Around Classical and Intuitionistic Linear Processes

Juan C. Jaramillo
Unversity of Groningen, The Netherlands

Dan Frumin
Unversity of Groningen, The Netherlands

Jorge A. Pérez
Unversity of Groningen, The Netherlands

Abstract

Curry-Howard correspondences between Linear Logic (LL) and session types provide a firm foundation for concurrent processes. As the correspondences hold for intuitionistic and classical versions of LL (ILL and CLL), we obtain two different families of type systems for concurrency. An open question remains: *how do these two families exactly relate to each other?* Based upon a translation from CLL to ILL due to Laurent, we provide two complementary answers, in the form of full abstraction results based on a typed observational equivalence due to Atkey. Our results elucidate hitherto missing formal links between seemingly related yet different type systems for concurrency.

2012 ACM Subject Classification Theory of computation → Linear logic; Theory of computation → Type theory; Theory of computation → Process calculi

Keywords and phrases Process calculi, session types, linear logic

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2024.30

Related Version *Full Version*: `http://arxiv.org/abs/2407.06391`

Funding *Juan C. Jaramillo*: Ministry of Science of Colombia (Minciencias).
Jorge A. Pérez: Support of the Dutch Research Council (NWO) under project No.016.Vidi.189.046 (Unifying Correctness for Communicating Software) is gratefully acknowledged.

Acknowledgements We are most grateful to Bas van den Heuvel for initial discussions on the topic of this paper. We also thank the anonymous reviewers for their helpful suggestions.

1 Introduction

We address an open question on the logical foundations of concurrency, as resulting from Curry-Howard correspondences between linear logic (LL) and session types. These correspondences, often referred to as *"propositions-as-sessions"*, connect LL propositions and session types, proofs in LL and π-calculus processes, as well as cut-elimination in LL and process synchronization. The result is type systems that elegantly ensure important forms of communication correctness for processes. The correspondence was discovered by Caires and Pfenning, who relied on an *intuitionistic* presentation of LL (ILL) [8]; Wadler later presented it using *classical* LL (CLL) [24]. These two works triggered the emergence of multiple type systems for concurrency with firm logical foundations, based on (variants of) ILL (e.g., [21, 17, 3, 7]) and CLL (e.g., [6, 12, 15, 14, 19]). While key differences between these two families of type systems, intuitionistic and classical, have been observed [22], in this paper we ask: can we formally relate them from the standpoint of (typed) process calculi?

From a logical standpoint, the mere existence of two different families of type systems may seem surprising – after all, the relationship between ILL and CLL is well understood [20, 11, 16]. Laurent has given a thorough account of these relationships, including a translation from CLL to ILL [16]. A central insight in our work is the following: while translations from

$$\begin{tikzcd}
\boxed{\vdash^c \Delta} \arrow[rr, dashed, "(-)^{\bullet\perp}_{\mathsf{R}}\ [\S\ 3]"] \arrow[dr, "(-)^{\bullet}_{\mathsf{R}}\ [16]"'] & & \boxed{\vdash^c \Delta^{\bullet\perp}_{\mathsf{R}}, \mathsf{R}} & \mathsf{CLL} \\
& \boxed{\Delta^{\bullet}_{\mathsf{R}} \vdash^i \mathsf{R}} \arrow[ur, "(-)^{\perp}"'] & & \mathsf{ILL}
\end{tikzcd}$$

Figure 1 Translations between CLL and ILL. In this paper, we shall fix $\mathsf{R} = \mathbf{1}$.

CLL to ILL are useful, they alone do not entail formal results for typed processes, and a satisfactory answer from the "propositions-as-sessions" perspective must include process calculi considerations.

Let us elaborate. Given some context Δ and a formula A, let us write $\vdash^c \Delta$ and $\Delta \vdash^i A$ to denote sequents in CLL and ILL, respectively. Under the concurrent interpretation induced by "propositions-as-sessions", these sequents are annotated as $P \vdash^c \Delta$ and $\Delta \vdash^i P :: x : A$, respectively, where P is a process, x is a name, and Δ is now a finite collection of assignments $x_1 : A_1, \ldots, x_n : A_n$. An assignment specifies a name's intended session protocol. This way, e.g., "$x : A \otimes B$" (resp. "$x : A ⅋ B$") says that x *outputs* (resp. *inputs*) a name of type A before continuing as described by B. Also, "$x : \mathbf{1}$" says that x has completed its protocol and has no observable behavior. The judgment $\Delta \vdash^i P :: x : A$ has a rely-guarantee flavor: "P *relies* on the behaviors described by Δ to *offer* a protocol A on x". Hence, the assignment $x : A$ in the right-hand side plays a special role: this is *the* only observation made about the behavior of P. Differently, the judgment $P \vdash^c \Delta$ simply reads as "P implements the behaviors described by Δ"; as such, all assignments in Δ are equally relevant for observing the behavior of P.

Unsurprisingly, these differences between intuitionistic and classical processes arise in their associated (typed) behavioral equivalences [18, 15, 1, 13]. For intuitionistic processes, theories of *logical relations* [18, 13] induce contextual equivalences in which only the right-hand side assignment matters in comparisons; the assignments in Δ are used to construct appropriate contexts. For classical processes, we highlight Atkey's *observed communication semantics* [1], whose induced observational equivalence accounts for the entire typing context.

Laurent's *negative translation* from CLL to ILL [16], denoted $(-)^{\bullet}_{\mathsf{R}}$, translates formulas using the parameter R (an arbitrary formula in ILL) as a "residual" element. We have, e.g.,:

$$(A \otimes B)^{\bullet}_{\mathsf{R}} = ((A^{\bullet}_{\mathsf{R}} \multimap \mathsf{R}) \otimes (B^{\bullet}_{\mathsf{R}} \multimap \mathsf{R})) \multimap \mathsf{R}$$

As Figure 1 shows, using $(-)^{\bullet}_{\mathsf{R}}$ we can transform $\vdash^c \Delta$ into $(\Delta)^{\bullet}_{\mathsf{R}} \vdash^i \mathsf{R}$. Now, from the view of "propositions-as-sessions", we see that $(-)^{\bullet}_{\mathsf{R}}$ increases the size of formulas/protocols and that fixing $\mathsf{R} = \mathbf{1}$ results into the simplest residual protocol. Given this, Laurent's translation transforms $P \vdash^c \Delta$ into $(\Delta)^{\bullet}_{\mathsf{R}} \vdash^i (P)^{\bullet} :: w : \mathbf{1}$ (for some fresh z), where $(P)^{\bullet}$ is a process that reflects the translation. The translation has an unfortunate effect, however: a classical process P with observable behavior given by Δ is transformed into an intuitionistic process $(P)^{\bullet}$ *without observable behavior* (given by $w : \mathbf{1}$). We conclude that, independently of the chosen R, the translation $(-)^{\bullet}_{\mathsf{R}}$ alone does not adequately relate the concurrent interpretations of CLL and ILL, as it does not uniformly account for observable behavior in P and $(P)^{\bullet}$.

Our goal is to complement the scope of Laurent's translation, in a way that is consistent with existing theories of (typed) behavioral equivalence for logic-based processes.

We proceed in two steps, shown in Figure 1. In the following, we shall fix $\mathsf{R} = \mathbf{1}$ and omit "R" when clear from the context. First, there is a well-known translation from ILL to CLL, denoted $(-)^{\perp}$, under which a sequent $\Delta \vdash^{i} A$ is transformed into $\vdash^{c} \Delta^{\perp}, A$. Our observation is that $(-)^{\bullet\perp}$ (the composition of the two translations) goes from CLL into itself, translating $\vdash^{c} \Delta$ into $\vdash^{c} \Delta^{\bullet\perp}, \mathbf{1}$. At the level of processes, this allows us to consider the corresponding processes P and $(P)^{\bullet}$ in the common setting of classical processes. To reason about their observable behavior we employ Atkey's observational equivalence [1], denoted "$\simeq$".

Our second step leads to our **main contributions**: two full abstraction results that connect $\vdash^{c} \Delta$ and $\vdash^{c} \Delta^{\bullet\perp}, \mathbf{1}$ from the perspective of "propositions-as-sessions".

- The first result, given in § 3, adopts a *denotational* approach to ensure that P (typable with $\vdash^{c} \Delta$) and $(P)^{\bullet}$ (typable with *both* $\Delta^{\bullet} \vdash^{i} \mathbf{1}$ *and* $\vdash^{c} \Delta^{\bullet\perp}, \mathbf{1}$) are behaviorally equivalent. This full abstraction result ensures that $P \simeq Q$ iff $(P)^{\bullet} \simeq (Q)^{\bullet}$ (Corollary 3.12).
- The second result, given in § 4, is an *operational* bridge between $\vdash^{c} \Delta$ and $\vdash^{c} \Delta^{\bullet\perp}, \mathbf{1}$: Corollary 4.15 ensures that $P \simeq Q$ iff $C[P] \simeq C[Q]$, where C is a so-called *transformer context*, which "adapts" observable behavior in processes using types in Δ.

Next, we recall CP (Wadler's concurrent interpretation of CLL), Atkey's observational equivalence, and Laurent's translation. § 3 and § 4 develop our full abstraction results. § 5 further discusses our contributions; in particular, we discuss how they are related to the *locality* principle – one of the known distinguishing features between typed processes based on "propositions-as-sessions" [22, 9, 25].

2 Background

Propositions-as-Sessions / Classical Processes (CP). We shall work with *classical processes* (CP) as proposed by Wadler [24]. Assuming an infinite set of *names* $(x, y, z, \ldots)$, the set of *processes* $(P, Q, \ldots)$ is defined as follows:

$$\begin{aligned} P, Q ::= {} & \mathbf{0} \mid (\nu x)P \mid P \mid Q \mid [x \leftrightarrow y] \mid x[y].(P \mid Q) \mid x(y).P \mid !x(y).P \mid ?x[y].P \\ & \mid x[i].P \mid x.\mathit{case}(P, Q) \mid x[\,] \mid x(\,).P \qquad \text{for } i \in \{1, 2\} \end{aligned}$$

We write $P\{x/y\}$ to denote the capture-avoiding substitution of y for x in P. We have usual constructs for inaction, restriction, and parallel composition. The forwarder $[x \leftrightarrow y]$ equates x and y. We then have $x[y].(P \mid Q)$ (send the restricted name y along x, proceed as $P \mid Q$) and $x(y).P$ (receive a name z along x, proceed as $P\{z/y\}$). Processes $!x(y).P$ and $?x[y].P$ denote a replicated input (server) and a client request, respectively. Process $x[i].P$ denotes the selection of one of the two alternatives of a corresponding branching process $x.\mathit{case}(P, Q)$. Processes $x[\,]$ and $x(\,).P$ enable coordinated closing of the session along x. In a statement, a name is *fresh* if it is not among the names of the objects of the statement (e.g., processes).

In $(\nu x)P$, name x is bound in P; also, in $x[y].(P \mid Q)$, $x(y).P$, $!x(y).P$, and $?x[y].P$, name y is bound in P but not in Q.

The types are assigned to names and correspond to the following formulas of CLL:

$$A, B ::= \mathbf{1} \mid \perp \mid A \otimes B \mid A \parr B \mid A \oplus B \mid A \& B \mid !A \mid ?A$$

The assignment $x : A$ says that the session protocol through x goes as described by A. As we have seen, $x : A \otimes B$ and $x : A \parr B$ are read as sending and receiving along x, respectively. Also, $x : A \oplus B$ denotes the selection of either A or B along x, whereas $x : A \& B$ denotes the offer of A and B along x. Finally, $x : !A$ and $x : ?A$ assign server and client behaviors to x, respectively. There then is a clear duality in the interpretation of the following pairs: $\otimes$

$$\frac{}{[x \leftrightarrow y] \vdash^c x : A, y : A^\perp}\ \textsc{Id} \qquad \frac{}{x[\,] \vdash^c x : \mathbf{1}}\ 1 \qquad \frac{P \vdash^c \Gamma}{x(\,).P \vdash^c \Gamma, x : \bot}\ \bot$$

$$\frac{P \vdash^c \Gamma, y : A \qquad Q \vdash^c \Delta, x : B}{x[y].(P \mid Q) \vdash^c \Gamma, \Delta, x : A \otimes B}\ \otimes \qquad \frac{P \vdash^c \Gamma, y : A, x : B}{x(y).P \vdash^c \Gamma, x : A \parr B}\ \parr \qquad \frac{P \vdash^c \Gamma, x : A_i}{x[i].P \vdash^c \Gamma, x : A_1 \oplus A_2}\ \oplus_i$$

$$\frac{P \vdash^c \Gamma, x : A_1 \qquad Q \vdash^c \Gamma, x : A_2}{x.case(P, Q) \vdash^c \Gamma, x : A \& B}\ \& \qquad \frac{P \vdash^c ?\Delta, y : A}{!x(y).P \vdash^c ?\Delta, x : !A}\ !$$

$$\frac{P \vdash^c \Delta, x : A}{?x[y].P \vdash^c \Delta, x : ?A}\ ? \qquad \frac{P \vdash^c \Delta, x_1 : ?A, x_2 : ?A}{P\{x_1/x_2\} \vdash^c \Delta, x_1 : ?A}\ \mathrm{C} \qquad \frac{P \vdash^c \Gamma}{P \vdash^c \Gamma, x : ?A}\ \mathrm{W}$$

$$\frac{P \vdash^c \Gamma, x : A \qquad Q \vdash^c \Delta, x : A^\perp}{(\nu x)(P \mid Q) \vdash^c \Gamma, \Delta}\ \textsc{Cut} \qquad \frac{P \vdash^c \Delta \qquad Q \vdash^c \Gamma}{P \mid Q \vdash^c \Delta, \Gamma}\ \textsc{Mix}_2 \qquad \frac{}{\mathbf{0} \vdash^c \cdot}\ \textsc{Mix}_0$$

Figure 2 Typing rules. CP does not include "mix". CP_0 is CP + $\textsc{Mix}_0$, CP_{02} is CP_0 + $\textsc{Mix}_2$.

and $\parr$; $\oplus$ and $\&$; and ! and ?. It reflects reciprocity between the behavior of a name: when a process on one side sends, the process on the opposite side must receive, and vice versa. Formally, the dual type of A, denoted $A^\perp$, is defined as

$$\begin{array}{llll} \mathbf{1}^\perp := \bot & (A \otimes B)^\perp := A^\perp \parr B^\perp & (A \& B)^\perp := A^\perp \oplus B^\perp & (!A)^\perp := ?A^\perp \\ \bot^\perp := \mathbf{1} & (A \parr B)^\perp := A^\perp \otimes B^\perp & (A \oplus B)^\perp := A^\perp \& B^\perp & (?A)^\perp := !A^\perp \end{array}$$

Duality is an involution, i.e., $(A^\perp)^\perp = A$. We write Δ, Γ to denote *contexts*, a finite collection of assignments $x : A$. The empty context is denoted " $\cdot$ ". The typing judgments are then of the form $P \vdash^c \Delta$, with typing rules as in Figure 2. For technical convenience, we shall consider mix principles (rules $\textsc{Mix}_0$ and $\textsc{Mix}_2$, not included in [24]), which enable the typing of useful forms of process composition. This way, in § 3 we will consider CP_0: the variant of CP with $\textsc{Mix}_0$; in § 4 we will consider CP_{02}: the extension of CP_0 with $\textsc{Mix}_2$.

Note that the type system CP_0 corresponds exactly to the sequent calculus for CLL if we ignore name and process annotations. This correspondence goes beyond typing: an important aspect of "propositions-as-sessions" is that the dynamic behavior of processes (process reductions) corresponds to simplification of proofs (cut elimination). In the following we will not need this reduction semantics, which can be found in, e.g., [24]. Rather, we will use the denotational semantics of CP_0 as defined by Atkey [1], which we recall next.

Denotational Semantics for CP. We adopt Atkey's denotational semantics for CP_0 [1], which allows us to reason about observational equivalence of processes. Here we recall the notions of configurations, observations, and denotations as needed for our purposes.

The observational equivalence on processes relies on the notion of *configuration*, which is a process that has some of its names selected for the purposes of "observations". Configurations, defined in Figure 3, are typed as $C \vdash_{\mathsf{cfg}} \Delta \mid \Theta$, where Δ contains free/unconnected names, and Θ contains the names that we intend to observe. Rule C:$\textsc{Cut}$, for example, states that we can compose two configurations along a name x, and make the name observable.

$$\textsc{C:Proc}\ \frac{P \vdash^{c} \Gamma}{P \vdash_{\mathsf{cfg}} \Gamma \mid \cdot} \qquad \textsc{C:Cut}\ \frac{C_1 \vdash_{\mathsf{cfg}} \Gamma_1, x : A \mid \Theta_1 \qquad C_2 \vdash_{\mathsf{cfg}} \Gamma_2, x : A^{\perp} \mid \Theta_2}{C_1 \mid_x C_2 \vdash_{\mathsf{cfg}} \Gamma_1, \Gamma_2 \mid \Theta_1, \Theta_2, x : A} \qquad \textsc{C:0}\ \frac{}{\underline{\mathbf{0}} \vdash_{\mathsf{cfg}} \cdot \mid \cdot}$$

$$\textsc{C:W}\ \frac{C \vdash_{\mathsf{cfg}} \Gamma \mid \Theta}{C \vdash_{\mathsf{cfg}} \Gamma, x : ?A \mid \Theta} \qquad \textsc{C:Con}\ \frac{C \vdash_{\mathsf{cfg}} \Gamma, x_1 : ?A, x_2 : ?A \mid \Theta}{C\{x_1/x_2\} \vdash_{\mathsf{cfg}} \Gamma, x_1 : ?A \mid \Theta}$$

Figure 3 Classical Processes: Configurations.

$$\frac{}{\underline{\mathbf{0}} \Downarrow ()}\ \textsc{Stop} \qquad \frac{C[C'[\{x/y\}]] \Downarrow \theta[x \mapsto a]}{C[[x \leftrightarrow y] \mid_x C'] \Downarrow \theta[x \mapsto a, y \mapsto a]}\ \textsc{Link} \qquad \frac{C[P \mid_x Q] \Downarrow \theta[x \mapsto a]}{C[(\nu x)(P \mid Q)] \Downarrow \theta}\ \textsc{Comm}$$

$$\frac{C[\underline{\mathbf{0}}] \Downarrow \theta}{C[\mathbf{0}] \Downarrow \theta}\ 0 \qquad \frac{C[P \mid_y (Q \mid_x R)] \Downarrow \theta[x \mapsto a, y \mapsto b]}{C[x[y].(P \mid Q) \mid_x x(y).R] \Downarrow \theta[x \mapsto (a,b)]}\ \otimes⅋$$

$$\frac{C[P] \Downarrow \theta}{C[x[\,] \mid_x x(\,).P] \Downarrow \theta[x \mapsto *]}\ 1\perp \qquad \frac{C[P \mid_x Q_i] \Downarrow \theta[x \mapsto a]}{C[x[i].P \mid_x x.case(Q_0, Q_1)] \Downarrow \theta[x \mapsto (i,a)]}\ \oplus\&$$

$$\frac{C[P \mid_y Q] \Downarrow \theta[y \mapsto a]}{C[!x(y).P \mid_x ?x[y].Q] \Downarrow \theta[x \mapsto \lbag a \rbag]}\ !? \qquad \frac{C[C'] \Downarrow \theta}{C[!x(y).P \mid_x C'] \Downarrow \theta[x \mapsto \emptyset]}\ !\mathrm{W}$$

$$\frac{C[!x_1(y).P \mid_{x_1} (!x_2(y).P \mid_{x_2} C')] \Downarrow \theta[x_1 \mapsto \alpha, x_2 \mapsto \beta]}{C[!x_1(y).P \mid_{x_1} C'\{x_1/x_2\}] \Downarrow \theta[x_1 \mapsto \alpha \uplus \beta]}\ !\mathrm{C} \qquad \frac{C' \Downarrow \theta \qquad C \equiv C'}{C \Downarrow \theta}\ \equiv$$

Figure 4 Classical Processes: Observations.

Observations for configurations are given in Figure 4. The observation relation $C \Downarrow \theta$ is defined for closed configurations $C \vdash_{\mathsf{cfg}} \cdot \mid \Theta$, and the shape observation $\theta \in \llbracket \Theta \rrbracket$ is defined based on the shape of types in Θ. For a type A, the set of observations $\llbracket A \rrbracket$ is defined as

$$\llbracket \mathbf{1} \rrbracket = \llbracket \perp \rrbracket = \{*\} \qquad \llbracket !A \rrbracket = \llbracket ?A \rrbracket = \mathcal{M}_f(\llbracket A \rrbracket)$$

$$\llbracket A \otimes B \rrbracket = \llbracket A ⅋ B \rrbracket = \llbracket A \rrbracket \times \llbracket B \rrbracket \qquad \llbracket A_0 \oplus A_1 \rrbracket = \llbracket A_0 \& A_1 \rrbracket = \sum_{i \in \{0,1\}} \llbracket A_i \rrbracket$$

and we set $\llbracket \Theta \rrbracket = \llbracket x_1 : A_1, \cdots, x_n : A_n \rrbracket = \llbracket A_1 \rrbracket \times \cdots \times \llbracket A_n \rrbracket$. Here $\mathcal{M}_f(X)$ denotes finite multisets with elements from X. We use the standard notations $\emptyset$, $\uplus$, and $\lbag a_1, \ldots, a_n \rbag$ to denote the empty multiset, multiset union, and multiset literals, respectively.

If $\theta \in \llbracket x_1 : A_1, \cdots, x_n : A_n \rrbracket$, then we write $\theta[x_i \mapsto \theta_i]$ for the observation which is identical to θ, except that its ith component is set to θ_i. In Figure 4, Rule Stop says that $\mathbf{0}$ has no observations; Rule $\otimes⅋$ collects observations a and b into a single observation (a, b).

Using these notions, there is an immediate canonical notion of observational equivalence. In the following, we write $P, Q \vdash^c \Gamma$ whenever $P \vdash^c \Gamma$ and $Q \vdash^c \Gamma$ hold.

▶ **Definition 2.1** (Observational equivalence [1]). *Let $P, Q \in CP_0$ such that $P, Q \vdash^c \Gamma$. They are observationally equivalent, written $P \simeq Q$, if for all configurations-process context $C[-]$ where $C[P], C[Q] \vdash^c \cdot \mid \Theta$, and all $\theta \in \llbracket \Theta \rrbracket$, $C[P] \Downarrow \theta \iff C[Q] \Downarrow \theta$.*

$$[\![[x \leftrightarrow y] \vdash^c x : A, y : A^\perp]\!] = \{(a,a) \mid a \in [\![A]\!]\} \qquad [\![x[\,] \vdash^c x : \mathbf{1}]\!] = \{(*)\} \qquad [\![\mathbf{0} \vdash^c \ ]\!] = \{()\}$$

$$[\![x(\,).P \vdash^c \Gamma, x : \bot]\!] = \{(\gamma, *) \mid \gamma \in [\![P \vdash^c \Gamma]\!]\}$$

$$[\![(\nu x)(P \mid Q) \vdash^c \Gamma, \Delta]\!] = \{(\gamma, \delta) \mid (\gamma, a) \in [\![P \vdash^c \Gamma, x : A]\!], (\delta, a) \in [\![Q \vdash^c \Delta, x : A^\perp]\!]\}$$

$$[\![x[y].(P \mid Q) \vdash^c \Gamma, \Delta, x : A \otimes B]\!] = \left\{ (\gamma, \delta, (a,b)) \mid \begin{array}{l} (\gamma, a) \in [\![P \vdash^c \Gamma, y : A]\!], \\ (\delta, b) \in [\![Q \vdash^c \Delta, x : B]\!] \end{array} \right\}$$

$$[\![x(y).P \vdash^c \Delta, x : A \parr B]\!] = \{(\gamma, \delta, (a,b)) \mid (\gamma, a, b) \in [\![P \vdash^c \Delta, y : A, x : B]\!]\}$$

$$[\![x[i].P \vdash^c \Gamma, x : A_1 \oplus A_2]\!] = \{(\gamma, (i,a)) \mid (\gamma, a) \in [\![P \vdash^c \Gamma, x : A_i]\!]\}$$

$$[\![x.case(P_1, P_2) \vdash^c \Gamma, x : A_1 \& A_2]\!] = \bigcup_{i \in \{1,2\}} \{(\gamma, (i,a)) \mid (\gamma, a) \in [\![P_i \vdash^c \Gamma, x : A_i]\!]\}$$

$$[\![!x(y).P \vdash^c ?\Delta, x : !A]\!] = \left\{ \begin{array}{l} (\uplus_{j=1}^{k} \alpha_j^1, \ldots, \uplus_{j=1}^{k} \alpha_j^n, \wr a_1, \ldots, a_n \wr) \\ \mid \forall i \in \{1, \ldots, k\}. (\alpha_i^1, \ldots, \alpha_i^k, a_i) \in [\![P \vdash^c ?\Delta, y : A]\!] \end{array} \right\}$$

$$[\![?x[y].P \vdash^c \Gamma, x : ?A]\!] = \{(\gamma, \wr a \wr) \mid (\gamma, a) \in [\![P \vdash^c \Gamma, y : A]\!]\}$$

$$[\![P \vdash^c \Gamma, x : ?A]\!] = \{(\gamma, \emptyset) \mid \gamma \in [\![P \vdash^c \Gamma]\!]\}$$

$$[\![P\{x_1/x_2\} \vdash^c \Gamma, x_1 : ?A]\!] = \{(\gamma, \alpha_1 \uplus \alpha_2) \mid (\gamma, \alpha_1, \alpha_2) \in [\![P \vdash^c \Gamma, x_1 : ?A, x_2 : ?A]\!]\}$$

Figure 5 Classical Processes: Denotational Semantics.

We take this notion of observational equivalence as *the* equivalence of CP_0, sometimes writing $P \simeq Q \vdash^c \Gamma$ to emphasize the typing of processes we are comparing. Establishing observational equivalence of two processes directly is complicated, due to the universal quantification over all potential configurations C. To establish equivalence in a compositional way, we recall Atkey's notion of denotational semantics for CP in Figure 5: it assigns to each process $P \vdash^c \Delta$ a denotation $[\![P \vdash^c \Delta]\!]$ as a subset of observations $[\![\Delta]\!]$ on its names. When the typing of a process P is clear from the context we simply write $[\![P]\!] \subseteq [\![\Delta]\!]$. The denotational semantics are sound and complete w.r.t. the observations:

▶ **Theorem 2.2** (Adequacy [1]). *If $C \vdash^c \cdot \mid \Theta$, then $C \Downarrow \theta$ iff $\theta \in [\![C \vdash^c \cdot \mid \Theta]\!]$.*

Hence, we can use denotational semantics to prove observational equivalence:

▶ **Corollary 2.3** ([1]). *If $P, Q \vdash^c \Gamma$ and $[\![P]\!] = [\![Q]\!]$, then $P \simeq Q$.*

Above, the condition $P, Q \vdash^c \Gamma$ is important, as there are processes with different types that have the same denotations. Examples are $x[1].x[\,] \vdash^c x : \mathbf{1} \oplus \mathbf{1}$ and $x.case(x[\,], x[\,]) \vdash^c x : \mathbf{1} \& \mathbf{1}$.

▶ Remark 2.4. Atkey shows that $\simeq$ captures many equalities on processes induced by proof transformations, such as cut permutations and commuting conversions; see [1, Sect. 5].

Laurent's Translation $(-)^{\bullet}_{\mathsf{R}}$. As mentioned above, Laurent gives a parametric translation from CLL to ILL. Here we recall this translation following [16, § 2.1], considering only the class of formulas needed for our purposes. The formulas of ILL are built using the grammar:

$$I, J ::= \mathbf{1} \mid I \otimes J \mid I \multimap J \mid I \oplus J \mid I \& J \mid !I$$

Table 1 Translations $(-)^{\bullet}_{\mathsf{R}}$ and $(-)^{\bullet\perp}_{\mathsf{R}}$.

F	$F^{\bullet}_{\mathsf{R}}$ (ILL)	$F^{\bullet}_{\mathsf{R}}$ (CLL)	$F^{\bullet\perp}_{\mathsf{R}}$
$\perp$	$\mathbf{1}$	$\mathbf{1}$	$\perp$
$\mathbf{1}$	$\mathbf{1} \multimap \mathsf{R}$	$\perp ⅋ \mathsf{R}$	$\mathbf{1} \otimes \mathsf{R}^{\perp}$
$A \otimes B$	$((A^{\bullet}_{\mathsf{R}} \multimap \mathsf{R}) \otimes (B^{\bullet}_{\mathsf{R}} \multimap \mathsf{R})) \multimap \mathsf{R}$	$((A^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R}) \otimes (B^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R}))^{\perp} ⅋ \mathsf{R}$	$((A^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R}) \otimes (B^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R})) \otimes \mathsf{R}^{\perp}$
$A ⅋ B$	$A^{\bullet}_{\mathsf{R}} \otimes B^{\bullet}_{\mathsf{R}}$	$A^{\bullet}_{\mathsf{R}} \otimes B^{\bullet}_{\mathsf{R}}$	$A^{\bullet\perp}_{\mathsf{R}} ⅋ B^{\bullet\perp}_{\mathsf{R}}$
$A \oplus B$	$((A^{\bullet}_{\mathsf{R}} \multimap \mathsf{R}) \oplus (B^{\bullet}_{\mathsf{R}} \multimap \mathsf{R})) \multimap \mathsf{R}$	$((A^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R}) \oplus (B^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R}))^{\perp} ⅋ \mathsf{R}$	$((A^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R}) \oplus (B^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R})) \otimes \mathsf{R}^{\perp}$
$A \& B$	$A^{\bullet}_{\mathsf{R}} \oplus B^{\bullet}_{\mathsf{R}}$	$A^{\bullet}_{\mathsf{R}} \oplus B^{\bullet}_{\mathsf{R}}$	$A^{\bullet\perp}_{\mathsf{R}} \& B^{\bullet\perp}_{\mathsf{R}}$
$!A$	$!(A^{\bullet}_{\mathsf{R}} \multimap \mathsf{R}) \multimap \mathsf{R}$	$(!(A^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R}))^{\perp} ⅋ \mathsf{R}$	$!(A^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R}) \otimes \mathsf{R}^{\perp}$
$?A$	$!((A^{\bullet}_{\mathsf{R}} \multimap \mathsf{R}) \multimap \mathsf{R})$	$!((A^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R})^{\perp} ⅋ \mathsf{R})$	$?((A^{\bullet\perp}_{\mathsf{R}} ⅋ \mathsf{R}) \otimes \mathsf{R}^{\perp})$

The sequent calculus for ILL (omitted for space reasons) works on the judgments of the form $\Delta \vdash^{i} I$. Let R be a fixed but arbitrary formula in ILL. We have the following derivable rules:

$$\frac{\Gamma, I \vdash^{i} \mathsf{R}}{\Gamma \vdash^{i} I \multimap \mathsf{R}}\ \mathsf{R}_R \qquad\qquad \frac{\Gamma \vdash^{i} I \qquad \mathsf{R} \vdash^{i} \mathsf{R}}{\Gamma, I \multimap \mathsf{R} \vdash^{i} \mathsf{R}}\ \mathsf{R}_L$$

By using Rule R_R, the formula I in the left-hand side of $\vdash^{i}$ becomes $I \multimap \mathsf{R}$ on the right-hand side. Similarly, by using Rule R_L, the formula I on the right-hand side of $\vdash^{i}$ becomes $I \multimap \mathsf{R}$ on the left-hand side. Moving the formula I from one side to the other of $\vdash^{i}$ results in $I \multimap \mathsf{R}$, which allows us to mimic in ILL the one-sided sequents of CLL. The translation in ILL of a CLL formula F, denoted $(F)^{\bullet}_{\mathsf{R}}$, is inductively defined using this movement of formulas; see Table 1 (second column).

The amount of (nested) occurrences of "$\multimap \mathsf{R}$" indicates how many times a formula has to be moved. Not all connectives require such transformations; we will expand on this in § 3. This translation extends to contexts as expected; it is correct, in the following sense:

▶ **Theorem 2.5** ([16]). *If $\vdash^{c} \Delta$ is provable in CLL then $\Delta^{\bullet}_{\mathsf{R}} \vdash^{i} \mathsf{R}$ is provable in ILL.*

Given an R such that $\bigotimes_{n} \mathsf{R} \vdash^{i} \mathsf{R}$ (for all $n > 0$), the theorem extends to CLL_{02} – CLL with the corresponding $\textsc{Mix}_0$ and $\textsc{Mix}_2$ rules (obtained from CP_{02} in Figure 2). The following result considers the case $\mathsf{R} = \mathbf{1}$; it will be useful in § 4, where we use CP_{02}.

▶ **Lemma 2.6** ([16]). *Let $\mathsf{R} = \mathbf{1}$. $\vdash^{c} \Delta$ is provable in CLL_{02} iff $\Delta^{\bullet}_{\mathsf{R}} \vdash^{i} \mathsf{R}$ is provable in ILL.*

As already mentioned, since we interpret propositions as sessions, we pick the simplest residual formula/protocol that satisfies the premise of Lemma 2.6, i.e., we fix $\mathsf{R} = \mathbf{1}$. Considering this, in the remainder of the paper we refer to the translation simply as $(-)^{\bullet}$.

3 A Denotational Characterization of Laurent's Translation

Here we study the effect of Laurent's translation $(-)^{\bullet}$ on processes typed under CP_0. We prove our first full abstraction result (Corollary 3.12) by lifting $(-)^{\bullet}$ to the level of denotations.

The Composed Translation. As discussed in § 1, we wish to compare processes in the uniform setting of CLL. We know that if $\Delta \vdash^{i} A$ is provable in ILL, then $\vdash^{c} \Delta^{\perp}, A$ is provable in CLL (see, e.g., [16]). Hence, we can interpret sequents in ILL as sequents in CLL.

Notice that formulas in ILL can be treated as formulas in CLL by letting $A \multimap B := A^\perp \parr B$. Using this transformation within the composition of $(-)^\bullet$ and $(-)^\perp$, we obtain the desired transformation on CLL proofs. From now on, we shall write $(-)^{\bullet\perp}$ to denote the translation given in Table 1 (rightmost column).

▶ **Theorem 3.1.** *If $\vdash^c \Delta$ is provable in CLL$_0$, then $\vdash^c \Delta^{\bullet\perp}, \mathbf{1}$ is provable in CLL$_0$.*

We shall write $A \in \mathsf{CP}_0$, when is clear from the context that A is a type. Similarly, $A \in \mathsf{CLL}_0$ says that A is a formula in CLL$_0$.

The Translation on Processes. $(-)^{\bullet\perp}$ induces a translation on processes, denoted $(-)^\bullet$, which is defined inductively on typing derivations (Definition 3.3). This translation is the computational interpretation of the composition of the two steps in Figure 1. Before detailing its definition, we examine two illustrative cases: output and input.

Let us first consider the process $P = x[y].(P_1 \mid P_2)$, which is typed as follows:

$$\dfrac{P_1 \vdash^c \Delta, y : A \qquad P_2 \vdash^c \Gamma, x : B}{x[y].(P_1 \mid P_2) \vdash^c \Delta, \Gamma, x : A \otimes B}\otimes$$

From the standpoint of the "propositions-as-sessions" interpretation, by Theorem 2.5 there exists a process $(P)^\bullet$ and fresh names z and w such that $\Delta^\bullet, z : (A \otimes B)^\bullet \vdash^i (P)^\bullet :: w : \mathbf{1}$. As Table 1 (second column) shows, $(A \otimes B)^\bullet = ((A^\bullet \multimap \mathbf{1}) \otimes (B^\bullet \multimap \mathbf{1})) \multimap \mathbf{1}$. To determine the shape of $(P)^\bullet$, we can reason inductively and apply Theorem 2.5 to the judgments $P_1 \vdash^c \Delta, y : A$ and $P_2 \vdash^c \Gamma, x : B$. This gives us $\Delta^\bullet, y : A^\bullet \vdash^i (P_1)^\bullet :: z_1 : \mathbf{1}$ and $\Gamma^\bullet, x : B^\bullet \vdash^i (P_2)^\bullet :: z_2 : \mathbf{1}$, respectively. We can then obtain the following typing derivation (and shape) for $(P)^\bullet$:

$$\dfrac{\dfrac{\dfrac{\Delta^\bullet, y : A^\bullet \vdash^i (P_1)^\bullet :: z_1 : \mathbf{1}}{\Delta^\bullet \vdash^i z_1(y).(P_1)^\bullet :: z_1 : A^\bullet \multimap \mathbf{1}} \qquad \dfrac{\Gamma^\bullet, x : B^\bullet \vdash^i (P_2)^\bullet :: z_2 : \mathbf{1}}{\Gamma^\bullet \vdash^i z_2(x).(P_2)^\bullet :: z_2 : B^\bullet \multimap \mathbf{1}}}{\Delta^\bullet, \Gamma^\bullet \vdash^i z_2[z_1].(z_1(y).(P_1)^\bullet \mid z_2(x).(P_2)^\bullet) :: z_2 : (A^\bullet \multimap \mathbf{1}) \otimes (B^\bullet \multimap \mathbf{1})} \quad (\star)}{\Delta^\bullet, \Gamma^\bullet, x' : (A \otimes B)^\bullet \vdash^i \underbrace{x'[z_2].(z_2[z_1].(z_1(y).(P_1)^\bullet \mid z_2(x).(P_2)^\bullet) \mid [x' \leftrightarrow w])}_{(P)^\bullet} :: w : \mathbf{1}}$$

where $(\star)$ stands for $x' : \mathbf{1} \vdash^i [x' \leftrightarrow w] :: w : \mathbf{1}$. Above, we see how each nested "$\multimap \mathbf{1}$" induced by Laurent's translation entails extra actions on the level of processes, due to the interpretation of $\multimap$ as input (when introduced on the right, as in this case): moving $y : A^\bullet$ and $x : B^\bullet$ to the right-hand side induces the inputs along z_1 and z_2, respectively. Subsequently, we use the $\otimes$ rule on the right, which produces the output on z_2; we then move the resulting assignment for z_2 back to the left, finally obtaining $x' : (A \otimes B)^\bullet$. This last movement adds the final "$\multimap \mathbf{1}$": because it is introduced on the left, we obtain an output along x'. At this point, we can return to the classical setting by applying the translation $(-)^\perp$ to the derivation above, which leads to the following typing derivation for $(P)^\bullet$ in CP$_0$:

$$\dfrac{\dfrac{\dfrac{(P_1)^\bullet \vdash^c \Delta^{\bullet\perp}, y : A^{\bullet\perp}, z_1 : \mathbf{1}}{z_1(y).(P_1)^\bullet \vdash^c \Delta^{\bullet\perp}, z_1 : A^{\bullet\perp} \parr \mathbf{1}} \qquad \dfrac{(P_2)^\bullet \vdash^c \Gamma^{\bullet\perp}, x : B^{\bullet\perp}, z_2 : \mathbf{1}}{z_2(x).(P_2)^\bullet \vdash^c \Gamma^{\bullet\perp}, z_2 : B^{\bullet\perp} \parr \mathbf{1}}}{z_2[z_1].(z_1(y).(P_1)^\bullet \mid z_2(x).(P_2)^\bullet) \vdash^c \Delta^{\bullet\perp}, \Gamma^{\bullet\perp}, z_2 : (A^{\bullet\perp} \parr \mathbf{1}) \otimes (B^{\bullet\perp} \parr \mathbf{1})} \quad (\star\star)}{\underbrace{x'[z_2].(z_2[z_1].(z_1(y).(P_1)^\bullet \mid z_2(x).(P_2)^\bullet) \mid [x' \leftrightarrow w])}_{(P)^\bullet} \vdash^c \Delta^{\bullet\perp}, \Gamma^{\bullet\perp}, x' : (A \otimes B)^{\bullet\perp}, w : \mathbf{1}}$$

where $(\star\star)$ stands for $[x' \leftrightarrow w] \vdash^c x' : \perp, w : \mathbf{1}$. Importantly, while the translation $(-)^\perp$ modifies the types for $(P)^\bullet$, it does not change its shape. Also, it is worth noticing how the output on x in P is mimicked by $(P)^\bullet$ through the output on z_2, not by the output on x'.

Now consider the case when $Q = x(y).Q_1$, which is typed as:

$$\frac{Q_1 \vdash^c \Delta, y : A, x : B}{x(y).Q_1 \vdash^c \Delta, x : A \parr B}$$

In this case, we expect to obtain a process $(Q)^\bullet$ such that $\Delta^\bullet, x' : (A \parr B)^\bullet \vdash^i (Q)^\bullet :: w : \mathbf{1}$, where $(A \parr B)^\bullet = A^\bullet \otimes B^\bullet$ (cf. Table 1, second column). By reasoning inductively on $Q_1 \vdash^c \Delta, y : A, x : B$, we obtain $\Delta^\bullet, y : A^\bullet, x' : B^\bullet \vdash^i (Q_1)^\bullet :: w : \mathbf{1}$, which enables us to obtain the following derivation:

$$\frac{\Delta^\bullet, y : A^\bullet, x' : B^\bullet \vdash^i (Q_1)^\bullet :: w : \mathbf{1}}{\Delta^\bullet, x' : A^\bullet \otimes B^\bullet \vdash^i \underbrace{x'(y).(Q_1)^\bullet}_{(Q)^\bullet} :: w : \mathbf{1}}$$

Differently from the case of $\otimes$, here the transformation $(-)^\bullet$ does not add any "$\multimap \mathbf{1}$". This is relevant, because it ensures that the process $(Q)^\bullet$ does not have input/output actions in front of the input on x', which mimics the input on x in Q. By applying the translation $(-)^\perp$ to the derivation above, we obtain the following typing derivation for $(Q)^\bullet$ in CP_0:

$$\frac{(Q_1)^\bullet \vdash^c \Delta^{\bullet\perp}, y : A^{\bullet\perp}, x' : B^{\bullet\perp}, w : \mathbf{1}}{\underbrace{x'(y).(Q_1)^\bullet}_{(Q)^\bullet} \vdash^c \Delta^{\bullet\perp}, x' : (A \parr B)^{\bullet\perp}, w : \mathbf{1}}$$

Once again, notice that the translation $(-)^\perp$ does not modify the shape of $(Q)^\bullet$.

A key observation is that although $P = x[y].(P_1 \mid P_2)$ and $Q = x(y).Q_1$ are compatible (i.e., they have complementary actions on x), their translations $(P)^\bullet$ and $(Q)^\bullet$ are not. In general, given two composable processes $P \vdash^c \Delta, x : A$ and $Q \vdash^c \Gamma, x : A^\perp$, we have:

$$(P)^\bullet \vdash^c \Delta^{\bullet\perp}, x' : A^{\bullet\perp}, w : \mathbf{1} \qquad (Q)^\bullet \vdash^c \Gamma^{\bullet\perp}, x' : A^{\perp^{\bullet\perp}}, z : \mathbf{1}$$

and so $(P)^\bullet$ and $(Q)^\bullet$ cannot be composed directly: the types of x' are not dual ($(A^{\bullet\perp})^\perp \neq A^{\perp^{\bullet\perp}}$). To circumvent this difficulty, we shall consider *synchronizer processes* $\mathcal{S}^A_{z,w}$ such that

$$\mathcal{S}^A_{z,w} \vdash^c w : A^\bullet \otimes \perp, z : A^{\perp^\bullet} \otimes \perp, s : \mathbf{1}$$

Using synchronizers, a mediated composition between $(P)^\bullet$ and $(Q)^\bullet$ is then possible:

$$(\nu w)((\nu z)(z(y).(Q)^\bullet \mid \mathcal{S}^A_{z,w}) \mid w(x').(P)^\bullet)$$

Synchronizer processes have a purely logical origin: Laurent [16] shows that for any A the sequent $\vdash^c A^\bullet \otimes \perp, A^{\perp^\bullet} \otimes \perp, \mathbf{1}$ is provable; using this result, the definition of synchronizers (given next) arises by reading off the process associated with this proof.

▶ **Definition 3.2** (Synchronizer). *Given $F \in CP_0$ and names z, w, and s, we define the synchronizer process $\mathcal{S}^A_{z,w}$, satisfying $\mathcal{S}^A_{z,w} \vdash^c z : A^\bullet \otimes \perp, w : A^{\perp^\bullet} \otimes \perp, s : \mathbf{1}$, by recursion on A.*

Armed with the notion of synchronizer processes, we can finally define:

▶ **Definition 3.3** (Laurent's translation on processes). *Let $\mathcal{S}^A_{z,w}$ be a synchronizer as in Definition 3.2. Given a typed process $P \vdash^c \Delta$, we define $(P)^\bullet$ inductively in Figure 6.*

The next lemma ensures that for a given CP_0 process P, $(P)^\bullet$ is well-typed.

$$
\begin{aligned}
([x \leftrightarrow y])^{\bullet} &= x'[x].([x \leftrightarrow y] \mid [x' \leftrightarrow w]) \\
(x(\,).P)^{\bullet} &= x'(\,).(P)^{\bullet} \\
(x[\,])^{\bullet} &= x'[x].(x[\,] \mid [x' \leftrightarrow w]) \\
(x(y).P)^{\bullet} &= x'(y).(P)^{\bullet} \\
(x[y].(P \mid Q))^{\bullet} &= x'[z_2].(z_2[z_1].(z_1(y).(P)^{\bullet} \mid z_2(x).(Q)^{\bullet}) \mid [x' \leftrightarrow w]) \\
(x.case(P_1, P_2))^{\bullet} &= x'.case((P_1)^{\bullet}, (P_2)^{\bullet}) \\
(x[i].P)^{\bullet} &= x'[z].(z[i].z(y).(P)^{\bullet} \mid [x' \leftrightarrow w]) \\
(!x(y).P)^{\bullet} &= x'[x].(!x(v).v(y).(P)^{\bullet} \mid [x' \leftrightarrow w]) \\
(?x[y].P)^{\bullet} &= ?x'[m].m[v].(v(y).(P)^{\bullet} \mid [m \leftrightarrow w]) \\
((\nu x)(P \mid Q))^{\bullet} &= (\nu w)((\nu z)(z(x').(Q)^{\bullet} \mid \mathcal{S}^{A}_{z,w}) \mid w(x').(P)^{\bullet})
\end{aligned}
$$

Figure 6 Laurent's translation on CP processes (Definition 3.3).

▶ **Lemma 3.4.** *Let $P \vdash^c \Delta$ be a process in CP$_0$, then $(P)^{\bullet} \vdash^c \Delta^{\bullet\perp}, w : \mathbf{1}$.*

▶ **Example 3.5.** To illustrate mediated composition, consider the processes $x[\,] \vdash^c x : \mathbf{1}$ and $x(\,).P \vdash^c \Delta, x : \perp$. By Lemma 3.4, we have $x'[x].(x[\,] \mid [x' \leftrightarrow w]) \vdash^c x' : \mathbf{1}^{\bullet\perp}, w : \mathbf{1}$ and $z(m).m(\,).(P)^{\bullet} \vdash^c \Delta^{\bullet\perp}, m : \perp, z : \mathbf{1}$, respectively. These two processes can be composed with the synchronizer for $A = \mathbf{1}$:

$$\mathcal{S}^{\mathbf{1}}_{w,z} = w[x'].(x'(x).z[m_1].([m_1 \leftrightarrow x] \mid [x' \leftrightarrow z]) \mid [w \leftrightarrow s])$$

By expanding Definition 3.3, we obtain the following observational equivalence:

$$
\begin{aligned}
((\nu x)(x[\,] \mid x(\,).P))^{\bullet} &= (\nu z)((\nu w)(w(x').(x[\,])^{\bullet} \mid \mathcal{S}^{\mathbf{1}}_{z,w}) \mid z(m).(x(\,).P)^{\bullet}) \\
&= (\nu z)((\nu w)(w(x').x'[x].(x[\,] \mid [x' \leftrightarrow w]) \\
&\qquad \mid w[x'].(x'(x).z[m].([m \leftrightarrow x] \mid [x' \leftrightarrow z]) \mid [w \leftrightarrow s])) \\
&\qquad \mid z(m).m(\,).(P)^{\bullet}) \\
&\simeq (P)^{\bullet}\{s/z\}
\end{aligned}
$$

Properties. The logical translations strongly suggest that P and $(P)^{\bullet}$ should be equivalent in some sense. How to state this relation? Our technical insight is to bring $(-)^{\bullet\perp}$ to the level of denotations: we define the function $\mathbb{L}_A(-) : [\![A]\!] \to [\![A^{\bullet\perp}]\!]$, which "saturates" $[\![A]\!]$ by adding as many "$*$" (the observation of $\mathbf{1}$ and $\perp$) as residual $\perp$s and $\mathbf{1}$s are induced by $(-)^{\bullet\perp}$.

▶ **Definition 3.6** (Transformations on Denotations). *Given $A \in$ CP$_0$, $\mathbb{L}_A(-) : [\![A]\!] \mapsto [\![A^{\bullet\perp}]\!]$ is defined in Figure 7. Given $\Delta = x_1 : A_1, \dots, x_n : A_n$, we define $\mathbb{L}^*_{\Delta}(-) : [\![\Delta]\!] \mapsto [\![\Delta^{\bullet\perp}, 1]\!]$ as*

$$\mathbb{L}^*_{\Delta}((a_1, \dots, a_n)) = (\mathbb{L}_{A_1}(a_1), \dots, \mathbb{L}_{A_n}(a_n), *)$$

Our goal is to show that $\mathbb{L}^*_{\Delta}([\![P \vdash^c \Delta]\!]) = [\![(P)^{\bullet} \vdash^c \Delta^{\bullet\perp}, w : \mathbf{1}]\!]$ (Theorem 3.11). In particular we use synchronizers (and their observations) to prove the property for processes with cut. Also, the following two properties will be useful:

▶ **Lemma 3.7.** *For any A and Δ in CP$_0$, both $\mathbb{L}_A(-)$ and $\mathbb{L}^*_{\Delta}(-)$ are injective.*

$$\begin{aligned}
\mathbb{L}_{\perp}(*) &= * & \mathbb{L}_{!A}(\lbag a_1, a_2 \rbag) &= (\lbag(\mathbb{L}_A(a_1), *), (\mathbb{L}_A(a_2), *)\rbag, *) \\
\mathbb{L}_{\mathbf{1}}(*) &= (*, *) & \mathbb{L}_{?A}(\lbag a_1, a_2 \rbag) &= \lbag((\mathbb{L}_A(a_1), *), *), ((\mathbb{L}_A(a_2), *), *)\rbag \\
\mathbb{L}_{A \parr B}((a, b)) &= (\mathbb{L}_A(a), \mathbb{L}_B(b)) & \mathbb{L}_{A \otimes B}((a, b)) &= ((\mathbb{L}_A(a), *), (\mathbb{L}_B(b), *), *) \\
\mathbb{L}_{A_1 \& A_2}((i, a)) &= (i, \mathbb{L}_{A_i}(a)) & \mathbb{L}_{A_1 \oplus A_2}((i, a)) &= ((i, (\mathbb{L}_{A_i}(a), *)), *)
\end{aligned}$$

Figure 7 Transformation on denotations induced by Laurent's translation (Definition 3.6). The generalized definitions $\mathbb{L}_{!A}(\lbag a_1, \ldots, a_n \rbag)$ and $\mathbb{L}_{?A}(\lbag a_1, \ldots, a_n \rbag)$ arise as expected.

▶ **Lemma 3.8.** *Let* $?A \in \mathit{CP}_0$. *Suppose* $\alpha_j \in \llbracket ?A \rrbracket$ *for all* $j = 1, \ldots, k$. *Then* $\mathbb{L}_{?A}(\uplus_{j=1}^{k} \alpha_j) = \uplus_{j=1}^{k} \mathbb{L}_{?A}(\alpha_j)$.

Proof sketch. It follows by Definition 3.6 and definition of $\uplus$. ◀

As explained above, synchronizers mediate between the translation of two processes. The following lemma ensures that a synchronizer $\mathcal{S}^{A}_{w,z}$ acts as a forwarder:

▶ **Lemma 3.9.** *Let* $\Delta = z : A^{\bullet} \otimes \bot, w : A^{\perp\bullet} \otimes \bot, s : \mathbf{1}$, *for some* $A \in \mathit{CP}_0$. *Then* $\llbracket \mathcal{S}^{A}_{z,w} \vdash^{c} \Delta \rrbracket = \{((\mathbb{L}_A(a), *), (\mathbb{L}_{A^\perp}(a), *), *) \mid a \in \llbracket A \rrbracket\}$.

Proof sketch. The proof follows by induction on A. ◀

The next lemma is crucial to ensure that the denotations of a composed process correspond (in the sense of Definition 3.6) with those of its translation:

▶ **Lemma 3.10** (Synchronizers are well-behaved)**.** *Let* $P, P', Q, Q' \in \mathit{CP}_{02}$, *such that*

$$\begin{aligned}
\llbracket P' \vdash^{c} \Delta', x' : A^{\bullet\perp}, w : \mathbf{1} \rrbracket &= \mathbb{L}^{*}_{\Delta, x:A}(\llbracket P \vdash^{c} \Delta, x : A \rrbracket) \\
\llbracket Q' \vdash^{c} \Gamma', x' : A^{\perp\bullet\perp}, z : \mathbf{1} \rrbracket &= \mathbb{L}^{*}_{\Gamma, y:A^\perp}(\llbracket Q \vdash^{c} \Gamma, x : A^{\perp} \rrbracket)
\end{aligned}$$

Then:

$$(\delta, \gamma) \in \llbracket (\nu x)(P \mid Q) \rrbracket \;\Leftrightarrow\; (\mathbb{L}_\Delta(\delta), \mathbb{L}_\Gamma(\gamma), *) \in \llbracket (\nu w)(w(x').P' \mid (\nu z)(z(x').Q' \mid \mathcal{S}^{A}_{z,w})) \rrbracket.$$

Proof.

$$\begin{aligned}
&(\delta, \gamma) \in \llbracket (\nu x)(P \mid Q) \rrbracket \\
&\Leftrightarrow\; (\delta, a) \in \llbracket P \rrbracket \wedge (\gamma, a) \in \llbracket Q \rrbracket && \text{(by Figure 5)} \\
&\Leftrightarrow\; (\mathbb{L}_\Delta(\delta), \mathbb{L}_A(a), *) \in \llbracket P' \rrbracket \wedge (\mathbb{L}_\Gamma(\gamma), \mathbb{L}_{A^\perp}(a), *) \in \llbracket Q' \rrbracket && \text{(by assumption)} \\
&\Leftrightarrow\; (\mathbb{L}_\Delta(\delta), (\mathbb{L}_A(a), *)) \in \llbracket w(x').P' \rrbracket \wedge (\mathbb{L}_\Gamma(\gamma), (\mathbb{L}_{A^\perp}(a), *)) \in \llbracket z(x').Q' \rrbracket && \text{(by Figure 5)} \\
&\Leftrightarrow\; (\mathbb{L}_\Delta(\delta), (\mathbb{L}_{A^\perp}(a), *), *) \in \llbracket (\nu w)(w(x').P' \mid \mathcal{S}^{A}_{z,w}) \rrbracket \\
&\qquad\qquad \wedge (\mathbb{L}_\Gamma(\gamma), (\mathbb{L}_{A^\perp}(a), *)) \in \llbracket z(x').Q' \rrbracket && \text{(by Lemma 3.9)} \\
&\Leftrightarrow\; (\mathbb{L}_\Delta(\delta), \mathbb{L}_\Gamma(\gamma), *) \in \llbracket (\nu w)(w(x').P' \mid (\nu z)(z(x').Q' \mid \mathcal{S}^{A}_{z,w})) \rrbracket && \text{(by Figure 5)}
\end{aligned}$$

◀

The next result, Theorem 3.11, states that the lifting of Laurent's transformation $(-)^{\bullet\perp}$ to the level of denotations is correct.

▶ **Theorem 3.11.** *Let* $P \vdash^{c} \Gamma$ *be a* CP_0 *process. Then* $\mathbb{L}^{*}_{\Gamma}(\llbracket P \vdash^{c} \Gamma \rrbracket) = \llbracket (P)^{\bullet} \vdash^{c} \Gamma^{\bullet\perp}, w : \mathbf{1} \rrbracket$.

$$\frac{C \Downarrow \theta \qquad C' \Downarrow \gamma}{C \mid C' \Downarrow (\sigma, \gamma)}\ \textsc{ObsMix} \qquad \frac{C_1 \vdash_{\mathsf{cfg}} \Gamma_1 \mid \Sigma_1 \qquad C_2 \vdash_{\mathsf{cfg}} \Gamma_2 \mid \Sigma_2}{C_1 \mid C_2 \vdash_{\mathsf{cfg}} \Gamma_1, \Gamma_2 \mid \Sigma_1, \Sigma_2}\ \textsc{cfgMix}$$

$$[\![P \mid Q \vdash^c \Gamma, \Delta]\!] = \{(\gamma, \delta) \mid \gamma \in [\![P \vdash^c \Gamma]\!], \delta \in [\![Q \vdash^c \Delta]\!]\}$$

Figure 8 Extensions concerning $\textsc{Mix}_2$.

Proof sketch. By induction on the structure of $P \vdash^c \Gamma$, with a case analysis in the last rule applied. We give a representative case. Consider $!x(y).P \vdash^c ?\Delta, x : !A$, with $\Delta = x_1 : A_1, \ldots, x_n : A_n$. In one direction, we apply Lemma 3.8 and Definition 3.6 to show

$$\begin{aligned}
&\mathbb{L}^*_{?\Delta,x:!A}([\![!x(y).P \vdash^c ?\Delta, x : !A]\!]) \\
&= \{\mathbb{L}^*_{?\Delta,x:!(A)}(\uplus_{j=1}^k \alpha_j^1, \cdots, \uplus_{j=1}^k \alpha_j^n, \wr a_1, , \cdots, a_k \wr) \mid \\
&\qquad \forall i \in \{1, \cdots, k\}.(\alpha_i^1, \cdots, \alpha_i^n, a_i) \in [\![P \vdash^c ?\Delta, y : A]\!]\} \\
&= \{(\uplus_{j=1}^k \beta_j^1, \cdots, \uplus_{j=1}^k \beta_j^n, (\wr(b_1, *), \cdots, (b_k, *)\wr, *), *) \mid && \text{with } \mathbb{L}_{A_i}(\alpha^i) = \beta^i, \\
&\qquad \forall i \in \{1, \cdots, k\}.(\alpha_i^1, \cdots, \alpha_i^n, a_i, *) \in [\![P \vdash^c ?\Delta, y : A]\!]\} && \text{(and } \mathbb{L}_A(a) = b)
\end{aligned}$$

In the other direction, by the I.H., we obtain:

$$\begin{aligned}
&[\![(!x(y).P)^\bullet \vdash^c (?\Delta)^{\bullet\perp}, m : (!A)^{\bullet\perp}, w : \mathbf{1}]\!] \\
&\quad = \{(\uplus_{j=1}^k \beta_j^1, \cdots, \uplus_{j=1}^k \beta_j^n, (\wr(b_1, *), \cdots, (b_k, *)\wr, *), *) \mid \\
&\qquad \forall i \in \{1, \cdots, k\}.(\alpha_i^1, \cdots, \alpha_i^n, a_i, *) \in [\![P \vdash^c ?\Delta, y : A]\!]\}
\end{aligned}$$

when the last rule is cut, we rely on I.H. and Lemma 3.10. ◀

By combining Theorems 2.2 and 3.11 and Lemma 3.7, we obtain our first full abstraction result:

▶ **Corollary 3.12** (Full Abstraction (I)). *Suppose $P, Q \vdash^c \Delta$. Then $P \simeq Q$ iff $(P)^\bullet \simeq (Q)^\bullet$.*

Proof. We have the following equivalences:

$$\begin{aligned}
P \simeq Q \quad &\Leftrightarrow \quad [\![P \vdash^c \Delta]\!] = [\![Q \vdash^c \Delta]\!] && \text{(by Theorem 2.2)} \\
&\Leftrightarrow \quad \mathbb{L}^*_\Delta([\![P \vdash^c \Delta]\!]) = \mathbb{L}^*_\Delta([\![Q \vdash^c \Delta]\!]) && \text{(by Lemma 3.7)} \\
&\Leftrightarrow \quad [\![(P)^\bullet \vdash^c \Delta^{\bullet\perp}, w : \mathbf{1}]\!] = [\![(Q)^\bullet \vdash^c \Delta^{\bullet\perp}, w : \mathbf{1}]\!] && \text{(by Theorem 3.11)} \\
&\Leftrightarrow \quad (P)^\bullet \simeq (Q)^\bullet && \text{(by Theorem 2.2)}
\end{aligned}$$

◀

4 An Operational Characterization of Laurent's Translation

In this section, we show that the translation $(-)^\bullet$ can be *internalized* as an evaluation context. That is, given $P \vdash^c \Delta$, we can define a corresponding *transformer context*, denoted $\widehat{\top}_\Delta[-]$. Using this context, we obtain a process denoted $\widehat{\top}_\Delta[P]$, in which the behavior of P is adapted following Δ, so as to produce $\Delta^{\bullet\perp}, w : \mathbf{1}$. This is clearly different from translating P into $(P)^\bullet$ by examining its structure. We shall show that $\widehat{\top}_\Delta[P]$ is equivalent to $(P)^\bullet$ (Corollary 4.10). As in § 3, we will also show a full-abstraction result for $\widehat{\top}_\Delta[-]$ (Corollary 4.15).

$$\begin{aligned}
\mathrm{T}^{\perp}_{x,x'} &= [x \leftrightarrow y] \vdash^{c} x : \mathbf{1}, x' : \perp \\
\mathrm{T}^{\mathbf{1}}_{x,x'} &= x'[y].([y \leftrightarrow x] \mid x'(\,).\mathbf{0}) \vdash^{c} x : \perp, x' : \mathbf{1}\otimes\perp \\
\mathrm{T}^{A\otimes B}_{x,z} &= x(y).z[z_2].(z_1[z_2].(z_1(y).(\mathrm{T}^{A}_{y,y'} \mid z_1[\,]) \mid z_2(x').(\mathrm{T}^{B}_{x,x'} \mid z_2[\,]) \mid z(\,).\mathbf{0} \vdash^{c} \Delta \\
&\qquad (\text{with } \Delta = x : A^{\perp}\parr B^{\perp}, z : (A\otimes B)^{\bullet\perp}) \\
\mathrm{T}^{A\parr B}_{x,x'} &= x'(y').x[y].(\mathrm{T}^{A}_{y,y'} \mid \mathrm{T}^{B}_{x,x'}) \vdash^{c} x : A^{\perp}\otimes B^{\perp}, x' : A\parr B^{\bullet\perp} \\
\mathrm{T}^{!A}_{x,x'} &= x'[w'].(!w'(w).?x[y].w(y').(\mathrm{T}^{A}_{y,y'} \mid w[\,]) \mid x'(\,).\mathbf{0}) \vdash^{c} x : ?(A^{\perp}), x' : (!A)^{\bullet\perp} \\
\mathrm{T}^{?A}_{x,x'} &= !x(y).?x'[m].m[z].(z(y').(\mathrm{T}^{A}_{y,y'} \mid z[\,]) \mid m(\,).\mathbf{0}) \vdash^{c} x : !(A^{\perp}), x' : (?A)^{\bullet\perp} \\
\mathrm{T}^{A_1 \& A_2}_{x,x'} &= x'.\mathit{case}(x[1].\mathrm{T}^{A_1}_{x,x'}, x[2].\mathrm{T}^{A_2}_{x,x'}) \vdash^{c} x : A_1^{\perp}\oplus A_2^{\perp}, x' : (A_1 \& A_2)^{\bullet\perp} \\
\mathrm{T}^{A_1 \oplus A_2}_{x,x'} &= y.\mathit{case}(P_1, P_2) \vdash^{c} x : A_1^{\perp}\& A_2^{\perp}, x' : (A_1\oplus A_2)^{\bullet\perp} \\
&\qquad (\text{with } P_i = m[w].(w[i].w(y').(\mathrm{T}^{A_i}_{x,x'} \mid w[\,]) \mid m(\,).\mathbf{0}))
\end{aligned}$$

Figure 9 Transformer processes (Definition 4.3).

This strategy works in presence of Rule Mix$_2$ (cf. Figure 2). Hence, in this section we work with typed processes in CP$_{02}$. Accordingly, we extend the denotational semantics (cf. § 2) as given in Figure 8. It is easy to check that soundness and completeness (Theorem 2.2 and Corollary 2.3) still hold for CP$_{02}$. In CP$_{02}$, additional observational equivalences arise from permutation of Rule Mix$_2$ with other rules:

▶ **Lemma 4.1.** *Given $P, Q, R \in CP_{02}$, we have: $x.case(P, Q) \mid R \simeq x.case(P \mid R, Q \mid R)$, $(\nu x)(P \mid Q) \mid R \simeq (\nu x)(P \mid (Q \mid R)) \simeq (\nu x)((P \mid R) \mid Q)$ and $x[y].(P \mid (Q \mid R)) \simeq x[y].(P \mid Q) \mid R$.*

We also need to extend $(-)^{\bullet}$ (Definition 3.3). Given processes $P \vdash^{c} \Delta$ and $Q \vdash^{c} \Gamma$ (with their translations $(P)^{\bullet} \vdash^{c} \Delta^{\bullet\perp}, y : \mathbf{1}$ and $(Q)^{\bullet} \vdash^{c} \Gamma^{\bullet\perp}, x : \mathbf{1}$, respectively), we define:

$$(P \mid Q)^{\bullet} = (\nu x)(x[y].((P)^{\bullet} \mid (Q)^{\bullet}) \mid M_x)$$

where $M_x = x(y).y(\,).[x \leftrightarrow m]$. It is easy to check that $M_x \vdash^{c} x : \perp\parr\perp, m : \mathbf{1}$.

▶ Remark 4.2. The results about $\mathbb{L}^{*}(-)$ in § 3 can be adapted to CP$_{02}$.

Transformers. We define *transformer processes*, which adapt the behavior of one session on a given name.

▶ **Definition 4.3** (Transformers). *Given a type A in CP_{02}, we define the* transformer process *$\mathrm{T}^{A}_{x,x'} \vdash^{c} x : A^{\perp}, x' : A^{\bullet\perp}$ by induction on the type A as in Figure 9. With a slight abuse of notation, in the figure we write $\mathrm{T}^{A}_{x,x'} = P \vdash^{c} \Delta$ to express that $\mathrm{T}^{A}_{x,x'} = P$ with $P \vdash^{c} \Delta$.*

We now define *transformer contexts*, which adapt an entire context Δ using transformer processes. We first define *typed contexts.*

Typed Process Contexts. A typed context is a typed process with a typed hole. We write $K[\square] \vdash^{c}_{k} \Delta \parallel \Gamma$ for a typed context which contains a typed hole $\square$, and which produces a process of type Γ. That is, given a process $P \vdash^{c} \Delta$, we can fill $K[\square]$ as $K[P] \vdash^{c} \Gamma$, by

$$\dfrac{K[\square] \vdash^{\mathsf{c}}_{k} \Sigma \parallel \Gamma, x : A \qquad Q \vdash^{c} \Delta, x : A^{\perp}}{(\nu x)(K[\square] \mid Q) \vdash^{\mathsf{c}}_{k} \Sigma \parallel \Gamma, \Delta}\ \textsc{KCut}_1$$

$$\dfrac{K[\square] \vdash^{\mathsf{c}}_{k} \Sigma \parallel \Gamma \qquad P \vdash^{c} \Delta}{K[\square] \mid P \vdash^{\mathsf{c}}_{k} \Sigma \parallel \Gamma, \Delta}\ \textsc{KMix} \qquad\qquad \dfrac{}{\square \vdash^{\mathsf{c}}_{k} \Delta \parallel \Delta}\ \textsc{KHole}$$

Figure 10 Typed Contexts.

replacing the unique occurrence of $\square$ with P. Figure 10 gives the rules for forming typed contexts. As an example, consider the derivation for a parallel context $(\nu x)(\square \mid \mathrm{T}^{A}_{x,y})$:

$$\dfrac{\dfrac{}{\square \vdash^{\mathsf{c}}_{k} x : A \parallel x : A} \qquad \mathrm{T}^{A}_{x,y} \vdash^{c} x : A^{\perp}, y : A^{\bullet\perp}}{(\nu x)(\square \mid \mathrm{T}^{A}_{x,y}) \vdash^{\mathsf{c}}_{k} x : A \parallel y : A^{\bullet\perp}}$$

Above, we can replace the use of Rule KHole with a typing derivation for $P \vdash^{c} x : A$, thus obtaining $(\nu x)(P \mid \mathrm{T}^{A}_{x,y}) \vdash^{c} y : A^{\bullet\perp}$. Such "filling" of contexts can be done in general:

▶ **Lemma 4.4.** *Given $K[\square] \vdash^{\mathsf{c}}_{k} \Delta \parallel \Gamma$ and $P \vdash^{c} \Delta$, we have that $K[P] \vdash^{c} \Gamma$ is derivable.*

Transformer contexts. As we have seen, contexts in our setting are hardly arbitrary: only type-compatible processes are inserted into holes. Based on this observation, and following the typing rules, we define *transformer contexts* and *transformer contexts with closing name*:

▶ **Definition 4.5.** *Let $\Delta = x_1 : A_1, \ldots, x_n : A_n$ be a typing context. We define:*

- *transformer contexts:* $\mathrm{T}_{\Delta}[\square] = (\nu x_n)(\cdots(\nu x_1)(\square \mid \mathrm{T}^{A_1}_{x_1,y_1}) \mid \cdots) \mid \mathrm{T}^{A_n}_{x_n,y_n})$
- *transformer contexts with closing name (z is fresh wrt Δ):* $\widehat{\mathsf{T}}_{\Delta}[\square] = \mathrm{T}_{\Delta}[\square] \mid z[\,]$.

Note that by Remark 2.4 the order of the cuts in $\mathrm{T}_{\Delta}[-]$ does not matter.

We will show that transformers are correct: the transformed process $\widehat{\mathsf{T}}_{\Delta}[P]$ is equivalent to the translated process $(P)^{\bullet}$ (Lemma 4.9). We need auxiliary results about transformers.

▶ **Lemma 4.6.** *The following observational equivalences hold:*

- $x'(y').\widehat{\mathsf{T}}_{\Delta,y:A,x:B}[P] \simeq \widehat{\mathsf{T}}_{\Delta,x:A\parr B}[x(y).P]$
- $x'[w'].(!w'(w).w(y').\widehat{\mathsf{T}}_{?(\Delta),y:A}[P] \mid x'(\,).\mathbf{0}) \mid n[\,] \simeq \widehat{\mathsf{T}}_{\Delta,x:!A}[!x(y).P]$
- $?x[m].m[z].(z(y).\widehat{\mathsf{T}}_{\Delta,y:A}[P] \mid m(\,).\mathbf{0}) \simeq \widehat{\mathsf{T}}_{\Delta,x:?A}[?x[y].P]$
- $m[w].(w[i].w(y).\widehat{\mathsf{T}}_{\Delta,y:A}[P] \mid m(\,).\mathbf{0}) \mid n[\,] \simeq \widehat{\mathsf{T}}_{y:A_1\oplus A_2}[y[i].P]$
- $y'.\mathit{case}(\widehat{\mathsf{T}}_{\Delta,y:A_1}[P_1], \widehat{\mathsf{T}}_{\Delta,y:A_2}[P_2]) \simeq \widehat{\mathsf{T}}_{\Delta y:A_1\& A_2}[y.\mathit{case}(P_1, P_2)]$
- $z[z_2].(z_2[z_1].(z_1(y').\widehat{\mathsf{T}}_{\Delta,y:A}[P_1] \mid z_2(x').\widehat{\mathsf{T}}_{\Gamma,x:B}[P_2]) \mid z(\,).\mathbf{0}) \mid w[\,] \simeq \widehat{\mathsf{T}}_{\Delta,\Gamma,x:A\otimes B}[x[y].(P_1 \mid P_2)]$

Proof sketch. The proof follows from Remark 2.4 and Lemma 4.1. ◀

▶ **Lemma 4.7.** $\llbracket [x \leftrightarrow y] \vdash^{c} x : \mathbf{1}, y : \perp \rrbracket = \llbracket x[\,] \mid y(\,).\mathbf{0} \vdash^{c} x : \mathbf{1}, y : \perp \rrbracket$

▶ **Lemma 4.8.** *Let $\Delta = x_1 : !A^{\perp}, x_1' : ?((A \parr \mathbf{1}) \otimes \perp)$. We have:*

$$\llbracket \mathrm{T}^{?A}_{x_1,x_1'} \vdash^{c} \Delta \rrbracket = \left\{ \begin{array}{l} \quad (\Lbag a_1, \ldots, a_k \Rbag, \biguplus_{j=1}^{k} \Lbag ((a', *), *) \Rbag) \\ \mid \forall i \in \{1, \ldots, k\}.(a_i, a_i') \in \llbracket \mathrm{T}^{A}_{y,y'} \vdash^{c} y : A^{\perp}, y' : A^{\bullet\perp} \rrbracket \end{array} \right\}$$

We may now establish the correctness of transformers:

▶ **Lemma 4.9.** *Suppose* $P \vdash^c \Gamma$*. Then* $[\![(P)^\bullet \vdash^c \Gamma^{\bullet\perp}, w : \mathbf{1}]\!] = [\![\widehat{\mathsf{T}}_\Gamma[P] \vdash^c \Gamma^{\bullet\perp}, w : \mathbf{1}]\!]$.

Proof sketch. By induction on the structure of P. We consider a number of illustrative cases. If $P = !x(y).P'$, then:

$$\begin{aligned}(!x(y).P')^\bullet &= m[x].(!x(w).w(y).(P')^\bullet \mid [m \leftrightarrow n]) \\ &\simeq m[x].(!x(w).w(y).\widehat{\mathsf{T}}_{\Delta,y:A}[P'] \mid m(\,).\mathbf{0}) \mid n[\,] && \text{(by I.H. and Lemma 4.7)} \\ &\simeq \widehat{\mathsf{T}}_{\Delta,x:!(A)}[P'] && \text{(by Lemma 4.6)}\end{aligned}$$

If $P = (\nu x)(R \mid Q)$, then we need to show:

$$(P)^\bullet \simeq (\nu z)(z(y).\widehat{\mathsf{T}}_{\Gamma,x:A^\perp}[Q] \mid (\nu w)(w(x').\widehat{\mathsf{T}}_{\Delta,x:A}[R] \mid \mathcal{S}^A_{z,w})) \simeq \widehat{\mathsf{T}}_{\Gamma,\Delta}[(\nu x)(R \mid Q)]$$

By I.H. and Theorem 3.11 we know that:

$$\begin{aligned}&(\mathbb{L}_\Delta(\delta), \mathbb{L}_\Gamma(\gamma), *) \in [\![(\nu z)(z(y).\widehat{\mathsf{T}}_{\Gamma,x:A^\perp}[Q] \mid (\nu w)(w(x').\widehat{\mathsf{T}}_{\Delta,x:A}[R] \mid \mathcal{S}^A_{z,w}))]\!] \\ &\quad \Leftrightarrow (\delta, \gamma) \in [\![(\nu x)(R \mid Q)]\!]\end{aligned}$$

Thus, the observations on the left-hand side are exactly those from $(\nu x)(R \mid Q)$ under some transformation; that transformation being the one induced by the transformers, having thus the same observation as $\widehat{\mathsf{T}}_{\Gamma,\Delta}[(\nu x)(R \mid Q)]$. When the last rule applied is either W or C, we rely on the I.H. and Lemma 4.8. ◀

▶ **Corollary 4.10.** *Given* $P \vdash^c \Delta$*, then* $(P)^\bullet \simeq \widehat{\mathsf{T}}_\Delta[P]$.

Proof. The proof follows from Corollary 2.3 and Lemma 4.9. ◀

Transformer contexts and $\mathbb{L}^*_\Delta(-)$. Transformer contexts induce a function on denotations, similar to $\mathbb{L}^*_\Delta(-)$ (Definition 3.6). In general, we have the following result, for any context.

▶ **Definition 4.11.** *For* $K[\square] \vdash^{\mathsf{c}}_k \Delta \parallel \Gamma$*, we define* $[\![K[\square]]\!] : \mathcal{P}([\![\Delta]\!]) \mapsto \mathcal{P}([\![\Gamma]\!])$ *inductively:*

$$[\![\square \vdash^{\mathsf{c}}_k \Delta \parallel \Delta]\!](X) = X$$

$$[\![(\nu x)(K[\square] \mid Q) \vdash^{\mathsf{c}}_k \Delta \parallel \Sigma, \Gamma]\!](X) = \left\{ (\sigma, \gamma) \;\middle|\; \begin{array}{c} (\sigma, a) \in [\![K[\square] \vdash^{\mathsf{c}}_k \Delta \parallel \Sigma, x : A]\!](X), \\ (\gamma, a) \in [\![Q \vdash^c \Gamma, x : A^\perp]\!] \end{array} \right\}$$

$$[\![K[\square] \mid Q \vdash^{\mathsf{c}}_k \Delta \parallel \Sigma, \Gamma]\!](X) = \{(\sigma, \gamma) \mid \sigma \in [\![K[\square] \vdash^{\mathsf{c}}_k \Delta \parallel \Sigma]\!](X), \quad \gamma \in [\![Q \vdash^c \Gamma]\!]\}$$

▶ **Lemma 4.12.** *Let* $K[\square] \vdash^{\mathsf{c}}_k \Delta \parallel \Gamma$ *and* $P \vdash^c \Delta$ *be a typed context and process, respectively. Then* $[\![K[\square] \vdash^{\mathsf{c}}_k \Delta \parallel \Gamma]\!]([\![P \vdash^c \Delta]\!]) = [\![K[P] \vdash^c \Gamma]\!]$.

Definition 4.11 can be specialized to transformer contexts, so as to obtain the following function: $[\![\widehat{\mathsf{T}}_\Delta[\square]]\!] : \mathcal{P}([\![\Delta]\!]) \to \mathcal{P}([\![\Delta^{\bullet\perp}, w : \mathbf{1}]\!])$. Putting all these elements together, we can show that transformers also internalize Laurent's translation on the level of denotations.

▶ **Lemma 4.13.** *For any type* $A \in \mathit{CP}_{02}$*, and for any* $a \in [\![A]\!]$*,* $b \in [\![A^{\bullet\perp}]\!]$*, we have*

$$(a, b) \in [\![\mathrm{T}^A_{x,y}]\!] \iff \mathbb{L}_A(a) = b$$

Proof. By induction on the type A. ◀

▶ **Theorem 4.14.** *For any typing context* Δ*, and for any set* $X \subseteq [\![\Delta]\!]$*,*

$$[\![\widehat{\mathsf{T}}_\Delta[\square] \vdash^{\mathsf{c}}_k \Delta \parallel \Delta^{\bullet\perp}, w : \mathbf{1}]\!](X) = \mathbb{L}^*_\Delta(X)$$

Proof. Let $\Delta = x_1 : A_1, \ldots, x_n : A_n$. Then, $[\![\Delta^{\bullet\perp}, w : \mathbf{1}]\!] = [\![{A_1}^{\bullet\perp}]\!] \times \cdots \times [\![{A_n}^{\bullet\perp}]\!] \times \{*\}$. Then, we reason as follows:

$$
\begin{aligned}
& (\delta_1, \ldots, \delta_n, *) \in [\![\widehat{\mathsf{T}}_\Delta[\square]]\!](X) \\
\iff & (\delta_1, \ldots, \delta_n) \in [\![\mathrm{T}_\Delta[\square]]\!](X) = [\![(\nu x_n)(\cdots(\nu x_1)(\square \mid \mathrm{T}^{A_1}_{x_1,x_1'}) \mid \cdots) \mid \mathrm{T}^{A_n}_{x_n,x_n'})]\!](X) \\
\iff & \exists d_n \in [\![A_n]\!].\ \begin{array}{l} (d_n, \delta_n) \in [\![\mathrm{T}^{A_n}_{x_n,x_n'}]\!] \quad \wedge \\ (\delta_1, \ldots, \delta_{n-1}, d_n) \in [\![(\nu x_{n-1})(\cdots(\nu x_1)(\square \mid \mathrm{T}^{A_1}_{x_1,x_1'}) \mid \cdots))]\!](X) \end{array} \\
\iff & \exists d_n \in [\![A_n]\!], \ldots, d_1 \in [\![A_1]\!].\ \begin{array}{l} (d_n, \delta_n) \in [\![\mathrm{T}^{A_n}_{x_n,x_n'}]\!] \ \wedge \cdots \wedge \ (d_1, \delta_1) \in [\![\mathrm{T}^{A_1}_{x_1,x_1'}]\!] \wedge \\ (d_1, \ldots, d_n) \in [\![\square]\!](X) \end{array} \\
\iff & \exists (d_1, \ldots, d_n) \in X.\ \mathbb{L}_{A_1}(d_1) = \delta_1 \wedge \cdots \wedge \mathbb{L}_{A_n}(d_n) = \delta_n \\
\iff & (\delta_1, \ldots, \delta_n, *) \in \mathbb{L}^*_\Delta(X)
\end{aligned}
$$
◀

Thus, Theorem 4.14 shows that $\widehat{\mathsf{T}}_\Delta[\square]$ is the proper internalization of Laurent's translation as a typed context in CP_{02}. In § 3 we have shown that Laurent's translation preserves and reflects equivalence of processes. Theorem 4.14 allows us to lift that result to processes with transformers, thus also obtaining a full abstraction result:

▶ **Corollary 4.15** (Full Abstraction (II)). *For all $P \in CP_{02}$,*

$$P \simeq Q \vdash^c \Delta \quad \Leftrightarrow \quad \widehat{\mathsf{T}}_\Delta[P] \simeq \widehat{\mathsf{T}}_\Delta[Q] \vdash^c \Delta^{\bullet\perp}, w : \mathbf{1}$$

Proof. Follows from the soundness of denotational semantics, Lemmata 3.7 and 4.12 and Theorem 4.14. ◀

5 Concluding Remarks

This paper has brought the translation $(-)^\bullet : \mathsf{CLL} \to \mathsf{ILL}$ (due to Laurent [16]), into the realm of concurrent interpretations of linear logic ("propositions-as-sessions"). As we have seen, under the "propositions-as-sessions" interpretation, the translation converts a classical process P into an intuitionistic process $(P)^\bullet$ (cf. Definition 3.3); then, exploiting the fact that $(P)^\bullet$ can be analyzed without changes in the classical setting, we contrast the behavior of P and $(P)^\bullet$ using Atkey's observational semantics for CP [1]. Our two full abstraction results (Corollary 3.12 and Corollary 4.15) give denotational and operational characterizations that extend the scope of Laurent's translation, and connect purely logical results with their corresponding computational interpretations. To our knowledge, ours is the first formal relationship of its kind.

Differences between classical and intuitionistic variants of "propositions-as-sessions" have already been observed by Caires and Pfenning [8] and by Wadler [24]. There are superficial differences, such as the nature/reading of typing judgments (already discussed) and the number of typing rules – classical interpretations have one rule per connective, whereas intuitionistic ones have two: one for expressing the reliance on a behavior, another for expressing an offer. But there are also more subtle differences, in particular the *locality* principle, which, informally speaking, ensures that received names can only be used for sending. Intuitionistic interpretations enforce locality for shared names. Consider, e.g., the process $P = x(y).!y(z).Q$, which uses the name y received on x to define a server behavior. Because P does not respect locality, it is not typable in the intuitionistic system of [8].

Prior work by Van den Heuvel and Pérez [22, 23] studies this specific difference: they study the sets of processes typable under classical and intuitionistic interpretations, and use non-local processes such as P to prove that the intuitionistic set is strictly included in the classical one. Crucially, this prior work focuses on typing, and does not formally relate the behavioral equivalences in the two classes, as we achieve here by coupling $(-)^{\bullet}$ with typed observational equivalences on processes. In fact, our results go beyond [22, 23] in that Definition 3.3 stipulates how to translate a process P with non-local servers into a corresponding process $(P)^{\bullet}$ with localized servers. This translation not only follows directly the logical translation by Laurent, but is also correct in a strong sense under the two different perspectives (denotational and operational) given by our full abstraction results.

The issue of translating non-local processes into local processes was studied, albeit in a different setting, by Boreale [5], who considers a calculus with locality as an intermediate language between the asynchronous π-calculus and the internal π-calculus. His work makes heavy use of link processes, which are closely related to the forwarding process $[x \leftrightarrow y]$ of CP. More fundamentally, because Borale's translations and results are framed in the untyped, asynchronous setting, comparisons with our work in the typed setting are difficult to draw.

We find it remarkable that our results leverage two separate, well-established technical ingredients, namely Laurent's translation and Atkey's observational equivalence and denotational semantics [1]. In particular, Atkey's denotational semantics, based on the relational semantics of CLL, is simple and effective for our purposes, and also amenable to extensions (like incorporating support for $\textsc{Mix}_2$). Indeed, our denotational characterization $\mathbb{L}_A(-)$ of Laurent's translation (Definition 3.6) benefits from this simplicity.

Our technical results make use of the mix principles – we use Rule $\textsc{Mix}_0$ in § 3 and also and Rule $\textsc{Mix}_2$ in § 4. The use of mix principles in the context of "propositions-as-sessions" has been analyzed by Atkey et al. [2]. Already, one difference between Wadler's presentation in [24] and Atkey's observational semantics in [1] is the use of Rule $\textsc{Mix}_0$. As we have briefly mentioned, our results in § 3 hold also for CP_{02}, the extension with both $\textsc{Mix}_0$ and $\textsc{Mix}_2$.

Finally, we note that Caires and Pfenning based their interpretation on Barber's Dual Intuitionistic Linear Logic (DILL) [4], which is based on sequents of the form $\Gamma; \Delta \vdash^d A$, where Γ and Δ specify unrestricted and linear assignments, respectively. This is a bit different from ILL as considered by Laurent. However, the two systems are equivalent (as logics), and so this difference does not jeopardize our results. Barber [4] provides translations between DILL and ILL and shows that ILL is isomorphic to the sub-system of DILL with sequents of the form $\cdot\,; \Delta \vdash^d A$. From the point of view of $(-)^{\bullet}$, this means that $\vdash^c \Delta$ is a provable in CLL iff $\cdot\,; \Delta^{\bullet} \vdash^d \mathbf{1}$ is provable in DILL. Hence we can regard $(P)^{\bullet}$ as a DILL process. This observation, together with the fact that $(P)^{\bullet}$ is typable with both $\Delta^{\bullet} \vdash^i \mathbf{1}$ and $\vdash^c \Delta^{\bullet\perp}, \mathbf{1}$, provides us with a solid groundwork for the computational interpretation of the translation.

Future Work. We intend to adapt our approach to other denotational semantics for typed languages under "propositions-as-sessions", such as the one by Kokke et al. [15], whose definition is inspired by Brzozowski derivatives and includes the polarity of names/channels. Also, we plan to study the potential of our full abstraction results as a tool for a better understanding of the locality principle for shared names in the session-typed setting. Moreover, it would be worthwhile exploring the consequences of varying the parameter R in Laurent's translation, which we currently instantiate with the simplest possible proposition/type.

From a more applied perspective, we believe that our work can shed light on connections between different existing implementation strategies for process calculi with session types based on linear logic. On the intuitionistic side, the work by Pfenning and Griffith develops SILL, a language based on ILL [17]; on the classical side, recent work by Caires and Toninho develops a Session Abstract Machine based on CLL [10]. It would be interesting to establish to what extent our work can be applied to connect such language implementations.

References

1 Robert Atkey. Observed communication semantics for classical processes. In Hongseok Yang, editor, *Programming Languages and Systems*, pages 56–82, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg. doi:10.1007/978-3-662-54434-1_3.

2 Robert Atkey, Sam Lindley, and J. Garrett Morris. *Conflation Confers Concurrency*, pages 32–55. Springer International Publishing, Cham, 2016. doi:10.1007/978-3-319-30936-1_2.

3 Stephanie Balzer and Frank Pfenning. Manifest sharing with session types. *Proc. ACM Program. Lang.*, 1(ICFP):37:1–37:29, 2017. doi:10.1145/3110281.

4 Andrew Barber. Dual intuitionistic linear logic, 1996. Technical report, available at https://www.lfcs.inf.ed.ac.uk/reports/96/ECS-LFCS-96-347/.

5 Michele Boreale. On the expressiveness of internal mobility in name-passing calculi. *Theoretical Computer Science*, 195(2):205–226, 1998. Concurrency Theory. doi:10.1016/S0304-3975(97)00220-X.

6 Luís Caires and Jorge A. Pérez. Linearity, control effects, and behavioral types. 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 229–259. Springer, 2017. doi:10.1007/978-3-662-54434-1_9.

7 Luís Caires, Jorge A. Pérez, Frank Pfenning, and Bernardo Toninho. Domain-aware session types. In Wan J. Fokkink and Rob van Glabbeek, editors, *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, volume 140 of *LIPIcs*, pages 39:1–39:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.CONCUR.2019.39.

8 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. doi:10.1007/978-3-642-15375-4_16.

9 Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logic propositions as session types. *Math. Struct. Comput. Sci.*, 26(3):367–423, 2016. doi:10.1017/S0960129514000218.

10 Luís Caires and Bernardo Toninho. The session abstract machine. In Stephanie Weirich, editor, *Programming Languages and Systems - 33rd European Symposium on Programming, ESOP 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part I*, volume 14576 of *Lecture Notes in Computer Science*, pages 206–235. Springer, 2024. doi:10.1007/978-3-031-57262-3_9.

11 Bor-Yuh Evan Chang, Kaustuv Chaudhuri, and Frank Pfenning. A judgmental analysis of linear logic. Technical Report CMU-CS-03-131R, Department of Computer Science, Carnegie Mellon University, November 2003. doi:10.1184/R1/6587498.v1.

12 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. doi:10.1007/978-3-319-89366-2_5.

13 Farzaneh Derakhshan, Stephanie Balzer, and Limin Jia. Session logical relations for noninterference. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–14. IEEE, 2021. doi:10.1109/LICS52264.2021.9470654.

14 Simon Fowler, Wen Kokke, Ornela Dardha, Sam Lindley, and J. Garrett Morris. Separating sessions smoothly. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference*, volume 203 of *LIPIcs*, pages 36:1–36:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPICS.CONCUR.2021.36`.

15 Wen Kokke, Fabrizio Montesi, and Marco Peressotti. Better late than never: a fully-abstract semantics for classical processes. *Proc. ACM Program. Lang.*, 3(POPL):24:1–24:29, 2019. `doi:10.1145/3290337`.

16 Olivier Laurent. Around classical and intuitionistic linear logics. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 629–638, New York, NY, USA, 2018. Association for Computing Machinery. `doi:10.1145/3209108.3209132`.

17 Frank Pfenning and Dennis Griffith. Polarized substructural session types. In Andrew M. Pitts, editor, *Foundations of Software Science and Computation Structures - 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9034 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2015. `doi:10.1007/978-3-662-46678-0_1`.

18 Jorge A. Pérez, Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logical relations and observational equivalences for session-based concurrency. *Information and Computation*, 239:254–302, 2014. `doi:10.1016/j.ic.2014.08.001`.

19 Zesen Qian, G. A. Kavvos, and Lars Birkedal. Client-server sessions in linear logic. *Proc. ACM Program. Lang.*, 5(ICFP):1–31, 2021. `doi:10.1145/3473567`.

20 Harold Schellinx. Some Syntactical Observations on Linear Logic. *Journal of Logic and Computation*, 1(4):537–559, September 1991. `doi:10.1093/logcom/1.4.537`.

21 Bernardo Toninho, Luís Caires, and Frank Pfenning. Dependent session types via intuitionistic linear type theory. In Peter Schneider-Kamp and Michael Hanus, editors, *Proceedings of the 13th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, July 20-22, 2011, Odense, Denmark*, pages 161–172. ACM, 2011. `doi:10.1145/2003476.2003499`.

22 Bas van den Heuvel and Jorge A. Pérez. Session type systems based on linear logic: Classical versus intuitionistic. In Stephanie Balzer and Luca Padovani, editors, *Proceedings of the 12th International Workshop on Programming Language Approaches to Concurrency- and Communication-cEntric Software, PLACES@ETAPS 2020, Dublin, Ireland, 26th April 2020*, volume 314 of *EPTCS*, pages 1–11, 2020. `doi:10.4204/EPTCS.314.1`.

23 Bas van den Heuvel and Jorge A. Pérez. Comparing session type systems derived from linear logic. *CoRR*, abs/2401.14763, 2024. `doi:10.48550/arXiv.2401.14763`.

24 Philip Wadler. Propositions as sessions. In *Proceedings of the 17th ACM SIGPLAN International Conference on Functional Programming*, ICFP '12, pages 273–286, New York, NY, USA, 2012. Association for Computing Machinery. `doi:10.1145/2364527.2364568`.

25 Philip Wadler. Propositions as sessions. *J. Funct. Program.*, 24(2-3):384–418, 2014. `doi:10.1017/S095679681400001X`.