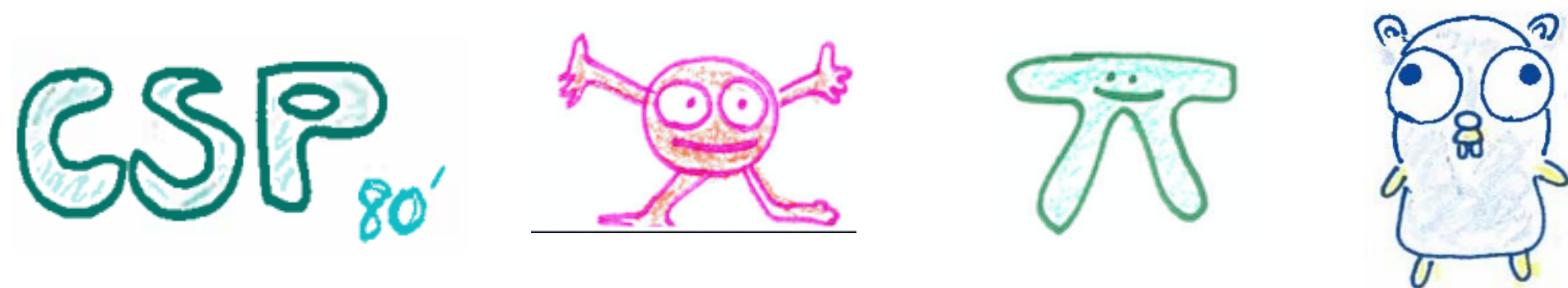


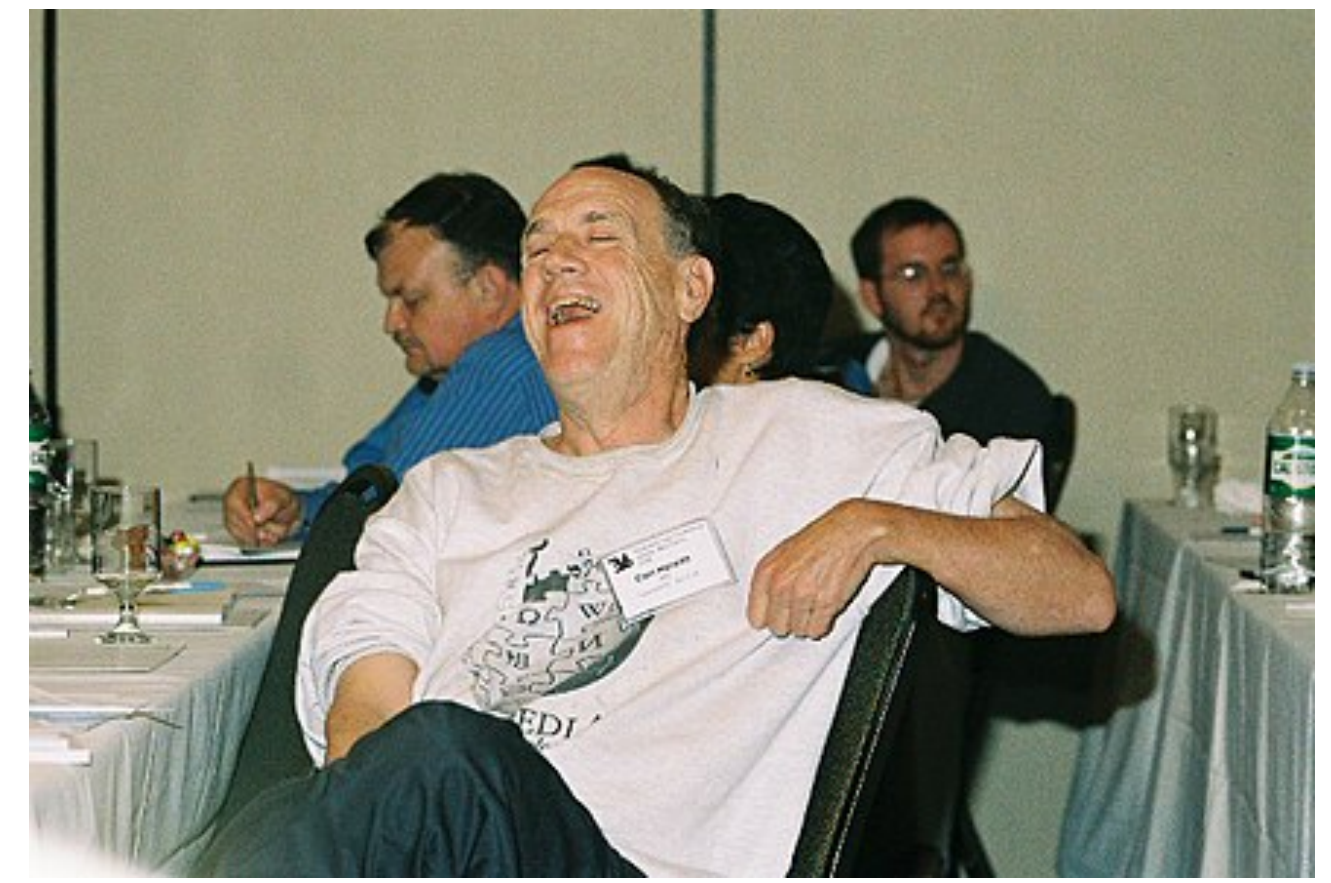
AIMS: Mini-Project Presentation

- **Automatic App Navigation** (with Amazon Web Services) conducted by Amazon Prime Video Team at London
- **Rust Programming** for Actor Languages (AcTyx AG (<https://www.actyx.com/>))
- **Verified Multi-Agent Programming with Actor Models** (Model Checking of Go) www.mrg.cs.ox.ac.uk



Nobuko Yoshida (nobuko.yoshida@cs.ox.ac.uk)

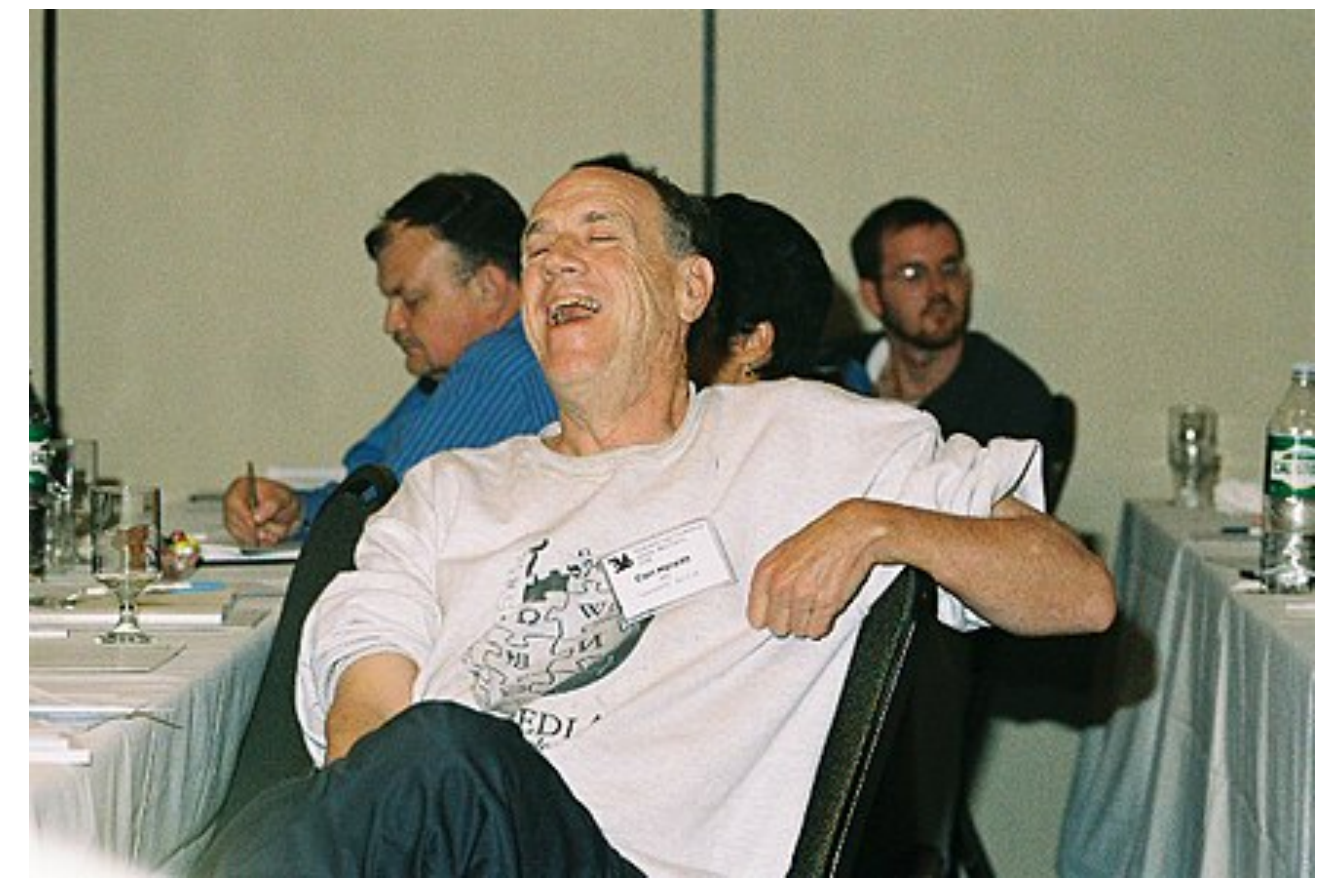
Actor Models



1944-2020

Actor Models

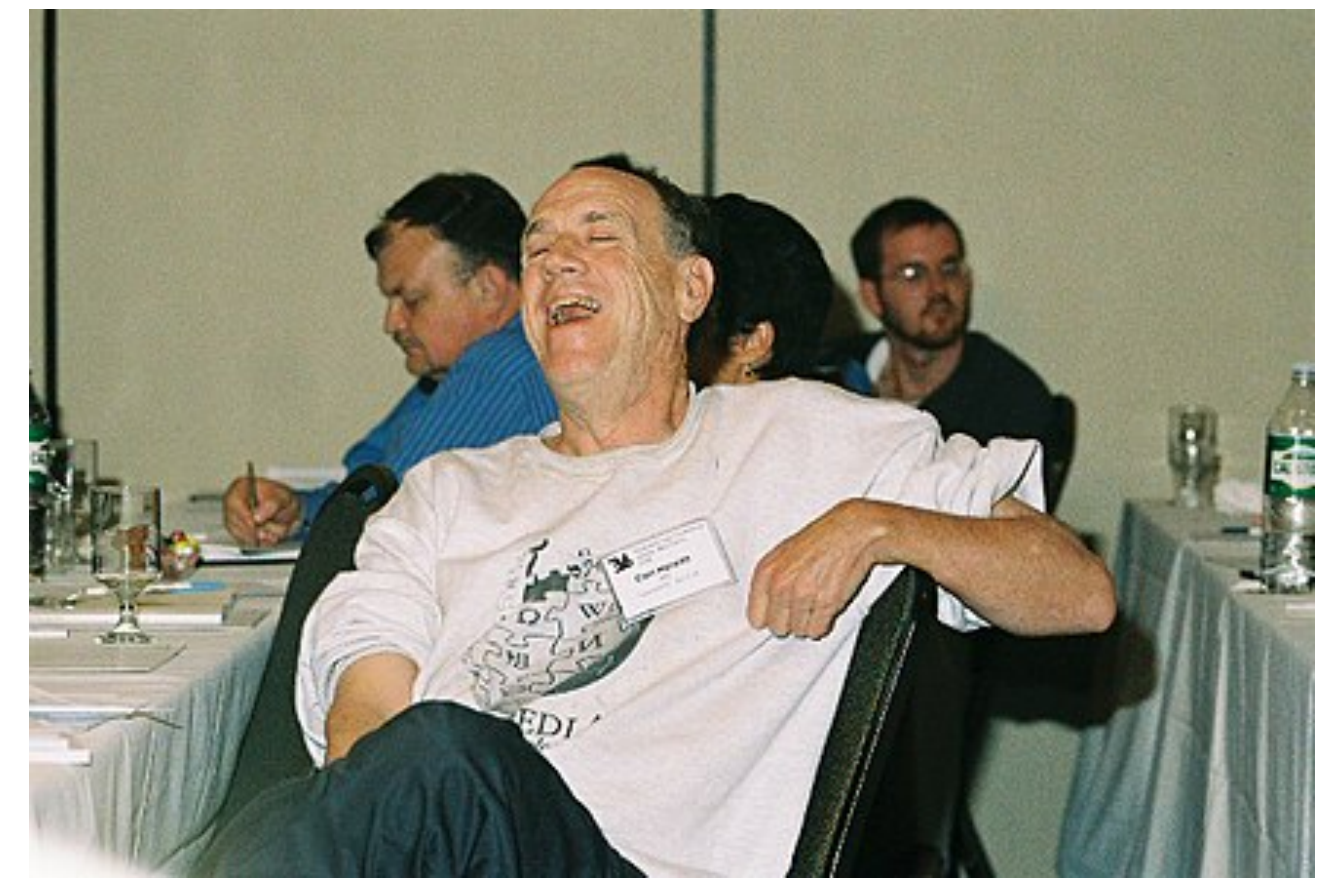
- **Carl Hewitt (MIT)**



1944-2020

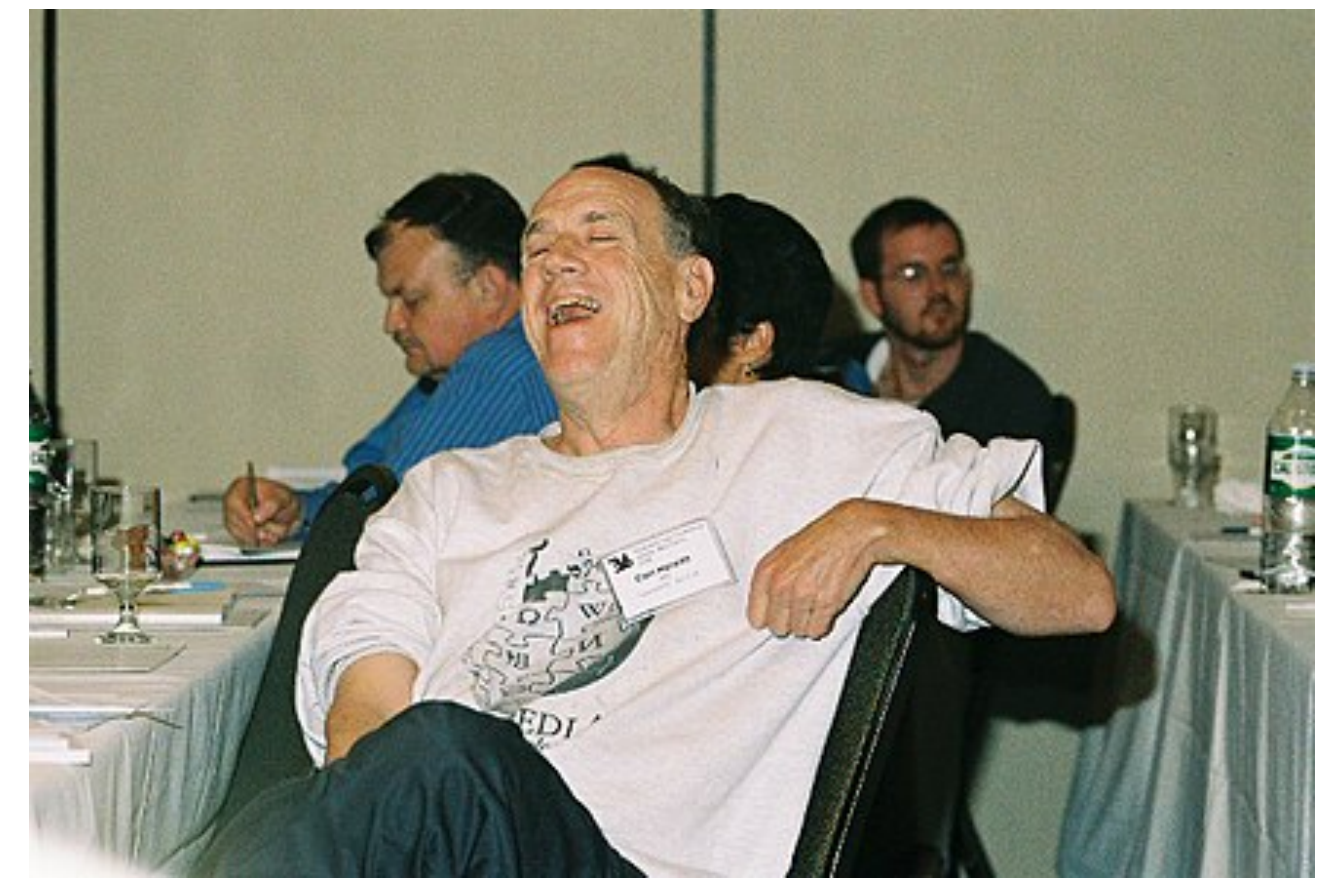
Actor Models

- Carl Hewitt (MIT)
- **Mathematical Models** for Concurrent Computations



1944-2020

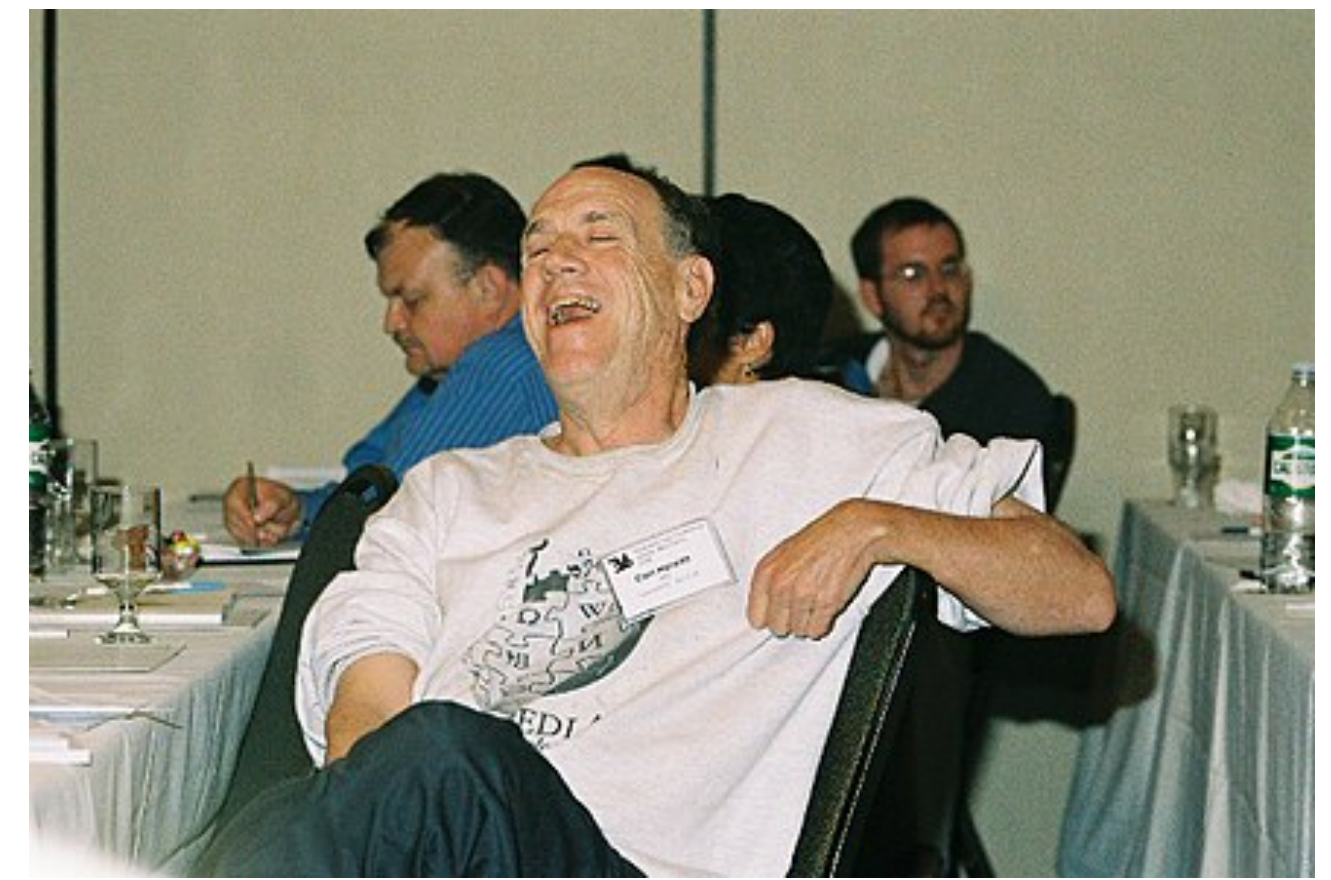
Actor Models



1944-2020

- Carl Hewitt (MIT)
- Mathematical Models for Concurrent Computations
- **Asynchronous Message Passings among Concurrent and Distributed Independent Agents**

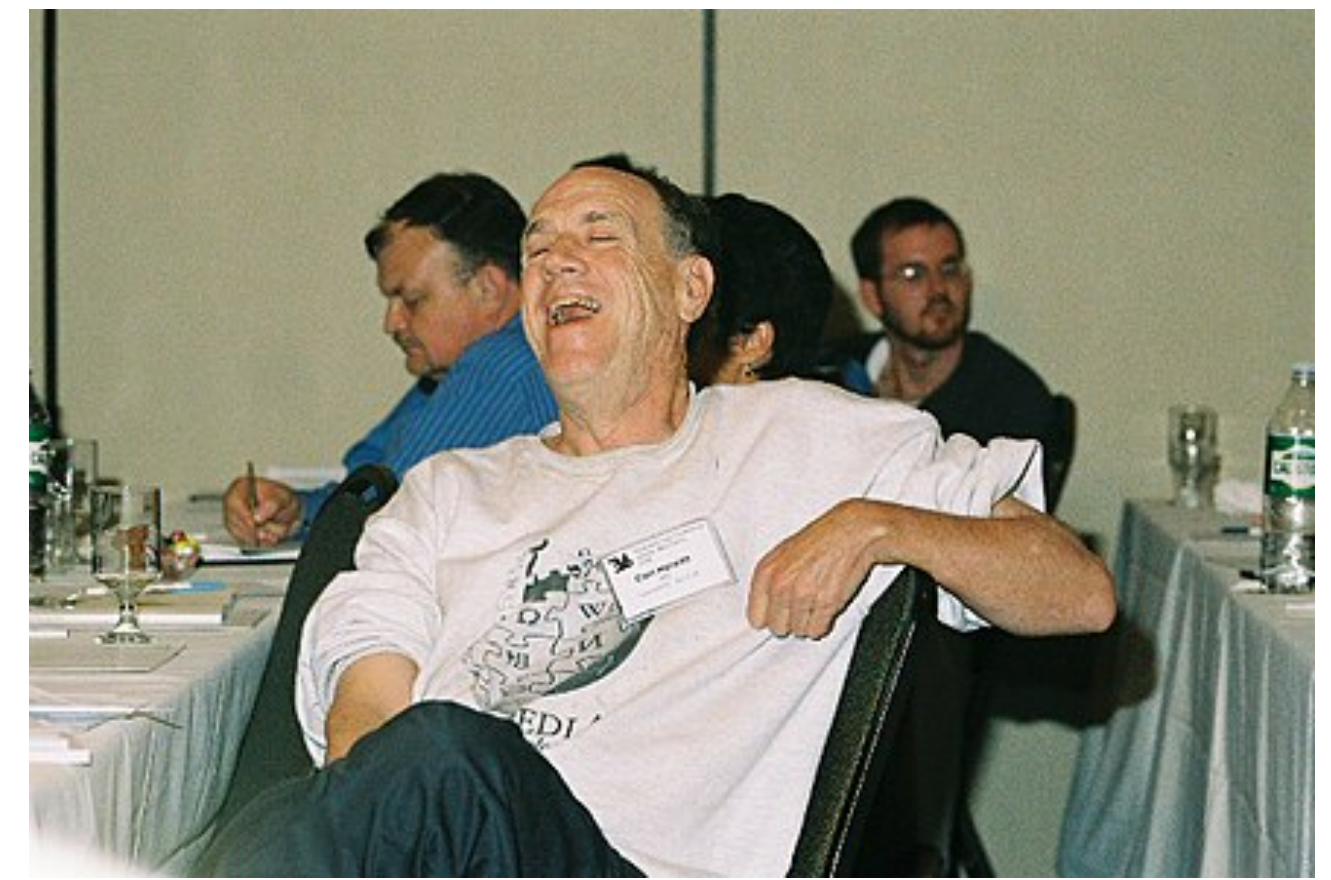
Actor Models



1944-2020

- Carl Hewitt (MIT)
- Mathematical Models for Concurrent Computations
- Asynchronous Message Passings among Concurrent and Distributed Independent Agents
- Concurrent Object Oriented Languages

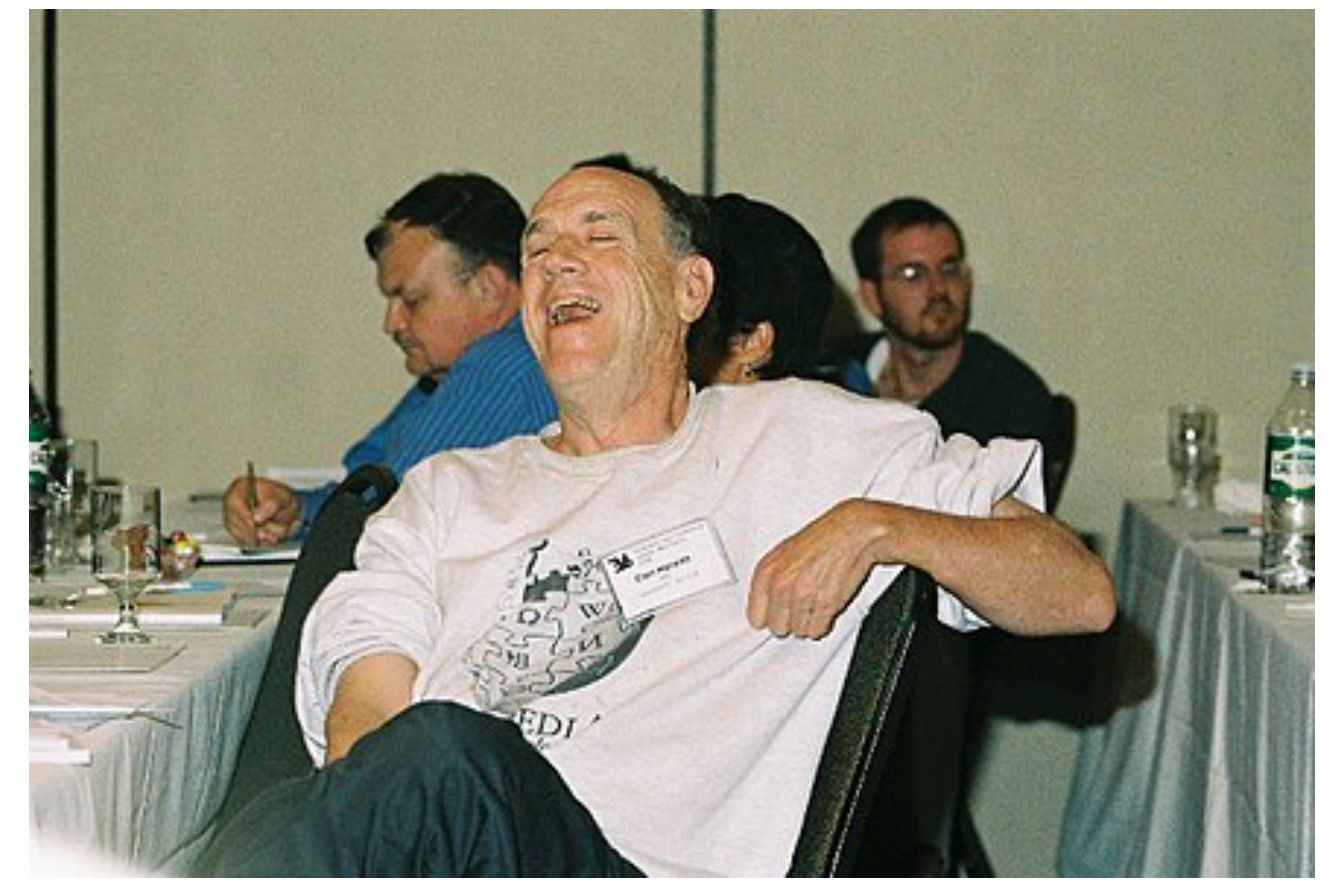
Actor Models



1944-2020

- Carl Hewitt (MIT)
- Mathematical Models for Concurrent Computations
- Asynchronous Message Passings among Concurrent and Distributed Independent Agents
 - Concurrent Object Oriented Languages
 - Multi Agent Systems

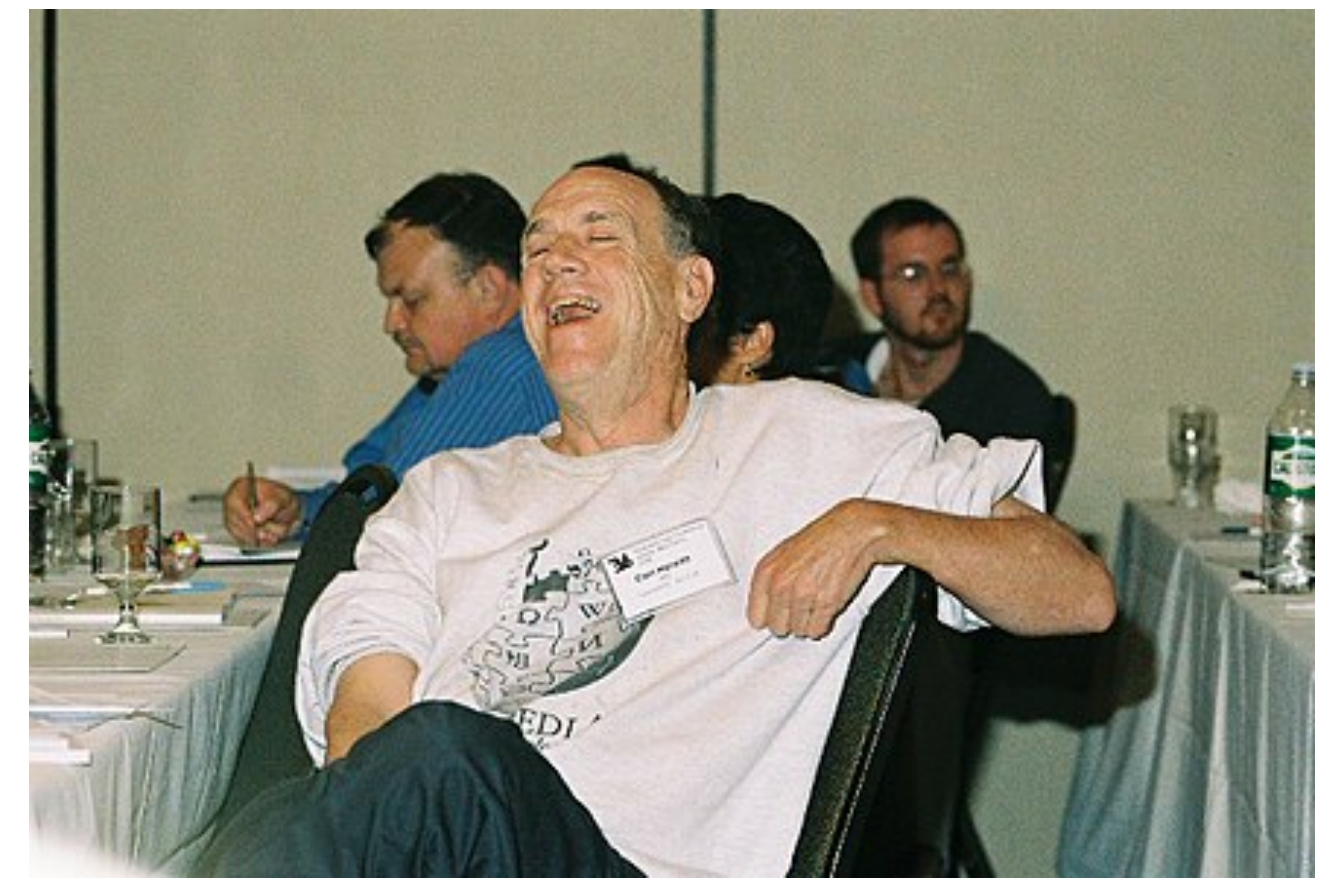
Actor Models



1944-2020

- Carl Hewitt (MIT)
- Mathematical Models for Concurrent Computations
- Asynchronous Message Passings among Concurrent and Distributed Independent Agents
 - Concurrent Object Oriented Languages
 - Multi Agent Systems
 - Web Services and Simple Object Access **Protocols** (SOAP)

Actor Models



1944-2020

- Carl Hewitt (MIT)
- Mathematical Models for Concurrent Computations
- Asynchronous Message Passings among Concurrent and Distributed Independent Agents
 - Concurrent Object Oriented Languages
 - Multi Agent Systems
 - Web Services and Simple Object Access **Protocols** (SOAP)
 - Mathematical Models (Logics and Process Algebra)

Communications are Ubiquitous

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



microservices



Problems: Ambiguity

- Protocol descriptions are **ambiguous**
- **SMTP: simple mail transfer protocol**
 - They are written in English, often very long



RFC 821

August 1982
Simple Mail Transfer Protocol

TABLE OF CONTENTS

1.	INTRODUCTION	1
2.	THE SMTP MODEL	2
3.	THE SMTP PROCEDURE	4
3.1.	Mail	4
3.2.	Forwarding	7
3.3.	Verifying and Expanding	8
3.4.	Sending and Mailing	11
3.5.	Opening and Closing	13
3.6.	Relaying	14
3.7.	Domains	17
3.8.	Changing Roles	18
4.	THE SMTP SPECIFICATIONS	19
4.1.	SMTP Commands	19
4.1.1.	Command Semantics	19
4.1.2.	Command Syntax	27
4.2.	SMTP Replies	34
4.2.1.	Reply Codes by Function Group	35
4.2.2.	Reply Codes in Numeric Order	36
4.3.	Sequencing of Commands and Replies	37
4.4.	State Diagrams	39
4.5.	Details	41
4.5.1.	Minimum Implementation	41
4.5.2.	Transparency	41
4.5.3.	Sizes	42

Problems: Ambiguity

- Protocol descriptions are **ambiguous**
- **SMTP: simple mail transfer protocol**
 - They are written in English, often very long



3.1. MAIL

There are three steps to SMTP mail transactions. The transaction is started with a MAIL command which gives the sender identification. A series of one or more RCPT commands follows giving the receiver information. Then a DATA command gives the mail data. And finally, the end of mail data indicator confirms the transaction.

The first step in the procedure is the MAIL command. The <reverse-path> contains the source mailbox.

```
MAIL <SP> FROM:<reverse-path> <CRLF>
```

This command tells the SMTP-receiver that a new mail transaction is starting and to reset all its state tables and buffers, including any recipients or mail data. It gives the reverse-path which can be used to report errors. If accepted, the receiver-SMTP returns a 250 OK reply.

The <reverse-path> can contain more than just a mailbox. The <reverse-path> is a reverse source routing list of hosts and source mailbox. The first host in the <reverse-path> should be the host sending this command.

The second step in the procedure is the RCPT command.

```
RCPT <SP> TO:<forward-path> <CRLF>
```

This command gives a forward-path identifying one recipient. If accepted, the receiver-SMTP returns a 250 OK reply, and stores the forward-path. If the recipient is unknown the receiver-SMTP returns a 550 Failure reply. This second step of the procedure can be repeated any number of times.

Problems: Concurrency Bugs

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

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]

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**.

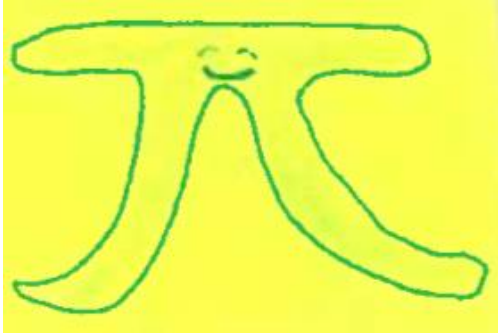


Why Session Types, Why Now?

Significant academic and industry interests via fundamental breakthroughs

Binary Session Types

ESOP'98



Joined W3C Standardization

2002



Multiparty Session Types

POPL'08



A collection of logos for various programming languages and frameworks, including Java, TypeScript, Scala, akka, Erlang, MPI, and OCaml, arranged horizontally below the Multiparty Session Types box.

What is a concept of *Types* in Programming?

- Types can avoid runtime error

Boolean, Natural Number

100 x 200

100 x true

What is a concept of *Types* in Programming?

- Types can avoid runtime error

Boolean, Natural Number (Data Types)

100 x 200 **Correct**

100 x true **Wrong**

Types for *Protocol*?

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

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

Ring Protocol

Example

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

Ring Protocol

Example

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

Ring Protocol

Example

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

Ring Protocol

Example

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

Ring Protocol

Example

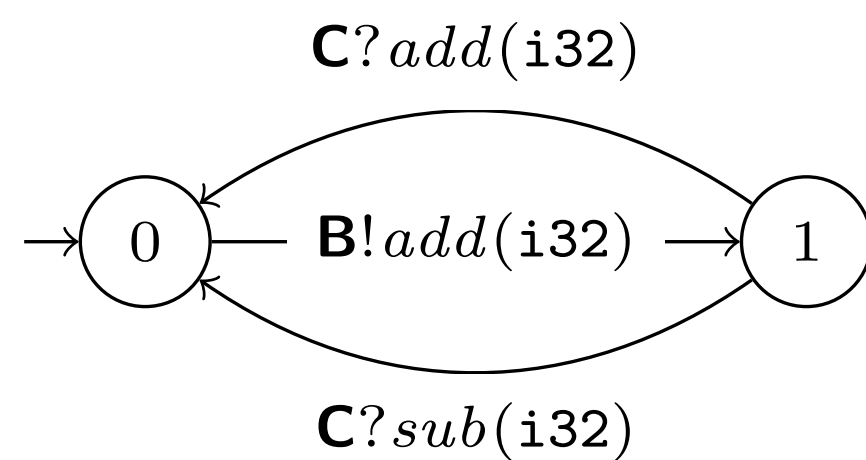
$$G = \mu t. \mathbf{A} \rightarrow \mathbf{B} : \left\{ \begin{array}{l} \mathit{add}(\mathit{i32}).\mathbf{B} \rightarrow \mathbf{C} : \left\{ \begin{array}{l} \mathit{add}(\mathit{i32}).\mathbf{C} \rightarrow \mathbf{A} : \{ \mathit{add}(\mathit{i32}).\mathbf{t} \} \\ \mathit{sub}(\mathit{i32}).\mathbf{C}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

Ring Protocol

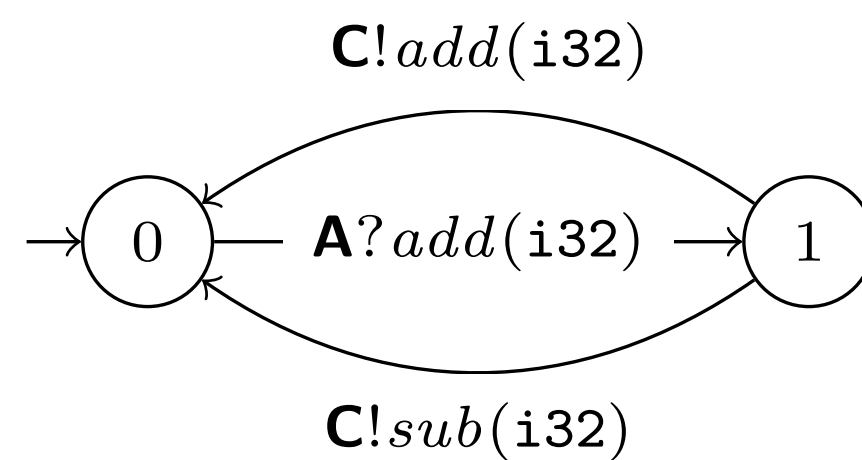
Example

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

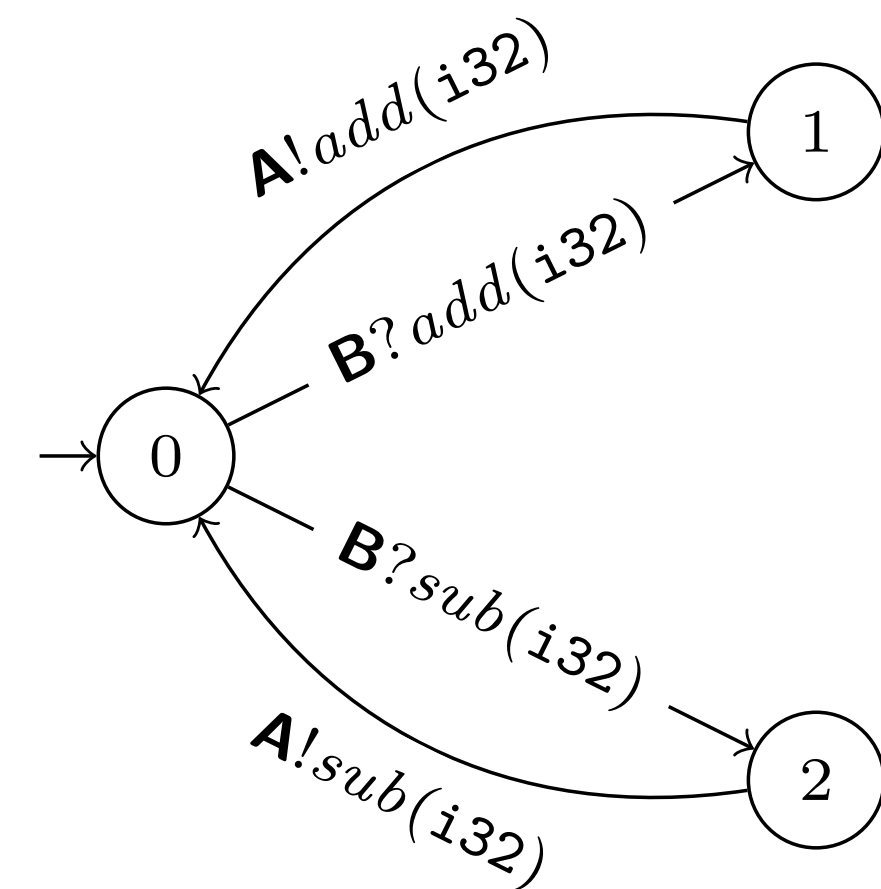
PROJECTION



PROJECTION



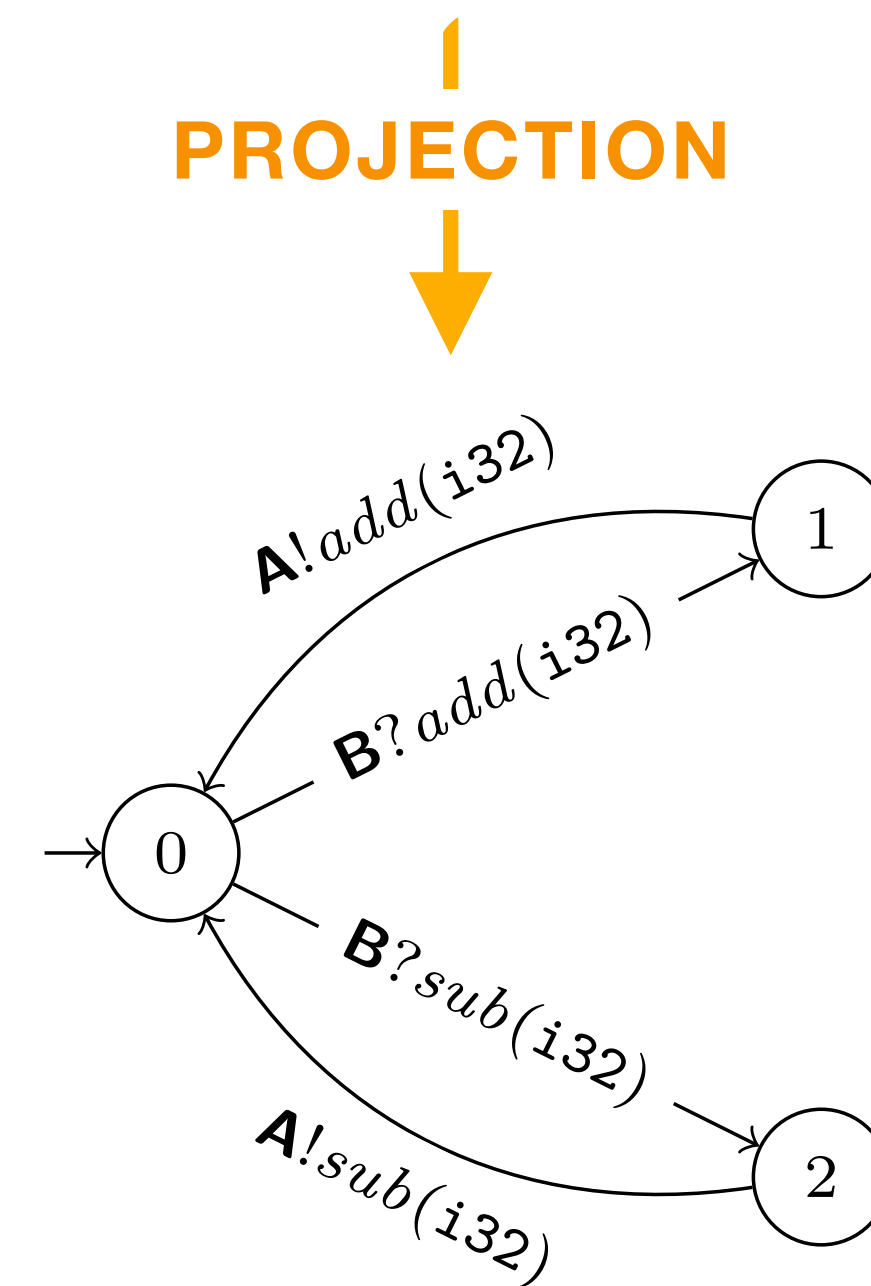
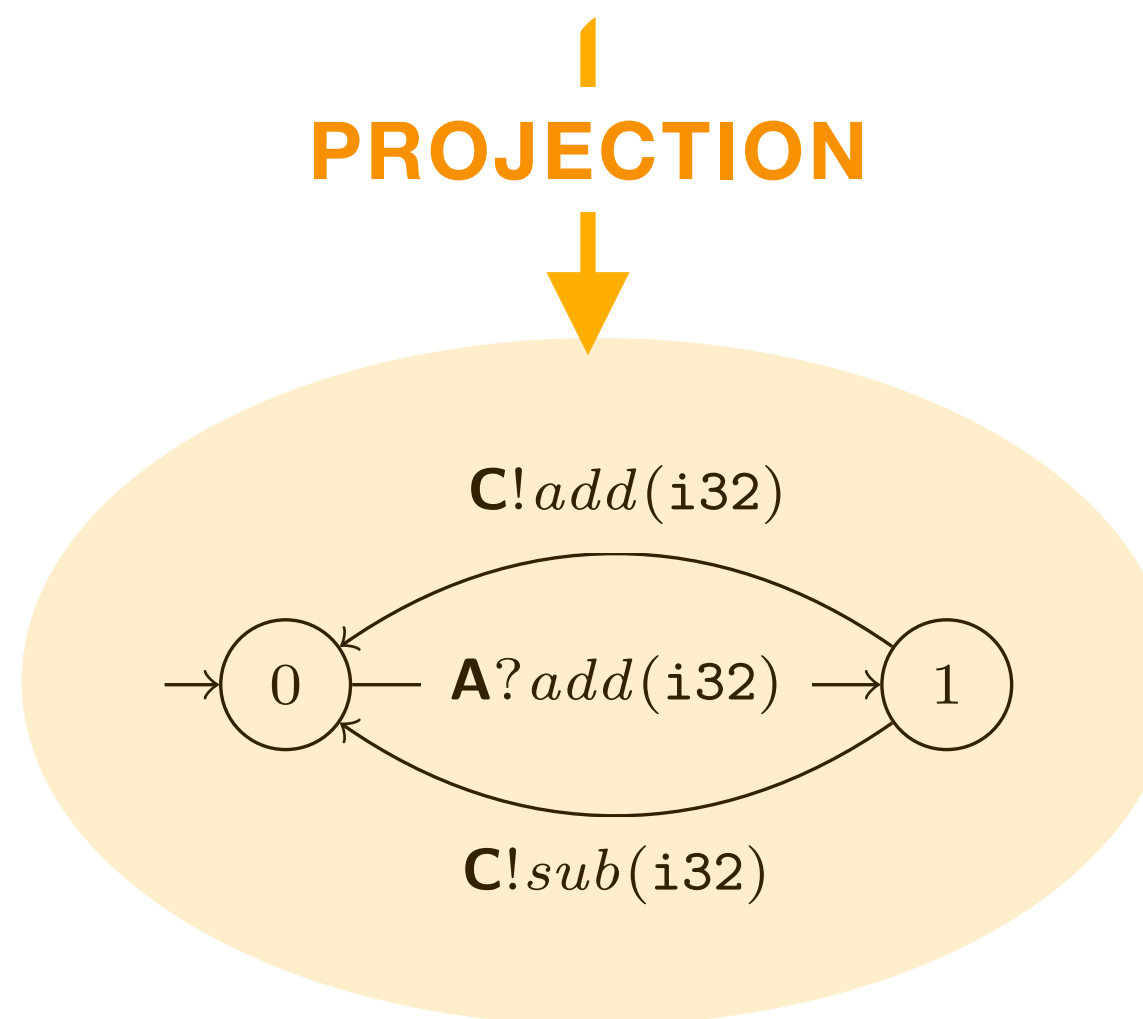
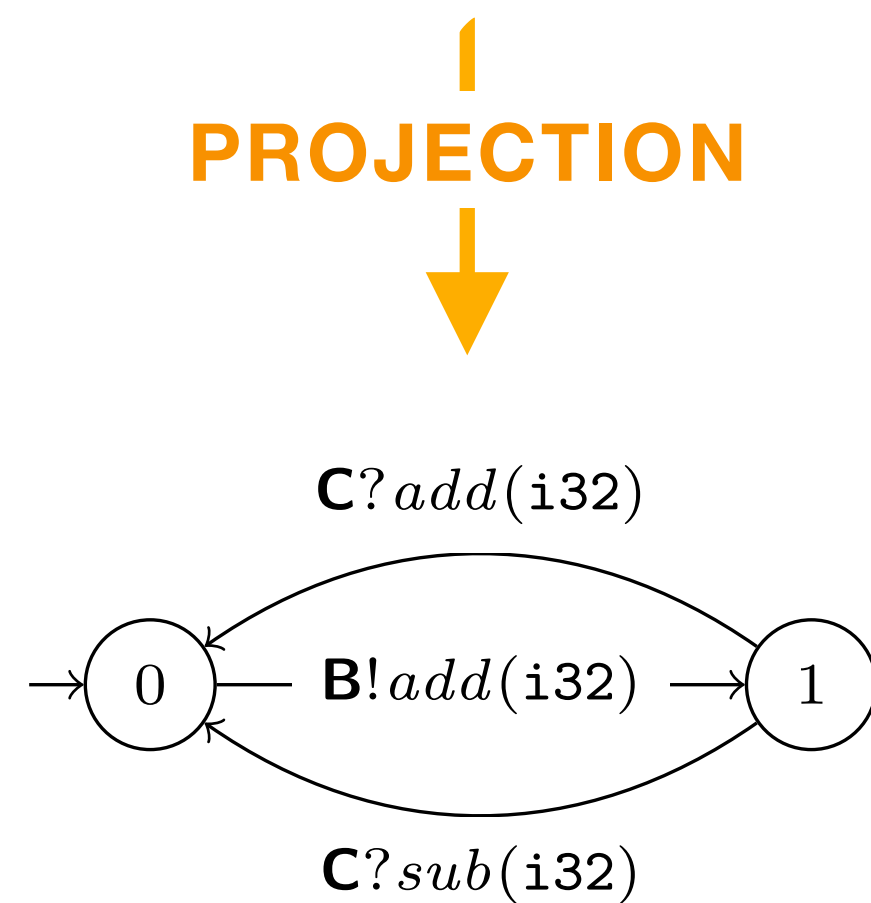
PROJECTION



Ring Protocol

Example

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



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>

The screenshot displays the vScr live web interface. The browser address bar shows <https://nuscr.github.io/nuscr/>. The page features a navigation bar with 'vScr', 'Documentation', and 'GitHub' links. The main content is divided into two sections: 'Global protocol' and 'Local types'.

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]

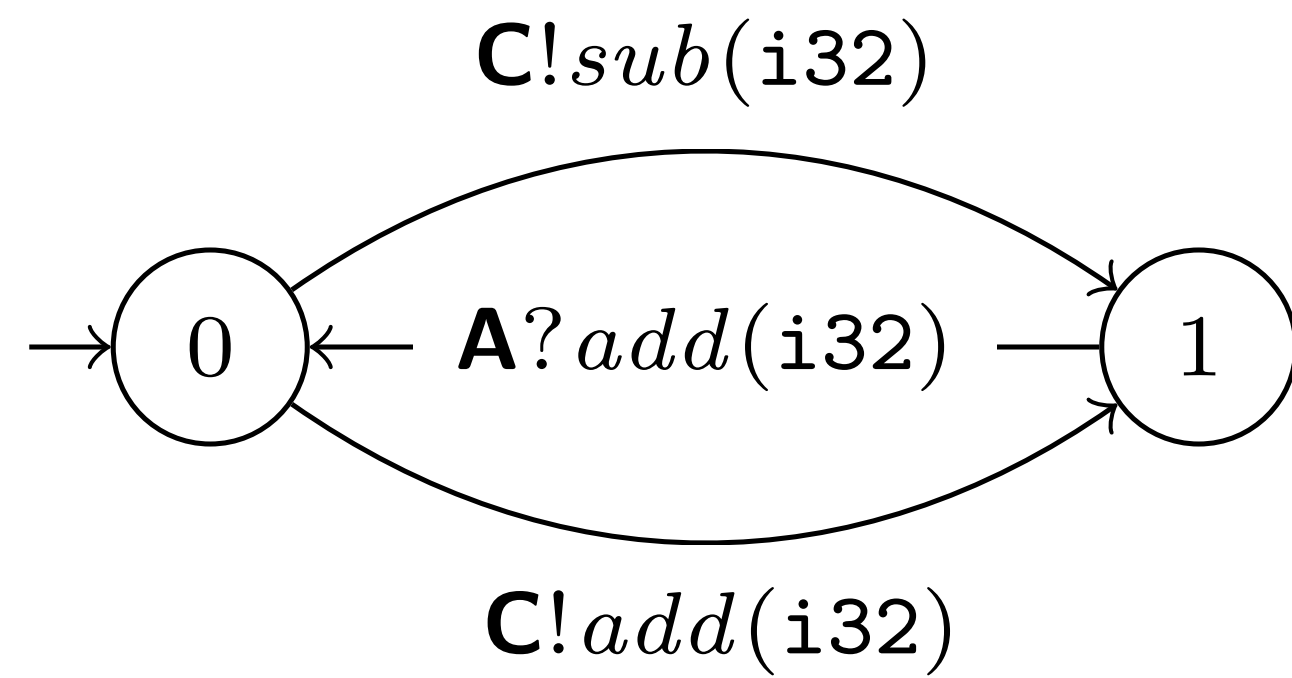
The local types section includes a state transition diagram with 8 states (1-8) and transitions labeled with session types:

- State 1 to State 2: S!HELLO(u: int)
- State 2 to State 7: S!BYE()
- State 2 to State 4: S!ADD(w: int)
- State 4 to State 5: S!ADD(v: int)
- State 5 to State 1: S?RES(f: int)
- State 7 to State 8: S?BYE()

At the bottom of the interface, there is a 'Load an example' dropdown menu and an 'Analyse' button.

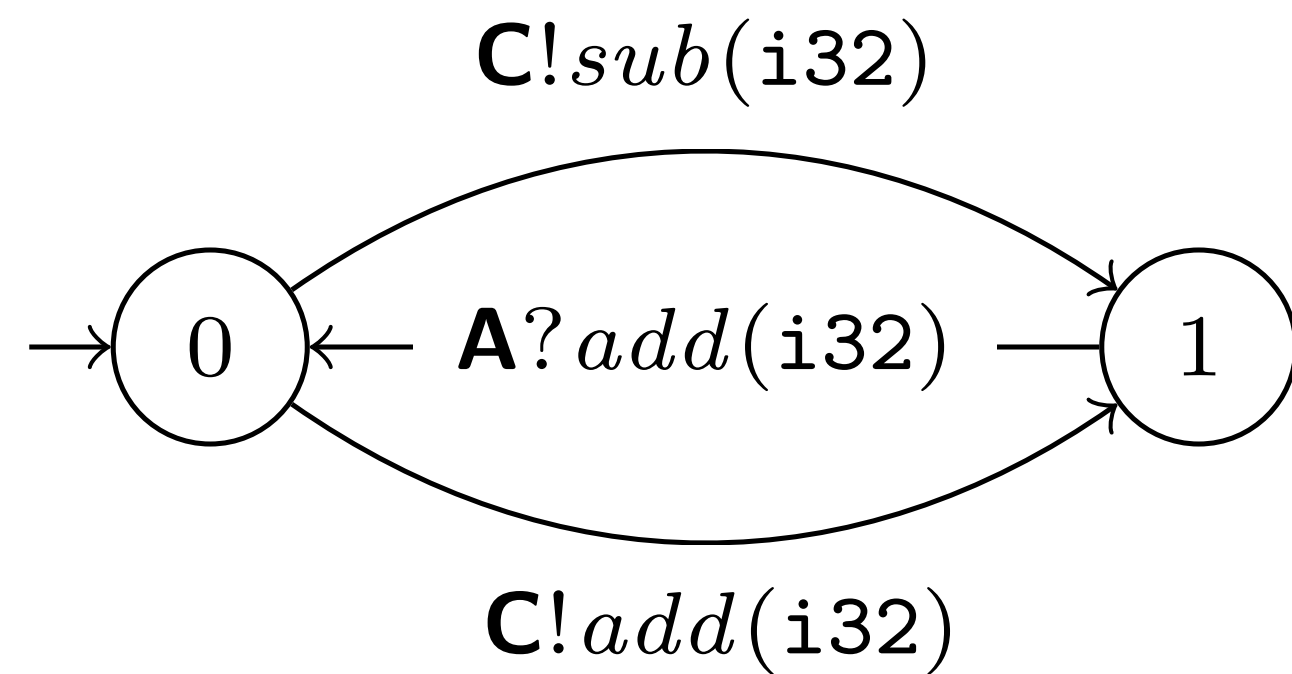
Ring Protocol

Rust API



Ring Protocol

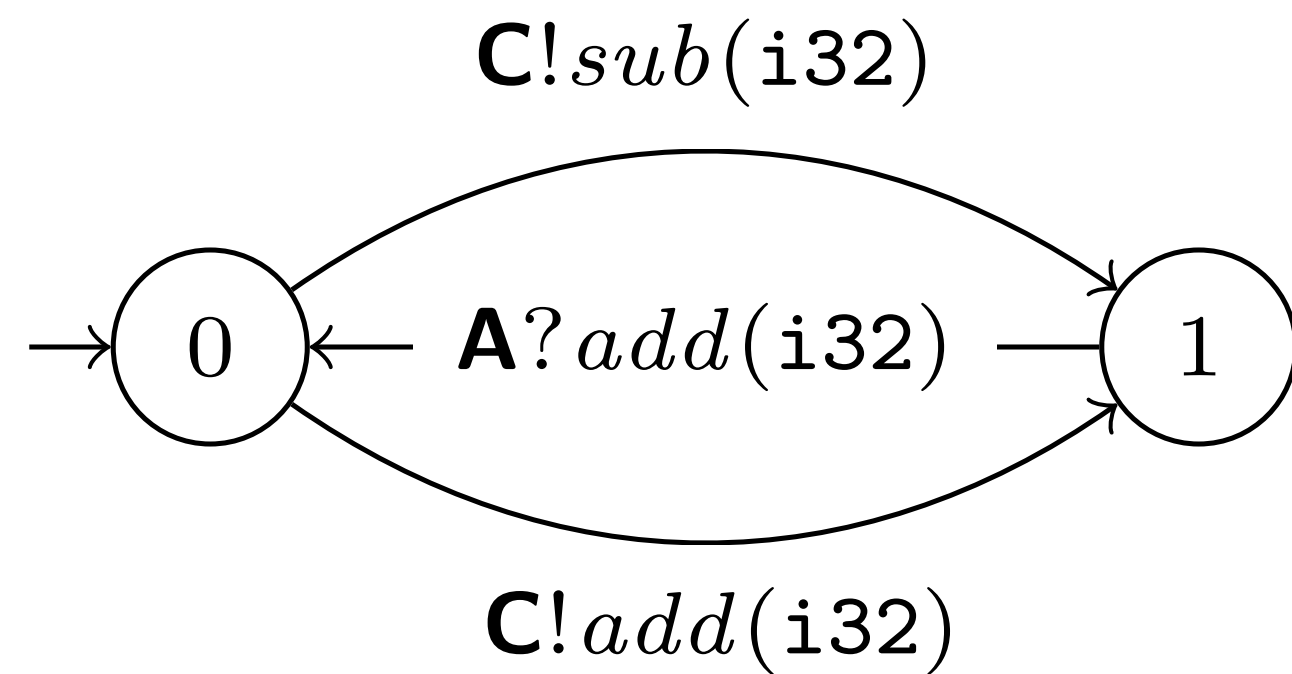
Rust API



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

Ring Protocol

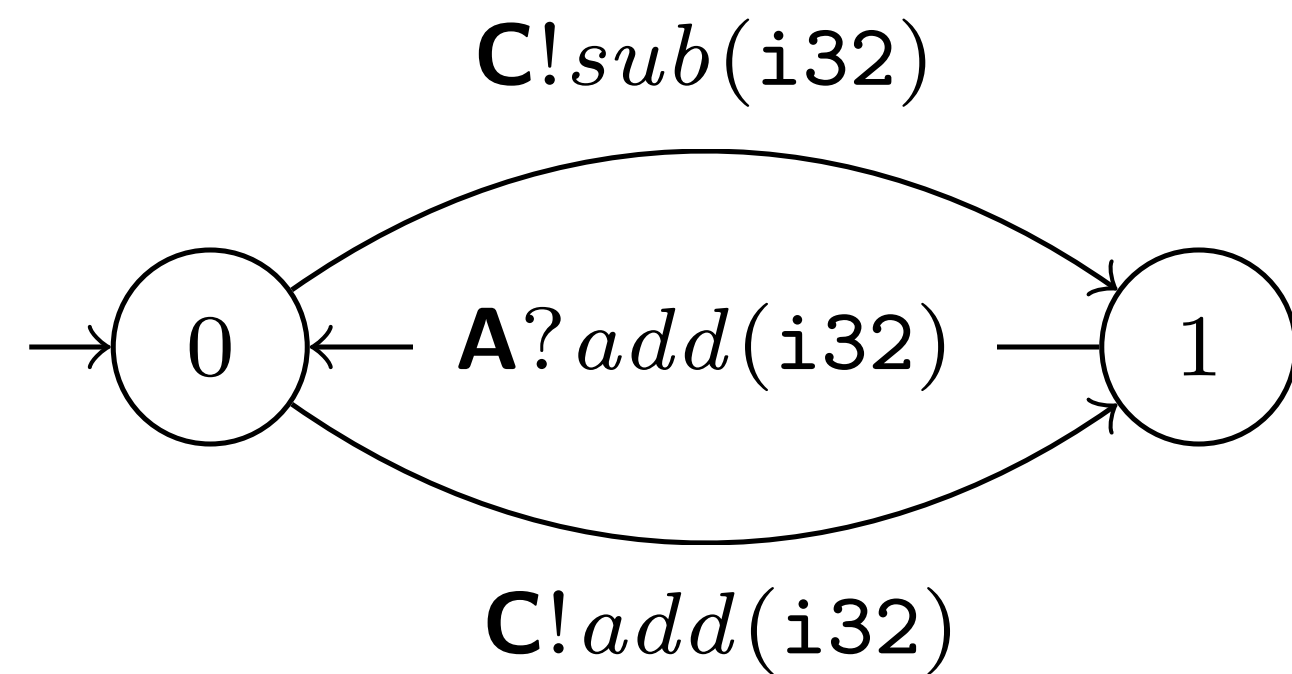
Rust API



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

Ring Protocol

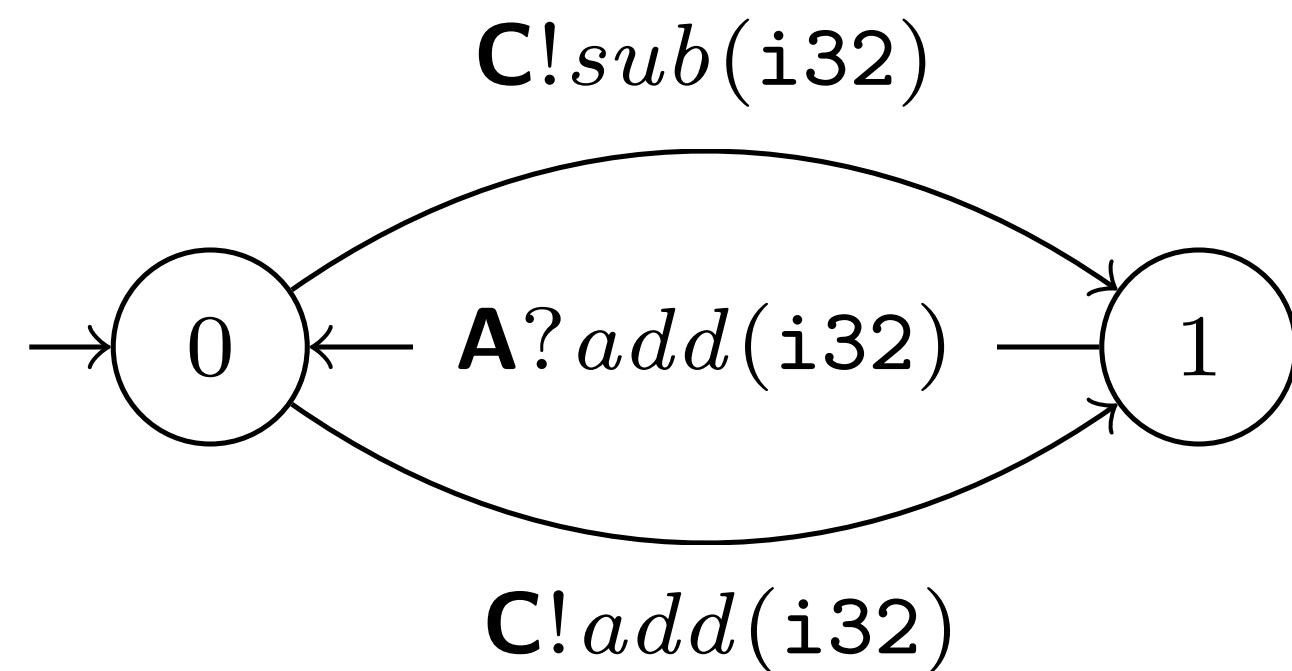
Rust API



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

Ring Protocol

Rust API



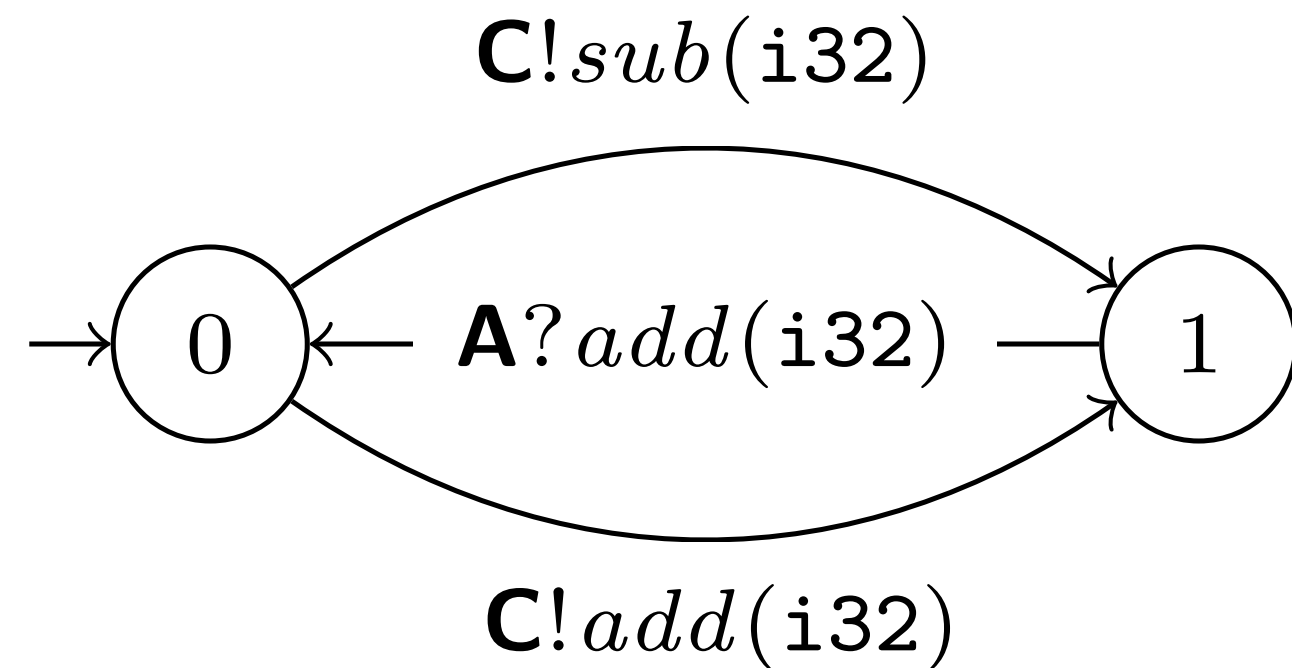
```
#[derive(Role)]  
#[message(Label)]  
struct B(#[route(A)] Receiver, #[route(C)] Sender);
```

```
#[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),
}

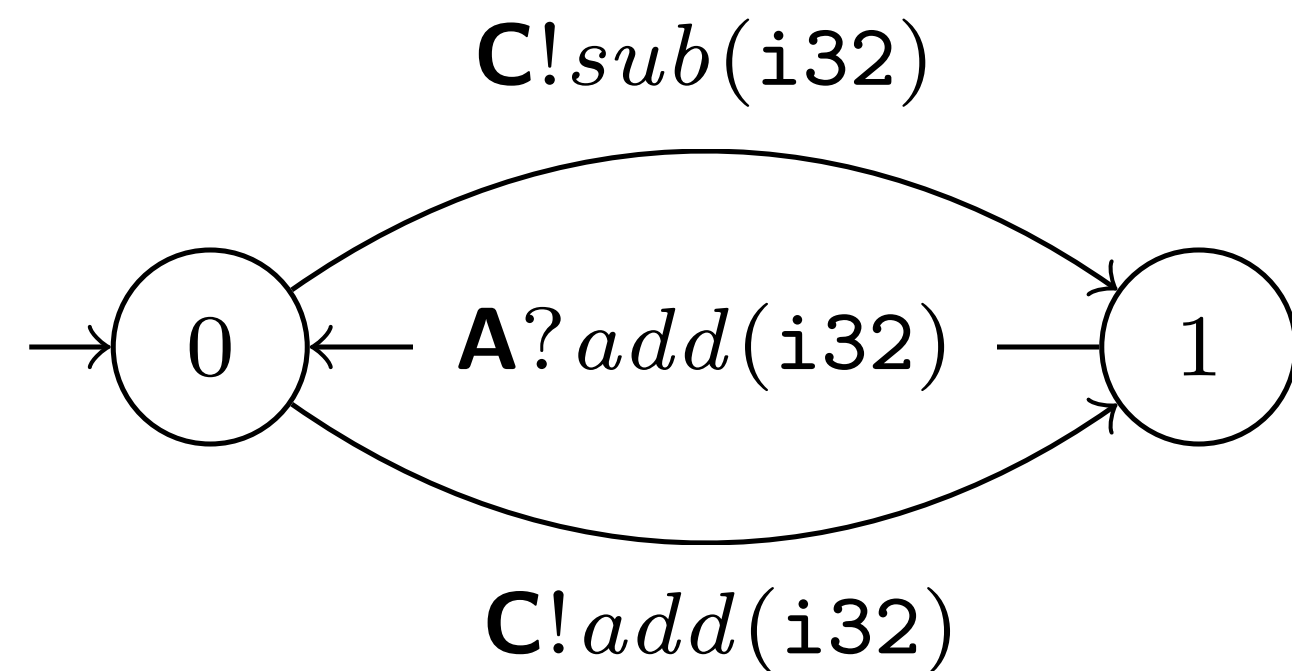
struct Add(i32);
struct Sub(i32);

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

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

Ring Protocol

Implementation

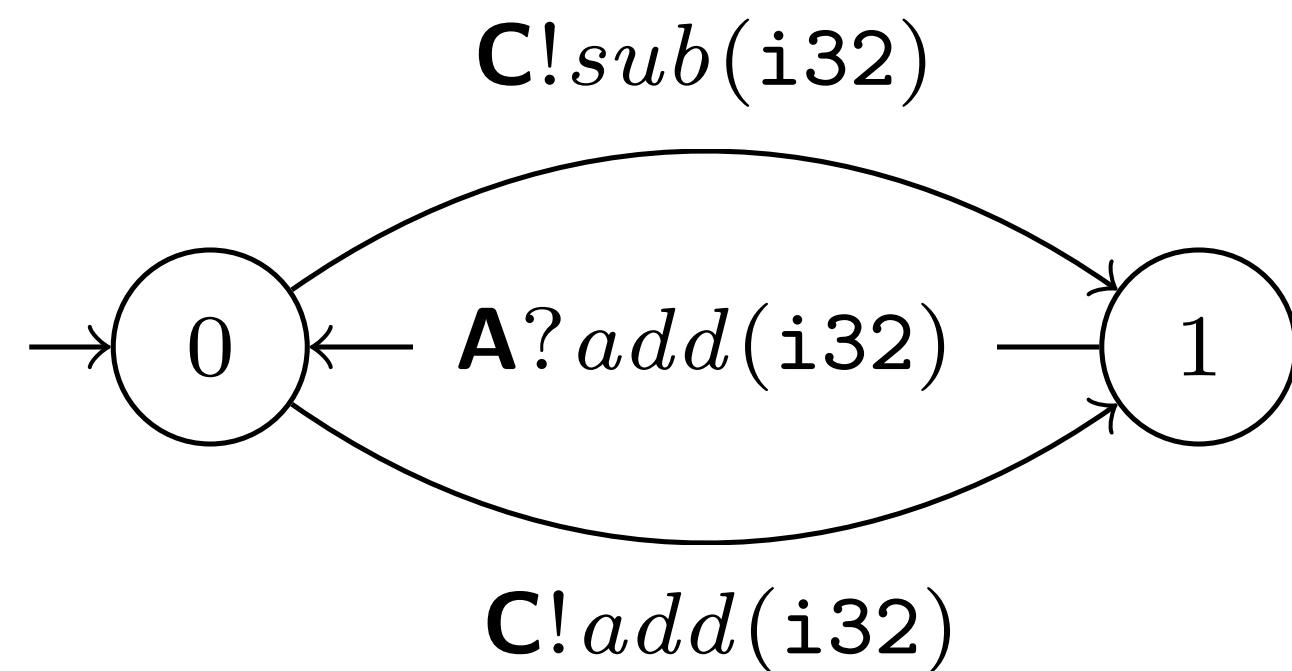


```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            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

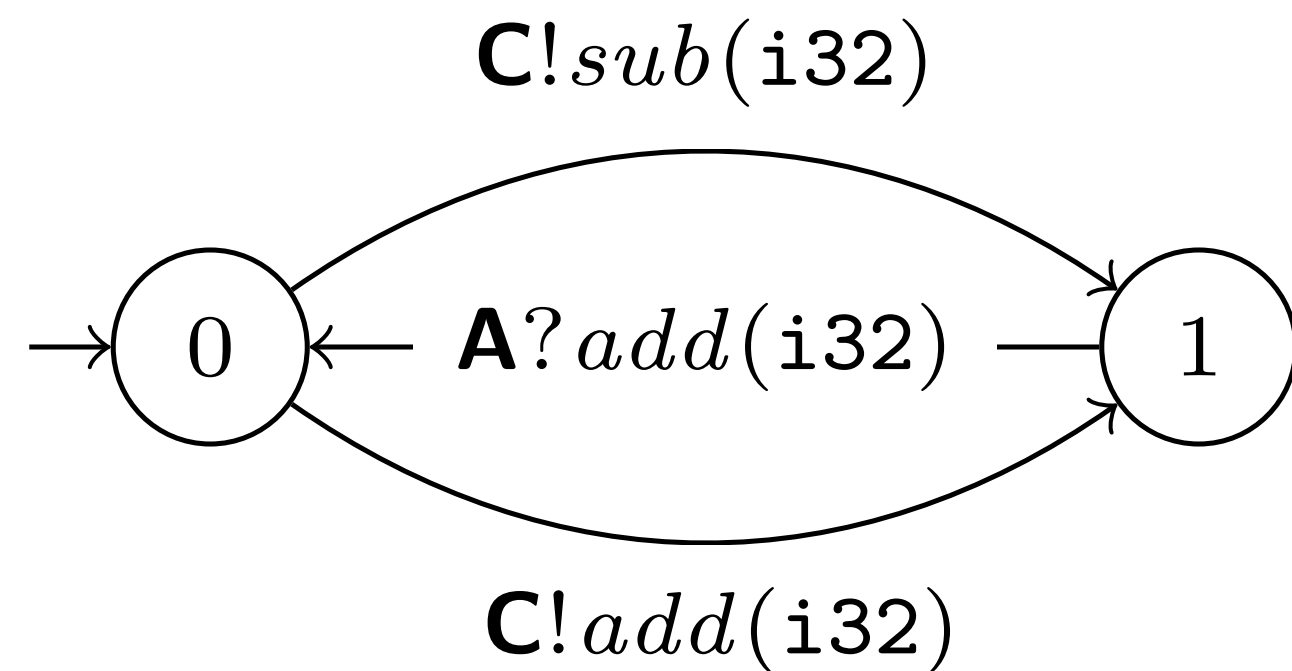


```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            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

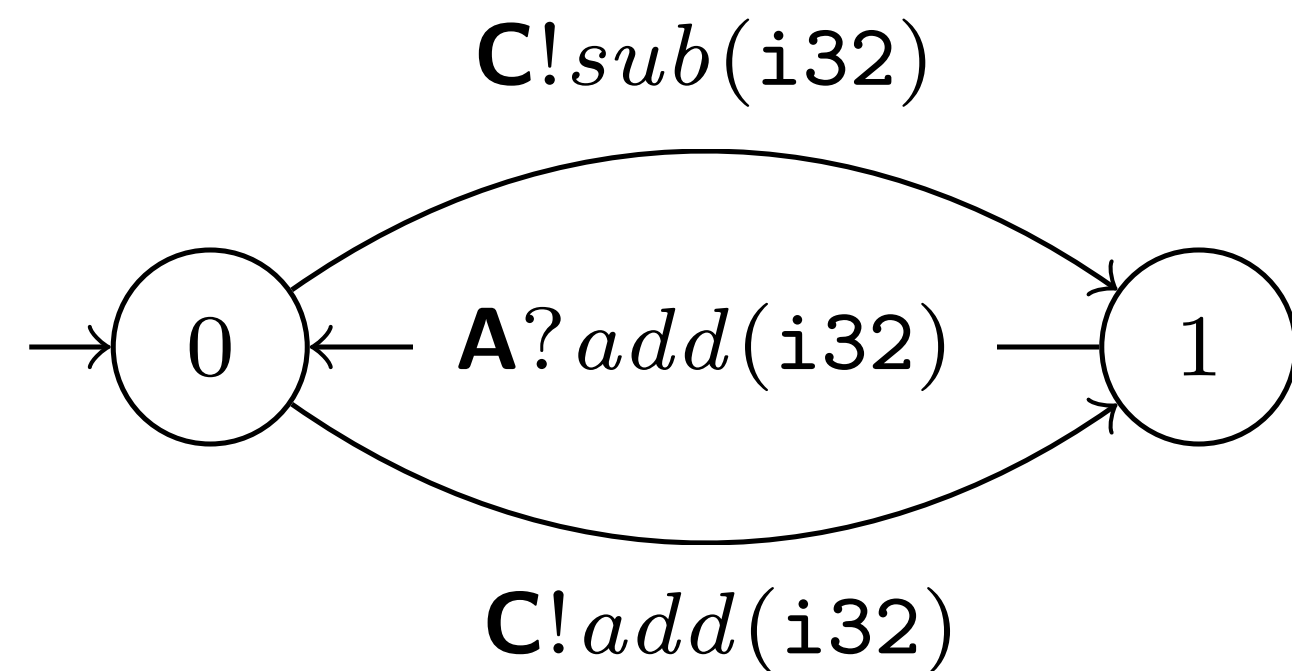


```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            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

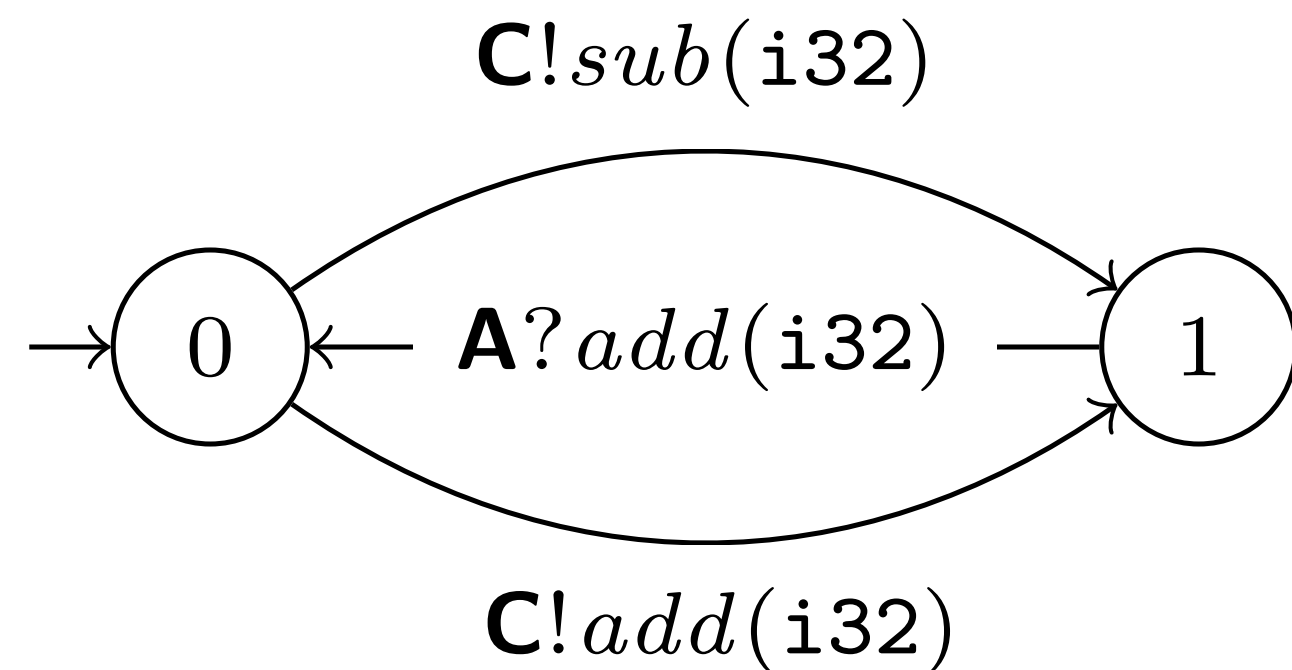


```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            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

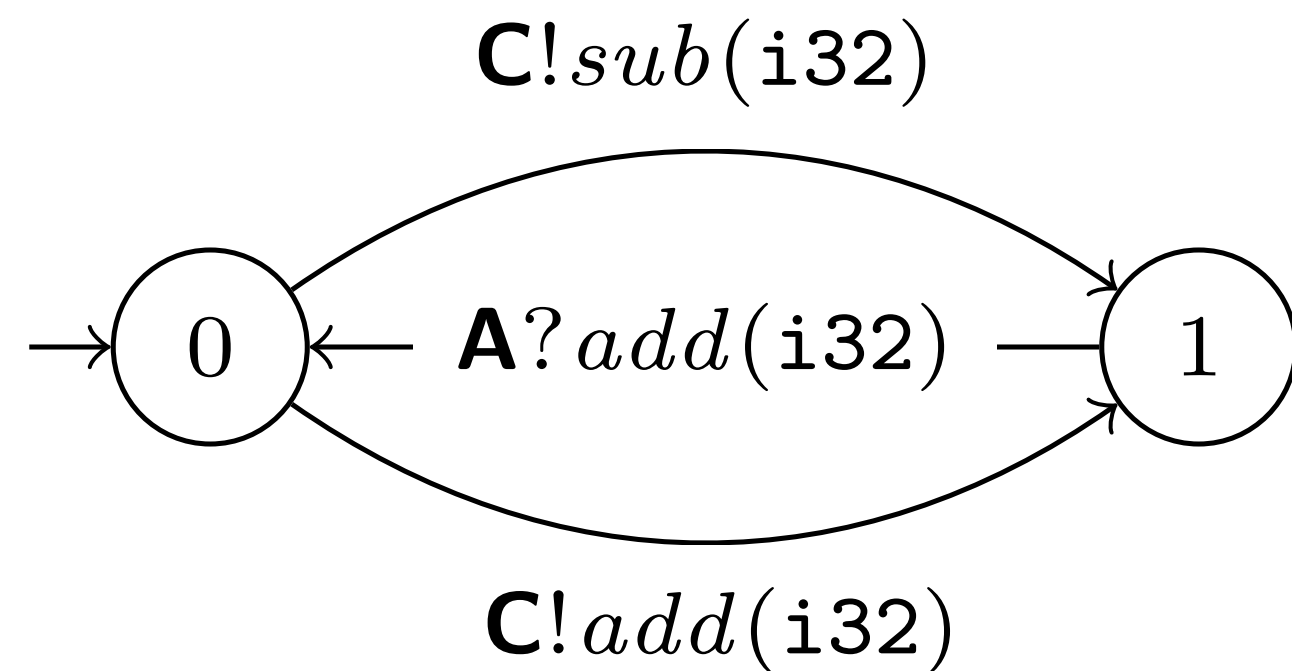


```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            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

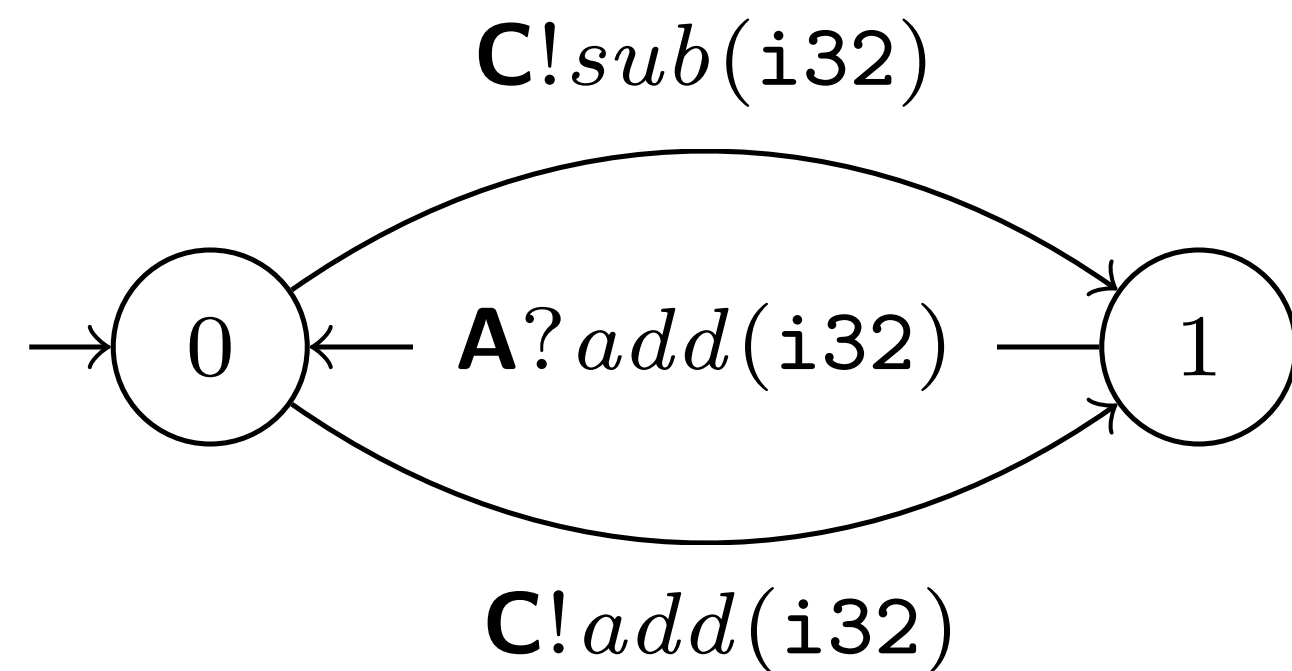


```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            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

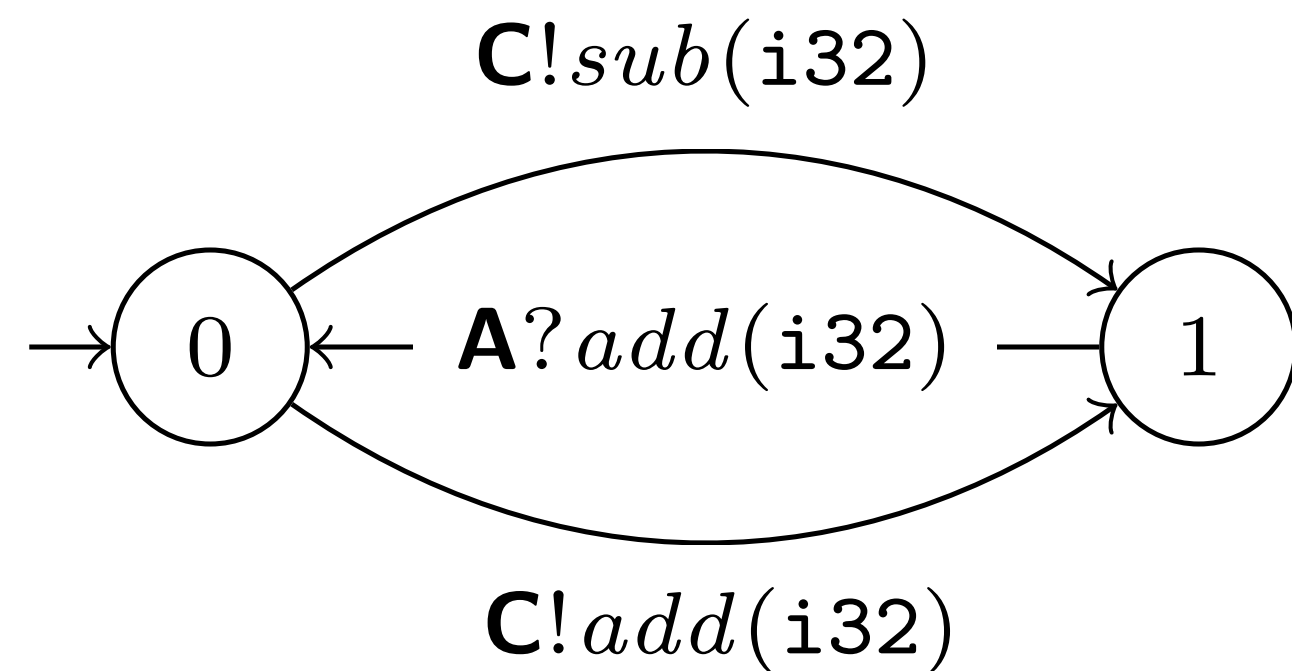


```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            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

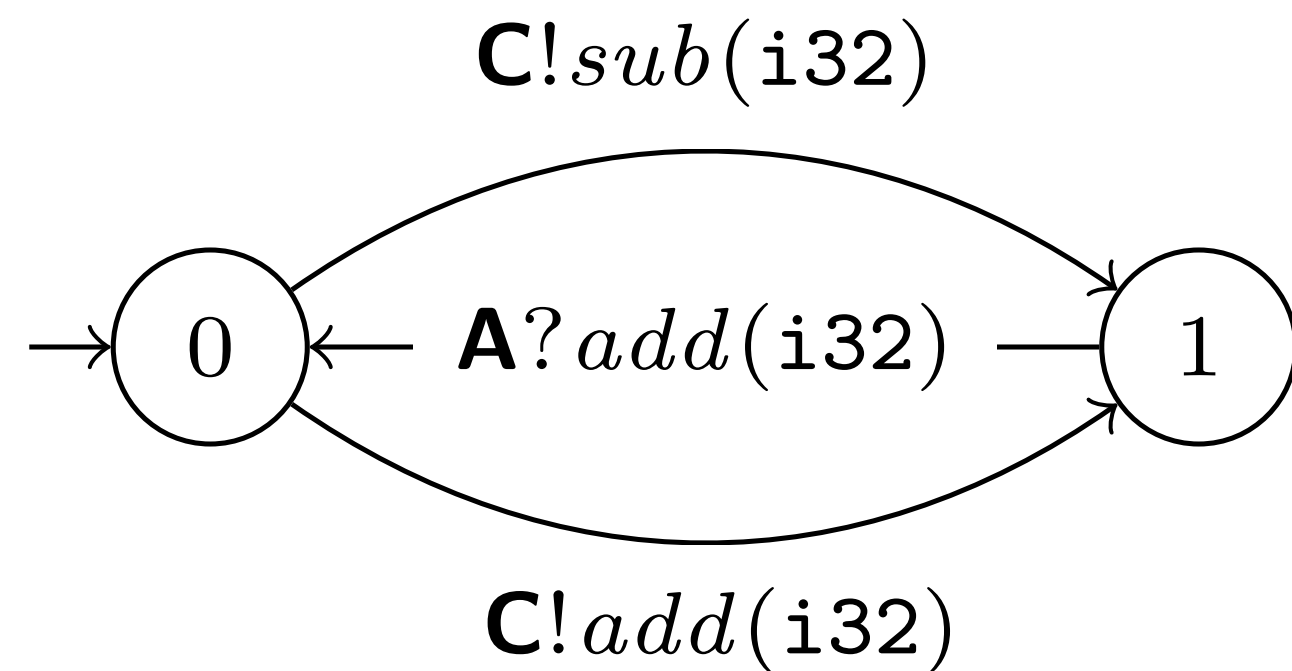


```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            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

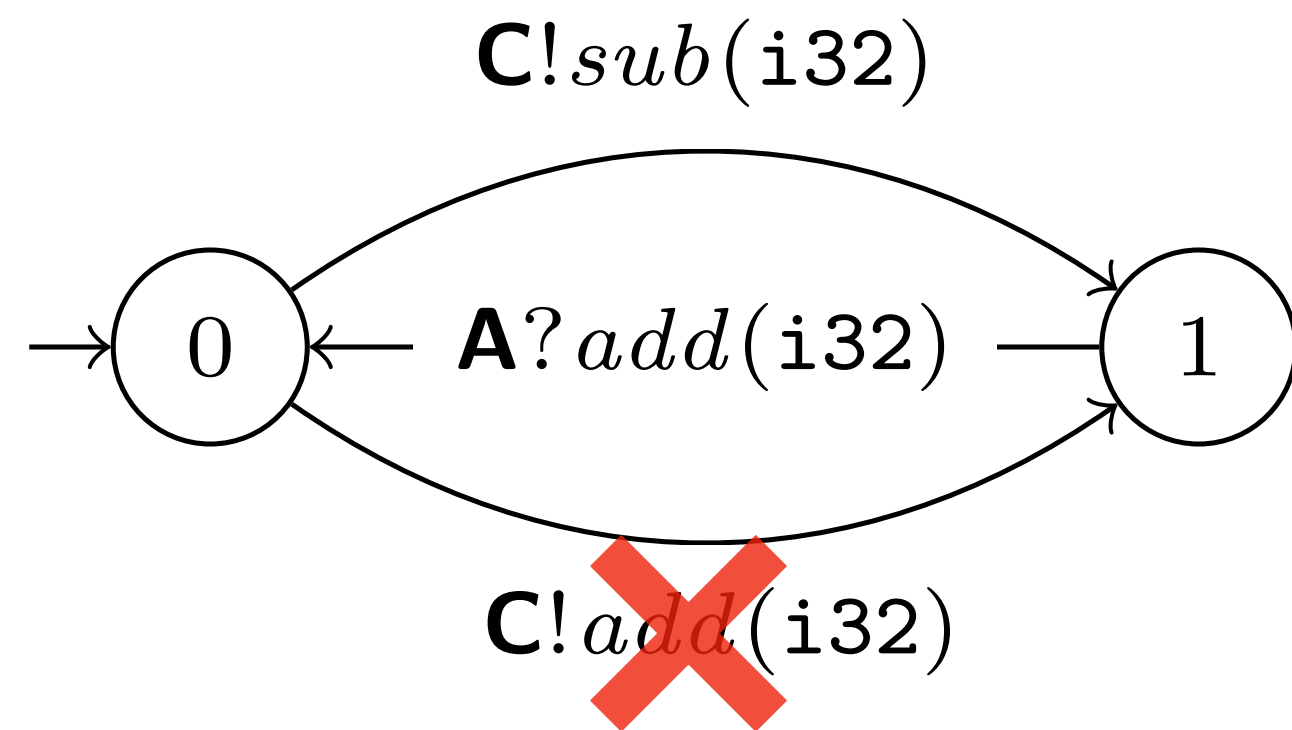


```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

            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

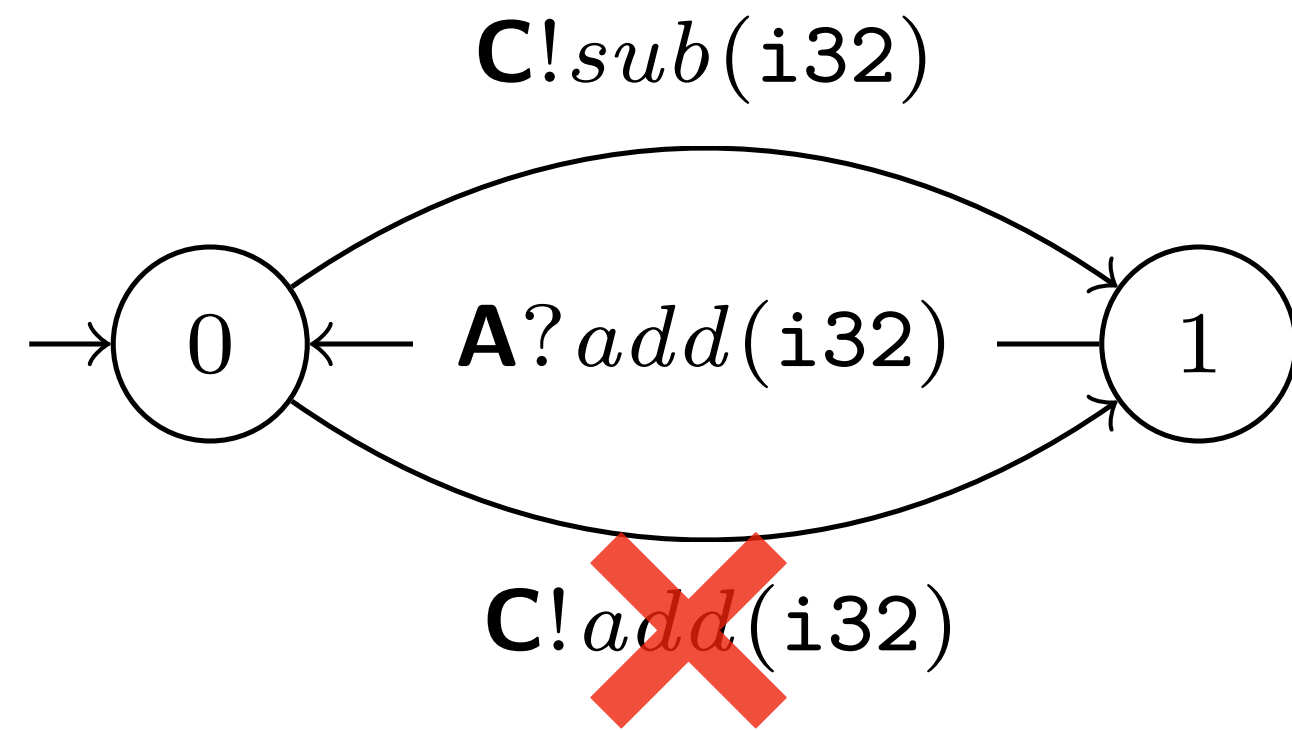


```
async fn ring_b(
    role: &mut B,
    mut input: i32,
) -> Result<Infallible> {
    try_session(role, |mut s: RingB<'_, _>| async {
        loop {
            let x = input * 2;

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

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

Mobility Reading Group

<http://mrg.cs.ox.ac.uk>



The screenshot shows the website for the Mobility Reading Group. At the top, there is a logo consisting of a blue pi symbol with the text "session type" above it. To the right of the logo is the text "MobilityReadingGroup" in a large, bold, black font, followed by "π-calculus, Session Types research at Imperial College" in a smaller, grey font. Below this is a dark grey navigation bar with the following items: "Home" (highlighted with a blue underline), "People", "Publications", "Grants", "Talks", "Tutorials", "Tools", "Awards", and "Kohei Honda". The main content area is divided into two columns. The left column is titled "NEWS" in large, blue, uppercase letters. It contains two news items, each with a date and a short paragraph. The right column is titled "SELECTED PUBLICATIONS" in large, blue, uppercase letters. It contains a section for the year "2021" with two publication entries, each with the authors' names and a link to the publication title.

session type **MobilityReadingGroup**
π-calculus, Session Types research at Imperial College

Home People Publications Grants Talks Tutorials Tools Awards Kohei Honda

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!

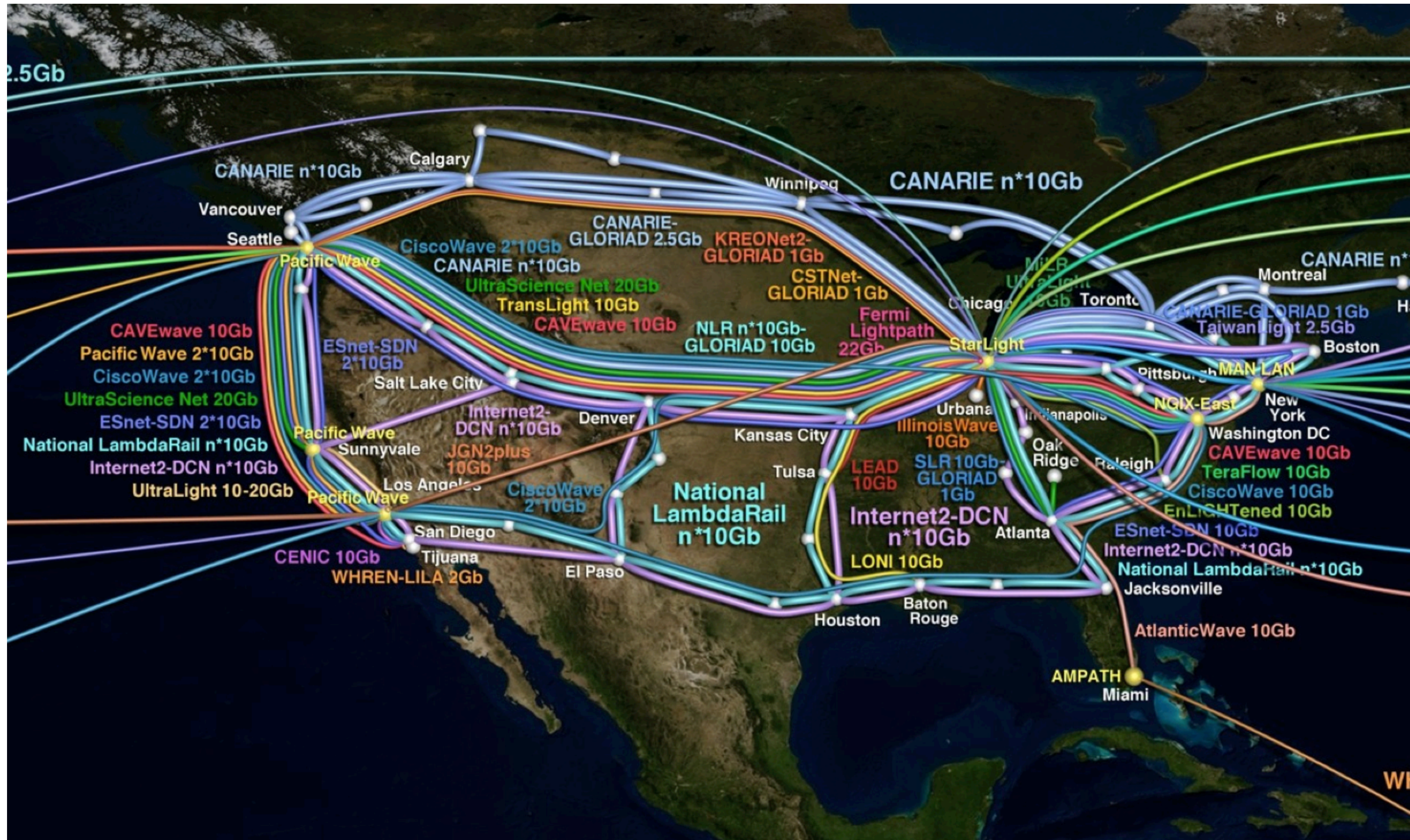
SELECTED PUBLICATIONS

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.

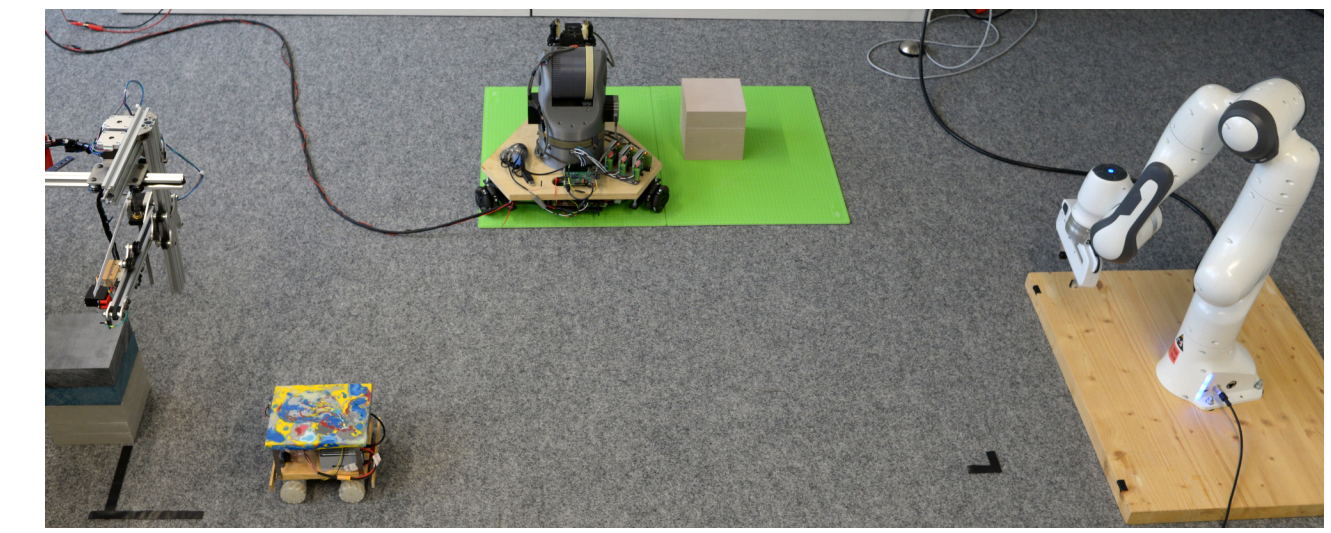
More Applications on Multiparty Session Types



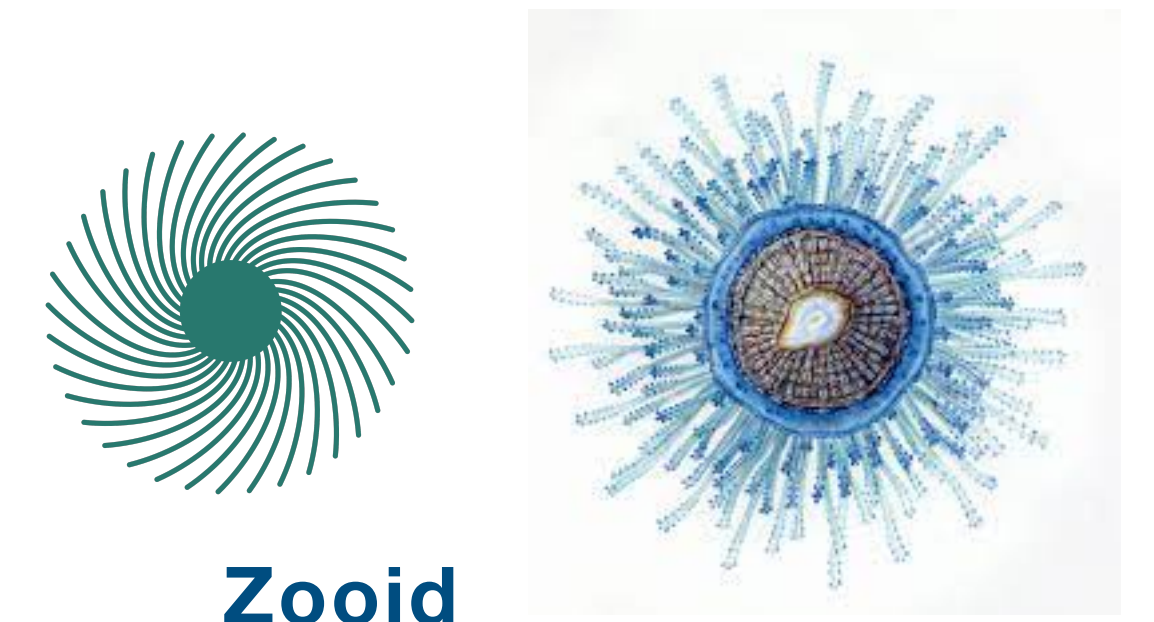
Ocean Observatories Initiative



Distributed Traces






Robotics



Zooid

Mechanisation

Rust and Multiparty Session Types

- Multiparty session types and communicating automata
 - Invited paper in the FCT '21 proceedings
 -  Scribble <https://github.com/scribble>
 -  <https://github.com/nuscr>
- Applications of multiparty session types using communicating automata
 -  <https://github.com/zakcutner/rumpsteak>

Mini-Project Proposals

- **Rust** Programming for **Autonomous Actors** or **Robotics Coordination** with AcTyx AG (<https://www.actyx.com/>)
- Verified Multi-Agent Programming with Actor Models (**Model Checking of Go**)
- **Theorem Provers (Coq)** to Verify Actor Models
- Efficient Machine Learning Algorithms with **Rust** or **Go**
- **Probabilistic Process Calculi/Programming Language Theory**
- Please contact with nobuko.yoshida@cs.ox.ac.uk if you wish to learn or do **programming language** or **theorem prover**'s projects