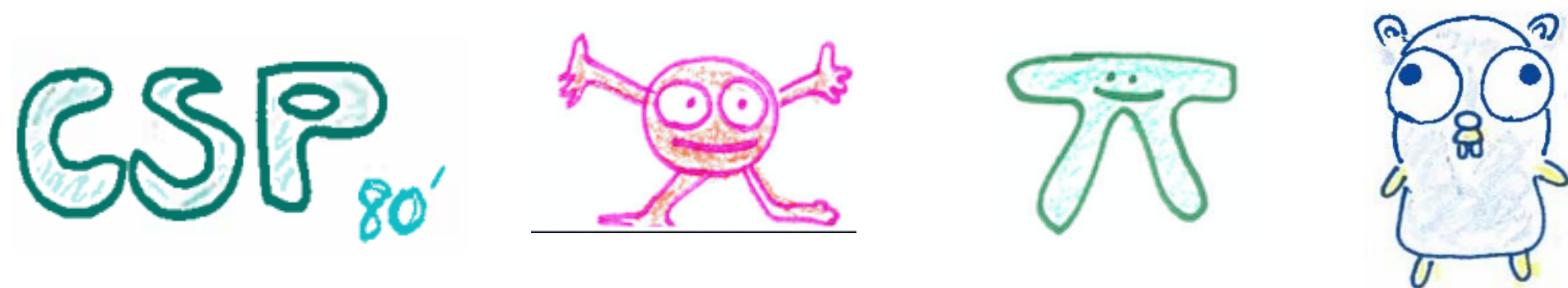


AIMS: Mini-Project Presentation

- **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)

Communications are Ubiquitous

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



microservices

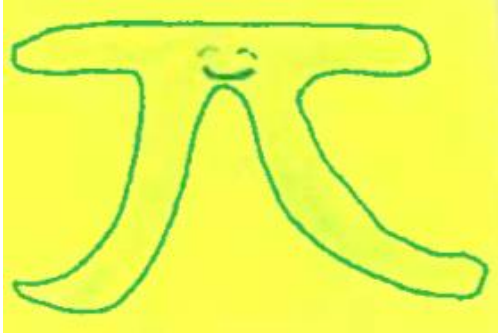


Why Session Types, Why Now?

Significant academic and industry interests via fundamental breakthroughs

Binary Session Types

ESOP'98



Joined W3C Standardization

2002

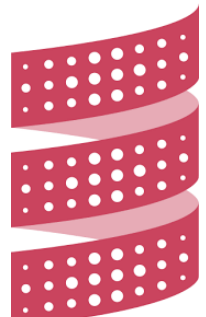


Multiparty Session Types

POPL' 08



TypeScript



Scala

akka



ERLANG

MPI



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**.



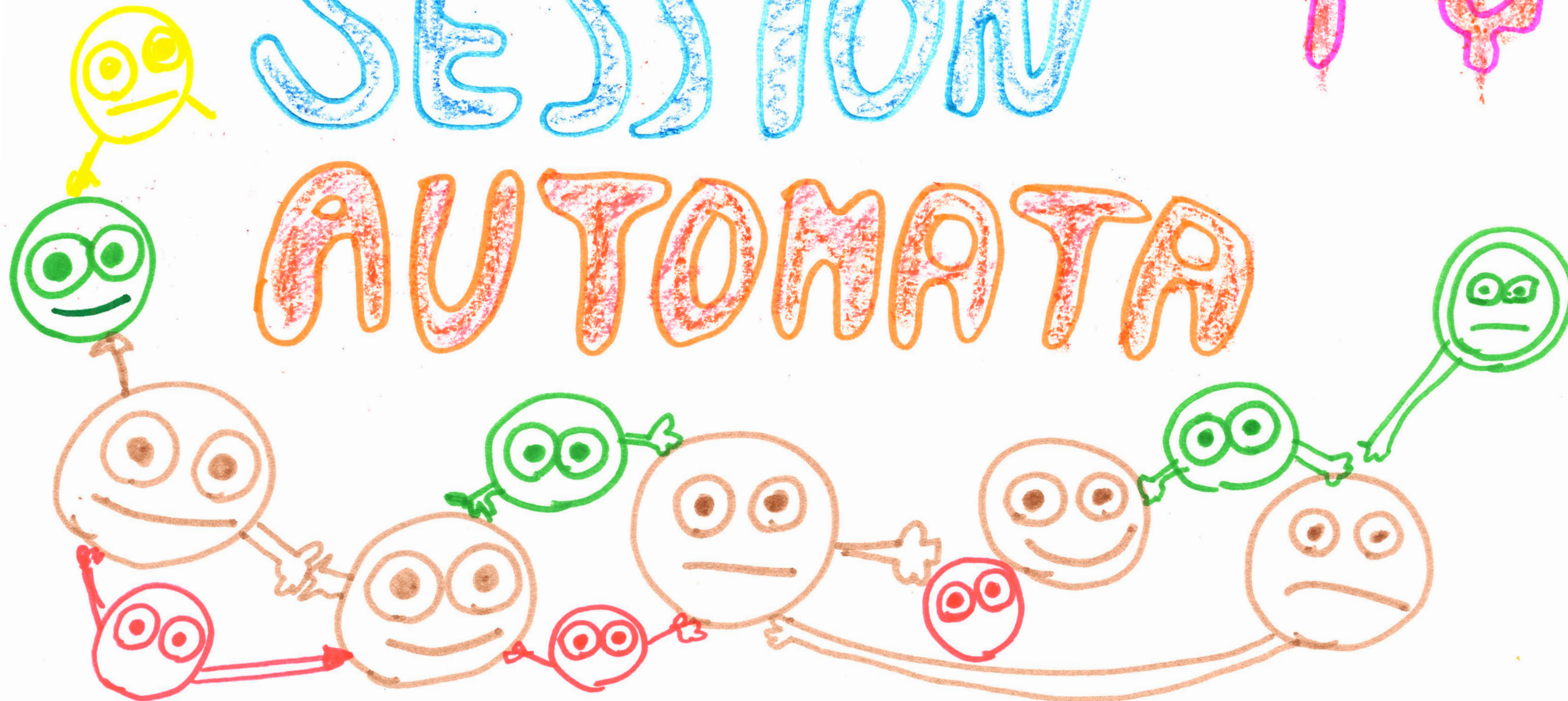
What class of
CA corresponds
to MPST?

MULTIPAR

SESSION

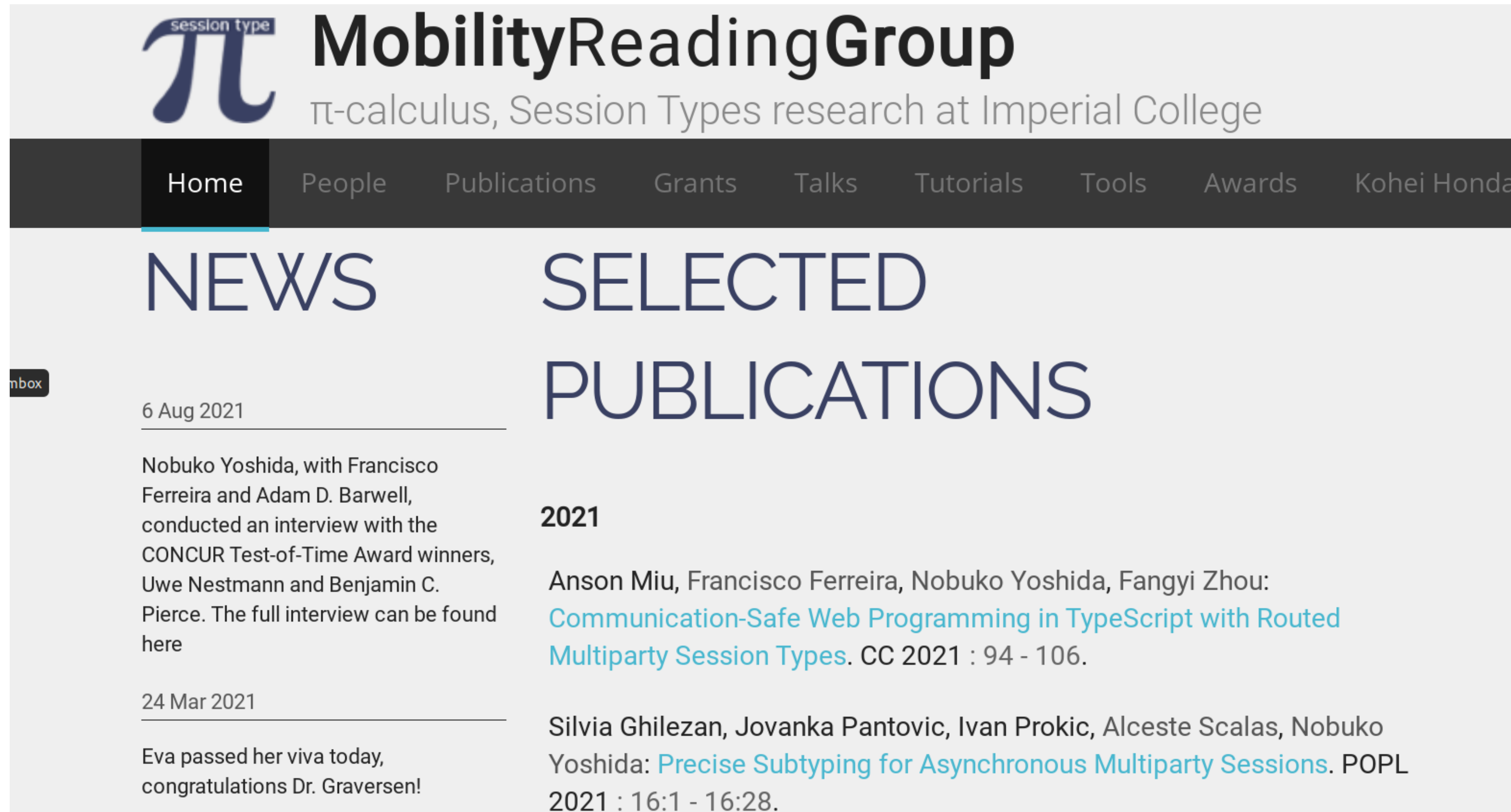
TV

AUTOMATA



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, followed by the text "MobilityReadingGroup" and "π-calculus, Session Types research at Imperial College". Below this is a dark navigation bar with links for "Home", "People", "Publications", "Grants", "Talks", "Tutorials", "Tools", "Awards", and "Kohei Honda". The main content area is split into two columns. The left column is titled "NEWS" and contains two entries: one dated "6 Aug 2021" about an interview with CONCUR Test-of-Time Award winners, and another dated "24 Mar 2021" congratulating Dr. Graversen. The right column is titled "SELECTED PUBLICATIONS" and lists two papers from 2021: "Communication-Safe Web Programming in TypeScript with Routed Multiparty Session Types" and "Precise Subtyping for Asynchronous Multiparty Sessions".

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.

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.

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>