \mathsf{str}; \mathsf{A} \to \mathsf{C} : \mathsf{bool}; \mathsf{A} \to \mathsf{B} : \mathsf{bool} \end{pmatrix} ; \mathsf{B} \to \mathsf{D} : \mathsf{str}$

[2] $\_;\_$ has precedence over $\_ + \_$

Class test

Figure out the graphical structure[2] of the following terms and for each of them say which one is well-branched

- $G_1 = A \to B : \text{int} + A \to B : \text{str}$
- $G_2 = A \to B : \text{int} + A \to C : \text{str}$
- $G_3 = \left(\begin{array}{l} A \to C : \text{int}; A \to B : \text{bool} \\ + \\ A \to C : \text{str}; A \to C : \text{bool}; A \to B : \text{bool} \end{array} \right); B \to D : \text{str}$

[2] _; _ has precedence over _ + _

Class test

Figure out the graphical structure[2] of the following terms and for each of them say which one is well-branched

- $G_1 = A \to B : int + A \to B : str$ 
- $G_2 = A \to B : int + A \to C : str$ 
- $G_3 = \begin{pmatrix} A \to C : int; A \to B : bool \\ + \\ A \to C : str; A \to C : bool; A \to B : bool \end{pmatrix} ; B \to D : str$ 

[2] $\_ ; \_$ has precedence over $\_ + \_$

– Act II –

[Some open problems]

– Scene 1 –

[Beyond holistic global specs]

Problem I(a): compositionality

How to compose global specs so to preserve "good" properties?

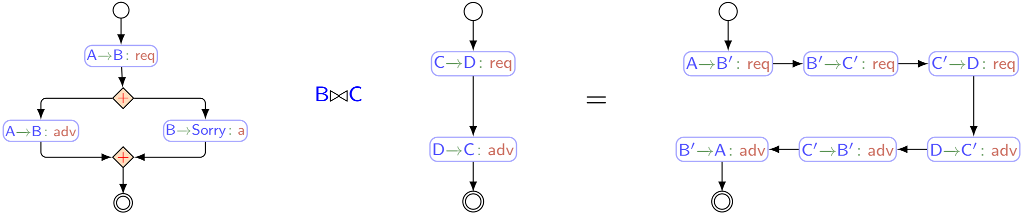
- Projections support modularity of local specs
- Global specs are typically holistic
  - Compositionality of global spec is harder
  - It is not clear what is to be used as an *interface*

An attempt: Preserving (dead)lock-freedom [BDLT20,BLT20]

Idea: compatible interface + gateways

A simple example

Composing global specs[a]



where

- Band C are the interfaces
- proj(B) compatible with dual(proj(C)) ... once channels are forgotten
- Band C are replaced by their gateways B' and C'

[a]We are also looking at similar results for local specs

Some initial results

- (Dead)lock-freedom, compositionally
  - Typable systems are lock-free &
  - $\text{-}\bowtie\text{-}$ preserves lock-freedom
  - $\implies$ the composition of typable systems is lock-free
- Gateways may be "merged" (semi-direct composition) or even removed (direct composition)!
- Oddly, the synchronous case for local specs is more involved than the asynchronous one

Refining global specs [dLMT20]

E. W. Dijkstra: Notes on Structured Programming

"The basic pattern of my approach will be to compose the program in minute steps, deciding each time as little as possible. As the problem analysis proceeds, so does the further refinement of my program"

Problem I(b): refinement

How to support setp-wise refinement of choreographies?

A simple idea

Adding refinable (and multiple) interaction:

$$\mathsf{G} \ ::= \ \cdots \mid \mathsf{A} \xrightarrow{\mathsf{m_1 \ldots m_n}} \mathsf{B_1 \ldots B_n} \qquad \text{where } n > 0$$

to be replaced by a well-formed *ground* $\hat{\mathsf{G}}$ such that

- unique initiator: A executes any first communication in $\hat{\mathsf{G}}$
- eventual reception: for all $1 \leq i \leq n$, the last action of B_i in any branch of $\hat{\mathsf{G}}$ is an input of message m_i

Examples

Which are legal refinements of the following?

$$C \overset{\text{md}}{\dashrightarrow} S + C \overset{\text{req}}{\dashrightarrow} S; S \overset{\text{done}}{\dashrightarrow} C$$

Sound refinements may be "wrong":

- $C \to S : \text{md} + C \to S : \text{req}; (S \to C : \text{stats}; S \to C : \text{done})$

Examples

Which are legal refinements of the following?

$$\mathsf{C} \overset{\mathsf{md}}{\dashrightarrow} \mathsf{S} + \mathsf{C} \overset{\mathsf{req}}{\dashrightarrow} \mathsf{S}; \mathsf{S} \overset{\mathsf{done}}{\dashrightarrow} \mathsf{C}$$

Sound refinements may be "wrong":

- $\mathsf{C} \rightarrow \mathsf{S} : \mathsf{md} + \mathsf{C} \rightarrow \mathsf{S} : \mathsf{req}; (\mathsf{S} \rightarrow \mathsf{C} : \mathsf{stats}; \mathsf{S} \rightarrow \mathsf{C} : \mathsf{done})$
- $(\mathsf{C} \rightarrow \mathsf{B} : \mathsf{md}; \mathsf{B} \rightarrow \mathsf{S} : \mathsf{md}) + \mathsf{C} \rightarrow \mathsf{S} : \mathsf{req}; (\mathsf{S} \rightarrow \mathsf{C} : \mathsf{stats}; \mathsf{S} \rightarrow \mathsf{C} : \mathsf{done})$

Examples

Which are legal refinements of the following?

$$C \overset{\text{md}}{\dashrightarrow} S + C \overset{\text{req}}{\dashrightarrow} S; S \overset{\text{done}}{\dashrightarrow} C$$

Sound refinements may be "wrong":

- $C \to S : \text{md} + C \to S : \text{req}; (S \to C : \text{stats}; S \to C : \text{done})$
- $(C \to B : \text{md}; B \to S : \text{md}) + C \to S : \text{req}; (S \to C : \text{stats}; S \to C : \text{done})$
- $(C \to B : \text{md}; B \to S : \text{md}) + (C \to B : \text{start}; B \to S : \text{req}); (S \to C : \text{stats}; S \to C : \text{done})$

Checking refinements

Idea

Devise a typying discipline sound w.r.t. well-formedness

Typing judgement $\quad \Pi \vdash \mathsf{G} : \langle \phi, \Lambda \rangle$

where

- Π are the participants in G,
- ϕ and Λ are the minimal and maximal actions in G

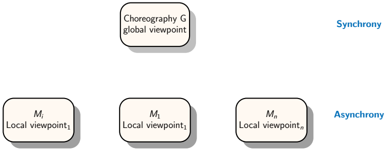
Preliminary results

- Ground specs have unique type
- Typable ground global specs are well-formed,
- ...but the vice versa does not hold
- Type inference is decidable for ground specs,
- ...but this is open for refinable specs

– Scene 2 –

[Beyond top-down development]

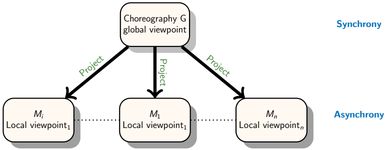
"Top-down"



Software

- Correctness-by-Design makes a lot of sense when going top-down

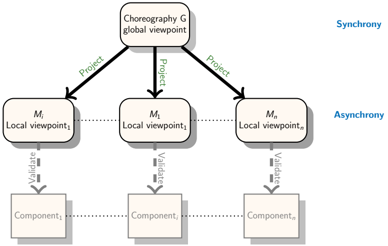
“Top-down”



Software

- Correctness-by-Design makes a lot of sense when going top-down

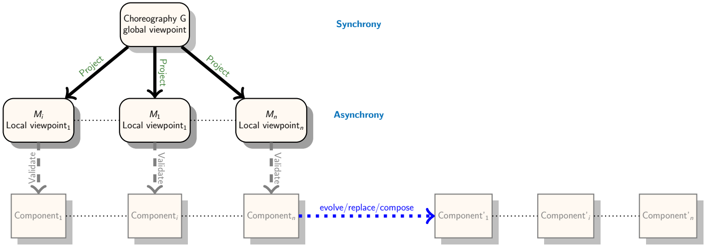
“Top-down”



Software

- Correctness-by-Design makes a lot of sense when going top-down

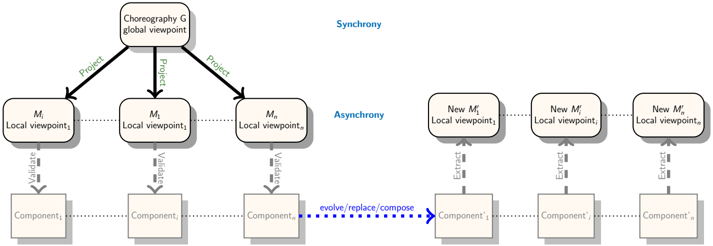
“Top-down”



Software evolves

- Correctness-by-Design makes a lot of sense when going top-down

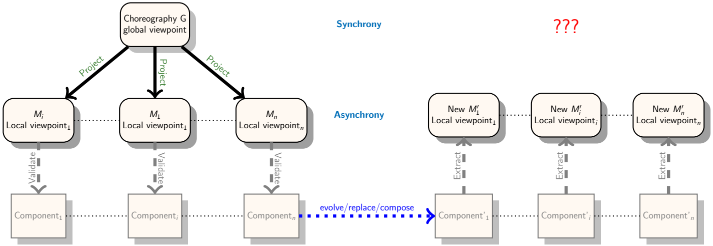
“Top-down” & “Bottom-up” approach



Software evolves

- Correctness-by-Design makes a lot of sense when going top-down

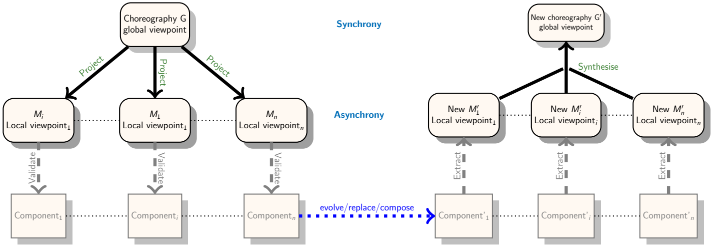
"Top-down" & "Bottom-up" approach



Software evolves

- Correctness-by-Design makes a lot of sense when going top-down

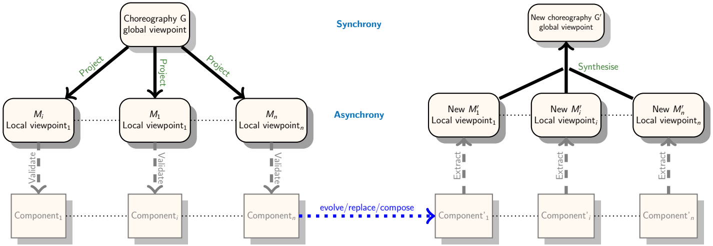
“Top-down” & “Bottom-up” approach



Software evolves

- Correctness-by-Design makes a lot of sense when going top-down

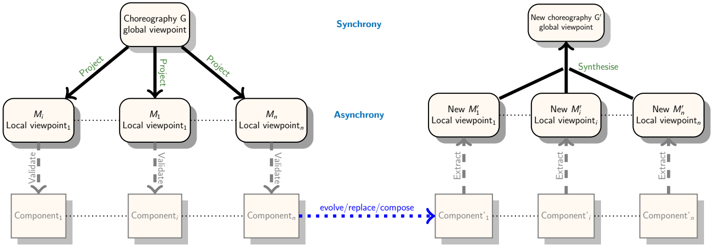
“Top-down” & “Bottom-up” approach



Software evolves

- Correctness-by-Design makes a lot of sense when going top-down
- $\pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$ [Heraclitus 6th century BC]

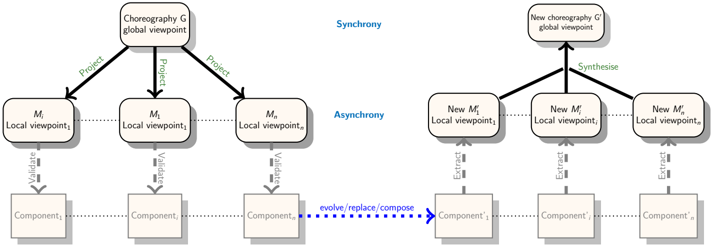
“Top-down” & “Bottom-up” approach



Software evolves

- Correctness-by-Design makes a lot of sense when going top-down
- $\pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$ [Heraclitus 6th century BC]
- Choreographies may help also going bottom-up [LTY15]

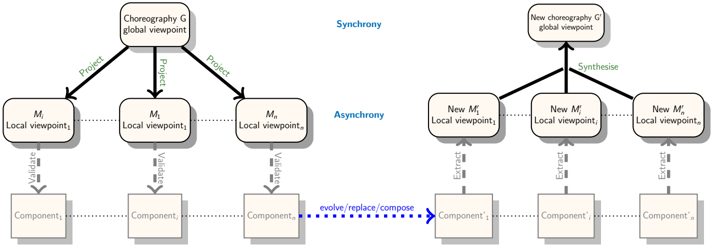
“Top-down” & “Bottom-up” approach



Software evolves

- Correctness-by-Design makes a lot of sense when going top-down
- $\pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$ [Heraclitus 6th century BC]
- Choreographies may help also going bottom-up [LTY15]

“Top-down” & “Bottom-up” approach



Software evolves

- Correctness-by-Design makes a lot of sense when going top-down
- $\pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$ [Heraclitus 6th century BC]
- Choreographies may help also going bottom-up [LTY15]

Problem II(a): harnessing round-trip engineering

Are there more usages of global specs than for projecting local specs?

Quite some work for binary ST

- Mezzina. How to Infer Finite Session Types in a Calculus of Services and Sessions. Coordination 2008.
- Collingbourne, Kelly. Inference of Session Types From Control Flow. ENTCS 238 (2010)
- Imai, Yuen. Session Type Inference in Haskell. PLACES 2010.
- Graversen, Harbo, Hüttel, Bjerregaard, Poulsen, Wahl. Type Inference for Session Types in the π-calculus. WS-FM 2016.
- Spaccasassi, Koutavas. Type-based Analysis for Session Inference. FORTE 2016.
- Lindley, Morris. Lightweight Functional Session Types. In Behavioural Types: from Theory to Tools. 2018.
- Padovani. Context-Free Session Type Inference. TOPLAS, 41. 2019
- ...

Does retrieving global specs matter?

Some good reasons

- Analysis
- Program comprehension
- Systematic way of documenting software
- Reuse of software

Problems

- Type inference is not all: it requires source code
- Process mining / model learning
  - Analysis / Comparison of protocols [TTWD16]
  - Adaptation: "incompatible" components can be adapted (e.g., with coordination delegates [AIT18,ADGPT19])

– Scene 3 –

[Choreographic-driven testing]

Is correcteness-by-construction sufficient?

- Local computations deal with data.

Example: $G_{\text{fact}} = \text{C} \to \text{S} : \text{Req}\ \text{int}; \text{S} \to \text{C} : \text{Res}\ \text{int}$

$$\text{factorialServer}(\text{Req}, \text{Res}) = \text{Req}?n.\text{Res}!\text{fact}(n) \qquad \text{where}$$

$$\text{fact} : \text{int} \to \text{int} \qquad \text{fact}(\texttt{int}\ \ n) = \texttt{if}\ \ 0 \leq n \leq 1\ \texttt{then}\ \ 1\ \texttt{else}\ \ n * \text{fact}(\ \text{n} + 1\ )$$

Is correcteness-by-construction sufficient?

- Local computations deal with data.

Example: $G_{\mathsf{fact}} = \mathsf{C} \to \mathsf{S} : \mathsf{Req}\ \mathsf{int}; \mathsf{S} \to \mathsf{C} : \mathsf{Res}\ \mathsf{int}$

$\mathsf{factorialServer}(\mathsf{Req}, \mathsf{Res}) = \mathsf{Req}?n.\mathsf{Res}!\mathsf{fact}(n)$ where

$\mathsf{fact} : \mathsf{int} \to \mathsf{int} \qquad \mathsf{fact}(\mathtt{int}\ n) = \mathtt{if}\ 0 \leq n \leq 1\ \mathtt{then}\ 1\ \mathtt{else}\ n * \mathsf{fact}(\ \mathsf{n} - 1\ )$

Is correcteness-by-construction sufficient?

- Local computations deal with data.

Example: $G_{\mathsf{fact}} = \mathsf{C} \to \mathsf{S} : \mathsf{Req}\ \mathsf{int}; \mathsf{S} \to \mathsf{C} : \mathsf{Res}\ \mathsf{int}$

$\mathsf{factorialServer}(\mathsf{Req}, \mathsf{Res}) = \mathsf{Req}?n.\mathsf{Res}!\mathsf{fact}(n)$ where

$\mathsf{fact} : \mathsf{int} \to \mathsf{int} \qquad \mathsf{fact}(\mathtt{int}\ n) = \mathtt{if}\ 0 \leq n \leq 1\ \mathtt{then}\ 1\ \mathtt{else}\ n * \mathsf{fact}(\mathsf{n} - 1)$

...and this is still not right! [BMT20,BHTY10]

Is correcteness-by-construction sufficient?

- Local computations deal with data.

Example: $G_{fact} = C \to S : \text{Req int}; S \to C : \text{Res int}$

$\text{factorialServer}(\text{Req}, \text{Res}) = \text{Req}?n.\text{Res}!\text{fact}(n)$ where

$\text{fact} : \text{int} \to \text{int} \qquad \text{fact}(\texttt{int}\ n) = \texttt{if}\ 0 \leq n \leq 1\ \texttt{then}\ 1\ \texttt{else}\ n * \text{fact}(\ n - 1\ )$

...and this is still not right! [BMT20,BHTY10]

- Evolution of components may alter communication patterns

Is correcteness-by-construction sufficient?

- Local computations deal with data.

Example: $G_{fact} = C \to S : \mathsf{Req}\ \mathsf{int}; S \to C : \mathsf{Res}\ \mathsf{int}$

$\mathsf{factorialServer}(\mathsf{Req}, \mathsf{Res}) = \mathsf{Req}?n.\mathsf{Res}!\mathsf{fact}(n)$ where

$\mathsf{fact} : \mathsf{int} \to \mathsf{int} \qquad \mathsf{fact}(\texttt{int}\ n) = \texttt{if}\ 0 \leq n \leq 1\ \texttt{then}\ 1\ \texttt{else}\ n * \mathsf{fact}(\mathsf{n-1})$

...and this is still not right! [BMT20,BHTY10]

- Evolution of components may alter communication patterns
- Openness enables changes to the execution context that may alter "compatibility"

Example: another server

$$\mathsf{factorialServer}(\mathsf{Req}, \mathsf{Res}) = \mathsf{Req}?n.\ \texttt{if}\ n < 0\ \texttt{then}\ \mathsf{Res}!\text{"error"}\ \texttt{else}\ \mathsf{Res}!\mathsf{fact}(n)$$

Problem II(b): harnessing round-trip engineering

Can global specs support software testing?

Problem II(b): harnessing round-trip engineering

Can global specs support software testing?

Choreographic models can be used

- as test case specifications
- to automatically generate executable tests
- to automatically generate mock components
- to assess coverage of test cases

Preliminary results [CGT20]

An abstract framework for model-based testing:

- Test cases: a composition of "some deterministic" CFSMs
- Automatic test generation

  $\prod(\text{split}(\text{proj}(\text{global spec})))$
- Test compliance: a criterion for test success (oracle problem)
- Suitable tests (not all tests make sense!)
- Theorem:

  if the global spec is well-formed then generated tests are suitable

– Scene 4 –

[Beyond Channel-based communication]

Abstract communication paradigms

Channel based communication could be too "low level"
Often other mechanisms are more appropriate

- Event-Notification
- Publish-Subscribe
- Generative communication
  - Distributed tuple spaces
  - Attribute-based

Abstract communication paradigms

Channel based communication could be too "low level"
Often other mechanisms are more appropriate

- Event-Notification
- Publish-Subscribe
- Generative communication
  - Distributed tuple spaces
  - Attribute-based

Problem III: Abstract coordination mechanisms

Develop new choreographic frameworks for sophisticated communication mechanism

A few (natural) questions

1. What safe assumptions on the (distributed) state after interactions?
2. What (behavioural) properties a given communication mechanism enforces?
3. How about statically guaranteeing such properties?
4. What are the relations between message-passing and more abstract communications?
5. Can behavioural abstractions support or improve run-time execution?
6. Can behavioural specifications foster quantitative analysis of CAS?

Drifting away from control-flow... [BCGMMT19,FMMT20,ITT20]

The emphasis is no longer on (dead)lock-freedom: progress becomes data-driven

Generalised interactions

$$\mathsf{A}_{|}\rho \xrightarrow{\; e \qquad e' \;} \mathsf{B}_{|}\rho'$$

any A *satisfying* ρ *generates data* e *for any* B *satisfying* ρ' *with* e' *matching* e.

Some benefits

- Weaker (hence more general) notions of correctness
- Choreographies for new domains (e.g., IoT, CPS, Autonomous Systems)
  - multi-roles: many instances may play many roles
  - correctness related to emergent behaviour
  - (limited) misbehaviour is tollerated

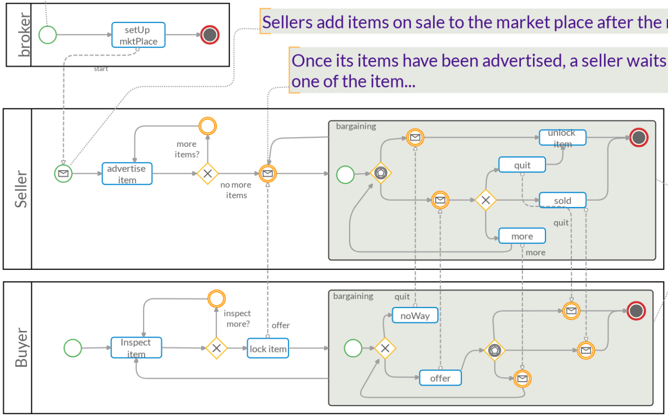
Some illustrative example

Market place

The broker triggers the protocol; for simplicity it does not take part in the rest of the choreography

Sellers add items on sale to the market place after the notification from the broker

Once its items have been advertised, a seller waits for buyers to flag interest in one of the item...

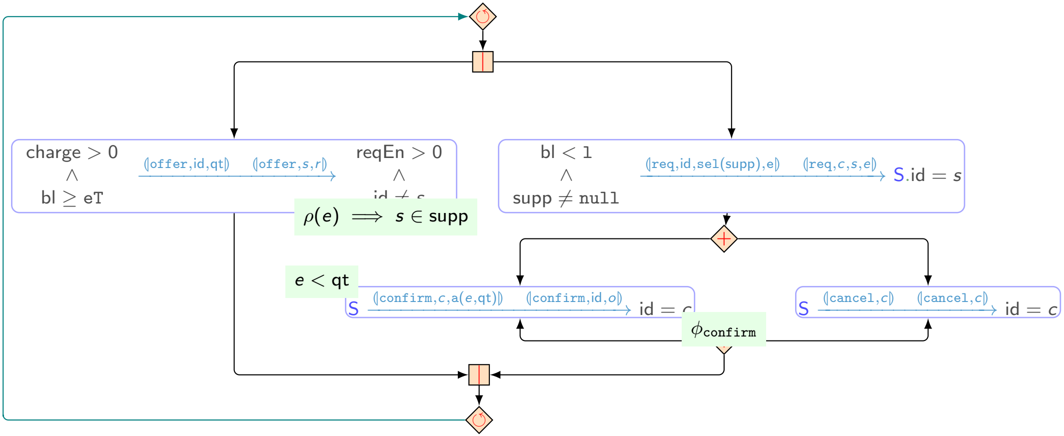


...when a buyer manifests interest in an item of a seller, buyer and seller start a subprocess for the bargaining phase. In this phase the buyer makes an offer and the seller decides if to quit the bargaining, ask for a better offer, or sell the item to the buyer

BEEPMN www.beepmn.com

Some illustrative example

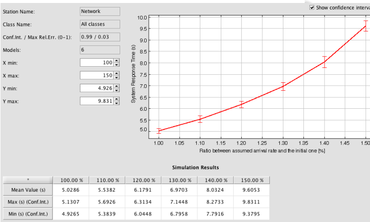
Robots: possibly playing two roles



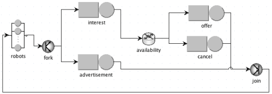
A positive side effect

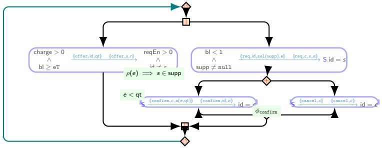
"Data-driven" specs seem more faithful to actual implementations

- going beyond simulations
- from global specs to queuing networks



*	100.00 %	110.00 %	120.00 %	130.00 %	140.00 %	150.00 %
Mean Value (s)	5.0286	5.5382	6.1791	6.9703	8.0324	9.6053
Max (s) (Conf.Int.)	5.1307	5.6926	6.3134	7.1448	8.2733	9.8311
Min (s) (Conf.Int.)	4.9265	5.3839	6.0448	6.7958	7.7916	9.3795





– Epilogue –

[...]

Summing up

A quick journey in choreographies in order to discuss

In order to focus on some open issues

- Compositionality
- Refinement
- Choreographic-driven Testing
- Generalisations

I resisted to talk about tool support (a crucial open problem in BehAPI)

References

I am immensely grateful to my collaborators

- [BDLT20] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ivan Lanese, eM.
  Composition and Decomposition of Multiparty Session. Submitted at JLAMP.
- [BLT20] Franco Barbanera, Ivan Lanese, eM.
  Composing Communicating Systems, Synchronously. ISoLA 2020
- [BMT20] Laura Bocchi, Hernán C. Melgratti, eM.
  Resolving Non-determinism in Choreographies. ESOP 2014. (Full version To appear on LMCS)
- [CGT20] Alex Coto, Roberto Guanciale, eM.
  An Abstract Framework for Choreographic Testing. ICE 2020 (To appear).
- [dLMT20] Ugo de'Liguoro, Hernán C. Melgratti, eM.
  Towards Refinable Choreographies. ICE 2020 (To appear).
- [FMMT20] Leonardo Frittelli, Facundo Maldonado, Hernán C. Melgratti, eM.
  A Choreography-Driven Approach to APIs: The OpenDXL Case Study. COORDINATION 2020
- [ITT20] Omar Inverso, Catia Trubiani, eM.
  Abstractions for Collective Adaptive Systems. (ISoLA 2020)
- [BHTY10] Laura Bocchi, Kohei Honda, eM, Nobuko Yoshida.
  A Theory of Design-by-Contract for Distributed Multiparty Interactions. CONCUR 2010
- [BCGMMT19] Roberto Bruni, Andrea Corradini, Fabio Gadducci, Hernán C. Melgratti, Ugo Montanari, eM.
  Data-Driven Choreographies à la Klaim. Models, Languages, and Tools for Concurrent and Distributed Programming 2019.
- [GT19] Roberto Guanciale, eM.
  Realisability of pomsets. J. Log. Algebraic Methods Program. 108 (2019)
- [TTWD16] Ramsey Taylor, eM, Neil Walkinshaw, John Derrick.
  Choreography-Based Analysis of Distributed Message Passing Programs. PDP 2016
- [LTY15] Julien Lange, eM, Nobuko Yoshida
  From Communicating Machines to Graphical Choreographies. POPL 2015

[Thank you!]