# on sessions, linear logic and unbounded interactions

Luca Padovani, University of Camerino

Forum Numerica, 12 jan 2023

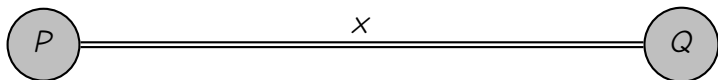# outline

# outline

# Q: what is a **session**?

A **session** is a private communication channel linking two processes, each using one **channel endpoint** according to a protocol specification called **session type**



Examples of session types
- First send integer, then receive string (*P*'s viewpoint)
- First receive integer, then send string (*Q*'s viewpoint)

# Q: what is a session type system useful for?

A: enable the **compositional static analysis** of distributed programs

If $P$ uses $\overbrace{x_1, \ldots, x_n}^{\text{channels}}$ as described by $\overbrace{A_1, \ldots, A_n}^{\text{protocols}}$

$$P \vdash x_1 : A_1, \ldots, x_n : A_n$$

then

- exchanged messages have the **expected type** **(safety)**
- interactions occur in the **expected order** **(fidelity)**
- interactions **don't get stuck** **(deadlock freedom)**
- pending actions are completed **(livelock freedom)**
- interactions terminate **(termination)**
- …

# Q: how do we define a session type system?

A: from linear logic and its extensions

| | | |
|---|---|---|
| linear logic propositions | $\Longleftrightarrow$ | session types |
| linear logic proofs | $\Longleftrightarrow$ | well-typed processes |
| cut reduction | $\Longleftrightarrow$ | communication |
| soundness of the logic | $\Longleftrightarrow$ | soundness of typing |

# Q: how do we define a session type system?

A: from linear logic and its extensions

| linear logic propositions | $\Longleftrightarrow$ | session types |
| linear logic proofs | $\Longleftrightarrow$ | well-typed processes |
| cut reduction | $\Longleftrightarrow$ | communication |
| soundness of the logic | $\Longleftrightarrow$ | soundness of typing |
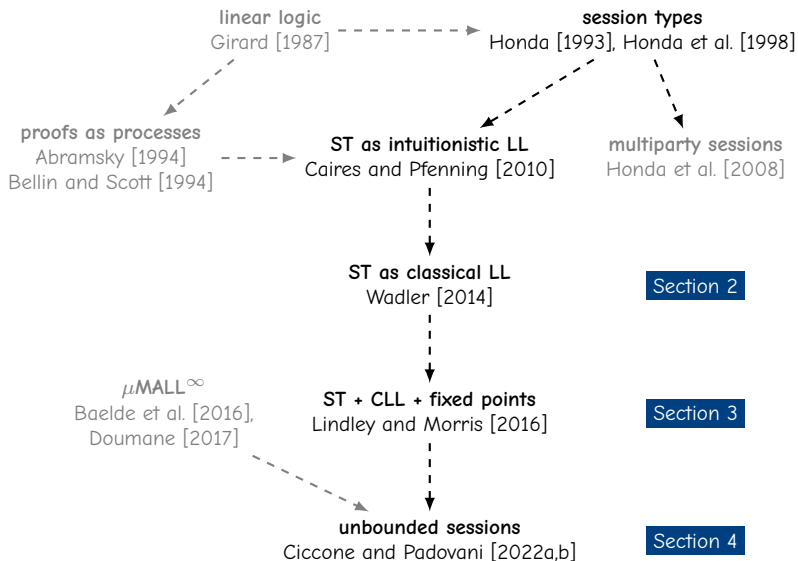
# outline

# MALL propositions as session types

$$
\begin{array}{rcll}
A, B & ::= & \mathbf{0} & \text{unit for } \oplus \\
& | & \top & \text{unit for } \& \\
& | & \mathbf{1} & \text{send termination signal} \\
& | & \bot & \text{receive termination signal} \\
& | & A \oplus B & \text{select either } A \text{ or } B \\
& | & A \,\&\, B & \text{offer choice of } A \text{ or } B \\
& | & A \otimes B & \text{output } A \text{ then behave as } B \\
& | & A \,\rotatebox[origin=c]{180}{\&}\, B & \text{input } A \text{ then behave as } B
\end{array}
$$

Example: session types for sending/receiving a boolean

$$
\mathbb{B} \stackrel{\text{def}}{=} \mathbf{1} \oplus \mathbf{1} \qquad\qquad \mathbb{B}^{\bot} \stackrel{\text{def}}{=} \bot \,\&\, \bot
$$

# MALL proof rules

no rule for 0

$$\overline{\vdash \Gamma, \top}$$

$$\overline{\vdash 1}$$

$$\frac{\vdash \Gamma}{\vdash \Gamma, \bot}$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B} \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B} \qquad \frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B} \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \,\parr\, B}$$

$$\overline{\vdash A, A^{\bot}} \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^{\bot}}{\vdash \Gamma, \Delta}$$

# cut rule = parallel composition + restriction

proof rule
$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}$$

typing rule
$$\frac{\vdash \Gamma, x : A \qquad \vdash \Delta, x : A^{\perp}}{\vdash \Gamma, \Delta}$$

# cut rule = parallel composition + restriction

proof rule
$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}$$

typing rule
$$\frac{P \vdash \Gamma, x : A \qquad Q \vdash \Delta, x : A^{\perp}}{\mathsf{new}\, x(P \mid Q) \vdash \Gamma, \Delta}$$

# rules for **1** and ⊥ = termination

$$\frac{}{\vdash x : \mathbf{1}} \qquad\qquad \frac{\vdash \Gamma}{\vdash \Gamma, x : \bot}$$

# rules for **1** and $\perp$ = termination

$$\frac{}{\mathsf{close}\,x \vdash x : \mathbf{1}}$$

$$\frac{P \vdash \Gamma}{\mathsf{wait}\,x.P \vdash \Gamma, x : \perp}$$

# rules for **1** and $\perp$ = termination

$$\frac{}{\mathsf{close}\,x \vdash x : \mathbf{1}}$$

$$\frac{P \vdash \Gamma}{\mathsf{wait}\,x.P \vdash \Gamma, x : \perp}$$

$$\frac{\dfrac{}{\mathsf{close}\,x \vdash x : \mathbf{1}} \quad \dfrac{P \vdash \Gamma}{\mathsf{wait}\,x.P \vdash \Gamma, x : \perp}}{\mathsf{new}\,x(\mathsf{close}\,x \mid \mathsf{wait}\,x.P) \vdash \Gamma} \qquad \rightarrow \qquad P \vdash \Gamma$$

Note: principal cut reduction defines process reduction

# rules for $\oplus$ and $\&$ = branch selection

$$\frac{\vdash \Gamma, x : A}{\vdash \Gamma, x : A \oplus B}$$

$$\frac{\vdash \Gamma, x : A \qquad \vdash \Gamma, x : B}{\vdash \Gamma, x : A \& B}$$

# rules for $\oplus$ and $\&$ = branch selection

$$\frac{P \vdash \Gamma, x : A}{\mathsf{left}\, x.P \vdash \Gamma, x : A \oplus B}$$

$$\frac{P \vdash \Gamma, x : A \qquad Q \vdash \Gamma, x : B}{\mathsf{case}\, x\{P, Q\} \vdash \Gamma, x : A \,\&\, B}$$

# rules for $\oplus$ and $\&$ = branch selection

$$\frac{P \vdash \Gamma, x : A}{\text{left}\, x.P \vdash \Gamma, x : A \oplus B}$$

$$\frac{P \vdash \Gamma, x : A \qquad Q \vdash \Gamma, x : B}{\text{case}\, x\{P, Q\} \vdash \Gamma, x : A \,\&\, B}$$

$$\frac{\dfrac{P \vdash \Gamma, x : A}{\text{left}\, x.P \vdash \Gamma, x : A \oplus B} \qquad \dfrac{Q \vdash \Delta, x : A^{\perp} \qquad R \vdash \Delta, x : B^{\perp}}{\text{case}\, x\{Q, R\} \vdash \Delta, x : A^{\perp} \,\&\, B^{\perp}}}{\text{new}\, x(\text{left}\, x.P \mid \text{case}\, x\{Q, R\}) \vdash \Gamma, \Delta}$$

$$\downarrow$$

$$\frac{P \vdash \Gamma, x : A \qquad Q \vdash \Delta, x : A^{\perp}}{\text{new}\, x(P \mid Q) \vdash \Gamma, \Delta}$$

# rules for $\otimes$ and $\bindnasrepma$ = channel communication

$$\frac{P \vdash \Gamma, y : A \qquad Q \vdash \Delta, x : B}{x[y].(P \mid Q) \vdash \Gamma, \Delta, x : A \otimes B} \qquad \frac{P \vdash \Gamma, y : A, x : B}{x(y).P \vdash \Gamma, x : A \bindnasrepma B}$$

$$\frac{\dfrac{P \vdash \Gamma, y : A \qquad Q \vdash \Gamma', x : B}{x[y].(P \mid Q) \vdash \Gamma, \Gamma', x : A \otimes B} \qquad \dfrac{R \vdash \Delta, y : A^{\perp}, x : B^{\perp}}{x(y).R \vdash \Delta, x : A^{\perp} \bindnasrepma B^{\perp}}}{\mathsf{new}\, x(x[y].(P \mid Q) \mid x(y).R) \vdash \Gamma, \Gamma', \Delta}$$

$$\downarrow$$

$$\frac{P \vdash \Gamma, y : A \qquad \dfrac{Q \vdash \Gamma', x : B \qquad R \vdash \Delta, y : A^{\perp}, x : B^{\perp}}{\mathsf{new}\, x(Q \mid R) \vdash \Gamma', \Delta, y : A^{\perp}}}{\mathsf{new}\, y(P \mid \mathsf{new}\, x(Q \mid R)) \vdash \Gamma, \Gamma', \Delta}$$

# example: negation of a boolean value

$$\mathsf{Neg}(x, y) \triangleq \mathsf{case}\, x\{\mathsf{wait}\, x.\mathsf{right}\, y.\mathsf{close}\, y,$$
$$\mathsf{wait}\, x.\mathsf{left}\, y.\mathsf{close}\, y\}$$

$$\cfrac{\cfrac{\cfrac{\overline{\mathsf{close}\, y \vdash y : \mathbf{1}}}{\mathsf{right}\, y.\mathsf{close}\, y \vdash y : \mathbb{B}}}{\mathsf{wait}\, x.\mathsf{right}\, y.\mathsf{close}\, y \vdash x : \bot, y : \mathbb{B}} \quad \cfrac{\cfrac{\overline{\mathsf{close}\, y \vdash y : \mathbf{1}}}{\mathsf{left}\, y.\mathsf{close}\, y \vdash y : \mathbb{B}}}{\mathsf{wait}\, x.\mathsf{left}\, y.\mathsf{close}\, y \vdash x : \bot, y : \mathbb{B}}}{\cfrac{\mathsf{case}\, x\{\mathsf{wait}\, x.\mathsf{right}\, y.\mathsf{close}\, y, \mathsf{wait}\, x.\mathsf{left}\, y.\mathsf{close}\, y\} \vdash x : \mathbb{B}^{\bot}, y : \mathbb{B}}{\mathsf{Neg}\langle x, y\rangle \vdash x : \mathbb{B}^{\bot}, y : \mathbb{B}}}$$

# properties of proofs and processes

## Cut elimination
The cut rule is **admissible**

## Deadlock freedom
If $P \vdash \Gamma$ then either $P \rightarrow$ or $P$ is not a cut ☺

## Successful termination
If $P \vdash x : \mathbf{1}$ then $P \Rightarrow \mathsf{close}\, x$ ☺

## Livelock freedom
If $P \vdash x : \mathbf{1}$ then $P$ is livelock free ☺

# observations

## Fact
MALL propositions allow us to describe **finite** protocols made of a **fixed** number of actions

## Role of the exponential modalities ?$A$ and !$A$
- useful to describe the behavior of **clients** and **servers**
- !$A$ = server providing $A$
- ?$A$ = clients requesting $A$

## Moral
The correspondence holds well, but each interaction between a client and a server is still of **fixed length**

# outline

# extending MALL with fixed points

[Lindley and Morris, 2016, Baelde et al., 2016, Doumane, 2017]

$$
\begin{array}{llll}
A, B & ::= & \cdots & \text{as before} \\
& & X & \text{recursion variable} \\
& & \mu X.A & \text{least fixed point} \\
& & \nu X.A & \text{greatest fixed point}
\end{array}
$$

Example: session types for sending/receiving a natural number

$$
\mathbb{N} \stackrel{\text{def}}{=} \mu X.(X \oplus \mathbf{1}) \qquad \mathbb{N}^{\perp} \stackrel{\text{def}}{=} \nu X.(X \,\&\, \perp)
$$

# rules for fixed points in $\mu$MALL$^\infty$

[Baelde et al., 2016, Doumane, 2017]

$$\frac{P \vdash \Gamma, x : A\{\nu X.A/X\}}{\mathsf{corec}\, x.P \vdash \Gamma, x : \nu X.A} \qquad \frac{P \vdash \Gamma, x : A\{\mu X.A/X\}}{\mathsf{rec}\, x.P \vdash \Gamma, x : \mu X.A}$$

- there is no distinction between least and greatest fixed points in these rules (but they **must** be distinguished at some point)
- $\mathsf{rec}\, x$ and $\mathsf{corec}\, x$ are used to **unfold** the type of the channel $x$

# example: successor of a natural number

$\text{Succ}(x, y) \triangleq \text{corec}\, x.\text{rec}\, y.\text{case}\, x\{\, \text{left}\, y.\text{Succ}\langle x, y \rangle,$
$\text{wait}\, x.\text{left}\, y.\text{rec}\, y.\text{right}\, y.\text{close}\, y \}$

go left once more before going right

# example: successor of a natural number



$$\cfrac{\text{close}\, y \vdash y : \mathbf{1}}{\cfrac{\text{right}\, y.\text{close}\, y \vdash y : \mathbb{N} \oplus \mathbf{1}}{\cfrac{\text{rec}\, y \ldots \vdash y : \mathbb{N}}{\text{left}\, y \ldots \vdash y : \mathbb{N} \oplus \mathbf{1}}}}$$

we need an infinite proof!

$$\vdots$$

$$\cfrac{\cfrac{\text{Succ}\langle x, y\rangle \vdash x : \mathbb{N}^{\perp}, y : \mathbb{N}}{\text{left}\, y.\text{Succ}\langle x, y\rangle \vdash x : \mathbb{N}^{\perp}, y : \mathbb{N} \oplus \mathbf{1}} \quad \cfrac{}{\text{wait}\, x \ldots \vdash x : \perp, y : \mathbb{N} \oplus \mathbf{1}}}{\cfrac{\text{case}\, x\{\ldots, \ldots\} \vdash x : \mathbb{N}^{\perp} \,\&\, \perp, y : \mathbb{N} \oplus \mathbf{1}}{\cfrac{\text{rec}\, y \ldots \vdash x : \mathbb{N}^{\perp} \,\&\, \perp, y : \mathbb{N}}{\cfrac{\text{corec}\, x \ldots \vdash x : \mathbb{N}^{\perp}, y : \mathbb{N}}{\text{Succ}\langle x, y\rangle \vdash x : \mathbb{N}^{\perp}, y : \mathbb{N}}}}}$$

# some infinite proofs are dangerous

$$\Omega \triangleq \Omega \qquad \frac{\dfrac{\vdots}{\Omega \vdash \Gamma}}{\Omega \vdash \Gamma}$$

Allowing arbitrary proofs compromises the soundness of the logic

## Proofs
- the cut rule is no longer admissible ☹
- the empty sequent and $\vdash \mathbf{0}$ become derivable ☹

## Processes
- safety properties still hold ☺
- liveness properties are lost ☹

# identifying valid proofs

[Baelde et al., 2016, Doumane, 2017]

## Valid infinite branch

An infinite branch of a proof is **valid** if there is a channel $x$ whose type is a **greatest fixed point** that is unfolded **infinitely many times**

## Valid proof

A proof is **valid** if every infinite branch in it is valid

## Intuition

- least fixed points **can** be unfolded finitely many times only
- greatest fixed points **must** be unfolded infinitely many times

## Warning: this is an oversimplified approximation

The exact definition of valid proof is technically more involved

# example: successor of a natural number



$$\frac{}{\text{close } y \vdash y : \mathbf{1}}$$

$$\frac{}{\text{right } y.\text{close } y \vdash y : \mathbb{N} \oplus \mathbf{1}}$$

$$\frac{}{\text{rec } y \ldots \vdash y : \mathbb{N}}$$

$$\vdots$$

$$\frac{}{\text{Succ}\langle x, y \rangle \vdash x : \mathbb{N}^{\perp}, y : \mathbb{N}} \qquad \frac{}{\text{left } y \ldots \vdash y : \mathbb{N} \oplus \mathbf{1}}$$

$$\frac{}{\text{left } y.\text{Succ}\langle x, y \rangle \vdash x : \mathbb{N}^{\perp}, y : \mathbb{N} \oplus \mathbf{1}} \qquad \frac{}{\text{wait } x \ldots \vdash x : \perp, y : \mathbb{N} \oplus \mathbf{1}}$$

$$\frac{}{\text{case } x\{\ldots, \ldots\} \vdash x : \mathbb{N}^{\perp} \,\&\, \perp, y : \mathbb{N} \oplus \mathbf{1}}$$

$$\frac{}{\text{rec } y \ldots \vdash x : \mathbb{N}^{\perp} \,\&\, \perp, y : \mathbb{N}}$$

$$\frac{}{\text{corec } x \ldots \vdash x : \mathbb{N}^{\perp}, y : \mathbb{N}}$$

$$\frac{}{\text{Succ}\langle x, y \rangle \vdash x : \mathbb{N}^{\perp}, y : \mathbb{N}}$$

# properties of valid proofs

In a valid proof **every** sequence of (principal) cut reductions is **finite** and the cut rule is **admissible** (in the limit)

Good news
All well-typed processes terminate ☺

**Termination** is a valuable property that entails **livelock freedom** when combined with **deadlock freedom**

Bad news
All well-typed processes terminate ☹

This is unfortunate since all (interesting) processes that engage into arbitrarily long (unbounded) interactions are ill typed

# properties of valid proofs

In a valid proof **every** sequence of (principal) cut reductions is **finite** and the cut rule is **admissible** (in the limit)

## Good news
All well-typed processes terminate ☺

**Termination** is a valuable property that entails **livelock freedom** when combined with **deadlock freedom**

## Bad news
All well-typed processes terminate ☹

This is unfortunate since all (interesting) processes that engage into arbitrarily long (unbounded) interactions are ill typed

# examples

## We can
Model a process that computes the successor of an arbitrary (but **specific**) natural number

## We cannot
Model a process that computes the successor of an arbitrary (**non-deterministically chosen**) natural number

## We can
Model a process that buys an arbitrary (but **specific**) number of items from a store

## We cannot
Model a process that buys an arbitrary (**non-deterministically chosen**) number of items from a store

# examples

**We can**
Model a process that computes the successor of an arbitrary (but **specific**) natural number

**We can**
Model a process that buys an arbitrary (but **specific**) number of items from a store

**We cannot**
Model a process that computes the successor of an arbitrary (**non-deterministically chosen**) natural number

**We cannot**
Model a process that buys an arbitrary (**non-deterministically chosen**) number of items from a store

# outline

# objective

Relax the type system so that (some) unbounded interactions can be modeled and livelock freedom is still guaranteed
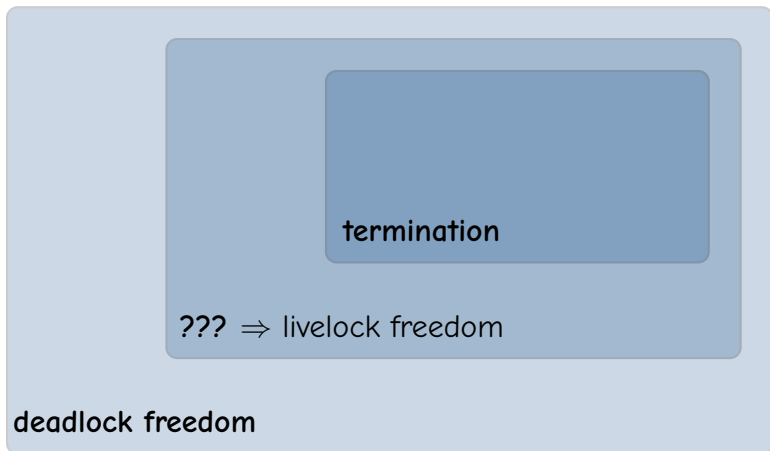
# from termination to **fair termination**

> run = maximal reduction sequence

### Termination
A process is **terminating** of all of its runs are finite

### Weak termination
A process is **weakly terminating** if at least one of its runs is finite

### Fair termination [Grumberg et al., 1984, Francez, 1986]
A process is **fairly terminating** if all of its **fair runs** are finite

- **In principle**, a fairly terminating process may diverge
- **In practice**, a fairly terminating process always terminates

# example: natural number generator

$$\mathsf{Nat}(x) \triangleq \mathsf{rec}\, x.(\mathsf{left}\, x.\mathsf{Nat}\langle x \rangle \oplus \mathsf{right}\, x.\mathsf{close}\, x)$$

non-deterministic choice

The run in which the process keeps going left is "unfair"

# fairness assumption(s)

**Dozens** of fairness assumptions have been studied in the literature, see van Glabbeek and Höfner [2019] for a recent survey

## In this talk
A run $P_0 \to P_1 \to P_2 \to \cdots$ is **fair** if it goes through finitely many weakly terminating processes

## Intuition
In an **unfair run** the process has infinitely many opportunities to terminate, but it takes none of them

## An example from real life: online shopping
The run in which you keep adding items into the cart is unfair

# example: natural number generator



$$
\frac{
\frac{
\frac{\vdots}{\mathsf{Nat}\langle x\rangle \vdash x : \mathbb{N}}
}{\mathsf{left}\,x.\mathsf{Nat}\langle x\rangle \vdash x : \mathbb{N}\oplus\mathbf{1}}
\qquad
\frac{
\frac{}{\mathsf{close}\,x \vdash x : \mathbf{1}}
}{\mathsf{right}\,x.\mathsf{close}\,x \vdash x : \mathbb{N}\oplus\mathbf{1}}
}{
\frac{
\frac{\mathsf{left}\,x.\mathsf{Nat}\langle x\rangle \oplus \mathsf{right}\,x.\mathsf{close}\,x \vdash x : \mathbb{N}\oplus\mathbf{1}}{\mathsf{rec}\,x.(\mathsf{left}\,x.\mathsf{Nat}\langle x\rangle \oplus \mathsf{right}\,x.\mathsf{close}\,x) \vdash x : \mathbb{N}}
}{\mathsf{Nat}\langle x\rangle \vdash x : \mathbb{N}}
}
$$

The infinite branch is **invalid**

# example: natural number generator



$$\frac{\vdots}{\mathsf{Nat}\langle x\rangle \vdash x : \mathbb{N}}$$

$$\frac{\mathsf{Nat}\langle x\rangle \vdash x : \mathbb{N}}{\mathsf{left}\,x.\mathsf{Nat}\langle x\rangle \vdash x : \mathbb{N} \oplus \mathbf{1}}$$

$$\frac{}{\mathsf{close}\,x \vdash x : \mathbf{1}}$$

$$\frac{\mathsf{close}\,x \vdash x : \mathbf{1}}{\mathsf{right}\,x.\mathsf{close}\,x \vdash x : \mathbb{N} \oplus \mathbf{1}}$$

$$\frac{\mathsf{left}\,x.\mathsf{Nat}\langle x\rangle \oplus \mathsf{right}\,x.\mathsf{close}\,x \vdash x : \mathbb{N} \oplus \mathbf{1}}{\mathsf{rec}\,x.(\mathsf{left}\,x.\mathsf{Nat}\langle x\rangle \oplus \mathsf{right}\,x.\mathsf{close}\,x) \vdash x : \mathbb{N}}$$

$$\mathsf{Nat}\langle x\rangle \vdash x : \mathbb{N}$$

The infinite branch is **invalid**, but it corresponds to an unfair run

# a small refinement to the notion of valid proof

## Fair branch

A branch of a proof is **fair** if it goes through finitely many weakly terminating processes

## Fairly valid proof

A proof is **fairly valid** if every **fair** infinite branch in it is valid

# properties of fairly valid proofs

In a **fairly** valid proof every **fair** sequence of (principal) cut reductions is **finite** and the cut rule is **admissible** (in the limit)

## Consequences

- All well-typed processes **fairly terminate**  ☺
- **Fair termination** entails **livelock freedom**  ☺
- At least some (interesting) processes that engage into arbitrarily long (unbounded) interactions are well typed  ☺

# outline

# wrap up

*Session type theory enjoys a natural correspondence (in the style of Curry-Howard) with linear logic*

Selection of further developments

- beyond **tree-like** network topologies [Dardha and Gay, 2018]
- beyond **race-free** interactions [Balzer and Pfenning, 2017, Balzer et al., 2019, Rocha and Caires, 2021, Qian et al., 2021]
- support for **general recursion** [Ciccone and Padovani, 2022a, Ciccone et al., 2022]

# wrap up

*Session type theory enjoys a natural correspondence (in the style of Curry-Howard) with linear logic*

Selection of further developments

- beyond **tree-like** network topologies [Dardha and Gay, 2018]
- beyond **race-free** interactions [Balzer and Pfenning, 2017, Balzer et al., 2019, Rocha and Caires, 2021, Qian et al., 2021]
- support for **general recursion** [Ciccone and Padovani, 2022a, Ciccone et al., 2022]

## thank you!

# references

Samson Abramsky. Proofs as processes. *Theor. Comput. Sci.*, 135(1):5–9, 1994. 📄

David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory: the multiplicative additive case. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 42:1–42:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. 📄

Stephanie Balzer and Frank Pfenning. Manifest sharing with session types. *Proc. ACM Program. Lang.*, 1(ICFP):37:1–37:29, 2017. 📄

# references (cont.)

Stephanie Balzer, Bernardo Toninho, and Frank Pfenning. *Manifest deadlock-freedom for shared session types.* In Luís Caires, editor, *Programming Languages and Systems - 28th European Symposium on Programming, ESOP 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11423 of *Lecture Notes in Computer Science*, pages 611–639. Springer, 2019. 🗎

Gianluigi Bellin and Philip J. Scott. *On the pi-calculus and linear logic.* *Theor. Comput. Sci.*, 135(1):11–65, 1994. 🗎

Luís Caires and Frank Pfenning. *Session types as intuitionistic linear propositions.* In Paul Gastin and François Laroussinie, editors, *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, volume 6269 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2010. 🗎

# references (cont.)

Luca Ciccone and Luca Padovani. Fair termination of binary sessions. *Proc. ACM Program. Lang.*, 6(POPL):1–30, 2022a. 📄

Luca Ciccone and Luca Padovani. An infinitary proof theory of linear logic ensuring fair termination in the linear $\pi$-calculus. In Bartek Klin, Slawomir Lasota, and Anca Muscholl, editors, *33rd International Conference on Concurrency Theory, CONCUR 2022, September 12-16, 2022, Warsaw, Poland*, volume 243 of *LIPIcs*, pages 36:1–36:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022b. 📄

Luca Ciccone, Francesco Dagnino, and Luca Padovani. Fair termination of multiparty sessions. In Karim Ali and Jan Vitek, editors, *36th European Conference on Object-Oriented Programming, ECOOP 2022, June 6-10, 2022, Berlin, Germany*, volume 222 of *LIPIcs*, pages 26:1–26:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. 📄

# references (cont.)

Ornela Dardha and Simon J. Gay. A new linear logic for deadlock-free session-typed processes. In Christel Baier and Ugo Dal Lago, editors, *Foundations of Software Science and Computation Structures - 21st International Conference, FOSSACS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, volume 10803 of *Lecture Notes in Computer Science*, pages 91–109. Springer, 2018. 🗎

Amina Doumane. *On the infinitary proof theory of logics with fixed points. (Théorie de la démonstration infinitaire pour les logiques à points fixes)*. PhD thesis, Paris Diderot University, France, 2017. URL `https://tel.archives-ouvertes.fr/tel-01676953`.

Nissim Francez. *Fairness*. Texts and Monographs in Computer Science. Springer, 1986. ISBN 978-3-540-96235-9. 🗎

Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987. 🗎

# references (cont.)

Orna Grumberg, Nissim Francez, and Shmuel Katz. Fair termination of communicating processes. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, PODC '84, pages 254–265, New York, NY, USA, 1984. Association for Computing Machinery. ISBN 0897911431.

Kohei Honda. Types for dyadic interaction. In Eike Best, editor, *CONCUR '93, 4th International Conference on Concurrency Theory, Hildesheim, Germany, August 23-26, 1993, Proceedings*, volume 715 of *Lecture Notes in Computer Science*, pages 509–523. Springer, 1993.

Kohei Honda, Vasco Thudichum Vasconcelos, and Makoto Kubo. Language primitives and type discipline for structured communication-based programming. In Chris Hankin, editor, *Programming Languages and Systems - ESOP'98, 7th European Symposium on Programming, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98,*

# references (cont.)

*Lisbon, Portugal, March 28 - April 4, 1998, Proceedings*, volume 1381 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 1998. 📄

Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In George C. Necula and Philip Wadler, editors, *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, pages 273–284. ACM, 2008. 📄

Sam Lindley and J. Garrett Morris. Talking bananas: structural recursion for session types. In Jacques Garrigue, Gabriele Keller, and Eijiro Sumii, editors, *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, ICFP 2016, Nara, Japan, September 18-22, 2016*, pages 434–447. ACM, 2016. 📄

Zesen Qian, G. A. Kavvos, and Lars Birkedal. Client-server sessions in linear logic. *Proc. ACM Program. Lang.*, 5(ICFP):1–31, 2021. 📄

# references (cont.)

Pedro Rocha and Luís Caires. Propositions-as-types and shared state. *Proc. ACM Program. Lang.*, 5(ICFP):1–30, 2021.

Rob van Glabbeek and Peter Höfner. Progress, justness, and fairness. *ACM Comput. Surv.*, 52(4):69:1–69:38, 2019.

Philip Wadler. Propositions as sessions. *J. Funct. Program.*, 24(2-3): 384–418, 2014.