Session Type Systems Compared: The Case of Deadlock Freedom

Ornela Dardha[1]*and Jorge A. Pérez[2]

[1] School of Computing Science, University of Glasgow, UK
ornela.dardha@glasgow.ac.uk
[2] University of Groningen, The Netherlands
j.a.perez@rug.nl

Abstract

This note summarizes our recent work [6], in which we develop a comparative study of different type systems for message-passing processes that guarantee deadlock freedom. We actually compare two classes of deadlock-free typed processes, denoted $\mathcal{L}$ and $\mathcal{K}$. The class $\mathcal{L}$ stands out for its canonicity: it results from Curry-Howard interpretations of classical linear logic propositions as session types. The class $\mathcal{K}$, obtained by encoding session types into Kobayashi's linear types with usages, includes processes not typable in other type systems. We show that $\mathcal{L}$ is strictly included in $\mathcal{K}$, and identify the precise conditions under which they coincide. We also provide two type-preserving translations of processes in $\mathcal{K}$ into processes in $\mathcal{L}$.

1 Introduction

We are interested in formally relating different type systems for concurrent processes specified in the π-calculus [10]. More precisely, our interest is in *session-based concurrency*, the model of concurrency captured by session types. Session types promote a type-based approach to communication correctness: dialogues between participants are structured into *sessions*, basic communication units; descriptions of interaction sequences are then abstracted as session types which are checked against process specifications. In session-based concurrency, types enforce correct communications through different safety and liveness properties. Two basic (and intertwined) correctness properties are *communication safety* and *session fidelity*. A very desirable liveness property for safe processes is that they should never "get stuck", namely the property of *deadlock freedom*.

In our recent work [6], we present the *first formal comparison* between different type systems for the π-calculus that enforce liveness properties related to (dead)lock freedom. More concretely, we compare $\mathcal{L}$ and $\mathcal{K}$, two salient classes of deadlock-free (session) typed processes, which are induced by different type systems:

- The class $\mathcal{L}$ contains session processes that are well-typed under the Curry-Howard correspondence between (classical) linear logic propositions and session types [1, 2, 12]. Requiring well-typedness suffices, because the type system derived from such a correspondence simultaneously ensures communication safety, session fidelity, and deadlock freedom.
- The class $\mathcal{K}$ contains session processes that enjoy communication safety and session fidelity (as ensured by the type system of Vasconcelos [11]) as well as satisfy deadlock freedom. This class of processes is defined indirectly, by combining Kobayashi's type system based on usages [7, 8, 9] with encodability results by Dardha et al. [5].

*Work supported by the EU H2020 MSCA RISE project BehAPI and by the Dutch Research Council (NWO) under project No. 016.Vidi.189.046 (Unifying Correctness for Communicating Software).

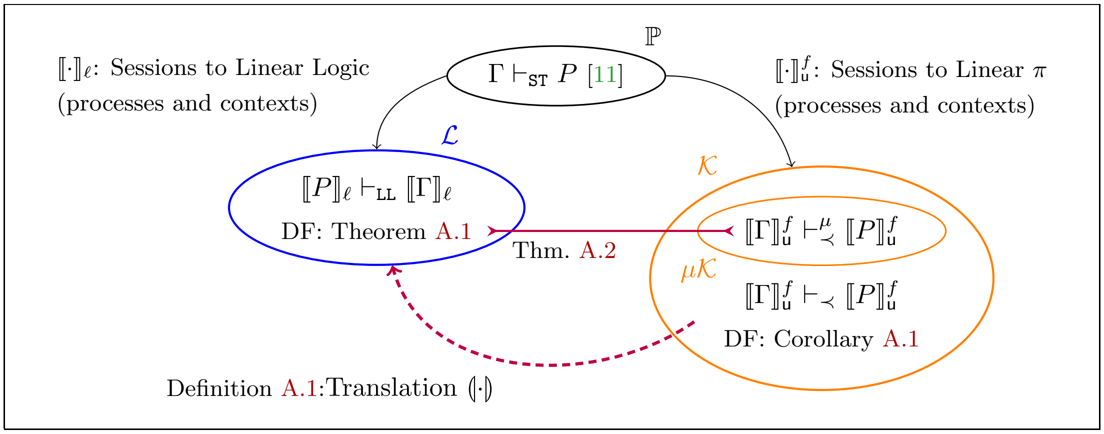

Figure 1: Summary of type systems, languages with deadlock freedom (DF), and encodings between them (indicated by solid black arrows). Main results are denoted by purple lines: our separation result, based on the coincidence of $\mathcal{L}$ and $\mu\mathcal{K}$ is indicated by the solid line with reversed arrowheads; our unifying result is indicated by the dashed arrow.

2 Contributions

Our work develops two kinds of technical results, summarized by Figure 1. On the one hand, we give results that *separate* the classes $\mathcal{L}$ and $\mathcal{K}$ by precisely characterizing the fundamental differences between them; on the other hand, we precisely explain how to *unify* these classes by showing how their differences can be overcome to translate processes in $\mathcal{K}$ into processes into $\mathcal{L}$. More in details:

- To *separate* $\mathcal{L}$ from $\mathcal{K}$, we define $\mu\mathcal{K}$: a sub-class of $\mathcal{K}$ whose definition internalizes the key aspects of the Curry-Howard interpretations of session types. In particular, $\mu\mathcal{K}$ adopts the principle of "composition plus hiding", a distinguishing feature of the interpretations in [1, 12], by which concurrent cooperation is restricted to the sharing of *exactly one* session channel.

  We show that $\mathcal{L}$ and $\mu\mathcal{K}$ coincide (Theorem A.2). This gives us a separation result: there are deadlock-free session processes that *cannot* be typed by systems derived from the Curry-Howard interpretation of session types [1, 2, 12], but that are admitted as typable by the (indirect) approach of [3, 4].

- To *unify* $\mathcal{L}$ and $\mathcal{K}$, we define two *translations* of processes in $\mathcal{K}$ into processes in $\mathcal{L}$ (Definition A.1). Intuitively, because the difference between $\mathcal{L}$ and $\mathcal{K}$ lies in the forms of parallel composition they admit (restricted in $\mathcal{L}$, liberal in $\mathcal{K}$), it is natural to transform a process in $\mathcal{K}$ into another, more parallel process in $\mathcal{L}$. In essence, the first translation, denoted $(\!|\cdot|\!)$ (Definition A.1), exploits type information to replace sequential prefixes with representative parallel components; the second translation refines this idea by considering *value dependencies*, i.e., causal dependencies between independent sessions not captured by types. We detail the first translation, which satisfies type-preservation and operational correspondence properties (Theorems A.3 and A.4).

References

[1] Luís Caires and Frank Pfenning. Session types as intuitionistic linear propositions. In *CONCUR 2010*, volume 6269 of *LNCS*, pages 222–236. Springer, 2010.

[2] Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logic propositions as session types. *Mathematical Structures in Computer Science*, 26(3):367–423, 2016.

[3] Marco Carbone, Ornela Dardha, and Fabrizio Montesi. Progress as compositional lock-freedom. In *Coordination Models and Languages - 16th IFIP WG 6.1 International Conference, COORDINATION 2014, Held as Part of the 9th International Federated Conferences on Distributed Computing Techniques, DisCoTec*, volume 8459 of *LNCS*, pages 49–64. Springer, 2014.

[4] Ornela Dardha. *Type Systems for Distributed Programs: Components and Sessions*, volume 7 of *Atlantis Studies in Computing*. Springer / Atlantis Press, 2016.

[5] Ornela Dardha, Elena Giachino, and Davide Sangiorgi. Session types revisited. In *PPDP'12*, pages 139–150. ACM, 2012.

[6] Ornela Dardha and Jorge A. Pérez. Comparing type systems for deadlock freedom. *J. Log. Algebraic Methods Program.*, 124:100717, 2022.

[7] Naoki Kobayashi. A type system for lock-free processes. *Inf. Comput.*, 177(2):122–159, 2002.

[8] Naoki Kobayashi. A new type system for deadlock-free processes. In *CONCUR 2006*, volume 4137 of *LNCS*, pages 233–247. Springer, 2006. Full version available at `http://www-kb.is.s.u-tokyo.ac.jp/~koba/papers/concur2006-full.pdf`.

[9] Naoki Kobayashi. Type systems for concurrent programs. Extended version of [?], Tohoku University. `www.kb.ecei.tohoku.ac.jp/~koba/papers/tutorial-type-extended.pdf`, 2007.

[10] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I. *Inf. Comput.*, 100(1):1–40, 1992.

[11] Vasco T. Vasconcelos. Fundamentals of session types. *Inf. Comput.*, 217:52–70, 2012.

[12] Philip Wadler. Propositions as sessions. In *ICFP'12*, pages 273–286, 2012.

A Results

A.1 DF in Sessions, Linear π and Linear Logic

For any P, define $live(P)$ if and only if $P \equiv (\boldsymbol{\nu}\widetilde{n})(\pi.Q \mid R)$, where π is an input, output, selection, or branching prefix.

Theorem A.1 (Deadlock Freedom)**.** *If $P \vdash_{\mathtt{LL}} \cdot$ and $live(P)$ then $P \longrightarrow Q$, for some Q.*

The following result states deadlock freedom by encodability, following [3].

Corollary A.1. *Let $\vdash_{\mathtt{ST}} P$ be a session process. If $\vdash_{\prec} [\![P]\!]^f_{\mathtt{u}}$ is deadlock-free then P is deadlock-free.*

A.2 Relating $\mathcal{L}$, $\mu\mathcal{K}$ and $\mathcal{K}$

Theorem A.2. *$\mathcal{L} = \mu\mathcal{K}$.*

Definition A.1 (Translation into $\mathcal{L}$)**.** *Let P be such that $\Gamma \vdash_{\mathtt{ST}} P$ and $P \in \mathcal{K}$. The set of $\mathcal{L}$ processes $(\!|\Gamma \vdash_{\mathtt{ST}} P|\!)$ is defined in Figure 2.*

We present two important results about our translation. First, it is type preserving, up to the encoding of types:

$$(\!|\Gamma^{\mathsf{un}} \vdash_{\mathsf{ST}} \mathbf{0}|\!) \triangleq \{\mathbf{0}\}$$

$$(\!|\Gamma, x : !T.S, v : T \vdash_{\mathsf{ST}} \overline{x}\langle v\rangle.P'|\!) \triangleq \{\overline{x}(z).([v \leftrightarrow z] \mid Q) \ : \ Q \in (\!|\Gamma, x : S \vdash_{\mathsf{ST}} P'|\!)\}$$

$$(\!|\Gamma_1, \Gamma_2, x : !T.S \vdash_{\mathsf{ST}} (\boldsymbol{\nu} zy)\overline{x}\langle y\rangle.(P_1 \mid P_2)|\!) \triangleq$$
$$\{\overline{x}(y).(Q_1 \mid Q_2) \ : \ Q_1 \in (\!|\Gamma_1, z : \overline{T} \vdash_{\mathsf{ST}} P_1|\!) \wedge Q_2 \in (\!|\Gamma_2, x : S \vdash_{\mathsf{ST}} P_2|\!)\}$$

$$(\!|\Gamma, x : ?T.S \vdash_{\mathsf{ST}} x(y : T).P'|\!) \triangleq \{x(y).Q \ : \ Q \in (\!|\Gamma, x : S, y : T \vdash_{\mathsf{ST}} P'|\!)\}$$

$$(\!|\Gamma, x : \oplus\{l_i : S_i\}_{i\in I} \vdash_{\mathsf{ST}} x \triangleleft l_j.P'|\!) \triangleq \{x \triangleleft l_j.Q \ : \ Q \in (\!|\Gamma, x : S_j \vdash_{\mathsf{ST}} P'|\!)\}$$

$$(\!|\Gamma, x : \&\{l_i : S_i\}_{i\in I} \vdash_{\mathsf{ST}} x \triangleright \{l_i : P_i\}_{i\in I}|\!) \triangleq \{x \triangleright \{l_i : Q_i\}_{i\in I} \ : \ Q_i \in (\!|\Gamma, x : S_i \vdash_{\mathsf{ST}} P_i|\!)\}$$

$$(\!|\Gamma_1, [\widetilde{x : S}] \star \Gamma_2, [\widetilde{y : T}] \vdash_{\mathsf{ST}} (\boldsymbol{\nu}\widetilde{x}\widetilde{y} : \widetilde{S})(P_1 \mid P_2)|\!) \triangleq$$
$$\{C_1[Q_1] \mid G_2 \ : \ Q_1 \in (\!|\Gamma_1, \widetilde{x : S} \vdash_{\mathsf{ST}} P_1|\!),\, C_1 \in \mathcal{C}_{\widetilde{x:T}},\, G_2 \in \langle\!|\Gamma_2|\!\rangle\}$$
$$\cup$$
$$\{G_1 \mid C_2[Q_2] \ : \ Q_2 \in (\!|\Gamma_2, \widetilde{y : T} \vdash_{\mathsf{ST}} P_2|\!),\, C_2 \in \mathcal{C}_{\widetilde{y:S}},\, G_1 \in \langle\!|\Gamma_1|\!\rangle\}$$

Figure 2: Translation $(\!|\cdot|\!)$ (cf. Definition A.1).

Theorem A.3 (The Translation $(\!|\cdot|\!)$ is Type Preserving)**.** *Let $\Gamma \vdash_{\mathsf{ST}} P$. Then, for all $Q \in (\!|\Gamma \vdash_{\mathsf{ST}} P|\!)$, we have that $Q \vdash_{\mathsf{LL}} [\![\Gamma]\!]_\ell$.*

Definition A.2 (Parallelization Relation)**.** *Let P and Q be processes such that $P, Q \vdash_{\mathsf{LL}} \Gamma$. We write $P \doteqdot Q$ if and only if there exist processes P_1, P_2, Q_1, Q_2 and contexts Γ_1, Γ_2 such that the following hold:*

$$P = P_1 \mid P_2 \qquad Q = Q_1 \mid Q_2 \qquad P_1, Q_1 \vdash_{\mathsf{LL}} \Gamma_1 \qquad P_2, Q_2 \vdash_{\mathsf{LL}} \Gamma_2 \qquad \Gamma = \Gamma_1, \Gamma_2$$

By definition, the relation $\doteqdot$ is reflexive. We may now state:

Theorem A.4 (Operational Correspondence for $(\!|\cdot|\!)$)**.** *Let P be such that $\Gamma \vdash_{\mathsf{ST}} P$ for some typing context Γ. Then, we have:*

1. *If $P \to P'$, then for all $Q \in (\!|\Gamma \vdash_{\mathsf{ST}} P|\!)$ there exist Q', R such that $Q \to\hookrightarrow Q'$, $Q' \doteqdot R$, and $R \in (\!|\Gamma \vdash_{\mathsf{ST}} P'|\!)$.*
2. *If $Q \in (\!|\Gamma \vdash_{\mathsf{ST}} P|\!)$, such that $P \in \mathcal{K}$, and $Q \to\hookrightarrow Q'$, then there exist P', R such that $P \to P'$, $Q' \doteqdot R$, and $R \in (\!|\Gamma \vdash_{\mathsf{ST}} P'|\!)$.*