fair termination of binary sessions

Luca Padovani, Università di Torino (joint work with Luca Ciccone)

22nd Italian Conference on Theoretical Computer Science, 13 Sep 2021

outline

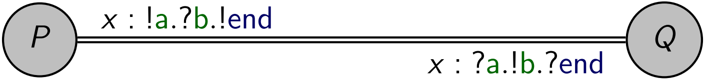
outline

1. Introduction
2. Subtyping
3. Fair termination
4. Fair subtyping
5. Conclusion

general ideas

Definition

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



Session types with branching points

$$?\mathsf{a}.S + ?\mathsf{b}.T \qquad\qquad !\mathsf{a}.S \oplus !\mathsf{b}.T$$

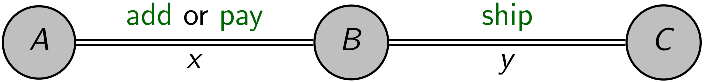
goals

- enable the **compositional static analysis** of distributed programs
- ensure that exchanged messages have the **expected type**, interactions occur in the **expected order** and processes **don't get stuck**
- **ensure that interactions terminate, eventually**

goals

- enable the **compositional static analysis** of distributed programs
- ensure that exchanged messages have the **expected type**, interactions occur in the the **expected order** and processes **don't get stuck**
- **ensure that interactions terminate, eventually**

the shopper, the store and the shipper



$$
\begin{aligned}
A(x) &\triangleq \ldots\text{shopper adds items to cart and pays}\ldots \\
B(x,y) &\triangleq x?\{\mathsf{add} : B\langle x,y\rangle, \mathsf{pay} : \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y\} \\
C(y) &\triangleq y?\mathsf{ship}.\mathsf{wait}\ y.\mathsf{done}
\end{aligned}
$$

$$(x)(A\langle x\rangle \mid (y)(B\langle x,y\rangle \mid C\langle y\rangle))$$

session type checking

- type checking ≈ matching the structure of processes and types

$$
\begin{aligned}
B(x : T, y : S) &\triangleq x?\{\mathsf{add} : B\langle x, y\rangle, \mathsf{pay} : \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y\} \\
T &= ?\mathsf{add}.T + ?\mathsf{pay}.?\mathsf{end} \\
S &= !\mathsf{ship}.!\mathsf{end}
\end{aligned}
$$

$$
\frac{}{x : T, y : S \vdash x?\{\mathsf{add} : B\langle x, y\rangle, \mathsf{pay} : \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y\}}
$$

session type checking

- type checking ≈ matching the structure of processes and types

$$B(x : T, y : S) \triangleq x?\{\mathsf{add} : B\langle x, y\rangle, \mathsf{pay} : \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y\}$$

$$T = ?\mathsf{add}.T + ?\mathsf{pay}.?\mathsf{end}$$

$$S = !\mathsf{ship}.!\mathsf{end}$$

$$\dfrac{\dfrac{}{x : T, y : S \vdash B\langle x, y\rangle}}{x : T, y : S \vdash x?\{\mathsf{add} : B\langle x, y\rangle, \mathsf{pay} : \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y\}}$$

session type checking

- type checking $\approx$ matching the structure of processes and types

$$
\begin{aligned}
B(x : T, y : S) &\triangleq x?\{\mathsf{add} : B\langle x, y\rangle, \mathsf{pay} : \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y\} \\
T &= ?\mathsf{add}.T + ?\mathsf{pay}.?\mathsf{end} \\
S &= !\mathsf{ship}.!\mathsf{end}
\end{aligned}
$$

$$
\dfrac{\dfrac{}{x : T, y : S \vdash B\langle x, y\rangle} \qquad \dfrac{}{x : ?\mathsf{end}, y : S \vdash \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y}}{x : T, y : S \vdash x?\{\mathsf{add} : B\langle x, y\rangle, \mathsf{pay} : \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y\}}
$$

session type checking

- type checking ≈ matching the structure of processes and types

$$
\begin{aligned}
B(x : T, y : S) &\triangleq x?\{\mathsf{add} : B\langle x, y\rangle, \mathsf{pay} : \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y\} \\
T &= ?\mathsf{add}.T + ?\mathsf{pay}.?\mathsf{end} \\
S &= !\mathsf{ship}.!\mathsf{end}
\end{aligned}
$$

$$
\dfrac{\dfrac{}{x : T, y : S \vdash B\langle x, y\rangle} \qquad \dfrac{\dfrac{}{y : S \vdash y!\mathsf{ship}.\mathsf{close}\ y}}{x : ?\mathsf{end}, y : S \vdash \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y}}{x : T, y : S \vdash x?\{\mathsf{add} : B\langle x, y\rangle, \mathsf{pay} : \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y\}}
$$

session type checking

- type checking ≈ matching the structure of processes and types

$$
\begin{aligned}
B(x : T, y : S) &\triangleq x?\{\mathsf{add} : B\langle x, y\rangle, \mathsf{pay} : \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y\} \\
T &= ?\mathsf{add}.T + ?\mathsf{pay}.?\mathsf{end} \\
S &= !\mathsf{ship}.!\mathsf{end}
\end{aligned}
$$

$$
\dfrac{\dfrac{}{x : T, y : S \vdash B\langle x, y\rangle} \qquad \dfrac{\dfrac{\dfrac{}{y : !\mathsf{end} \vdash \mathsf{close}\ y}}{y : S \vdash y!\mathsf{ship}.\mathsf{close}\ y}}{x : ?\mathsf{end}, y : S \vdash \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y}}{x : T, y : S \vdash x?\{\mathsf{add} : B\langle x, y\rangle, \mathsf{pay} : \mathsf{wait}\ x.y!\mathsf{ship}.\mathsf{close}\ y\}}
$$

parallel composition and duality

$$\dfrac{\dfrac{}{x : T, y : S \vdash B\langle x, y\rangle} \qquad \dfrac{}{y : S^{\perp} \vdash C\langle y\rangle}}{x : T \vdash (y)(B\langle x, y\rangle \mid C\langle y\rangle)}$$

- store and shipper use y according to **dual** session types

$$S = !\mathsf{ship}.!\mathsf{end} \qquad\qquad S^{\perp} = ?\mathsf{ship}.?\mathsf{end}$$

outline

one store, many shoppers

The store complies with one "canonical" protocol

$$T = ?\mathsf{add}.T + ?\mathsf{pay}.?\mathsf{end}$$

Shoppers may comply with **several** feasible protocols

$T^\perp = R = !\mathsf{add}.R \oplus !\mathsf{pay}.!\mathsf{end}$	**any number** of items
$R_1 = !\mathsf{add}.R$	**at least one** item
$R_{\mathsf{odd}} = !\mathsf{add}.(!\mathsf{add}.R_{\mathsf{odd}} \oplus !\mathsf{pay}.!\mathsf{end})$	**odd number** of items
$\ldots$	many possibilities

subtyping for session types [Gay and Hole, 2005]

$$S \leqslant T$$

Left-to-right substitution of endpoints [Liskov and Wing, 1994]

- an endpoint of type S can be safely used where an endpoint of type T is expected

$$?a \leqslant ?a + ?b \qquad !a \oplus !b \leqslant !a$$

Right-to-left substitution of processes [Gay, 2016]

- a process complying with protocol T can be safely used where a process complying with protocol S is expected

expected versus actual shopper

$$R_{\mathsf{odd}} = !\mathsf{add}.(!\mathsf{add}.R_{\mathsf{odd}} \oplus !\mathsf{pay}.!\mathsf{end}) \qquad \text{actual behavior}$$

$$T^{\perp} = R = !\mathsf{add}.R \oplus !\mathsf{pay}.!\mathsf{end} \qquad \text{expected behavior}$$

$$\dfrac{\dfrac{\overline{x : R_{\mathsf{odd}} \vdash A\langle x \rangle}}{x : T^{\perp} \vdash A\langle x \rangle}\; T^{\perp} \leqslant R_{\mathsf{odd}} \qquad \dfrac{\vdots}{x : T \vdash (y)(B\langle x, y \rangle \mid C\langle y \rangle)}}{\emptyset \vdash (x)(A\langle x \rangle \mid (y)(B\langle x, y \rangle \mid C\langle y \rangle))}$$

from safety to liveness

Theorem (nothing bad ever happens)

In a well-typed program

- *exchanged message have the expected type* *(comm. safety)*
- *interactions occur in the expected order* *(protocol fidelity)*
- *programs don't get stuck* *(deadlock freedom)*

Desideratum (something good eventually happens)

Also, in a well-typed program

- all sessions eventually terminate

outline

fair termination

Definition (fair termination)

We say that P is **fairly terminating** if each finite execution of P may be extended to done. That is

$$P \Longrightarrow Q \qquad \text{implies} \qquad Q \Longrightarrow \mathsf{done}$$

Remark

In a fairly terminating program

- every message sent may be received
- every process waiting for a message may receive one
- every session may terminate

$$P \Longrightarrow (x)(x!\mathsf{a}.P' \mid Q)$$

fair termination

Definition (fair termination)

We say that P is **fairly terminating** if each finite execution of P may be extended to done. That is

$$P \Longrightarrow Q \qquad \text{implies} \qquad Q \Longrightarrow \mathsf{done}$$

Remark

In a fairly terminating program

- every message sent may be received
- every process waiting for a message may receive one
- every session may terminate

$$P \Longrightarrow (x)(x!\mathsf{a}.P' \mid Q) \Longrightarrow \mathsf{done}$$

why "fair termination"?

Definition (strong fairness [Francez, 1986])

A program execution is **strongly fair** if every reduction that is **infinitely often enabled** is **infinitely often performed**

Consider the shopper protocol $R = !\mathsf{add}.R \oplus !\mathsf{pay}.!\mathsf{end}$:

- an execution in which the shopper sends add forever is **unfair**
- in every maximal, fair execution the shopper eventually sends pay

Theorem

A (finite-state) program is fairly terminating if and only if every maximal, fair execution is finite and finishes with done

- In principle, a fairly-terminating program may execute forever
- In practice, *i.e.* if its realistic executions are fair, it doesn't

why “fair termination”?

Definition (strong fairness [Francez, 1986])

A program execution is **strongly fair** if every reduction that is **infinitely often enabled** is **infinitely often performed**

Consider the shopper protocol $R = !\mathsf{add}.R \oplus !\mathsf{pay}.!\mathsf{end}$:

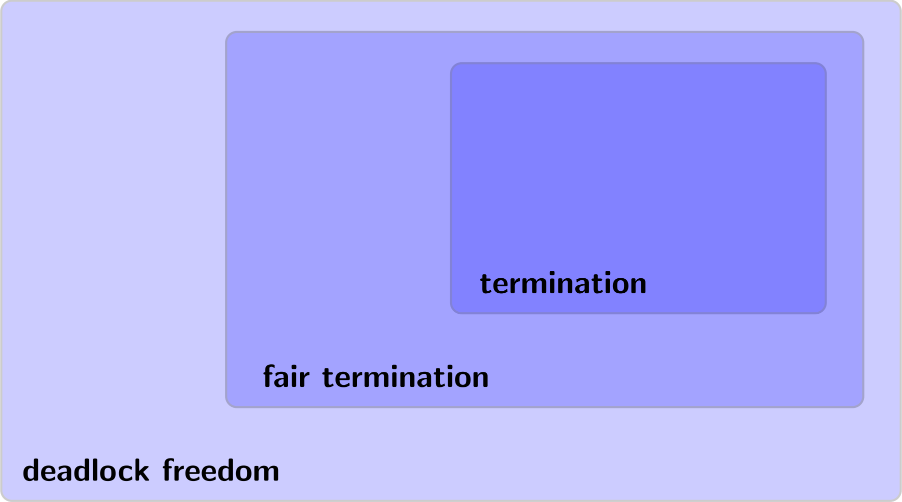
- an execution in which the shopper sends add forever is **unfair**
- in every maximal, fair execution the shopper eventually sends pay

Theorem

A (finite-state) program is fairly terminating if and only if every maximal, fair execution is finite and finishes with done

- In principle, a fairly-terminating program may execute forever
- In practice, *i.e.* if its realistic executions are fair, it doesn't

termination < fair termination < deadlock freedom



example: the compulsive shopper

$$A(x) \stackrel{\triangle}{=} x!\mathsf{add}.A\langle x\rangle \qquad R_\infty = !\mathsf{add}.R_\infty$$

$$\dfrac{\dfrac{\dfrac{}{x : R_\infty \vdash A\langle x\rangle}}{x : T^\perp \vdash A\langle x\rangle}\; T^\perp \leqslant R_\infty \qquad \dfrac{\vdots}{x : T \vdash (y)(B\langle x, y\rangle \mid C\langle y\rangle)}}{\emptyset \vdash (x)(A\langle x\rangle \mid (y)(B\langle x, y\rangle \mid C\langle y\rangle))}$$

1. this program is deadlock-free but not fairly terminating
2. the strong fairness assumption alone doesn't turn an ordinary session type system into one that ensures fair termination

example: the compulsive shopper

$$A(x) \triangleq x!\mathsf{add}.A\langle x\rangle \qquad R_\infty = !\mathsf{add}.R_\infty$$

$$\dfrac{\dfrac{\overline{x : R_\infty \vdash A\langle x\rangle}}{x : T^\perp \vdash A\langle x\rangle}\ T^\perp \leqslant R_\infty \qquad \dfrac{\vdots}{x : T \vdash (y)(B\langle x, y\rangle \mid C\langle y\rangle)}}{\emptyset \vdash (x)(A\langle x\rangle \mid (y)(B\langle x, y\rangle \mid C\langle y\rangle))}$$

1. this program is deadlock-free but not fairly terminating
2. the strong fairness assumption alone doesn't turn an ordinary session type system into one that ensures fair termination

$\leqslant$ is designed to preserve safety, not liveness

outline

fair subtyping [Padovani, 2013, 2016, Ciccone and Padovani, 2021a]

defined by a **generalized inference system** [Ancona et al., 2017, Dagnino, 2019]

$$\frac{}{p\,\mathsf{end} \leqslant p\,\mathsf{end}} \qquad \frac{S_k \leqslant T_k}{!\{\mathsf{a}_i : S_i\}_{i\in I} \leqslant !\{\mathsf{a}_j : T_j\}_{j\in J}} \text{ corule}$$

$$\frac{S_i \leqslant T_i \ ^{(i\in I)}}{?\{\mathsf{a}_i : S_i\}_{i\in I} \leqslant ?\{\mathsf{a}_i : T_i\}_{i\in I\cup J}} \qquad \frac{S_i \leqslant T_i \ ^{(i\in I)}}{!\{\mathsf{a}_i : S_i\}_{i\in I\cup J} \leqslant !\{\mathsf{a}_i : T_i\}_{i\in I}}$$

We say that S is a **fair subtype** of T, notation $S \leqslant T$, if

- there is an **arbitrary** derivation of $S \leqslant T$ **using just rules**
- there is a **finite** derivation $S \leqslant T$ **using rules and corules**

example of fair subtyping

$$R = !\mathsf{add}.R \oplus !\mathsf{pay}.!\mathsf{end} \qquad R_1 = !\mathsf{add}.R$$

$$\dfrac{\dfrac{\dfrac{\vdots}{R \leqslant R} \qquad \dfrac{}{!\mathsf{end} \leqslant !\mathsf{end}}}{R \leqslant R}}{R \leqslant R_1} \qquad\qquad \dfrac{\dfrac{\dfrac{}{!\mathsf{end} \leqslant !\mathsf{end}}}{R \leqslant R}}{R \leqslant R_1}$$

$$R \leqslant R_1$$

Note

- there is no finite derivation for $R \leqslant R_1$ without the corule

example of **unfair** subtyping

$R = {!}\mathsf{add}.R \oplus {!}\mathsf{pay}.{!}\mathsf{end}$ $\qquad$ $R_\infty = {!}\mathsf{add}.R_\infty$

$$\dfrac{\dfrac{\vdots}{R \leqslant R_\infty}}{R \leqslant R_\infty} \qquad\qquad \dfrac{\dfrac{\vdots}{R \leqslant R_\infty}}{R \leqslant R_\infty}$$

$$R \not\leqslant R_\infty$$

Note

- there is no finite derivation for $R \leqslant R_1$, **even with the corule**

properties of fair subtyping

Fact

Fair subtyping is a **liveness-preserving** subtyping relation closely related to *fair testing* [Natarajan and Cleaveland, 1995] and *should testing* [Rensink and Vogler, 2007]

- see Bugliesi et al. [2009], Bravetti and Zavattaro [2009], Padovani [2013, 2016], Bravetti et al. [2021] for details
- see Ciccone and Padovani [2021a] for an Agda formalization

compulsive shopping is not allowed...

$$\dfrac{\dfrac{\dfrac{}{x : R_\infty \vdash A\langle x\rangle}}{\color{red}{x : T^\perp \vdash A\langle x\rangle}}\ \cancel{T^\perp \leqslant R_\infty} \qquad \dfrac{\vdots}{x : T \vdash (y)(B\langle x, y\rangle \mid C\langle y\rangle)}}{\emptyset \vdash (x)(A\langle x\rangle \mid (y)(B\langle x, y\rangle \mid C\langle y\rangle))}$$

compulsive shopping is not allowed... or is it?

A different typing derivation for the compulsive shopper

$$A(x) \triangleq x!\mathsf{add}.A\langle x\rangle$$
$$R = !\mathsf{add}.R \oplus !\mathsf{pay}.!\mathsf{end}$$
$$R_1 = !\mathsf{add}.R$$

$$\dfrac{\dfrac{\overline{x : R \vdash A\langle x\rangle}}{x : R_1 \vdash x!\mathsf{add}.A\langle x\rangle}}{x : R \vdash x!\mathsf{add}.A\langle x\rangle}\ R \leqslant R_1$$

- $R \leqslant R_1$ is used in the typing derivation of a recursive process
- "infinitely many" usages of fair subtyping ($R \leqslant R_1$) may have the same overall effect of unfair subtyping ($R \leqslant R_\infty$)
- well-typed processes should only be allowed to perform a **bounded number of casts**

cast boundedness

$$\Gamma \vdash^n P$$

- P is well-typed in Γ and has **rank** n
- n is an upper bound to the number of casts performed by P

$$\dfrac{\dfrac{\overline{x : R \vdash^{n} A\langle x\rangle}}{x : R_1 \vdash^{n} x!\mathsf{add}.A\langle x\rangle}}{x : R \vdash^{n+1} x!\mathsf{add}.A\langle x\rangle}\ R \leqslant R_1$$

- we cannot assign a finite rank to this typing derivation

fair termination, at last

Theorem

If $\emptyset \vdash^n P$, then P is fairly terminating

Proof idea.

Show that typing is preserved by reductions (subject reduction):

- if $\Gamma \vdash^n P$ and $P \longrightarrow Q$, then $\Gamma \vdash^n Q$

Define a **measure** for well-typed processes that includes n as well as the effort required to terminate all open sessions:

- $\Gamma \vdash^{\mu} P$

Show that for every non-terminated, well-typed program **there exists** a reduct with a **strictly smaller** measure:

- if $\emptyset \vdash^{\mu} P$, either $P = \mathsf{done}$ or $P \longrightarrow Q$ and $\emptyset \vdash^{\nu} Q$ where $\nu < \mu$

Note: the measure **may increase** if new sessions are opened. □

outline

summary

A compositional static analysis ensuring fair termination

- well-typed programs terminate, if the fairness assumption is true
- infinite executions are possible, but only in principle

A nice application of generalized inference systems

- definition of fair subtyping
- typing corules (not discussed in this talk, see below)

Want more?

- many simplifications in this talk
- see Ciccone and Padovani [2021b] for details
  (higher-order sessions, proofs, type checking algorithm, ...)

ongoing and future work

Type checker implementation

- ready, soon available on my home page

Other communication models

- multiparty sessions (straightforward)
- actors, many-to-one communications (challenging)

ongoing and future work

Type checker implementation

- ready, soon available on my home page

Other communication models

- multiparty sessions (straightforward)
- actors, many-to-one communications (challenging)

thank you!

references

Davide Ancona, Francesco Dagnino, and Elena Zucca. Generalizing inference systems by coaxioms. In Hongseok Yang, editor, *Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, volume 10201 of *Lecture Notes in Computer Science*, pages 29–55. Springer, 2017.

Mario Bravetti and Gianluigi Zavattaro. A theory of contracts for strong service compliance. *Math. Struct. Comput. Sci.*, 19(3):601–638, 2009.

references (cont.)

Mario Bravetti, Julien Lange, and Gianluigi Zavattaro. Fair refinement for asynchronous session types. In Stefan Kiefer and Christine Tasson, editors, *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*, volume 12650 of *Lecture Notes in Computer Science*, pages 144–163. Springer, 2021.

Michele Bugliesi, Damiano Macedonio, Luca Pino, and Sabina Rossi. Compliance preorders for web services. In Cosimo Laneve and Jianwen Su, editors, *Web Services and Formal Methods, 6th International Workshop, WS-FM 2009, Bologna, Italy, September 4-5, 2009, Revised Selected Papers*, volume 6194 of *Lecture Notes in Computer Science*, pages 76–91. Springer, 2009.

references (cont.)

Luca Ciccone and Luca Padovani. Inference Systems with Corules for Fair Subtyping and Liveness Properties of Binary Session Types. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP'21)*, volume 198 of *LIPIcs*, pages 125:1–125:16, Dagstuhl, Germany, 2021a. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Luca Ciccone and Luca Padovani. Fair Termination of Binary Sessions. Technical report, 2021b. URL `http://www.di.unito.it/~padovani/assets/downloads/fair-termination.pdf`. unpublished.

Francesco Dagnino. Coaxioms: flexible coinductive definitions by inference systems. *Log. Methods Comput. Sci.*, 15(1), 2019.

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

references (cont.)

Simon J. Gay. Subtyping supports safe session substitution. In Sam Lindley, Conor McBride, Philip W. Trinder, and Donald Sannella, editors, *A List of Successes That Can Change the World - Essays Dedicated to Philip Wadler on the Occasion of His 60th Birthday*, volume 9600 of *Lecture Notes in Computer Science*, pages 95–108. Springer, 2016.

Simon J. Gay and Malcolm Hole. Subtyping for session types in the pi calculus. *Acta Informatica*, 42(2-3):191–225, 2005.

Barbara Liskov and Jeannette M. Wing. A behavioral notion of subtyping. *ACM Trans. Program. Lang. Syst.*, 16(6):1811–1841, 1994.

V. Natarajan and Rance Cleaveland. Divergence and fair testing. In Zoltán Fülöp and Ferenc Gécseg, editors, *Automata, Languages and Programming, 22nd International Colloquium, ICALP95, Szeged, Hungary, July 10-14, 1995, Proceedings*, volume 944 of *Lecture Notes in Computer Science*, pages 648–659. Springer, 1995.

references (cont.)

Luca Padovani. Fair subtyping for open session types. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 373–384. Springer, 2013.

Luca Padovani. Fair subtyping for multi-party session types. *Math. Struct. Comput. Sci.*, 26(3):424–464, 2016.

Arend Rensink and Walter Vogler. Fair testing. *Inf. Comput.*, 205(2): 125–198, 2007.