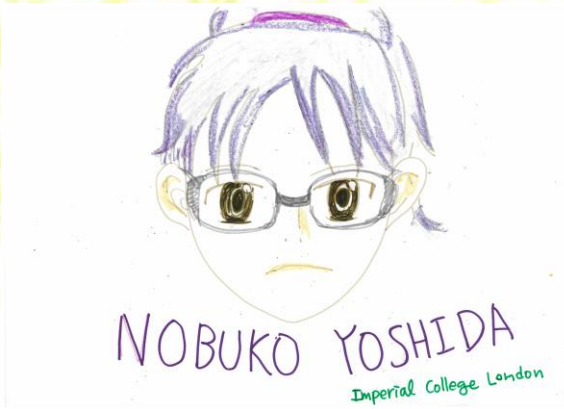
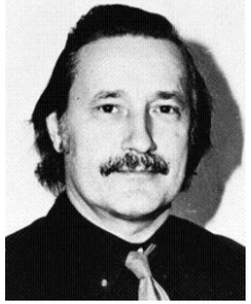
OPEN PROBLEMS OF SESSION



NOBUKO YOSHIDA
Imperial College London

Multiparty Session Types

Specification-Guided Concurrent and Distributed Programming



1916-1975

- **Christopher Strachey** (sequential computation)
  - ***Types*** = **abstract** and **digest computation (data types, polymorphism)**
  - **Structured programming = High-level programming**
- **Session types** (concurrency & communication)
  - Structured programming = ***protocols***

Problems: Concurrency Bugs

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

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 code base [PLDI 2022]

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

deadlock

channel errors

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

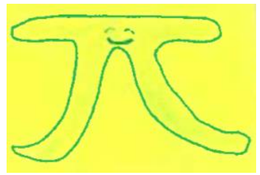
- Prevent concurrency bugs.
- Can abstract, implement and manage communications as **Protocols.**
- **Clean, Cheap** and **Retrofittable.**

Why Session Types, Why Now?

Significant academic and industry interests via fundamental breakthroughs

Milner, Honda, NY

Binary Session Types — ESOP'98



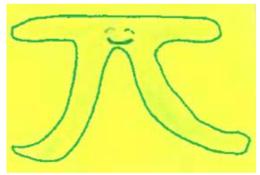
↓

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

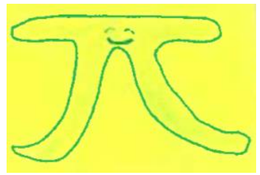
↓

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

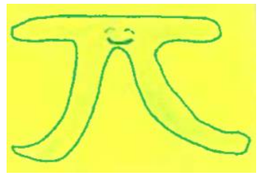
Red Hat

Scribble

Why Session Types, Why Now?

Significant academic and industry interests via fundamental breakthroughs

Binary Session Types — ESOP'98



Joined W3C Standardization — 2002

Multiparty Session Types — POPL' 08

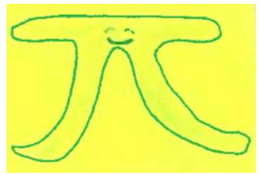
25M US$ large cyber infrastructures

middleware standardisations

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







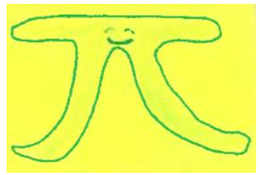



Why Session Types, Why Now?

Significant academic and industry interests via fundamental breakthroughs

ETAPS Test Time Award 2019

Binary Session Types — ESOP'98



Joined W3C Standardization — 2002

POPL Influential Paper Award 2018

Multiparty Session Types — POPL' 08

My Open Problems of Session Types

Foundations

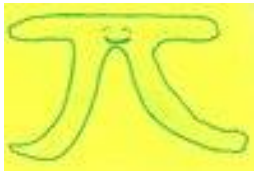
- Semantics
- Verification

Applications

- Parallel Computing Systems
- Distributed Systems
- Open Systems

Foundations of Session Types

- The Original Honda-Vasco-Kubo [1998]



- Linear Logic-Based Session Types:
  Caires & Pfenning [2010] & Wadler [2012]
- Foundamentals of Session Types: Vasconcelos [2009]
- Communicating Automata Theory [1995]
  e.g., Model Checking (mCLR2)

$\lambda \quad M ::= x \mid \lambda x.M \mid MN$

$\pi \quad P ::= \sum \pi_i.P_i \mid P|Q \mid (\nu x)P \mid !P \mid \emptyset.$

with $\pi ::= x(\tilde{y}) \mid \bar{x}\langle \tilde{y} \rangle.$

λ in π

$[\![x]\!]_u \stackrel{def}{=} \bar{x}\langle u \rangle.$

$[\![\lambda x.M]\!]_u \stackrel{def}{=} u(x u').[\![M]\!]_{u'}.$

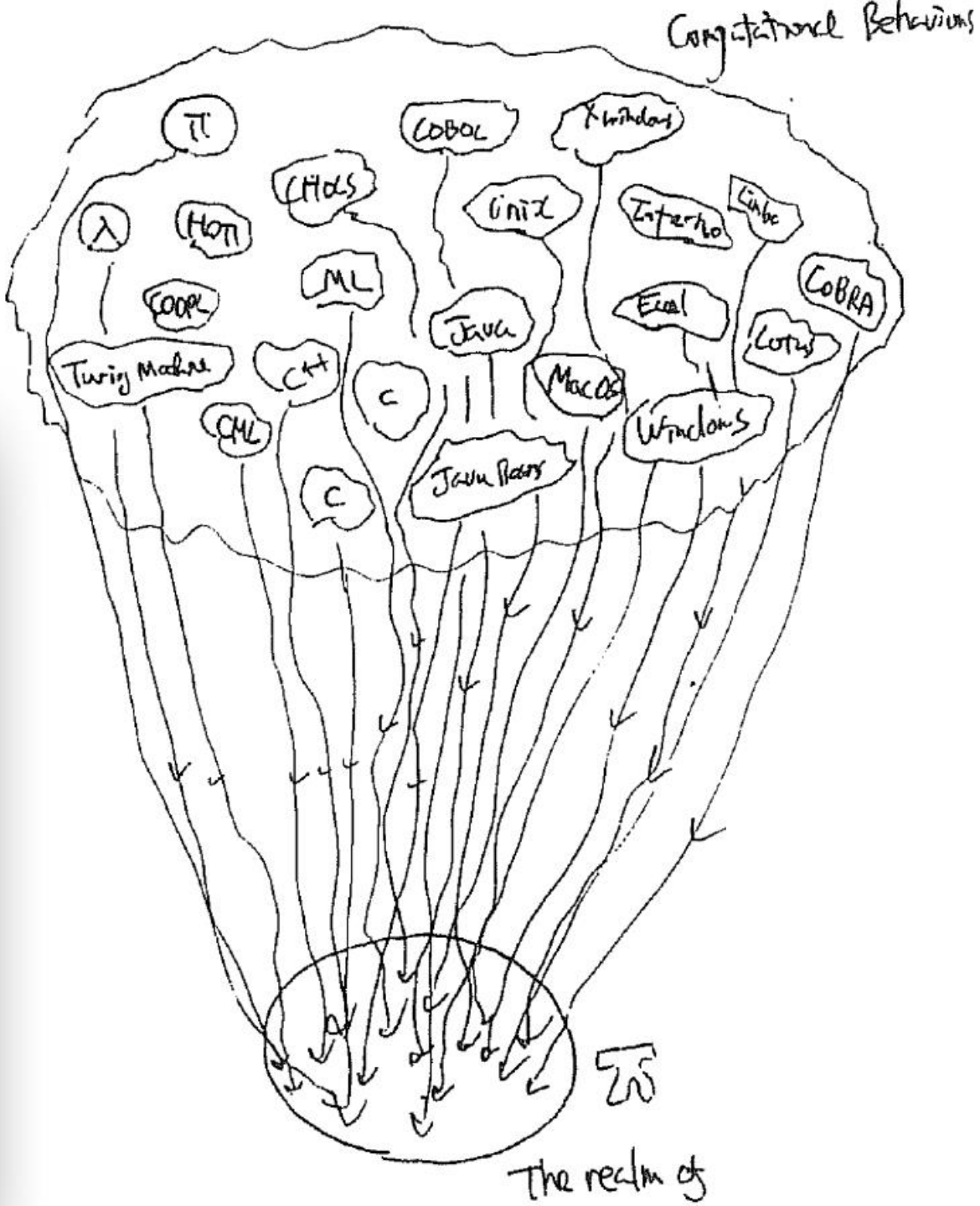
$[\![MN]\!]_u \stackrel{def}{=} (\nu f x)\left([\![M]\!]_f \mid \bar{f}\langle x u \rangle \mid [\![x=N]\!]\right)$

with $[\![x=N]\!] \stackrel{def}{=} !x(u').[\![N]\!]_{u'}.$

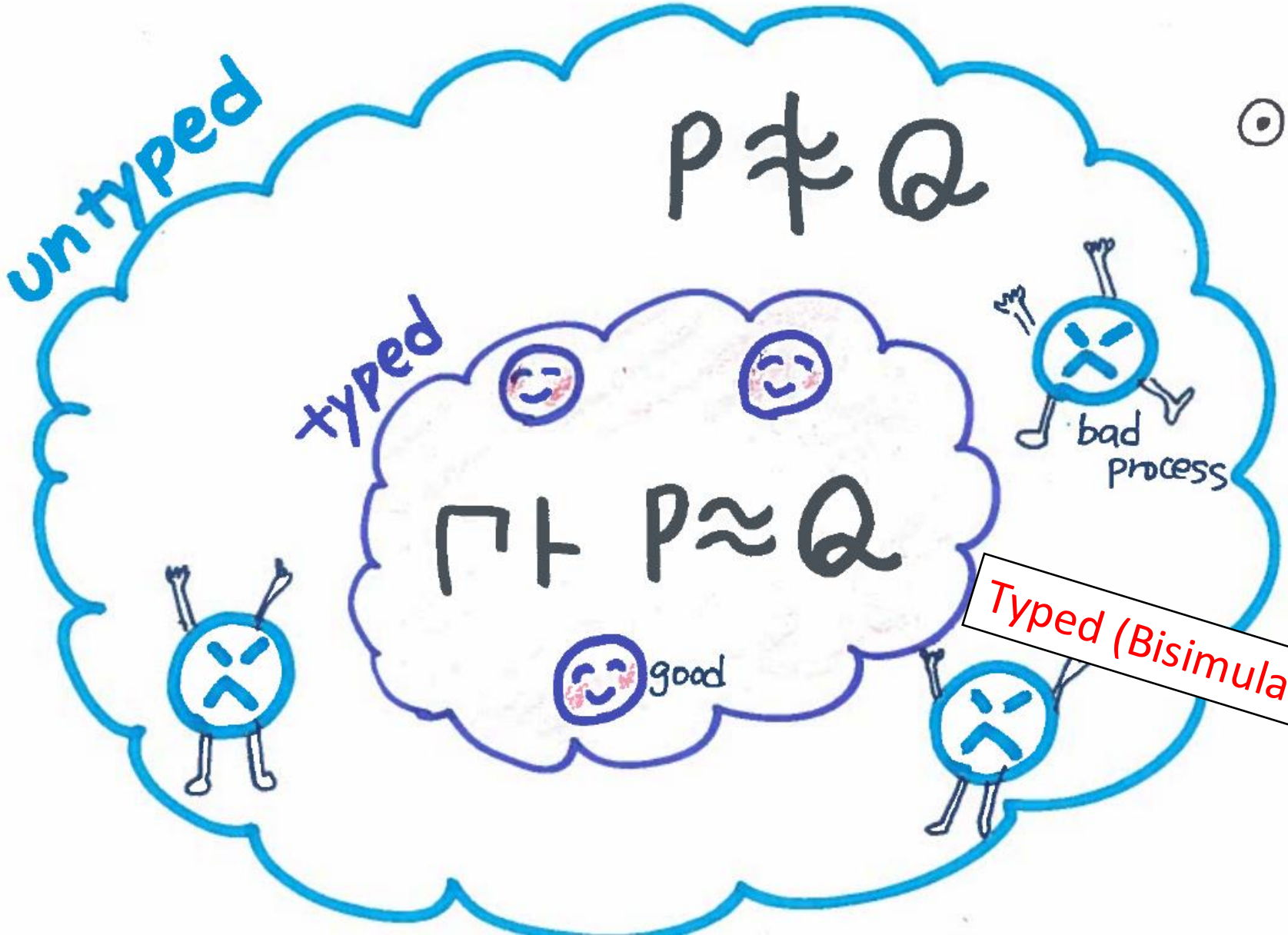
Pi-Calculus vs Lambda-Calculus
Kohei Honda [1995]

Pi-Calculus
Kohei Honda [1995]

- λ-calculus [MPW89, Milner90, Milner92, ...]
- Concurrent Object [Walker81]
- ω-order term passing [Sangiorgi 92]
- Various data structures [Milner 92, ...]
- Proof Nets [Bellin and Scott 93]
- Arbitrary 'constant' interaction [HYS4]
- strategies on Games [HO95]
- ⋮



IO-subtyping , Linear types, Secure Information Flow, ...



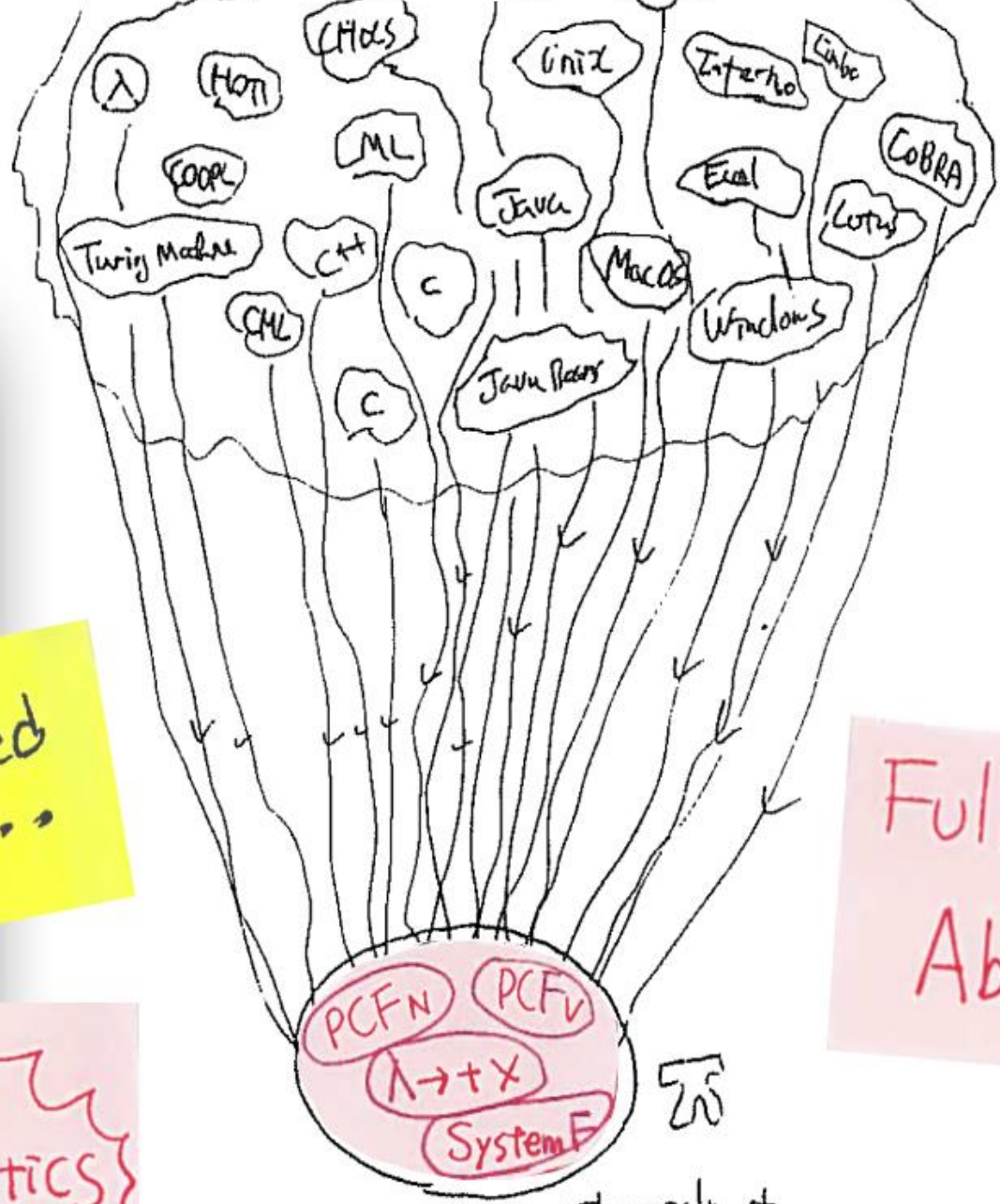
- ⊙ Correctness of Encoding
- ⊙ Limit environment Γ
  ⇒ Equate more processes
- ⊙ Compositional

Typed (Bisimulation) Semantics

- λ-calculus [MPW89, Milner90, Milner92, ...]
- Concurrent Object [Walker91]
- ω-order term passing [Sangiorgi92]
- Various data structures [Milner
- Proof Nets [Bellin and Scott 93]
- Arbitrary "constant" interaction [
- strategies on Games [HO95]

Complicated ...

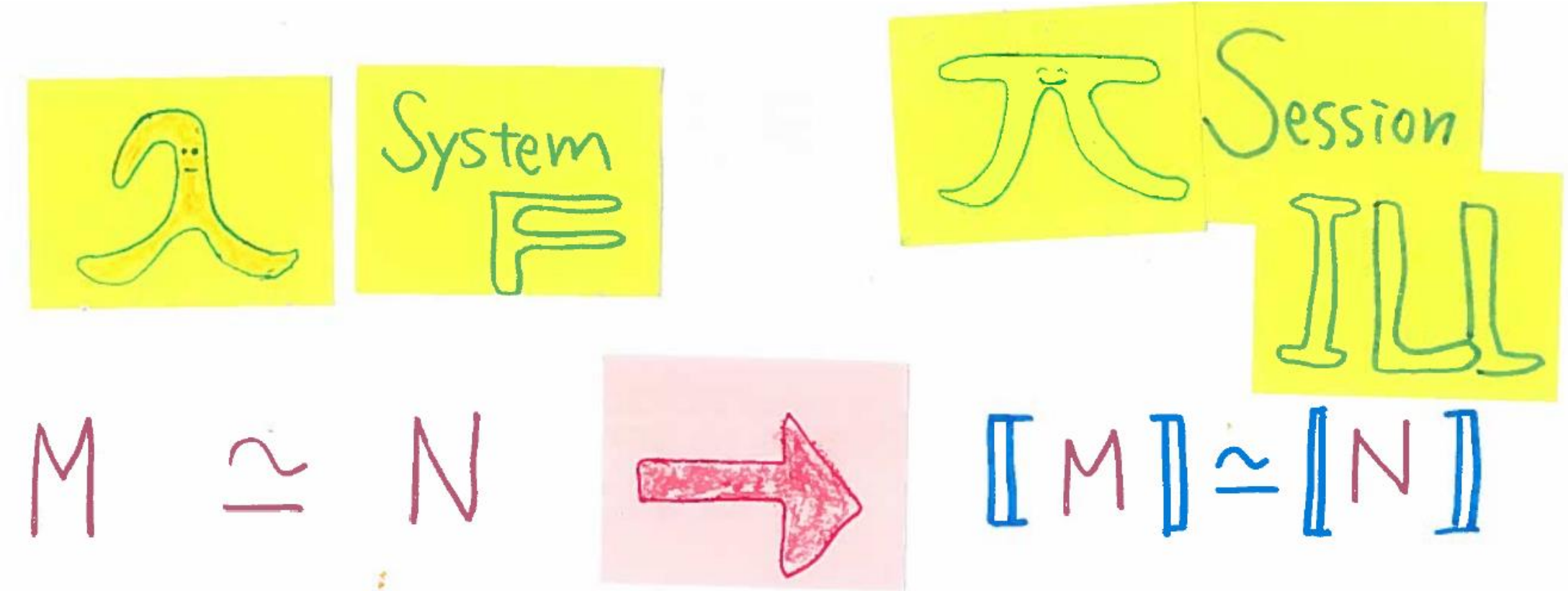
Game Semantics



The realm of Name Passing Interactions

Fully Abstract

Linear Logic-Based Session Types [Toninho & NY 2021]



λ in π

Milner's Encoding 1991

$$[\![x]\!]_u \stackrel{\text{def}}{=} \overline{x}\langle u \rangle.$$

$$[\![\lambda x.M]\!]_u \stackrel{\text{def}}{=} u(xu').[\![M]\!]_{u'}$$

$$[\![MN]\!]_u \stackrel{\text{def}}{=} (\nu f x)\left([\![M]\!]_f \mid \overline{f}\langle xu \rangle \mid [\![x=N]\!]\right)$$

with $[\![x=N]\!] \stackrel{\text{def}}{=} !x(u').[\![N]\!]_{u'}$.

$[\![M]\!]_a$

↑ name

Session $\Pi\Pi$

λ in π

$$[\![x]\!]_u \stackrel{\text{def}}{=} \overline{x}(u).$$

$$[\![\lambda x.M]\!]_u \stackrel{\text{def}}{=} u(x u'). [\![M]\!]u'.$$

$$[\![MN]\!]_u \stackrel{\text{def}}{=} (\nu f x)\left([\![M]\!]_f \mid \overline{f}\langle x u\rangle \mid [\![x{=}N]\!]\right)$$

with $[\![x{=}N]\!] \stackrel{\text{def}}{=} !x(u'). [\![N]\!]u'.$

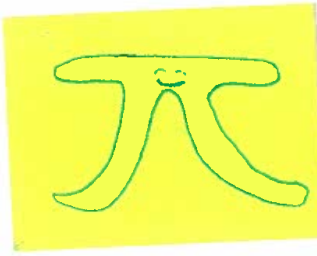
$$[\![x]\!]_a = [x \leftrightarrow a]$$

$$[\![\langle\rangle]\!]_a = 0$$

$$[\![\lambda x.M]\!]_a = a(x). [\![M]\!]_a$$

$$[\![MN]\!]_a = [\![M]\!]x \mid \overline{x}(y). [\![N]\!]_y \mid [x \leftrightarrow a]$$

System F



Session Π

$M \simeq N$

$[\![M]\!] \simeq [\![N]\!]$

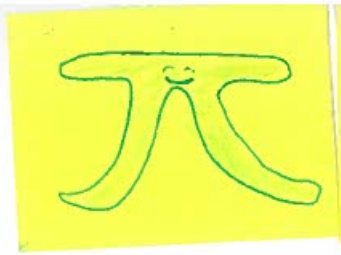
$[\![P]\!] \simeq [\![Q]\!]$

$P \simeq Q$

Reverse

New

System F



Session

$M \simeq N$

$[\![M]\!] \simeq [\![N]\!]$

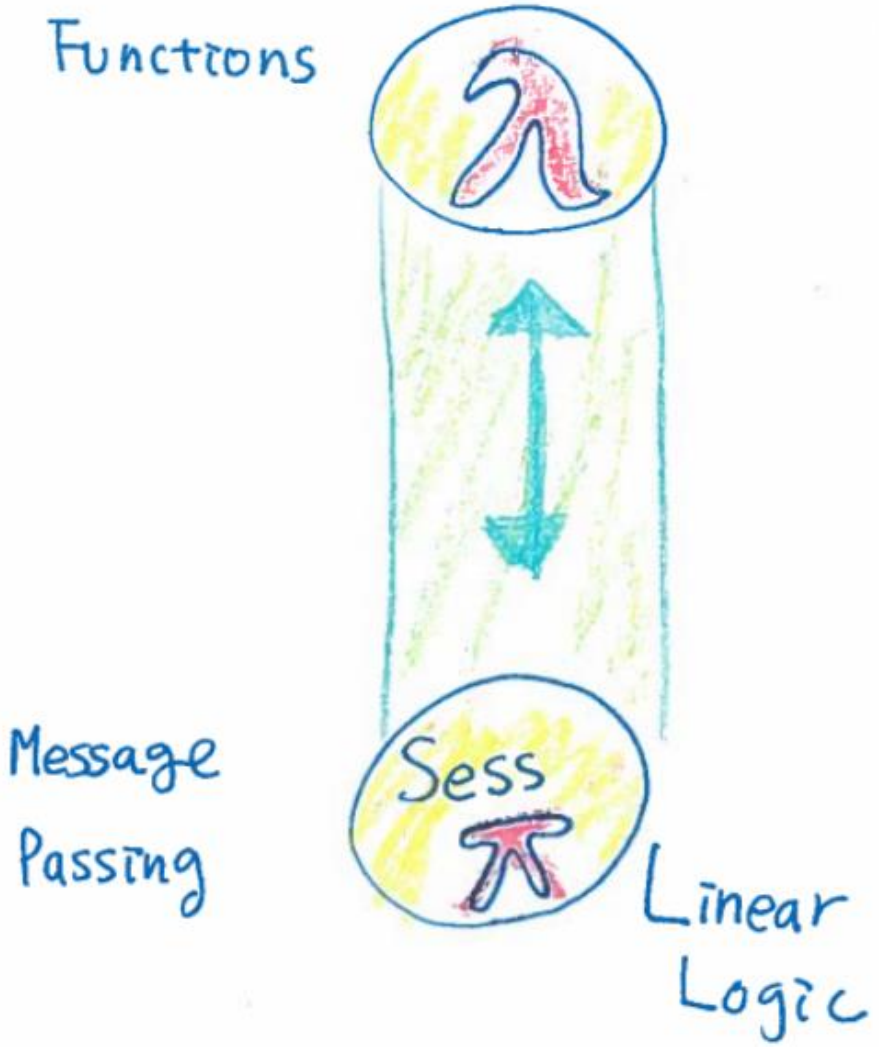
$[\![P]\!] \simeq [\![Q]\!]$

$P \simeq Q$

Reverse

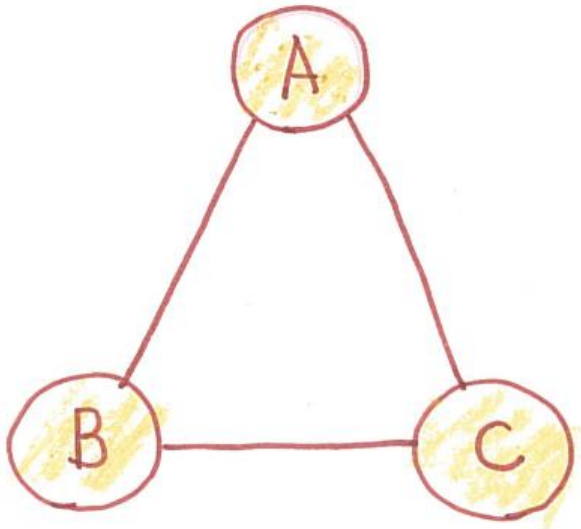
New

Summary

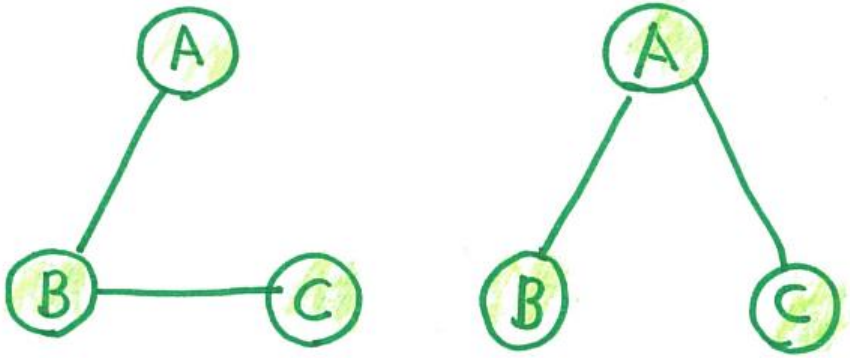


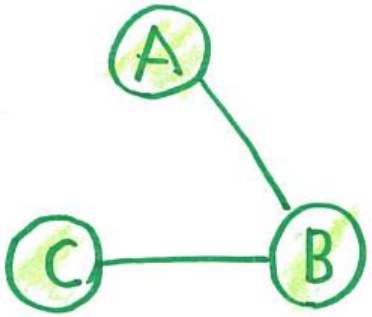
Distributability & Interconnectability

[Toninho & NY 19, Peters, Nestmann & Schmitt 23]



Multi Party





Linear Logical Session

cf. [1995] Specification Structures and Proposition as Types

Mixed Binary Sessions based on Fundamentals of Session Types [CMV'22]

$q ::=$ Qualifiers:

lin linear

un unrestricted

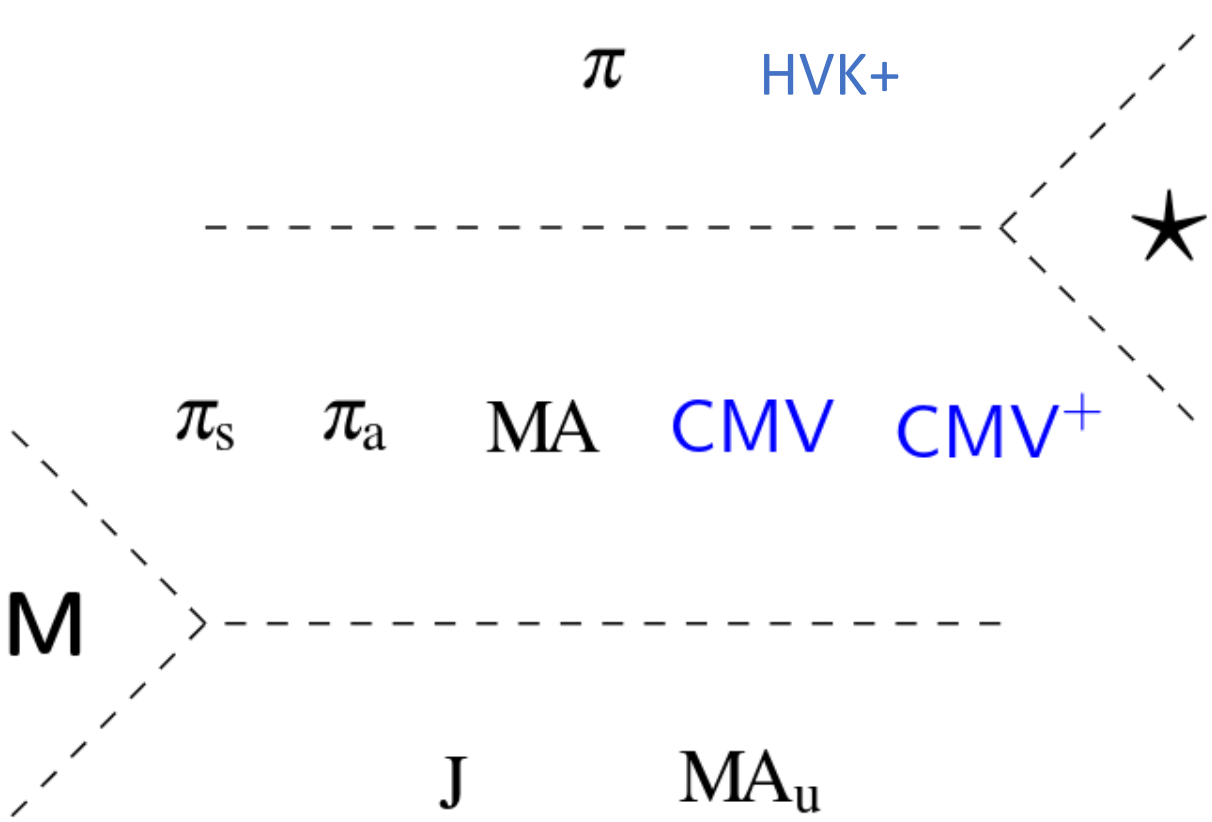
π: $\bar{y}z.P + R \mid y(x).Q + N \longmapsto P \mid Q\{z/x\}$ $\tau.P + R \longmapsto P$

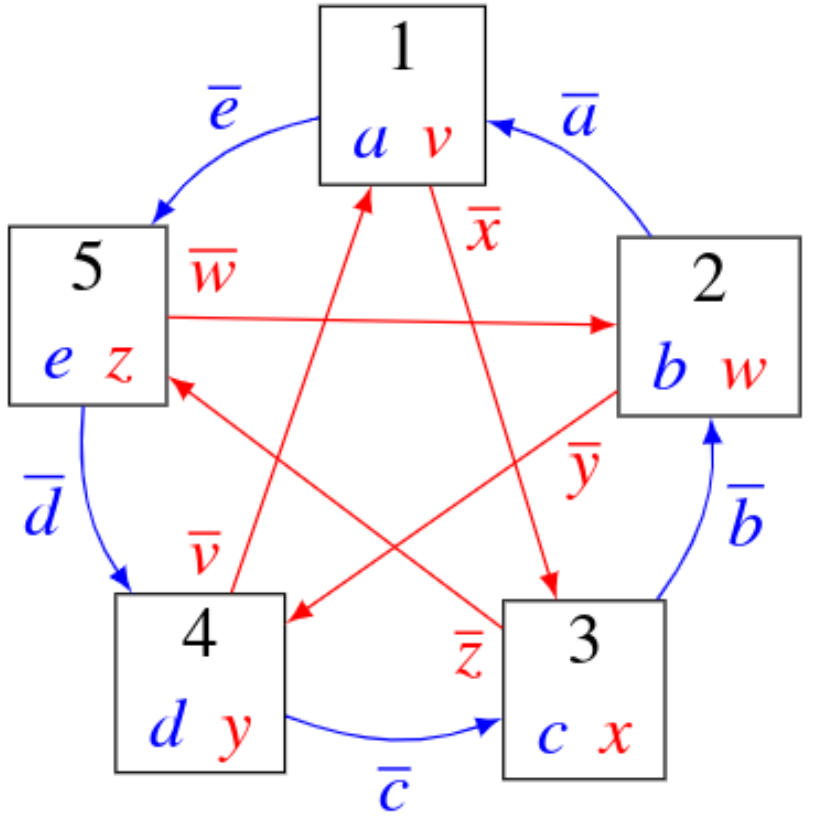
CMV$^+$/CMV: if true then P else $Q \longmapsto P$ if false then P else $Q \longmapsto Q$

CMV$^+$: $(\nu yz)(y\,(1!v.P + M) \mid z\,(1?x.Q + N) \mid R) \longmapsto (\nu yz)(P \mid Q\{v/x\} \mid R)$

CMV: $(\nu yz)(y!v.P \mid z?xQ \mid R) \longmapsto (\nu yz)(P \mid Q\{v/x\} \mid R)$

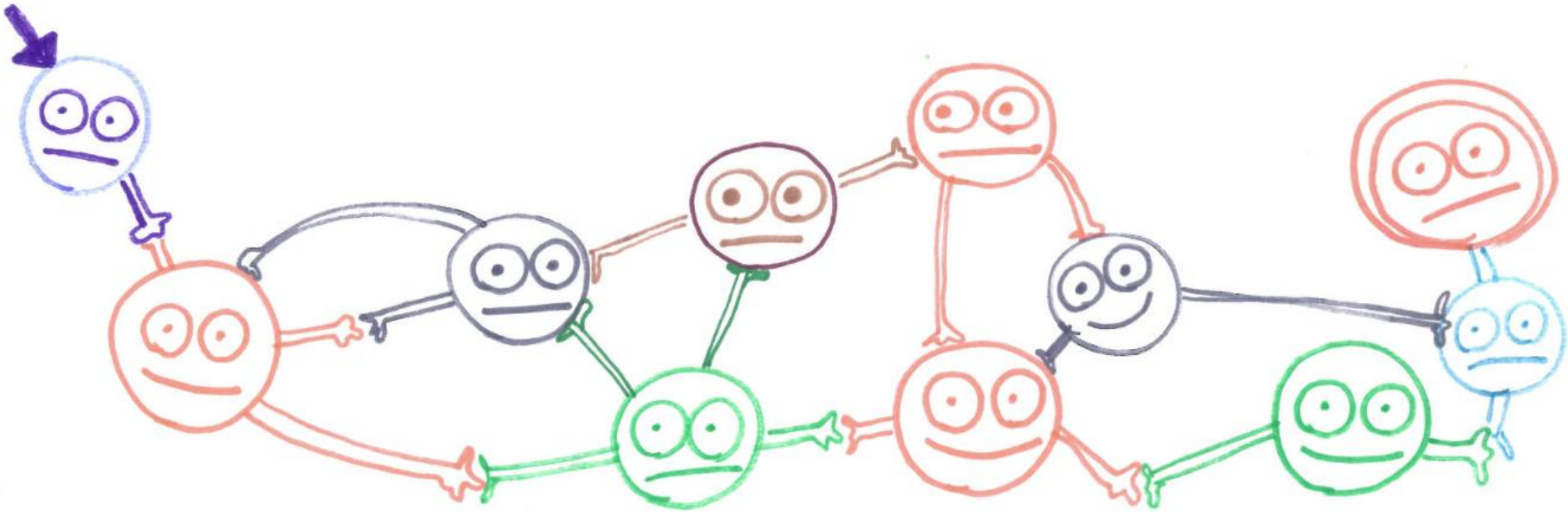
$(\nu yz)\left(y \triangleleft 1_j.P \mid z \triangleright \{1_i : Q_i\}_{i \in I} \mid R\right) \longmapsto (\nu yz)(P \mid Q_j \mid R)$

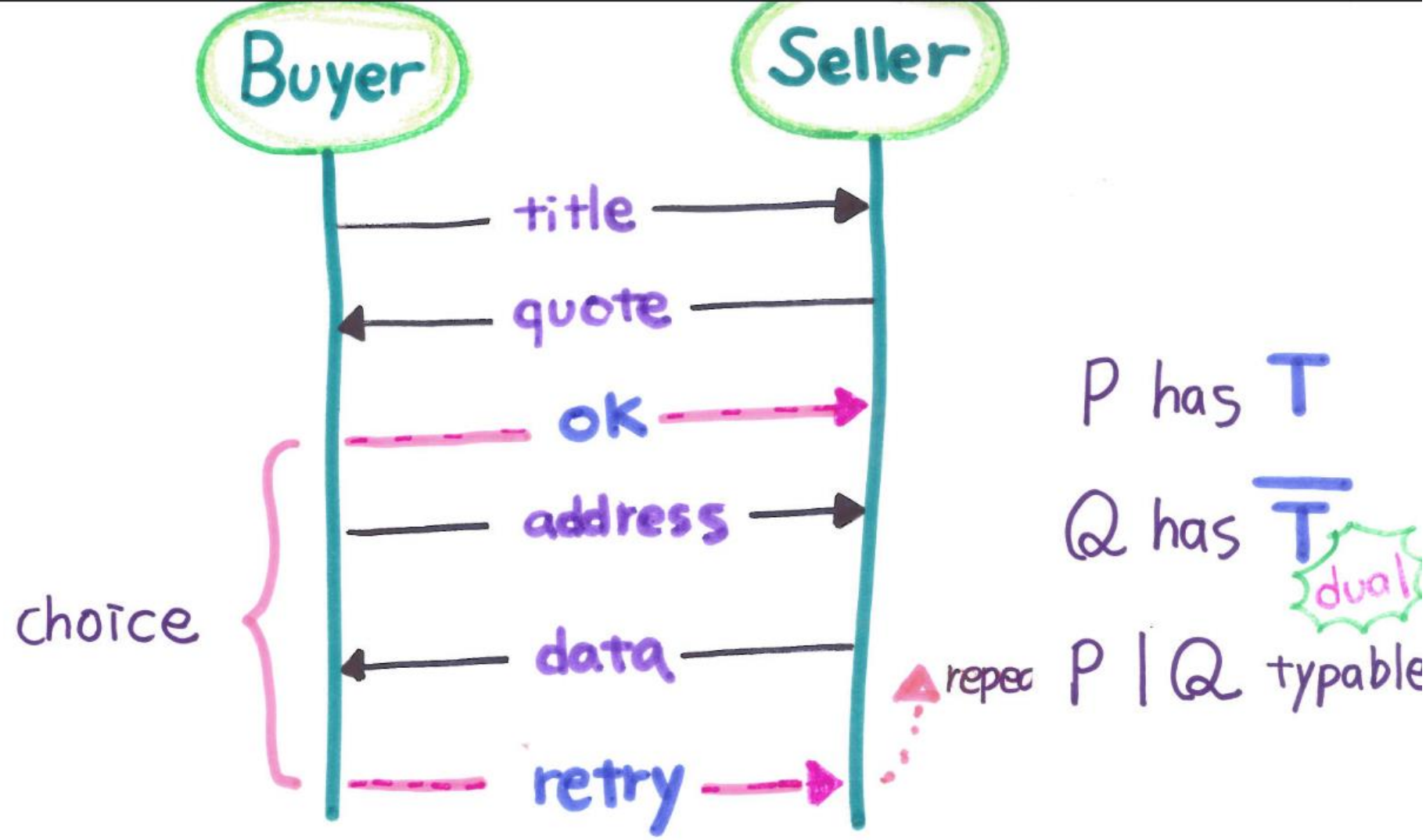




Foundamentails of Session Types [Vasconcelos 09]
Mixed Sessions [CNV+21] = Separate Choices [Peters & NY 22]

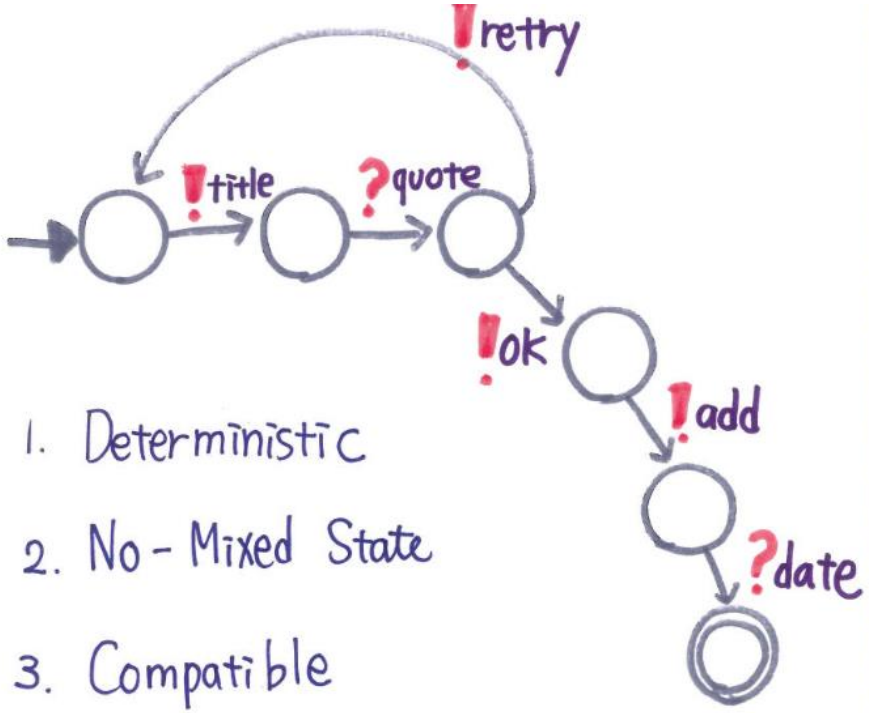
Communicating Automata [1983]





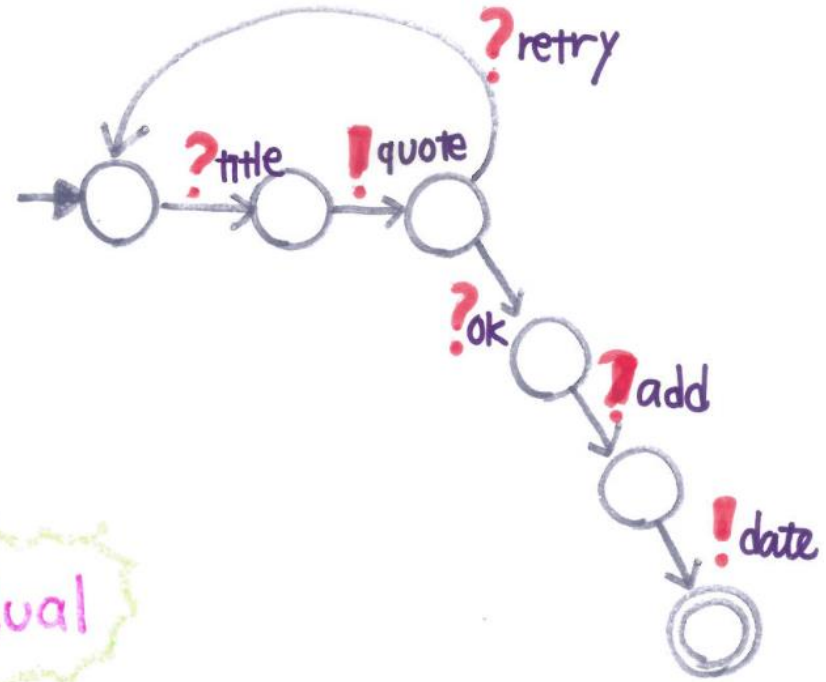
$\mu t.!\,\text{Title};?\,\text{Quote};!\{\textbf{ok}:!\,\text{Add};?\,\text{Date},\ \textbf{retry}:t\}$

$\mu t.?\,\text{Title};!\,\text{Quote};?\{\textbf{ok}:?\,\text{Add};!\,\text{Date},\ \textbf{retry}:t\}$



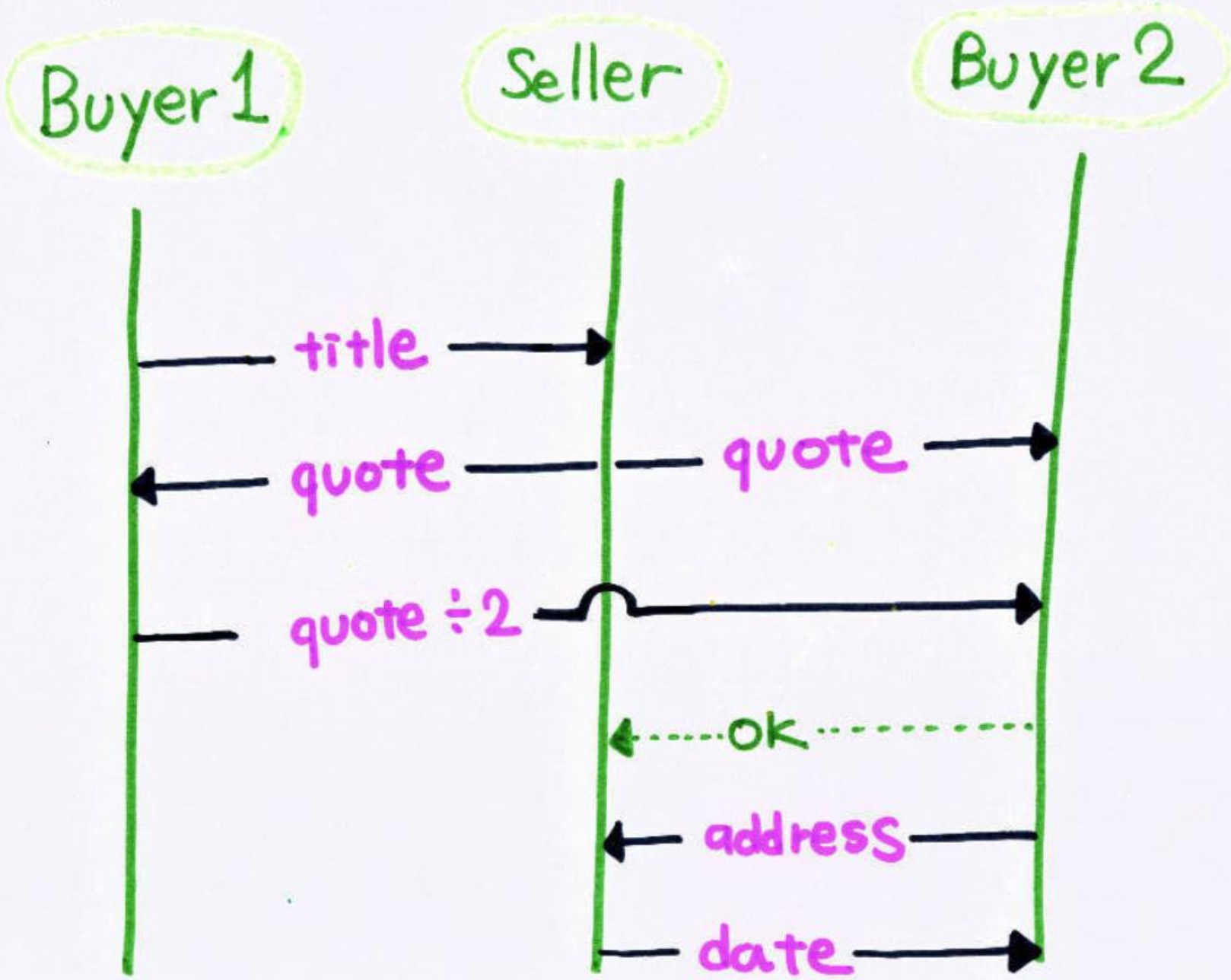
1. Deterministic
2. No-Mixed State
3. Compatible

dual



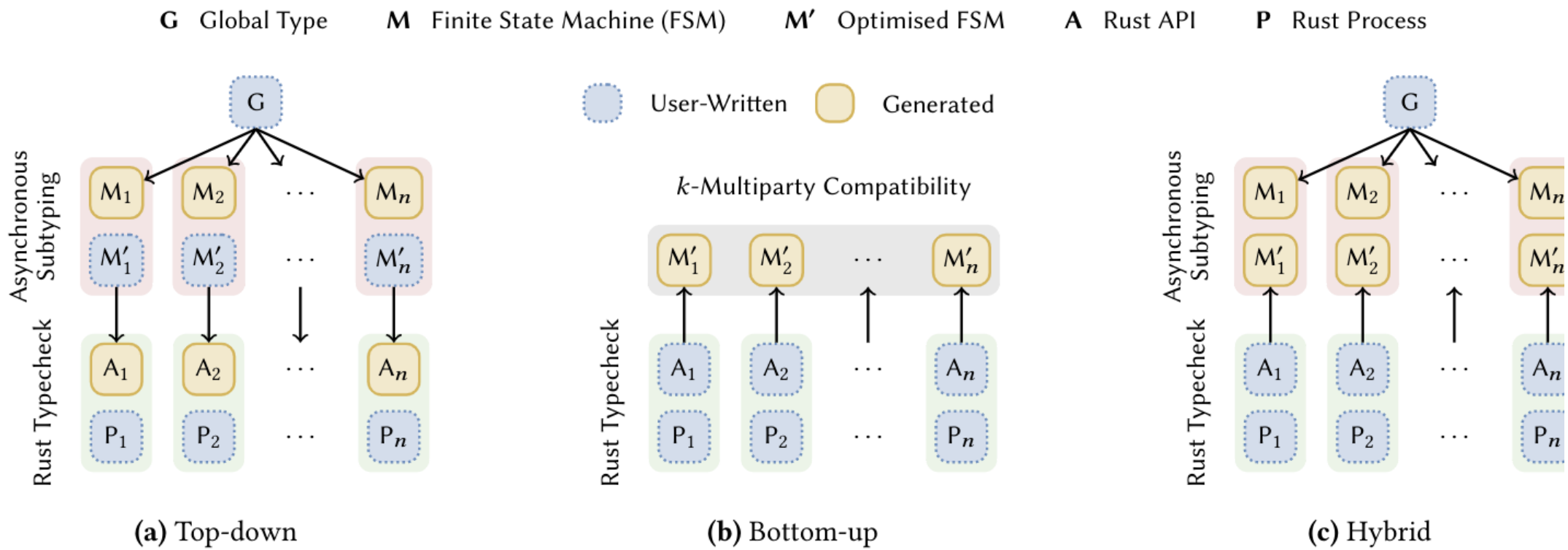
[Gouda et al 1986] Two compatible machines without mixed states which are deterministic satisfy deadlock-freedom.

Multiparty Session Types



Rumpsteak Framework

Three Approaches



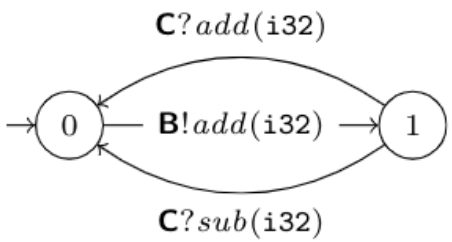
(a) Top-down **(b)** Bottom-up **(c)** Hybrid

Ring Protocol

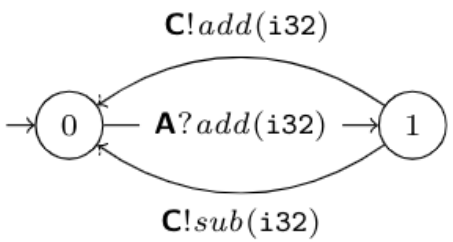
Example

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

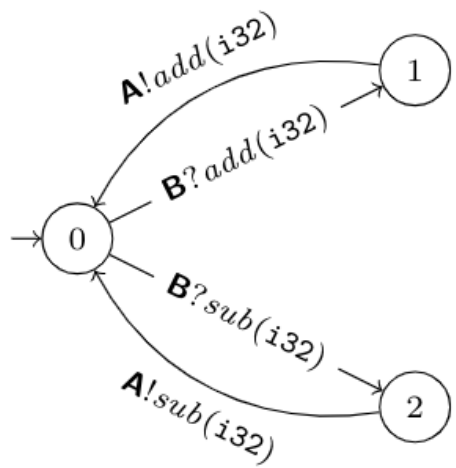
PROJECTION



PROJECTION

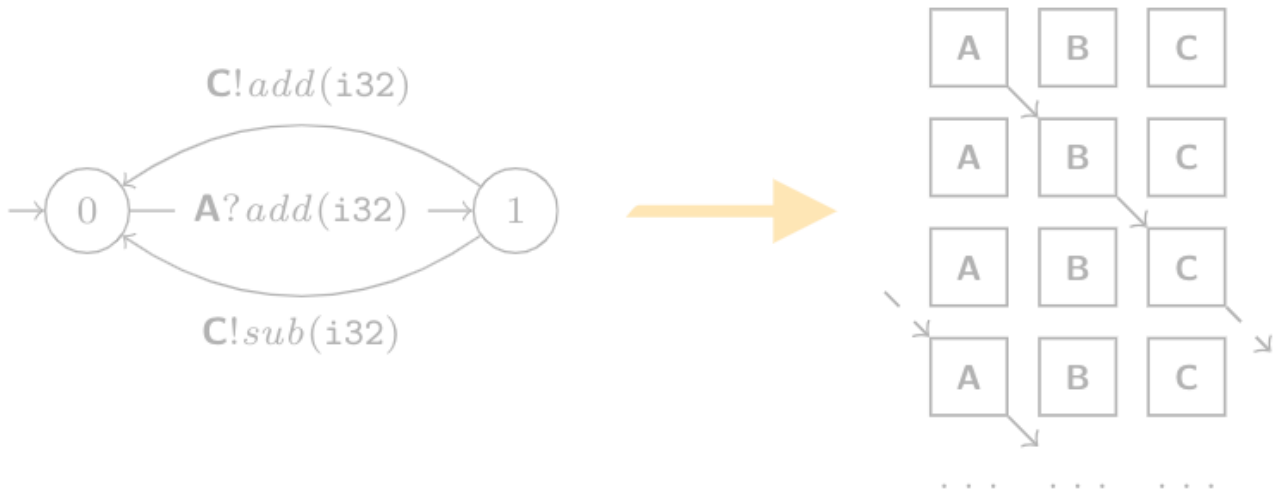


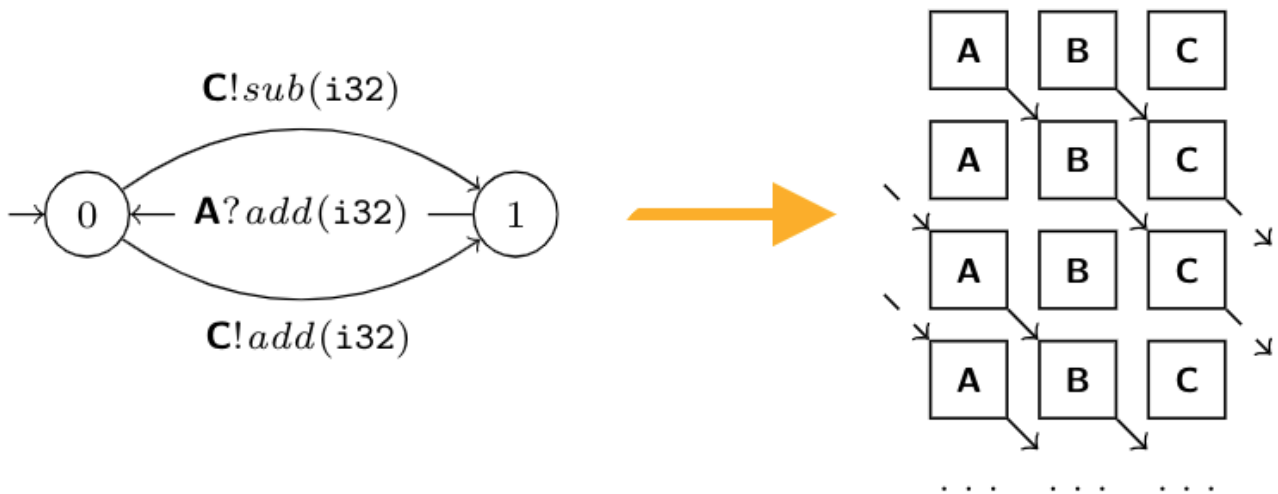
PROJECTION



Ring Protocol

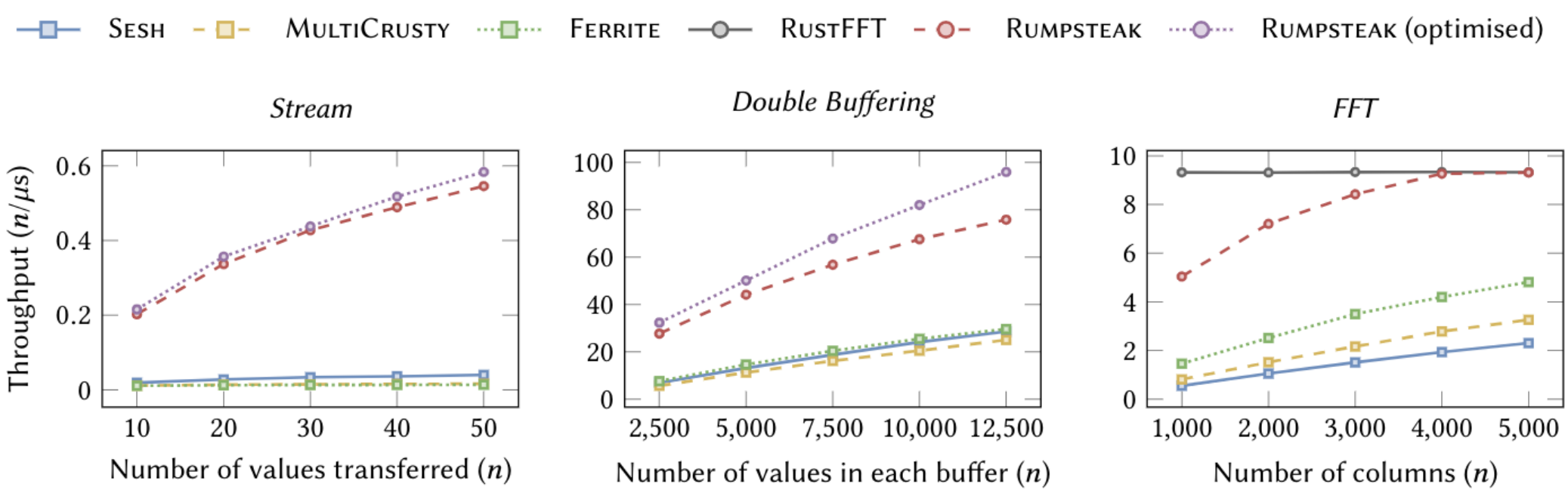
Example





Evaluation

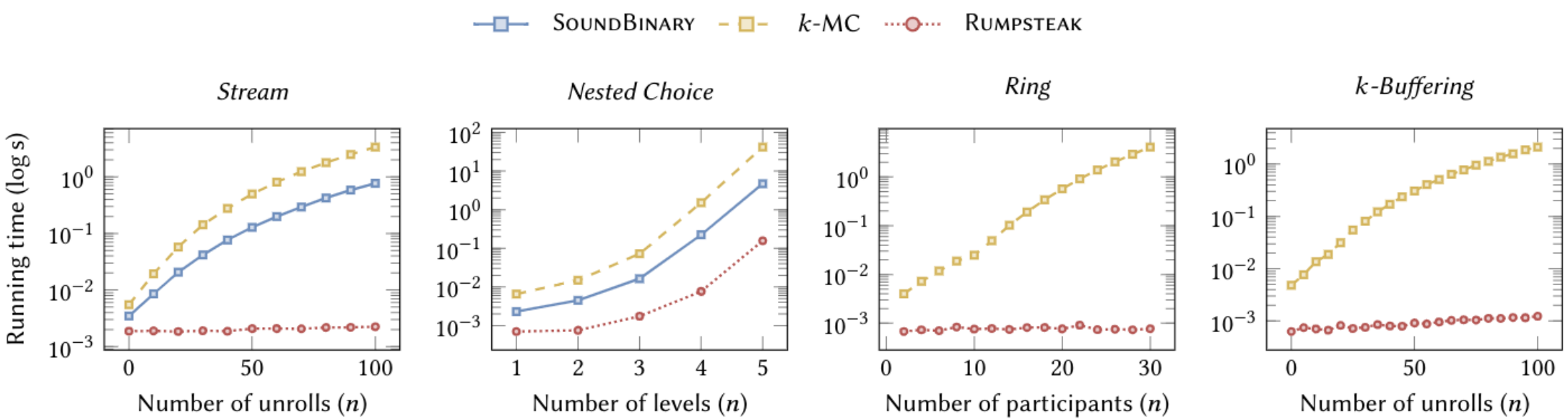
Rust Framework Benchmarks



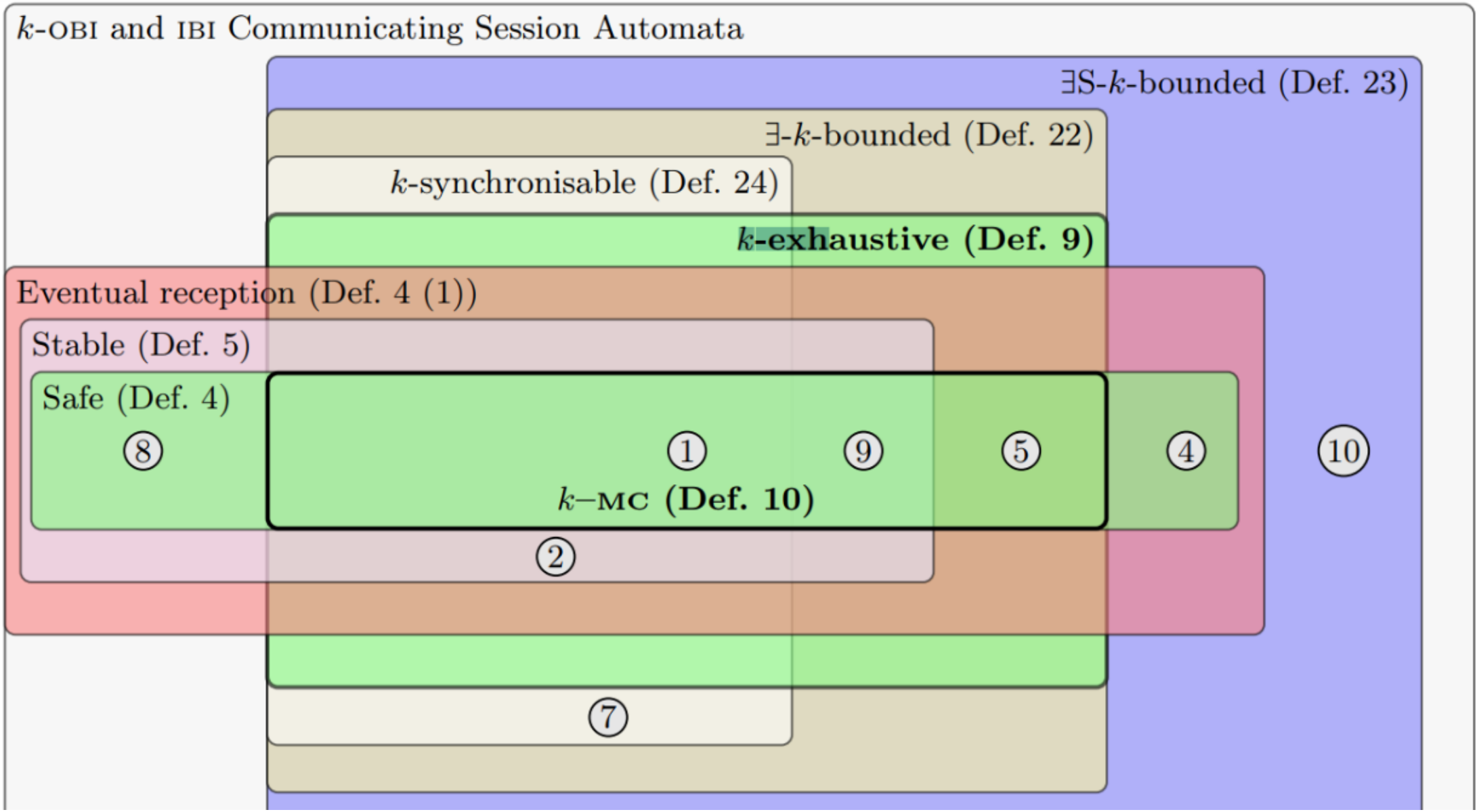
16-core AMD OpteronTM 6200 Series CPU @ 2.6GHz with hyperthreading, 128GB of RAM, Ubuntu 18.04.5 LTS and Rust Nightly 2021-07-06.

Evaluation

Asynchronous Reordering Benchmarks



k-Multiparty Compatibility [CAV'19]



Communicating Automata & Session Types

Asynchronous Subtyping Sound Algorithms [Bravetti, Lange and Zavattaro LMCS21, FoSSaCS'21]

Global Analysis (Bottom Up) k-MC [Langue and NY 19], Model-checking [Scalas & NY'19, Barwell et al 22]

Higher-Order Message Sequent Charts and Global Types (Top-Down)

Complete Multiparty Session Type Projection with Automata [CAV'23, Li et al.] [ECOOP'23, Stutz]

Problem: A gap between end-point types and processes [Scalas and NY'19]

History of Session Types with mCRL2

- [TACAS'17] Using mCRL2 to benchmark **session subtyping** checking
- [POPL'17] Analysing properties (e.g., deadlock-freedom) of **Go** programs
- [ICSE'18] Analysing properties of **message-passing Go** programs
- [POPL'19] Analysing properties of **synchronous multiparty session pi-calculus**
- [PLDI'19] Analysing properties of **distributed Scala programs**
- [ECOOP'20] Analysing properties of **shared memory Go** programs
- [CONCUR'22] Extension of [POPL'19] with **Stop-Failure**

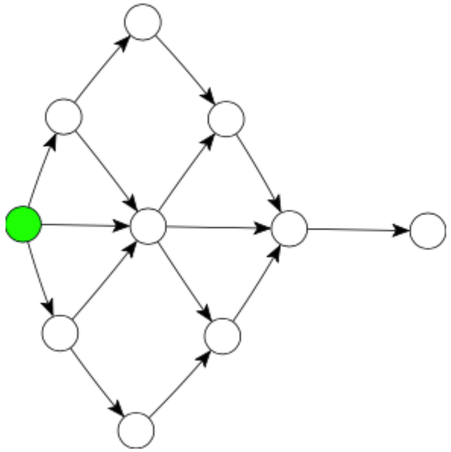
History of Session Types with mCRL2

- [TACAS'17] Using mCRL2 to benchmark **session subtyping** checking
- [POPL'17] Analysing properties (e.g., deadlock-freedom) of **Go** programs
- [ICSE'18] Analysing properties of **message-passing Go** programs
- [POPL'19] Analysing properties of **synchronous multiparty session pi-calculus**
- [PLDI'19] Analysing properties of **distributed Scala programs**
- [ECOOP'20] Analysing properties of **shared memory Go** programs
- [CONCUR'22] Extension of [POPL'19] with **Stop-Failure**

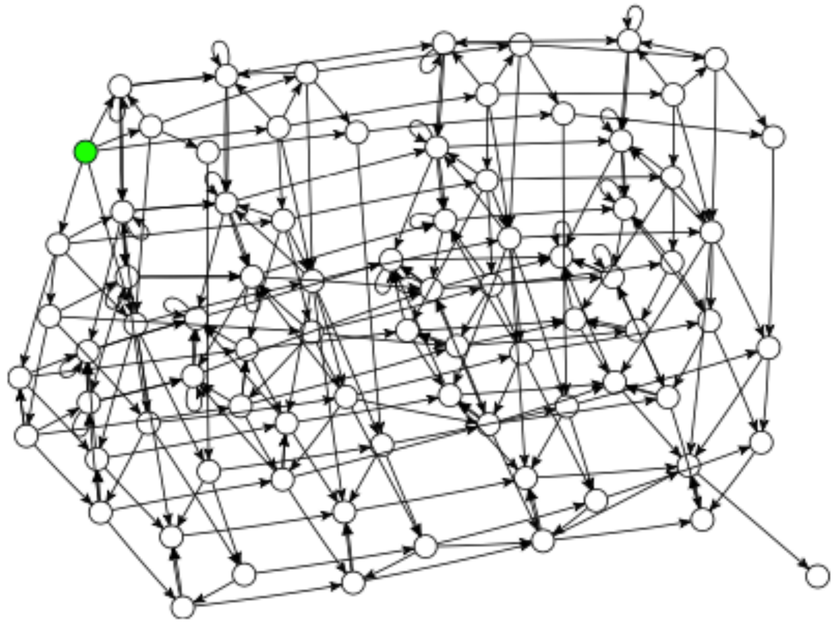
No gap between type and process properties

Why mCRL2?

- Can analyse properties than the top-down approach
- Can scale



No Failure



with Failure

Type-Level Model Checking is Practical

example	states	transitions	safe	df	live	nterm	term
DNS	101	427	$12.28 \pm 1\%$	$17.14 \pm 1\%$	$11.24 \pm 1\%$	$15.47 \pm 0\%$	$12.33 \pm 0\%$
Adder	37	159	$12.43 \pm 0\%$	$15.74 \pm 0\%$	$12.24 \pm 1\%$	$14.46 \pm 0\%$	$12.06 \pm 1\%$
TwoBuyers	1409	10248	$45.6 \pm 0\%$	$88.26 \pm 0\%$	$31.33 \pm 0\%$	$77.2 \pm 0\%$	$45.65 \pm 0\%$
Negotiate	1089	8106	$34.61 \pm 0\%$	$55.07 \pm 0\%$	$25.69 \pm 0\%$	$47.46 \pm 0\%$	$26.04 \pm 0\%$
Broadcast	161	925	$17.99 \pm 1\%$	$28.13 \pm 0\%$	$14.08 \pm 0\%$	$25.72 \pm 1\%$	$17.74 \pm 0\%$

- Property checking takes at $<$ 100ms on our example suite
- Our prototype tool, MPSTK, is available on GitHub at `https://github.com/alcestes/mpstk-crash-stop`

Type-Level Model Checking is Practical

example	states	transitions	safe	df	live	nterm	term
DNS	101	427	$12.28 \pm 1\%$	$17.14 \pm 1\%$	$11.24 \pm 1\%$	$15.47 \pm 0\%$	$12.33 \pm 0\%$
Adder	37	159	$12.43 \pm 0\%$	$15.74 \pm 0\%$	$12.24 \pm 1\%$	$14.46 \pm 0\%$	$12.06 \pm 1\%$
TwoBuyers	1409	10248	$45.6 \pm 0\%$	$88.26 \pm 0\%$	$31.33 \pm 0\%$	$77.2 \pm 0\%$	$45.65 \pm 0\%$
Negotiate	1089	8106	$34.61 \pm 0\%$	$55.07 \pm 0\%$	$25.69 \pm 0\%$	$47.46 \pm 0\%$	$26.04 \pm 0\%$
Broadcast	161	925	$17.99 \pm 1\%$	$28.13 \pm 0\%$	$14.08 \pm 0\%$	$25.72 \pm 1\%$	$17.74 \pm 0\%$

- Property checking takes at $<$ 100ms on our example suite
- Our prototype tool, MPSTK, is available on GitHub at `https://github.com/alcestes/mpstk-crash-stop`

Type-Level Model Checking is Practical

example	states	transitions	safe	df	live	nterm	term
DNS	101	427	$12.28 \pm 1\%$	$17.14 \pm 1\%$	$11.24 \pm 1\%$	$15.47 \pm 0\%$	$12.33 \pm 0\%$
Adder	37	159	$12.43 \pm 0\%$	$15.74 \pm 0\%$	$12.24 \pm 1\%$	$14.46 \pm 0\%$	$12.06 \pm 1\%$
TwoBuyers	1409	10248	$45.6 \pm 0\%$	$88.26 \pm 0\%$	$31.33 \pm 0\%$	$77.2 \pm 0\%$	$45.65 \pm 0\%$
Negotiate	1089	8106	$34.61 \pm 0\%$	$55.07 \pm 0\%$	$25.69 \pm 0\%$	$47.46 \pm 0\%$	$26.04 \pm 0\%$
Broadcast	161	925	$17.99 \pm 1\%$	$28.13 \pm 0\%$	$14.08 \pm 0\%$	$25.72 \pm 1\%$	$17.74 \pm 0\%$

- Property checking takes at $<$ 100ms on our example suite
- Our prototype tool, MPSTK, is available on GitHub at `https://github.com/alcestes/mpstk-crash-stop`

mCRL2 improvement [POPL'19]

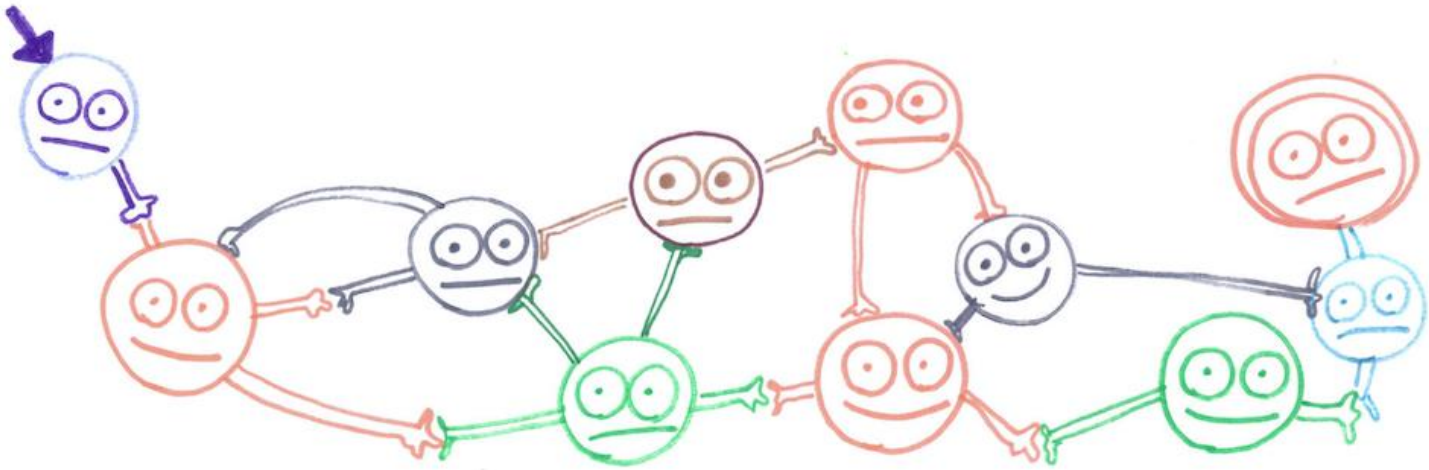
Table 2. Average time (in seconds ± std. dev.) for the verification of the protocols in Fig. 4. Protocols (3) and (4) are instantiated with $n = 3$. The outcome of the verification is shown in Table 1. *(Benchmarking specs: Intel Core i7-4790 CPU, 3.60GHz, 16 GB RAM, mCRL2 201808.0 invoked 30 times (by* `mpstk`*) with:* `pbes2bool --strategy=2`*)*

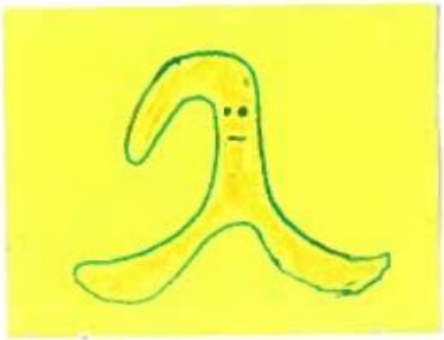
	states	safe	deadlock-free	live	live$^+$	live^{++}	never-terminat.	terminat.
(1) OAuth2 fragment	37	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	0.98 ± 9%
(2) Rec. two-buyers	85	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	0.99 ± 3%
(3) Rec. map/reduce	2561	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	1.00 ± 0%	0.99 ± 3%
(4) MP workers	442369	1.01 ± 4%	0.98 ± 8%	0.98 ± 9%	1.03 ± 14%	1.02 ± 7%	0.99 ± 6%	1.00 ± 1%

Session Types and Beyond

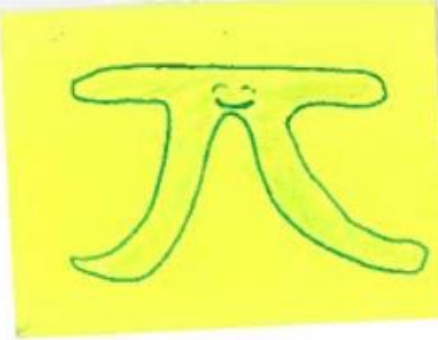
- What is 'Session Types'? Semantics (Behavioural Equivalences) and Expressiveness
- Mechanisations (Coq, Isabelle, etc)
- Fault Tolerance
- Security and Cryptography
- Probability
- Complexity (Computations and Verification)
- Applications: Programming Languages & Open Systems

Discussions





System F



Session

$M \simeq N$

$[\![M]\!] \simeq [\![N]\!]$

$[\![P]\!] \simeq [\![Q]\!]$

$P \simeq Q$

Reverse

My Observations

- Successful Integrations of Session Types to (Mainstream) Programming Languages and Libraries
- Use Case Driven: A Family of Session Types