fair termination of binary sessions

Luca Padovani, Università di Torino

joint work with Luca Ciccone
in proceedings of 49th annual Symposium on **Principles of Programming Languages** (POPL 2022)

outline

1. a quick introduction to binary sessions
2. on subtyping and why it matters
3. fair termination
4. on fair subtyping and how to use it
5. concluding remarks

outline

1 a quick introduction to binary sessions

2 on subtyping and why it matters

3 fair termination

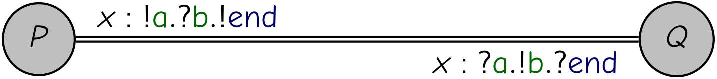
4 on fair subtyping and how to use it

5 concluding remarks

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 may have branching points

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

session types may be "recursive" (*i.e.* infinite regular trees)

$$S = ?\mathsf{a}.S + ?b.T$$

goal

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

during the execution of a well-typed distributed program...

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

...and in this work

- **all sessions terminate, sooner or later** (fair termination)

goal

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

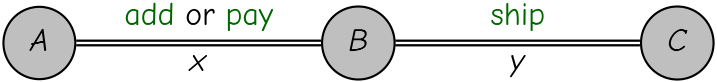
during the execution of a well-typed distributed program...

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

...and in this work

- **all sessions terminate, sooner or later** (fair termination)

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, in one slide

structure of types $\Longleftrightarrow$ structure of process

$$\Gamma \quad \vdash \quad P$$

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

$$T = ?\mathsf{add}.T + ?\mathsf{pay}.?\mathsf{end} \qquad S = !\mathsf{ship}.!\mathsf{end}$$

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

session type checking, in one slide

structure of types $\Longleftrightarrow$ structure of process

$$\Gamma \quad \vdash \quad P$$

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

$$T = ?\mathsf{add}.T + ?\mathsf{pay}.?\mathsf{end} \qquad S = !\mathsf{ship}.!\mathsf{end}$$

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

session type checking, in one slide

structure of types $\Longleftrightarrow$ structure of process

$$\Gamma \quad \vdash \quad P$$

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

$$T = ?\mathsf{add}.T + ?\mathsf{pay}.?\mathsf{end} \qquad S = !\mathsf{ship}.!\mathsf{end}$$

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

session type checking, in one slide

structure of types $\Longleftrightarrow$ structure of process

$$\Gamma \quad \vdash \quad P$$

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

$$T = ?\mathsf{add}.T + ?\mathsf{pay}.?\mathsf{end} \qquad S = !\mathsf{ship}.!\mathsf{end}$$

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

session type checking, in one slide

structure of types $\iff$ structure of process

$$\Gamma \quad \vdash \quad P$$

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

$$T = ?\mathsf{add}.T + ?\mathsf{pay}.?\mathsf{end} \qquad S = !\mathsf{ship}.!\mathsf{end}$$

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

on parallel composition and duality

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

Notes

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

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

- checking that a parallel composition is well typed boils down to checking a simple property of types

(compositional analysis!)

outline

1 a quick introduction to binary sessions

2 on subtyping and why it matters

3 fair termination

4 on fair subtyping and how to use it

5 concluding remarks

a moltitude of shopper protocols

The store complies with **one** protocol

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

The shopper may comply with **many** different 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 more possibilities

Only R is the dual of T, but all should be "compatible" with T

subtyping for session types, two viewpoints

[Gay and Hole, 2005]

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

- when $S \leqslant T$ an endpoint of type T can be safely replaced by an endpoint of type S

$?a \leqslant ?a + ?b$	$!a \oplus !b \leqslant !a$
covariant inputs	contravariant outputs

left-to-right subst. of processes [De Nicola and Hennessy, 1984]

- when $S \leqslant T$ a process complying with protocol S can be safely replaced by a process complying with protocol S

expected versus actual shopper

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

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

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

soundness... and lack thereof

Theorem

In a well-typed program

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

Desideratum

Also, in a well-typed program

- all sessions eventually terminate (fair termination)

Facts

There exist well-typed processes in which sessions **don't terminate**, sent messages are **not delivered**, awaited messages are **not sent**...

soundness…and lack thereof

Theorem

In a well-typed program

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

Desideratum

Also, in a well-typed program

- all sessions eventually terminate (fair termination)

Facts

There exist well-typed processes in which sessions **don't terminate**, sent messages are **not delivered**, awaited messages are **not sent**…

outline

1 a quick introduction to binary sessions

2 on subtyping and why it matters

3 fair termination

4 on fair subtyping and how to use it

5 concluding remarks

fair termination

Definition (fair termination)

We say that P is **fairly terminating** if $P \Longrightarrow Q$ implies $Q \Longrightarrow \mathsf{done}$

Intuition

If termination is **always possible** then (we assume) it is **inevitable**

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

- in theory, the shopper may send add forever
- in practice, the shopper eventually sends pay and terminates

In the literature

- instance of **relative fairness** [Queille and Sifakis, 1983]
- instance of **∞-fairness** [Best, 1984]

properties of fairly terminating programs

In a fairly terminating program

- every sent message is eventually delivered (no junk)
- every expected message eventually arrives (no starvation)
- every session eventually terminates (fair session termination)

$$\overbrace{P \longrightarrow \cdots \longrightarrow Q}^{\text{partial execution}}$$

properties of fairly terminating programs

In a fairly terminating program

- every sent message is eventually delivered (no junk)
- every expected message eventually arrives (no starvation)
- every session eventually terminates (fair session termination)

$$\underbrace{\overbrace{P \longrightarrow \cdots \longrightarrow Q}^{\text{partial execution}} \longrightarrow \cdots \longrightarrow \mathsf{done}}_{\text{maximal fair execution}}$$

feasibility [Apt et al., 1987] aka **machine closure** [Lamport, 2000]

- every partial execution can be extended to a maximal fair one

problem: the compulsive shopper

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

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

Notes

- this program is deadlock-free but not fairly terminating
- the sessions x and y don't (and cannot) terminate
- the shipper awaits for a message that is never sent

problem: the compulsive shopper

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

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

Notes

- this program is deadlock-free but not fairly terminating
- the sessions x and y don't (and cannot) terminate
- the shipper awaits for a message that is never sent

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

outline

1 a quick introduction to binary sessions

2 on subtyping and why it matters

3 fair termination

4 on fair subtyping and how to use it

5 concluding remarks

fair subtyping

[Padovani, 2013, 2016, Ciccone and Padovani, 2021]

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

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

We say that S is a **fair subtype** of T if

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

Instance of **generalized inference system** [Ancona et al., 2017]

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 of $R \leqslant R_1$ without the corule

example of **unfair** subtyping

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

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

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

Note

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

compulsive shopping is not allowed...

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

compulsive shopping is not allowed…or is it?

A different typing derivation for the compulsive shopper

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

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

Poset of session types ordered by fair subtyping is not ω-complete

- "infinitely many" usages of fair subtyping ($R \leqslant R_1$) may have the same overall effect of unfair subtyping ($R \leqslant R_\infty$)

$$
R \leqslant !\mathsf{add}.R \leqslant !\mathsf{add}.!\mathsf{add}.R \leqslant \cdots \not\leqslant R_\infty
$$

- well-typed processes should only be allowed to perform a **bounded number of casts**

cast boundedness

Enrich typing judgments with a rank

$$\Gamma \vdash^{n} P$$

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

The compulsive shopper has no finite rank

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

fair termination, at last

Theorem

If P is well typed 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 $\square$

outline

1 a quick introduction to binary sessions

2 on subtyping and why it matters

3 fair termination

4 on fair subtyping and how to use it

5 concluding remarks

summary

A compositional static analysis ensuring fair termination

- well-typed sessions (fairly) terminate

Want more?

- many simplifications in this talk
- see Ciccone and Padovani [2022] for details
  (higher-order sessions, proofs, type checking algorithm, ...)
- presented at POPL next week

further and future work

FairCheck

- Haskell implementation of the type checker
- available on GitHub (link from my home page)

Application to other communication models

- multiparty sessions (easy)
- actors, concurrent objects, smart contracts (harder)

further and future work

FairCheck

- Haskell implementation of the type checker
- available on GitHub (link from my home page)

Application to other communication models

- multiparty sessions (easy)
- actors, concurrent objects, smart contracts (harder)

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.

Krzysztof R. Apt, Nissim Francez, and Shmuel Katz. Appraising fairness in languages for distributed programming. In *Conference Record of the Fourteenth Annual ACM Symposium on Principles of Programming Languages, Munich, Germany, January 21-23, 1987*, pages 189–198. ACM Press, 1987.

Eike Best. Fairness and conspiracies. *Inf. Process. Lett.*, 18(4):215–220, 1984. URL `https://doi.org/10.1016/0020-0190(84)90114-5`.

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, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Luca Ciccone and Luca Padovani. Fair Termination of Binary Sessions. *Proceedings of the ACM on Programming Languages*, 6, 2022.

Rocco De Nicola and Matthew Hennessy. Testing equivalences for processes. *Theor. Comput. Sci.*, 34:83–133, 1984.

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

Leslie Lamport. Fairness and hyperfairness. *Distributed Comput.*, 13(4): 239–245, 2000.

references (cont.)

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

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.

Jean-Pierre Queille and Joseph Sifakis. Fairness and related properties in transition systems - A temporal logic to deal with fairness. *Acta Informatica*, 19:195–220, 1983. URL `https://doi.org/10.1007/BF00265555`.