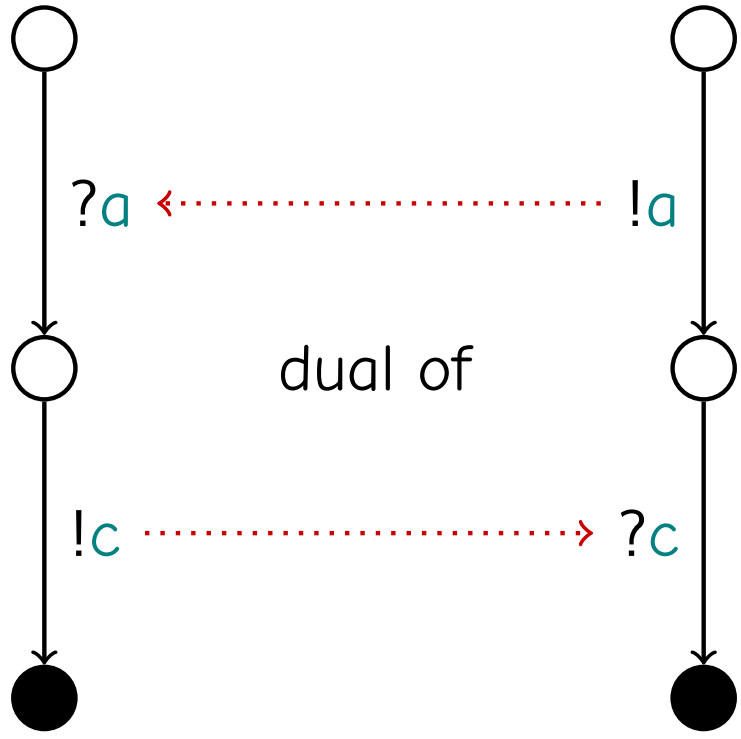
fair termination of asynchronous binary sessions

Luca Padovani Gianluigi Zavattaro

Department of Computer Science and Engineering – University of Bologna

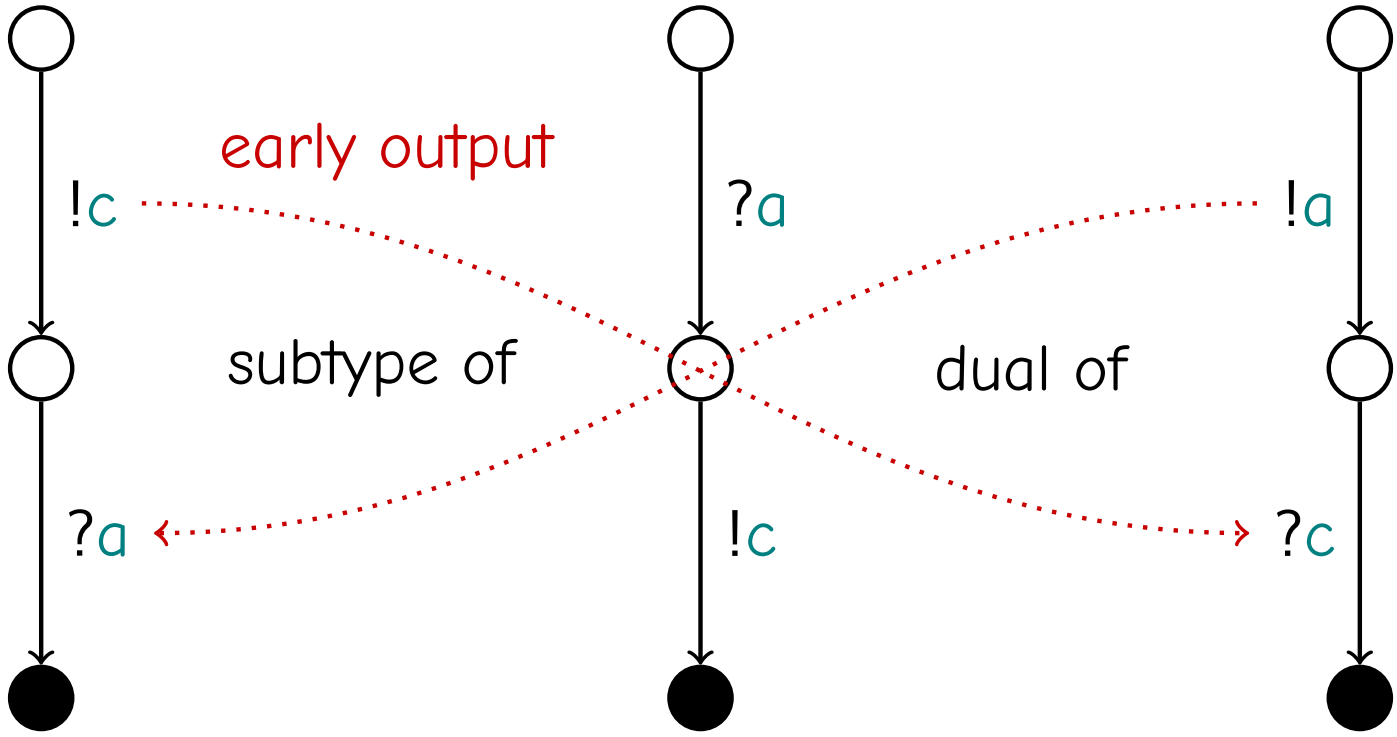
duality, asynchrony and subtyping



- communication safety
- progress
- half-duplex communication

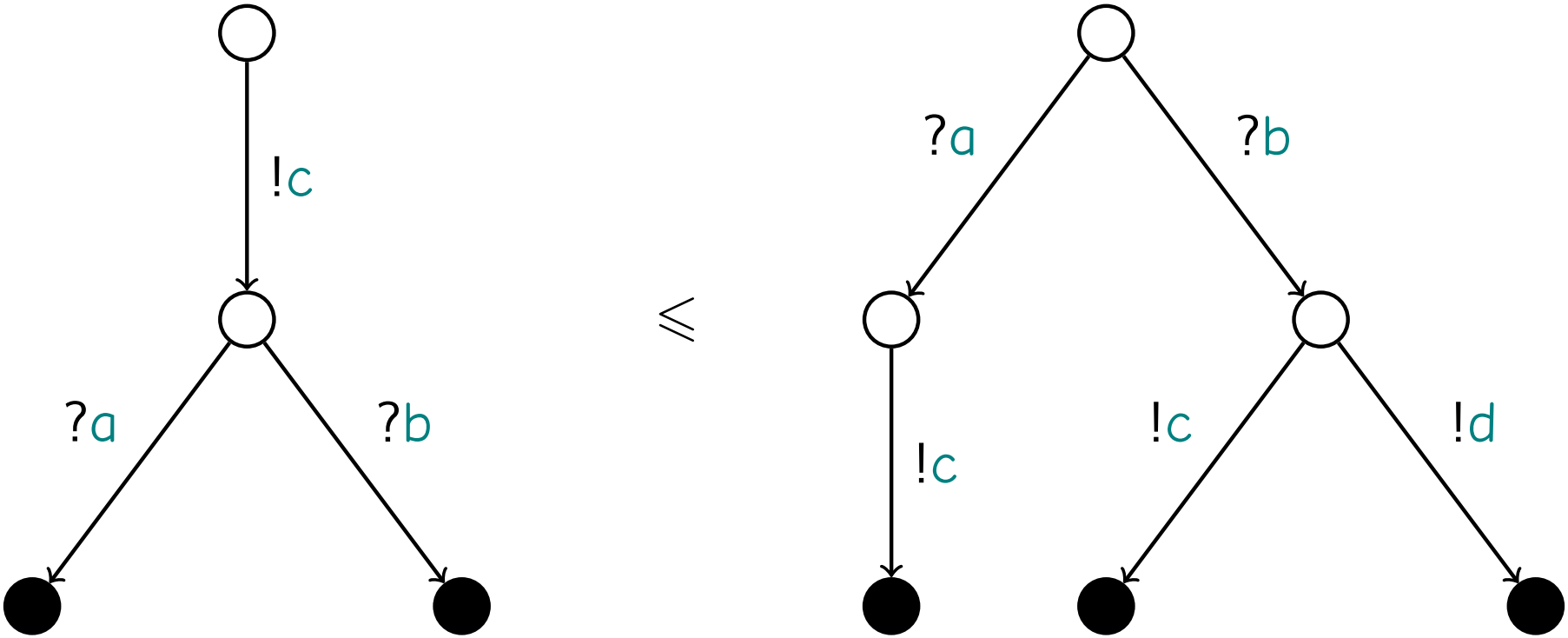
duality, asynchrony and subtyping

Mostrous et al. [2009], Chen et al. [2017], Ghilezan et al. [2023]



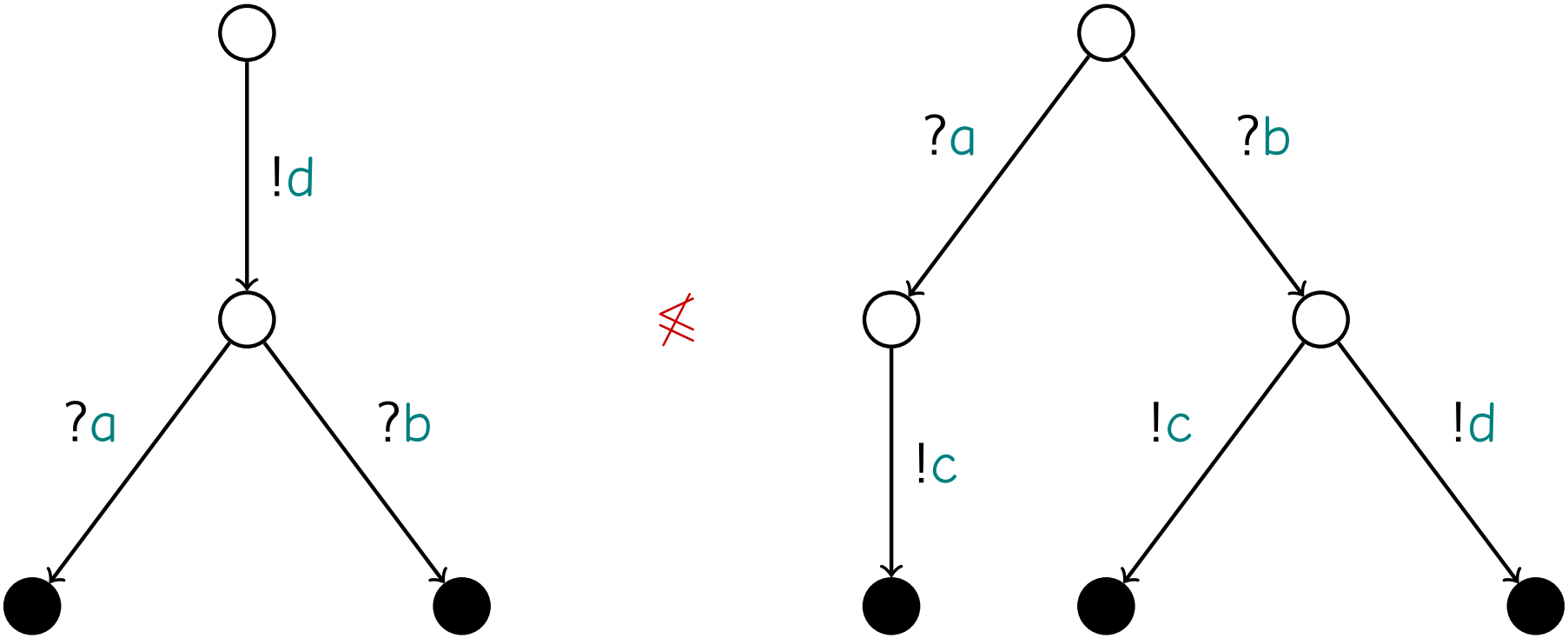
- communication safety
- progress
- full-duplex communication

output anticipation and causal dependencies



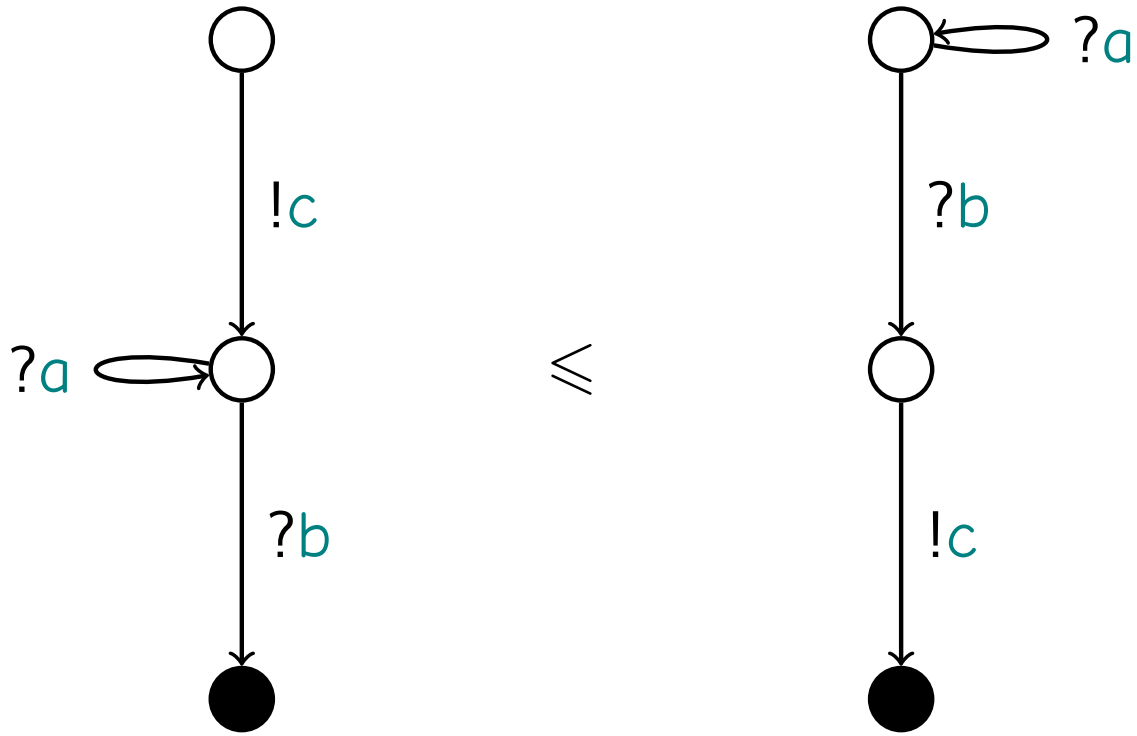
- c can be sent early (doesn't depend on the previous input)

output anticipation and causal dependencies



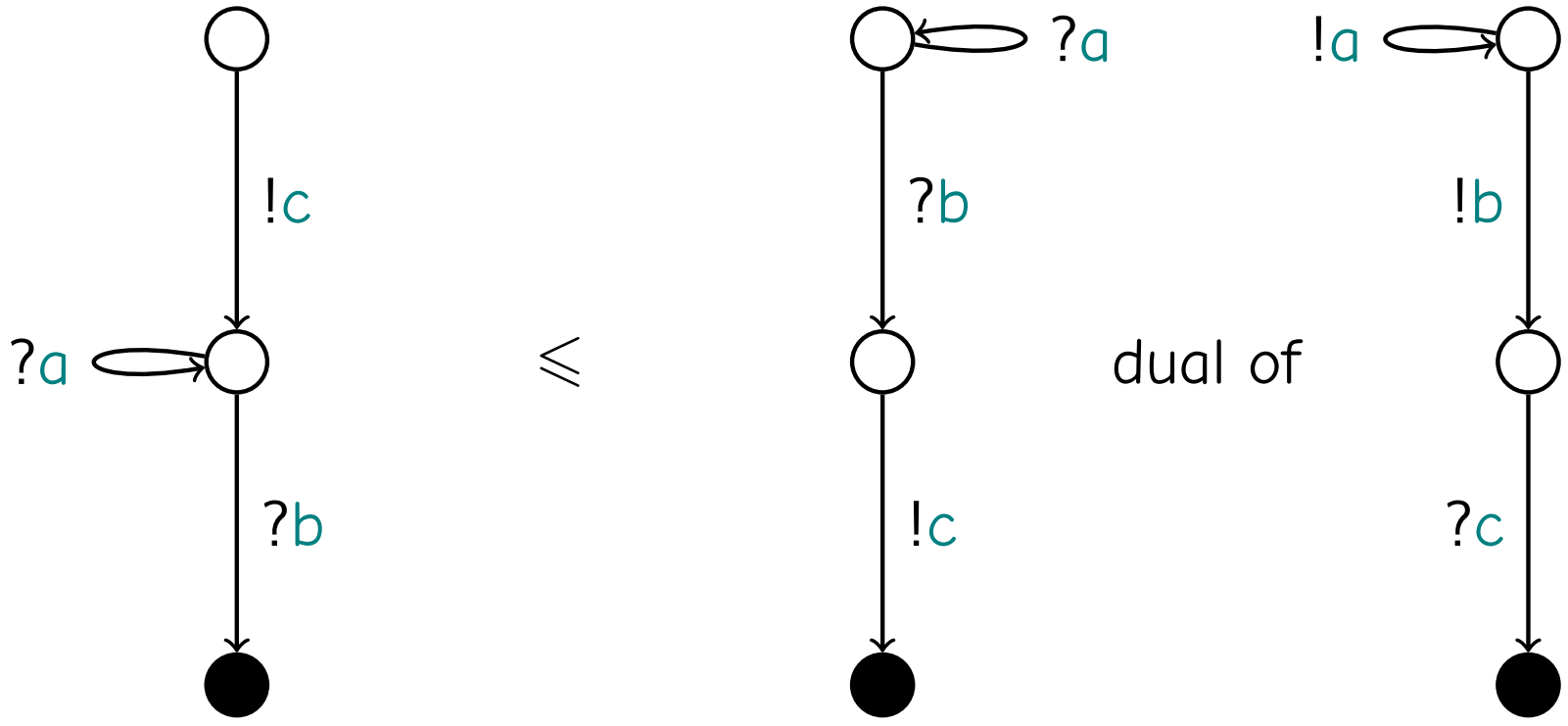
- c can be sent early (doesn't depend on the previous input)
- d cannot be sent early (depends on the previous input)

unbounded output anticipation



- aforementioned approaches allow **bounded** output anticipation
- here we consider **unbounded** output anticipation as well

unbounded output anticipation



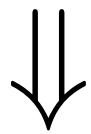
- aforementioned approaches allow **bounded** output anticipation
- here we consider **unbounded** output anticipation as well
- without a fairness assumption we can have orphan messages

contribution

fair asynchronous subtyping

+

type system ensuring weak process termination



unbounded output anticipation

no orphan messages

towards asynchronous subtyping

Our approach

- capture asynchrony in the **transition relation** of session types
- transitions say what a process can do, not what it does

$$\frac{k \in I}{\oplus\{a_i : S_i\}_{i \in I} \xrightarrow{!a_k} S_k} \qquad \frac{\forall i \in I : S_i \xrightarrow{!c} T_i}{\&\{a_i : S_i\}_{i \in I} \xrightarrow{!c} \&\{a_i : T_i\}_{i \in I}}$$

Example: deriving an **early output**

$$\frac{\dfrac{}{!c.S_1 \xrightarrow{!c} S_1} \qquad \dfrac{}{!c.S_2 \oplus !d.T \xrightarrow{!c} S_2}}{?a.!c.S_1 \,\&\, ?b.(!c.S_2 \oplus !d.T) \xrightarrow{!c} ?a.S_1 \,\&\, ?b.S_2}$$

a problem with recursive session types

$$S = ?\mathsf{a}.S \,\&\, ?\mathsf{b}.!\mathsf{c}.U \qquad T = ?\mathsf{a}.T \,\&\, ?\mathsf{b}.U$$

$$\dfrac{\dfrac{}{!\mathsf{c}.U \xrightarrow{!\mathsf{c}} U}}{S \xrightarrow{!\mathsf{c}} T}$$

a problem with recursive session types

$$S = ?\mathsf{a}.S \,\&\, ?\mathsf{b}.!\mathsf{c}.U \qquad T = ?\mathsf{a}.T \,\&\, ?\mathsf{b}.U$$

$$\frac{S \xrightarrow{!\mathsf{c}} T \qquad \overline{!\mathsf{c}.U \xrightarrow{!\mathsf{c}} U}}{S \xrightarrow{!\mathsf{c}} T}$$

a problem with recursive session types

$$S = ?\mathsf{a}.S \,\&\, ?\mathsf{b}.!\mathsf{c}.U \qquad T = ?\mathsf{a}.T \,\&\, ?\mathsf{b}.U$$

$$\dfrac{\dfrac{}{!\mathsf{c}.U \xrightarrow{!\mathsf{c}} U}}{S \xrightarrow{!\mathsf{c}} T} \qquad \dfrac{}{!\mathsf{c}.U \xrightarrow{!\mathsf{c}} U}}{S \xrightarrow{!\mathsf{c}} T}$$

a problem with recursive session types

$$S = ?\mathsf{a}.S \,\&\, ?\mathsf{b}.!\mathsf{c}.U \qquad T = ?\mathsf{a}.T \,\&\, ?\mathsf{b}.U$$

$$\dfrac{\dfrac{\dfrac{\vdots}{S \xrightarrow{!\mathsf{c}} T} \qquad \dfrac{}{!\mathsf{c}.U \xrightarrow{!\mathsf{c}} U}}{S \xrightarrow{!\mathsf{c}} T} \qquad \dfrac{}{!\mathsf{c}.U \xrightarrow{!\mathsf{c}} U}}{S \xrightarrow{!\mathsf{c}} T}$$

a problem with recursive session types

$$S = ?a.S \,\&\, ?b.!c.U \qquad T = ?a.T \,\&\, ?b.U$$

$$\dfrac{\dfrac{\vdots}{S \xrightarrow{!c} T} \qquad \dfrac{}{!c.U \xrightarrow{!c} U}}{S \xrightarrow{!c} T} \qquad \dfrac{}{!c.U \xrightarrow{!c} U}$$

$$\dfrac{S \xrightarrow{!c} T \qquad !c.U \xrightarrow{!c} U}{S \xrightarrow{!c} T}$$

we turn the problem into the solution

- define the LTS **coinductively** 😜
- there's a catch: make sure no phony transitions are derivable
- use a **generalized inference system** [Ancona et al., 2017]

fair asynchronous subtyping

Definition (simplified, first order)

Fair asynchronous subtyping is the largest $\leqslant$ s.t. $S \leqslant T$ implies

1. $S \xrightarrow{!a} S'$ implies $T \xrightarrow{!a} T'$ and $S' \leqslant T'$
2. $T \xrightarrow{?a} T'$ implies $S \xrightarrow{?a} S'$ and $S' \leqslant T'$

Properties

- *"same" as synchronous subtyping* *(input/output variance)*

fair asynchronous subtyping

Definition (simplified, first order)

Fair asynchronous subtyping is the largest $\leqslant$ s.t. $S \leqslant T$ implies

1. $S \xrightarrow{!a} S'$ implies $T \xrightarrow{!a} T'$ and $S' \leqslant T'$
2. $T \xrightarrow{?a} T'$ implies $S \xrightarrow{?a} S'$ and $S' \leqslant T'$

Properties

- *"same" as synchronous subtyping* *(input/output variance)*
- $S \leqslant T$ *implies* $T^{\perp} \leqslant S^{\perp}$ *(closure under duality)*

fair asynchronous subtyping

Definition (simplified, first order)

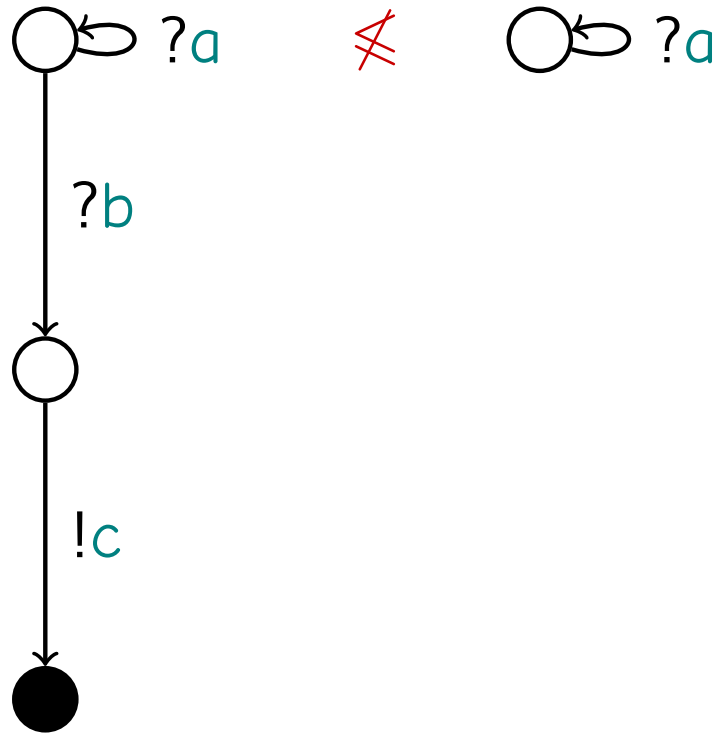
Fair asynchronous subtyping is the largest $\leqslant$ s.t. $S \leqslant T$ implies

1. $S \xrightarrow{!a} S'$ implies $T \xrightarrow{!a} T'$ and $S' \leqslant T'$
2. $T \xrightarrow{?a} T'$ implies $S \xrightarrow{?a} S'$ and $S' \leqslant T'$

Properties

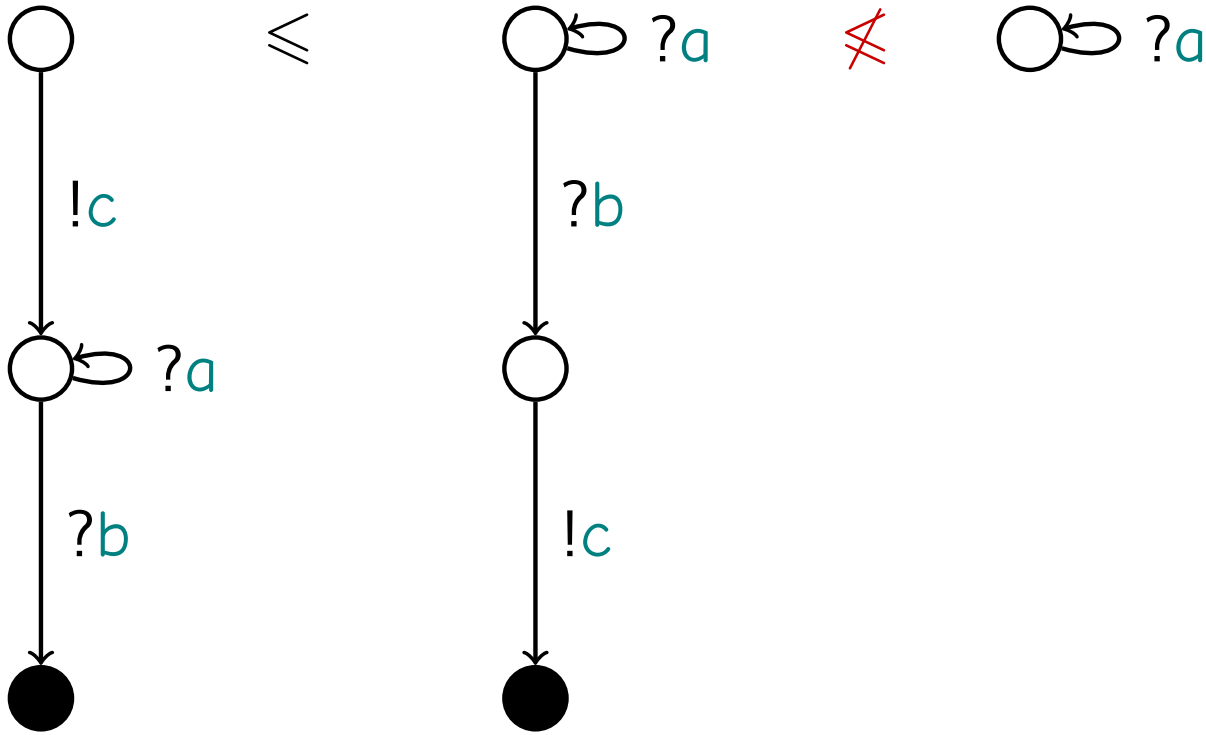
- *"same" as synchronous subtyping* *(input/output variance)*
- $S \leqslant T$ *implies* $T^{\perp} \leqslant S^{\perp}$ *(closure under duality)*
- *undecidable*

some corner cases



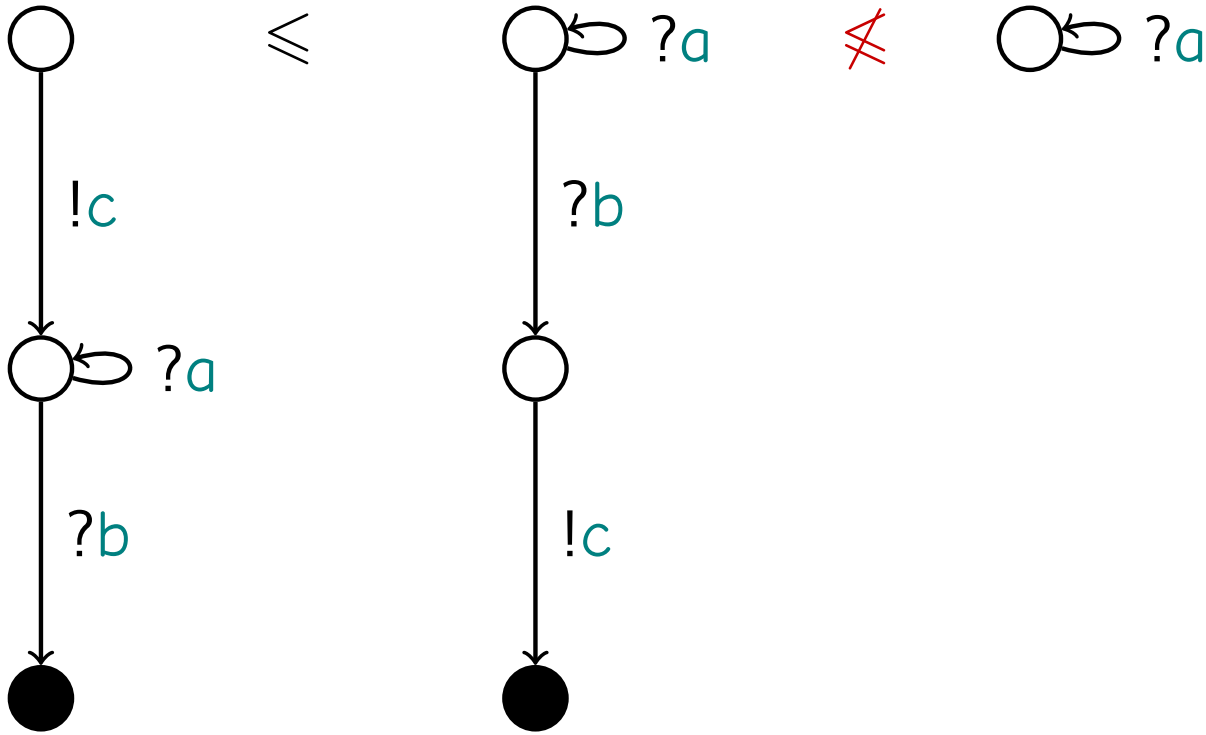
- holds for synchronous subtyping [Gay and Hole, 2005]

some corner cases



- holds for synchronous subtyping [Gay and Hole, 2005]

some corner cases



- holds for synchronous subtyping [Gay and Hole, 2005]
- no big deal, ↺ ?a is not inhabited in our type system

ensuring weak program termination

Starting point: type system based on **linear logic**

- (in)finitary logic with fixed points *e.g.* [Doumane, 2017]
- infinitary logic with measures [Dagnino and Padovani, 2024]

heavier on annotations but simpler proofs

(weak) cut elimination $\Rightarrow$ (weak) program termination

...but what about asynchrony?

asynchrony in a logical setting

Queueless asynchronous semantics: add "deep" cut reductions

$$(x)(x \lhd c.P \mid \quad x \rhd c.Q\ ) \to (x)(P \mid \quad Q\ )$$

asynchrony in a logical setting

Queueless asynchronous semantics: add "deep" cut reductions

$$(x)(x \lhd c.P \mid \mathcal{B}_x[x \rhd c.Q]) \to (x)(P \mid \mathcal{B}_x[Q])$$

sequence of output prefixes on x

asynchrony in a logical setting

Queueless asynchronous semantics: add "deep" cut reductions

$$(x)(x \lhd c.P \mid \mathcal{B}_x[x \rhd c.Q]) \to (x)(P \mid \mathcal{B}_x[Q])$$

sequence of output prefixes on x

Asynchronous subtyping: use **explicit coercions**

classical linear logic

$$\frac{S^\perp = T}{x \leftrightarrow y \vdash x : S, y : T}$$

our type system

$$\frac{S^\perp \leqslant T}{x \leftrightarrow y \vdash x : S, y : T}$$

(closure under duality)

properties of well-typed processes

The usual stuff

- subject reduction
- deadlock freedom
- weak termination

In addition

- if $P \vdash \emptyset$, then P is **orphan message free**

Proof

- Suppose $P \to^* Q$ where Q contains some floating message c
- $Q \vdash \emptyset$ (subject reduction)
- $Q \to^*$ done (no deadlocks & weak termination)
- c must have been consumed (messages don't vanish)

concluding remarks

Fair asynchronous subtyping

- coinductive LTS
- simple characterization *à la* Gay and Hole [2005]
- nice properties (variance, closure, unbounded anticipation)

Type system

- based on "asynchronous" linear logic
- ensures weak termination hence absence of orphan messages

concluding remarks

Fair asynchronous subtyping

- coinductive LTS
- simple characterization *à la* Gay and Hole [2005]
- nice properties (variance, closure, unbounded anticipation)

Type system

- based on "asynchronous" linear logic
- ensures weak termination hence absence of orphan messages

Conjecture

- easy extension to multiparty sessions

concluding remarks

Fair asynchronous subtyping

- coinductive LTS
- simple characterization *à la* Gay and Hole [2005]
- nice properties (variance, closure, unbounded anticipation)

Type system

- based on "asynchronous" linear logic
- ensures weak termination hence absence of orphan messages

Conjecture

- easy extension to multiparty sessions

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.

Tzu-Chun Chen, Mariangiola Dezani-Ciancaglini, Alceste Scalas, and Nobuko Yoshida. On the preciseness of subtyping in session types. *Log. Methods Comput. Sci.*, 13(2), 2017.

references (cont.)

Francesco Dagnino and Luca Padovani. small caps: An infinitary linear logic for a calculus of pure sessions. In Alessandro Bruni, Alberto Momigliano, Matteo Pradella, Matteo Rossi, and James Cheney, editors, *Proceedings of the 26th International Symposium on Principles and Practice of Declarative Programming, PPDP 2024, Milano, Italy, September 9-11, 2024*, pages 4:1–4:13. ACM, 2024.

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`.

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

Silvia Ghilezan, Jovanka Pantovic, Ivan Prokic, Alceste Scalas, and Nobuko Yoshida. Precise subtyping for asynchronous multiparty sessions. *ACM Trans. Comput. Log.*, 24(2):14:1–14:73, 2023.

references (cont.)

Dimitris Mostrous, Nobuko Yoshida, and Kohei Honda. Global principal typing in partially commutative asynchronous sessions. In Giuseppe Castagna, editor, *Programming Languages and Systems, 18th European Symposium on Programming, ESOP 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5502 of *Lecture Notes in Computer Science*, pages 316–332. Springer, 2009.