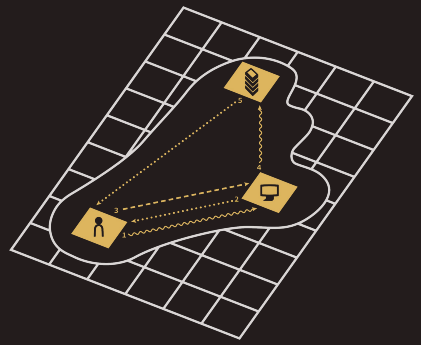


Choreographic Programming

its essence, beauty, and necessity

Fabrizio Montesi
University of Southern Denmark

www.fabriziomontesi.com
linkedin.com/in/fmontesi
𝕏.com/famontesi

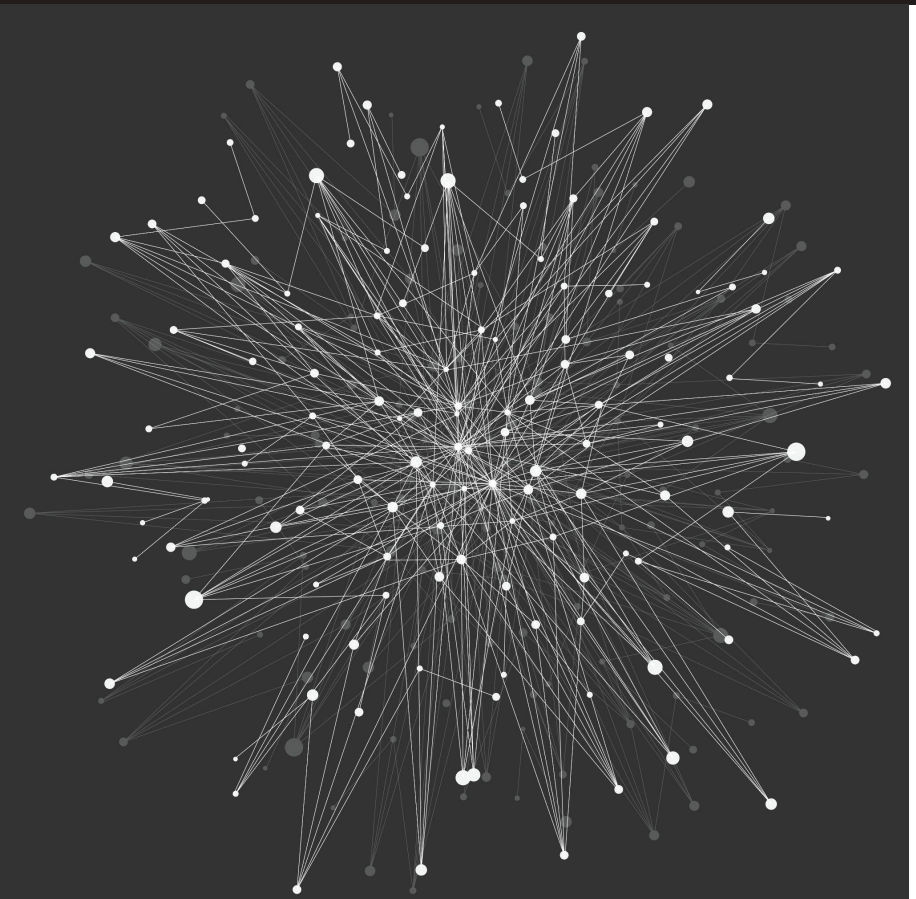
PROJECT CHORDS

erc

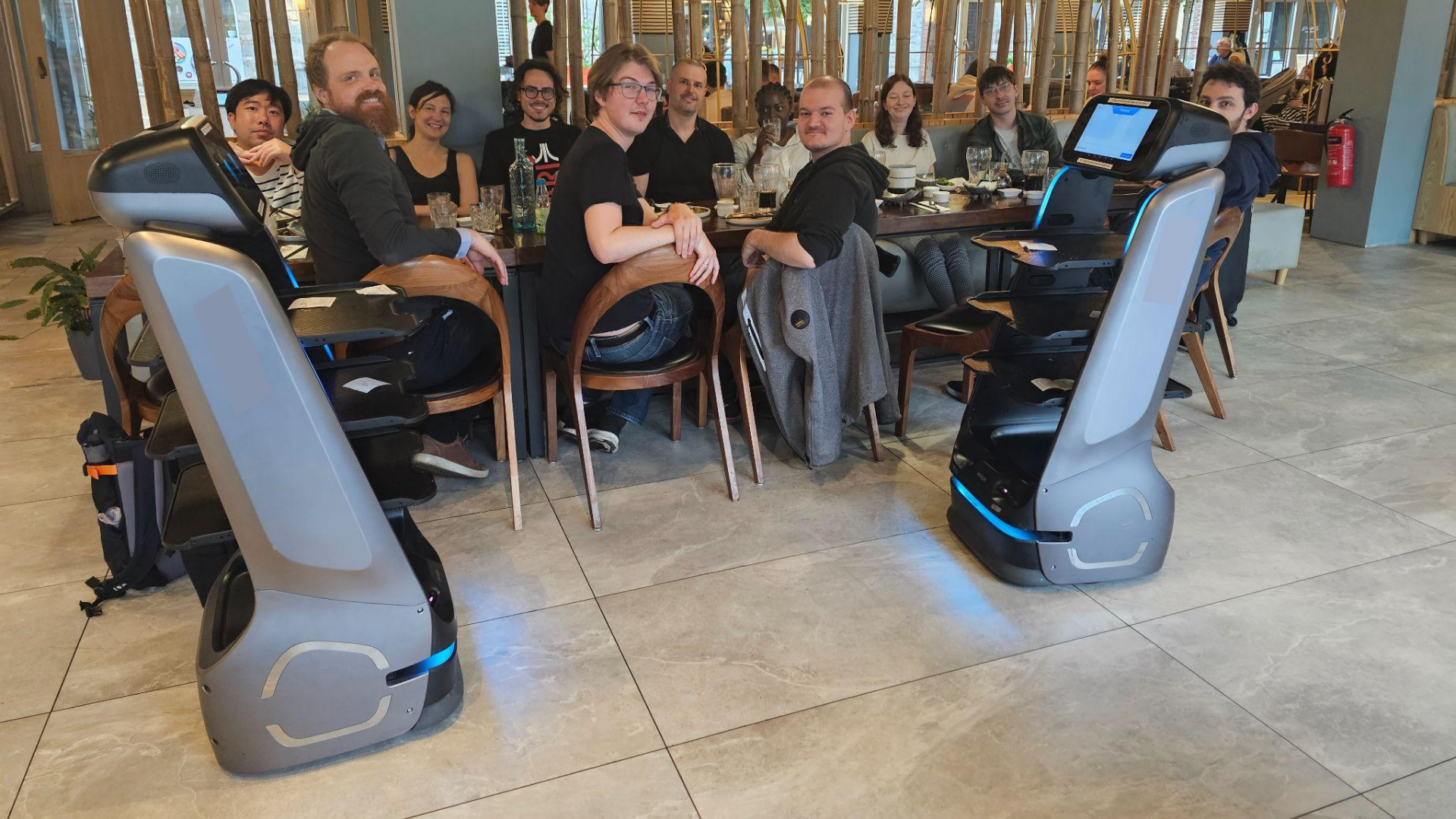
SDU
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

VILLUM FONDEN

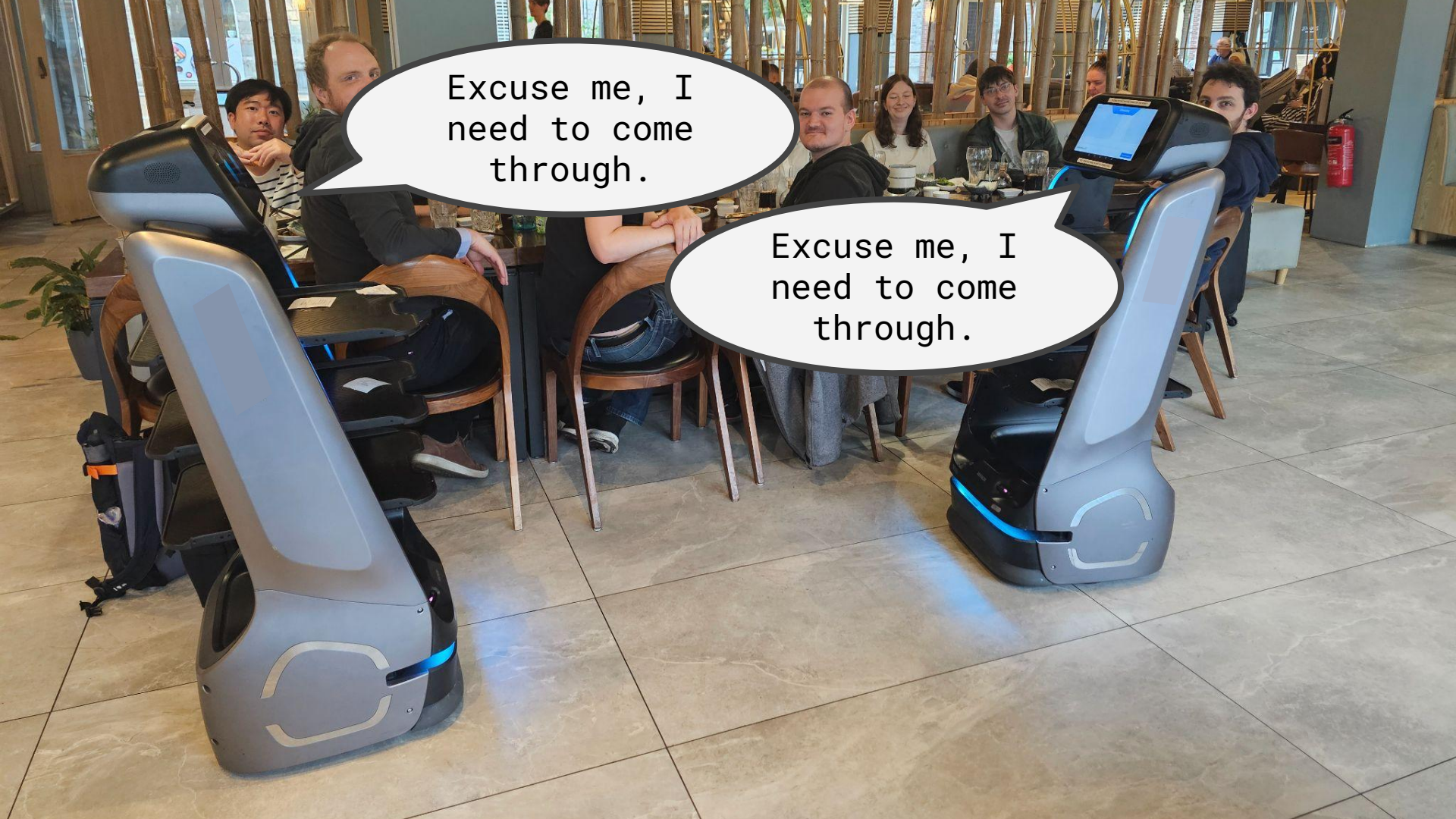
The setting: concurrent and distributed systems



- ➔ Multiple things happening at the same time.
- ➔ Processes and resources at (possibly) different locations.
- ➔ Emergent behaviour.




Excuse me, I need to come through.


Excuse me, I need to come through.
Excuse me, I need to come through.

Excuse me, I need to come through.

I e

A distributed program

Code for a

```
recv x from c;
if valid(x) {
    send OK to ws;
} else {
    send KO to ws;
}
```

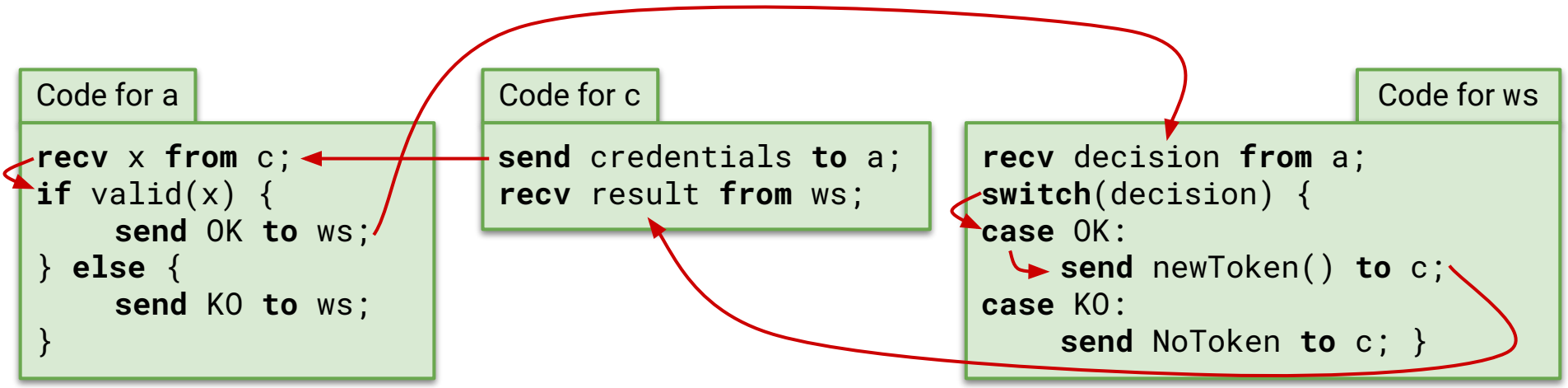
Code for c

```
send credentials to a;
recv result from ws;
```

Code for ws

```
recv decision from a;
switch(decision) {
case OK:
    send newToken() to c;
case KO:
    send NoToken to c; }
```

A distributed program



Aim

What is simple should look simple.

Aim

What is simple should look simple.

KISS

Aim

What is simple should look simple.

Chef’s KISS

Aim

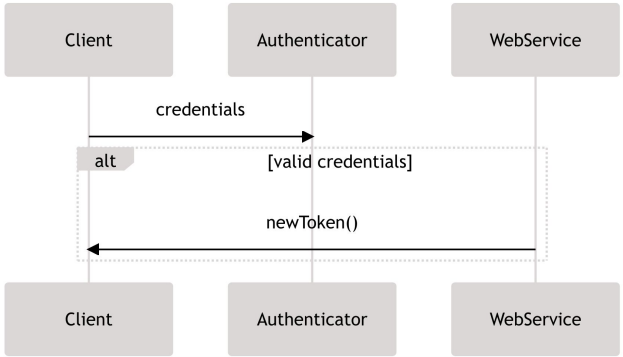
What is simple should look simple.

Choreographies

Coordination plans for concurrent and distributed systems.

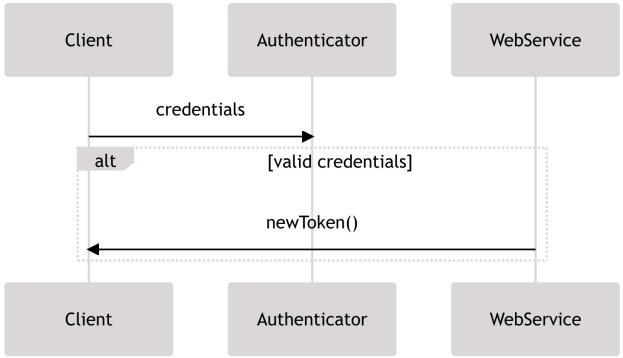
Choreographies

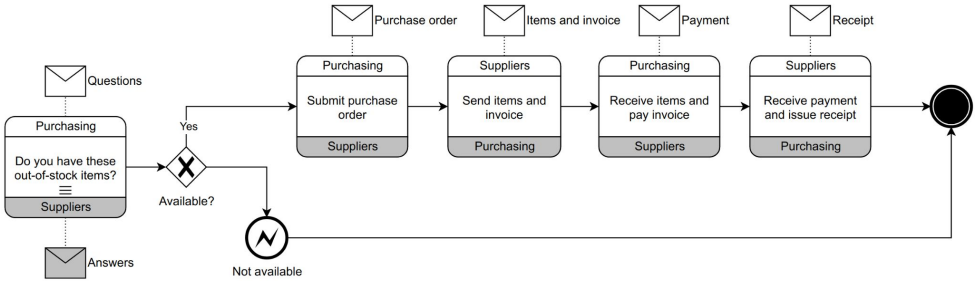
Coordination plans for concurrent and distributed systems.



Choreographies

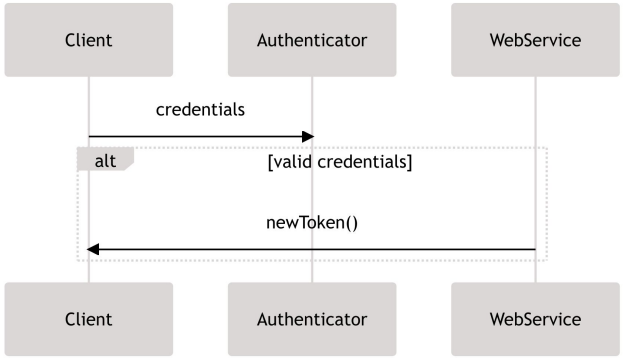
Coordination plans for concurrent and distributed systems.



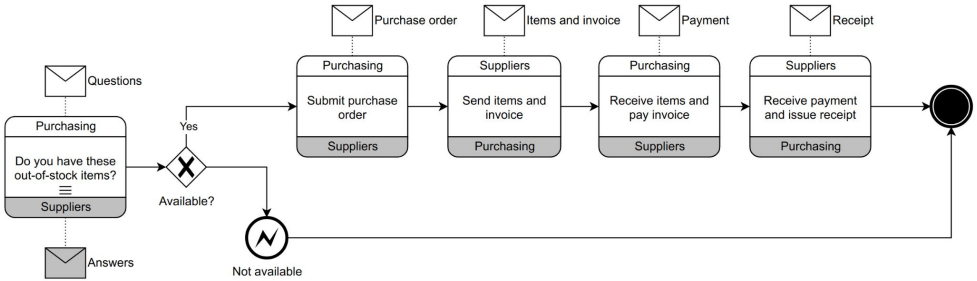


Choreographies

Coordination plans for concurrent and distributed systems.

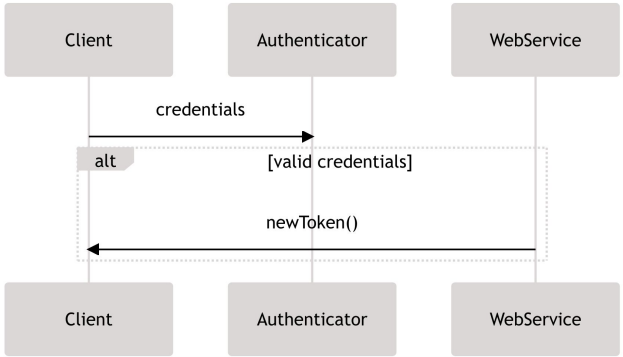



```
[...]
The sensor communicates its data to the
edge server, which then pseudonymises
the data before uploading it to the
cloud server. The data is then analysed
and displayed on the user desktop, which
can authorise whether the data should be
permanently stored or not.
Then... [...]
```

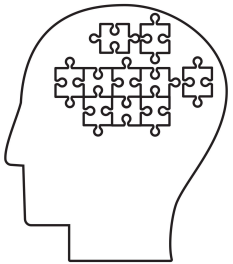


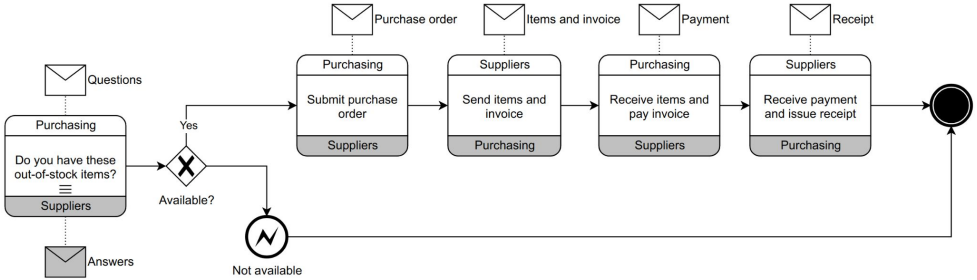
Choreographies

Coordination plans for concurrent and distributed systems.




```
[...]
The sensor communicates its data to the
edge server, which then pseudonymises
the data before uploading it to the
cloud server. The data is then analysed
and displayed on the user desktop, which
can authorise whether the data should be
permanently stored or not.
Then... [...]
```





Choreographies

State explosion problem

2008

Learning from Mistakes — A Comprehensive Study on Real World Concurrency Bug Characteristics

Shan Lu, Soyeon Park, Eunsoo Seo and Yuanyuan Zhou
Department of Computer Science,
University of Illinois at Urbana Champaign, Urbana, IL 61801
{shanlu,soyeon,eseo2,yyzhou}@uiuc.edu

2016

TaxDC: A Taxonomy of Non-Deterministic Concurrency Bugs in Datacenter Distributed Systems

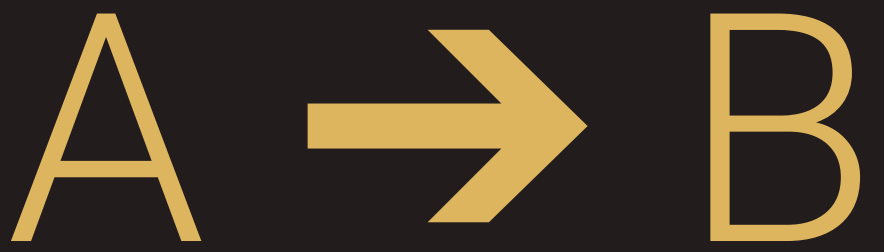
Tanakorn Leesatapornwongsa
University of Chicago
tanakorn@cs.uchicago.edu

Jeffrey F. Lukman
Surya University
lukman@cs.uchicago.edu

Shan Lu
University of Chicago
shanlu@uchicago.edu

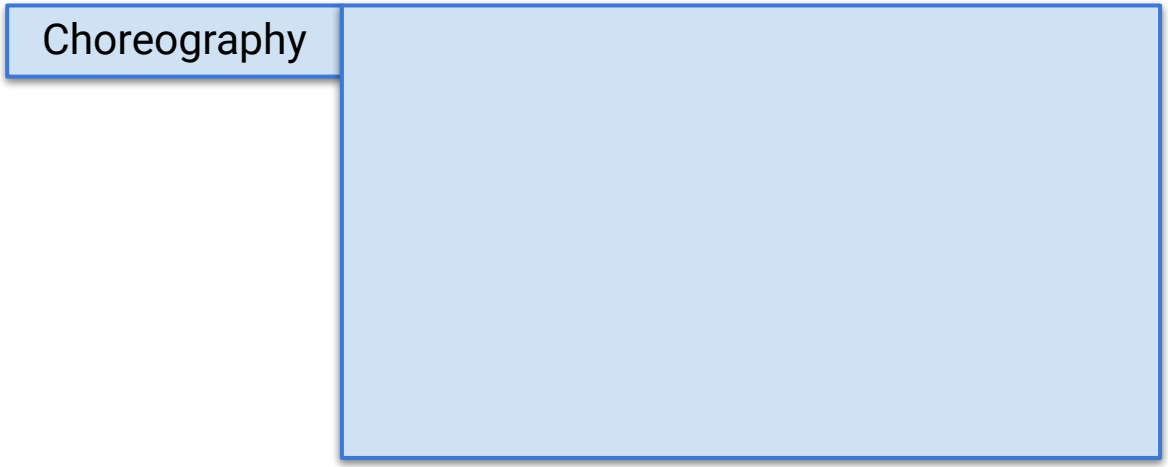
Haryadi S. Gunawi
University of Chicago
haryadi@cs.uchicago.edu

Even expert programmers **easily** make mistakes!

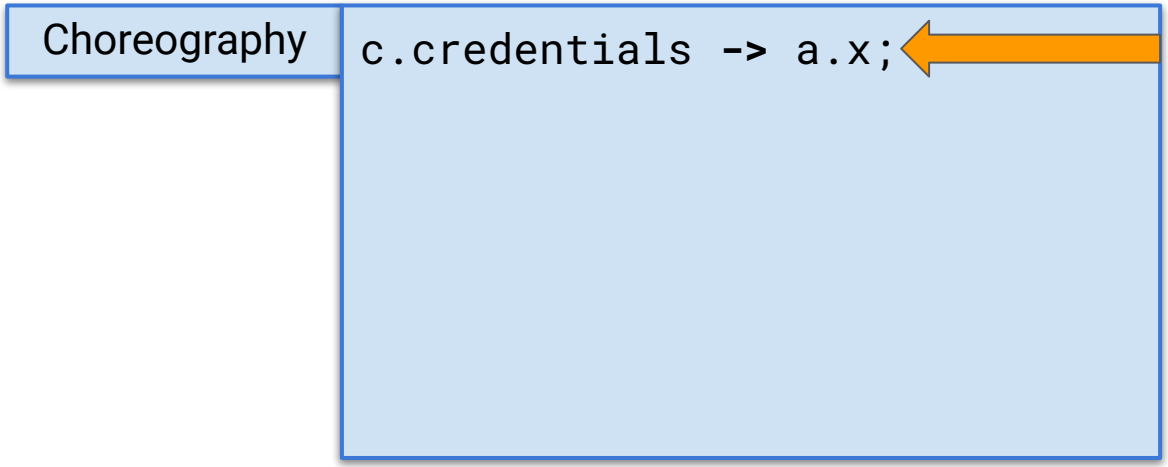

A
B


p
q

Choreographic Programming



Choreographic Programming



Choreographic Programming

Choreography

```
c.credentials -> a.x;
if a.valid(x) {


} else {


}
```

Choreographic Programming

Choreography

```
c.credentials -> a.x;
if a.valid(x) {
    a.OK -> ws.decision;

} else {


}
```

Choreographic Programming

Choreography

```
c.credentials -> a.x;
if a.valid(x) {
    a.OK -> ws.decision;
    ws.newToken() -> c.result;
} else {


}
```

Choreographic Programming

Choreography

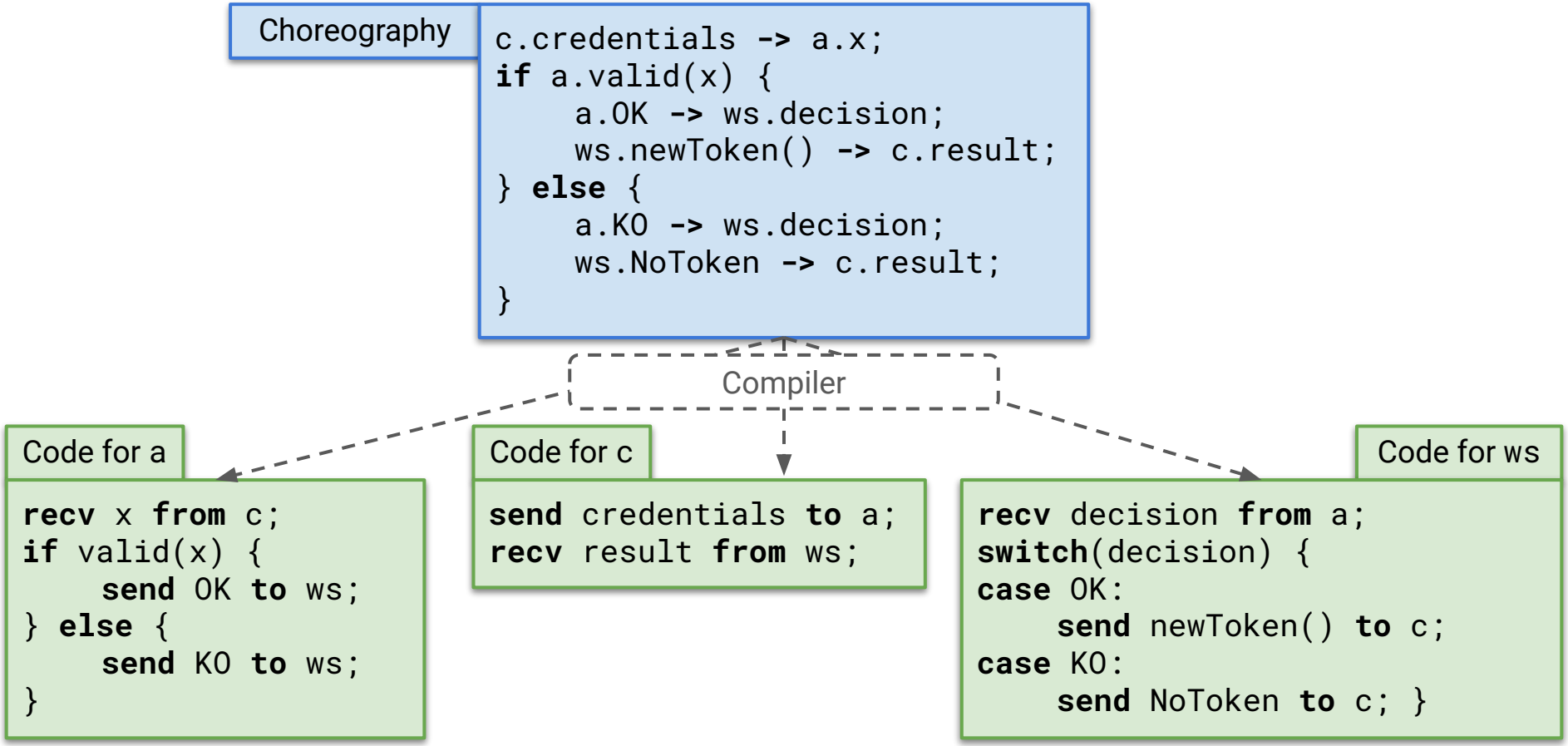
```
c.credentials -> a.x;
if a.valid(x) {
    a.OK -> ws.decision;
    ws.newToken() -> c.result;
} else {
    a.KO -> ws.decision;

}
```

Choreographic Programming

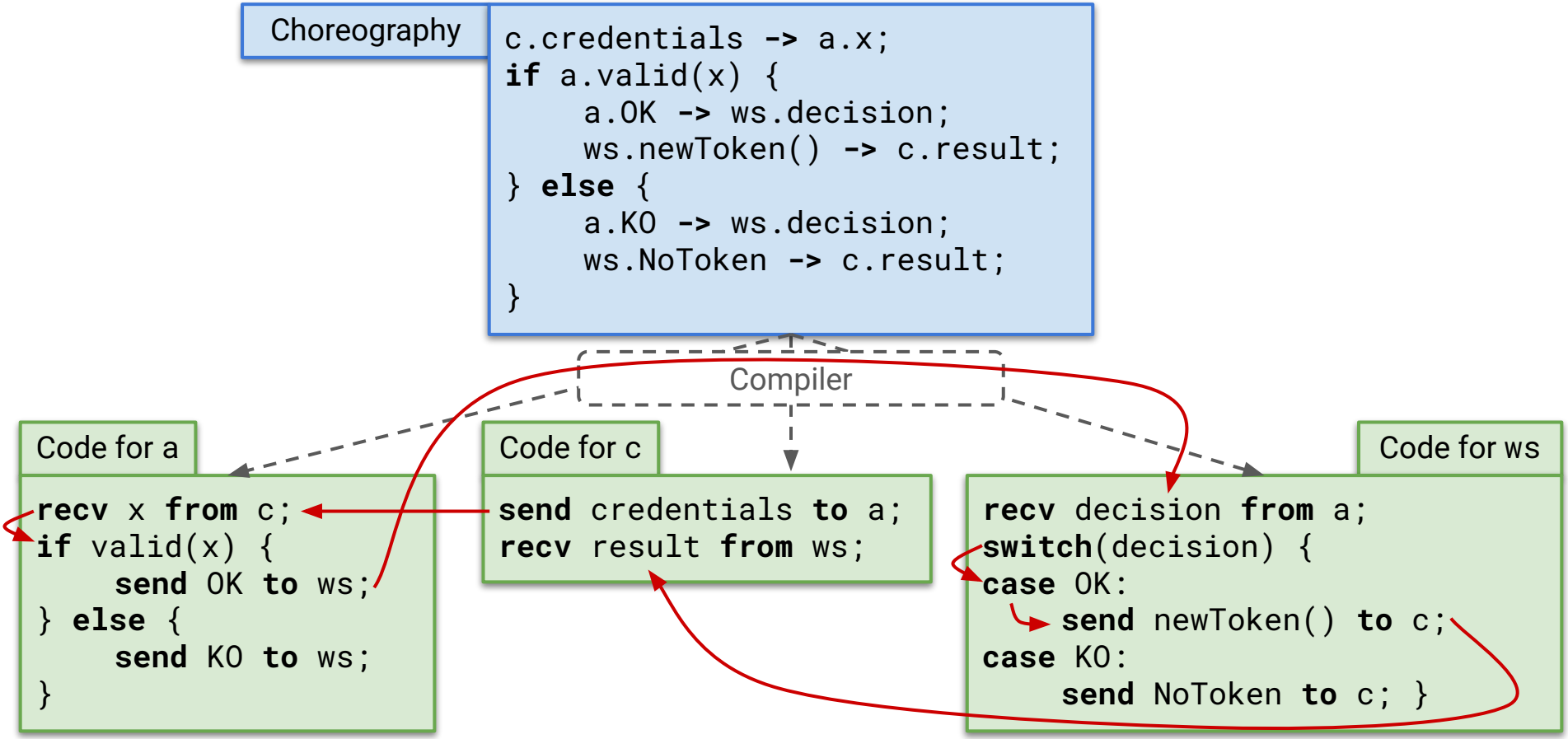
Choreography

```
c.credentials -> a.x;
if a.valid(x) {
    a.OK -> ws.decision;
    ws.newToken() -> c.result;
} else {
    a.KO -> ws.decision;
    ws.NoToken -> c.result;
}
```

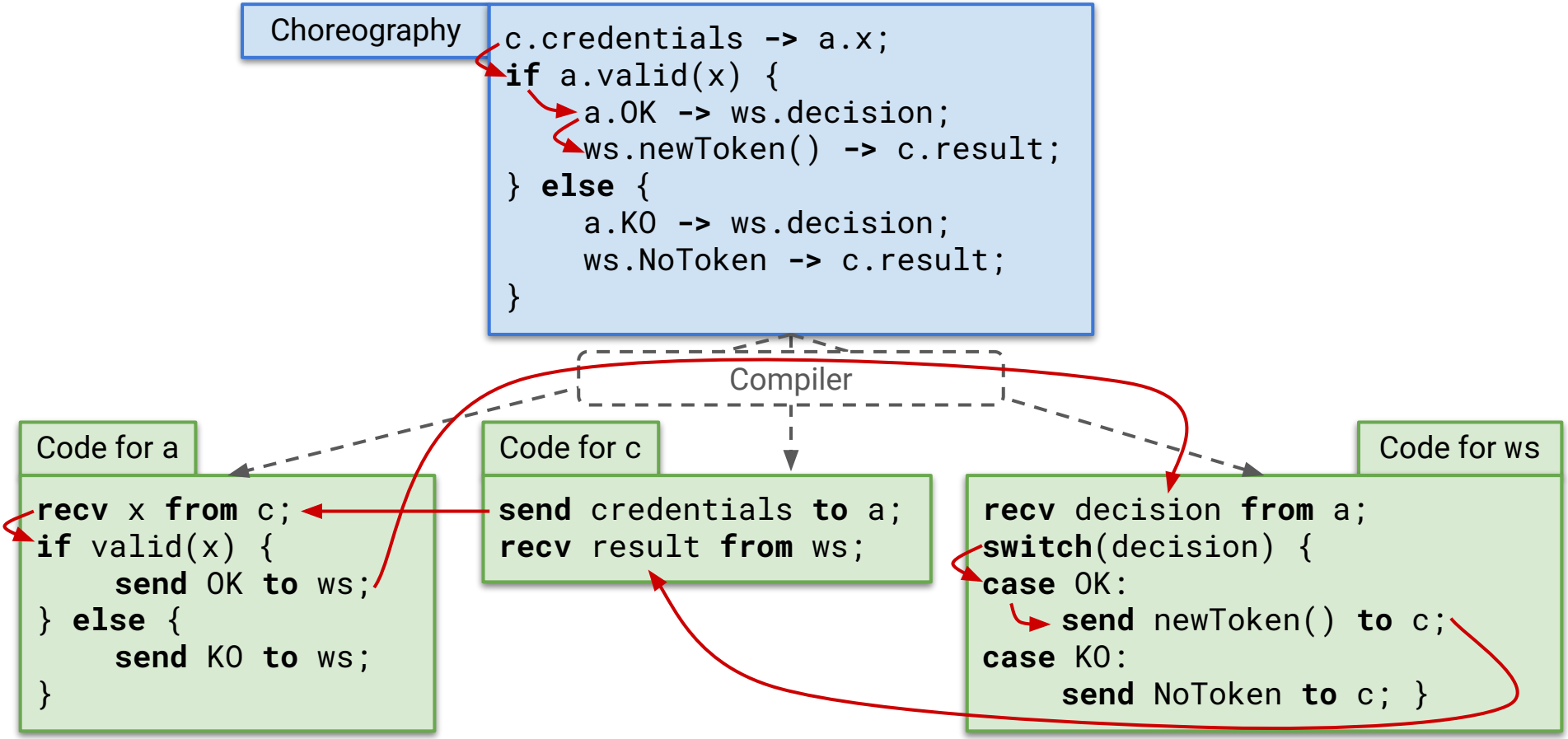
Choreographic Programming



Choreographic Programming



Choreographic Programming



Choreographic Programming

Choreography

```
c.credentials -> a.x;
if a.valid(x) {
    a.OK -> ws.decision;
    ws.newToken() -> c.result;
} else {
    a.KO -> ws.decision;
    ws.NoToken -> c.result;
}
```

Compiler

Code for a

```
recv x from c;
if valid(x) {
    send OK to ws;
} else {
    send KO to ws;
}
```

Code for c

```
send credentials to a;
recv result from ws;
```

Code for ws

```
recv decision from a;
switch(decision) {
case OK:
    send newToken() to c;
case KO:
    send NoToken to c; }
```

Properties

Compliance

The system behaves as prescribed by the choreography.

Properties

Compliance

The system behaves as prescribed by the choreography.

Deadlock-Freedom

The system can always progress as a whole.

Deadlock-freedom-by-design: Multiparty Asynchronous Global Programming

Marco Carbone Fabrizio Montesi

```
c.credentials -> a.x;
if a.valid(x) {
    ...
}
```

How to use: some examples

How to use: some examples

- ★ Choreographies as applications.

How to use: some examples

- ★ Choreographies as applications.
- ★ Choreographies as skeletons.

Code for c

```
send credentials to a;
recv result from ws;
// INSERT YOUR CODE HERE
```

How to use: some examples

- ★ Choreographies as applications.
- ★ Choreographies as skeletons.
- ★ Choreographies as libraries.

Code for c

```
AuthResult auth(Credentials credentials) {
    send credentials to a;
    recv result from ws;
    return result;
}
```

1978

1978

1978

Using Encryption for Authentication in Large Networks of Computers

Roger M. Needham and
Michael D. Schroeder
Xerox Palo Alto Research Center

1978

```
A -> B : M
```

‘The protocol opens with A communicating in clear to AS [...]’

$$A \rightarrow AS: \qquad A, B, I_{A1}$$

‘The next transaction [...]’

$$AS \rightarrow A: \qquad \{I_{A1}, B, CK, \{CK, A\}^{KB}\}^{KA}$$

1978

1978

‘[...] it is an **obvious** way to present such protocols.’

1978

‘[...] it is an **obvious** way to present such protocols.’

1978

‘[...] it is an **obvious** way to present such protocols.’

‘[...] an **obvious** way of expressing what is going on.’

1978

‘[...] it is an **obvious** way to present such protocols.’

‘[...] an **obvious** way of expressing what is going on.’

2005

W3C

Web Services Choreography Description Language Version 1.0

Editors:

Nickolas Kavantzas, Oracle
David Burdett, Commerce One
Gregory Ritzinger, Novell
Tony Fletcher, Choreology
Yves Lafon, W3C
Charlton Barreto, Adobe Systems Incorporated

2005

```
<interaction name="myInteraction" channelVariable="k" operation="buy">
  <participate relationshipType="pandq" fromRoleTypeRef="tns:p"
    toRoleTypeRef="tns:q" />
  <exchange name="myExchange" action="request">
    <send variable="cdl:getVariable('tns:x','')" />
    <receive variable="cdl:getVariable('tns:y','','')" />
  </exchange>
</interaction>
```

2005

```
<interaction name="myInteraction" channelVariable="k" operation="buy">
  <participate relationshipType="pandq" fromRoleTypeRef="tns:p"
    toRoleTypeRef="tns:q" />
  <exchange name="myExchange" action="request">
    <send variable="cdl:getVariable('tns:x','')" />
    <receive variable="cdl:getVariable('tns:y','','')" />
  </exchange>
</interaction>
```

```
p.x -> q.y : buy(k)
```

2005

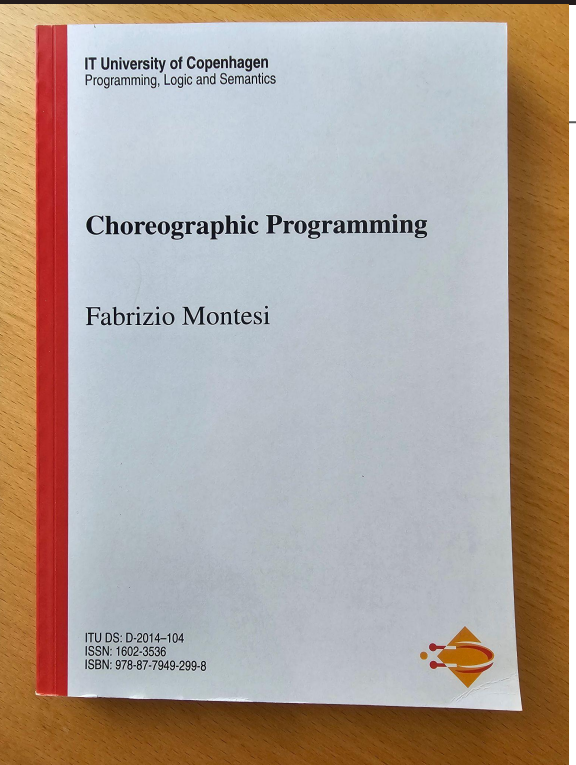
W3C

Web Services Choreography Description Language Version 1.0

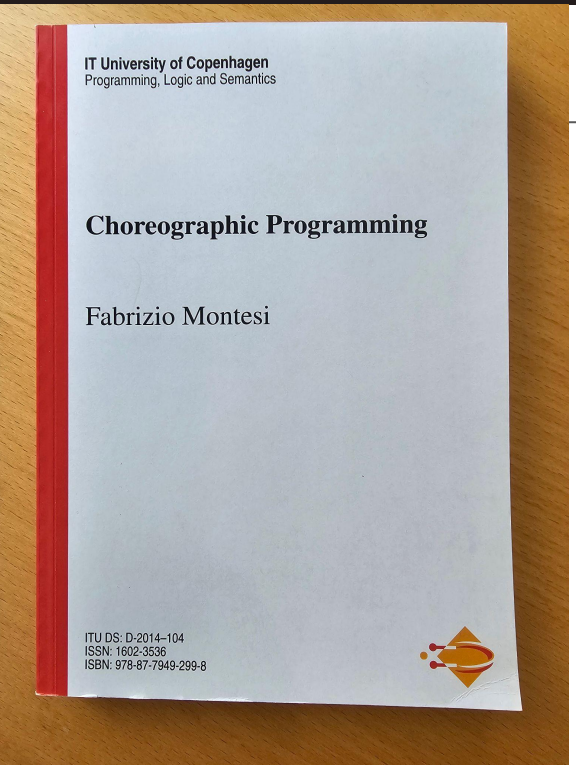
Editors:

Nickolas Kavantzas, Oracle
David Burdett, Commerce One
Gregory Ritzinger, Novell
Tony Fletcher, Choreology
Yves Lafon, W3C
Charlton Barreto, Adobe Systems Incorporated


1978
2005
2013
A -> B : M
</>
p.e -> q.x


IT University of Copenhagen
Programming, Logic and Semantics
Choreographic Programming
Fabrizio Montesi
ITU DS: D-2014–104
ISSN: 1602-3536
ISBN: 978-87-7949-299-8

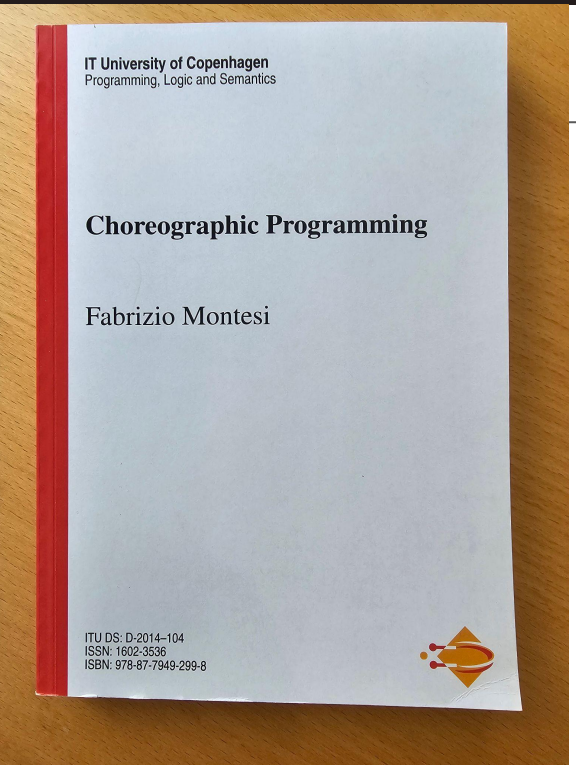
2013



2013

Deadlock-freedom-by-design: Multiparty Asynchronous Global Programming

Marco Carbone Fabrizio Montesi

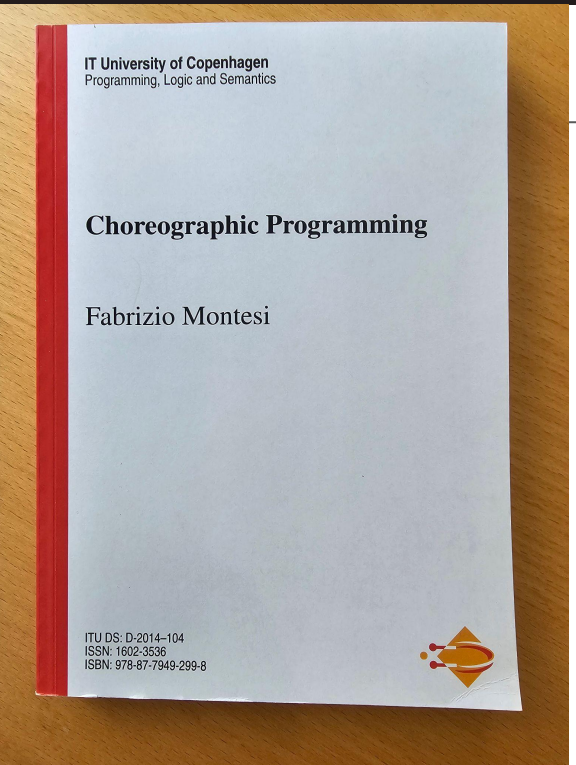


2013

Deadlock-freedom-by-design: Multiparty Asynchronous Global Programming

Marco Carbone Fabrizio Montesi

```
p1.e1 -> q1.x1;          ≡    p2.e2 -> q2.x2;
p2.e2 -> q2.x2;               p1.e1 -> q1.x1;
```



2013

Chor

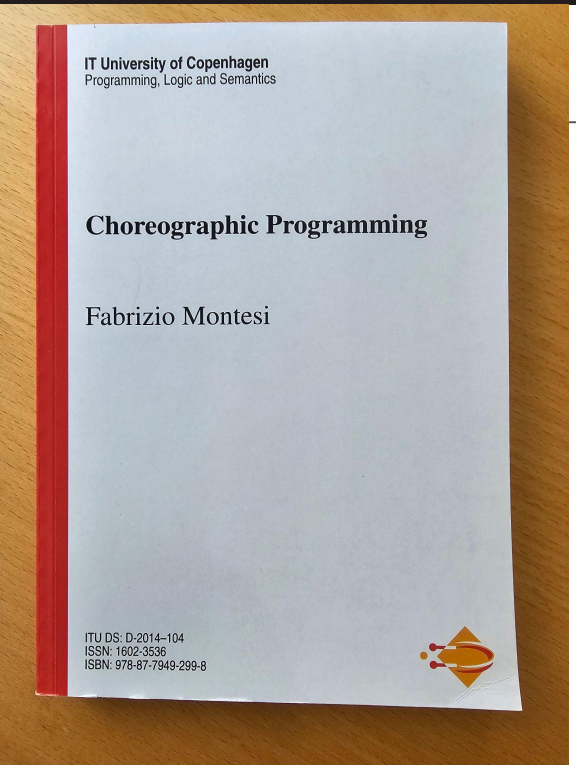
```
client.msg -> server.x : hi(k);
show@server(msg);
```

1978 2005 2013

A -> B : M

</>

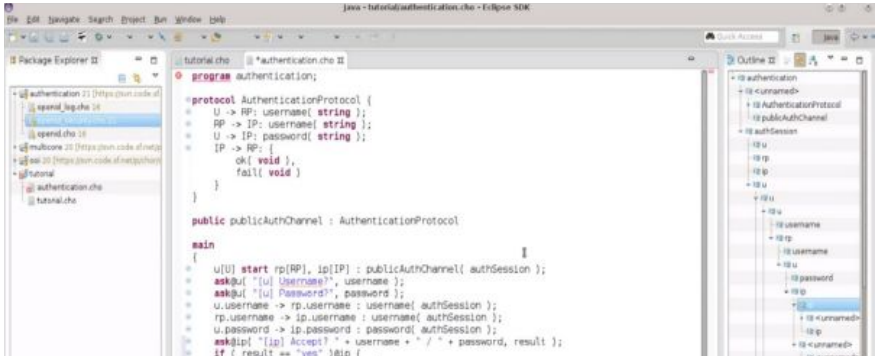
p.e -> q.x



2013

Chor

```
client.msg -> server.x : hi(k);
show@server(msg);
```

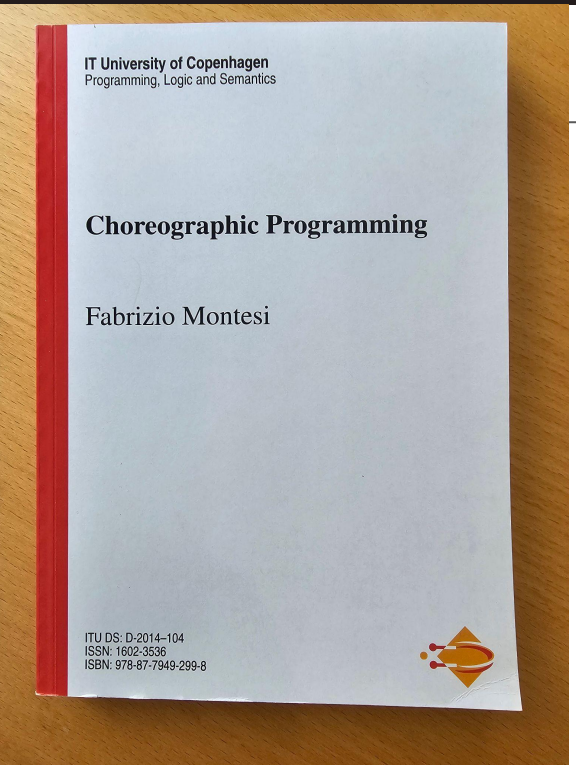


1978 2005 2013

A -> B : M

</>

p.e -> q.x

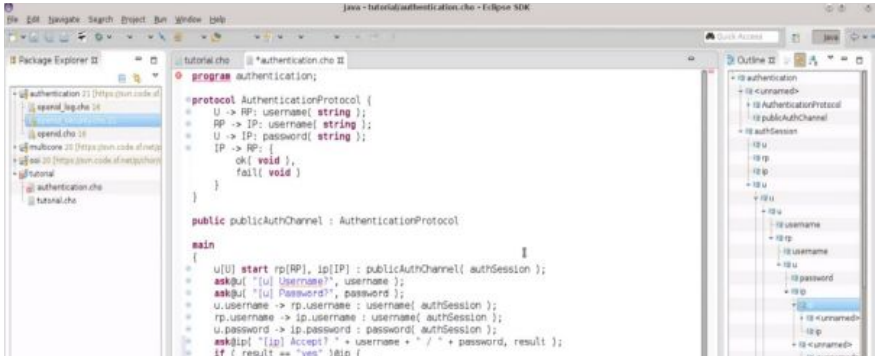


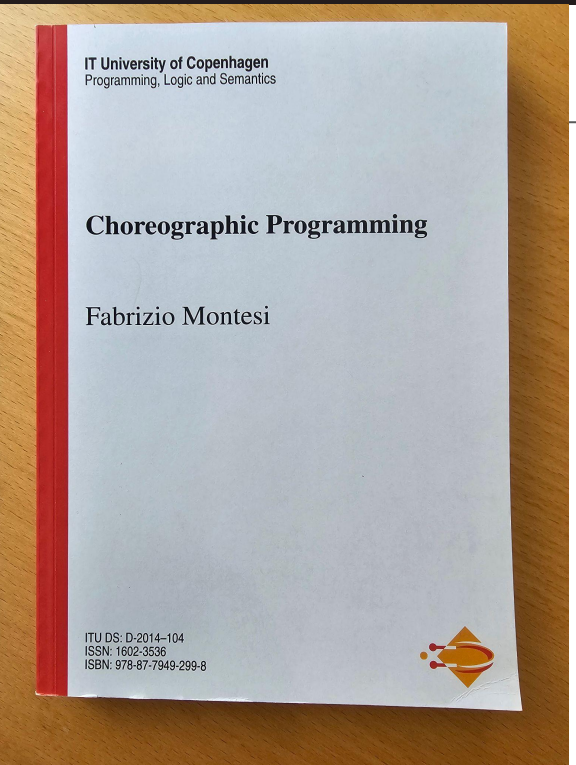
2013

Chor

```
client.msg -> server.x : hi(k);
show@server(msg);
```

Jolie





2013

Choreographies, Logically

Marco Carbone[1], Fabrizio Montesi[2⋆], and Carsten Schürmann[1]

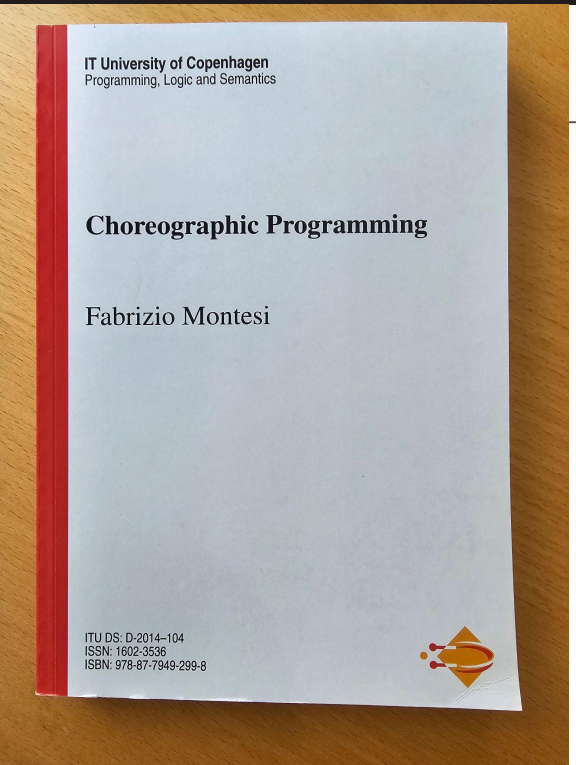
1978 2005 2013

A -> B : M

</>

p.e -> q.x

p -> q : x(y)



2013

Choreographies, Logically

Marco Carbone[1], Fabrizio Montesi[2⋆], and Carsten Schürmann[1]

- Proofs as Processes
- Propositions as Session Types
- Cut Elimination as Communication

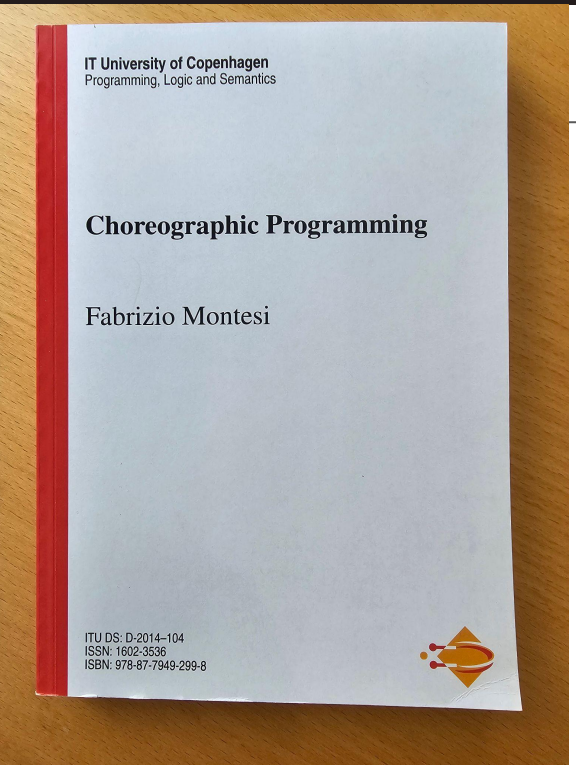
1978 2005 2013

A -> B : M

</>

p.e -> q.x

p -> q : x(y)



2013

Choreographies, Logically

Marco Carbone[1], Fabrizio Montesi[2⋆], and Carsten Schürmann[1]

- Proofs as Processes
- Propositions as Session Types
- Cut Elimination as **Choreographies**

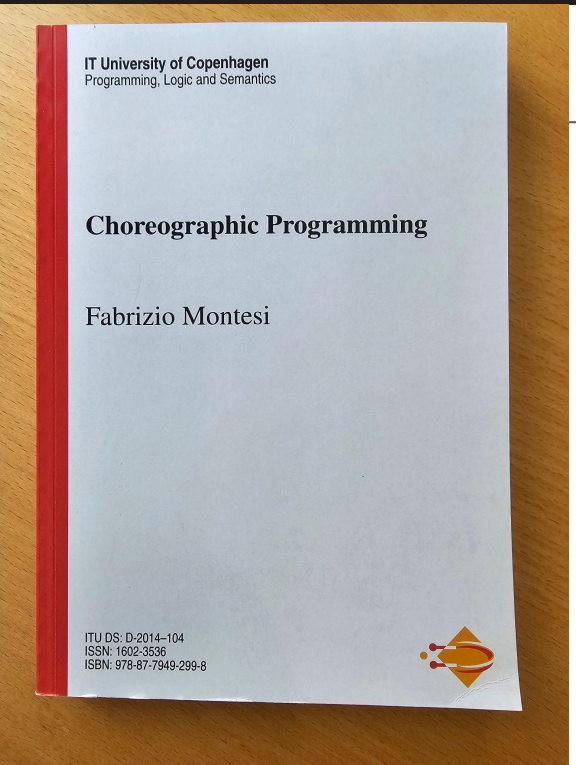
1978 2005 2013

A -> B : M

</>

p.e -> q.x

p -> q : x(y)



2013

Choreographies, Logically

Marco Carbone[1], Fabrizio Montesi[2⋆], and Carsten Schürmann[1]

- Proofs as Processes
- Propositions as Session Types
- Cut Elimination as **Choreographies**

```
p -> q : x(y)
```

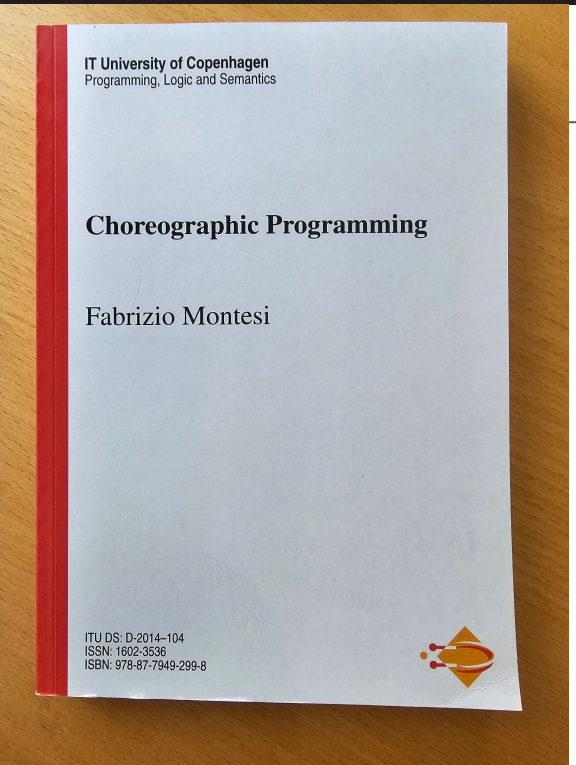
1978 2005 2013

A -> B : M

</>

p.e -> q.x

p -> q : x(y)



2013

Choreographies, Logically

Marco Carbone[1], Fabrizio Montesi[2⋆], and Carsten Schürmann[1]

- Proofs as Processes
- Propositions as Session Types
- Cut Elimination as **Choreographies**

```
p -> q : x(y)
```

```
p1 -> q1: x1(y1);
p2 -> q2: x2(y2);
```

≡

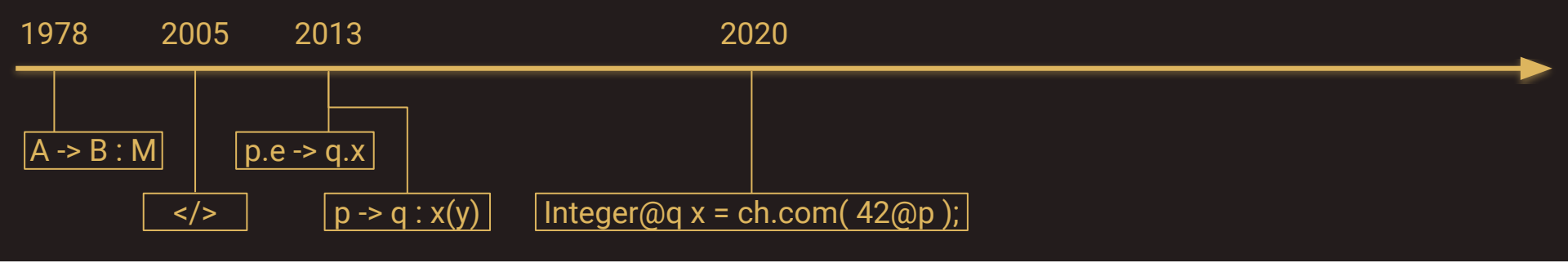
```
p2 -> q2: x2(y2);
p1 -> q1: x1(y1);
```

2020

CHORAL

Choral: Object-oriented Choreographic Programming

SAVERIO GIALLORENZO, Università di Bologna, Italy and INRIA, France
FABRIZIO MONTESI and MARCO PERESSOTTI, University of Southern Denmark, Denmark

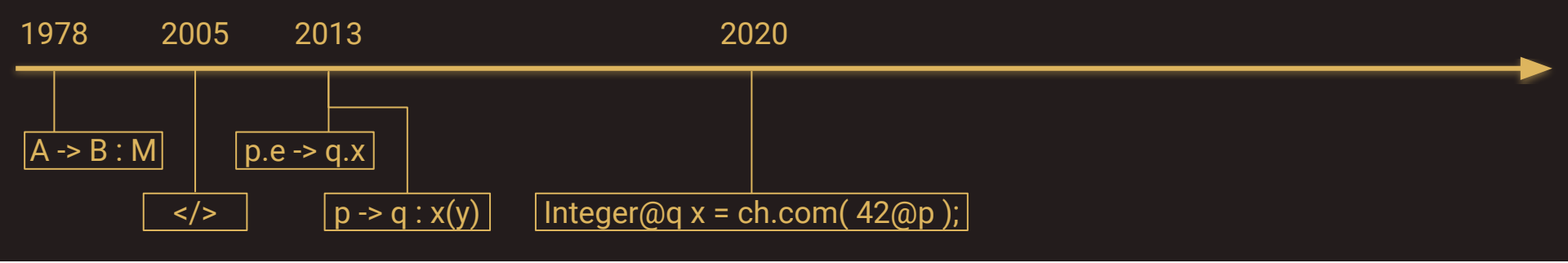


2020

```
AuthResult@ws auth(Credentials@c creds) { ... }
```

CHORAL

Java

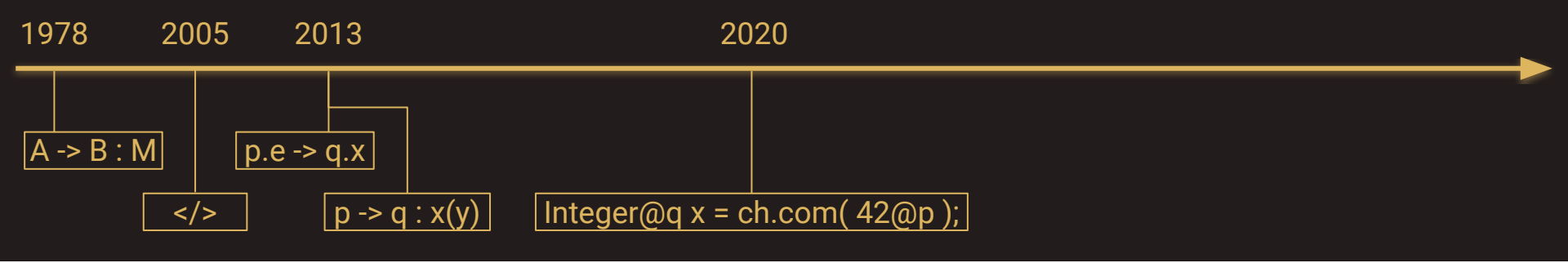


2020

```
AuthResult@ws auth(Credentials@c creds) { ... }
```

CHORAL

```
T@q com(T@p mesg) { ... }
```



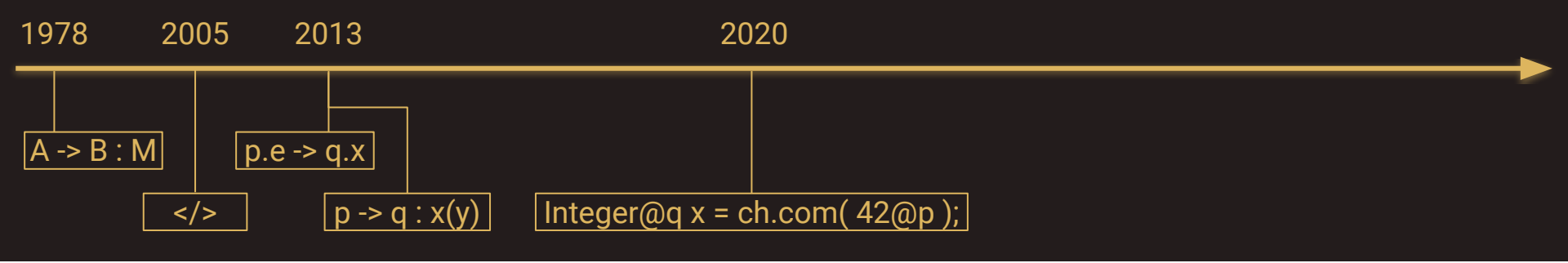
2020

CHORAL

```
AuthResult@ws auth(Credentials@c creds) { ... }
```

```
T@q com(T@p mesg) { ... }
```

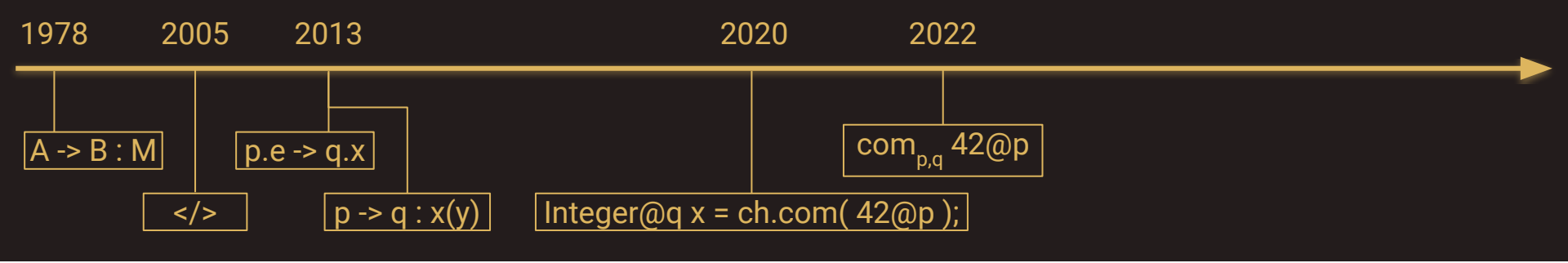
```
Integer@q x = com(42@p);
```



2020

CHORAL

- ★ Choreographies as applications.
- ★ Choreographies as skeletons.
- ★ Choreographies as libraries.



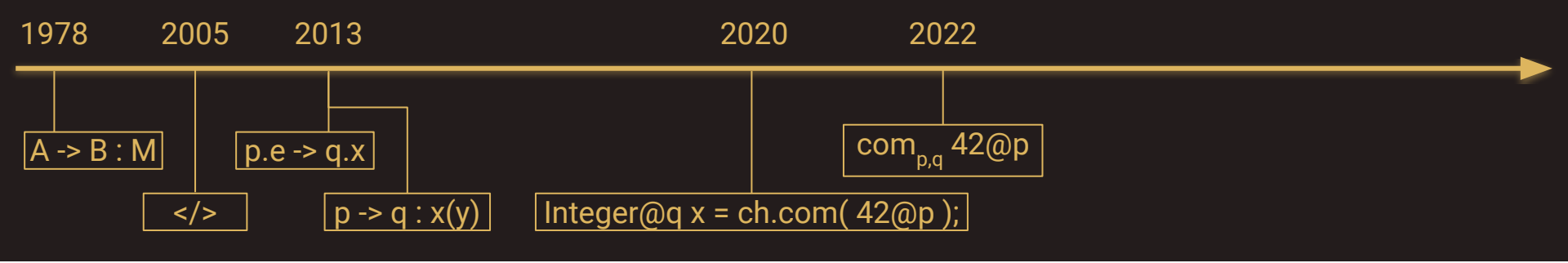
2022

Functional Choreographic Programming*

Luís Cruz-Filipe, Eva Graversen, Lovro Lugović, Fabrizio Montesi, and Marco Peressotti

$com_{p,q}$: T@p -> T@q

$com_{q,r}$ (f ($com_{p,q}$ 42@p))



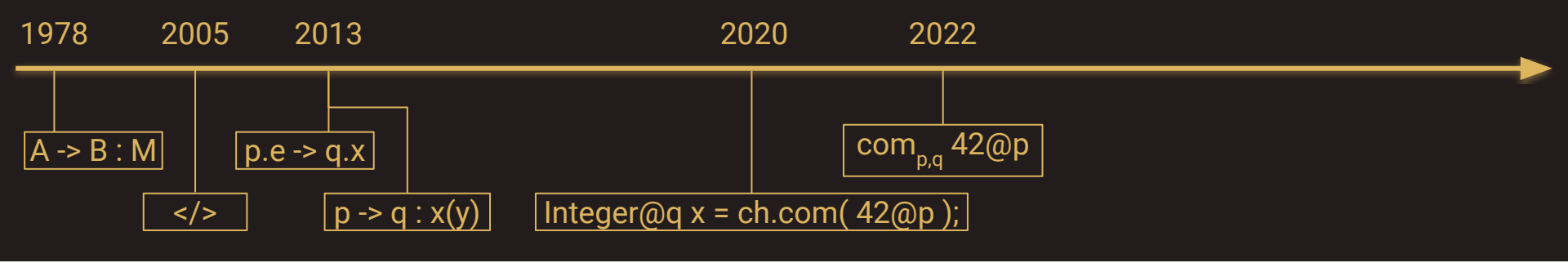
2022

Functional Choreographic Programming*

Luís Cruz-Filipe, Eva Graversen, Lovro Lugović, Fabrizio Montesi, and Marco Peressotti

$com_{p,q}$: T@p -> T@q

$com_{q,r}$ (f ($com_{p,q}$ 42@p))



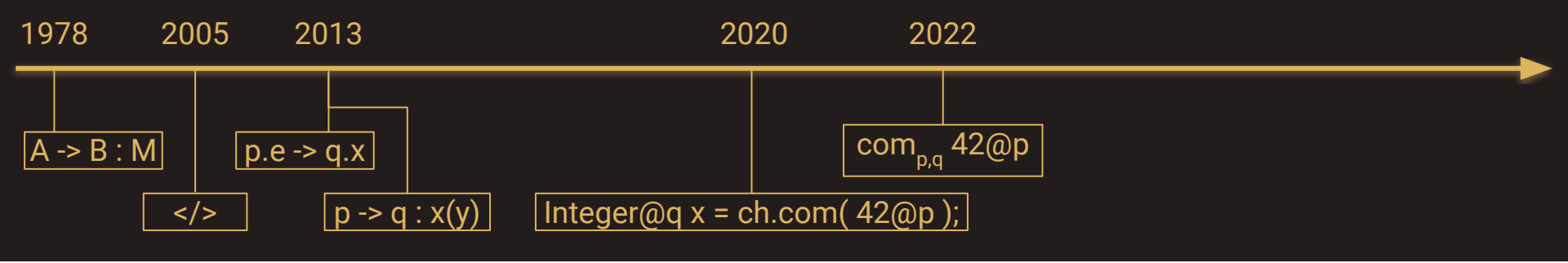
2022

Functional Choreographic Programming*

Luís Cruz-Filipe, Eva Graversen, Lovro Lugović, Fabrizio Montesi, and Marco Peressotti

$com_{p,q}$: T@p -> T@q

$com_{q,r}$ (f ($com_{p,q}$ 42@p))



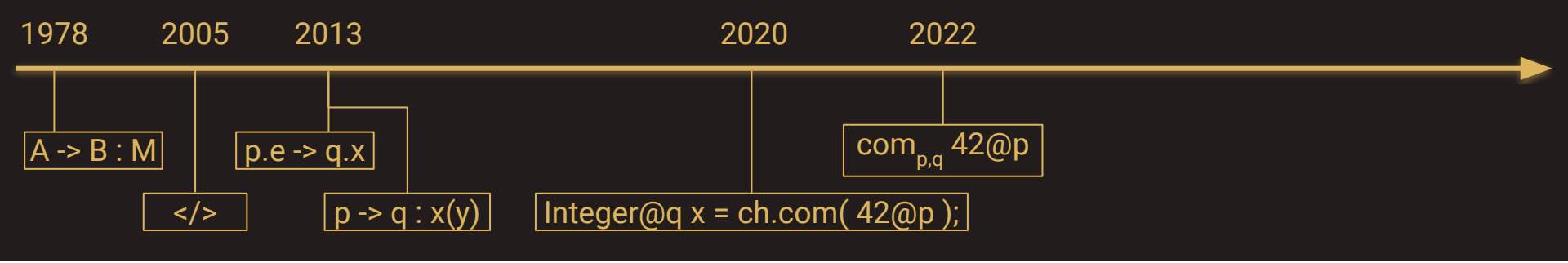
2022

Functional Choreographic Programming*

Luís Cruz-Filipe, Eva Graversen, Lovro Lugović, Fabrizio Montesi, and Marco Peressotti

$com_{p,q}$: T@p -> T@q

$com_{q,r}$ (f ($com_{p,q}$ 42@p))



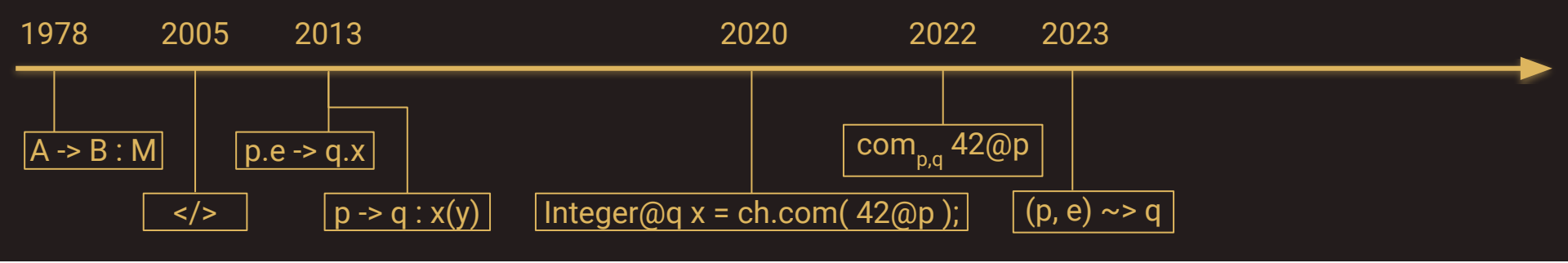
2022

Functional Choreographic Programming*

Luís Cruz-Filipe, Eva Graversen, Lovro Lugović, Fabrizio Montesi, and Marco Peressotti

$com_{p,q}$: T@p -> T@q

$com_{q,r}$ (f ($com_{p,q}$ 42@p))

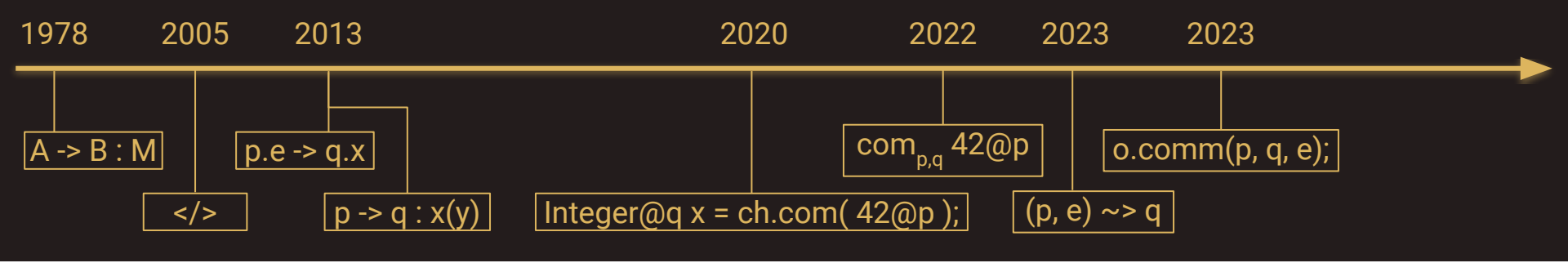


2023

HasChor: Functional Choreographic Programming for All (Functional Pearl)

GAN SHEN, University of California, Santa Cruz, USA
SHUN KASHIWA, University of California, Santa Cruz, USA
LINDSEY KUPER, University of California, Santa Cruz, USA

```
(p, e) ~> q
```

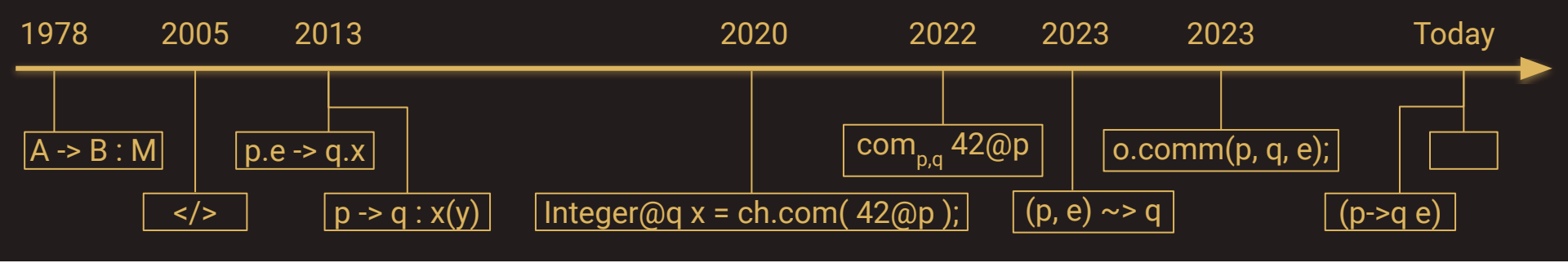


2023

Portable, Efficient, and Practical Library-Level Choreographic Programming

SHUN KASHIWA, University of California, Santa Cruz, USA
GAN SHEN, University of California, Santa Cruz, USA
SOROUSH ZARE, University of California, Santa Cruz, USA
LINDSEY KUPER, University of California, Santa Cruz, USA

```
o.comm(p, q, e);
```

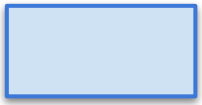


2024

CHOREOGRAPHIC PROGRAMMING

PLDI COPENHAGEN 2024

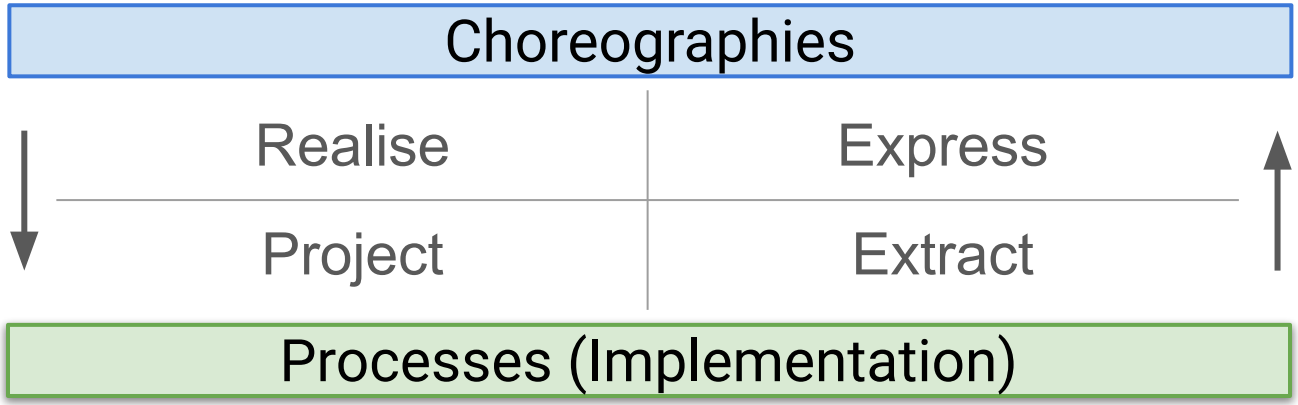
(p->q e)



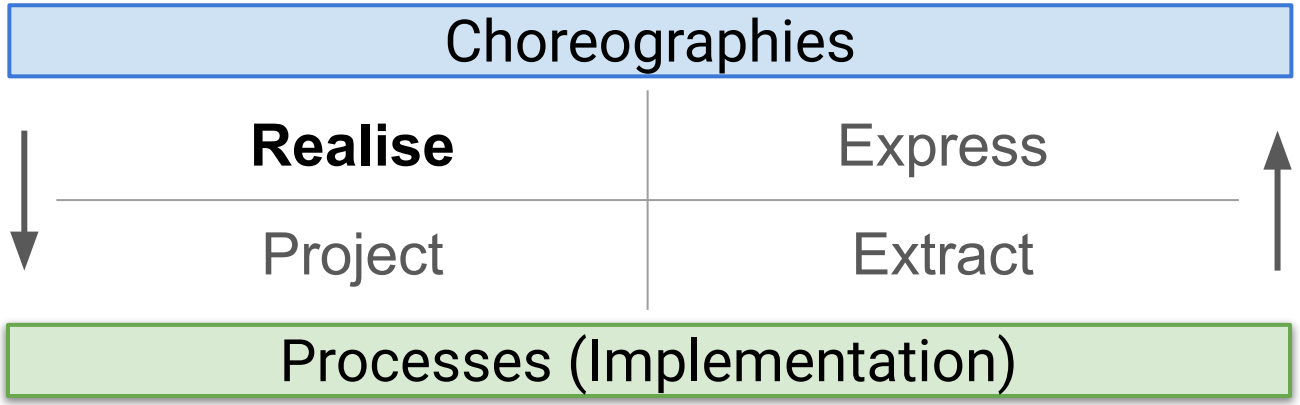
Choreographic Programming Languages

- ★ Locality.
- ★ Objective viewpoint.
- ★ Interaction abstraction.

Translational Questions

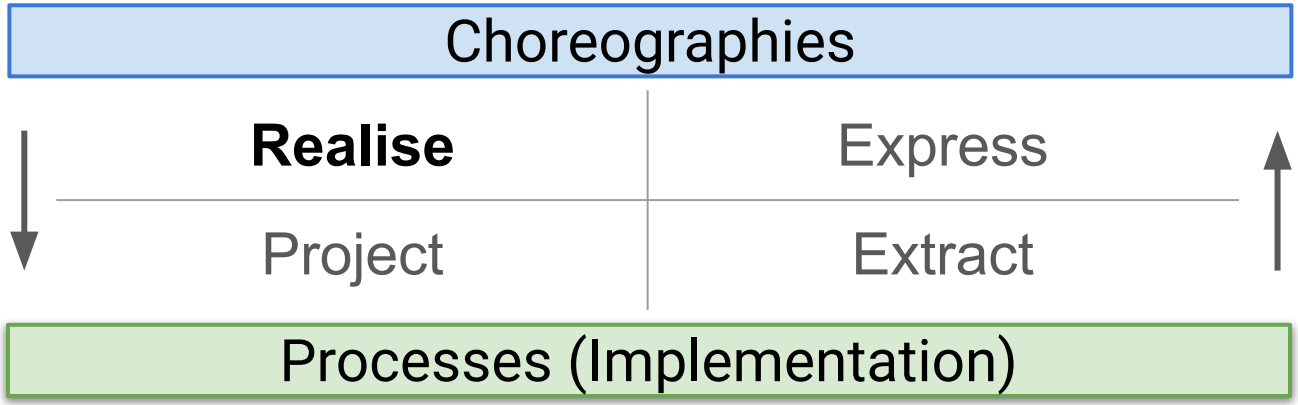


Translational Questions



Realisability (of a choreography):
There exists an implementation.

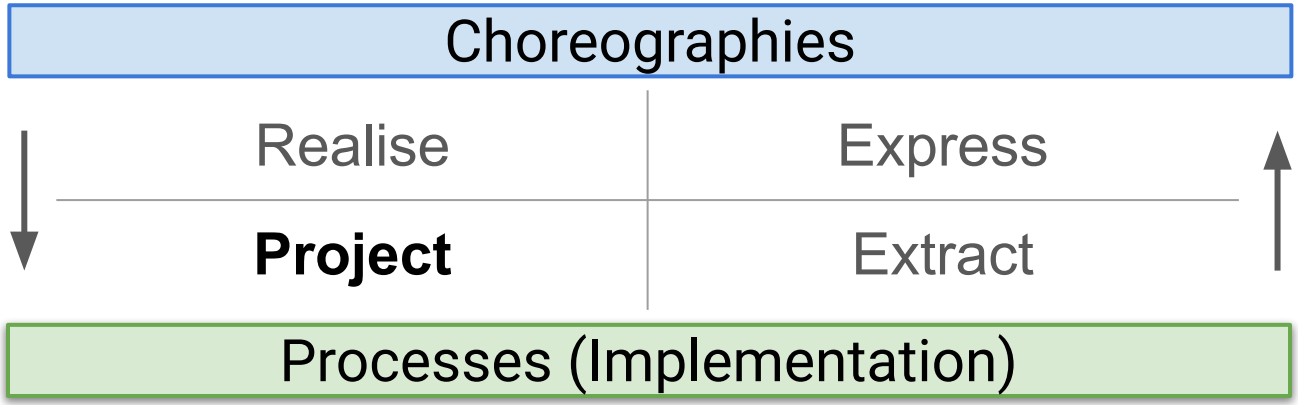
Translational Questions



Realisability (of a choreography):
There exists an implementation.

```
if a.valid(x) {
    ws.newToken() -> c.result;
} else {
    ws.NoToken -> c.result;
}
```

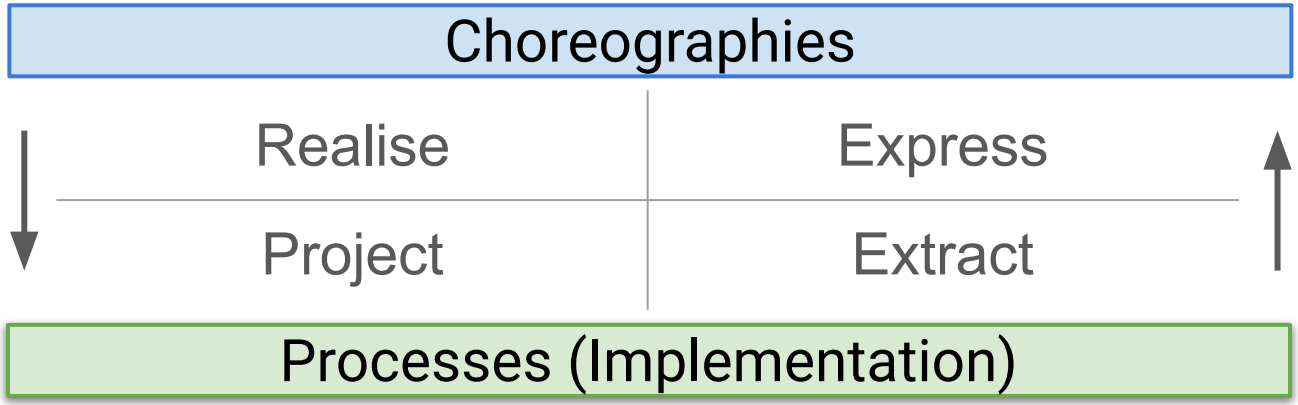
Translational Questions



Realisability (of a choreography):
There exists an implementation.

Projectability (of a choreography):
We can compute an implementation.

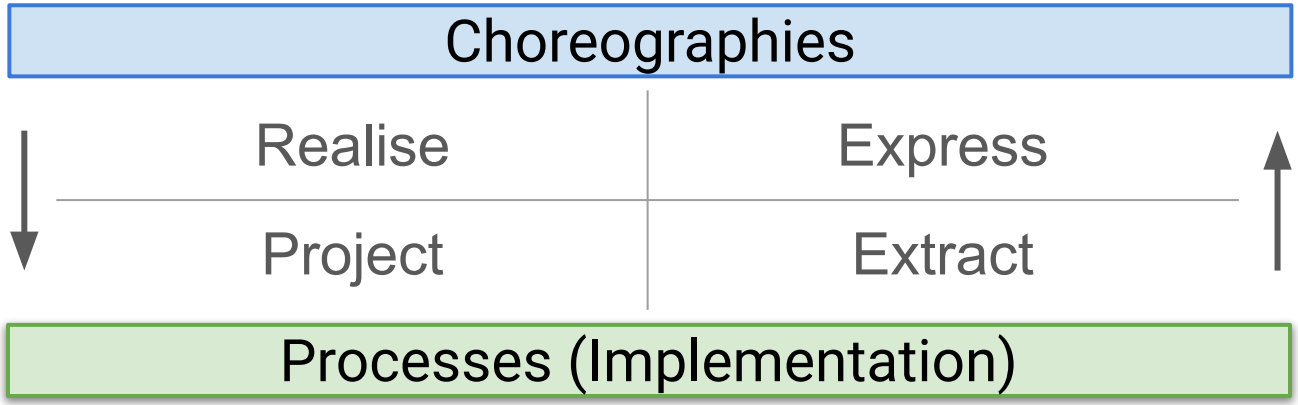
Translational Questions



Amendment:
The transformation of a choreography into a projectable one.

```
if a.valid(x) {
    ws.newToken() -> c.result;
} else {
    ws.NoToken -> c.result;
}
```

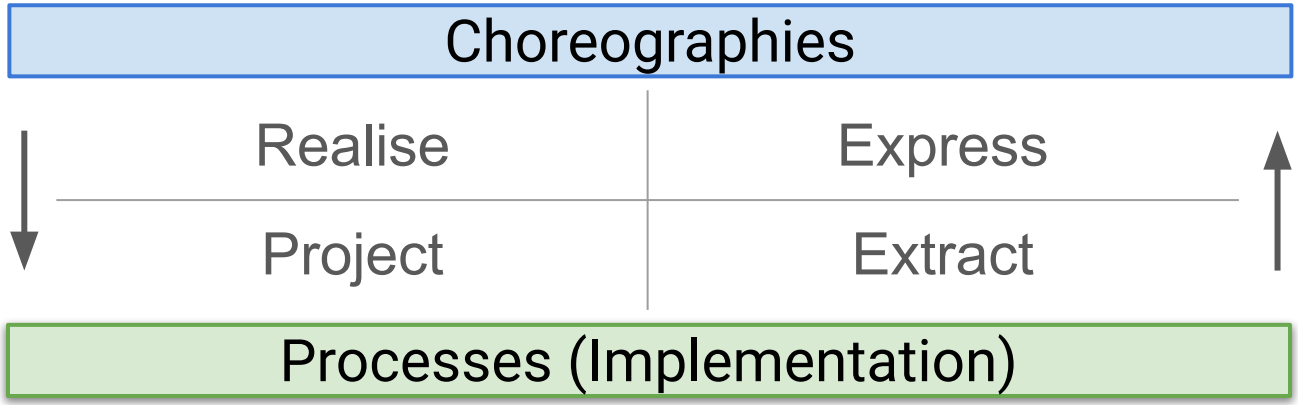
Translational Questions



Amendment:
The transformation of a choreography into a projectable one.

```
if a.valid(x) {

    ws.newToken() -> c.result;
} else {

    ws.NoToken -> c.result;
}
```

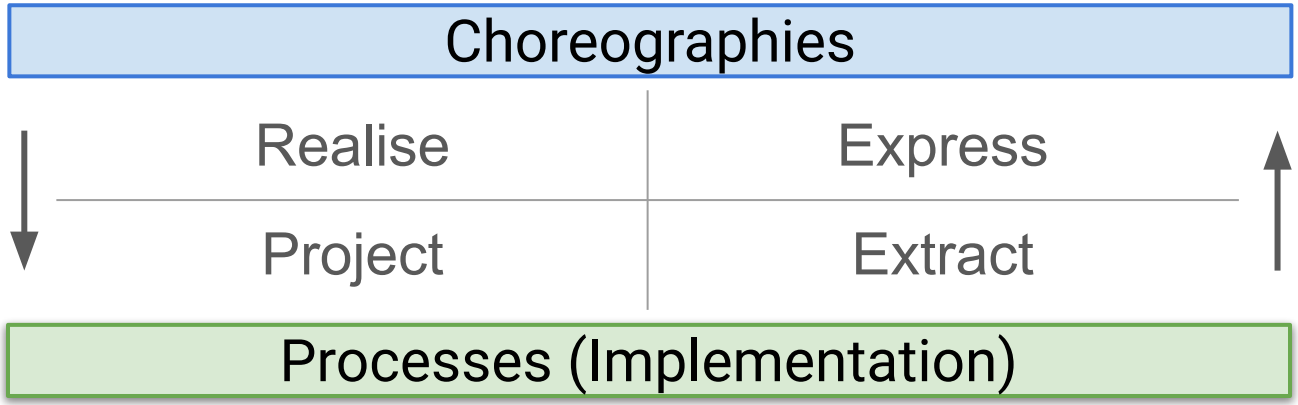
Translational Questions



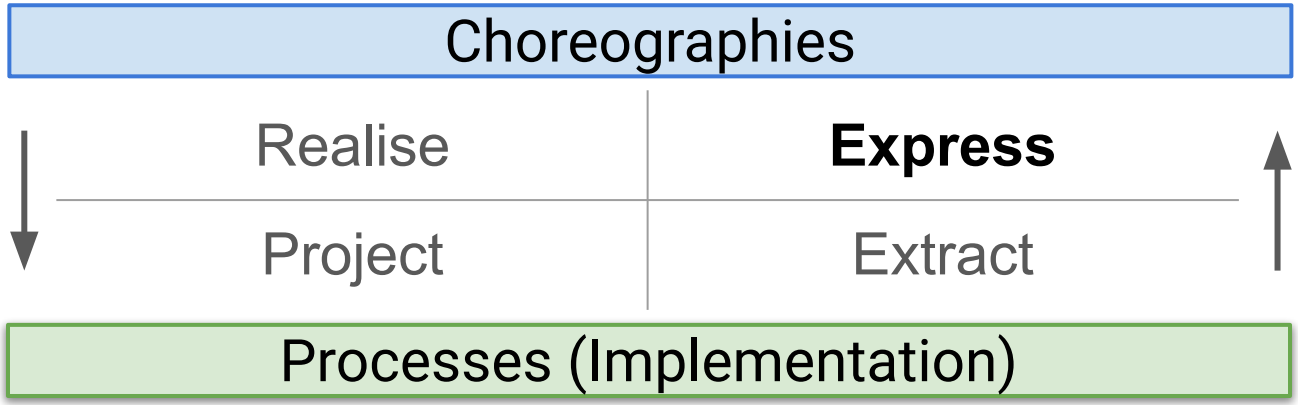
Amendment:
The transformation of a choreography into a projectable one.

```
if a.valid(x) {
    a.OK -> ws.decision;
    ws.newToken() -> c.result;
} else {
    a.KO -> ws.decision;
    ws.NoToken -> c.result;
}
```

Translational Questions

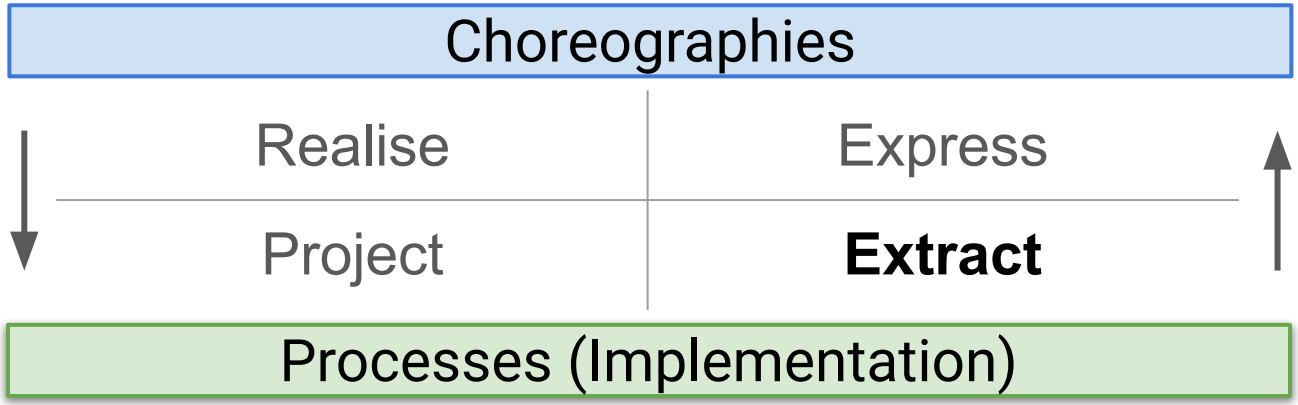


Translational Questions



Expressivity:
Can we write a choreography of a given implementation?

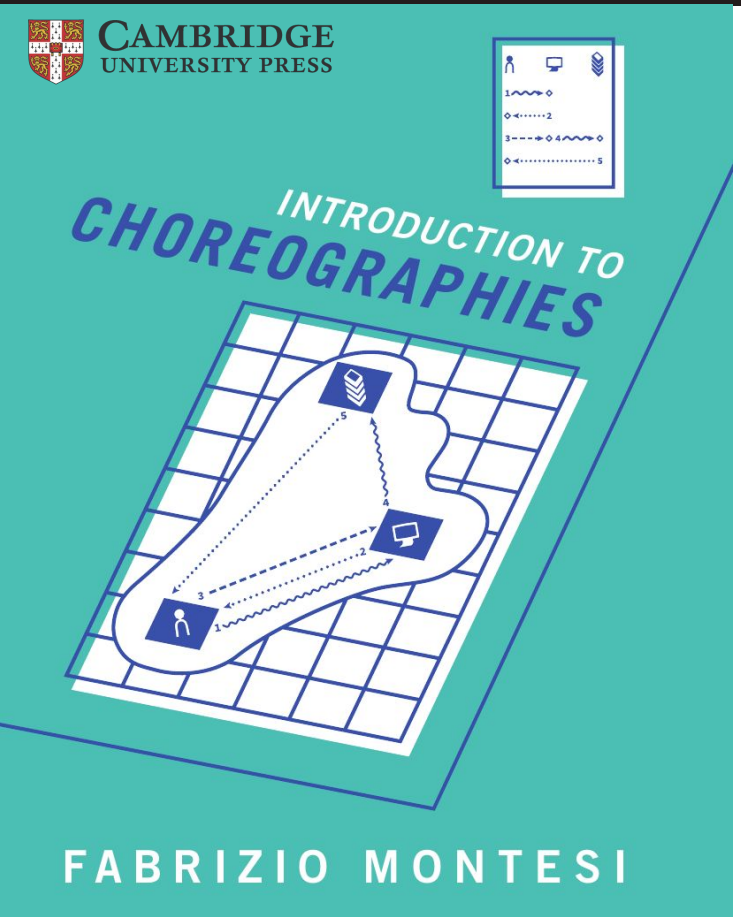
Translational Questions



Expressivity:
Can we write a choreography of a given implementation?

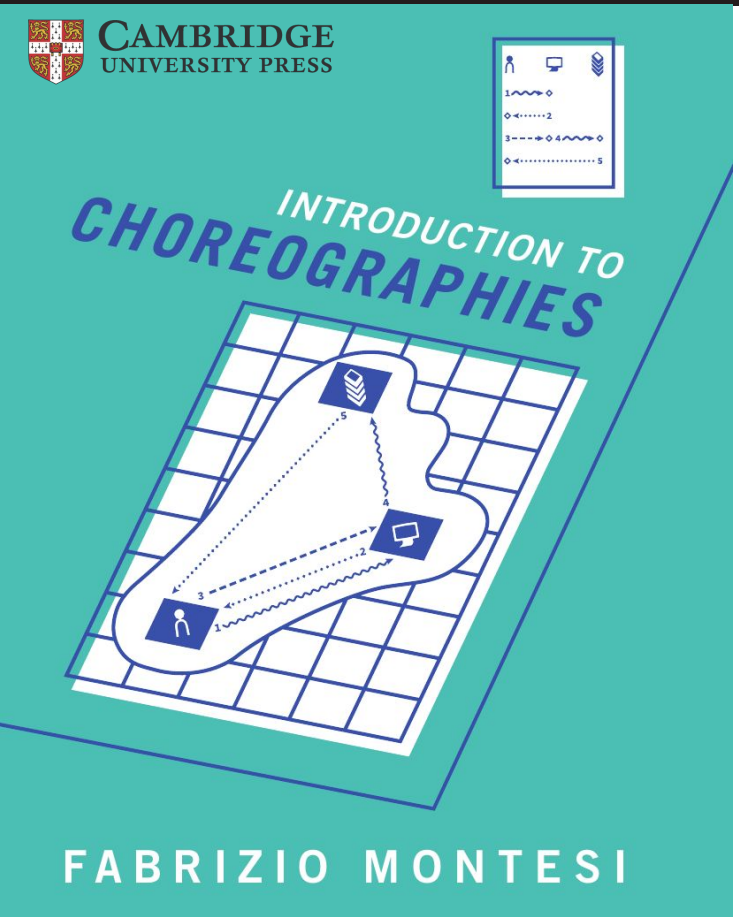
Extraction:
Can we compute a choreography of a given implementation?

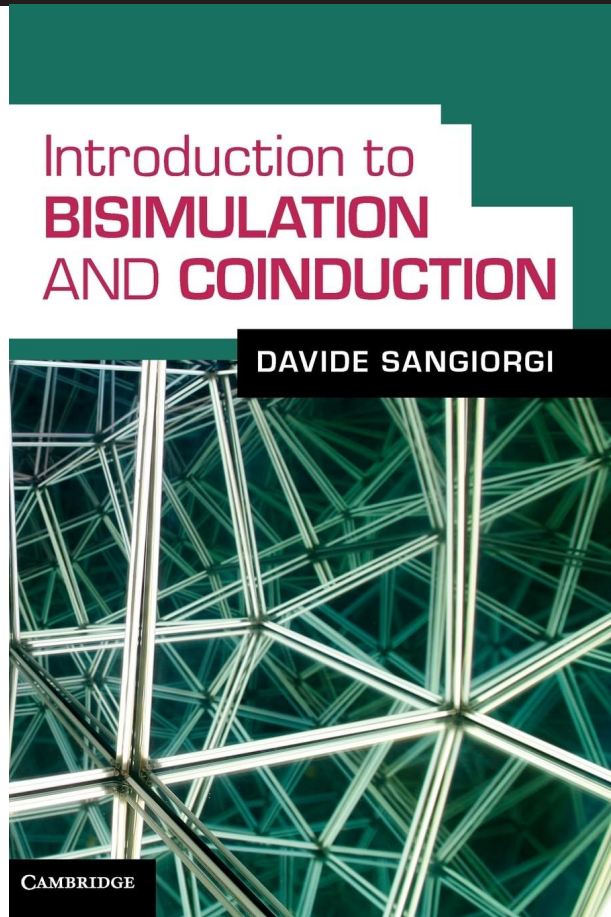
Material



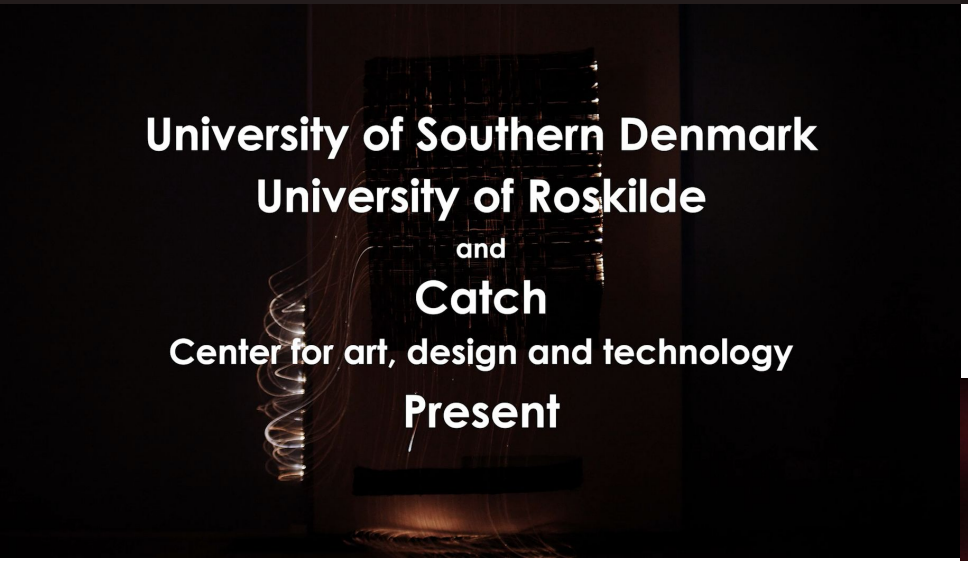
- ➔ Beginner friendly.
- ➔ Structural Operational Semantics.
- ➔ Algebraic properties for compilation.

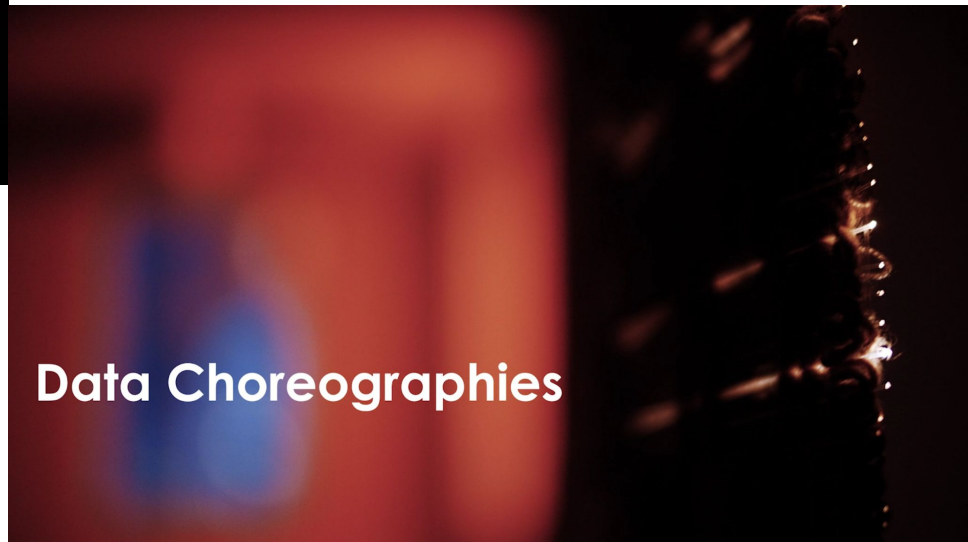
Material


CAMBRIDGE
UNIVERSITY PRESS
INTRODUCTION TO
CHOREOGRAPHIES
FABRIZIO MONTESI

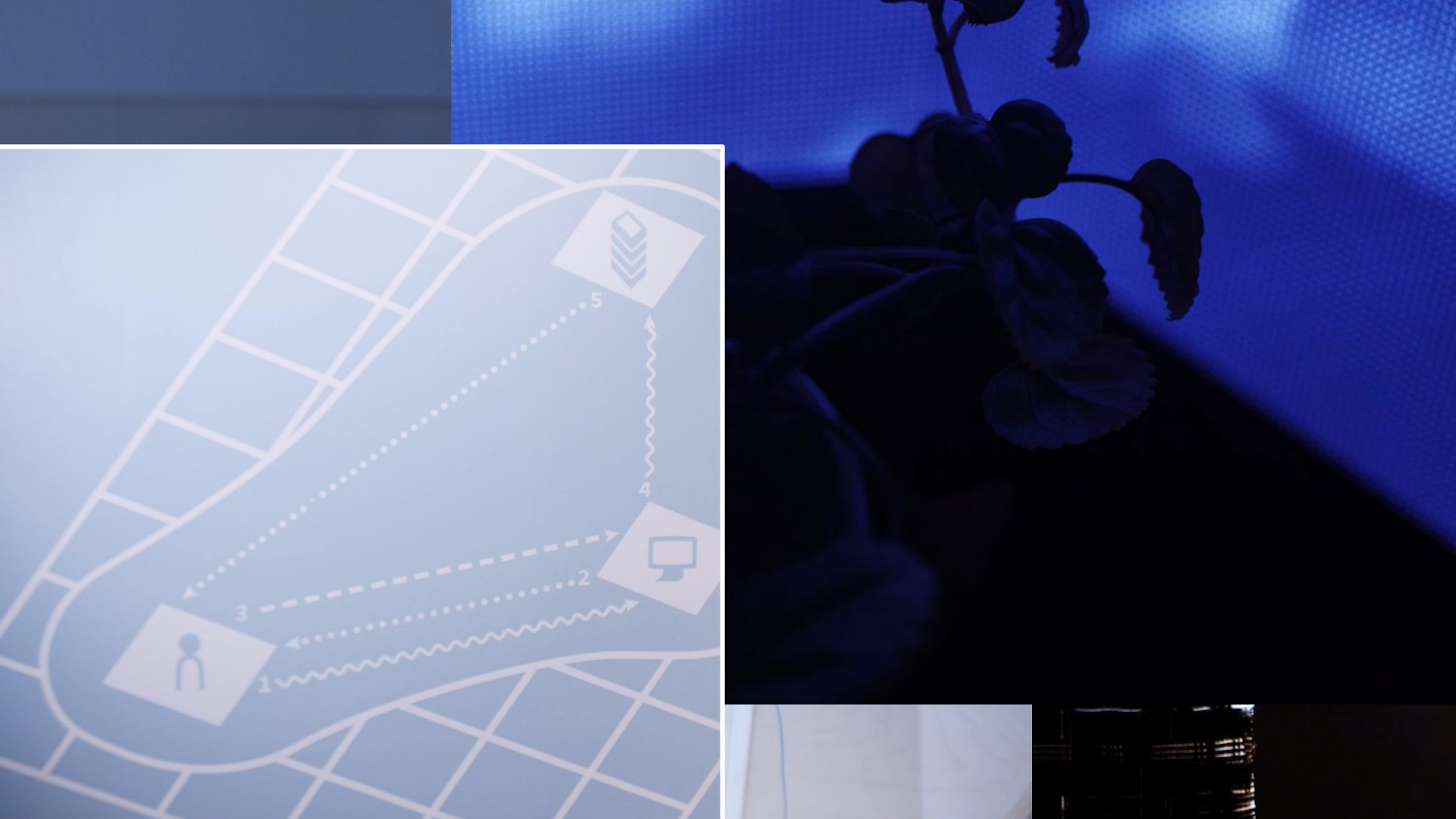

Introduction to
BISIMULATION
AND COINDUCTION
DAVIDE SANGIORGI
CAMBRIDGE

The ‘Data Choreographies’ art exhibition






Salesperson: Pizza is amazing. It's so delicious.
Dog: Pizza.
Salesperson: Yes, it's just so gooooood. It's like money in your mouth.
Dog: I don't like money in my mouth.
Salesperson: You just need to be more assertive when you eat cash


5
4
3
2
1

BOXED: People as Processes

UPCOMING PERFORMANCES

Malmø Festival
(16.08.2024, Malmø, Sweden)

Vestfyns Teaterforening
(18.08.2024, Assens, Denmark)

Tanzmesse
(29.-30.08.2024, Dusseldorf, Germany)

BOXED: People as Processes

UPCOMING PERFORMANCES

Malmø Festival
(16.08.2024, Malmø, Sweden)

Vestfyns Teaterforening
(18.08.2024, Assens, Denmark)

Tanzmesse
(29.-30.08.2024, Dusseldorf, Germany)

BOXED: People as Processes

UPCOMING PERFORMANCES

Malmø Festival
(16.08.2024, Malmø, Sweden)

Vestfyns Teaterforening
(18.08.2024, Assens, Denmark)

Tanzmesse
(29.-30.08.2024, Dusseldorf, Germany)


COLLABORATION

X-IDF: Explainable Internet Data Flows

Fabrizio Montesi and Claes de Vreese

VILLUM FONDEN

X-IDF: Explainable Internet Data Flows

Fabrizio Montesi and Claes de Vreese

VILLUM FONDEN

```
from User to Pharmacy
```

X-IDF: Explainable Internet Data Flows

Fabrizio Montesi and Claes de Vreese

VILLUM FONDEN

```
from User to Pharmacy
```

```
User -> Pharmacy
```

X-IDF: Explainable Internet Data Flows

Fabrizio Montesi and Claes de Vreese

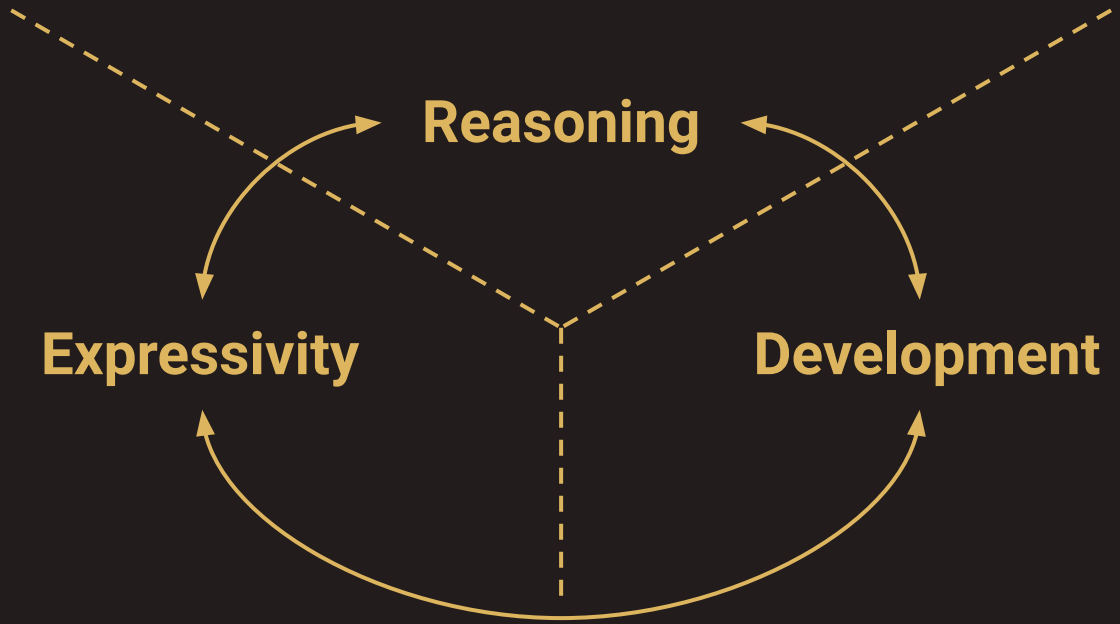
VILLUM FONDEN

```
from User to Pharmacy
```

```
User -> Pharmacy
```

PROJECT CHORDS

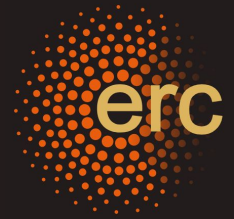
erc



PROJECT CHORDS

erc

PROJECT CHORDS



★ Organising knowledge on choreographic programming.

★ Organising knowledge on choreographic programming.

★ Generalised theory with intermediate languages.

PROJECT CHORDS

★ Organising knowledge on choreographic programming.

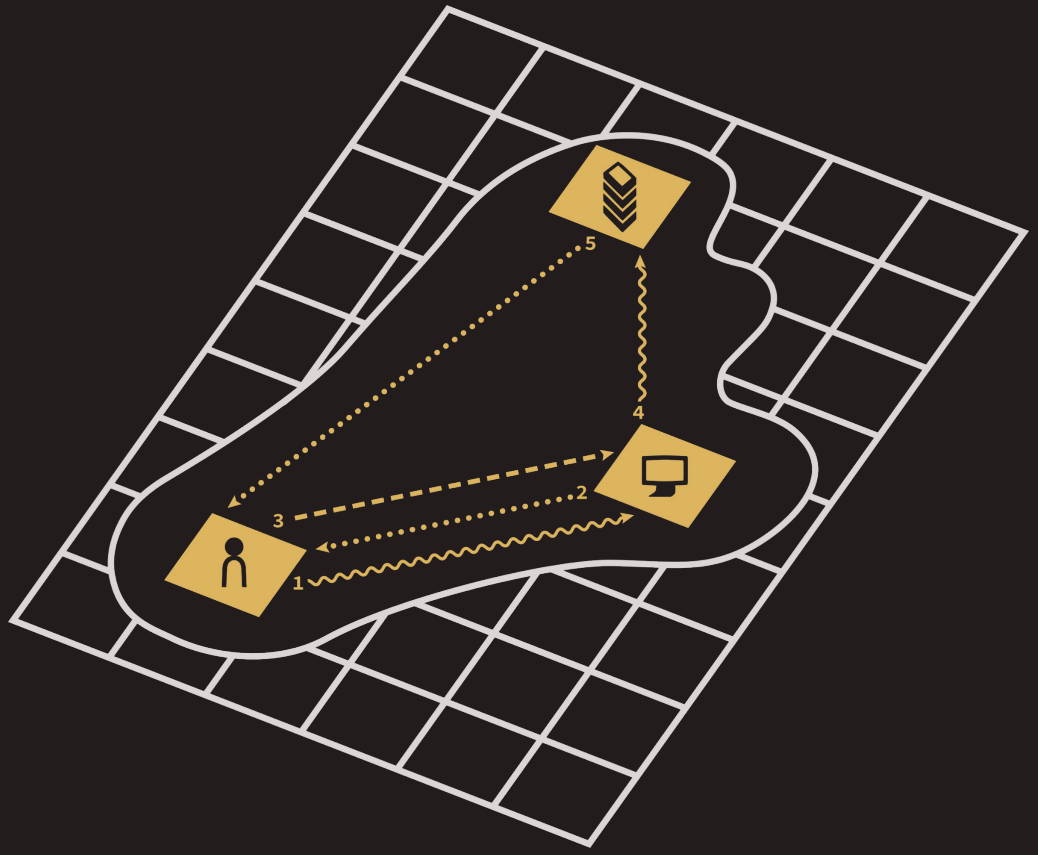
★ Generalised theory with intermediate languages.

PROJECT CHORDS

- ★ Organising knowledge on choreographic programming.
- ★ Generalised theory with intermediate languages.
- ★ Benchmark.

Thank you for listening!

Q&A



Funded by the European Union

European Research Council
Established by the European Commission

Co-funded by the European Union (ERC, CHORDS, 101124225). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

VILLUM FONDEN