# *reasoning about choreographic programs*

luís cruz-filipe

*with eva graversen, fabrizio montesi & marco peressotti*

department of mathematics and computer science
university of southern denmark

concurrency and logic seminar
june 27th, 2023

# *choreographic programming, conceptually*

### *what are choreographies?*

high-level global specifications of concurrent and distributed systems

### *a programming paradigm with good properties*

implementations for the local endpoints are automatically generated

- guaranteed to be deadlock-free
- guaranted to satisfy the specification

# an example

## authentication choreography

```
X = c.credentials --> ip.x;
    If ip.(check x)
    Then ip --> c[left]; ip.go --> s.b; s.token --> c.t
    Else ip --> c[right]; X
```

# an example

### authentication choreography

```
X = c.credentials --> ip.x;
    If ip.(check x)
    Then ip --> c[left]; ip.go --> s.b; s.token --> c.t
    Else ip --> c[right]; X
```

### the problem

how can we reason formally about what this choreography does?

- currently: *ad-hoc* properties can be proved by simulating execution
- better: have a more general framework for reasoning about choreographies

# *our contribution*

we propose a sound and complete hoare calculus for a simple choreography language

# our contribution

we propose a sound and complete hoare calculus for a simple choreography language

## design choices

avoid *ad-hoc* solutions

- use a pre-existing choreography language
  $\leadsto$ for the good and for the bad
- design a “standard” hoare calculus
- separation of concerns
- emphasis on parameterisation

# a simple choreography language

## syntax

choreography bodies are defined by the following grammar

$$C ::= I; C \mid \text{if } \mathsf{p}.b \text{ then } C_1 \text{ else } C_2 \mid X \mid \lceil \vec{\mathsf{q}}, X \rfloor C \mid \mathbf{0}$$
$$I ::= \mathsf{p}.x := e \mid \mathsf{p}.e \to \mathsf{q}.x \mid \mathsf{p} \to \mathsf{q}[\mathrm{L}]$$

- $\mathsf{p}.x := e$: local computation
- $\mathsf{p}.e \to \mathsf{q}.x$: value communication
- $\mathsf{p} \to \mathsf{q}[\mathrm{L}]$: label selection
- $X$: procedure call
- $\lceil \vec{\mathsf{q}}, X \rfloor C$: runtime term for partially entered procedures

# a simple choreography language

## syntax

choreography bodies are defined by the following grammar

$$C ::= I; C \mid \text{if } \mathsf{p}.b \text{ then } C_1 \text{ else } C_2 \mid X \mid \lceil \vec{\mathsf{q}}, X \rfloor C \mid \mathbf{0}$$
$$I ::= \mathsf{p}.x := e \mid \mathsf{p}.e \to \mathsf{q}.x \mid \mathsf{p} \to \mathsf{q}[\mathrm{L}]$$

## choreographic programs

$\langle \mathscr{C}, C \rangle$ where $\mathscr{C}$ maps procedure names to their definitions

# semantics

## ltl semantics

configurations are pairs $\langle C, \Sigma \rangle$ where $\Sigma$ is a memory state

- $\Sigma(\mathsf{p})(x)$ returns the value stored at p's variable $x$
- $e \downarrow_{\Sigma(\mathsf{p})} v$ returns the result of locally evaluating $e$ at p using $\Sigma$

# semantics

## ltl semantics

configurations are pairs $\langle C, \Sigma \rangle$ where $\Sigma$ is a memory state

## the rules

three groups of rules

- formalisation of the intuition behind the constructs, e.g. if $e \downarrow_{\Sigma(\mathsf{p})} v$, then

$$\langle \mathsf{p}.x := e; C, \Sigma \rangle \xrightarrow{\tau @ \mathsf{p}}_{\mathscr{C}} \langle C, \Sigma[\langle \mathsf{p}, x \rangle \mapsto v] \rangle$$

## semantics

### ltl semantics

configurations are pairs $\langle C, \Sigma \rangle$ where $\Sigma$ is a memory state

### the rules

three groups of rules

- formalisation of the intuition behind the constructs
- formalisation of out-of-order execution, e.g. if $\langle C, \Sigma \rangle \xrightarrow{\mu}_{\mathscr{C}} \langle C', \Sigma' \rangle$ and $I$ does not involve processes appearing in $\mu$, then

$$\langle I; C, \Sigma \rangle \xrightarrow{\mu}_{\mathscr{C}} \langle I; C', \Sigma' \rangle$$

# semantics

## ltl semantics

configurations are pairs $\langle C, \Sigma \rangle$ where $\Sigma$ is a memory state

## the rules

three groups of rules

- formalisation of the intuition behind the constructs
- formalisation of out-of-order execution
- rules allowing processes to enter procedure calls independently
  $\leadsto$ use runtime terms

# the intuition

## general idea

- judgements are triples $\{\varphi\}C\{\psi\}$
  *if $C$ is executed from a state where $\varphi$ holds and execution terminates, then $\psi$ holds in the final state*
- main inference rules match the rules of semantics

# the intuition

## general idea

- judgements are triples $\{\varphi\}C\{\psi\}$
  *if $C$ is executed from a state where $\varphi$ holds and execution terminates, then $\psi$ holds in the final state*
- main inference rules match the rules of semantics

## three challenges

- what formulas can we write about states?

# the intuition

## general idea

- judgements are triples $\{\varphi\}C\{\psi\}$
  *if $C$ is executed from a state where $\varphi$ holds and execution terminates, then $\psi$ holds in the final state*
- main inference rules match the rules of semantics

## three challenges

- what formulas can we write about states?
- the usual rule for assignment

$$\frac{}{\{\varphi'\}\mathsf{p}.x := e; \mathbf{0}\{\varphi\}}$$

where $\varphi'$ is obtained from $\varphi$ by replacing $\mathsf{p}.x$ with $e$ does not work

# the intuition

## general idea

- judgements are triples $\{\varphi\}C\{\psi\}$
  *if $C$ is executed from a state where $\varphi$ holds and execution terminates, then $\psi$ holds in the final state*
- main inference rules match the rules of semantics

## three challenges

- what formulas can we write about states?
- the usual rule for assignment does not work
- how do we deal with procedure calls?

# the state logic

## an equational logic

- parameterised on the language of expressions in the choreography language
- variables are localised, e.g. p.$x$
- parameterised on a decidable theory $\mathfrak{D}$ whose terms include logical variables $\mathcal{X}$

# the state logic

## an equational logic

- parameterised on the language of expressions in the choreography language
- variables are localised, e.g. p.$x$
- parameterised on a decidable theory $\mathfrak{D}$ whose terms include logical variables $\mathcal{X}$

## syntax

$$\varphi ::= (\mathcal{E} = \mathcal{X}) \mid \delta \mid \varphi \wedge \varphi \mid \neg\varphi$$

# the state logic

## syntax

$$\varphi ::= (\mathcal{E} = \mathcal{X}) \mid \delta \mid \varphi \wedge \varphi \mid \neg\varphi$$

## semantics

given an assignment $\rho$ assigning values to logical variables:

$$\frac{\mathcal{E} \downarrow_{\Sigma} \rho(\mathcal{X})}{\Sigma \Vdash_{\rho} \mathcal{E} = \mathcal{X}} \qquad \frac{\delta \in \mathfrak{D} \quad \delta\rho \text{ is true}}{\Sigma \Vdash_{\rho} \delta}$$

# localisation

**definition**

- $L(\mathsf{p}, e)$ is the logical expression obtained from $e$ by replacing every choreography variable $x$ with $\mathsf{p}.x$
- $\mathcal{E}[\mathsf{q}.x := \mathsf{p}.e]$ is the expression obtained from $\mathcal{E}$ by replacing every occurrence of $\mathsf{q}.x$ with $L(\mathsf{p}, e)$
- localised substitution extends to formulas in the natural way

# localisation

## definition

- $L(\mathsf{p}, e)$ is the logical expression obtained from $e$ by replacing every choreography variable $x$ with $\mathsf{p}.x$
- $\mathcal{E}[\mathsf{q}.x := \mathsf{p}.e]$ is the expression obtained from $\mathcal{E}$ by replacing every occurrence of $\mathsf{q}.x$ with $L(\mathsf{p}, e)$
- localised substitution extends to formulas in the natural way

## an example

let $\varphi$ be $\mathsf{p}.x = \mathcal{X}$ and $e$ be $y - z$, then:

- $L(\mathsf{p}, y - z) = \mathsf{p}.y - \mathsf{p}.z$
- $\varphi[\mathsf{p}.x := \mathsf{p}.(y - z)]$ is $\mathsf{p}.y - \mathsf{p}.z = \mathcal{X}$

# localisation

## definition

- $L(\mathsf{p}, e)$ is the logical expression obtained from $e$ by replacing every choreography variable $x$ with $\mathsf{p}.x$
- $\mathcal{E}[\mathsf{q}.x := \mathsf{p}.e]$ is the expression obtained from $\mathcal{E}$ by replacing every occurrence of $\mathsf{q}.x$ with $L(\mathsf{p}, e)$
- localised substitution extends to formulas in the natural way

## properties

- if $\rho(\mathcal{X}) = v$: then $e \downarrow_{\Sigma(\mathsf{p})} v$ iff $\Sigma \Vdash_\rho L(\mathsf{p}, e) = \mathcal{X}$
- if $e \downarrow_{\Sigma(\mathsf{p})} v$: then $\Sigma[\langle \mathsf{p}, x\rangle \mapsto v] \Vdash_\rho \varphi$ iff $\Sigma \Vdash_\rho \varphi[\mathsf{q}.x := \mathsf{p}.e]$ for all $\rho$

# dealing with procedure calls

## procedure specification maps

to reason about a program we need a description of the behaviour of the procedures

$$\mathfrak{C}(X) = \langle \varphi_X, \psi_X \rangle$$

- intended meaning: if $\varphi_X$ holds when $X$ is called and execution terminates, then $\psi_X$ holds in the final state

# the rules, part i

## about instructions...

$$\frac{\vdash_{\mathfrak{C}} \{\varphi\} C \{\psi\}}{\vdash_{\mathfrak{C}} \{\varphi[\mathsf{p}.x := \mathsf{p}.e]\} \mathsf{p}.x := e; C \{\psi\}} \ \mathrm{H|A{\small SSIGN}}$$

$$\frac{\vdash_{\mathfrak{C}} \{\varphi\} C \{\psi\}}{\vdash_{\mathfrak{C}} \{\varphi[\mathsf{q}.x := \mathsf{p}.e]\} \mathsf{p}.e \to \mathsf{q}.x; C \{\psi\}} \ \mathrm{H|C{\small OM}}$$

$$\frac{\vdash_{\mathfrak{C}} \{\varphi\} C \{\psi\}}{\vdash_{\mathfrak{C}} \{\varphi\} \mathsf{p} \to \mathsf{q}[\mathrm{L}]; C \{\psi\}} \ \mathrm{H|S{\small EL}}$$

# *the rules, part ii*

### *. . . about compound choreographies. . .*

$$\frac{}{\vdash_{\mathfrak{C}} \{\varphi\}\mathbf{0}\{\varphi\}}\ \mathrm{H|N{\small IL}}$$

$$\frac{\begin{array}{c}\vdash_{\mathfrak{C}} \{\varphi \land L(\mathsf{p}, b) \stackrel{\mathcal{X}}{=} \mathsf{true}\}C_1\{\psi\} \\ \vdash_{\mathfrak{C}} \{\varphi \land L(\mathsf{p}, b) \stackrel{\mathcal{X}}{=} \mathsf{false}\}C_2\{\psi\}\end{array} \qquad \mathcal{X}\ \mathsf{fresh}}{\vdash_{\mathfrak{C}} \{\varphi\}\mathsf{if}\ \mathsf{p}.b\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2\{\psi\}}\ \mathrm{H|C{\small OND}}$$

$$\frac{\mathfrak{C}(X) = \langle\varphi, \psi\rangle}{\vdash_{\mathfrak{C}} \{\varphi\}X\{\psi\}}\ \mathrm{H|C{\small ALL}}$$

$$\frac{\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}}{\vdash_{\mathfrak{C}} \{\varphi\}\lceil\vec{\mathsf{q}}, X\rfloor C\{\psi\}}\ \mathrm{H|C{\small ALL}'}$$

# *the rules, part iii*

## *. . . and structural rules*

$$\frac{\mathfrak{D} \models \varphi \to \varphi' \quad \vdash_{\mathfrak{C}} \{\varphi'\}C\{\psi'\} \quad \mathfrak{D} \models \psi' \to \psi}{\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}}\ \mathrm{H|W{\small EAK}}$$

# auxiliary results

## head reductions

$\langle C, \Sigma \rangle \overset{\mu}{\underset{\mathscr{C}}{\Longrightarrow}} \langle C', \Sigma' \rangle$ denotes that $C$ reduces to $C'$ without using out-of-order execution

# auxiliary results

**head reductions**

$\langle C, \Sigma \rangle \overset{\mu}{\Rightarrow}_{\mathscr{C}} \langle C', \Sigma' \rangle$ denotes that $C$ reduces to $C'$ without using out-of-order execution

**confluence**

choreography execution is confluent

- in particular, if $\langle C, \Sigma \rangle \rightarrow^*_{\mathscr{C}} \langle \mathbf{0}, \Sigma' \rangle$ then also $\langle C, \Sigma \rangle \Rightarrow^*_{\mathscr{C}} \langle \mathbf{0}, \Sigma' \rangle$
- together with determinism, this allows us to focus only on head reductions

# soundness

**consistency**

$\mathfrak{C}$ is *consistent* with $\mathscr{C}$ if $\vdash_{\mathfrak{C}} \{\varphi_X\}\mathscr{C}(X)\{\psi_X\}$ for every $X$

# soundness

## consistency

$\mathfrak{C}$ is *consistent* with $\mathscr{C}$ if $\vdash_{\mathfrak{C}} \{\varphi_X\}\mathscr{C}(X)\{\psi_X\}$ for every $X$

## theorem

$$\left.\begin{array}{l} \bullet\ \mathfrak{C} \text{ is consistent with } \mathscr{C} \\ \bullet\ \vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\} \\ \bullet\ \Sigma \Vdash_{\rho} \varphi \\ \bullet\ \langle C, \Sigma\rangle \to^{*}_{\mathscr{C}} \langle \mathbf{0}, \Sigma'\rangle \end{array}\right\} \text{implies } \Sigma' \Vdash_{\rho} \psi$$

## weakest liberal preconditions

**definition**

$$
\begin{aligned}
\mathsf{wlp}_{\mathfrak{C}}((\mathsf{p}.x := e; C), \psi) &= \mathsf{wlp}_{\mathfrak{C}}(C, \psi)[\mathsf{p}.x := \mathsf{p}.e] \\
\mathsf{wlp}_{\mathfrak{C}}((\mathsf{p}.e \to \mathsf{q}.x; C), \psi) &= \mathsf{wlp}_{\mathfrak{C}}(C, \psi)[\mathsf{q}.x := \mathsf{p}.e] \\
\mathsf{wlp}_{\mathfrak{C}}((\mathsf{p} \to \mathsf{q}[\mathrm{L}]; C), \psi) &= \mathsf{wlp}_{\mathfrak{C}}(C, \psi) \\
\mathsf{wlp}_{\mathfrak{C}}(\text{if } \mathsf{p}.b \text{ then } C_1 \text{ else } C_2, \psi) &= (L(\mathsf{p}, b) \stackrel{\mathcal{X}}{=} \mathsf{true} \to \mathsf{wlp}_{\mathfrak{C}}(C_1, \psi)) \\
&\quad \wedge (L(\mathsf{p}, b) \stackrel{\mathcal{X}}{=} \mathsf{false} \to \mathsf{wlp}_{\mathfrak{C}}(C_2, \psi)) \\
\mathsf{wlp}_{\mathfrak{C}}(X, \psi) &= \varphi_X \\
\mathsf{wlp}_{\mathfrak{C}}(\lceil \vec{\mathsf{q}}, X \rfloor C, \psi) &= \mathsf{wlp}_{\mathfrak{C}}(C, \psi) \\
\mathsf{wlp}_{\mathfrak{C}}(\mathbf{0}, \psi) &= \psi
\end{aligned}
$$

$\rightsquigarrow$ essentially read the rules "backwards"

# partial completeness

**adequacy**

$\mathfrak{C}$ is *adequate for* $\psi$ given $\mathscr{C}$ if, for all $X$:

- $\varphi_X$ is equivalent to $\mathrm{wlp}_{\mathfrak{C}}(\mathscr{C}(X), \psi)$
- $\psi_X = \psi$

# partial completeness

**adequacy**

$\mathfrak{C}$ is *adequate for* $\psi$ given $\mathscr{C}$ if, for all $X$:

- $\varphi_X$ is equivalent to $\mathrm{wlp}_{\mathfrak{C}}(\mathscr{C}(X), \psi)$
- $\psi_X = \psi$

**lemma**

if $\mathfrak{C}$ is adequate for $\psi$ given $\mathscr{C}$, then $\mathfrak{C}$ is consistent with $\mathscr{C}$

$\rightsquigarrow$ can be combined with soundness

# partial completeness

## adequacy

$\mathfrak{C}$ is *adequate for* $\psi$ given $\mathscr{C}$ if, for all $X$:

- $\varphi_X$ is equivalent to $\mathrm{wlp}_{\mathfrak{C}}(\mathscr{C}(X), \psi)$
- $\psi_X = \psi$

## theorem

$$\left.\begin{array}{l}\bullet\ \mathfrak{C} \text{ is adequate for } \psi \text{ given } \mathscr{C} \\ \bullet\ \text{whenever } \Sigma \Vdash_\rho \varphi \text{ and} \\ \quad \langle C, \Sigma\rangle \to^*_{\mathscr{C}} \langle \mathbf{0}, \Sigma'\rangle, \text{ then } \Sigma' \Vdash_\rho \psi\end{array}\right\} \text{ implies } \vdash_{\mathfrak{C}} \{\varphi\} C \{\psi\}$$

# (un)decidability

*the best...*

the judgement $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$ is decidable

# (un)decidability

### the best. . .

the judgement $\vdash_{\mathfrak{C}} \{\varphi\} C \{\psi\}$ is decidable

### . . . the good. . .

if the set of procedure names is finite:

- consistency between $\mathfrak{C}$ and $\mathscr{C}$ is decidable
- adequacy of $\mathfrak{C}$ for $\psi$ and $\mathscr{C}$ is decidable

# (un)decidability

*the best…*

the judgement $\vdash_{\mathfrak{C}} \{\varphi\}C\{\psi\}$ is decidable

*…the good…*

if the set of procedure names is finite:

- consistency between $\mathfrak{C}$ and $\mathscr{C}$ is decidable
- adequacy of $\mathfrak{C}$ for $\psi$ and $\mathscr{C}$ is decidable

*…and the not-so-good*

there is no algorithm that, given $\mathscr{C}$ and $\psi$, always returns $\mathfrak{C}$ that is adequate for $\psi$ given $\mathscr{C}$

$\rightsquigarrow$ proof idea: $\mathrm{wlp}_{\mathfrak{C}}(X, \bot) = \top$ iff execution of $X$ always diverges

# a note on divergence

## the big minus

- our calculus only proves properties of terminating executions
- since we do not have sequential composition, the target formula in all judgements holds in the (same) final state

# a note on divergence

## the big minus

- our calculus only proves properties of terminating executions
- since we do not have sequential composition, the target formula in all judgements holds in the (same) final state

$\rightsquigarrow$ but hey, you gotta start somewhere!

# a note on divergence

## the big minus

- our calculus only proves properties of terminating executions
- since we do not have sequential composition, the target formula in all judgements holds in the (same) final state

## a closer look at consistency

- whenever $X$ is called, $\varphi_X$ must hold
- because of out-of-order execution, there is no guarantee that the choreography ever passes those points. . .

## a note on divergence

### the big minus

- our calculus only proves properties of terminating executions
- since we do not have sequential composition, the target formula in all judgements holds in the (same) final state

### a closer look at consistency

- whenever $X$ is called, $\varphi_X$ must hold
- because of out-of-order execution, there is no guarantee that the choreography ever passes those points. . .

### future work – explore this idea

- formally prove the informal statement above
- capitalise on confluence to get stronger results

# in a nutshell

- a hoare calculus for a simple choreography language
- agnostic, modular, generalisable
- soundness, partial completeness and (some) decidability
- potentially extendable to reasoning about non-terminating systems: liveness, reactiveness, . . .

thank you!