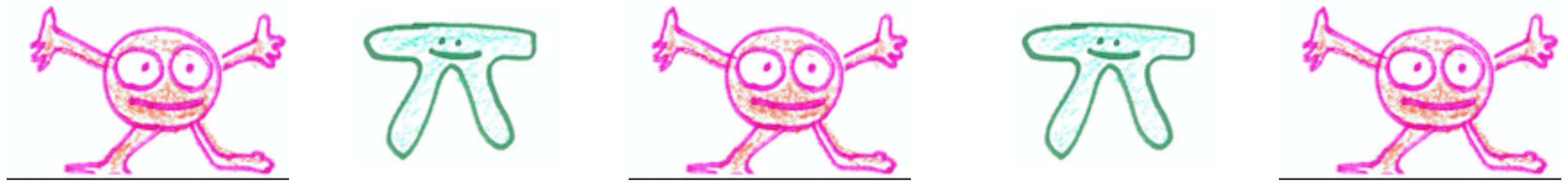
Deadlock-Free Asynchronous Message Reordering in Rust with Multiparty Session Types

27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming **[PPoPP 2022]**



Zak Cutner, NY and Martin Vassor

Imperial College London

Huawei 24th May 2022

Communications are Ubiquitous

- Increasingly, **communications** are the way to organise software and systems.
- Industry trend – programming languages with **explicit message-passing primitives.**

microservices

python™
Scala
Java
ERLANG
MPI
akka
OCaml

Problems: Concurrency Bugs

- Communications increase **concurrency bugs**
  - Survey of 4k users **[golang.org]**
  - Analysis of 6 large software systems **[ASPLOS 19]** **[PLDI 22]**

docker kubernetes Uber

The Go Gopher

Do not communicate by sharing memory; share memory by communicating

– Go Philosophy

Problems: Concurrency Bugs

- Communications increase **concurrency bugs**
  - Survey of 4K users **[golang.org]**
  - Analysis of 6 large software systems **[ASPLOS 19]**

Uber's 14 million lines of Go hosting 2100 microservices [PLDI 22]

deadlock

channel errors

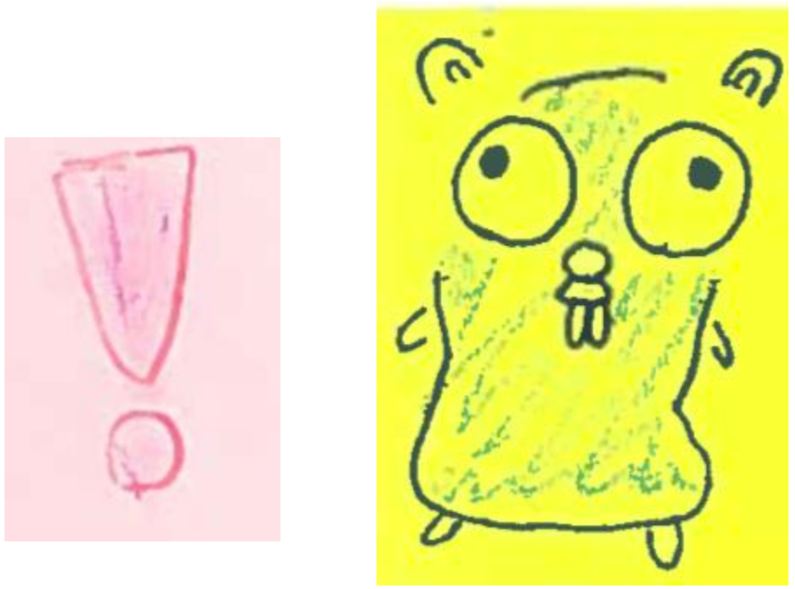
More than a half of concurrency bugs in Go are caused by communications.

The Go Gopher

Problems: Concurrency Bugs

- Communications increase **concurrency bugs**
  - Survey of 4k users **[golang.org]**
  - Analysis of 6 large software systems **[ASPLOS 19]** **[PLDI 22]**

More than a half of concurrency bugs in Go are caused by communications.



Session Types

- Prevent concurrency bugs.
- Can abstract, implement and manage communications as **Protocols.**
- **Clean, Cheap** and **Retrofittable.**

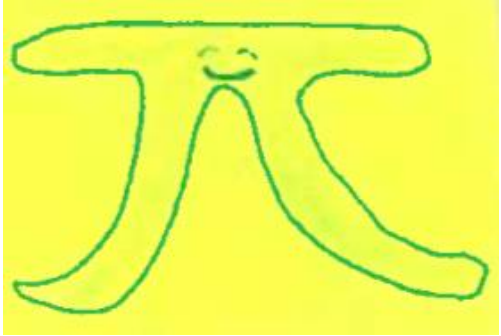
Why Session Types, Why Now?

Significant academic and industry interests via fundamental breakthroughs

Milner, Honda, NY

Binary Session Types

ESOP'98

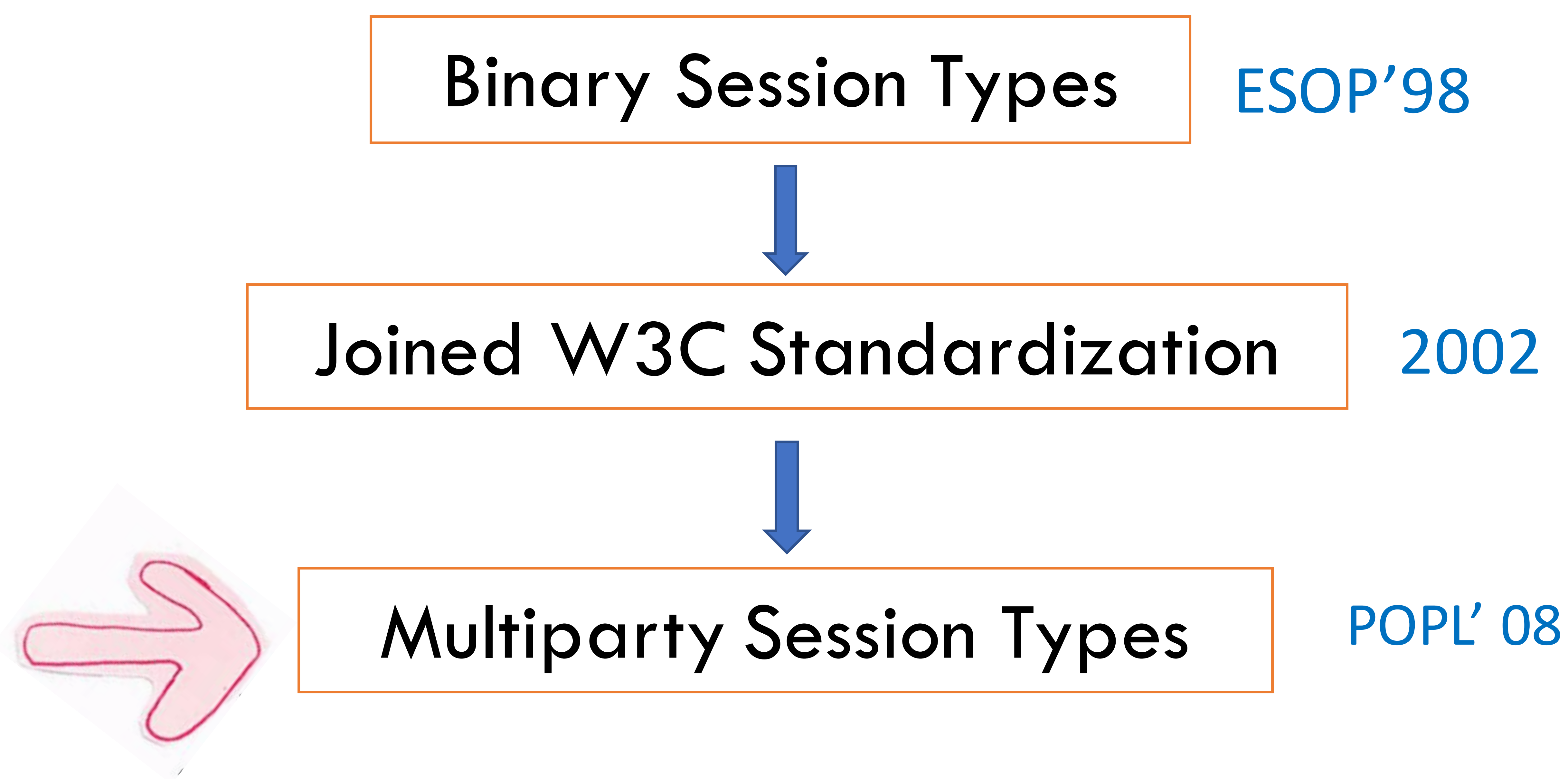


Joined W3C Standardization

2002

Why Session Types, Why Now?

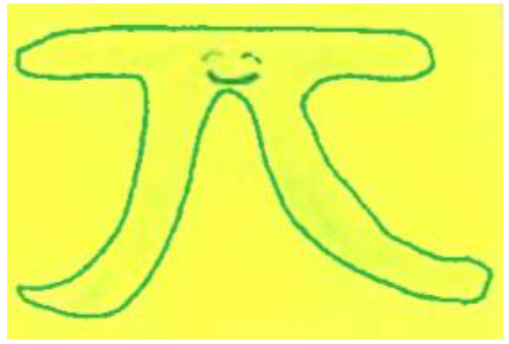
Significant academic and industry interests via fundamental breakthroughs



Binary Session Types ESOP'98

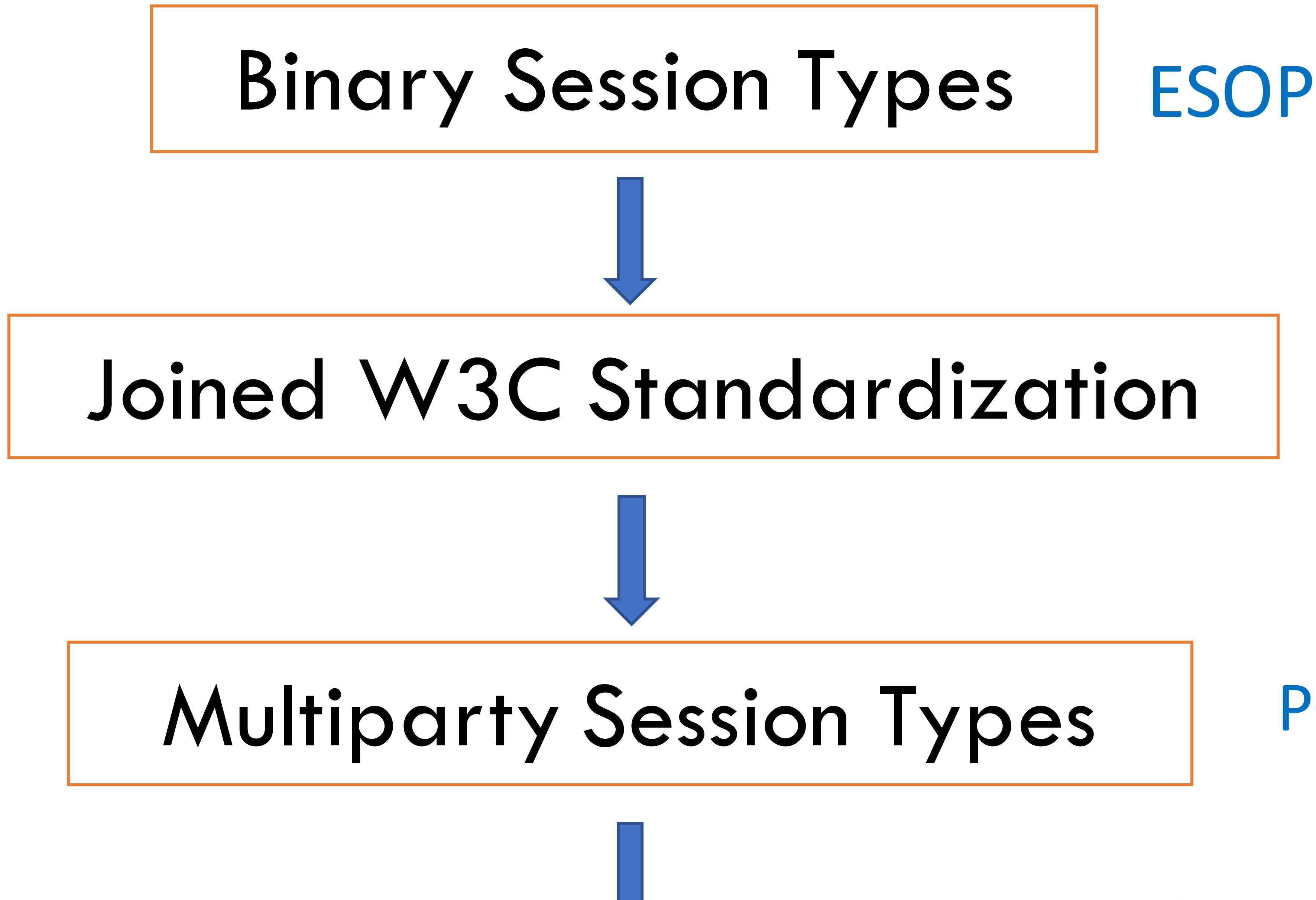
Joined W3C Standardization 2002

Multiparty Session Types POPL' 08

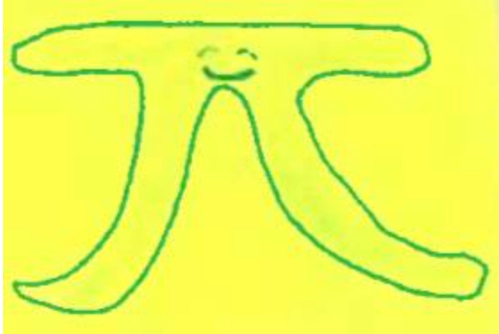


Why Session Types, Why Now?

Significant academic and industry interests via fundamental breakthroughs



Binary Session Types ESOP’98



Joined W3C Standardization 2002

Multiparty Session Types POPL’ 08

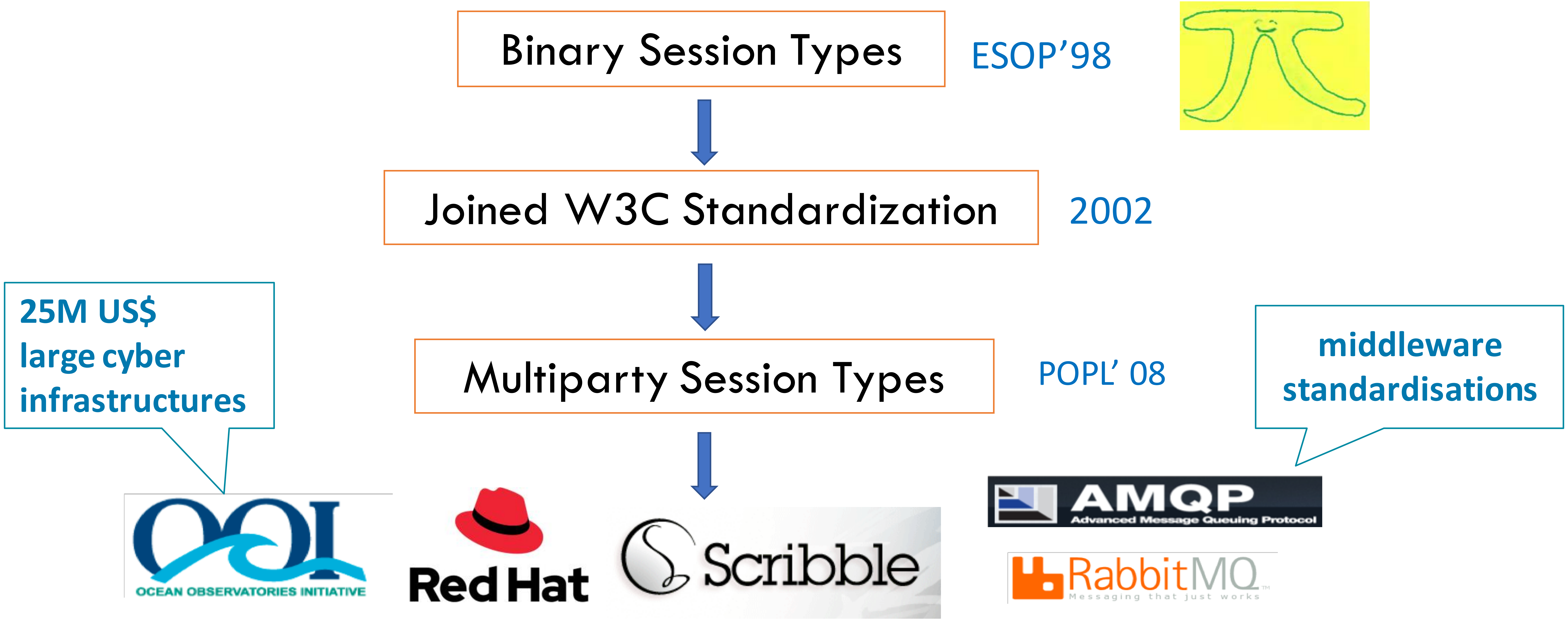
largest open source company in the world

Red Hat

Scribble

Why Session Types, Why Now?

Significant academic and industry interests via fundamental breakthroughs



Binary Session Types

ESOP’98

Joined W3C Standardization

2002

25M US$ large cyber infrastructures

Multiparty Session Types

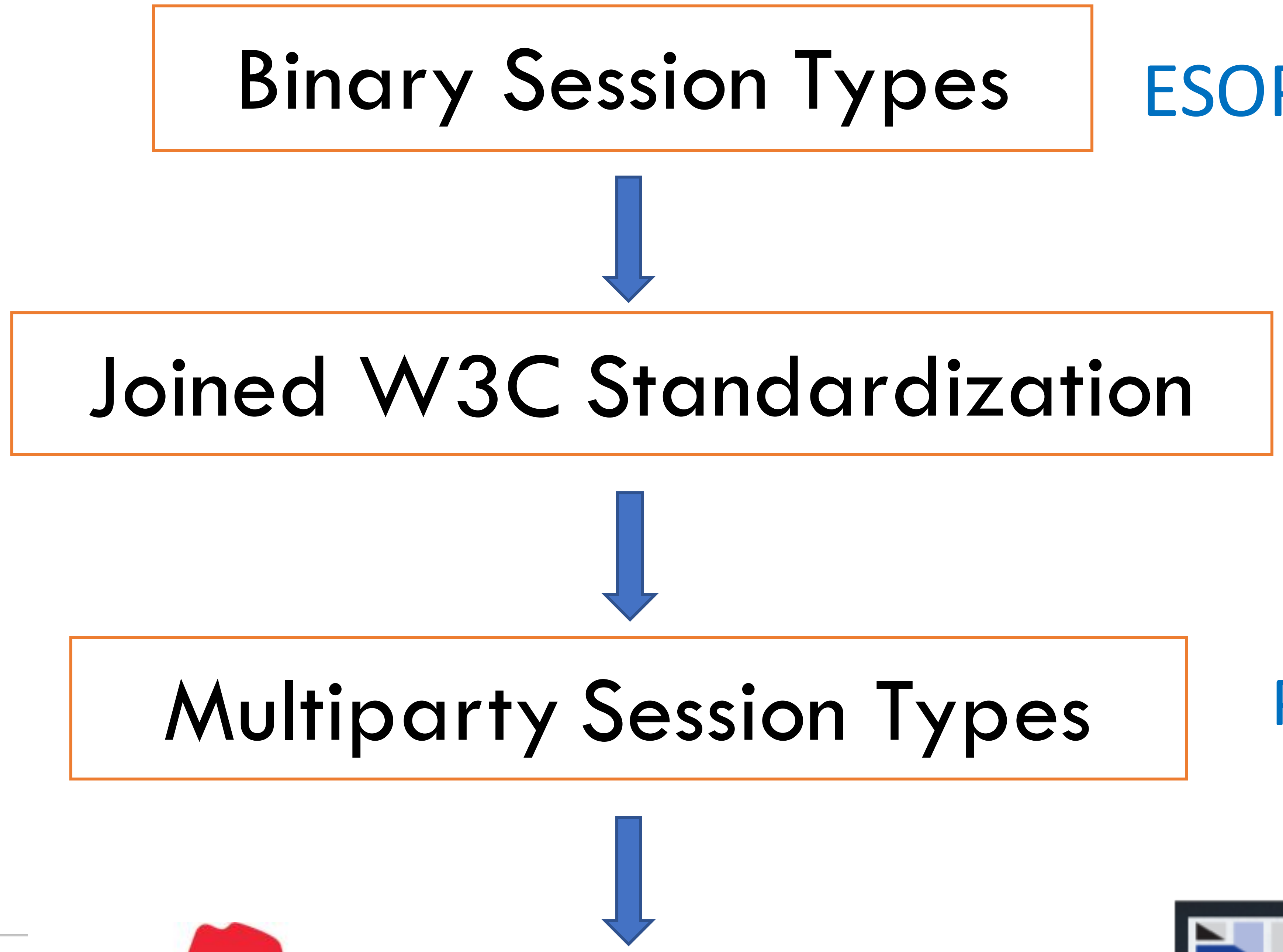
POPL’ 08

middleware standardisations

OOI
OCEAN OBSERVATORIES INITIATIVE
Red Hat
Scribble
AMQP
Advanced Message Queuing Protocol
RabbitMQ
Messaging that just works

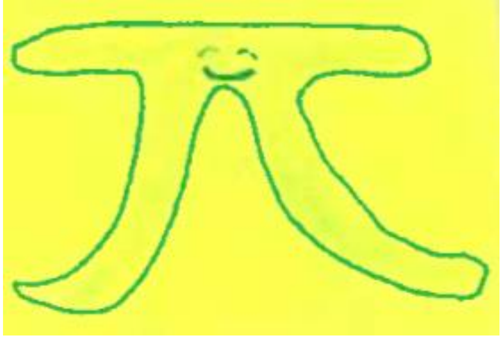
Why Session Types, Why Now?

Significant academic and industry interests via fundamental breakthroughs



Binary Session Types

ESOP’98



Joined W3C Standardization

2002

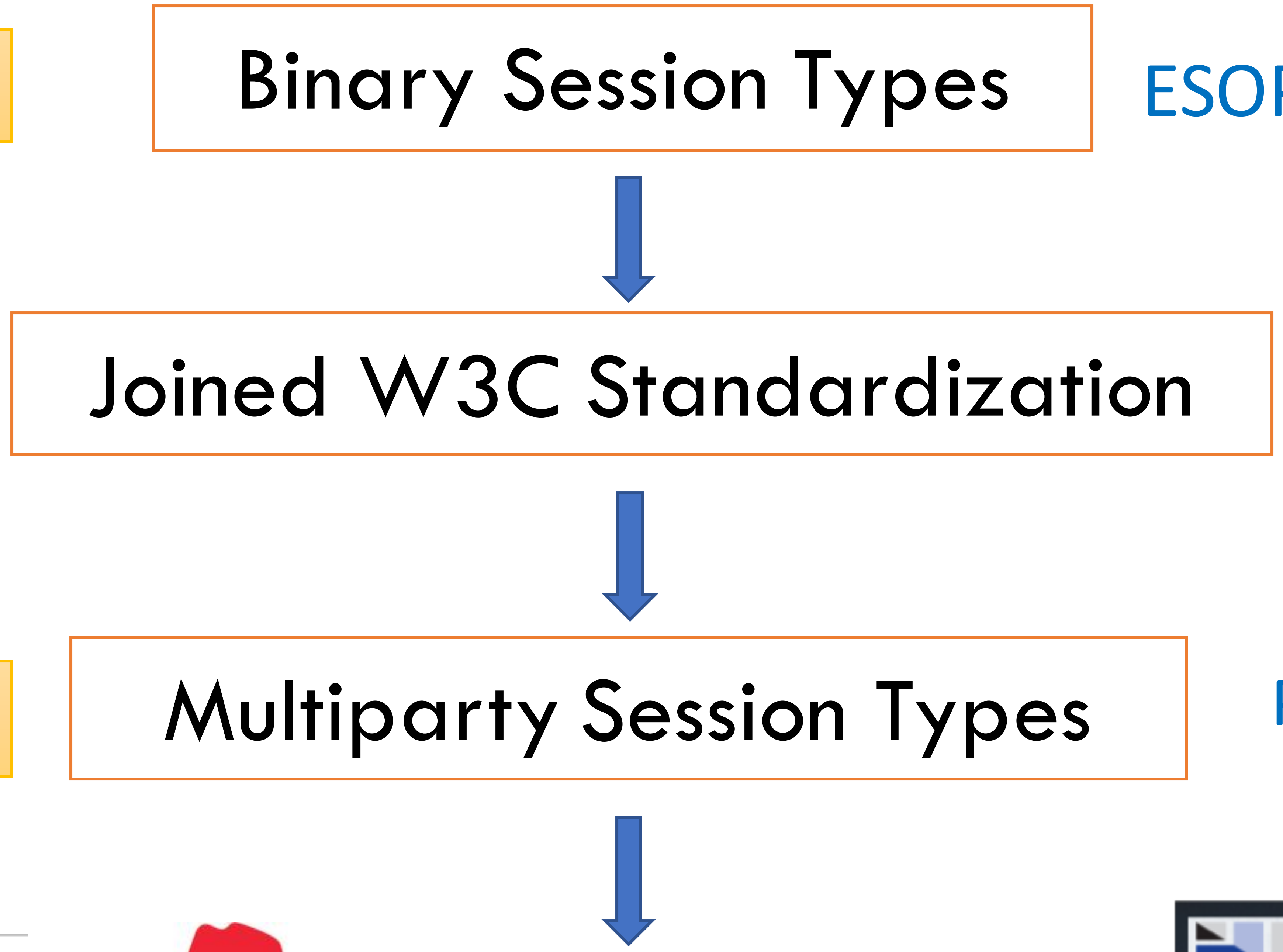
Multiparty Session Types

POPL’ 08

Why Session Types, Why Now?

Significant academic and industry interests via fundamental breakthroughs

ETAPS Test Time Award 2019

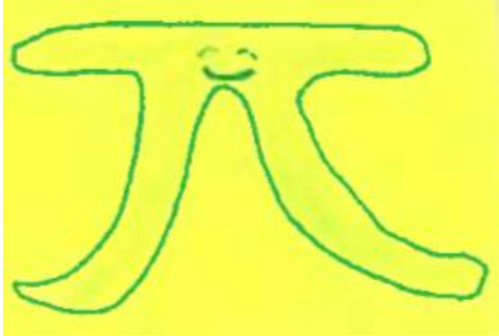


Binary Session Types

Joined W3C Standardization

Multiparty Session Types

ESOP’98



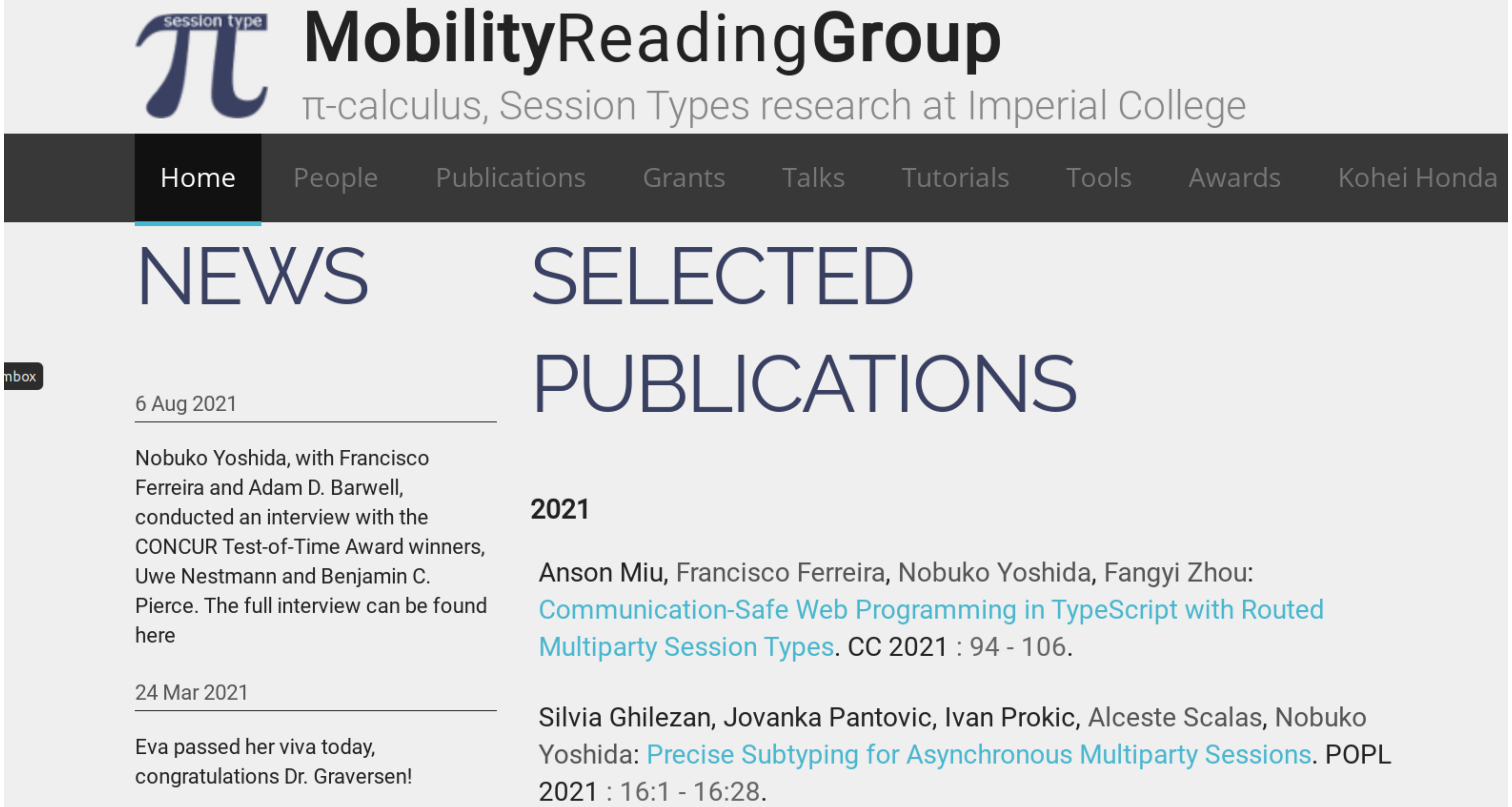
2002

POPL Influential Paper Award 2018

POPL’ 08

Mobility Reading Group

http://mrg.doc.ic.ac.uk/



MobilityReading**Group**

π-calculus, Session Types research at Imperial College

NEWS

6 Aug 2021

Nobuko Yoshida, with Francisco Ferreira and Adam D. Barwell, conducted an interview with the CONCUR Test-of-Time Award winners, Uwe Nestmann and Benjamin C. Pierce. The full interview can be found here

24 Mar 2021

Eva passed her viva today, congratulations Dr. Graversen!

SELECTED PUBLICATIONS

2021

Anson Miu, Francisco Ferreira, Nobuko Yoshida, Fangyi Zhou: Communication-Safe Web Programming in TypeScript with Routed Multiparty Session Types. CC 2021 : 94 - 106.

Silvia Ghilezan, Jovanka Pantovic, Ivan Prokic, Alceste Scalas, Nobuko Yoshida: Precise Subtyping for Asynchronous Multiparty Sessions. POPL 2021 : 16:1 - 16:28.

Introduction

Rust Language

- Modern systems language focussed on safety and performance

Introduction

Rust Language

- Modern systems language focussed on safety and performance
- “Most loved language” for past five years on StackOverflow

Introduction

Rust Language

- Modern systems language focussed on safety and performance
- “Most loved language” for past five years on StackOverflow
- Particular emphasis on safe concurrency using message passing

Introduction

Rust Language

- Modern systems language focussed on safety and performance
- “Most loved language” for past five years on StackOverflow
- Particular emphasis on safe concurrency using message passing
- Affine type system is well-suited to session types

Ring Protocol

Example

Global Type

$$\mathsf{G} = \mu \mathsf{t}.\mathbf{A} \to \mathbf{B} : \left\{ add(\texttt{i32}).\mathbf{B} \to \mathbf{C} : \left\{ \begin{array}{c} add(\texttt{i32}).\mathbf{C} \to \mathbf{A} : \{add(\texttt{i32}).\mathsf{t}\} \\ sub(\texttt{i32}).\mathbf{C} \to \mathbf{A} : \{sub(\texttt{i32}).\mathsf{t}\} \end{array} \right\} \right\}$$

Ring Protocol

Example

$$\mathsf{G} = \mu \mathsf{t}.\mathbf{A} \to \mathbf{B} : \left\{ add(\texttt{i32}).\mathbf{B} \to \mathbf{C} : \left\{ \begin{array}{l} add(\texttt{i32}).\mathbf{C} \to \mathbf{A} : \{add(\texttt{i32}).\mathsf{t}\} \\ sub(\texttt{i32}).\mathbf{C} \to \mathbf{A} : \{sub(\texttt{i32}).\mathsf{t}\} \end{array} \right\} \right\}$$

Ring Protocol

Example

$$\mathsf{G} = \mu\mathsf{t}.\mathbf{A} \to \mathbf{B} : \left\{ add(\mathtt{i32}).\mathbf{B} \to \mathbf{C} : \left\{ \begin{array}{l} add(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{add(\mathtt{i32}).\mathsf{t}\} \\ sub(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{sub(\mathtt{i32}).\mathsf{t}\} \end{array} \right\} \right\}$$

Ring Protocol

Example

$$\mathsf{G} = \mu\mathsf{t}.\mathbf{A} \to \mathbf{B} : \left\{ add(\mathtt{i32}).\mathbf{B} \to \mathbf{C} : \left\{ \begin{array}{l} add(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{add(\mathtt{i32}).\mathsf{t}\} \\ sub(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{sub(\mathtt{i32}).\mathsf{t}\} \end{array} \right\} \right\}$$

Ring Protocol

Example

$$\mathsf{G} = \mu \mathrm{t}.\mathbf{A} \to \mathbf{B} : \left\{ add(\mathtt{i32}).\mathbf{B} \to \mathbf{C} : \left\{ \begin{array}{l} add(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{add(\mathtt{i32}).\mathrm{t}\} \\ sub(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{sub(\mathtt{i32}).\mathrm{t}\} \end{array} \right\} \right\}$$

Ring Protocol

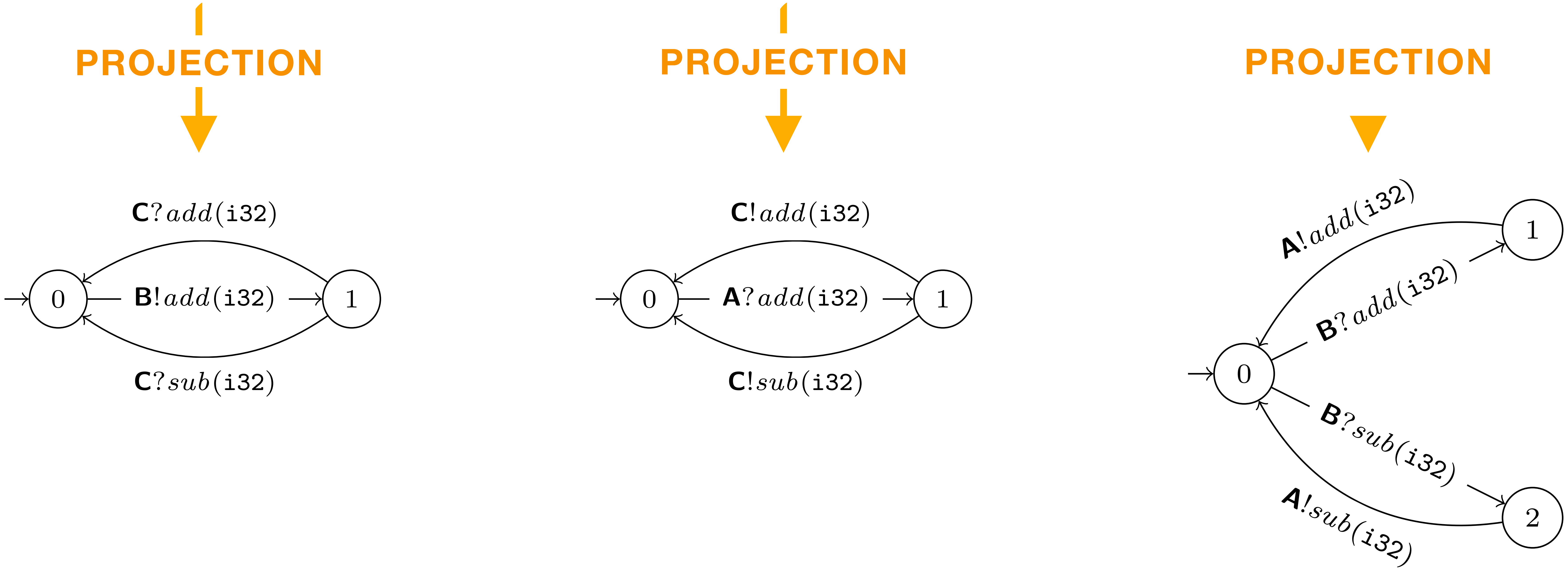
Example

$$\mathsf{G} = \mu \mathsf{t}.\mathbf{A} \rightarrow \mathbf{B} : \left\{ add(\mathtt{i32}).\mathbf{B} \rightarrow \mathbf{C} : \left\{ \begin{array}{c} add(\mathtt{i32}).\mathbf{C} \rightarrow \mathbf{A} : \{add(\mathtt{i32}).\mathsf{t}\} \\ sub(\mathtt{i32}).\mathbf{C} \rightarrow \mathbf{A} : \{sub(\mathtt{i32}).\mathsf{t}\} \end{array} \right\} \right\}$$

Ring Protocol

Example

$$\mathsf{G} = \mu \mathsf{t}.\mathbf{A} \to \mathbf{B} : \left\{ add(\mathtt{i32}).\mathbf{B} \to \mathbf{C} : \left\{ \begin{array}{l} add(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{add(\mathtt{i32}).\mathsf{t}\} \\ sub(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{sub(\mathtt{i32}).\mathsf{t}\} \end{array} \right\} \right\}$$



PROJECTION

PROJECTION

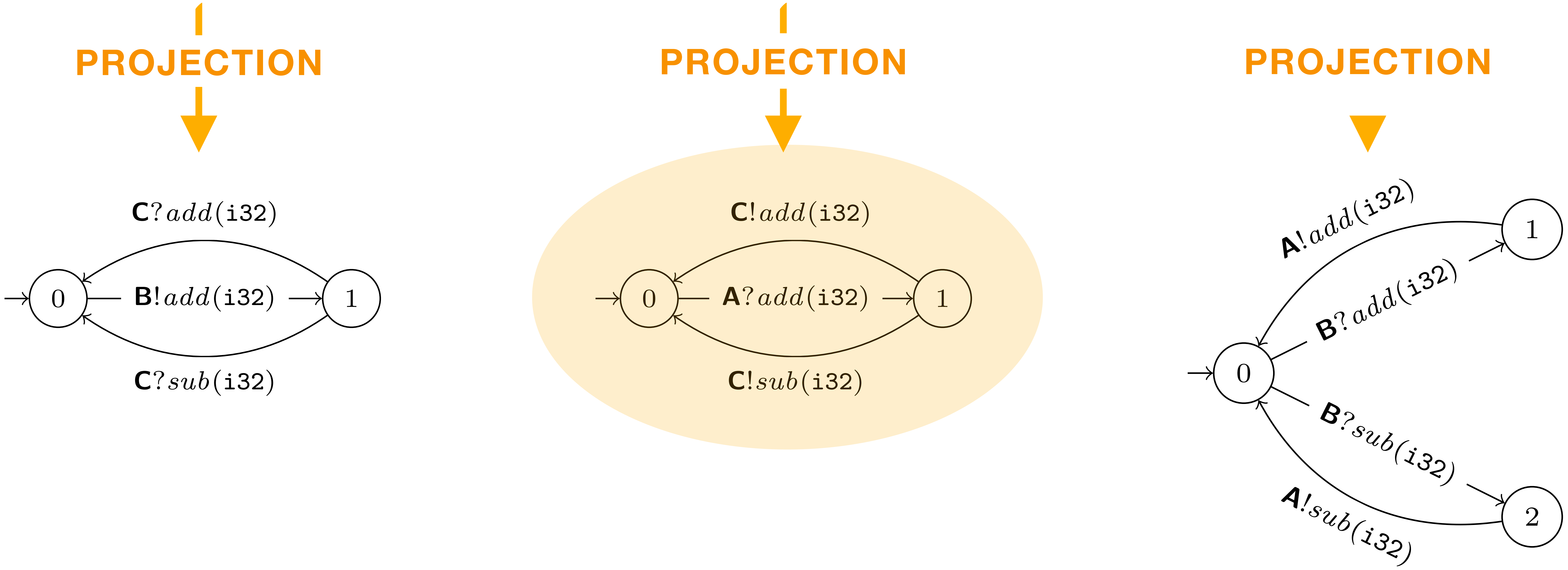
PROJECTION

0
1
$\mathbf{C}?add(\mathtt{i32})$
$\mathbf{B}!add(\mathtt{i32})$
$\mathbf{C}?sub(\mathtt{i32})$
0
1
$\mathbf{C}!add(\mathtt{i32})$
$\mathbf{A}?add(\mathtt{i32})$
$\mathbf{C}!sub(\mathtt{i32})$
0
1
2
$\mathbf{A}!add(\mathtt{i32})$
$\mathbf{B}?add(\mathtt{i32})$
$\mathbf{B}?sub(\mathtt{i32})$
$\mathbf{A}!sub(\mathtt{i32})$

Ring Protocol

Example

$$\mathsf{G} = \mu \mathsf{t}.\mathbf{A} \to \mathbf{B} : \left\{ add(\mathtt{i32}).\mathbf{B} \to \mathbf{C} : \left\{ \begin{array}{l} add(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{add(\mathtt{i32}).\mathsf{t}\} \\ sub(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{sub(\mathtt{i32}).\mathsf{t}\} \end{array} \right\} \right\}$$



Challenge

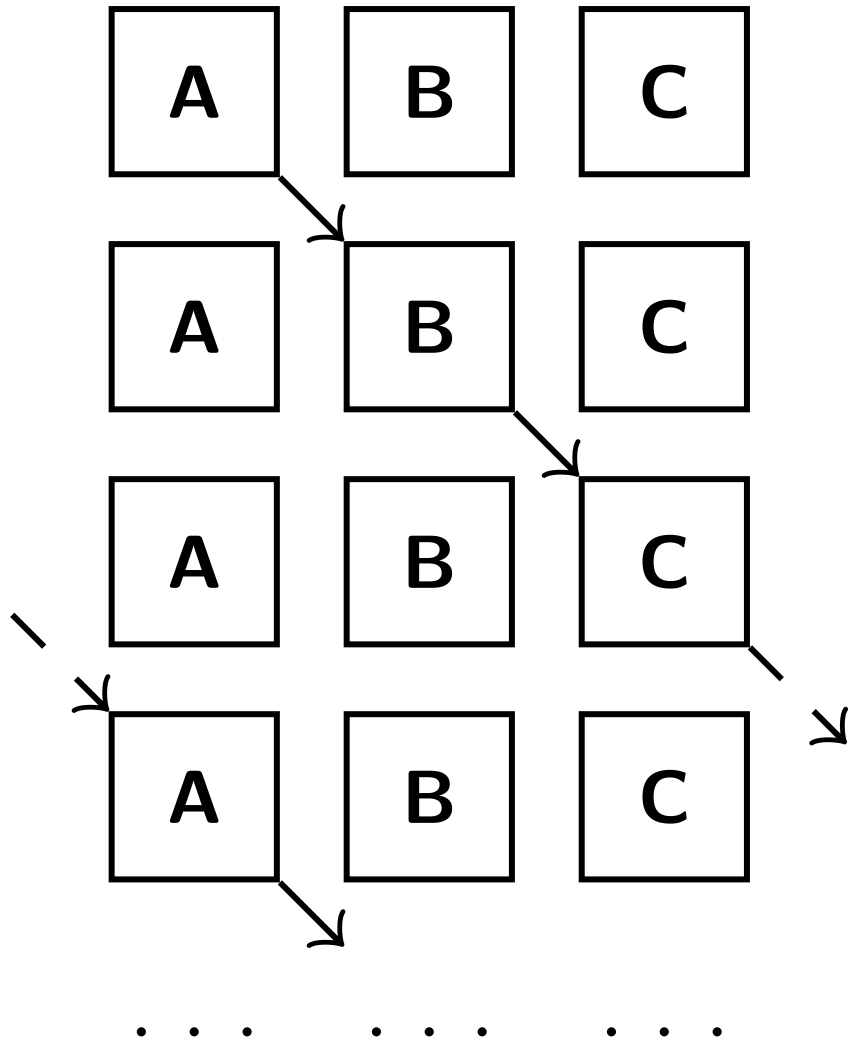
Asynchronous Orderings

- Global types are inherently synchronous

Challenge

Asynchronous Orderings

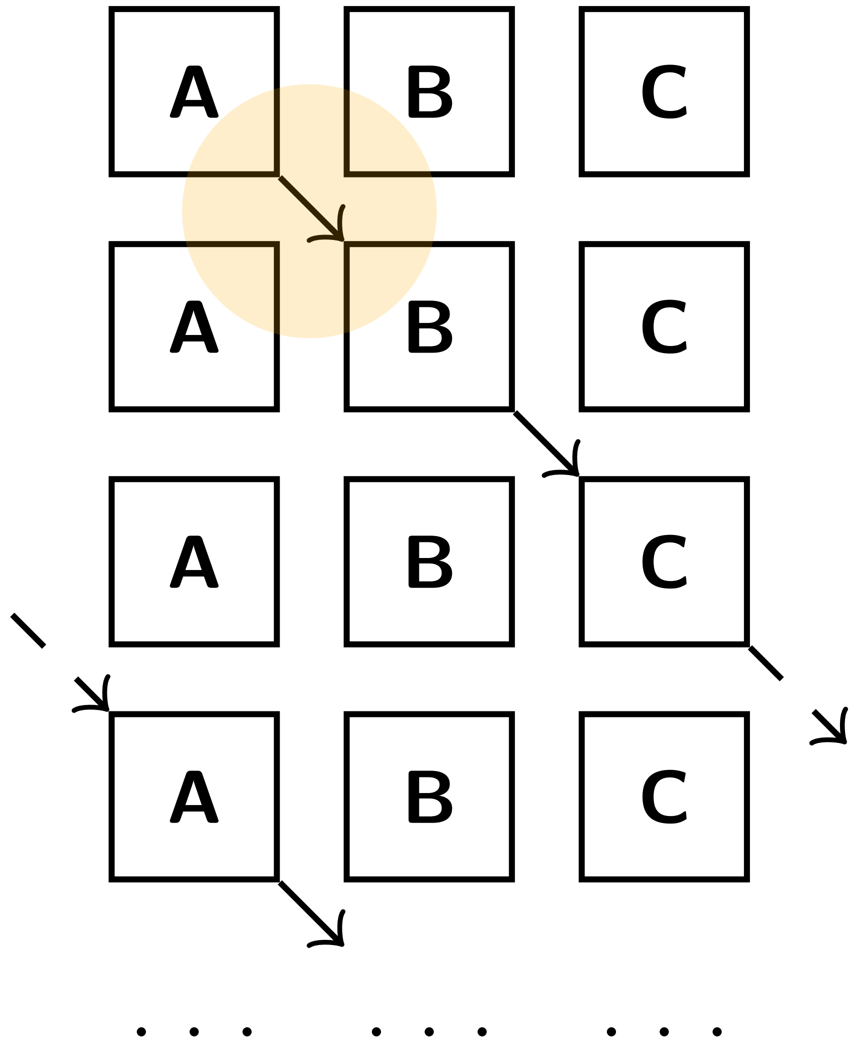
- Global types are inherently synchronous
  - Projection provides only one possible ordering



Challenge

Asynchronous Orderings

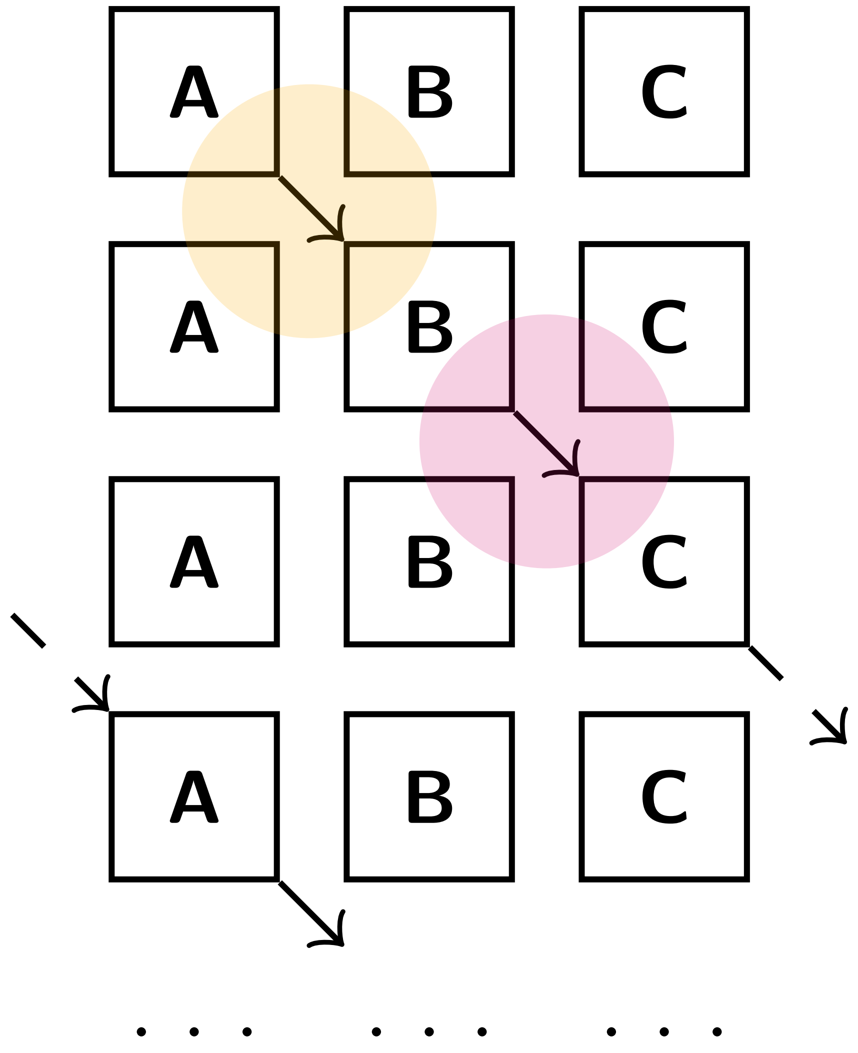
- Global types are inherently synchronous
  - Projection provides only one possible ordering



Challenge

Asynchronous Orderings

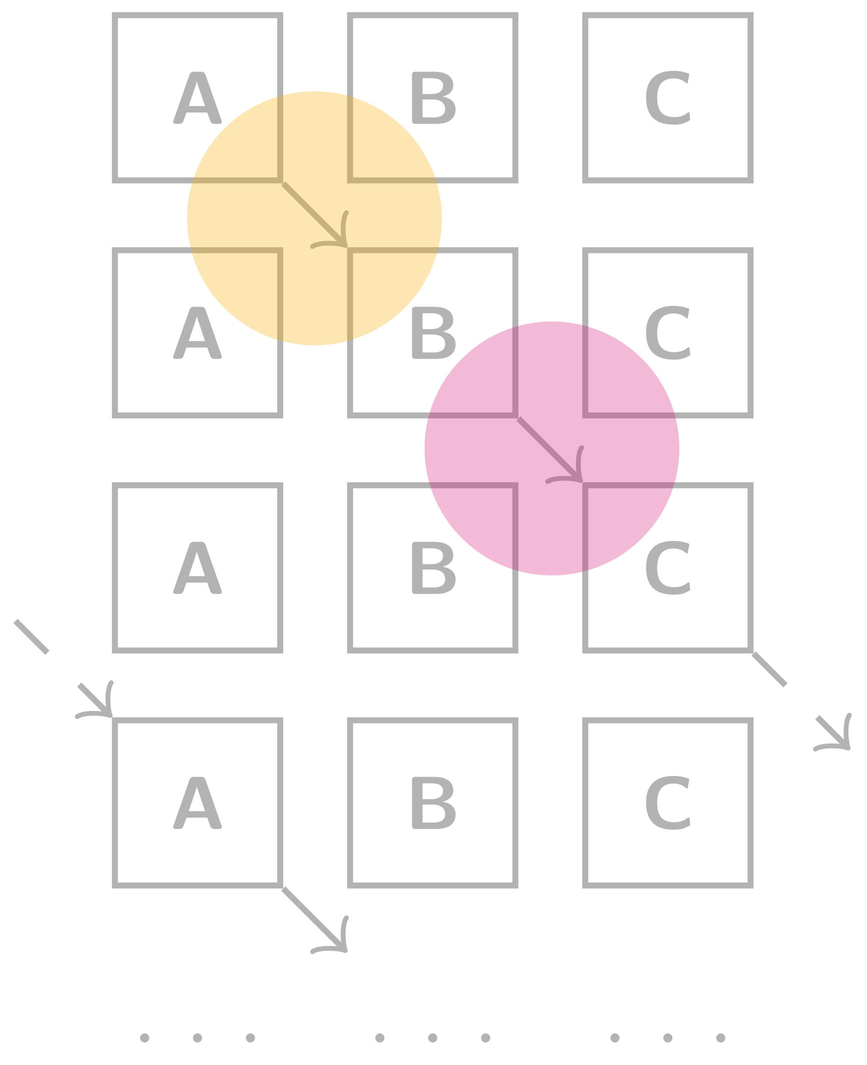
- Global types are inherently synchronous
  - Projection provides only one possible ordering



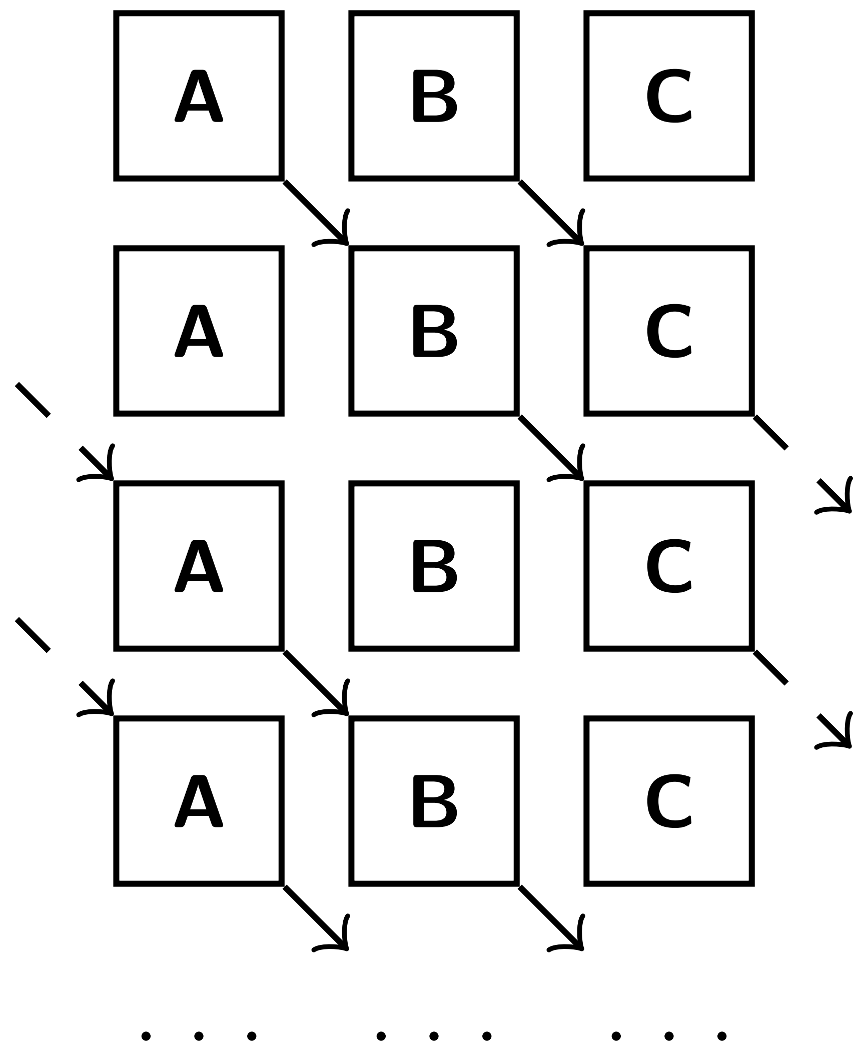
Challenge

Asynchronous Orderings

- Global types are inherently synchronous
  - Projection provides only one possible ordering



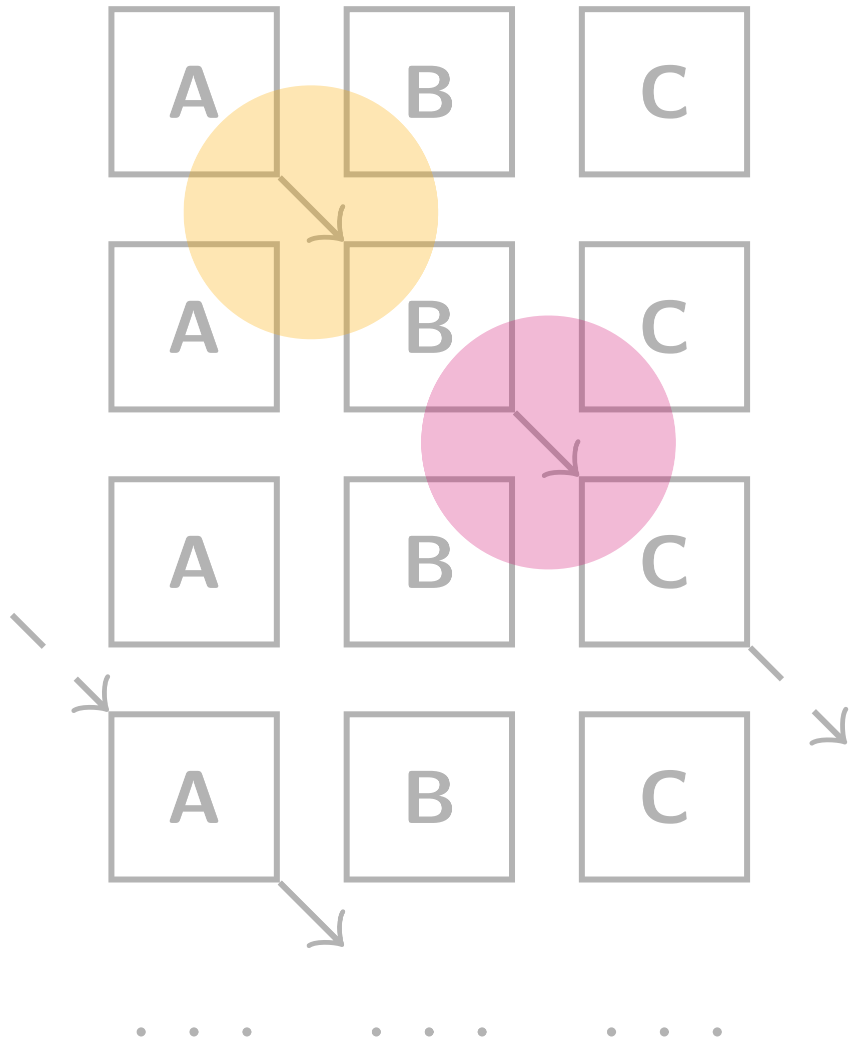
- Interactions can be reordered for efficiency while preserving safety

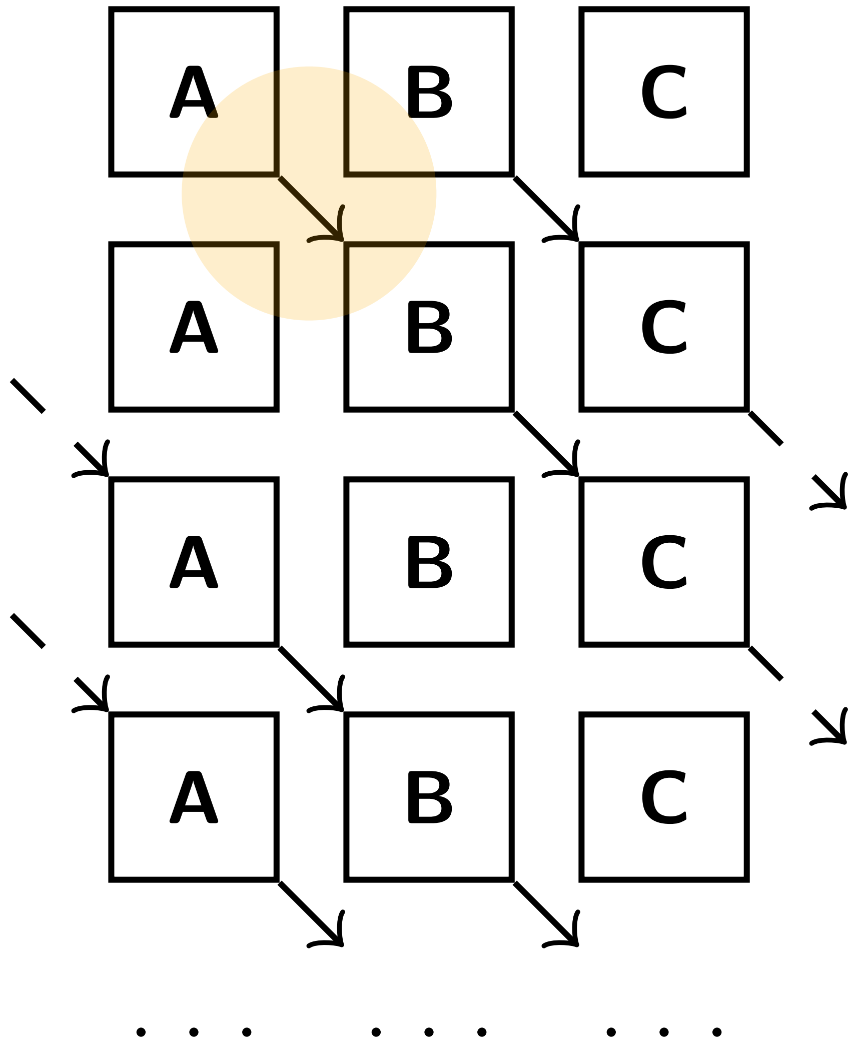


Challenge

Asynchronous Orderings

- Global types are inherently synchronous
  - Projection provides only one possible ordering
- Interactions can be reordered for efficiency while preserving safety

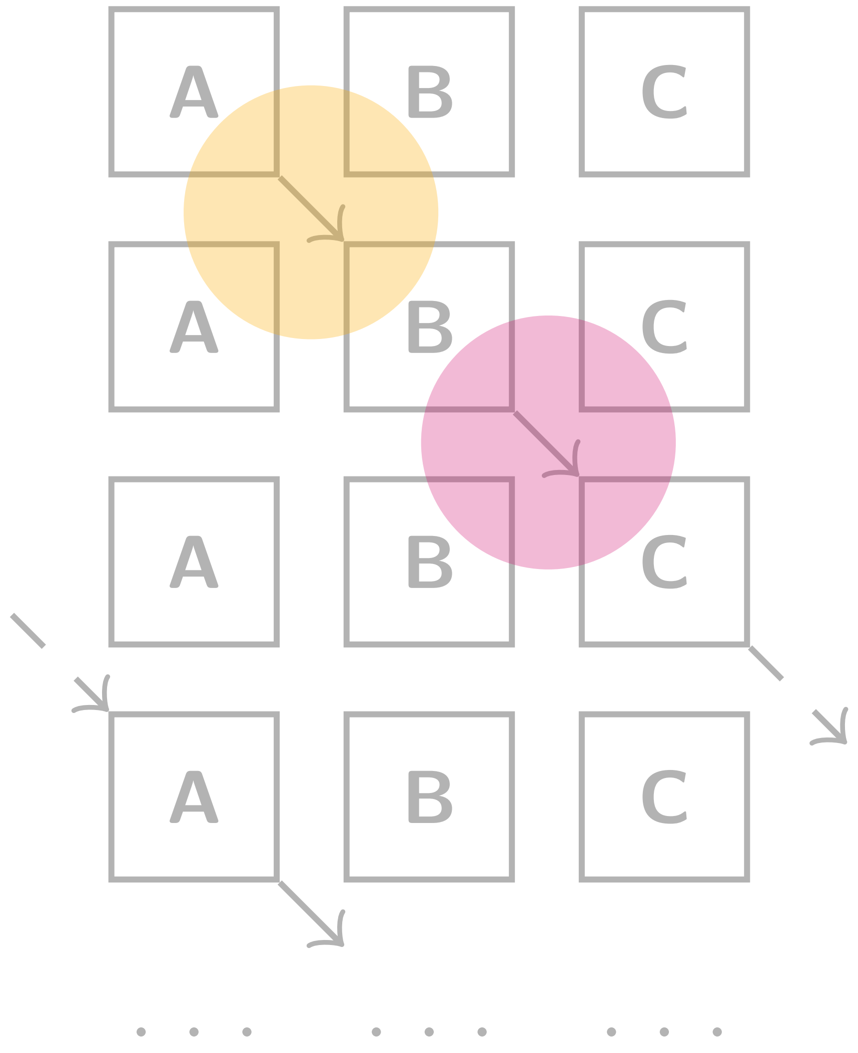


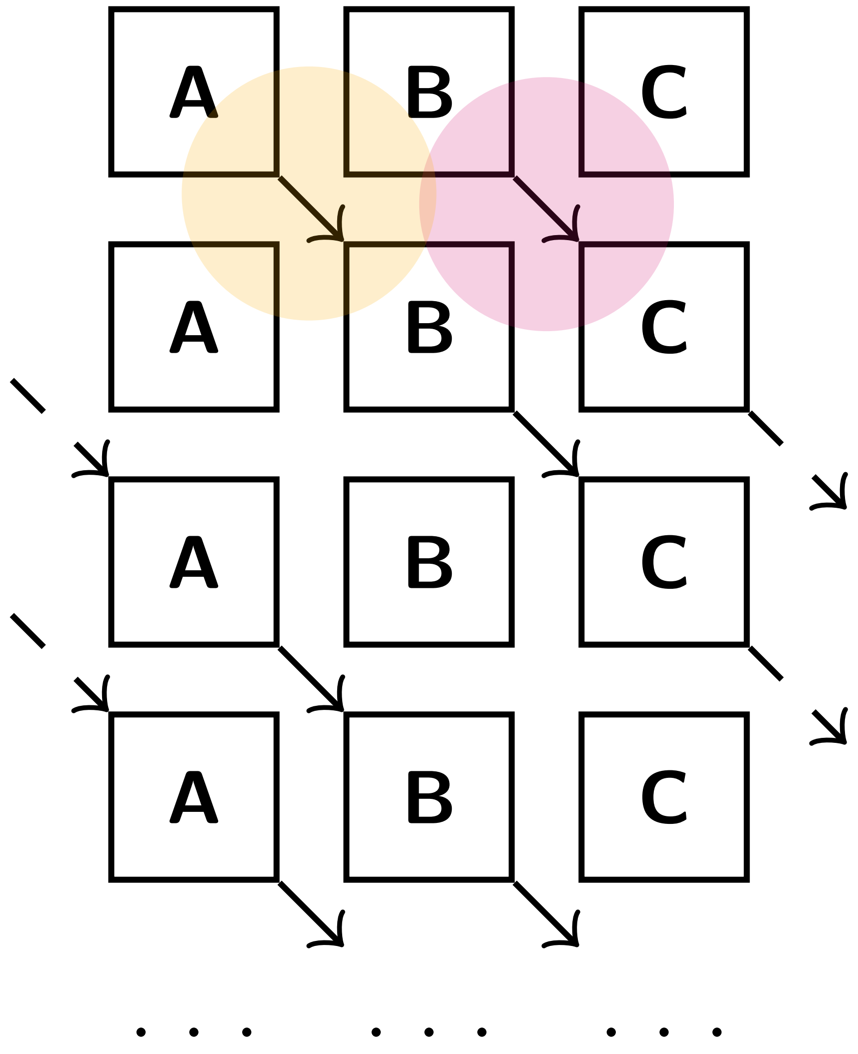


Challenge

Asynchronous Orderings

- Global types are inherently synchronous
  - Projection provides only one possible ordering
- Interactions can be reordered for efficiency while preserving safety



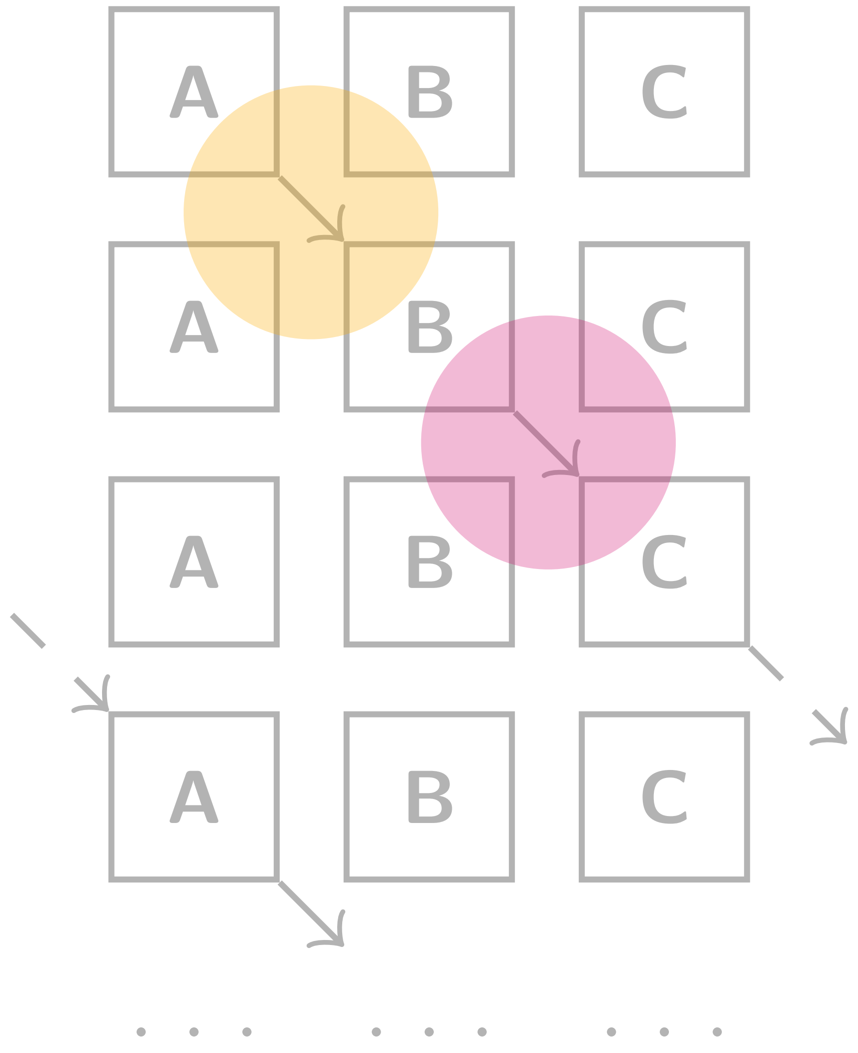


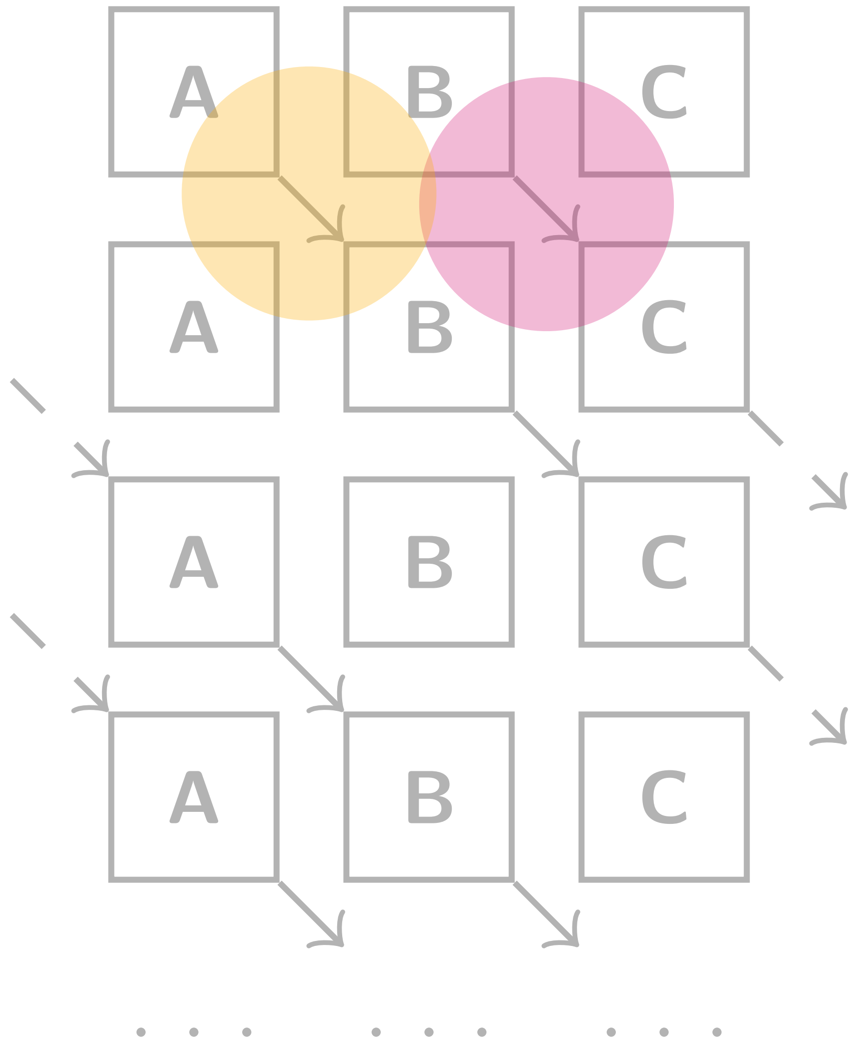
Challenge

Asynchronous Orderings

- Global types are inherently synchronous
  - Projection provides only one possible ordering
- Interactions can be reordered for efficiency while preserving safety

1. Data dependencies must be preserved



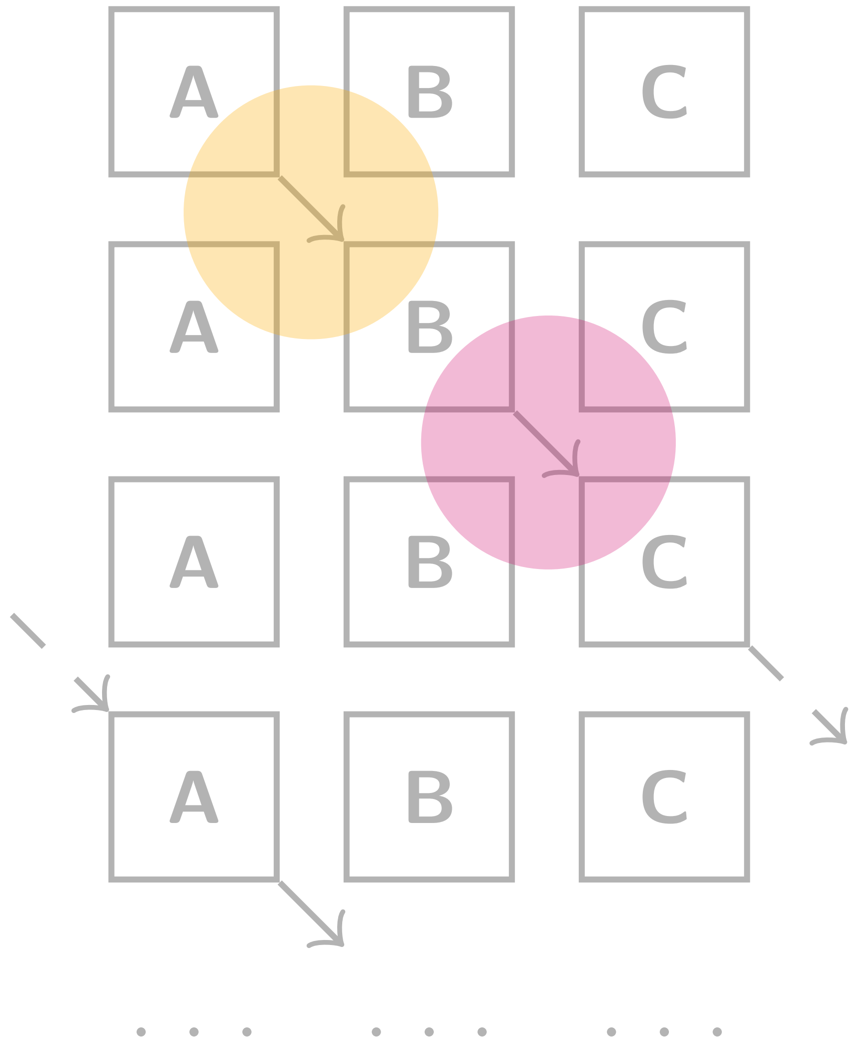


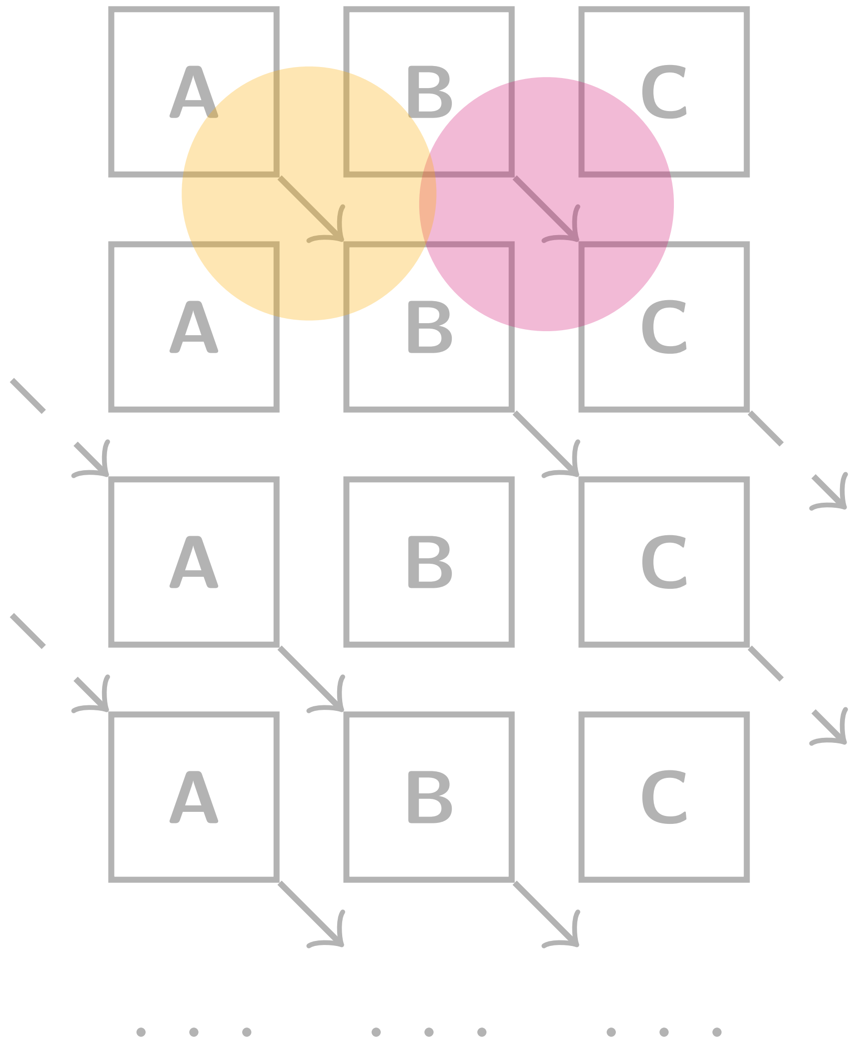
Challenge

Asynchronous Orderings

- Global types are inherently synchronous
  - Projection provides only one possible ordering
- Interactions can be reordered for efficiency while preserving safety

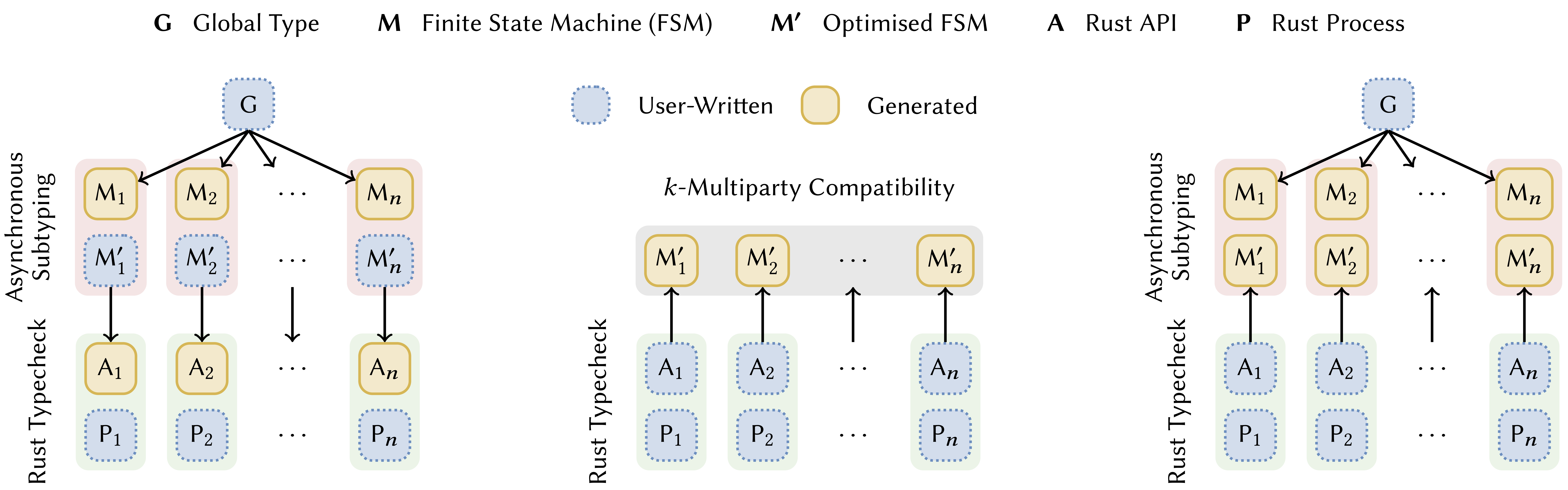
  1. Data dependencies must be preserved
  2. Sound and practical asynchronous reordering rules must be found





Rumpsteak Framework

Three Approaches



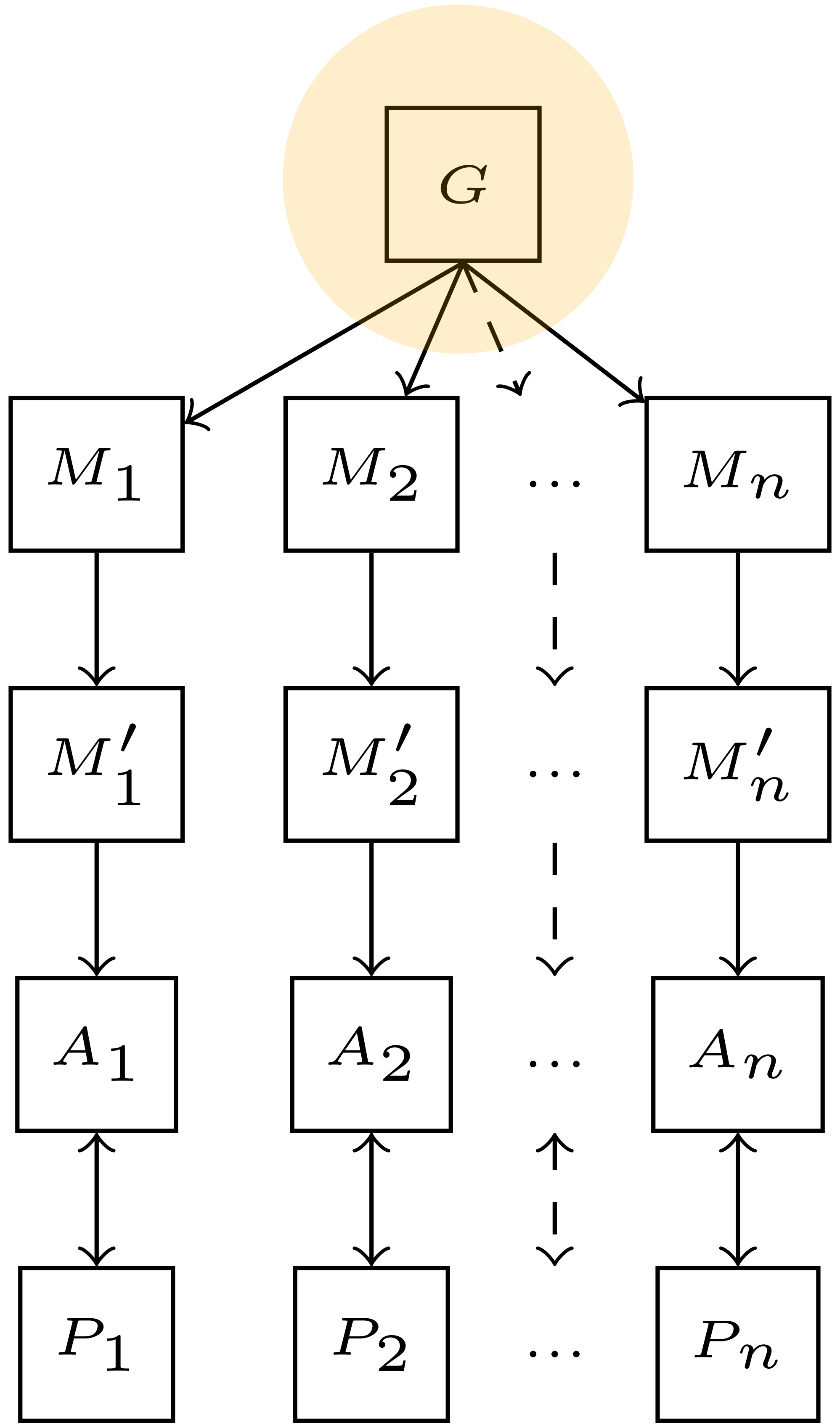
(a) Top-down

(b) Bottom-up

(c) Hybrid

Workflow

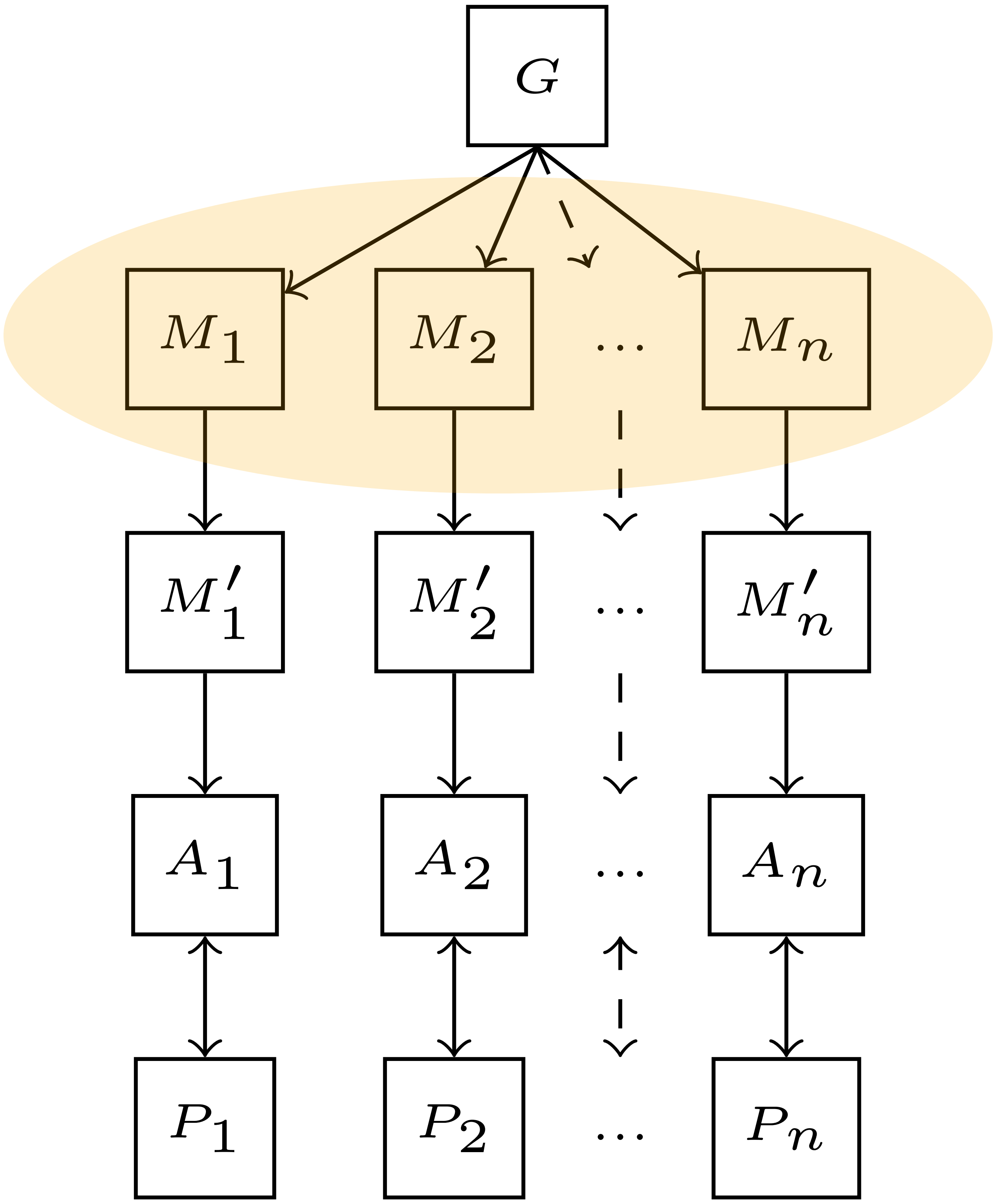
Top-Down Approach

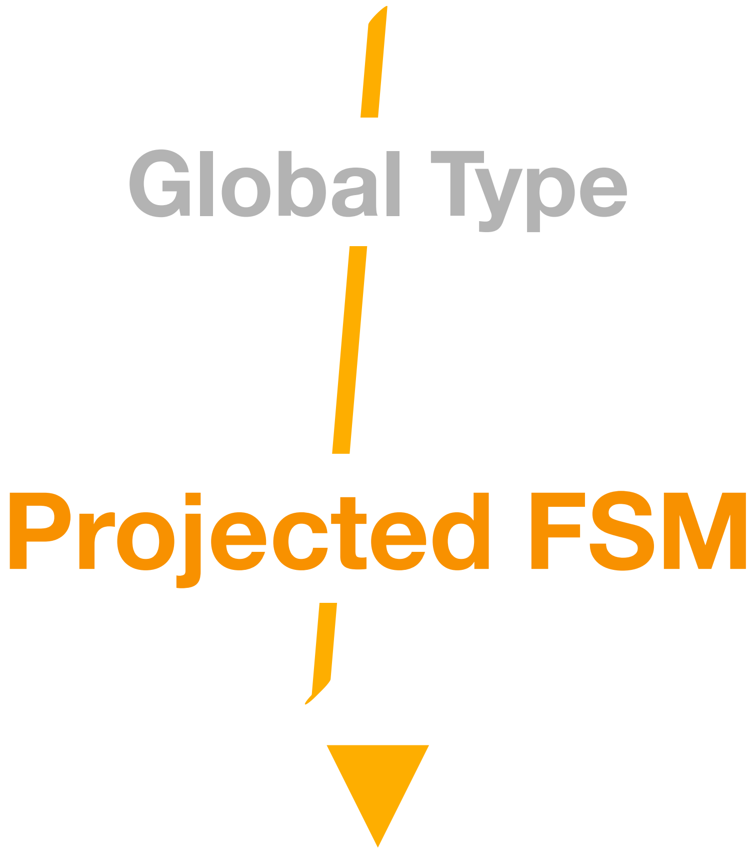


Global Type

Workflow

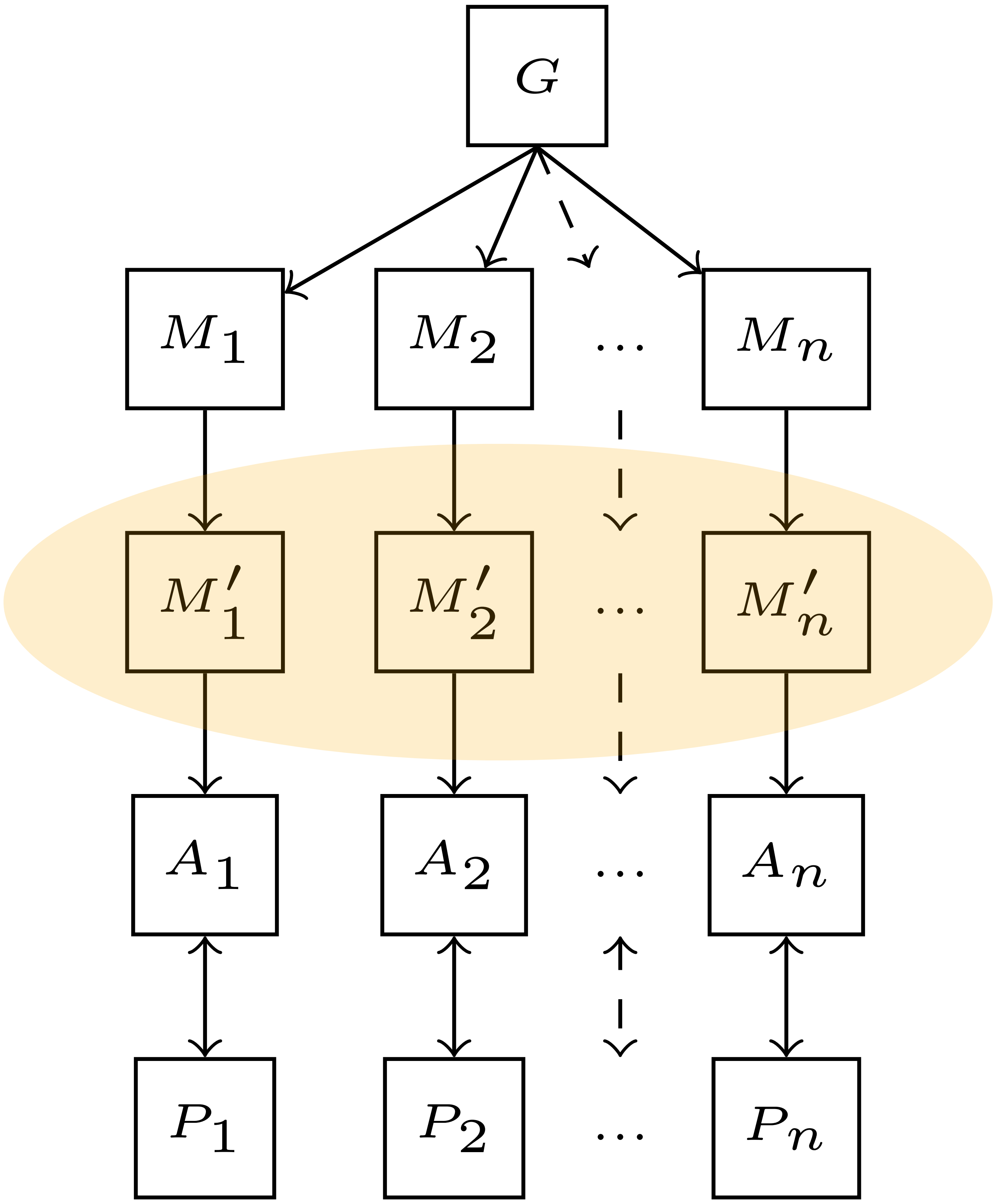
Top-Down Approach





Workflow

Top-Down Approach



Global Type

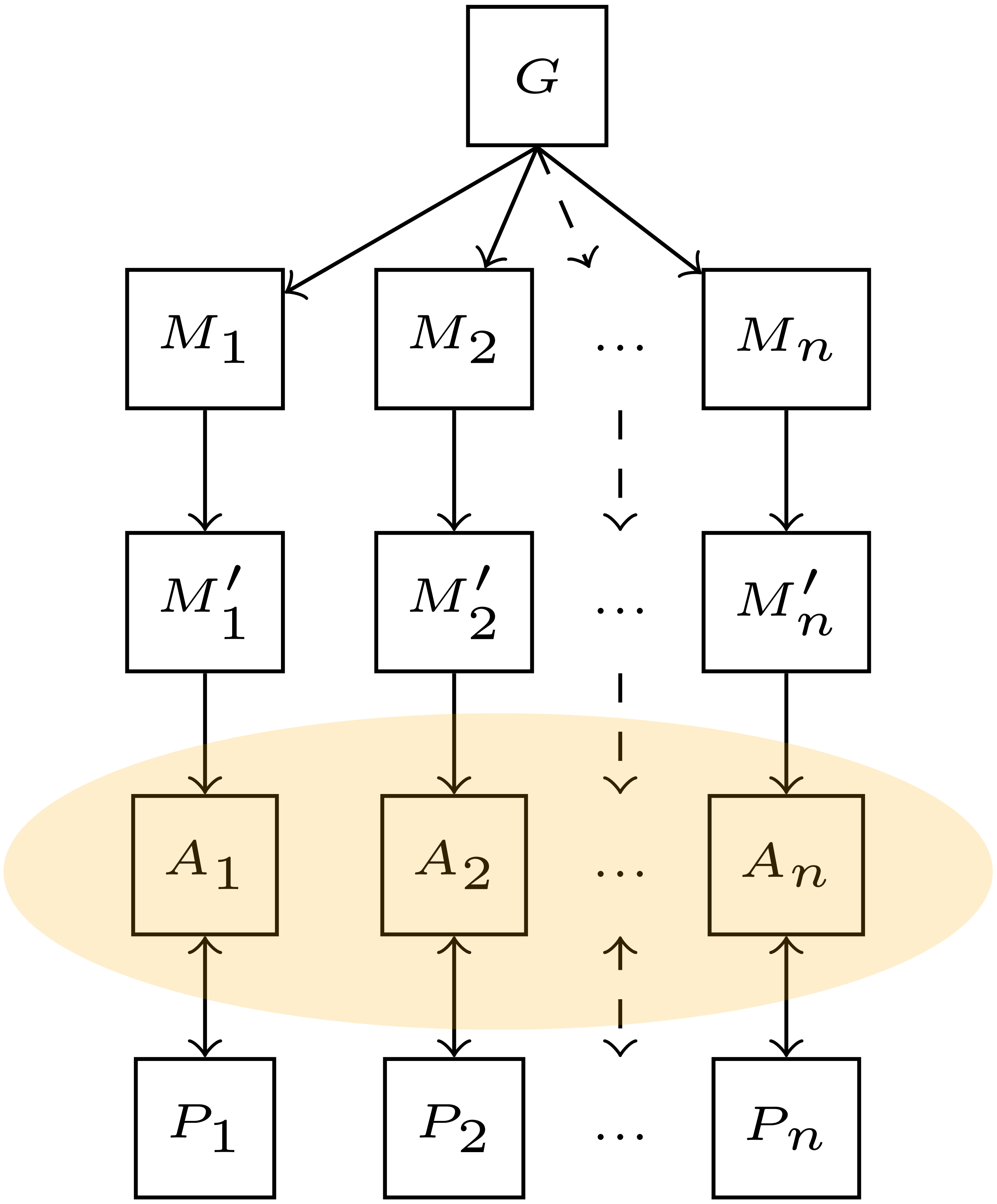
Projected FSM

Optimised FSM

▼

Workflow

Top-Down Approach



Global Type

Projected FSM

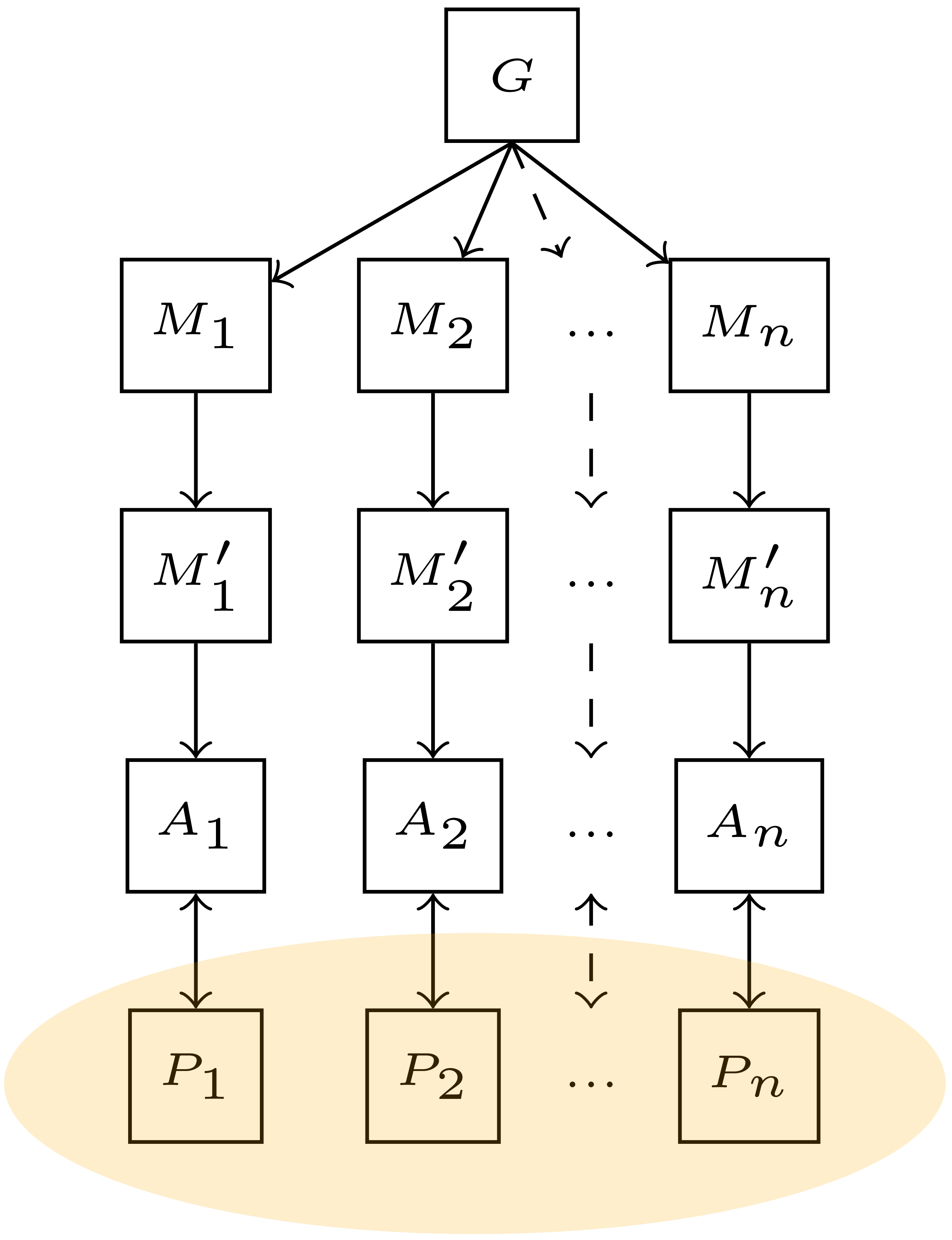
Optimised FSM

Rust API

▼

Workflow

Top-Down Approach



Global Type

Projected FSM

Optimised FSM

Rust API

Rust Implementation

Ring Protocol

Example

Global Type

$$\mathsf{G} = \mu \mathsf{t}.\mathbf{A} \to \mathbf{B} : \left\{ add(\mathtt{i32}).\mathbf{B} \to \mathbf{C} : \left\{ \begin{array}{l} add(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{add(\mathtt{i32}).\mathsf{t}\} \\ sub(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{sub(\mathtt{i32}).\mathsf{t}\} \end{array} \right\} \right\}$$

Ring Protocol

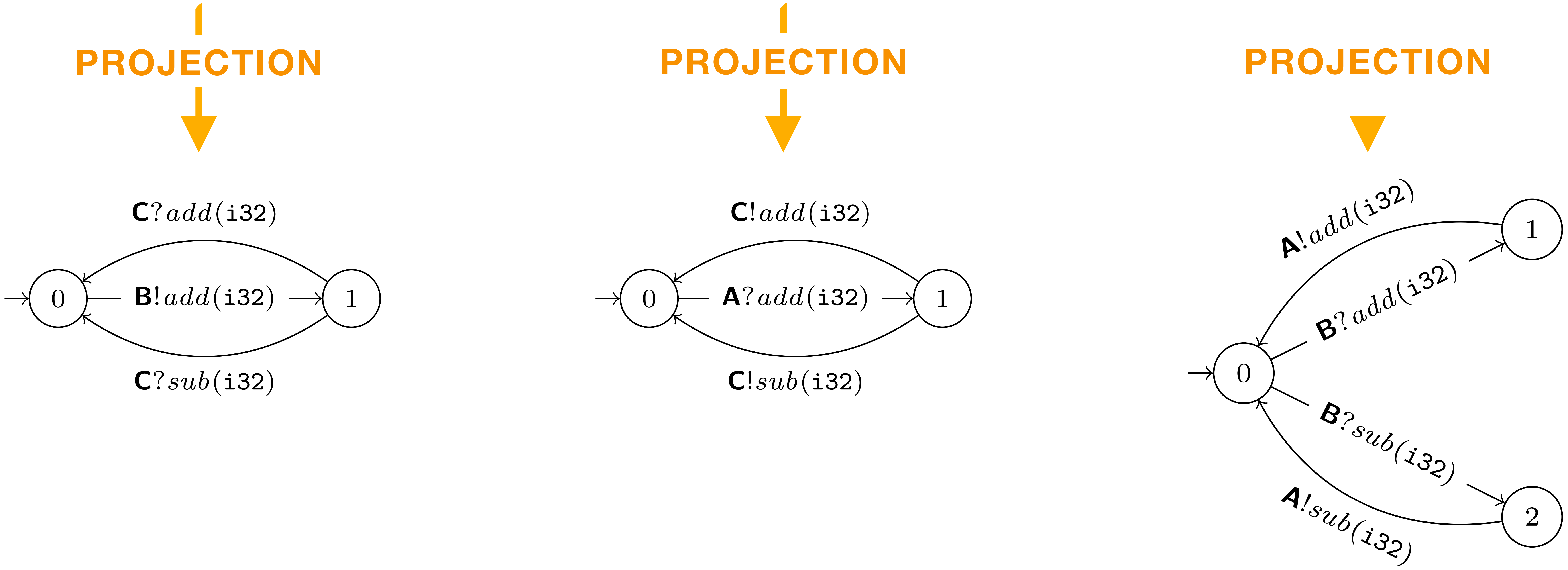
Example

$$\mathsf{G} = \mu \mathsf{t}.\mathbf{A} \to \mathbf{B} : \left\{ add(\mathtt{i32}).\mathbf{B} \to \mathbf{C} : \left\{ \begin{array}{c} add(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{add(\mathtt{i32}).\mathsf{t}\} \\ sub(\mathtt{i32}).\mathbf{C} \to \mathbf{A} : \{sub(\mathtt{i32}).\mathsf{t}\} \end{array} \right\} \right\}$$

Ring Protocol

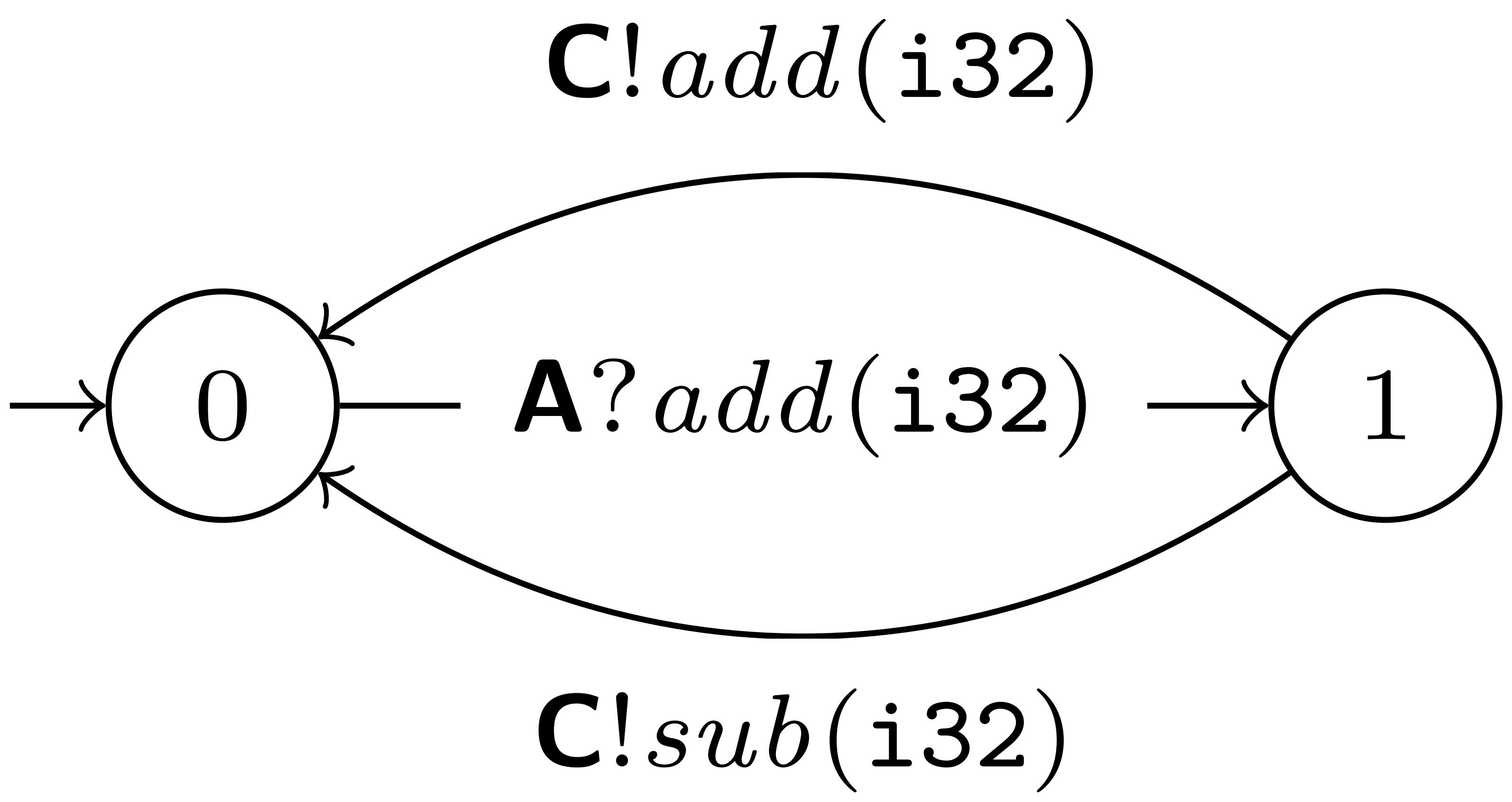
Example

$$\mathsf{G} = \mu \mathsf{t}.\mathbf{A} \to \mathbf{B} : \left\{ add(\texttt{i32}).\mathbf{B} \to \mathbf{C} : \left\{ \begin{array}{l} add(\texttt{i32}).\mathbf{C} \to \mathbf{A} : \{add(\texttt{i32}).\mathsf{t}\} \\ sub(\texttt{i32}).\mathbf{C} \to \mathbf{A} : \{sub(\texttt{i32}).\mathsf{t}\} \end{array} \right\} \right\}$$


PROJECTION
PROJECTION
PROJECTION
$\mathbf{C}?add(\texttt{i32})$
0
$\mathbf{B}!add(\texttt{i32})$
1
$\mathbf{C}?sub(\texttt{i32})$
$\mathbf{C}!add(\texttt{i32})$
0
$\mathbf{A}?add(\texttt{i32})$
1
$\mathbf{C}!sub(\texttt{i32})$
$\mathbf{A}!add(\texttt{i32})$
1
$\mathbf{B}?add(\texttt{i32})$
0
$\mathbf{B}?sub(\texttt{i32})$
2
$\mathbf{A}!sub(\texttt{i32})$

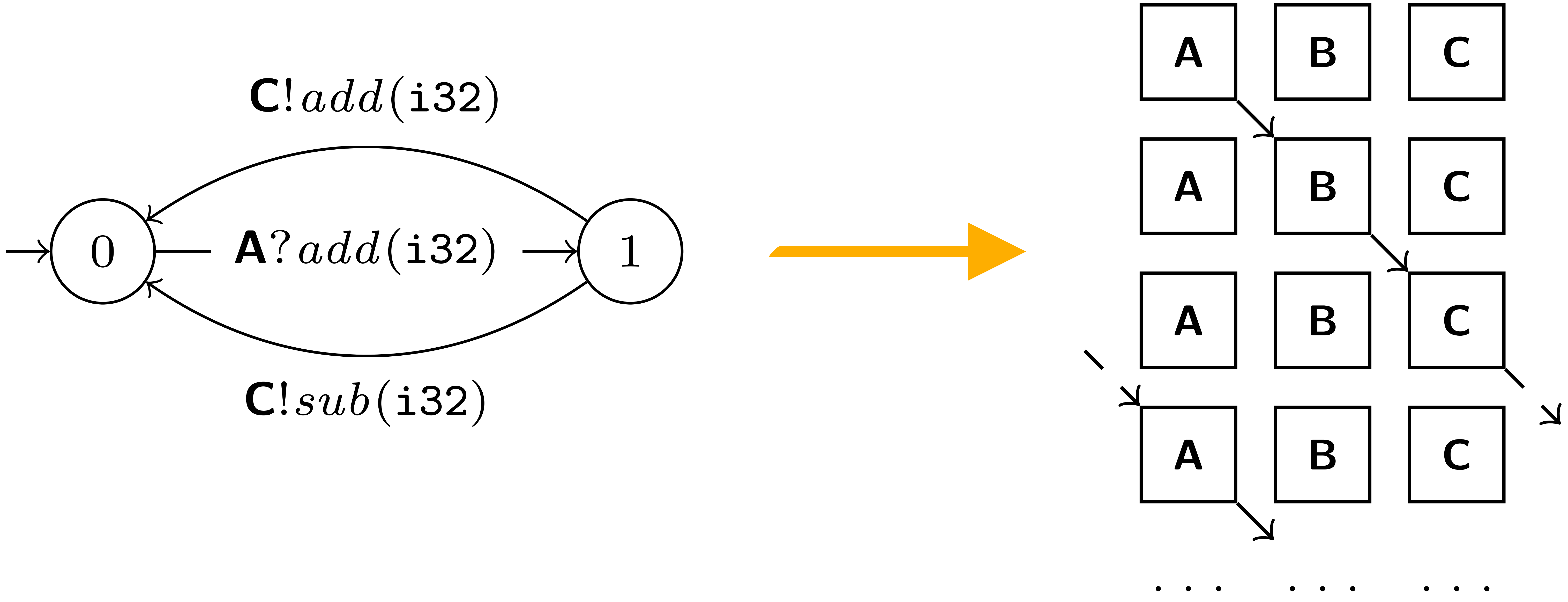
Ring Protocol

Example



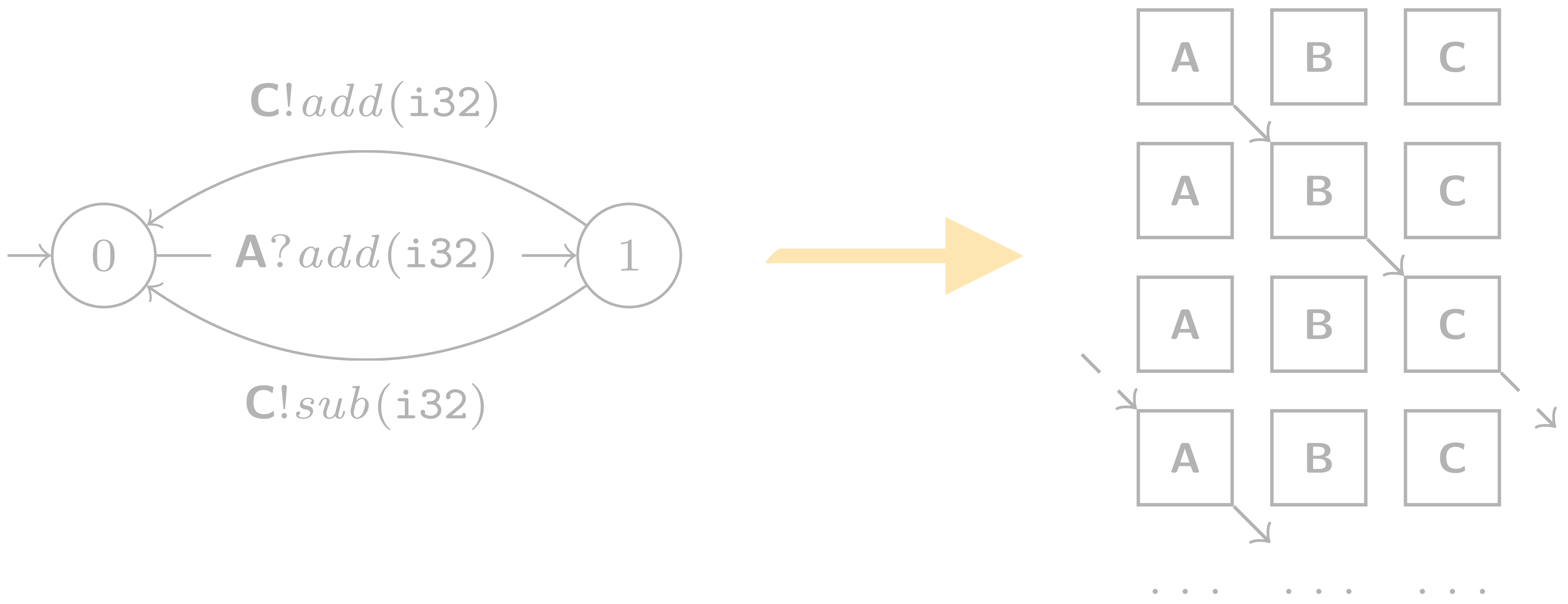
Ring Protocol

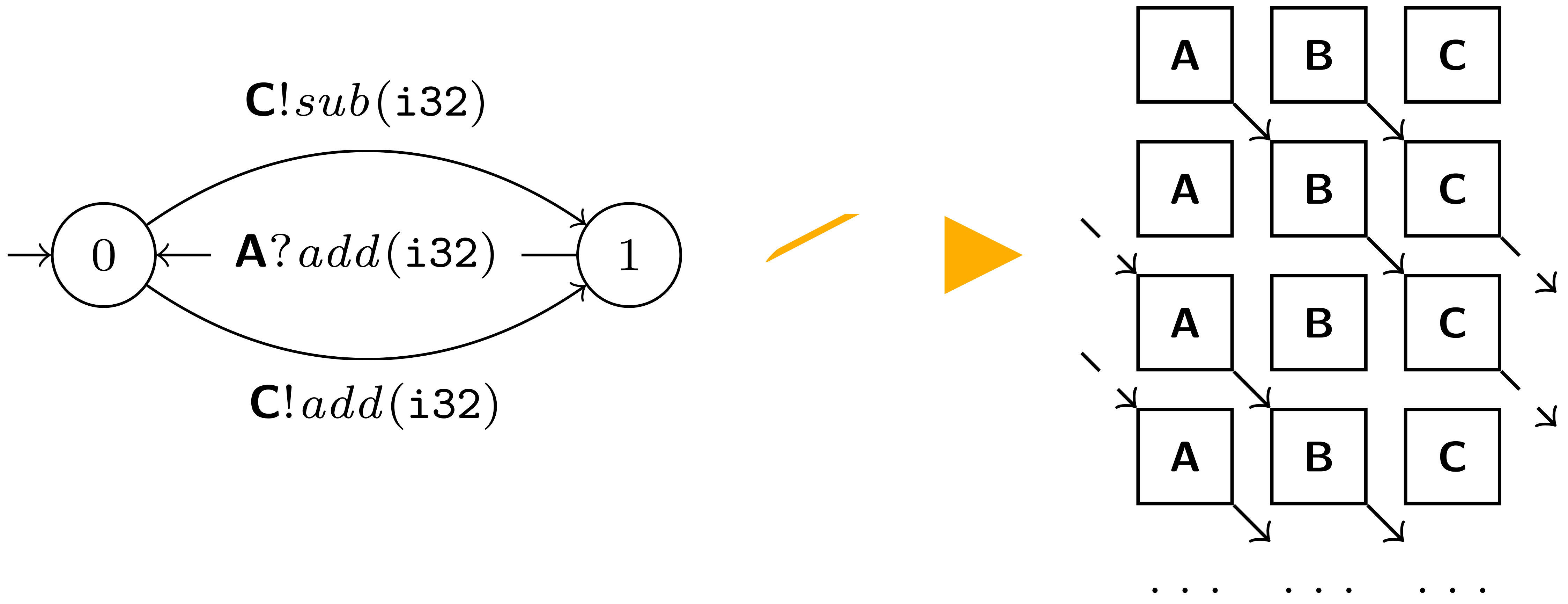
Example



Ring Protocol

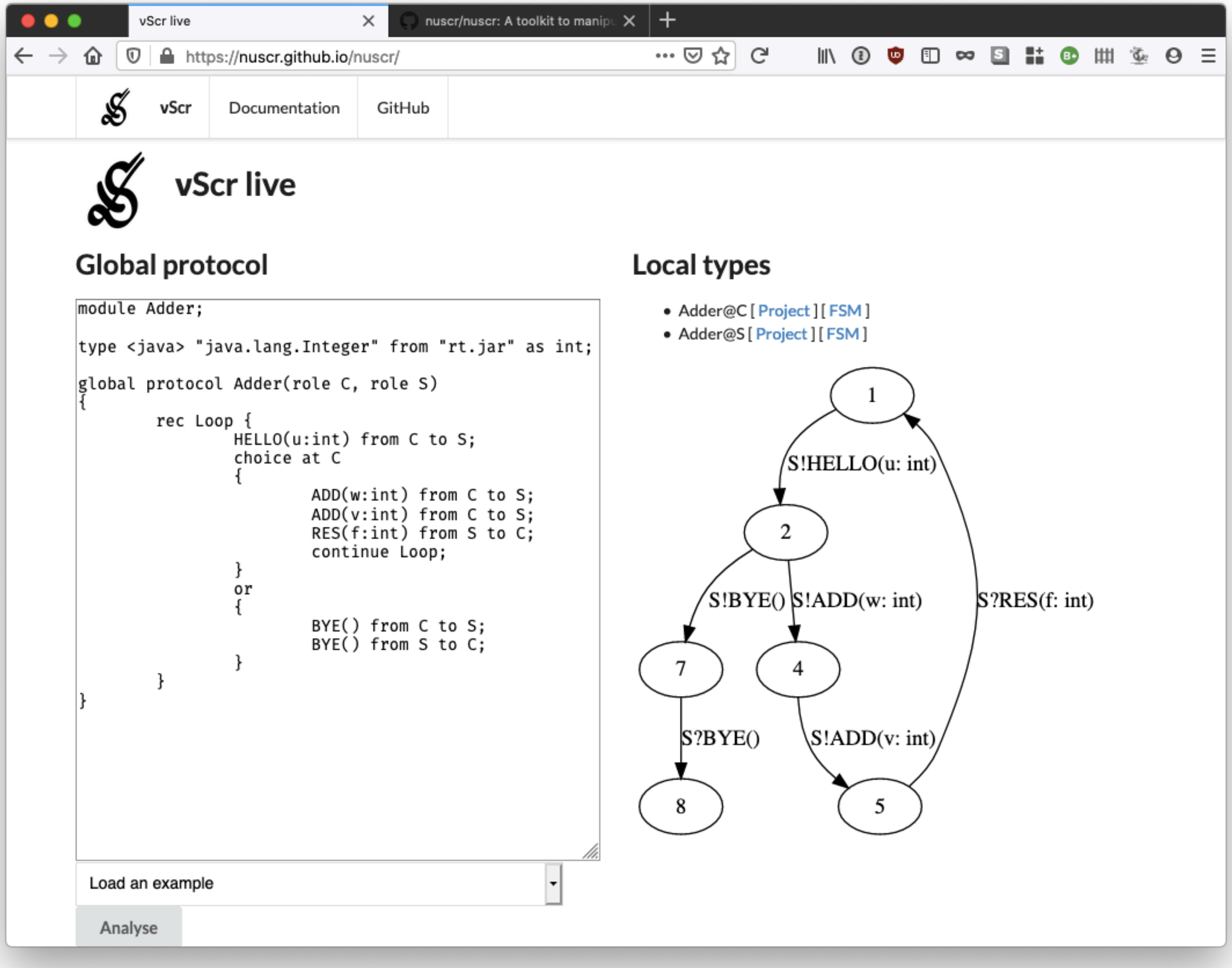
Example





vScr An Extensible Toolchain for Multiparty Session Types

- It’s small and easy to modify
- Available on opam
  - opam install nuscr
- Available on GitHub
  - https://github.com/nuscr
- Available on the web
  - https://nuscr.dev


vScr live
nuscr/nuscr: A toolkit to manip
https://nuscr.github.io/nuscr/
vScr
Documentation
GitHub

vScr live

Global protocol

```
module Adder;

type <java> "java.lang.Integer" from "rt.jar" as int;

global protocol Adder(role C, role S)
{
        rec Loop {
                HELLO(u:int) from C to S;
                choice at C
                {
                        ADD(w:int) from C to S;
                        ADD(v:int) from C to S;
                        RES(f:int) from S to C;
                        continue Loop;
                }
                or
                {
                        BYE() from C to S;
                        BYE() from S to C;
                }
        }
}
```

Local types

- Adder@C [Project] [FSM]
- Adder@S [Project] [FSM]

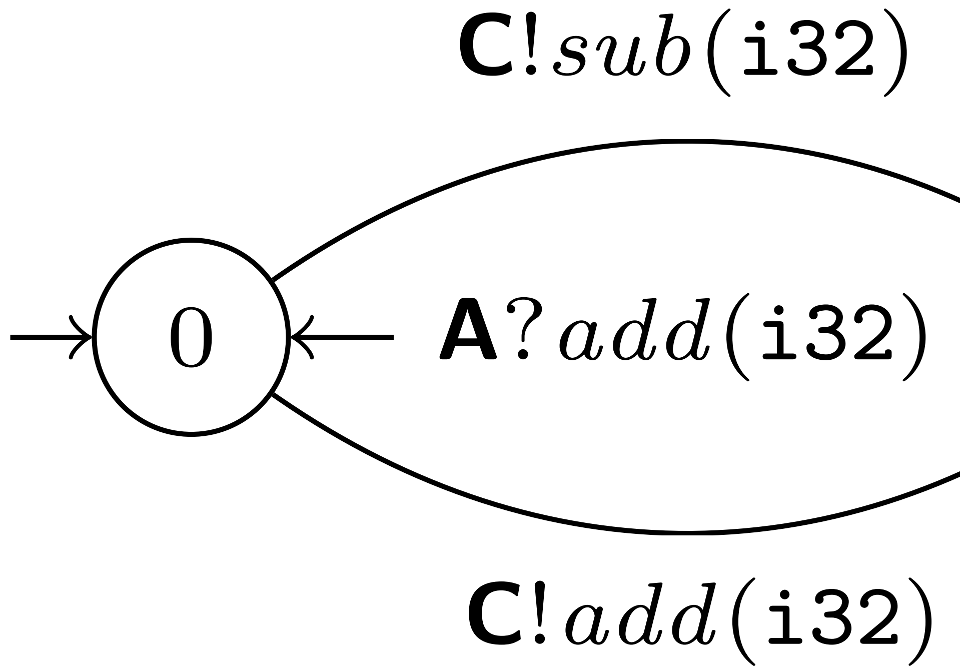
Scribble

Protocol Description Language

```
global protocol Ring(role A, role B, role C) {
    Add(i32) from A to B;
    choice at B {
        Add(i32) from B to C;
        Add(i32) from C to A;
        do Ring(A, B, C);
    } or {
        Sub(i32) from B to C;
        Sub(i32) from C to A;
        do Ring(A, B, C);
    }
}
```

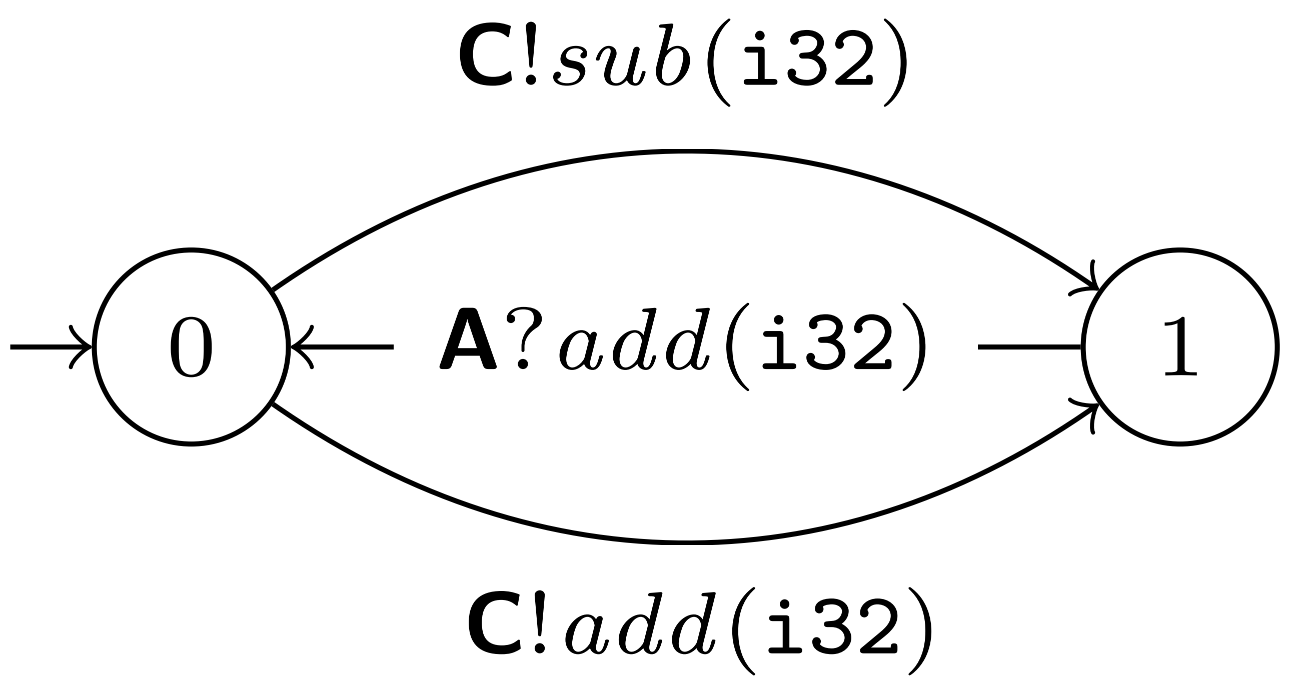
Ring Protocol

Rust API



Ring Protocol

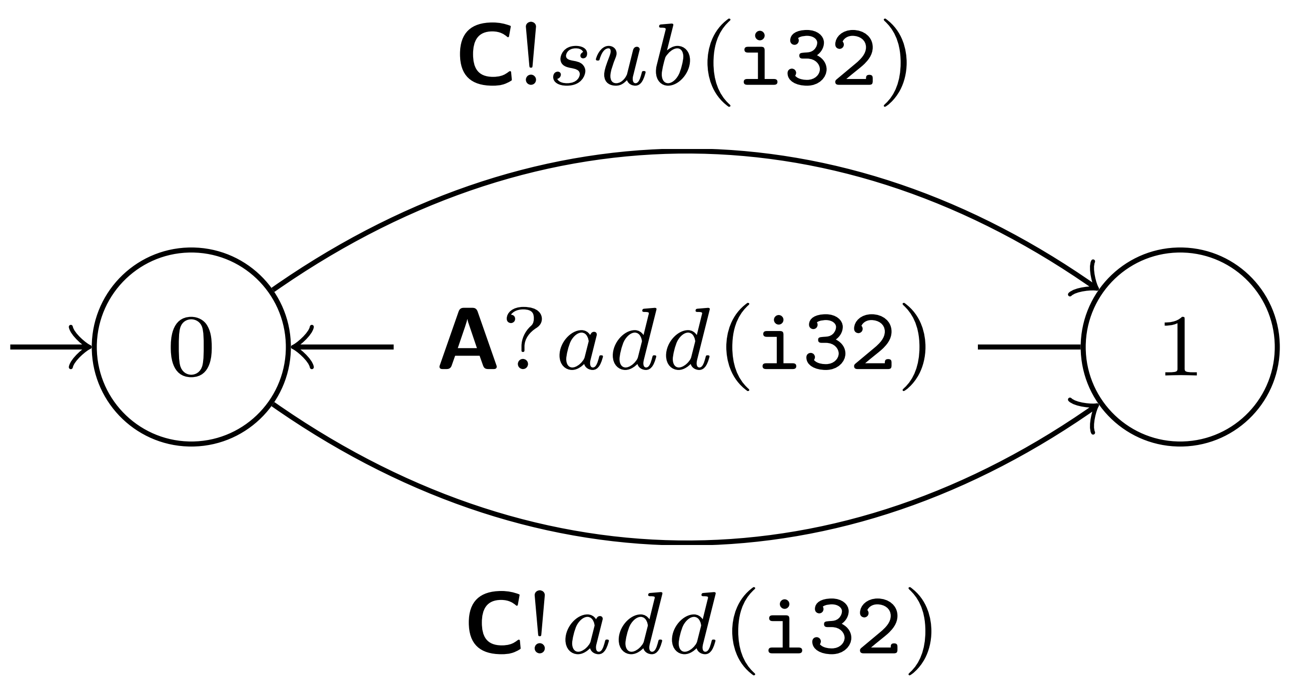
Rust API



```
#[derive(Role)]
#[message(Label)]
struct B(#[route(A)] Receiver, #[route(C)] Sender);
```

Ring Protocol

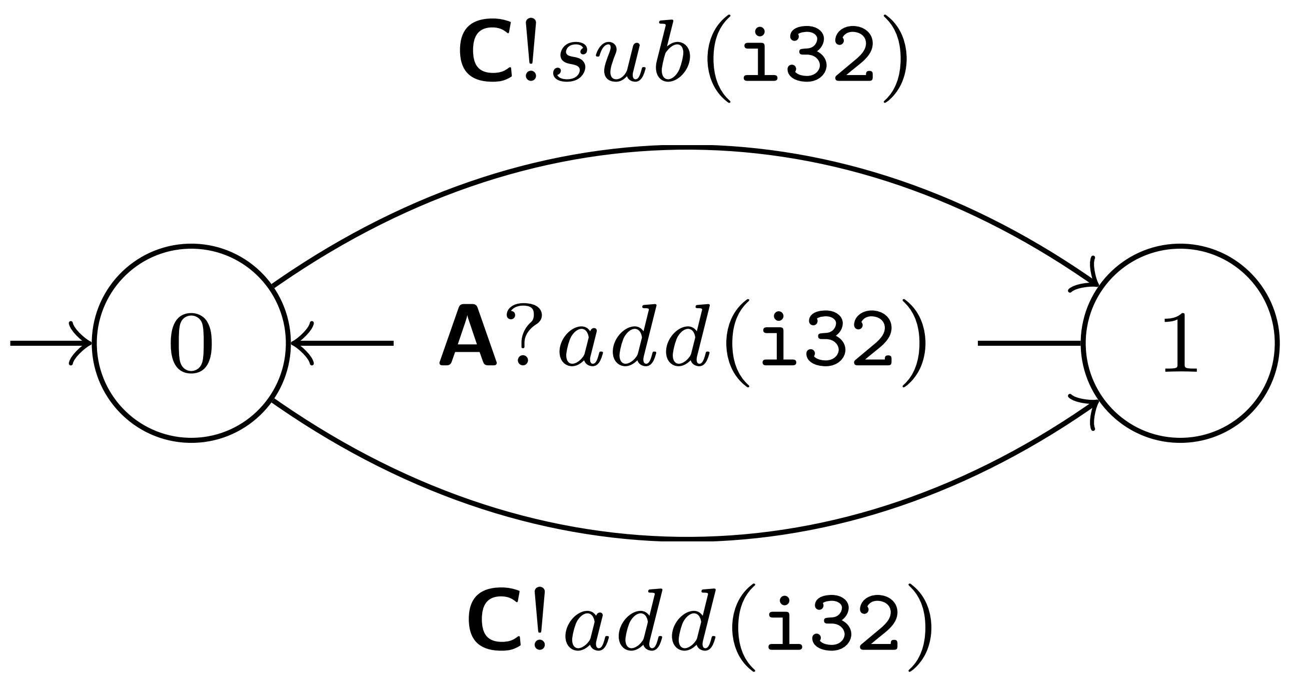
Rust API



```
#[derive(Role)]
#[message(Label)]
struct B(#[route(A)] Receiver, #[route(C)] Sender);
```

Ring Protocol

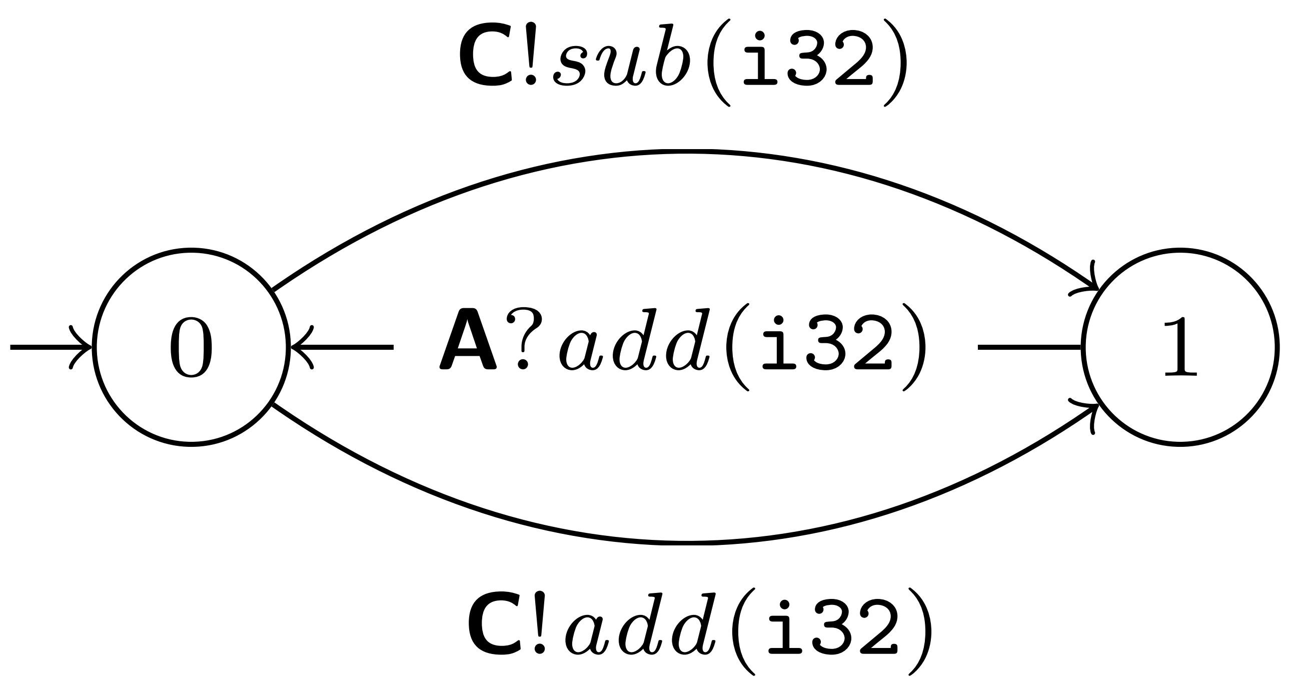
Rust API



```
#[derive(Role)]
#[message(Label)]
struct B(#[route(A)] Receiver, #[route(C)] Sender);
```

Ring Protocol

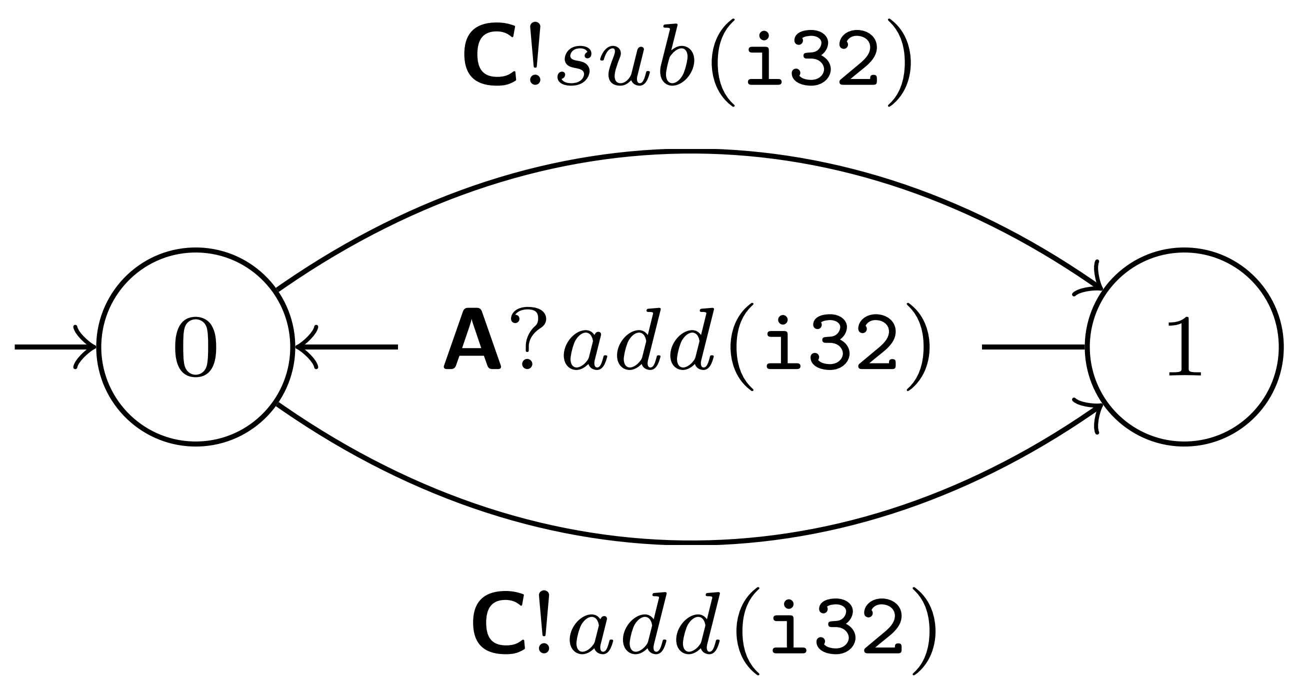
Rust API



```
#[derive(Role)]
#[message(Label)]
struct B(#[route(A)] Receiver, #[route(C)] Sender);

#[derive(Message)]
enum Label {
    Add(Add),
    Sub(Sub),
}

struct Add(i32);
struct Sub(i32);
```

Ring Protocol

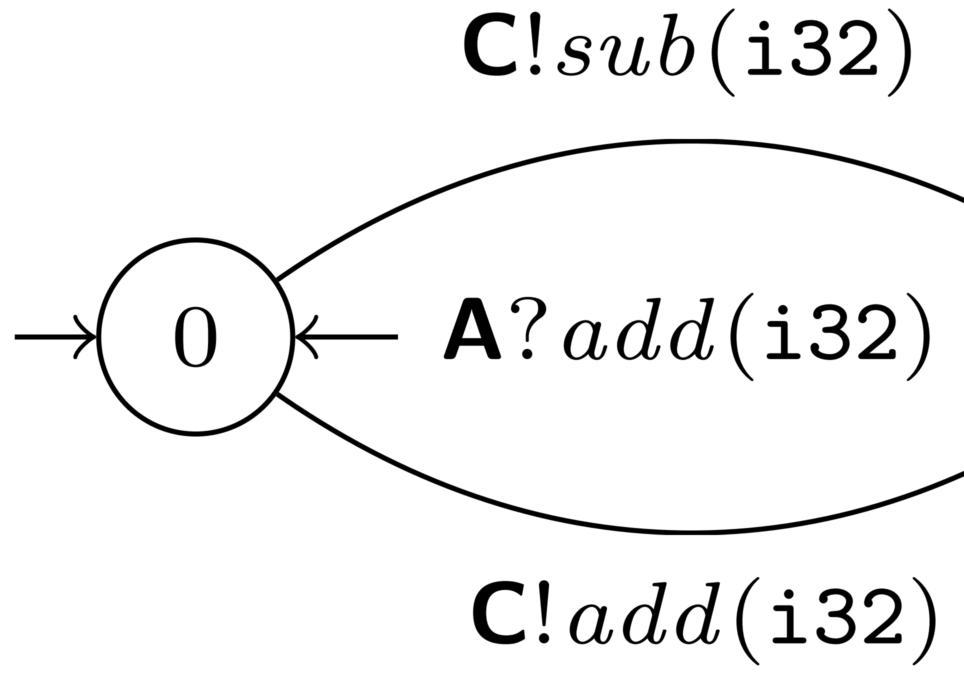
Rust API



```
#[derive(Role)]
#[message(Label)]
struct B(#[route(A)] Receiver, #[route(C)] Sender);

#[derive(Message)]
enum Label {
    Add(Add),
    Sub(Sub),
}

struct Add(i32);
struct Sub(i32);

#[session]
type RingB = Select<C, RingBChoice>;

#[session]
enum RingBChoice {
    Add(Add, Receive<A, Add, RingB>),
    Sub(Sub, Receive<A, Add, RingB>),
}
```

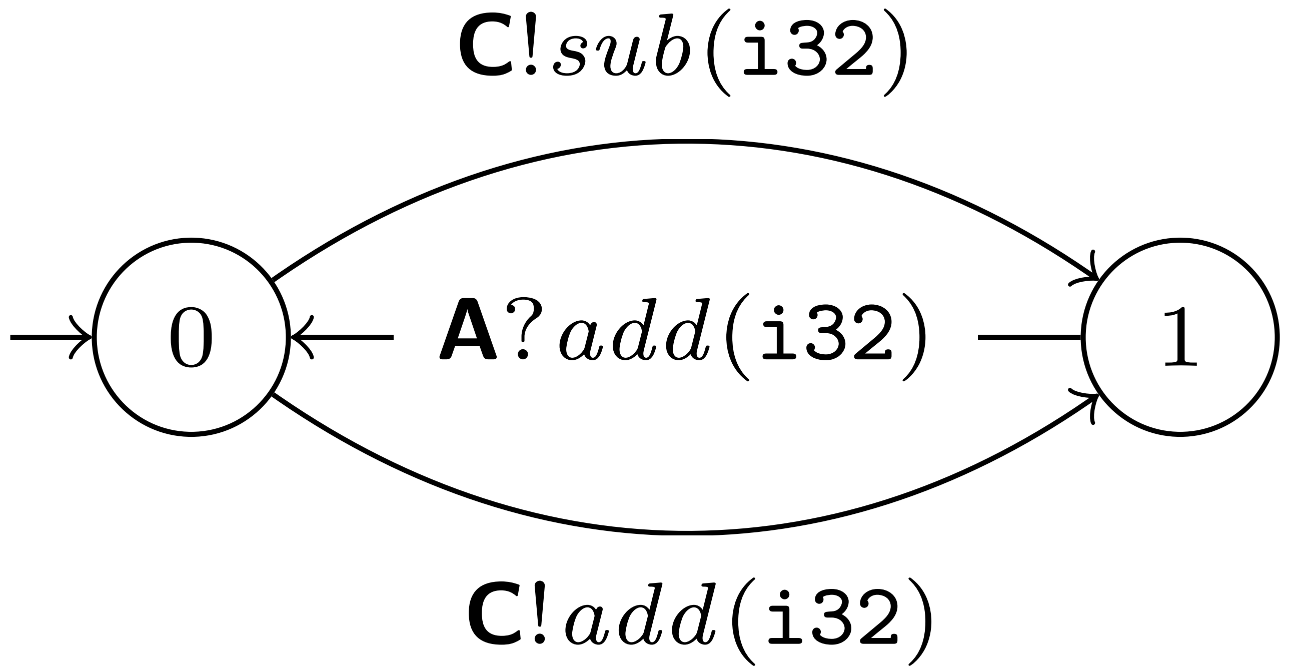
Ring Protocol

Rust API



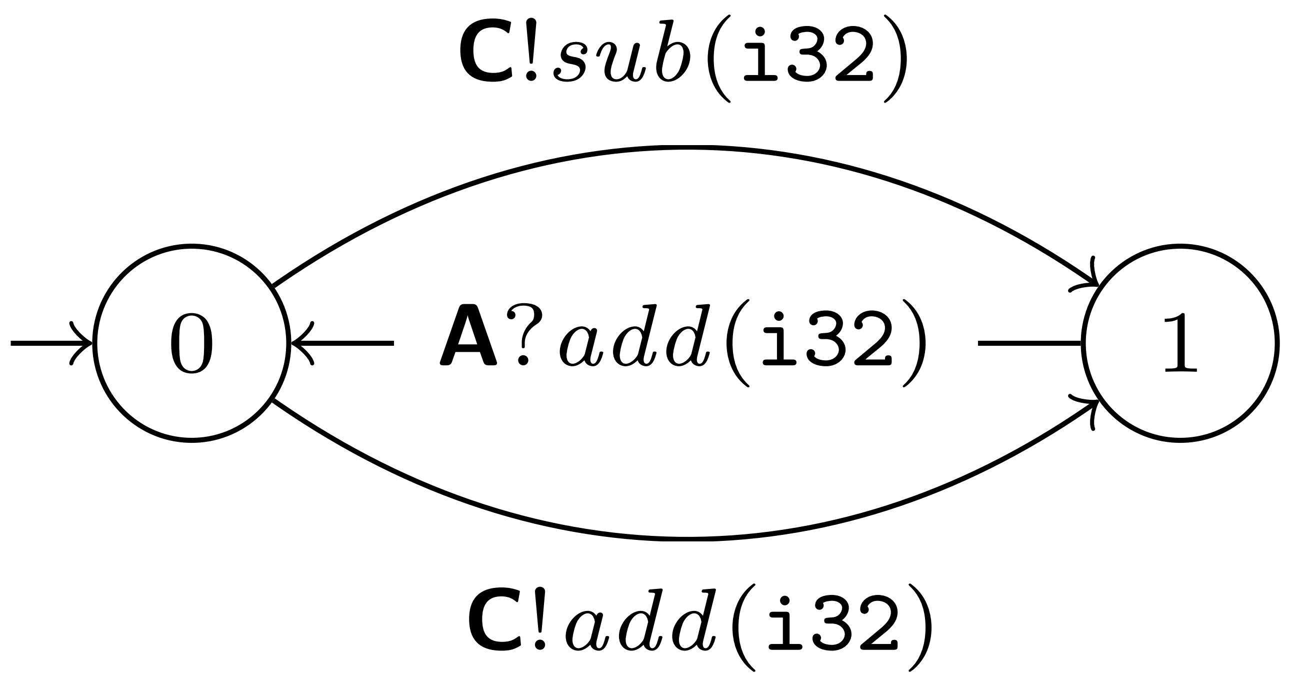
Ring Protocol

Implementation



Ring Protocol

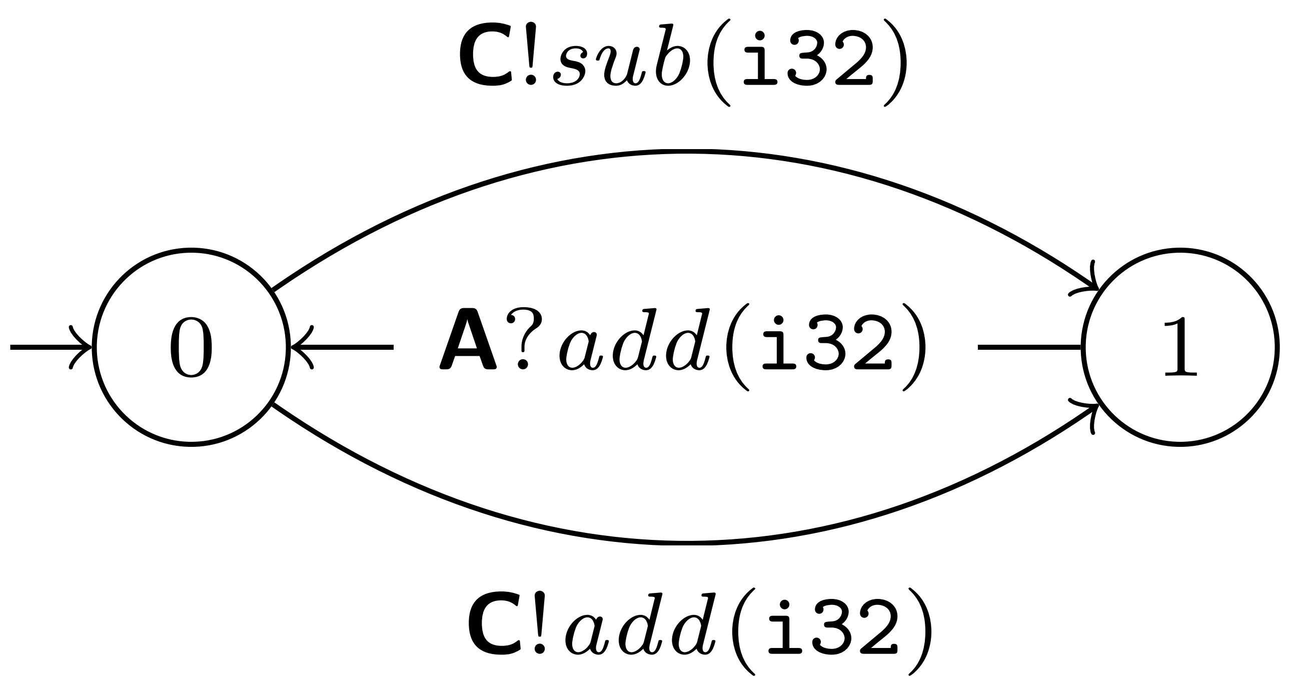
Implementation



```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            s = if x > 0 {
                let s = s.select(Add(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y + x;
                s
            } else {
                let s = s.select(Sub(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y - x;
                s
            };
        }
    })
    .await
}
```

Ring Protocol

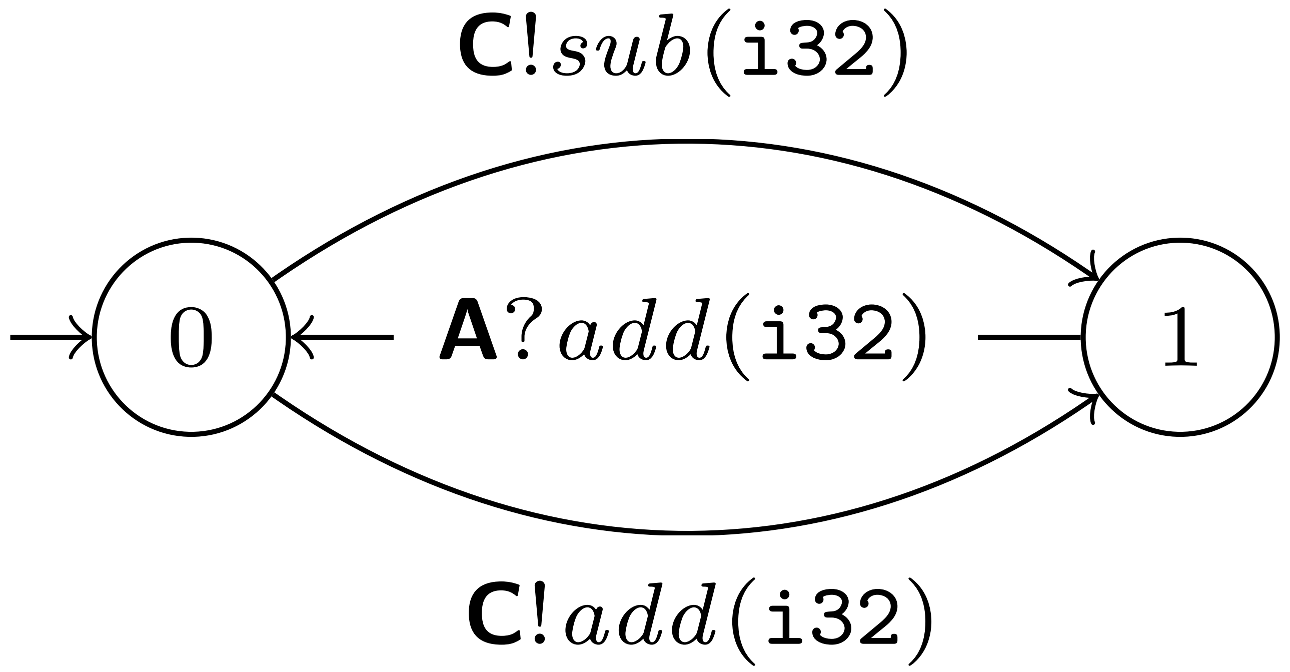
Implementation



```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            s = if x > 0 {
                let s = s.select(Add(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y + x;
                s
            } else {
                let s = s.select(Sub(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y - x;
                s
            };
        }
    })
    .await
}
```

Ring Protocol

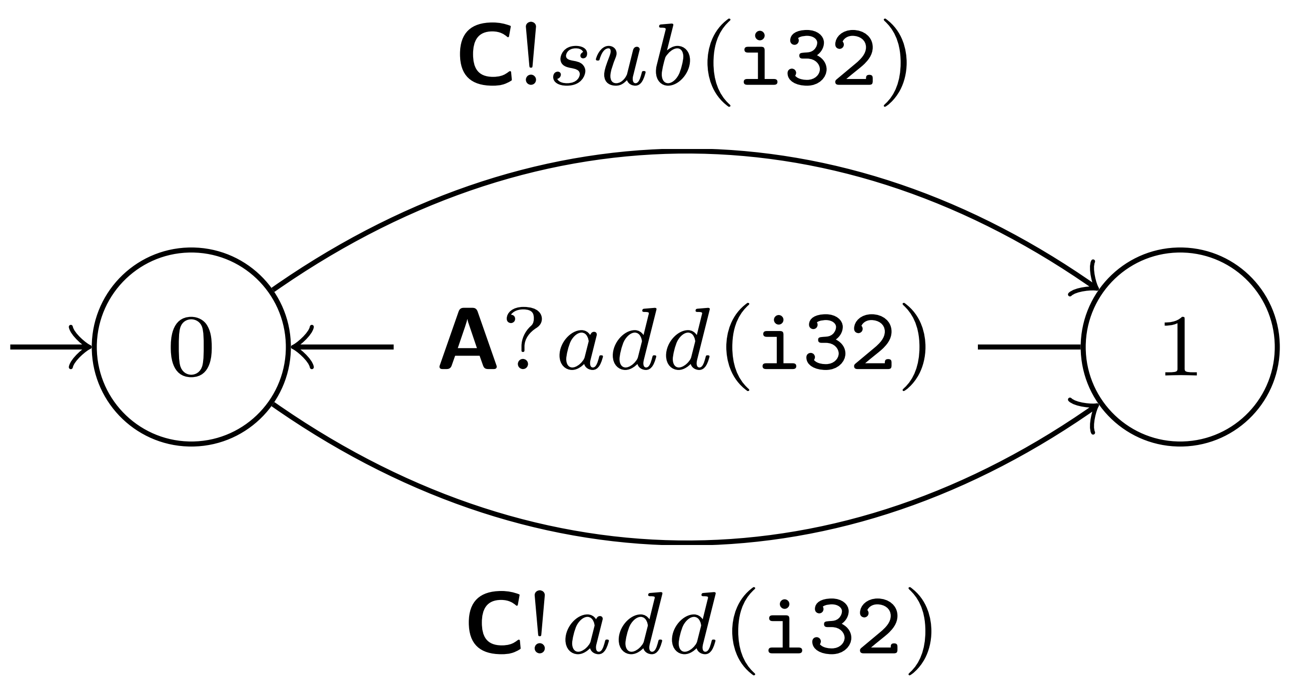
Implementation



```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            s = if x > 0 {
                let s = s.select(Add(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y + x;
                s
            } else {
                let s = s.select(Sub(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y - x;
                s
            };
        }
    })
    .await
}
```

Ring Protocol

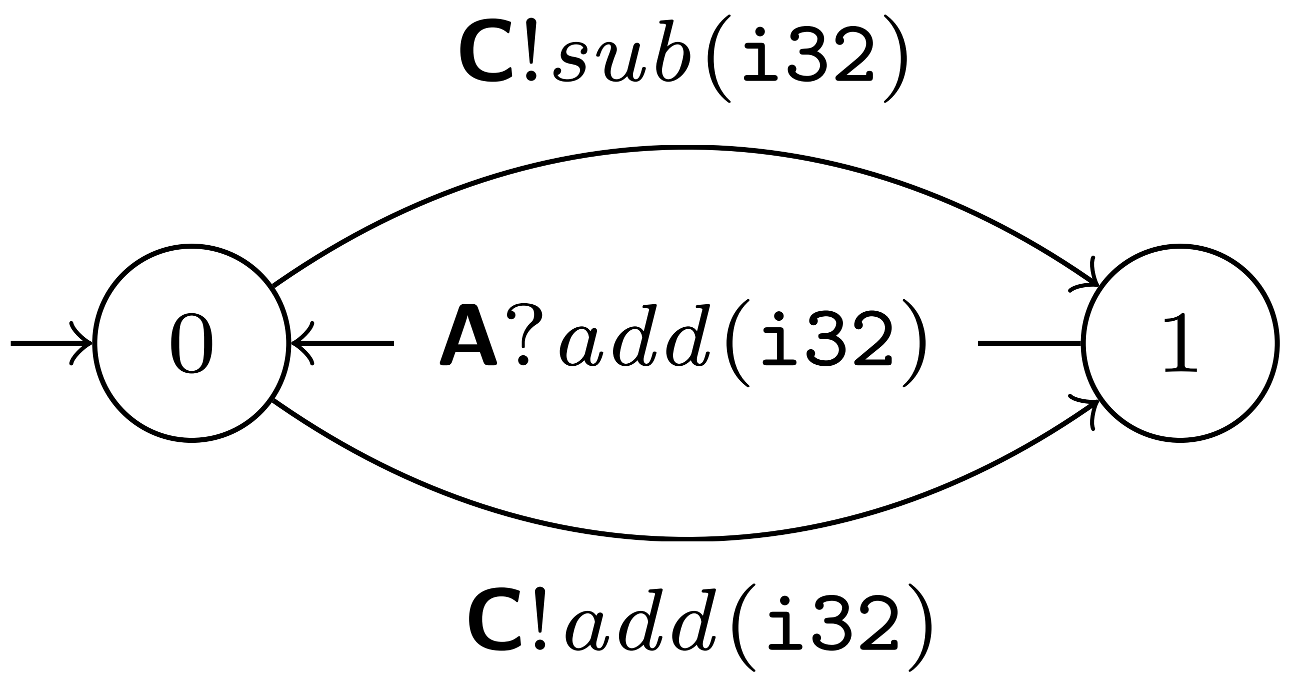
Implementation



```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            s = if x > 0 {
                let s = s.select(Add(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y + x;
                s
            } else {
                let s = s.select(Sub(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y - x;
                s
            };
        }
    })
    .await
}
```

Ring Protocol

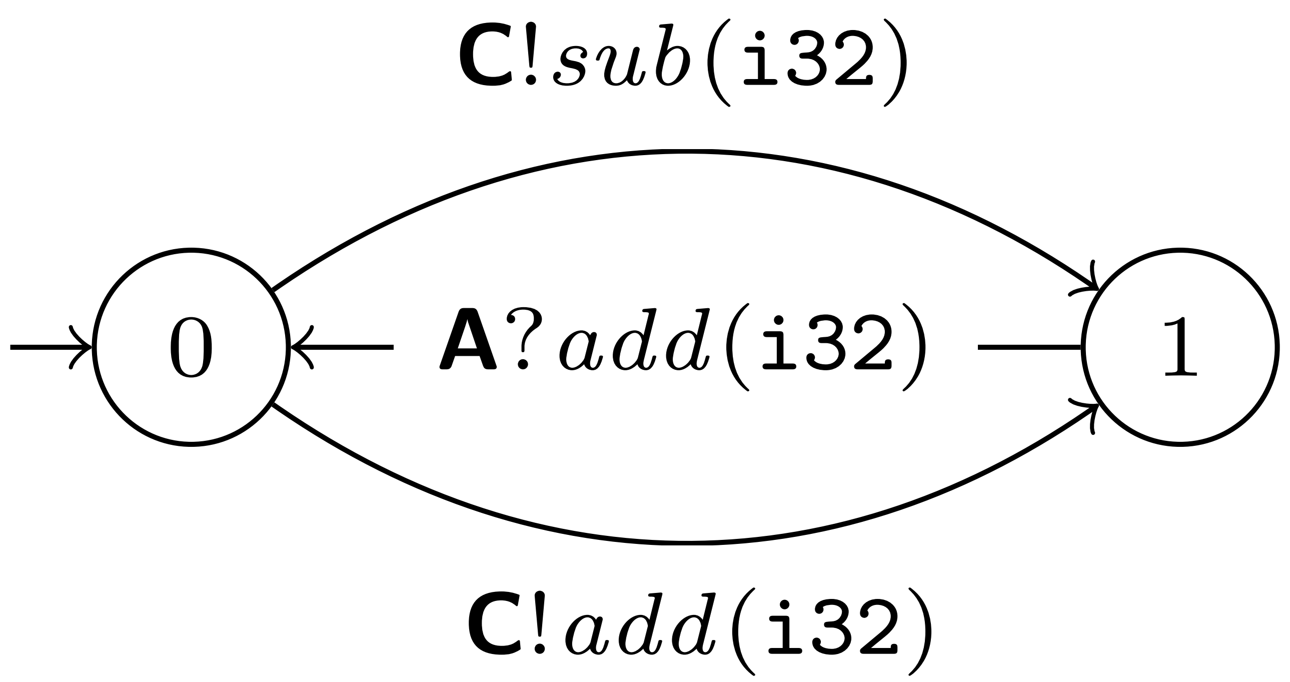
Implementation



```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            s = if x > 0 {
                let s = s.select(Add(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y + x;
                s
            } else {
                let s = s.select(Sub(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y - x;
                s
            };
        }
    })
    .await
}
```

Ring Protocol

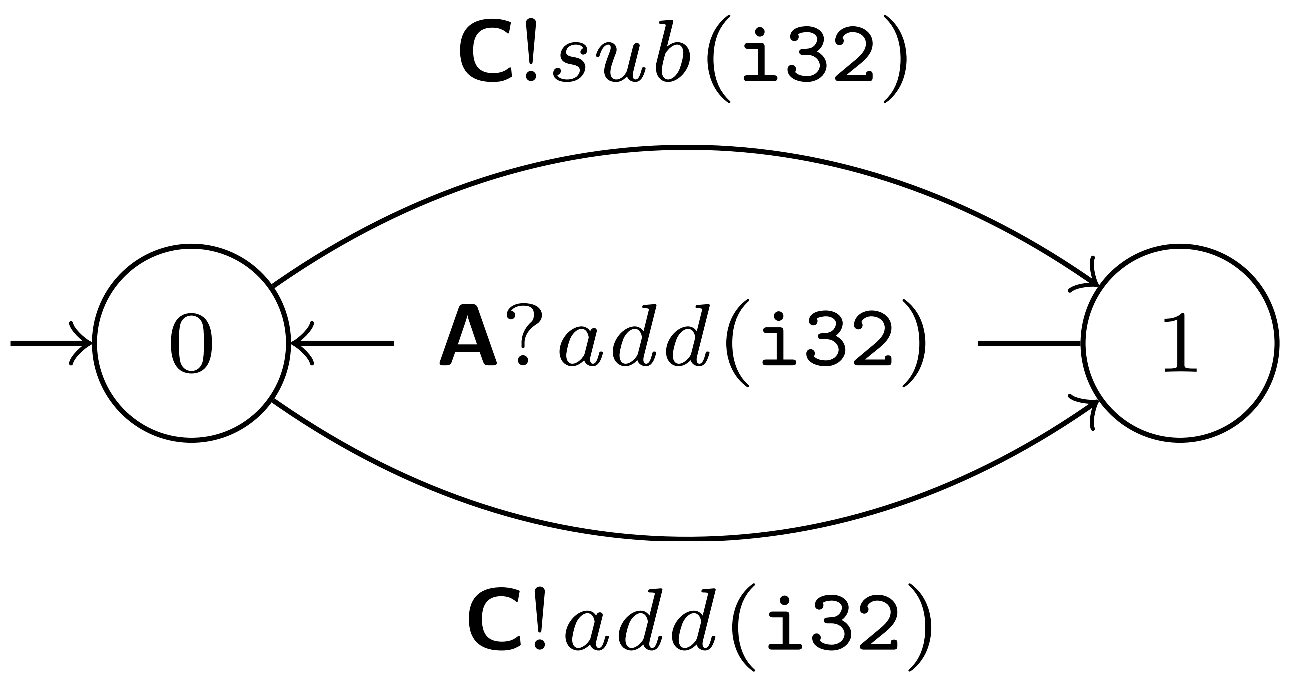
Implementation



```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            s = if x > 0 {
                let s = s.select(Add(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y + x;
                s
            } else {
                let s = s.select(Sub(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y - x;
                s
            };
        }
    })
    .await
}
```

Ring Protocol

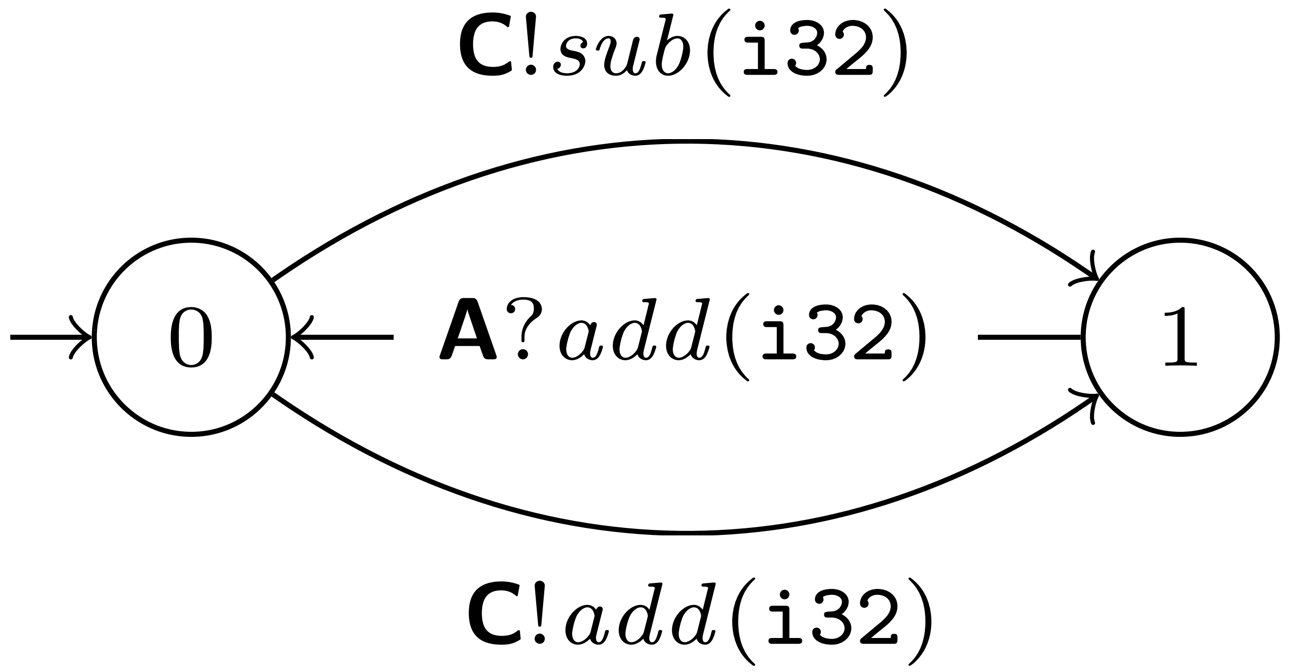
Implementation



```rust
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            s = if x > 0 {
                let s = s.select(Add(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y + x;
                s
            } else {
                let s = s.select(Sub(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y - x;
                s
            };
        }
    })
    .await
}
```

Ring Protocol

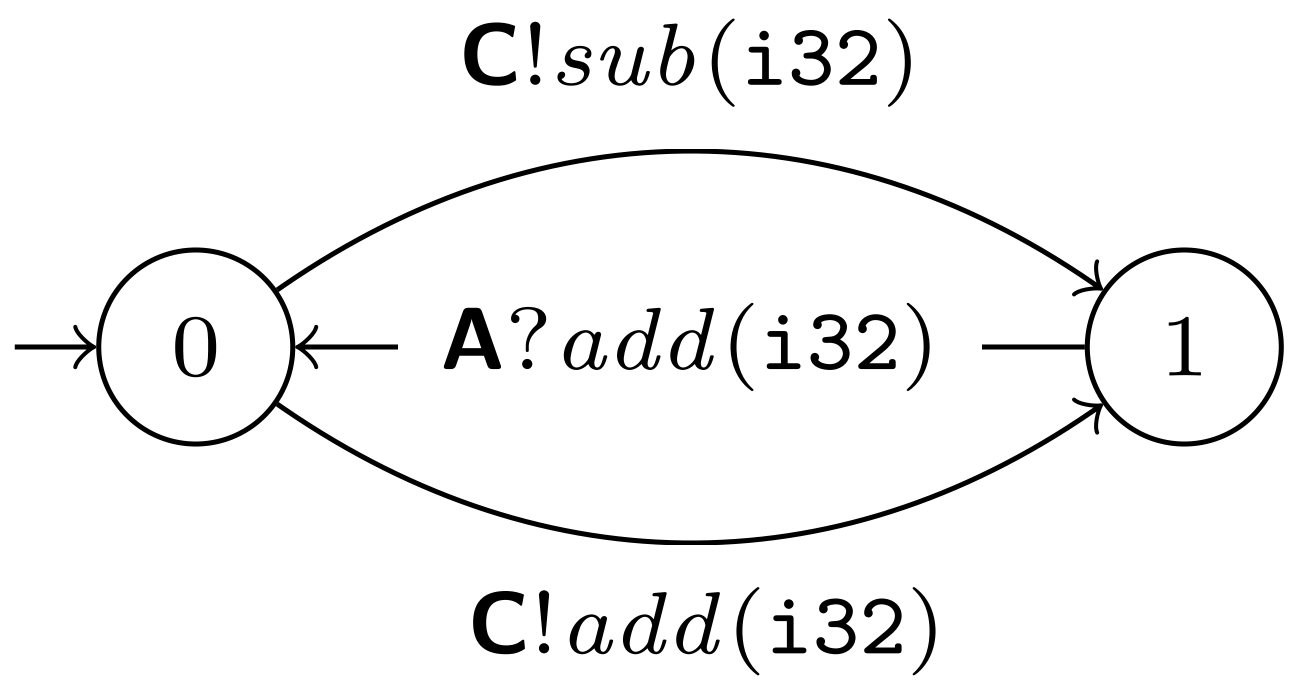
Implementation



```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            s = if x > 0 {
                let s = s.select(Add(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y + x;
                s
            } else {
                let s = s.select(Sub(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y - x;
                s
            };
        }
    })
    .await
}
```

Ring Protocol

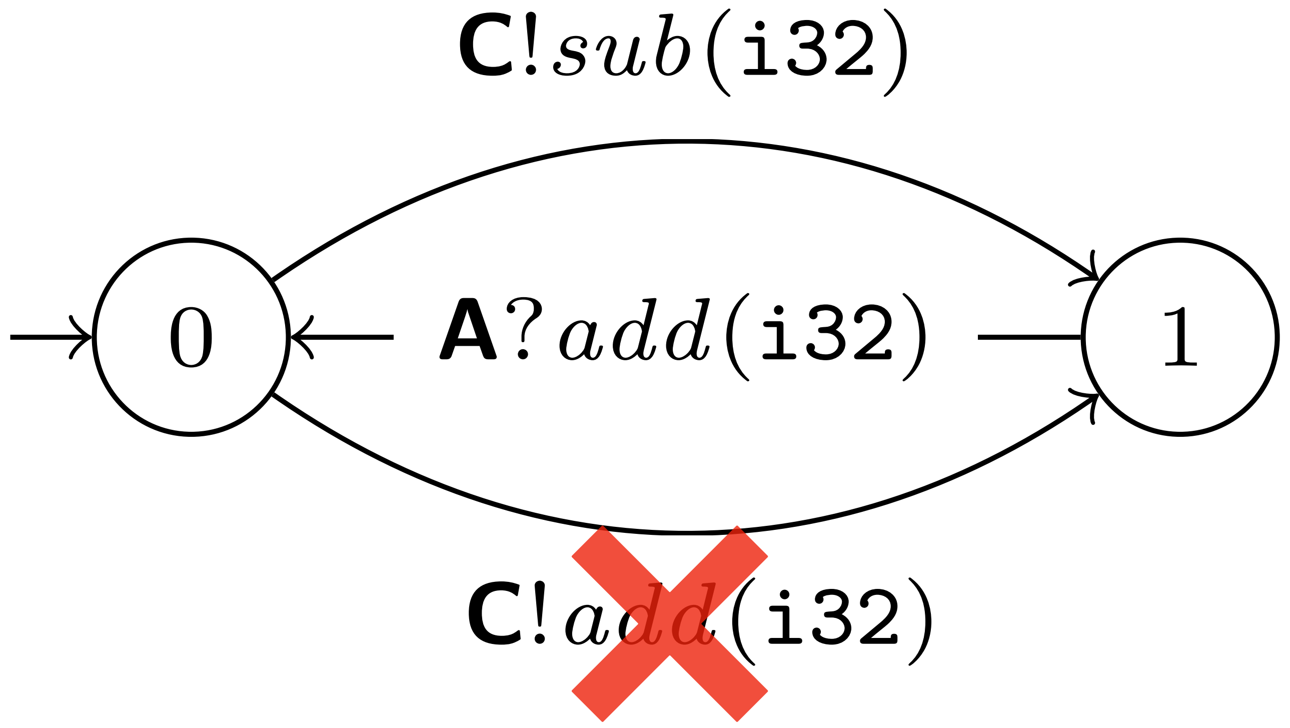
Implementation



```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            s = if x > 0 {
                let s = s.select(Add(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y + x;
                s
            } else {
                let s = s.select(Sub(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y - x;
                s
            };
        }
    })
    .await
}
```

Ring Protocol

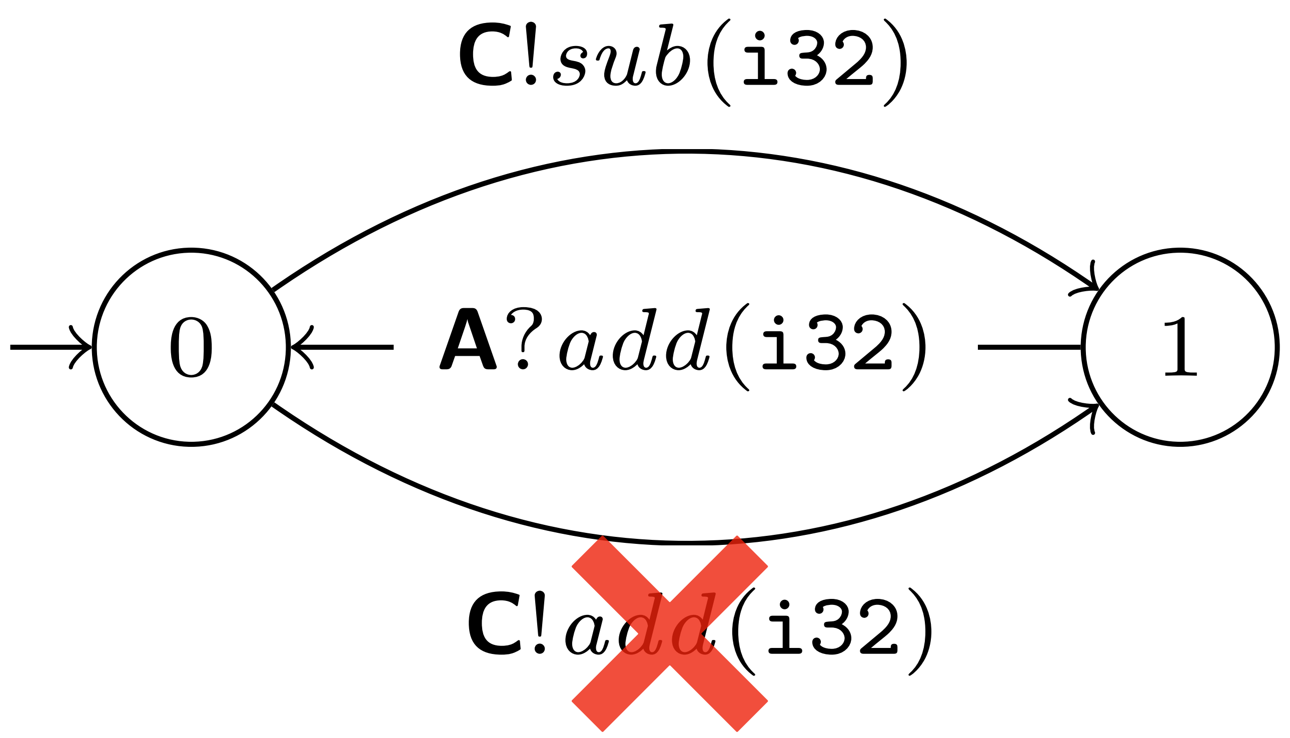
Implementation



```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            s = if x > 0 {
                ~~let s = s.select(Add(x)).await?;~~
                let (Add(y), s) = s.receive().await?;
                input = y + x;
                s
            } else {
                let s = s.select(Sub(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y - x;
                s
            };
        }
    })
    .await
}
```

Ring Protocol

Implementation



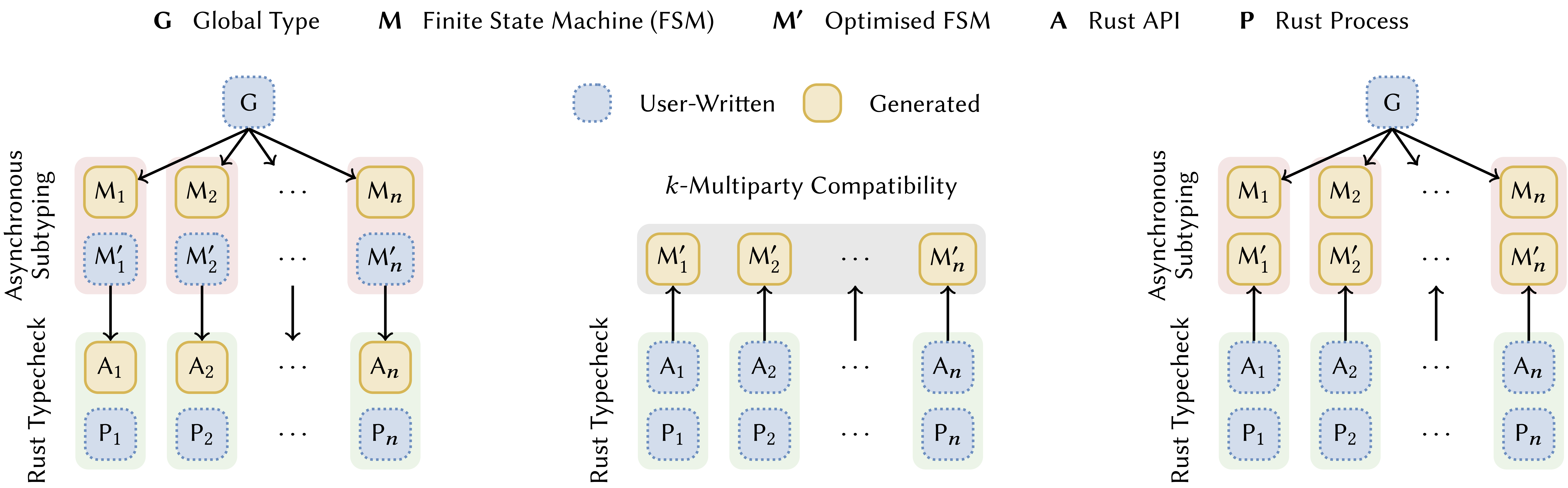
```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            s = if x > 0 {
                ~~let s = s.select(Add(x)).await?;~~
                let (Add(y), s) = s.receive().await?;
                input = y + x;
```

```
method not found in `rumpsteak::Select<'_, B, C, RingBChoice<'_, B>>`
```

```
                let s = s.select(Sub(x)).await?;
                let (Add(y), s) = s.receive().await?;
                input = y - x;
                s
            };
        }
    })
    .await
}
```

Rumpsteak Framework

Three Approaches



(a) Top-down

(b) Bottom-up

(c) Hybrid

Theories for Communication Optimisation

Asynchronous Reordering Revisited

How do we check that asynchronous reorderings are safe?

Theories for Communication Optimisation

Asynchronous Reordering Revisited

How do we check that asynchronous reorderings are safe?

1. Asynchronous subtyping [Ghilezan, Pantvic, Prokic, Scalas and NY **POPL'2021**]

Theories for Communication Optimisation

Asynchronous Reordering Revisited

How do we check that asynchronous reorderings are safe?

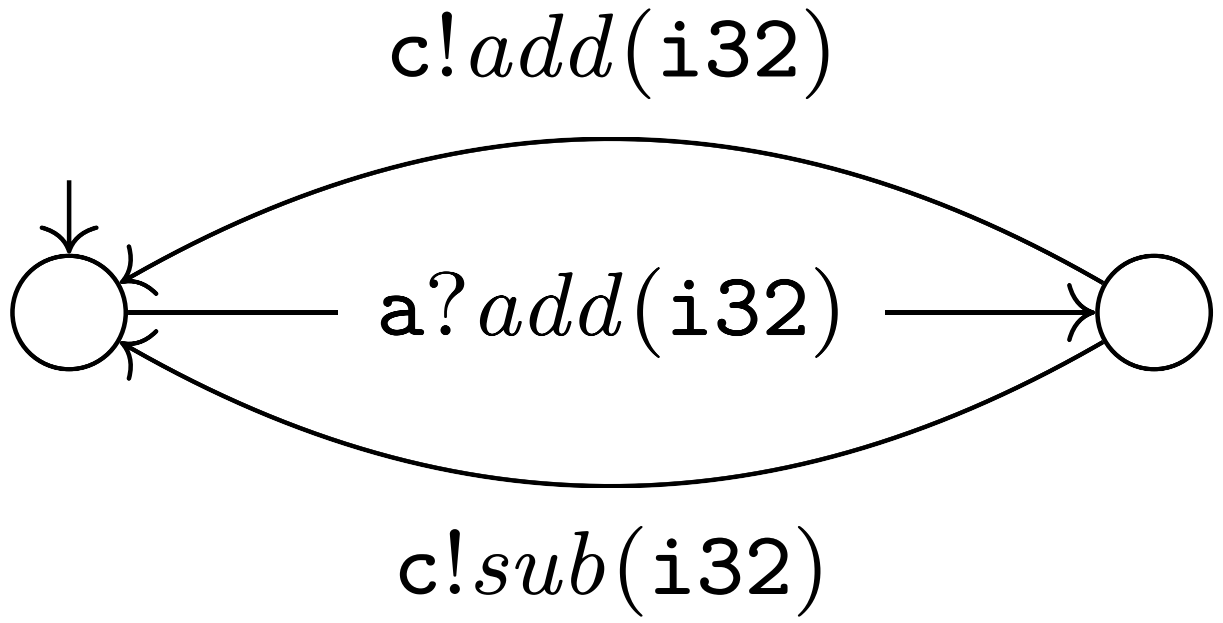
1. Asynchronous subtyping [Ghilezan, Pantvic, Prokic, Scalas and NY **POPL'2021**]

2. *k*-multiparty compatibility [Lange and NY, **CAV’2019**]

Safety

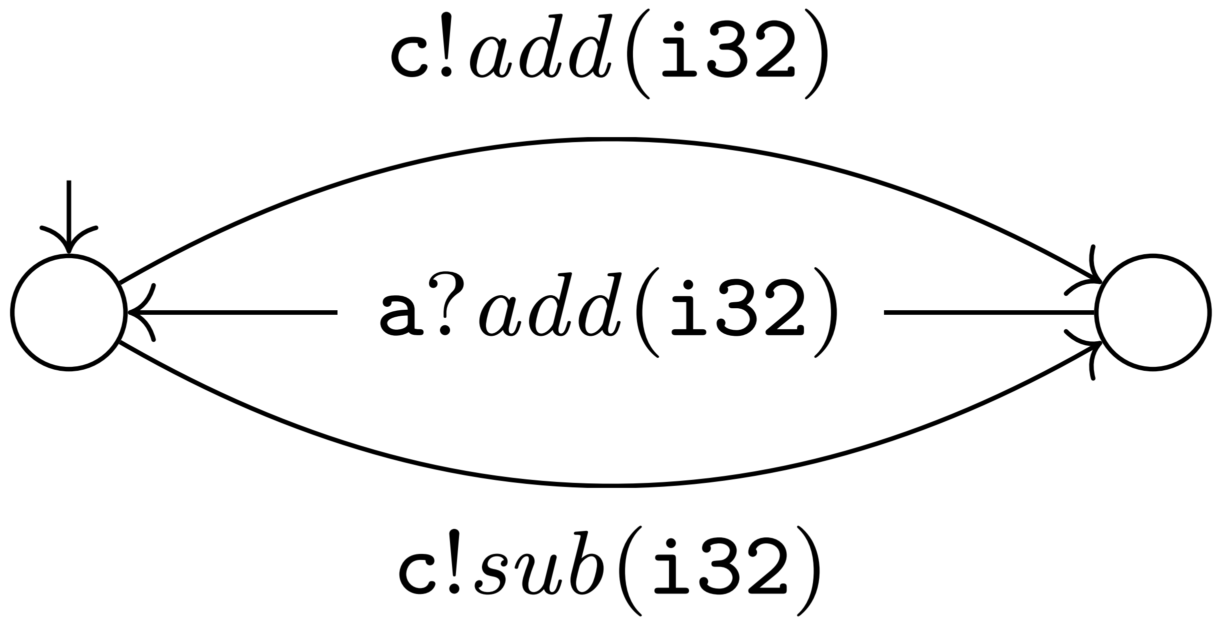
Asynchronous Subtyping

PROJECTED B



Safe?

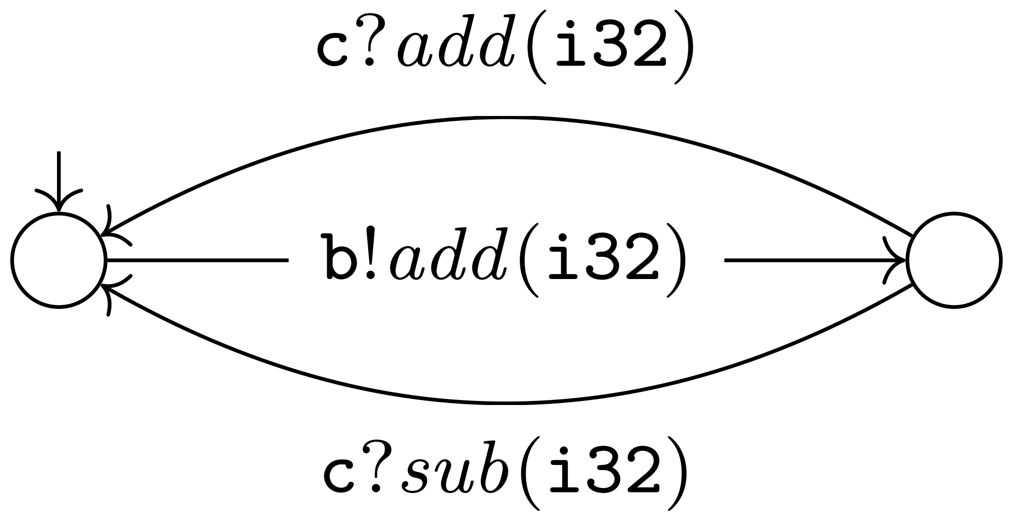
OPTIMISED B



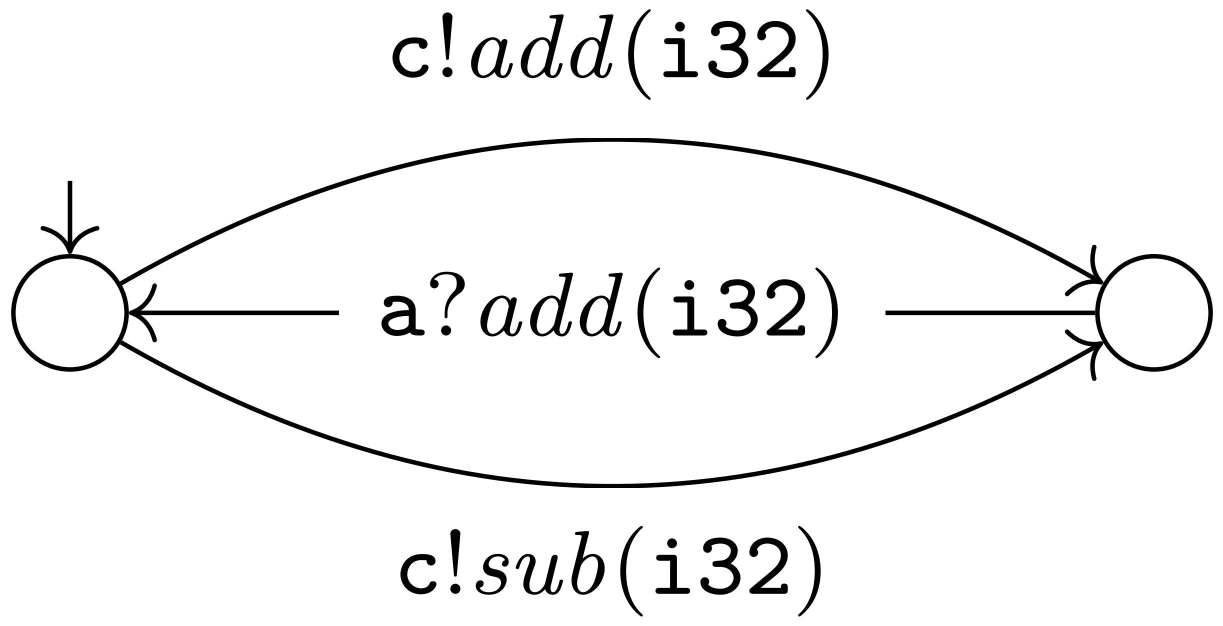
Safety

k-Multiparty Compatibility

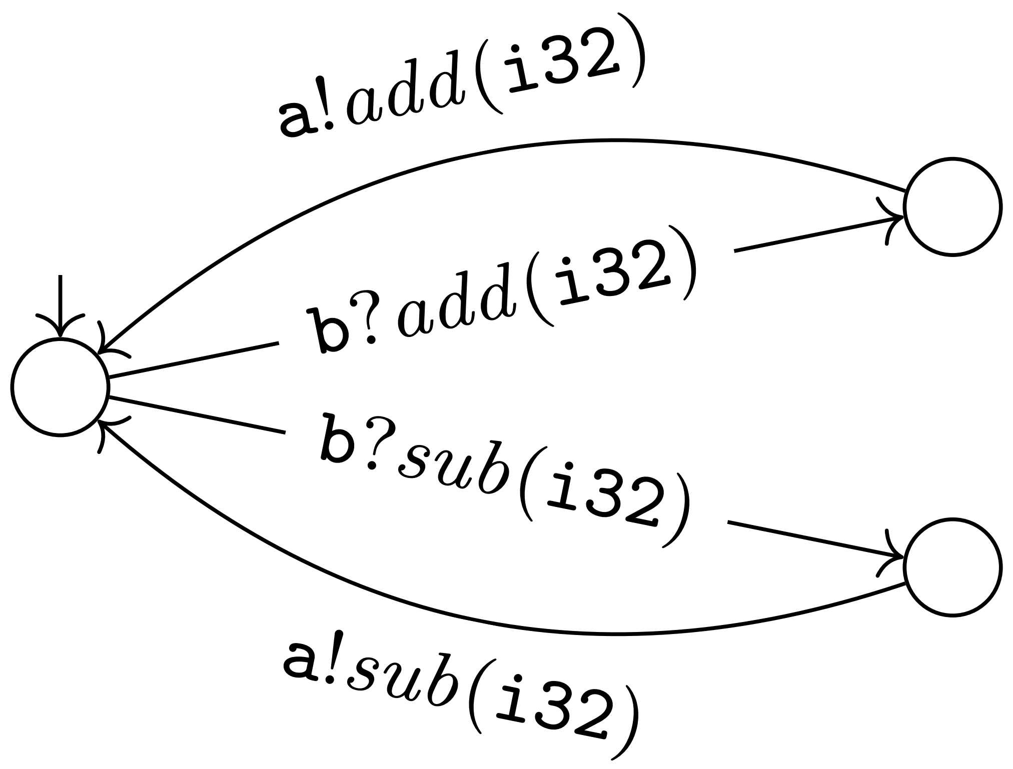
OPTIMISED A



OPTIMISED B

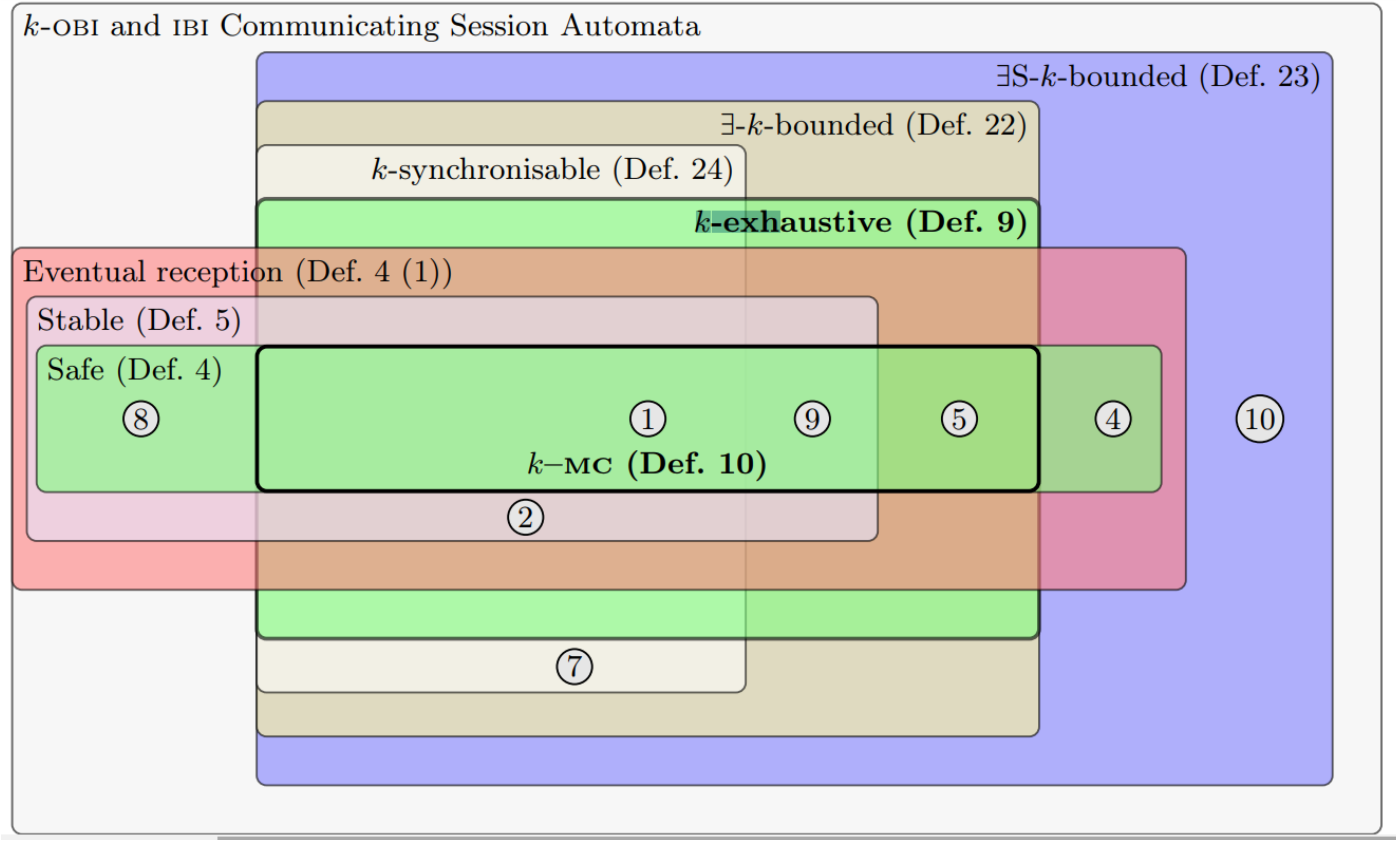


OPTIMISED C



Safe?

k-Multiparty Compatibility [CAV'19]


k-OBI and IBI Communicating Session Automata
∃S-k-bounded (Def. 23)
∃-k-bounded (Def. 22)
k-synchronisable (Def. 24)
k-exhaustive (Def. 9)
Eventual reception (Def. 4 (1))
Stable (Def. 5)
Safe (Def. 4)
8
1
9
5
4
10
k–MC **(Def. 10)**
2
7

Asynchronous Subtyping

Existing work

- Relation given by [Ghilezan et al., POPL 2021]

Theorem [POPL 2021]

Internal and external choices can be decomposed into single input and single output trees

Asynchronous Subtyping

Existing work

- Relation given by [Ghilezan et al., POPL 2021]
  - Sound ✅

Theorem [POPL 2021]

Internal and external choices can be decomposed into single input and single output trees

Asynchronous Subtyping

Existing work

- Relation given by [Ghilezan et al., POPL 2021]
  - Sound ✅
  - Complete ✅

Theorem [POPL 2021]

Internal and external choices can be decomposed into single input and single output trees

Asynchronous Subtyping

Existing work

- Relation given by [Ghilezan et al., POPL 2021]
  - Sound ✅
  - Complete ✅
  - Decidable [Lange and NY, FoSSaCs 2017] ❌

Theorem [POPL 2021]

Internal and external choices can be decomposed into single input and single output trees

Asynchronous Subtyping

Existing work

- Relation given by [Ghilezan et al., POPL 2021]
  - Sound ✅
  - Complete ✅
  - Decidable [Lange and NY, FoSSaCs 2017] ❌
- Our aim is a sound and decidable algorithm

Theorem [POPL 2021]

Internal and external choices can be decomposed into single input and single output trees

Asynchronous Subtyping

Existing work

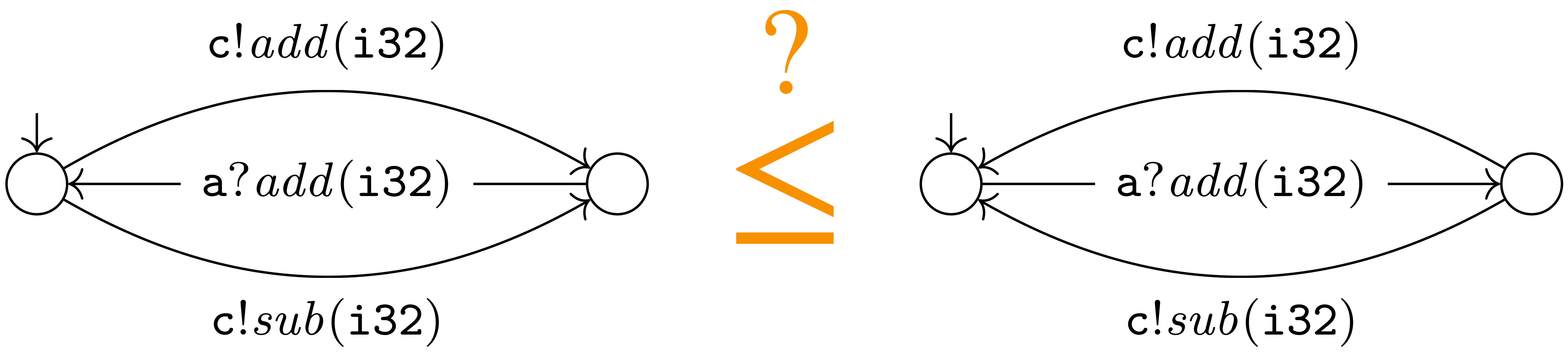
- Relation given by [Ghilezan et al., POPL 2021]
    - Sound ✅
    - Complete ✅
    - Decidable [Lange and NY, FoSSaCs 2017] ❌
- Our aim is a sound and decidable algorithm
- **Theorem [POPL 2021]**

  Internal and external choices can be decomposed into single input and single output trees

Asynchronous Subtyping

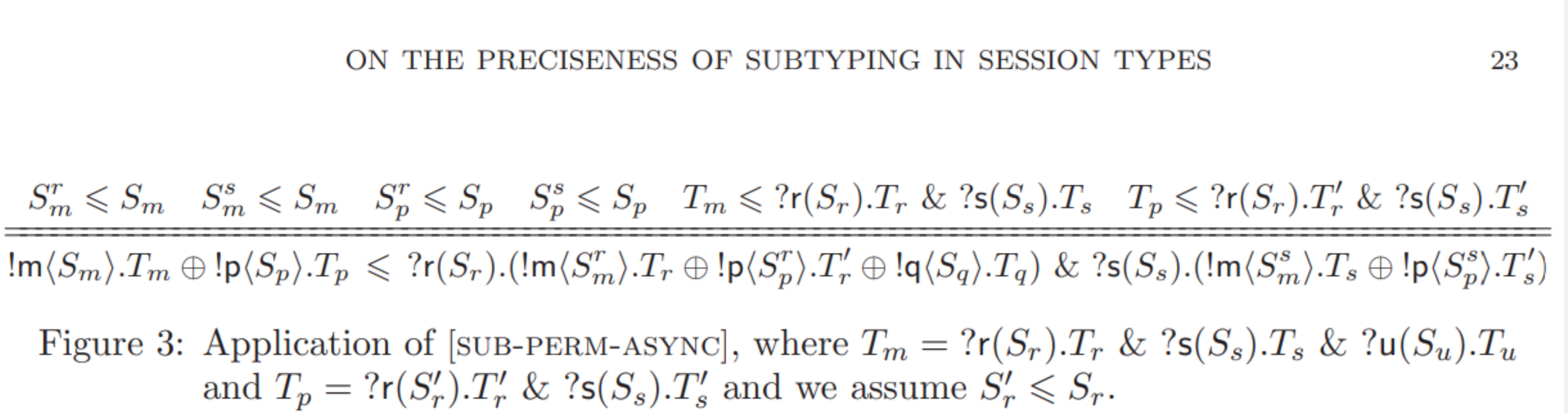
The Problem

- Choice and recursion make subtyping hard



Nested Session Asynchronous Subtyping

Precise Subtyping by Chen, Dezani et al



ON THE PRECISENESS OF SUBTYPING IN SESSION TYPES 23

$$\begin{array}{c} S^r_m \leqslant S_m \quad S^s_m \leqslant S_m \quad S^r_p \leqslant S_p \quad S^s_p \leqslant S_p \quad T_m \leqslant ?\mathsf{r}(S_r).T_r \,\&\, ?\mathsf{s}(S_s).T_s \quad T_p \leqslant ?\mathsf{r}(S_r).T'_r \,\&\, ?\mathsf{s}(S_s).T'_s \\ \hline\hline !\mathsf{m}\langle S_m\rangle.T_m \oplus !\mathsf{p}\langle S_p\rangle.T_p \leqslant ?\mathsf{r}(S_r).(!\mathsf{m}\langle S^r_m\rangle.T_r \oplus !\mathsf{p}\langle S^r_p\rangle.T'_r \oplus !\mathsf{q}\langle S_q\rangle.T_q) \,\&\, ?\mathsf{s}(S_s).(!\mathsf{m}\langle S^s_m\rangle.T_s \oplus !\mathsf{p}\langle S^s_p\rangle.T'_s) \end{array}$$

Figure 3: Application of [SUB-PERM-ASYNC], where $T_m = ?\mathsf{r}(S_r).T_r \,\&\, ?\mathsf{s}(S_s).T_s \,\&\, ?\mathsf{u}(S_u).T_u$ and $T_p = ?\mathsf{r}(S'_r).T'_r \,\&\, ?\mathsf{s}(S_s).T'_s$ and we assume $S'_r \leqslant S_r$.

$$\begin{aligned} \mathsf{T}_0 &= \mathsf{T}'_0 = \text{end} \\ \mathsf{T}_{n+1} &= !m.(?r.\mathsf{T}_n \,\&\, ?s.\mathsf{T}_n \,\&\, ?u.\mathsf{T}_n) \oplus !p.(?r.\mathsf{T}_n \,\&\, ?s.\mathsf{T}_n) \\ \mathsf{T}'_{n+1} &= ?r.(!m.\mathsf{T}'_n \oplus !p.\mathsf{T}'_n \oplus !q.\mathsf{T}'_n) \,\&\, ?s.(!m.\mathsf{T}'_n \oplus !p.\mathsf{T}'_n) \end{aligned}$$

Asynchronous Subtyping

SISO Refinement [POPL'21]

SISO trees are just paths — i.e. sequences of inputs and outputs!

$$\dfrac{}{\mathsf{end} \lesssim \mathsf{end}}$$

$$\dfrac{\mathsf{S}' \leqslant: \mathsf{S} \quad \mathsf{W} \lesssim \mathsf{W}'}{\mathbf{p}?\ell(\mathsf{S}).\mathsf{W} \lesssim \mathbf{p}?\ell(\mathsf{S}').\mathsf{W}'} \qquad \dfrac{\mathsf{S}' \leqslant: \mathsf{S} \quad \mathsf{W} \lesssim \mathcal{A}^{(\mathbf{p})}.\mathsf{W}' \quad \mathtt{act}(\mathsf{W}) = \mathtt{act}(\mathcal{A}^{(\mathbf{p})}.\mathsf{W}')}{\mathbf{p}?\ell(\mathsf{S}).\mathsf{W} \lesssim \mathcal{A}^{(\mathbf{p})}.\mathbf{p}?\ell(\mathsf{S}').\mathsf{W}'}$$

$$\dfrac{\mathsf{S} \leqslant: \mathsf{S}' \quad \mathsf{W} \lesssim \mathsf{W}'}{\mathbf{p}!\ell(\mathsf{S}).\mathsf{W} \lesssim \mathbf{p}!\ell(\mathsf{S}').\mathsf{W}'} \qquad \dfrac{\mathsf{S} \leqslant: \mathsf{S}' \quad \mathsf{W} \lesssim \mathcal{B}^{(\mathbf{p})}.\mathsf{W}' \quad \mathtt{act}(\mathsf{W}) = \mathtt{act}(\mathcal{B}^{(\mathbf{p})}.\mathsf{W}')}{\mathbf{p}!\ell(\mathsf{S}).\mathsf{W} \lesssim \mathcal{B}^{(\mathbf{p})}.\mathbf{p}!\ell(\mathsf{S}').\mathsf{W}'}$$

$$\mathcal{A}^{(\mathbf{p})} ::= \mathbf{q}?\ell(\mathsf{S}) \parallel \mathbf{q}?\ell(\mathsf{S}).\mathcal{A}^{(\mathbf{p})} \qquad \mathcal{B}^{(\mathbf{p})} ::= \mathbf{r}?\ell(\mathsf{S}) \parallel \mathbf{q}!\ell(\mathsf{S}) \parallel \mathbf{r}?\ell(\mathsf{S}).\mathcal{B}^{(\mathbf{p})} \parallel \mathbf{q}!\ell(\mathsf{S}).\mathcal{B}^{(\mathbf{p})} \quad (\mathbf{q} \neq \mathbf{p})$$

$$\dfrac{\forall \mathsf{U}' \in \llbracket \mathbb{T}' \rrbracket_{\mathsf{SO}} \quad \forall \mathsf{V} \in \llbracket \mathbb{T} \rrbracket_{\mathsf{SI}} \quad \exists \mathsf{W}' \in \llbracket \mathsf{U}' \rrbracket_{\mathsf{SI}} \quad \exists \mathsf{W} \in \llbracket \mathsf{V} \rrbracket_{\mathsf{SO}} \qquad \mathsf{W}' \lesssim \mathsf{W}}{\mathbb{T}' \leqslant \mathbb{T}}$$

Algorithm for Asynchronous Subtyping

Practical, Sound and Terminating

1. Bound the number of times we unroll recursions
2. Only unwrap choice on demand

Asynchronous Subtyping

Session Type Prefix

$$
\begin{array}{rcll}
\pi, \rho & ::= & \epsilon & \text{empty prefix} \\
& \mid & \mathtt{p}!\ell(S) & \text{message send} \\
& \mid & \mathtt{p}?\ell(S) & \text{message receive} \\
& \mid & \pi_1 . \pi_2 & \text{concatenation}
\end{array}
$$

Asynchronous Subtyping

Reduction Rules

$$\mathcal{A}^{(\mathtt{p})} \ ::= \ \mathtt{q}?\ell(S) \mid \mathtt{q}?\ell(S).\mathcal{A}^{(\mathtt{p})} \qquad (\mathtt{p} \neq \mathtt{q})$$

$$\frac{S' \leq: S}{\left\langle \mathtt{p}?\ell(S).\pi, \mathcal{A}^{(\mathtt{p})}.\mathtt{p}?\ell(S').\pi' \right\rangle \to \left\langle \pi, \mathcal{A}^{(\mathtt{p})}.\pi' \right\rangle}\ [\text{RED-}\mathcal{A}]$$

Asynchronous Subtyping

Reduction Rules

$$\mathcal{A}^{(\mathtt{p})} \;::=\; \mathtt{q}?\ell(S) \mid \mathtt{q}?\ell(S).\mathcal{A}^{(\mathtt{p})} \qquad (\mathtt{p} \neq \mathtt{q})$$

$$\frac{S' \leq: S}{\langle \mathtt{p}?\ell(S).\pi, \mathcal{A}^{(\mathtt{p})}.\mathtt{p}?\ell(S').\pi' \rangle \to \langle \pi, \mathcal{A}^{(\mathtt{p})}.\pi' \rangle}\,[\text{RED-}\mathcal{A}]$$

Asynchronous Subtyping

Reduction Rules

$$\mathcal{A}^{(\mathtt{p})} \ ::= \ \mathtt{q}?\ell(S) \mid \mathtt{q}?\ell(S).\mathcal{A}^{(\mathtt{p})} \qquad (\mathtt{p} \neq \mathtt{q})$$

$$\frac{S' \leq: S}{\langle \mathtt{p}?\ell(S).\pi, \mathcal{A}^{(\mathtt{p})}.\mathtt{p}?\ell(S').\pi' \rangle \to \langle \pi, \mathcal{A}^{(\mathtt{p})}.\pi' \rangle}\ [\text{RED-}\mathcal{A}]$$

Asynchronous Subtyping

Reduction Rules

$$\mathcal{A}^{(\mathtt{p})} \; ::= \; \mathtt{q}?\ell(S) \mid \mathtt{q}?\ell(S).\mathcal{A}^{(\mathtt{p})} \qquad (\mathtt{p} \neq \mathtt{q})$$

$$\frac{S' \leq: S}{\langle \mathtt{p}?\ell(S).\pi, \mathcal{A}^{(\mathtt{p})}.\mathtt{p}?\ell(S').\pi' \rangle \to \langle \pi, \mathcal{A}^{(\mathtt{p})}.\pi' \rangle} \; [\text{RED-}\mathcal{A}]$$

Asynchronous Subtyping

Reduction Rules

$$\langle \mathtt{p}?\ell(S).\mathtt{q}?m(S'), \mathtt{q}?m(S').\mathtt{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathtt{q}?m(S'), \mathtt{q}?m(S')\rangle$$

Asynchronous Subtyping

Reduction Rules

$$\langle \mathtt{p}?\ell(S).\mathtt{q}?m(S'), \mathtt{q}?m(S').\mathtt{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathtt{q}?m(S'), \mathtt{q}?m(S')\rangle$$

Asynchronous Subtyping

Reduction Rules

$$\langle \mathtt{p}?\ell(S).\mathtt{q}?m(S'), \mathtt{q}?m(S').\mathtt{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathtt{q}?m(S'), \mathtt{q}?m(S')\rangle$$

Asynchronous Subtyping

Reduction Rules

$$\langle \mathtt{p}?\ell(S).\mathtt{q}?m(S'), \mathtt{q}?m(S').\mathtt{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathtt{q}?m(S'), \mathtt{q}?m(S')\rangle$$ ✅

Asynchronous Subtyping

Reduction Rules

$$\langle \mathtt{p}?\ell(S).\mathtt{q}?m(S'), \underset{\substack{\uparrow \\ \mathcal{A}^{(\mathtt{p})}}}{\mathtt{q}?m(S')}.\mathtt{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathtt{q}?m(S'), \mathtt{q}?m(S')\rangle$$

✅

Asynchronous Subtyping

Reduction Rules

$$\langle \mathrm{p}?\ell(S).\mathrm{q}?m(S'), \mathrm{q}?m(S').\mathrm{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathrm{q}?m(S'), \mathrm{q}?m(S')\rangle$$ ✅

$$\langle \mathrm{p}?\ell(S).\mathrm{p}?m(S'), \mathrm{p}?m(S').\mathrm{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathrm{p}?m(S'), \mathrm{p}?m(S')\rangle$$

Asynchronous Subtyping

Reduction Rules

$$\langle \mathrm{p}?\ell(S).\mathrm{q}?m(S'), \mathrm{q}?m(S').\mathrm{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathrm{q}?m(S'), \mathrm{q}?m(S')\rangle$$ ✅

$$\langle \mathrm{p}?\ell(S).\mathrm{p}?m(S'), \mathrm{p}?m(S').\mathrm{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathrm{p}?m(S'), \mathrm{p}?m(S')\rangle$$

Asynchronous Subtyping

Reduction Rules

$$\langle \mathrm{p}?\ell(S).\mathrm{q}?m(S'), \mathrm{q}?m(S').\mathrm{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathrm{q}?m(S'), \mathrm{q}?m(S')\rangle$$ ✅

$$\langle \mathrm{p}?\ell(S).\mathrm{p}?m(S'), \mathrm{p}?m(S').\mathrm{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathrm{p}?m(S'), \mathrm{p}?m(S')\rangle$$

Asynchronous Subtyping

Reduction Rules

$$\langle \mathrm{p}?\ell(S).\mathrm{q}?m(S'), \mathrm{q}?m(S').\mathrm{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathrm{q}?m(S'), \mathrm{q}?m(S')\rangle$$ ✅

$$\langle \mathrm{p}?\ell(S).\mathrm{p}?m(S'), \mathrm{p}?m(S').\mathrm{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathrm{p}?m(S'), \mathrm{p}?m(S')\rangle$$ ❌

Asynchronous Subtyping

Reduction Rules

$$\langle \mathtt{p}?\ell(S).\mathtt{q}?m(S'), \mathtt{q}?m(S').\mathtt{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathtt{q}?m(S'), \mathtt{q}?m(S')\rangle$$ ✅

$$\langle \mathtt{p}?\ell(S).\mathtt{p}?m(S'), \underbrace{\mathtt{p}?m(S')}_{\mathcal{A}^{(\mathtt{p})}}.\mathtt{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathtt{p}?m(S'), \mathtt{p}?m(S')\rangle$$ ❌

Asynchronous Subtyping

Reduction Rules

$$\langle \mathrm{p}?\ell(S).\mathrm{q}?m(S'), \mathrm{q}?m(S').\mathrm{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathrm{q}?m(S'), \mathrm{q}?m(S')\rangle$$ ✅

$$\langle \mathrm{p}?\ell(S).\mathrm{p}?m(S'), \underbrace{\mathrm{p}?m(S')}_{\mathcal{A}^{(\mathrm{p})}}.\mathrm{p}?\ell(S)\rangle \xrightarrow{?} \langle \mathrm{p}?m(S'), \mathrm{p}?m(S')\rangle$$ ❌

$$\mathcal{A}^{(\mathrm{p})} \;::=\; \mathrm{q}?\ell(S) \mid \mathrm{q}?\ell(S).\mathcal{A}^{(\mathrm{p})} \qquad (\mathrm{p} \neq \mathrm{q})$$

Asynchronous Subtyping

Reduction Rules

$$\mathcal{B}^{(\mathtt{p})} \;::=\; \mathtt{r}?\ell(S) \mid \mathtt{q}!\ell(S) \mid \mathtt{r}?\ell(S).\mathcal{B}^{(\mathtt{p})} \mid \mathtt{q}!\ell(S).\mathcal{B}^{(\mathtt{p})} \qquad (\mathtt{p} \neq \mathtt{q})$$

$$\frac{S' \leq: S}{\langle \mathtt{p}!\ell(S).\pi, \mathcal{B}^{(\mathtt{p})}.\mathtt{p}!\ell(S').\pi' \rangle \rightarrow \langle \pi, \mathcal{B}^{(\mathtt{p})}.\pi' \rangle}\ [\text{RED-}\mathcal{B}]$$

Theorems

Termination, Soundness & Complexity

Lemma 3. *Given finite prefixes* π *and* π', $\langle \pi \mathbin{[\![]\!]} \pi' \rangle$ *can be reduced only a finite number of times.*

Theorem 4 (Termination)**.** *Our subtyping algorithm always eventually terminates.*

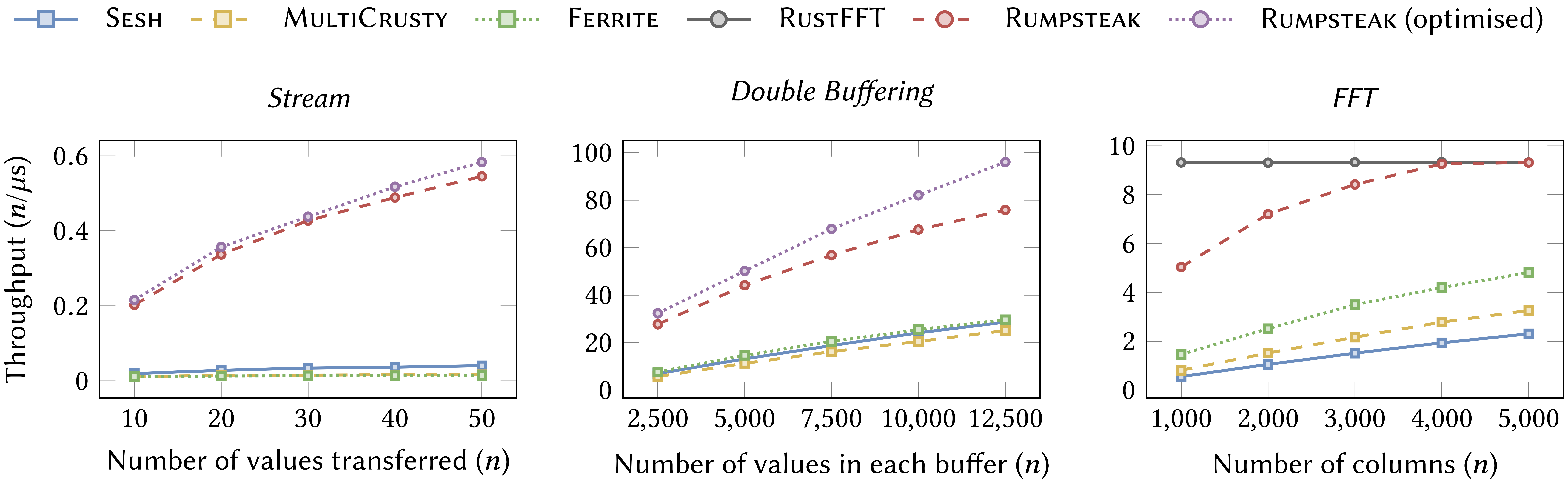
Theorem 5 (Soundness)**.** *Our subtyping algorithm is sound.*

Lemma 6. *Given finite prefixes* π *and* π'*, the time complexity of reducing* $\langle \pi \mathbin{[\![]\!]} \pi' \rangle$ *is* $\mathcal{O}(\min(|\pi|, |\pi'|))$.

Theorem 7 (Complexity)**.** *Consider* T *and* T' *as (possibly infinite) trees* $\mathcal{T}(\mathsf{T})$ *and* $\mathcal{T}(\mathsf{T}')$ *with* asymptotic branching factors b *and* b' *respectively. Our algorithm has time complexity* $\mathcal{O}(n \min(b, b')^n)$ *and space complexity* $\mathcal{O}(n \min(b, b'))$ *in the worst case to determine if* $\mathsf{T} \leq \mathsf{T}'$ *with bound* n.

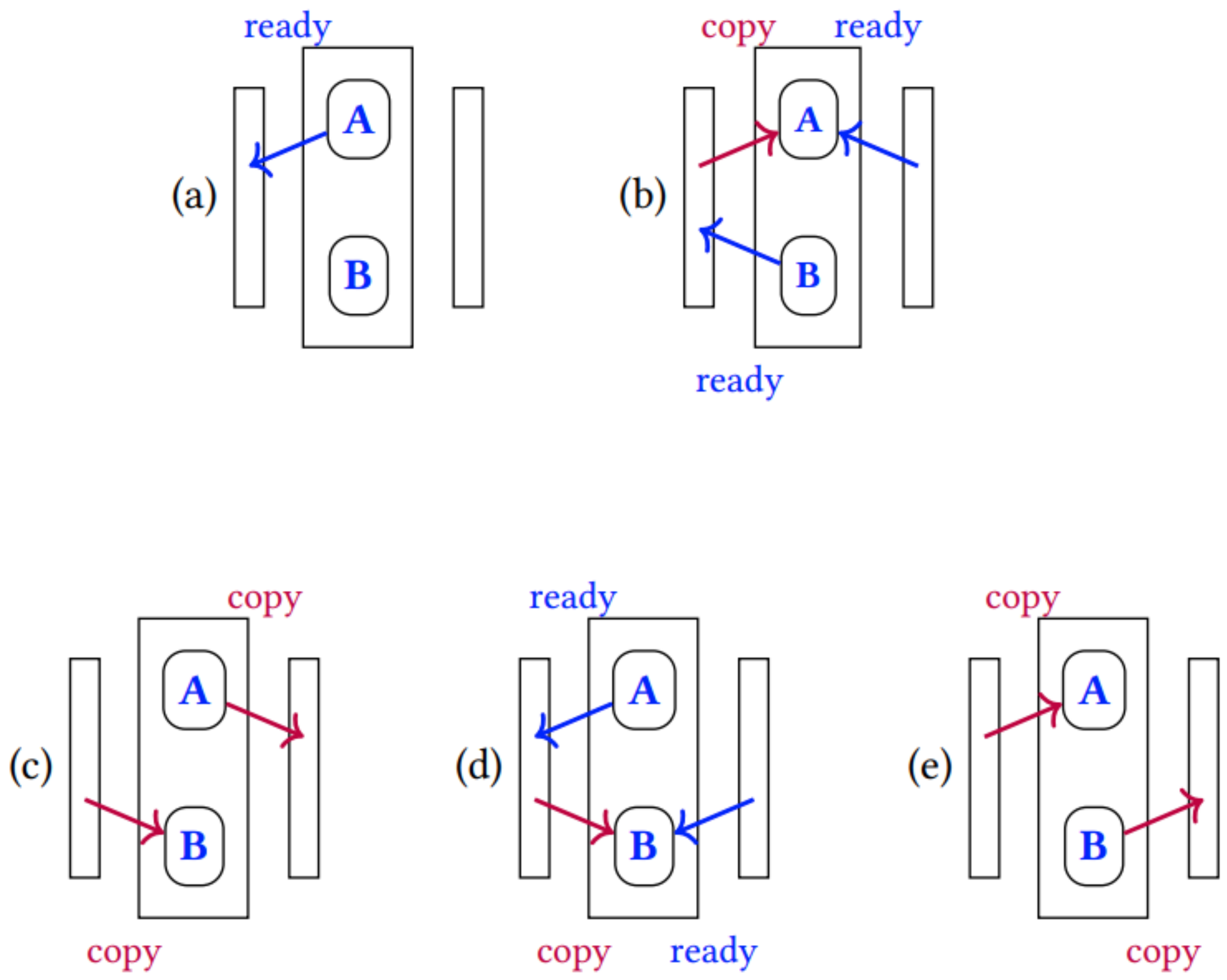
Evaluation

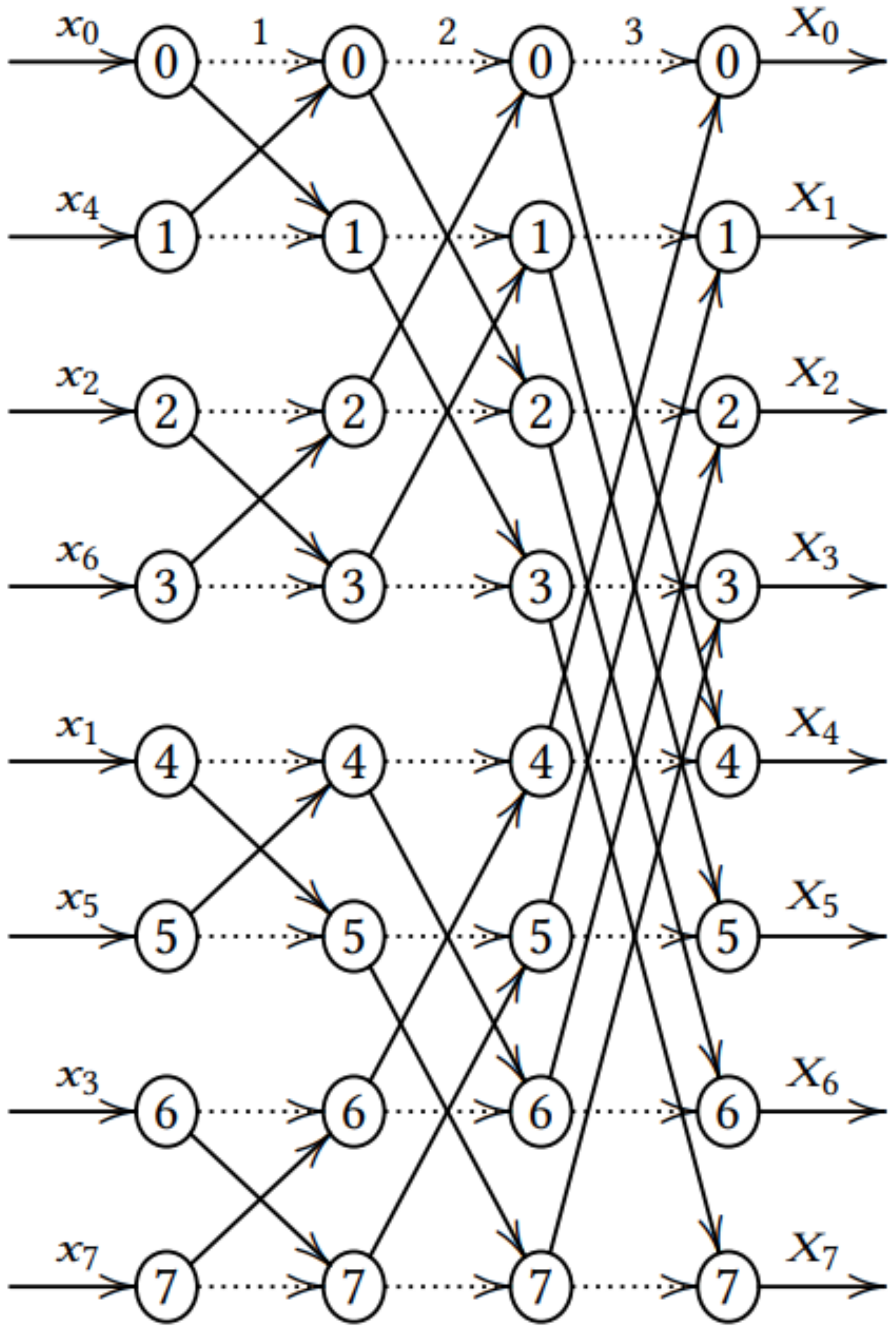
Rust Framework Benchmarks



16-core AMD OpteronTM 6200 Series CPU @ 2.6GHz with hyperthreading, 128GB of RAM, Ubuntu 18.04.5 LTS and Rust Nightly 2021-07-06.
We use version 0.3.5 of the Criterion.rs library and a multi-threaded asynchronous runtime from version 1.11.0 of the Tokio library.

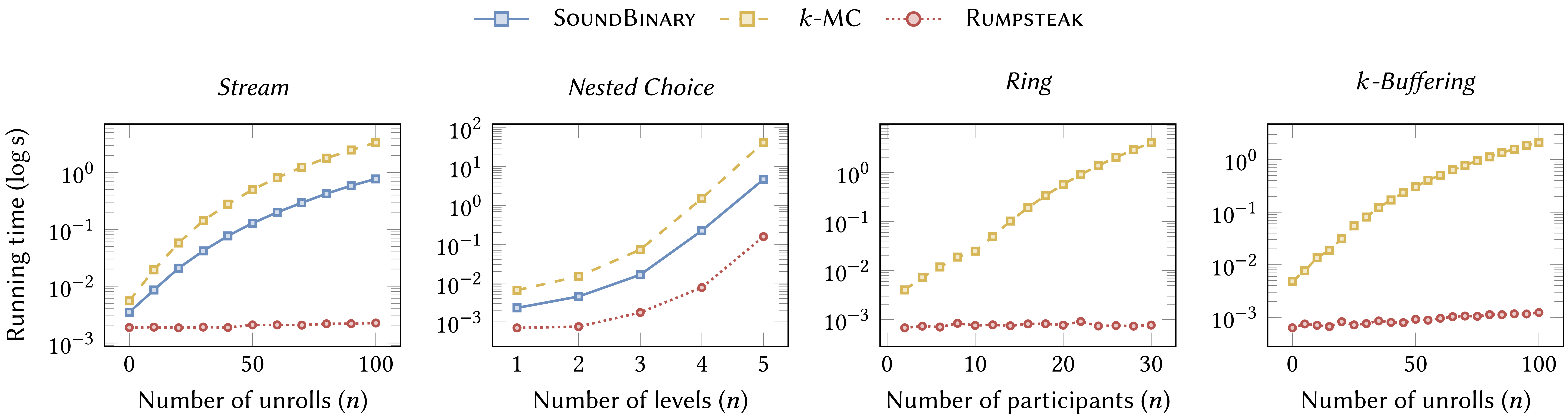
Double DB & Butterfly Topologies for FFT





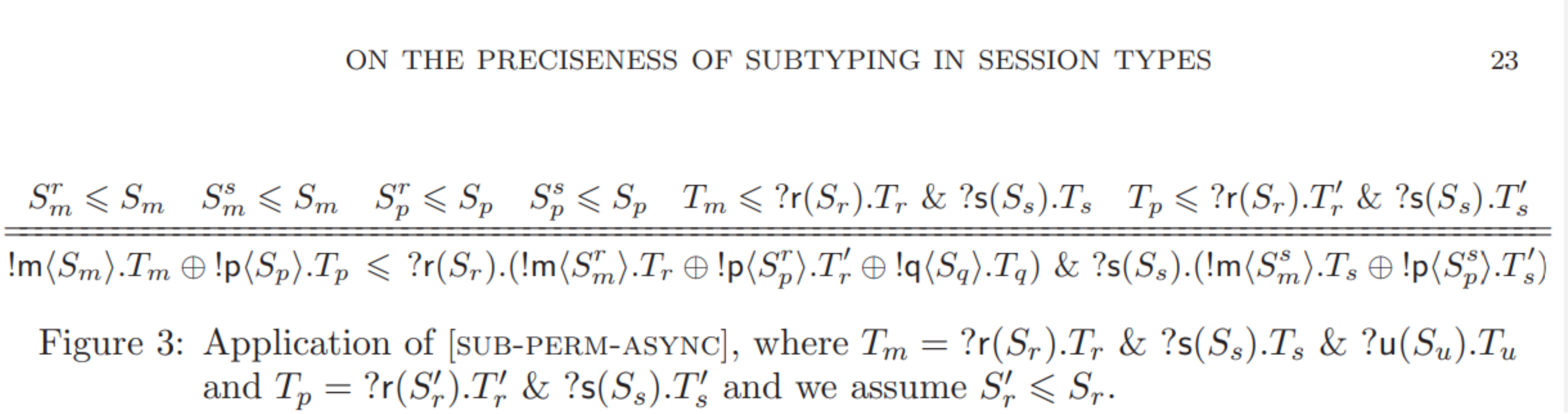
Evaluation

Asynchronous Reordering Benchmarks



Nested Session Asynchronous Subtyping

Precise Subtyping by Chen, Dezani et al



ON THE PRECISENESS OF SUBTYPING IN SESSION TYPES 23

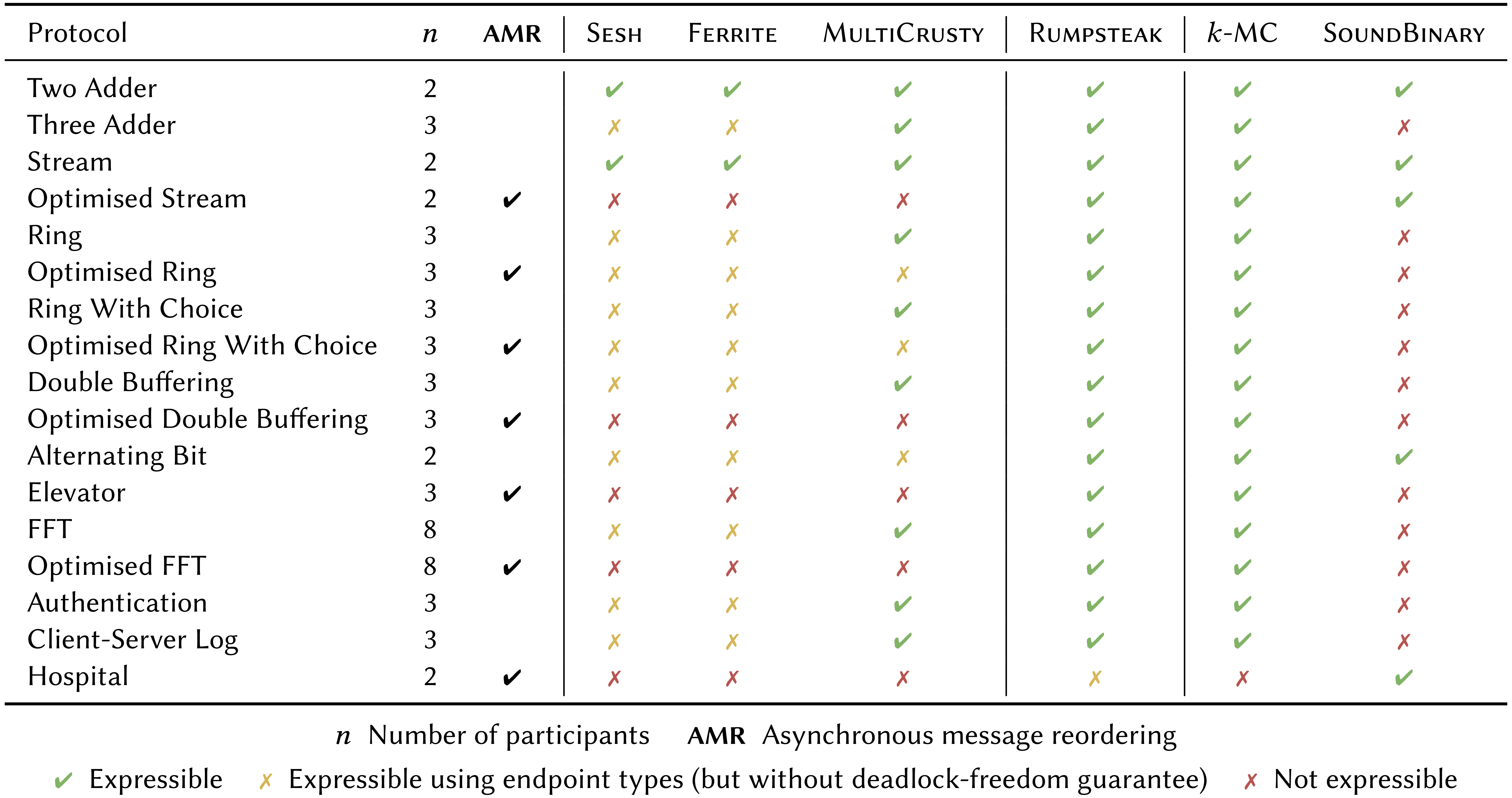
$$\frac{S^r_m \leqslant S_m \quad S^s_m \leqslant S_m \quad S^r_p \leqslant S_p \quad S^s_p \leqslant S_p \quad T_m \leqslant ?\mathsf{r}(S_r).T_r \ \&\ ?\mathsf{s}(S_s).T_s \quad T_p \leqslant ?\mathsf{r}(S_r).T'_r \ \&\ ?\mathsf{s}(S_s).T'_s}{!\mathsf{m}\langle S_m\rangle.T_m \oplus !\mathsf{p}\langle S_p\rangle.T_p \ \leqslant\ ?\mathsf{r}(S_r).(!\mathsf{m}\langle S^r_m\rangle.T_r \oplus !\mathsf{p}\langle S^r_p\rangle.T'_r \oplus !\mathsf{q}\langle S_q\rangle.T_q) \ \&\ ?\mathsf{s}(S_s).(!\mathsf{m}\langle S^s_m\rangle.T_s \oplus !\mathsf{p}\langle S^s_p\rangle.T'_s)}$$

Figure 3: Application of [SUB-PERM-ASYNC], where $T_m = ?\mathsf{r}(S_r).T_r \ \&\ ?\mathsf{s}(S_s).T_s \ \&\ ?\mathsf{u}(S_u).T_u$ and $T_p = ?\mathsf{r}(S'_r).T'_r \ \&\ ?\mathsf{s}(S_s).T'_s$ and we assume $S'_r \leqslant S_r$.

$$\begin{aligned}
\mathsf{T}_0 &= \mathsf{T}'_0 = \mathsf{end} \\
\mathsf{T}_{n+1} &= !m.(?r.\mathsf{T}_n \ \&\ ?s.\mathsf{T}_n \ \&\ ?u.\mathsf{T}_n) \oplus !p.(?r.\mathsf{T}_n \ \&\ ?s.\mathsf{T}_n) \\
\mathsf{T}'_{n+1} &= ?r.(!m.\mathsf{T}'_n \oplus !p.\mathsf{T}'_n \oplus !q.\mathsf{T}'_n) \ \&\ ?s.(!m.\mathsf{T}'_n \oplus !p.\mathsf{T}'_n)
\end{aligned}$$

Evaluation

Expressiveness



Protocol	n	AMR	Sesh	Ferrite	MultiCrusty	Rumpsteak	k-MC	SoundBinary
Two Adder	2		✔	✔	✔	✔	✔	✔
Three Adder	3		✗	✗	✔	✔	✔	✘
Stream	2		✔	✔	✔	✔	✔	✔
Optimised Stream	2	✓	✘	✘	✘	✔	✔	✔
Ring	3		✗	✗	✔	✔	✔	✘
Optimised Ring	3	✓	✗	✗	✗	✔	✔	✘
Ring With Choice	3		✗	✗	✔	✔	✔	✘
Optimised Ring With Choice	3	✓	✗	✗	✗	✔	✔	✘
Double Buffering	3		✗	✗	✔	✔	✔	✘
Optimised Double Buffering	3	✓	✘	✘	✘	✔	✔	✘
Alternating Bit	2		✗	✗	✗	✔	✔	✔
Elevator	3	✓	✘	✘	✘	✔	✔	✘
FFT	8		✗	✗	✔	✔	✔	✘
Optimised FFT	8	✓	✘	✘	✘	✔	✔	✘
Authentication	3		✗	✗	✔	✔	✔	✘
Client-Server Log	3		✗	✗	✔	✔	✔	✘
Hospital	2	✓	✘	✘	✘	✗	✘	✔

n Number of participants **AMR** Asynchronous message reordering

✔ Expressible ✗ Expressible using endpoint types (but without deadlock-freedom guarantee) ✘ Not expressible

References

Multiparty Session Types and Rust

- Multiparty session types and communicating automata
  - Invited paper in the FCT ’21 proceedings
  - Scribble **https://github.com/scribble**
  - **https://github.com/nuscr**
  - rumpsteak **https://github.com/zakcutner/rumpsteak**
- multi-crusty **http://mrg.doc.ic.ac.uk/tools/multicrusty/**
  - [ECOOP'22] **N. Lagaillardie (IC),** R. Neykova (Brunel), NY

Undergraduate and Master's Projects

- **Z. Cutner et al,** Deadlock-Free Asynchronous Message Reordering in Rust with Multiparty Session Types **[PPoPP 2022]**
- L. Gheri, I. Lanese, **N. Sayers,** E. Tuosto, NY, Design-by-Contract for Flexible Multiparty Session Protocols **[ECOOP 2022]**
- **A. Miu et al,** Communication-Safe Web Programming in TypeScript with Routed Multiparty Session Types **[CC 2021]**
- **F. Zhou et al**, Statically Verified Refinements for Multiparty Protocols **[OOPSLA 2020]**
- Castro-Perez & NY, Compiling First-Order Functions to Session-Typed Parallel Code **[CC 2020, Best Paper Award]**
- A. Scalas, NY, **E. Benussi**: Verifying message-passing programs with dependent behavioural types **[PLDI 2019]**
- R. Neykova, R. Hu, NY, **F. Abdeljallal**: A Session Type Provider: Compile-time API Generation for Distributed Protocols with Interaction Refinements in F# **[CC 2018]**

Current Projects

- **POST:** Protocols, Observabilities and Session Types **(EPSRC Established Career Fellowship)**
- Parallel Programming (Hardware)
  - **Morello-HAT**: Morello High-Level API and Tooling **(ISCF Digital Security by Design)** (GL, Essex)
  - **AppControl:** Enforcing Application Behaviour through Type-Based Constraints **(ISCF Digital Security by Design)** (GL, Essex)
  - **Border Patrol:** Improving Smart Device Security through Type-Aware Systems Design (GL, Heriot-Watt)
- Distributed Programming
  - **Stardust**: Session Types for Reliable Distributed Systems (GL, Kent)
  - **Turtles**: Protocol-Based Foundations for Distributed Multiagent Systems (Lancaster)
- Security and Safety (3 **VeTSS** projects on Rust, Go and CPS; **Safe-Trusted AI CDT**)

Mechanisations (Current & Ex Postdocs)

- **Zooid** **[PLDI'21]** Multiparty Session Types Framework in **Coq**
  - Castro-Perez (Kent), Ferriera (Royal Holloway), **Gheri** (IC) **& Vassor** (IC)
- **Idris & Agda**
  - **Barwell** (IC)
- More **Isabelle/HOL** & **Coq** Experts
  - **Hou** (IC) & **Gheri** (IC)
  - Christin Urban & Andrei Popescu

2.5Gb
Calgary
CANARIE n*10Gb
Winnipeg
CANARIE n*10Gb
Vancouver
Seattle
Pacific Wave
CANARIE-GLORIAD 2.5Gb
KREONet2-GLORIAD 1Gb
CiscoWave 2*10Gb
CANARIE n*10Gb
UltraScience Net 20Gb
TransLight 10Gb
CAVEwave 10Gb
CSTNet-GLORIAD 1Gb
CANARIE n*1
Montreal
MiLR
UltraLight
10Gb
Chicago
Toronto
Hal
CANARIE-GLORIAD 1Gb
TaiwanLight 2.5Gb
CAVEwave 10Gb
Pacific Wave 2*10Gb
CiscoWave 2*10Gb
UltraScience Net 20Gb
ESnet-SDN 2*10Gb
National LambdaRail n*10Gb
Internet2-DCN n*10Gb
UltraLight 10-20Gb
ESnet-SDN 2*10Gb
NLR n*10Gb-GLORIAD 10Gb
Fermi Lightpath 22Gb
StarLight
Boston
Salt Lake City
Pittsburgh
MAN LAN
Internet2-DCN n*10Gb
Denver
Urbana
Indianapolis
NGIX-East
New York
Pacific Wave
Sunnyvale
JGN2plus 10Gb
Kansas City
IllinoisWave 10Gb
Washington DC
Oak Ridge
Raleigh
CAVEwave 10Gb
Los Angeles
Pacific Wave
CiscoWave 2*10Gb
Tulsa
LEAD 10Gb
SLR 10Gb-GLORIAD 1Gb
TeraFlow 10Gb
CiscoWave 10Gb
National LambdaRail n*10Gb
Internet2-DCN n*10Gb
EnLIGHTened 10Gb
San Diego
Atlanta
ESnet-SDN 10Gb
Tijuana
Internet2-DCN n*10Gb
CENIC 10Gb
LONI 10Gb
National LambdaRail n*10Gb
WHREN-LILA 2Gb
El Paso
Jacksonville
Houston
Baton Rouge
AtlanticWave 10Gb
AMPATH
Miami
WH

Ocean Observatories Initiative

Dynamic Runtime Monitoring

2.5Gb
CANARIE n*10Gb
Calgary
Winnipeg
CANARIE n*10Gb
Vancouver
Seattle
Pacific Wave
CANARIE-GLORIAD 2.5Gb
KREONet2-GLORIAD 1Gb
CiscoWave 2*10Gb
CANARIE n*10Gb
UltraScience Net 20Gb
TransLight 10Gb
CAVEwave 10Gb
CSTNet-GLORIAD 1Gb
MiLR UltraLight 10Gb
CANARIE n*1
Montreal
Chicago
Toronto
Hal
CAVEwave 10Gb
Pacific Wave 2*10Gb
CiscoWave 2*10Gb
UltraScience Net 20Gb
ESnet-SDN 2*10Gb
National LambdaRail n*10Gb
Internet2-DCN n*10Gb
UltraLight 10-20Gb
ESnet-SDN 2*10Gb
NLR n*10Gb-GLORIAD 10Gb
Fermi Lightpath 22Gb
StarLight
CANARIE-GLORIAD 1Gb
TaiwanLight 2.5Gb
Boston
Salt Lake City
Pittsburgh
MAN LAN
New York
Internet2-DCN n*10Gb
Denver
Urbana
Indianapolis
NGIX-East
Pacific Wave
Sunnyvale
Kansas City
IllinoisWave 10Gb
Washington DC
JGN2plus 10Gb
Oak Ridge
Raleigh
CAVEwave 10Gb
Los Angeles
Tulsa
LEAD 10Gb
SLR 10Gb-GLORIAD 1Gb
TeraFlow 10Gb
Pacific Wave
CiscoWave 2*10Gb
National LambdaRail n*10Gb
CiscoWave 10Gb
EnLIGHTened 10Gb
San Diego
Internet2-DCN n*10Gb
Atlanta
ESnet-SDN 10Gb
CENIC 10Gb
Tijuana
LONI 10Gb
Internet2-DCN n*10Gb
WHREN-LILA 2Gb
El Paso
National LambdaRail n*10Gb
Jacksonville
Houston
Baton Rouge
AtlanticWave 10Gb
AMPATH
Miami
WH

Distributed Tracing: What is OpenTelemetry?

- “An observability framework for cloud-native software”
- Incubating Project of Cloud Native Computing Foundation (CNCF)
- Vendor-agnostic Specification of Telemetry Data
- Supports various languages: Java, Go, JavaScript, Python, Rust, Erlang…
- Supported by Industrial Stakeholders
- Open Source
- https://opentelemetry.io/

OpenTelemetry

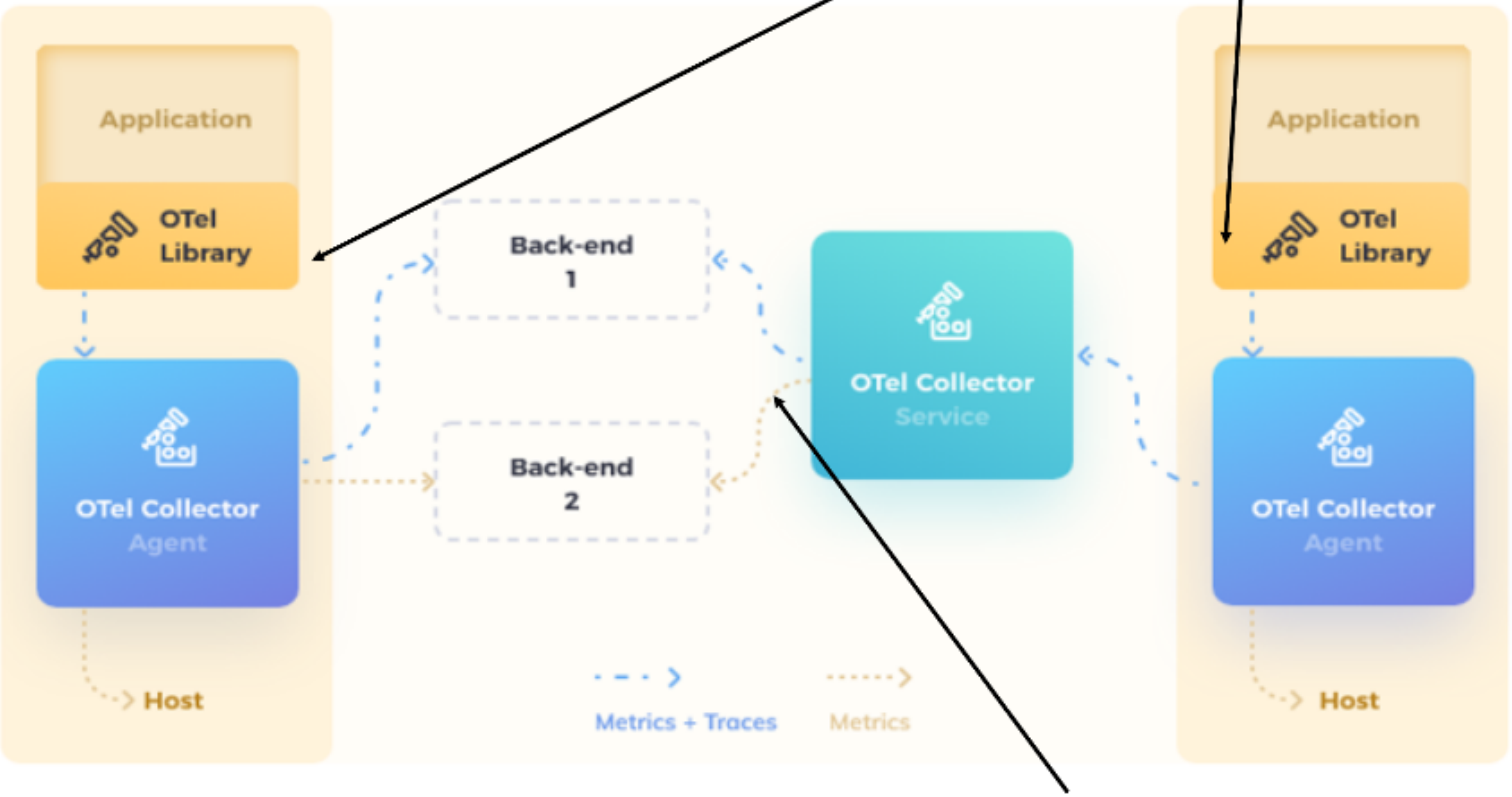
CLOUD NATIVE
COMPUTING FOUNDATION

What is OpenTelemetry?

(In slightly more technical detail)

Telemetry can be sent to Collector, or to Back-end

Instrumentation in application



Telemetry (metrics, logs, traces) pushed automatically

Collector can also send telemetry to Back-ends

https://github.com/fangyi-zhou/TypeScript-Multiparty-Sessions/blob/master/case-studies/TravelAgency/client/src/TravelAgency/A/A.tsx

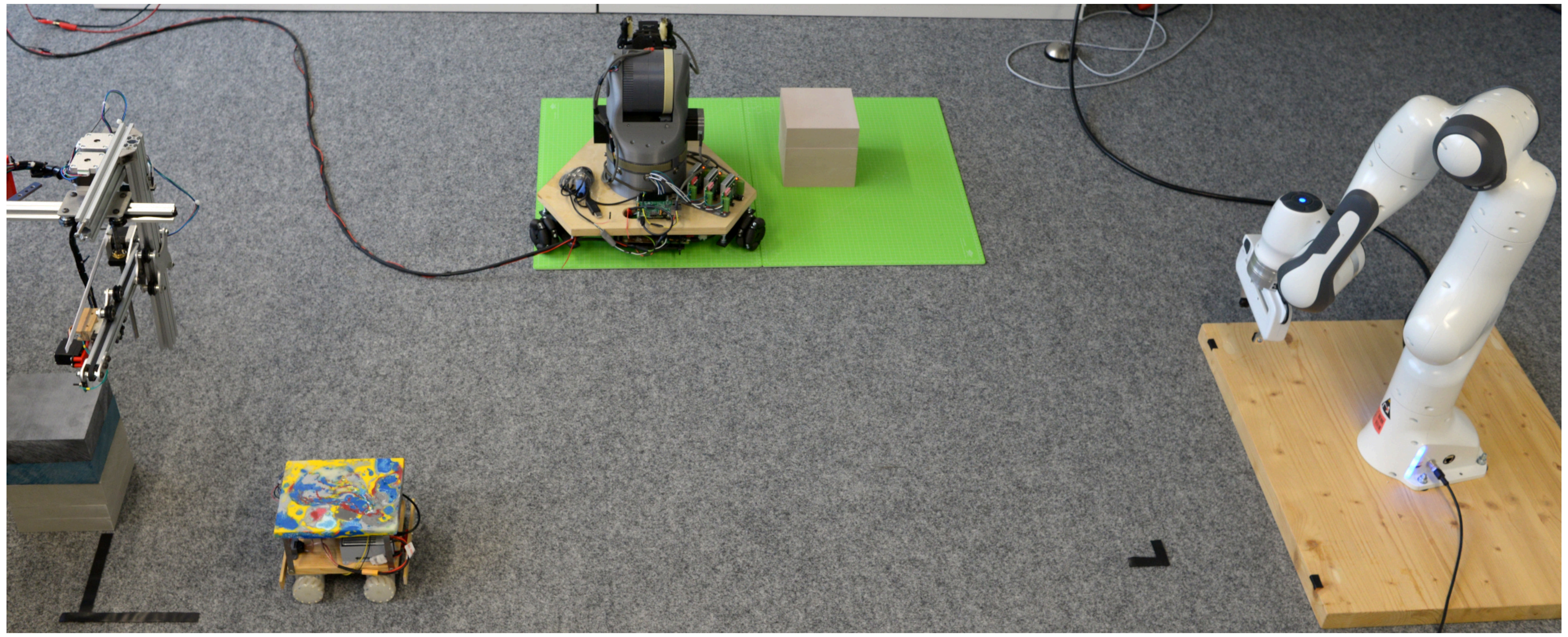
Monitoring a web application

Monitoring a web application

On-Going and Future Projects

- **Go 1.18** (**Generics types** with Google Go Team **[OOPSLA'2020-A]**) Collaborations with Security & Software Engineer Group at Pennsylvania State University
- Cost Analysis **[OOPSLA'2020-B]** applications to programming languages
- Uniform Concurrent Distributed Message-Passing Programming Languages Semantics with **Operational Game Semantics** by using π as the intermediate language **[POPL'2019]**
- Refinement Session Types **[OOPSLA'2020-C]** for Rust and Typescript
- Unreliable Session Types
  - Model-Checking & Scala
- Cyber Physical Systems

[ECOOP'19, OOPSLA'20-D]

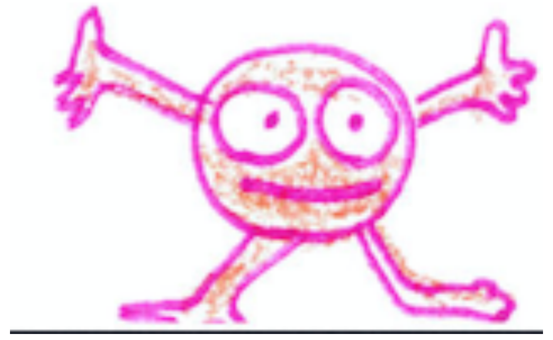


Department of Computing Imperial College London

Dalal

Department of Computing Imperial College London

Dalal



Thank you! Questions?

http://mrg.doc.ic.ac.uk/