now it compiles!

automatic repair of choreographic programs

luís cruz-filipe

with fabrizio montesi

department of mathematics and computer science
university of southern denmark

itp'23
august 3rd, 2023

choreographic programming, conceptually

what are choreographies?

high-level global specifications of concurrent and distributed systems

a new programming paradigm

implementations for the local endpoints are automatically generated

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

an example

authentication choreography

```
X = c.credentials --> ip.x;
    If ip.(check x)
    Then ip --> s[ok]; ip --> c[ok]; s.token --> c.t
    Else ip --> s[ko]; ip --> c[ko]; X
```

an example

authentication choreography

```
X = c.credentials --> ip.x;
    If ip.(check x)
    Then ip --> s[ok]; ip --> c[ok]; s.token --> c.t
    Else ip --> s[ko]; ip --> c[ko]; X
```

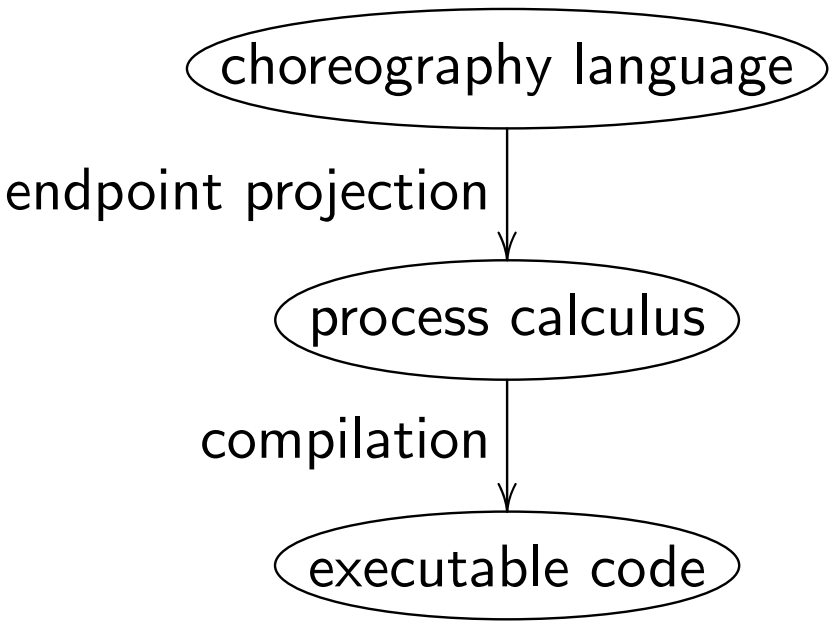
local implementations

X_c = ip!credentials; ip & {ok: s?t; ko: X_c }

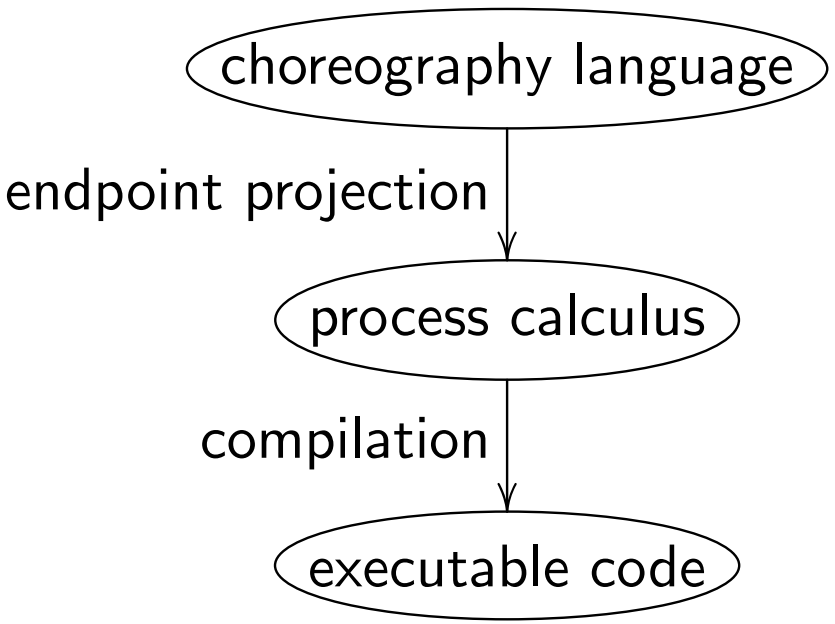
X_s = ip & {ok: c!token; ko: X_s }

X_{ip} = c?x; If (check x) Then (s(+)ok; c(+)ok)
Else (s(+)ko; c(+)ko; X_{ip})

a bird's-eye view



a bird's-eye view



- toolchain based on formalised projection (using coq)
- practical applications motivate extensions to the choreographic theory
- in this work: make the language simpler to use

why bother formalising?

choreographies are a popular topic...

- active research field
- many relevant applications
- potential in choreographic programming

why bother formalising?

choreographies are a popular topic...

- active research field
- many relevant applications
- potential in choreographic programming

...but there are some hiccups

process calculus and session types plagued by wrong proofs

- complex definitions, long proofs by structural induction
- situation pointed out at itp'15
  - formalisation of a published journal article
  - most proofs were wrong (but the theorems held)
- big revision of decidability results in the last few years
  - published proofs of both A and $\neg A$ for quite a few A...

our language

a minimal choreography language

- value communication

  ```
  p.e --> q.x
  ```

- label selections (for projection)

  ```
  p --> q[l]
  ```

- conditionals

  ```
  If p.b Then Ct Else Ce
  ```

- trailing procedure calls (for recursion)

  ```
  X
  ```

our language

a minimal choreography language

- value communication

  ```
  p.e --> q.x
  ```

- label selections (for projection)

  ```
  p --> q[l]
  ```

- conditionals

  ```
  If p.b Then Ct Else Ce
  ```

- trailing procedure calls (for recursion)

  ```
  X
  ```

agnostic language

- parametric on expressions and values
- only two labels

knowledge of choice

authentication choreography, wrong

```
X = c.credentials --> ip.x;
    If ip.(check x) Then s.token --> c.t
                    Else X
```

- choreographically, nothing wrong. . .

knowledge of choice

authentication choreography, wrong

```
X = c.credentials --> ip.x;
    If ip.(check x) Then s.token --> c.t
                    Else X
```

- choreographically, nothing wrong...
- ...but impossible to implement!

knowledge of choice (cont'd)

knowledge of choice

if a participant's behaviour depends on the result of a conditional evaluated elsewhere, then it must be notified of the outcome of the evaluation

- notification uses label selections
- notifications may be indirect (e.g. a notifies b who notifies c)

knowledge of choice (cont'd)

knowledge of choice

if a participant's behaviour depends on the result of a conditional evaluated elsewhere, then it must be notified of the outcome of the evaluation

- notification uses label selections
- notifications may be indirect (e.g. a notifies b who notifies c)

$\rightsquigarrow$ in our example, both client and server must be notified of the result of the authentication

choreography amendment

folklore

well-known facts

- selections can be inferred automatically
- adding them does not "significantly" change the protocol
- but it is useful to be able to place them manually

choreography amendment

folklore

well-known facts

- selections can be inferred automatically
- adding them does not "significantly" change the protocol
- but it is useful to be able to place them manually

"everyone knows that"

. . . does not mean that:

- it is true
- it has ever been proved

. . . likely means:

- no one proves it in detail
- no one checks proofs

our contribution

a formalisation of amendment for our choreography language

- definition of amendment in coq
- correspondence theorem
  (a.k.a. amendment does not “significantly” change the protocol)

our contribution

a formalisation of amendment for our choreography language

- definition of amendment in coq
- correspondence theorem
  (a.k.a. amendment does not "significantly" change the protocol)

surprise or not?

weaker correspondence than expected

amendment by example

authentication choreography *(amended)*

```
X = c.credentials --> ip.x;
    If ip.(check x)
    Then ip --> s[ok]; ip --> c[ok]; s.token --> c.t
    Else ip --> s[ko]; ip --> c[ko]; X
```

amendment by example

authentication choreography (amended)

```
X = c.credentials --> ip.x;
    If ip.(check x)
    Then ip --> s[ok]; ip --> c[ok]; s.token --> c.t
    Else ip --> s[ko]; ip --> c[ko]; X
```

amendment theorem

for every choreography C:

1. amend(C) is well-formed
2. amend(C) is projectable
3. $\langle C, \sigma \rangle \rightarrow^* \langle C', \sigma' \rangle$ iff $\langle \text{amend}(C), \sigma \rangle \rightarrow^* \langle \text{amend}(C'), \sigma' \rangle$

amendment by example

authentication choreography (amended)

```
X = c.credentials --> ip.x;
    If ip.(check x)
    Then ip --> s[ok]; ip --> c[ok]; s.token --> c.t
    Else ip --> s[ko]; ip --> c[ko]; X
```

amendment theorem

for every choreography C:

1. amend(C) is well-formed
2. amend(C) is projectable
3. $\langle C, \sigma \rangle \rightarrow^* \langle C', \sigma' \rangle$ iff $\langle \text{amend}(C), \sigma \rangle \rightarrow^* \langle \text{amend}(C'), \sigma' \rangle$

the subtle problem

in a truly distributed setting we cannot enforce causal dependencies between independent actions:

```
p.1 --> q.x; r.2 --> s.y
```

has two different reduction paths, since the two communications can happen in any order

the subtle problem

in a truly distributed setting we cannot enforce causal dependencies between independent actions:

```
p.1 --> q.x; r.2 --> s.y
```

has two different reduction paths, since the two communications can happen in any order

out-of-order execution

semantic rules allowing to reduce *inside* a choreography (under suitable conditions)

a dangerous mix

amendment can introduce new dependencies:

original choreography

```
p.e --> q.x;
If r.b Then r.e' --> p.y; End
       Else End
```

becomes

amended choreography

```
p.e --> q.x;
If r.b Then r --> p[ok]; r.e' --> p.y; End
       Else r --> p[ko]; End
```

a dangerous mix

amendment can introduce new dependencies:

original choreography

```
p.e --> q.x;
If r.b Then r.e' --> p.y; End
       Else End
```

becomes

amended choreography

```
p.e --> q.x;
If r.b Then r --> p[ok]; r.e' --> p.y; End
       Else r --> p[ko]; End
```

a dangerous mix

amendment can introduce new dependencies:

original choreography

```
p.e --> q.x;
If r.b Then r.e' --> p.y; End
       Else End
```

becomes

amended choreography

```
p.e --> q.x;
If r.b Then r --> p[ok]; r.e' --> p.y; End
       Else r --> p[ko]; End
```

penance time

interaction between amendment and out-of-order execution went unnoticed for a long time

- one conference publication (proofs in unpublished appendix)
- one journal publication (proof sketch)

penance time

interaction between amendment and out-of-order execution went unnoticed for a long time

- one conference publication (proofs in unpublished appendix)
- one journal publication (proof sketch)

. . . not because of sloppiness (we thought we had checked it)

- the proof is by case analysis on the possible transition
- there are too many very similar cases
- only one case fails

penance time

interaction between amendment and out-of-order execution went unnoticed for a long time

- one conference publication (proofs in unpublished appendix)
- one journal publication (proof sketch)

. . . not because of sloppiness (we thought we had checked it)

- the proof is by case analysis on the possible transition
- there are too many very similar cases
- only one case fails

coq as a research tool

it took all of 10 minutes to find the first counterexample. . .

finding the right correspondence

the original choreography may execute actions that are blocked by amendment

$\rightsquigarrow$ we need a more relaxed correspondence

finding the right correspondence

the original choreography may execute actions that are blocked by amendment

$\rightsquigarrow$ we need a more relaxed correspondence

an iterative proof process

1. use counterexample to find suitable generalisation

finding the right correspondence

the original choreography may execute actions that are blocked by amendment

$\rightsquigarrow$ we need a more relaxed correspondence

an iterative proof process

1. use counterexample to find suitable generalisation
2. check that the new statement holds using brain

finding the right correspondence

the original choreography may execute actions that are blocked by amendment

$\rightsquigarrow$ we need a more relaxed correspondence

an iterative proof process

1. use counterexample to find suitable generalisation
2. check that the new statement holds using brain
3. formalise the new statement in coq

finding the right correspondence

the original choreography may execute actions that are blocked by amendment

$\rightsquigarrow$ we need a more relaxed correspondence

an iterative proof process

1. use counterexample to find suitable generalisation
2. check that the new statement holds using brain
3. formalise the new statement in coq
4. fail miserably and generate new counterexample

finding the right correspondence

the original choreography may execute actions that are blocked by amendment

$\rightsquigarrow$ we need a more relaxed correspondence

an iterative proof process

1. use counterexample to find suitable generalisation
2. check that the new statement holds using brain
3. formalise the new statement in coq
4. fail miserably and generate new counterexample
5. rinse and repeat

finding the right correspondence

the original choreography may execute actions that are blocked by amendment

$\rightsquigarrow$ we need a more relaxed correspondence

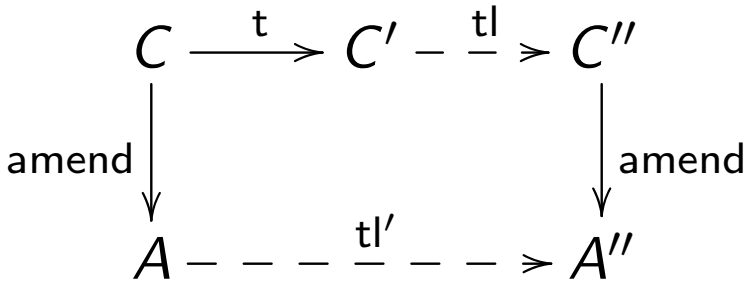
an iterative proof process

1. use counterexample to find suitable generalisation
2. check that the new statement holds using brain
3. formalise the new statement in coq
4. fail miserably and generate new counterexample
5. rinse and repeat

using coq for ego destruction

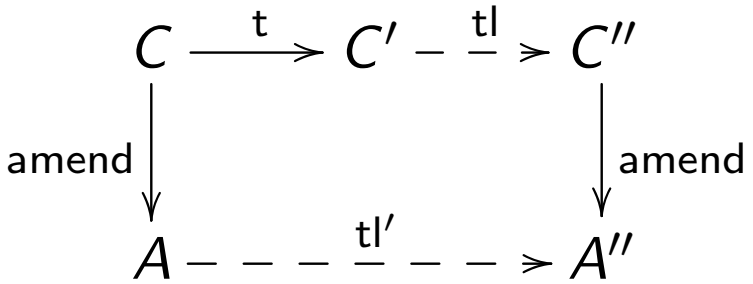
more iterations than we would like to admit

et voilà



where t :: tl can be obtained from tl′ by removing some selections and permuting the result

et voilà



where t :: tl can be obtained from tl′ by removing some selections and permuting the result

$\rightsquigarrow$ *a posteriori*, quite intuitive

closing remarks

conclusions

- a correct characterisation of the behaviour of amendment
- corollaries: turing completeness for interesting calculi (see paper)

closing remarks

conclusions

- a correct characterisation of the behaviour of amendment
- corollaries: turing completeness for interesting calculi (see paper)
- using coq helps finding counterexamples in subcase #39.23.a)

closing remarks

conclusions

- a correct characterisation of the behaviour of amendment
- corollaries: turing completeness for interesting calculi (see paper)
- using coq helps finding counterexamples in subcase #39.23.a)
- . . . and it also saves time in the publishing process

closing remarks

conclusions

- a correct characterisation of the behaviour of amendment
- corollaries: turing completeness for interesting calculi (see paper)
- using coq helps finding counterexamples in subcase #39.23.a)
- . . . and it also saves time in the publishing process

future work

- more flexible projection $\rightsquigarrow$ allows for simpler amendment
- more expressive language
  more automation $\rightsquigarrow$ more interesting in practice
- applications to cybersecurity, smart contracts, . . .

thank you!