# Behavioral Types for Local-First Software

*Roland Kuhn (Actyx)*     *Hernán Melgratti (UBA)*     *Emilio Tuosto (GSSI)*

ECOOP'23     July 17–21, 2023     Seattle, WA, USA

## Motivation

What we want:

- ❑ systems that **never stop**
- ❑ “keep going” is more important than “no mistakes”

Therefore:

- ❑ local agents must be able to act, always
- ❑ **perfect availability** (i.e. punting on strong consistency)

## Key idea

append events to **local logs**

→ **replicate** logs

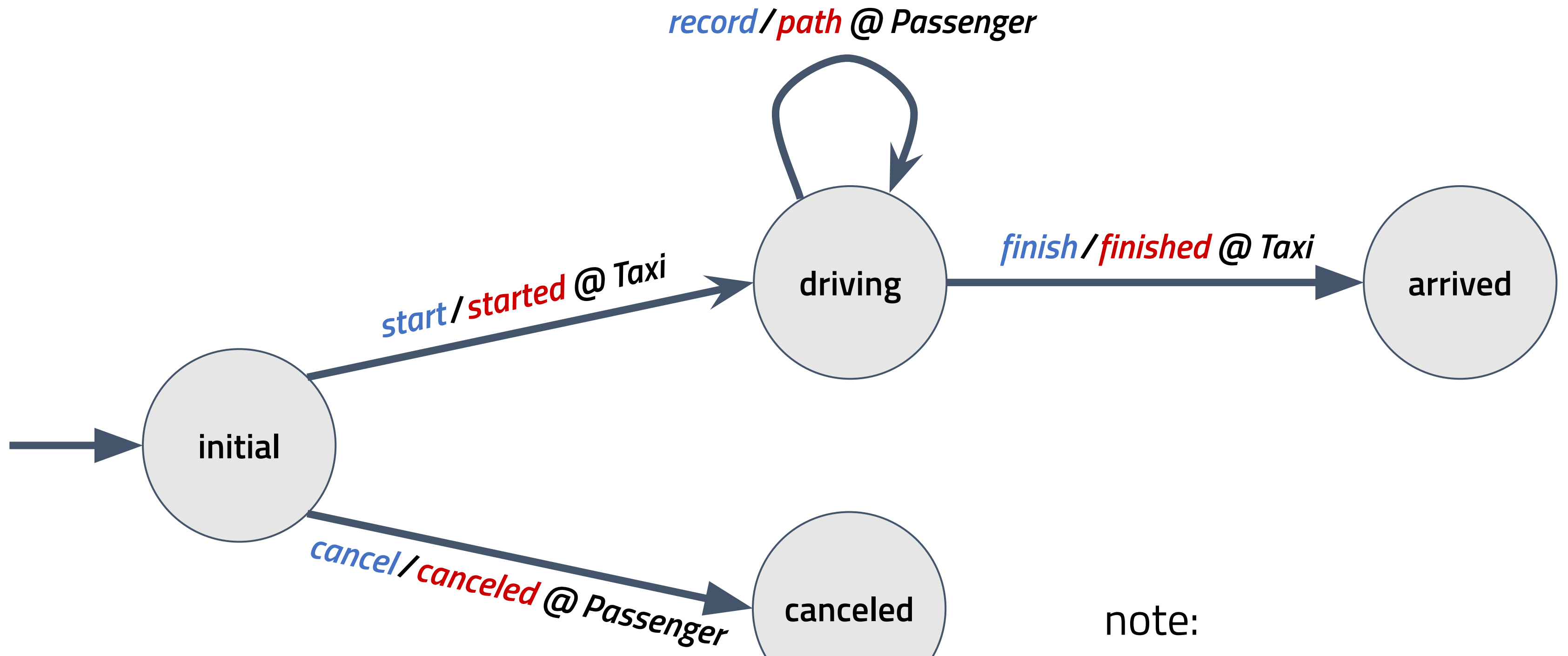
→ **merge** logs

→ locally **interpret** logs

→ *eventual consensus*

***instead of coordinating*** *the sequence of transactions added to a global log using consensus*

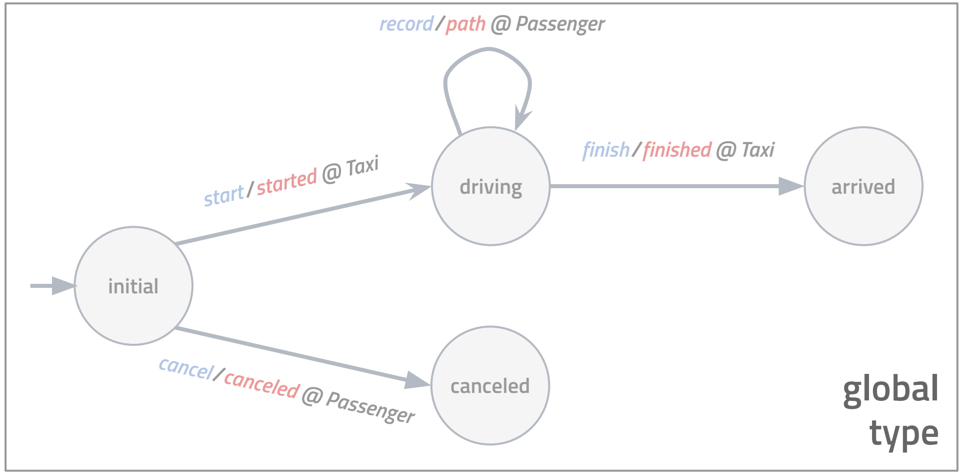
# Example – global type

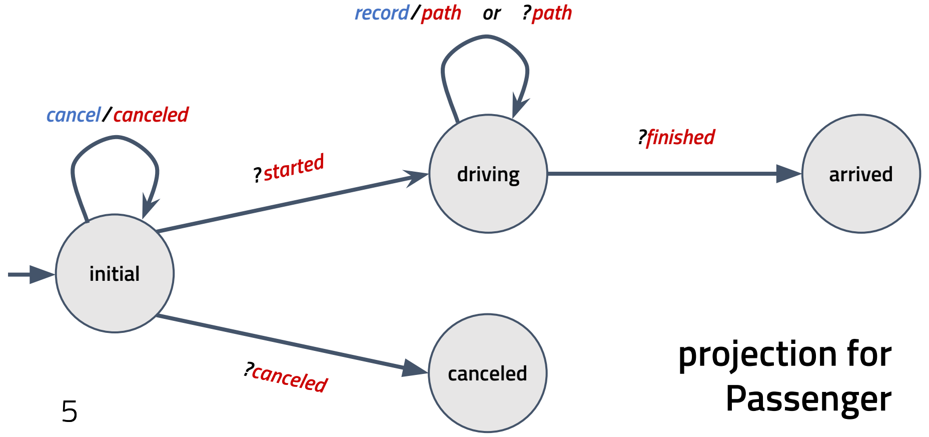


note:

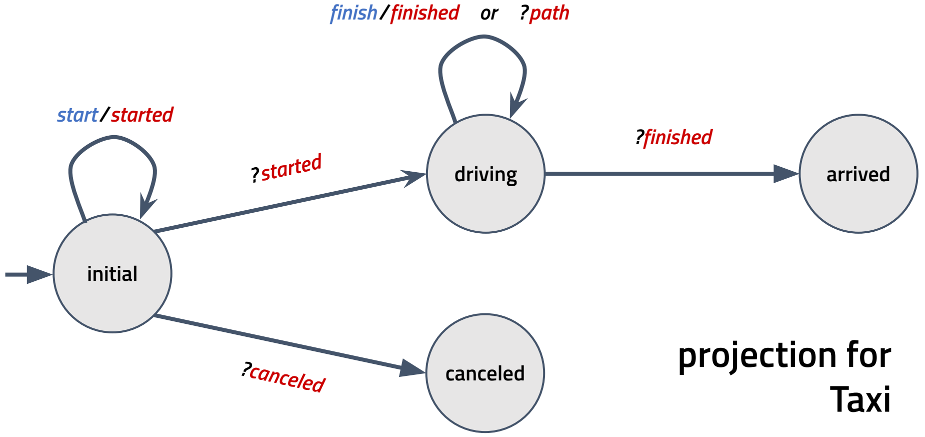
- ❏ non-adversarial setting
- ❏ roles can be replicated

# Example – local types





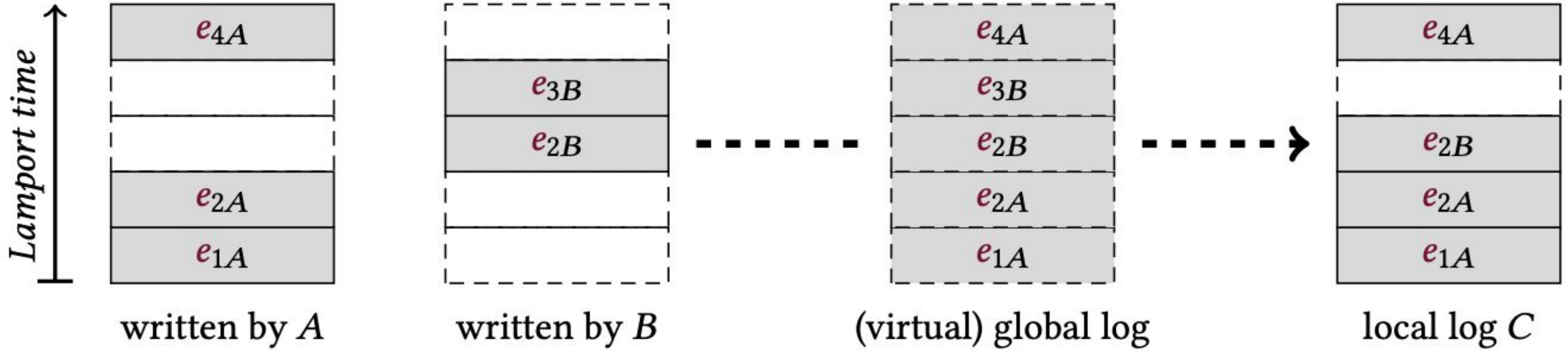
projection for Passenger



projection for Taxi

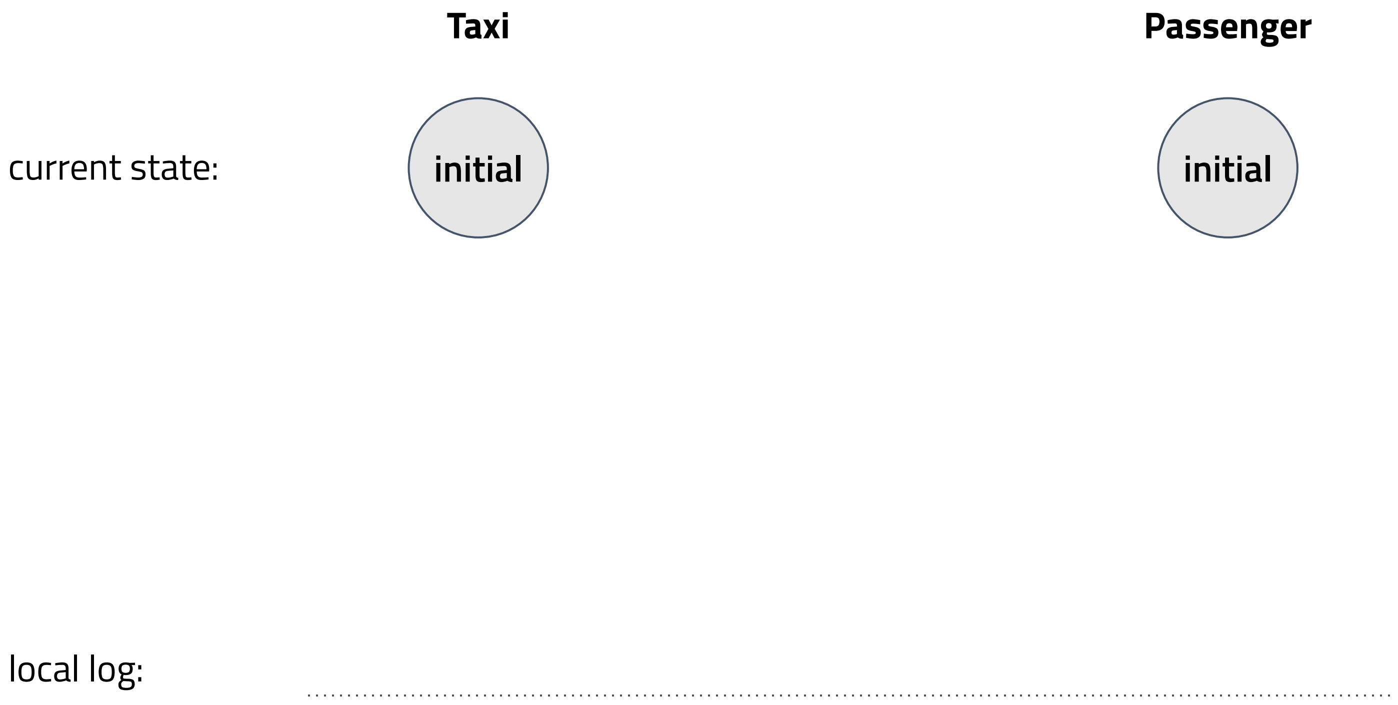
# Event replication

assuming a **coordination-free total order** (e.g. by Lamport timestamp and node ID)

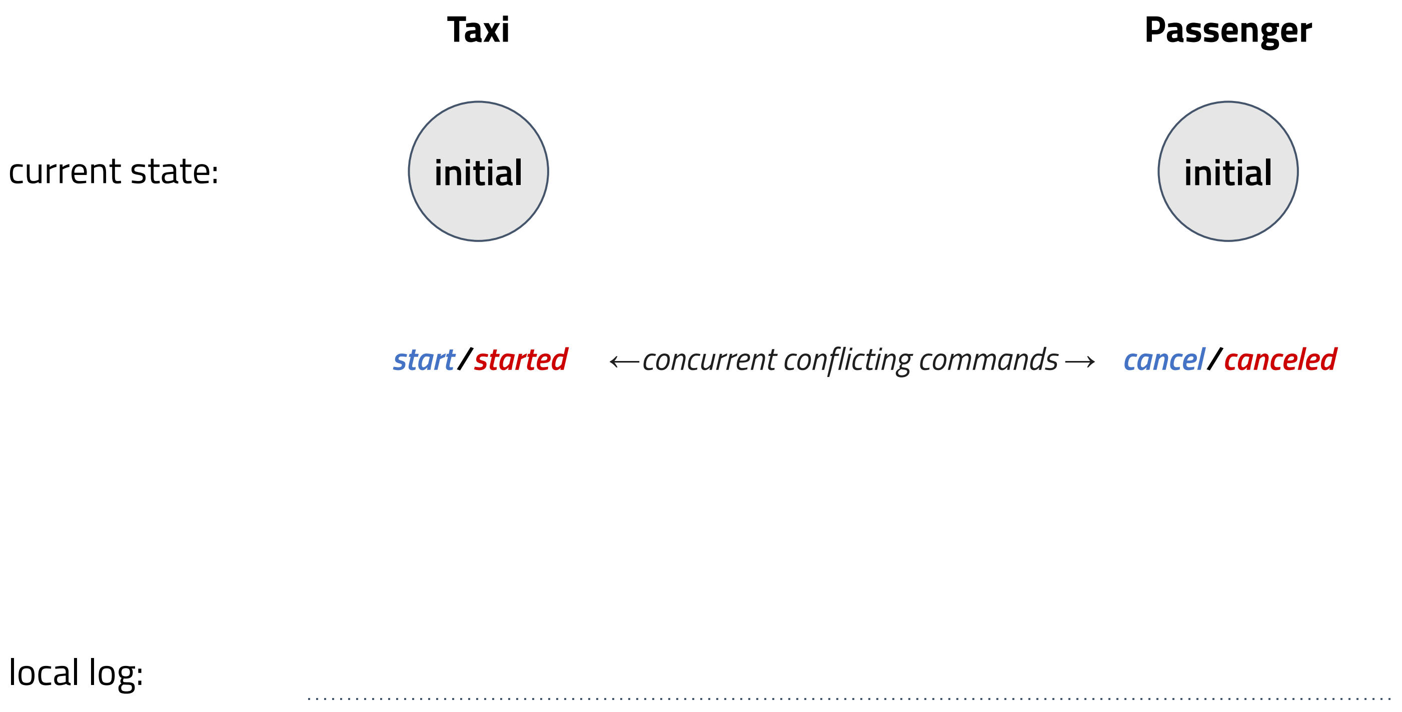


Subscripts of events specify Lamport timestamp and the identity of the machine generating them

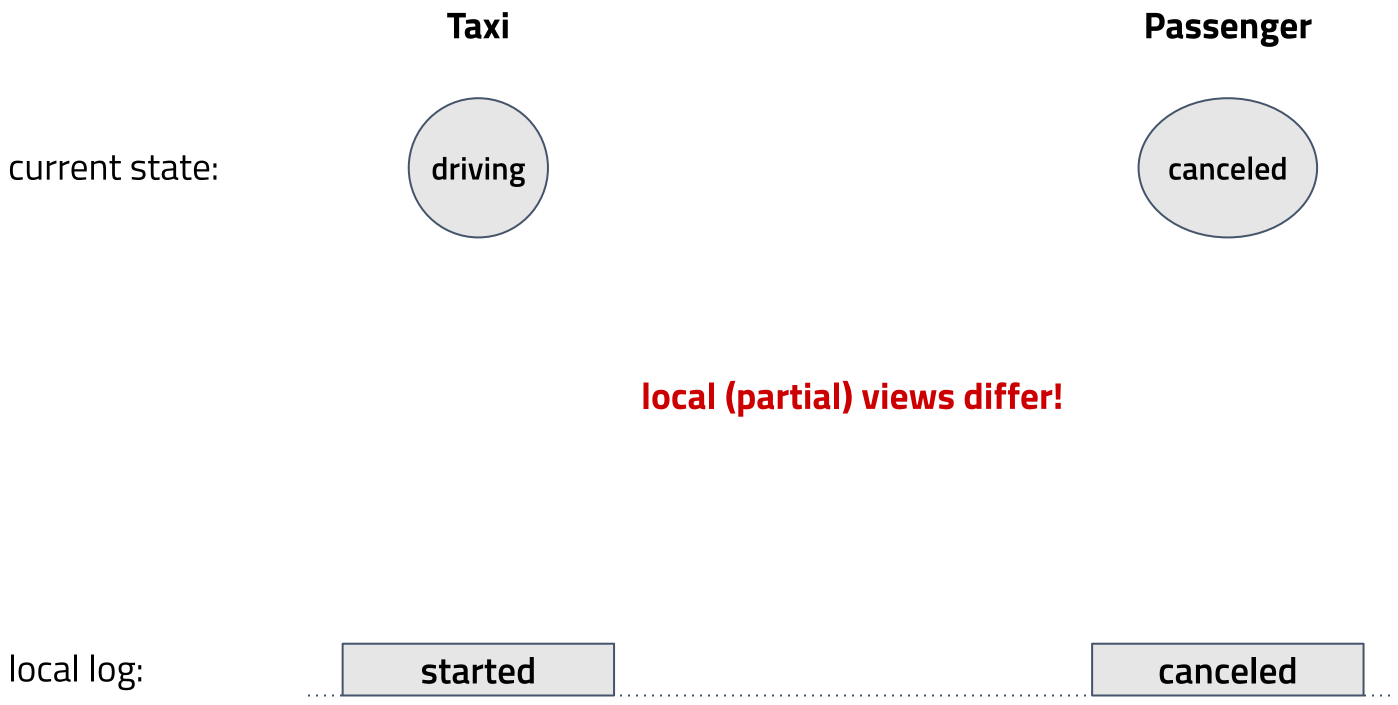
# Example execution



# Example execution

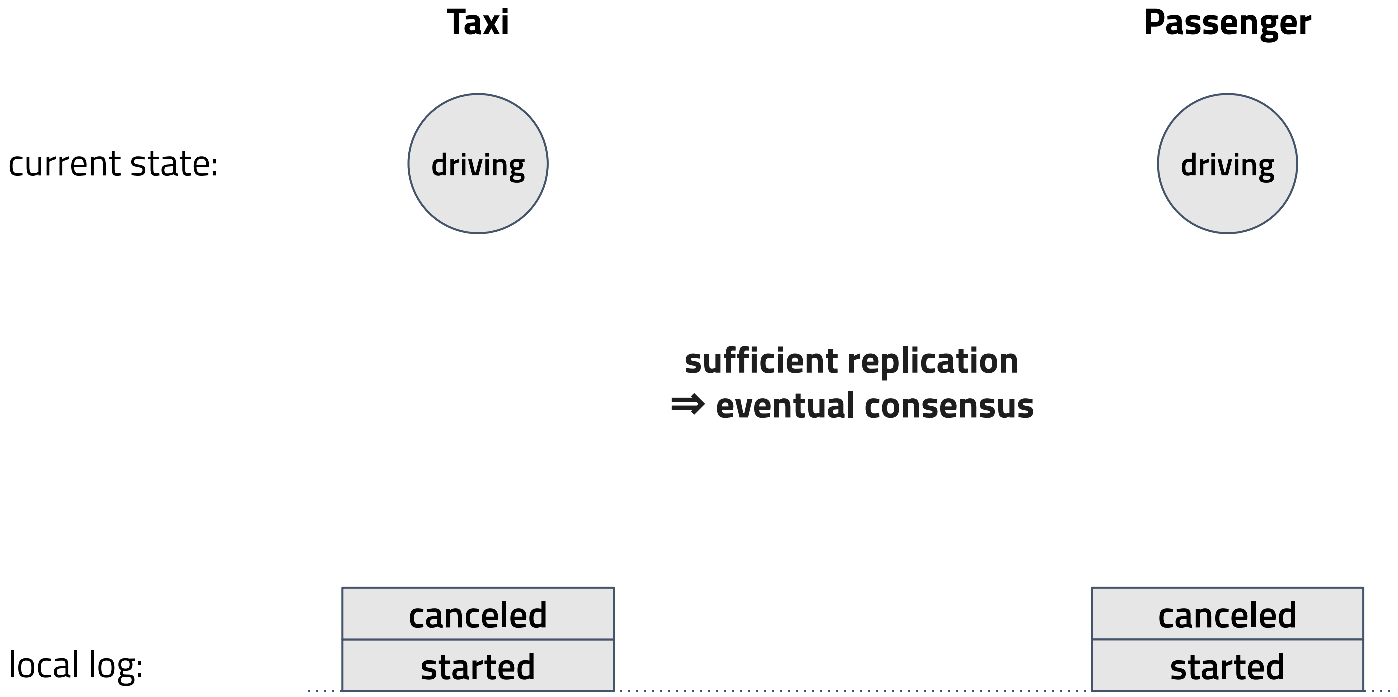


# Example execution



**local (partial) views differ!**

# Example execution



**sufficient replication**
**⇒ eventual consensus**

## Key idea

append to **local logs**

→ **replicate** logs

→ **merge** logs

→ locally **interpret** logs

→ *eventual consensus*

***instead of coordinating*** *the sequence of transactions added to a global log using consensus*

## Eventual Consensus

Not every role sees every event: **well-formedness conditions** needed!

*causality (react to own events, wait for enabling events)*

*determinacy (must follow along if involved later)*

*confusion-freeness (guard events must be used unambiguously)*

All three are decidable in less than $\mathcal{O}(n^3)$.

## Sequence of ideas

On typing disciplines:

- ❑ **multi-party session types** are easy to understand but not expressive enough
- ❑ session types with **timeouts or failures** solve some cases by forcing a new session
- ❑ **mailbox types** (de'Liguoro & Padovani, ECOOP'18) allow general concurrency but require all messages to be handled, with order chosen by recipient

On conflict resolution:

- ❑ **CRDTs** prevent conflicts but are difficult to design
- ❑ **time warp machine** (speculative execution and roll-back)

**Customer input:**

- ❑ process flow charts, activity diagrams, state machines, collaboration diagrams, ...

*references are cited in the paper*

## Current state

- ❏ **proven theory**
  - ❏ deadlock-free by construction
  - ❏ eventual consensus
  - ❏ communication-safe by filtering
  - ❏ orphans detected (later: → conflict compensation)
- ❏ protocol well-formedness and projection conformance checking is **implemented in Haskell & Rust**
- ❏ **TypeScript API** for machines has evolved already
  ⇒ ISSTA tool demonstration today (4–5pm)

## Future work

- ❏ refine well-formedness conditions to get closer to **necessity**
- ❏ refine evaluation model to achieve **branch non-interference**
- ❏ cover **adversarial** settings

Actyx

**Actyx AG**
Max-Bill-Straße 38
80807 Munich

contact@actyx.io | www.actyx.com