Math. Struct. in Comp. Science (2016), *vol.* 26, *pp.* 868–906. © Cambridge University Press 2014
doi:10.1017/S0960129514000413 First published online 10 November 2014

Relating reasoning methodologies in linear logic and process algebra

YUXIN DENG[†], ROBERT J. SIMMONS[‡] and ILIANO CERVESATO[§]

[†]*Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China*
Email: yuxindeng@sjtu.edu.cn
[‡]*School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.*
Email: rjsimmon@cs.cmu.edu
[§]*Carnegie Mellon University, Qatar campus, Qatar*
Email: iliano@cmu.edu

Received 26 April 2013; revised 3 July 2014

We show that the proof-theoretic notion of logical preorder coincides with the process-theoretic notion of barbed preorder for a CCS-like process calculus obtained from the formula-as-process interpretation of a fragment of linear logic. The argument makes use of other standard notions in process algebra, namely simulation and labelled transition systems. This result establishes a connection between an approach to reason about process specifications, the barbed preorder, and a method to reason about logic specifications, the logical preorder.

1. Introduction

By now, execution-preserving relationships between (fragments of) linear logic and (fragments of) process algebras are well established (see Cervesato and Scedrov (2009) for an overview). Abramsky observed early on that linear cut elimination resembles reduction in CCS and the π-calculus (Milner 1989), thereby identifying processes with (some) linear proofs and establishing the *process-as-term* interpretation (Abramsky 1994). The alternative *process-as-formula* encoding, pioneered by Miller around the same time (Miller 1992), maps process constructors to logical connectives and quantifiers, with the effect of relating reductions in process algebra with proof steps, in the same way that logic programming achieves computation via proof search. Specifically, it describes the state of an evolving concurrent system as a linear logic context. Transitions between such process states are therefore modelled as transitions between linear logic contexts. As a member of a context, a formula stands for an individual process in the process state. On the right-hand side of an intuitionistic derivability judgment, it is a specification that a process state can satisfy. The process-as-formula interpretation has been used extensively in a multitude of domains, in particular in the fields of programming languages (Cervesato *et al.* 2002; Cervesato and Scedrov 2009; Miller 1992) and security (Cervesato *et al.* 2000). For example, Cervesato and Scedrov (2009) developed it into a logically-based rewriting formalism that subsumes and integrates both process-based (e.g. the π-calculus) and transition-based languages (e.g. Petri nets) for specifying concurrent systems. In Ehrhard and Laurent (2010), a finitary π-calculus has been translated into a version of differential interaction nets. In Dam (1994), positive linear and relevant logics are used to provide logical accounts of static process structures by interpreting formulas on process terms in Milner's synchronous CCS. Recent work by Caires *et al.* Caires and Pfenning (2010), Caires *et al.* (2012) provides a concurrent computational interpretation of intuitionistic linear logic. Propositions are interpreted as session types, sequent proofs as processes in the π-calculus, cut reductions as process reductions, and vice versa. Wadler (2012) formalized a tight connection between a standard presentation of session types and linear logic.

Not as well established is the relationship between the rich set of notions and techniques used to reason about process specifications and the equally rich set of techniques used to reason about (linear) logic. Indeed, a majority of investigations have attempted to reduce some of the behavioural notions that are commonplace in process algebra to derivability within logic. For example, Miller identified a fragment of linear logic that could be used to observe traces in his logical encoding of the π-calculus, thereby obtaining a language that corresponds to the Hennessy–Milner modal logic, which characterizes observational equivalence (Miller 1992). A similar characterization was made in Lincoln and Saraswat (1991), where a sequent $\Gamma \vdash \Delta$ in a classical logic augmented with constraints was seen as process state Γ passing test Δ. Extensions of linear logic were shown to better capture other behavioural relations: for example, adding definitions allows expressing simulation as the derivability of a linear implication (McDowell *et al.* 2003), but falls short of bisimulation. Bisimulation was captured in $FO\lambda^{\Delta\nabla}$, a further extension of linear logic (Tiu and Miller 2004). Interestingly, the nominal aspect of this logic was critical in encoding late bisimulation, while universal quantification sufficed for its open variant.

This body of work embeds approaches for reasoning *about* process specifications (e.g. bisimulation or various forms of testing) into methods for reasoning *with* logic (mainly derivability). Little investigation has targeted notions used to reason about logic (e.g. proof-theoretic definitions of equivalence). More generally, techniques and tools developed for each formalism rarely cross over – and may very well be rediscovered in due time. Tellingly, process-based definitions and proofs are often coinductive in nature, while techniques based on proof-theory are generally inductive.

This paper outlines one such relationship – between the inductive methods used to reason about logic and the coinductive methods used to reason about process calculi. On the linear logic side, we focus on the inductively-defined notion of logical preorder (obtained by forsaking symmetry from derivability-based logical equivalence). On the process-algebraic side, we consider an extensional behavioural relation adapted from the standard coinductive notion of barbed preorder. We prove that, for two fragments of linear logic and matching process calculi, these notions coincide. Our proofs rely on other standard process algebraic notions as stepping stones, namely simulation and labelled transition systems. The fragments of linear logic, when equipped with our labelled transition systems, are to some extent analogous to fragments of asynchronous CCS. Therefore, the simulation relation is tailored for asynchronous communicating processes.

The rest of the paper is organized as follows. In Section 2, we briefly review the general fragment of linear logic we are focusing on and define the logical preorder. Then, in

Section 3, we recall the standard process-as-formula interpretation for a sublanguage without exponentials and define the notion of barbed preorder. In Section 4, we prove their equivalence through the intermediary of a simulation preorder defined on the basis of a labelled transition system. In Section 5, we extend our results and proof techniques to accommodate exponentials. We conclude in Section 6 by anticipating future developments enabled by this work.

2. First-order intuitionistic linear logic

The starting point for our investigation will be intuitionistic linear logic (Girard 1987) *sans* disjunction $A \oplus B$ and its unit $\mathbf{0}$. The formulas of this fragment of propositional intuitionistic linear logic are defined by the following grammar (a is an atomic formula):

$$A, B, C ::= a \mid \mathbf{1} \mid A \otimes B \mid A \multimap B \mid \top \mid A \,\&\, B \mid !A.$$

Intuitionistic derivability for this fragment is expressed by means of sequents of the form

$$\Gamma; \Delta \vdash A$$

where the *unrestricted context* Γ and the *linear context* Δ are multisets of formulas. We define contexts by means of the following grammar:

$$\Gamma, \Delta ::= \cdot \mid \Delta, A$$

where '$\cdot$' represents the empty context, and ',' denotes the context extension operation: Δ, A is the context obtained by adding the formula A to the context Δ. As usual, we will tacitly treat ',' as an associative and commutative context union operator 'Δ_1, Δ_2' with the unit '$\cdot$', which indeed allows us to think of contexts as multisets.

Derivability in intuitionistic linear logic is defined by the inference rules in Figure 1, which follow a standard presentation of propositional intuitionistic linear logic, called dual intuitionistic linear logic (or DILL) (Barber 1996; Cervesato and Scedrov 2009). A sequent $\Gamma; \Delta \vdash A$ is derivable if there is a valid derivation with $\Gamma; \Delta \vdash A$ as its root. A DILL sequent $\Gamma; \Delta \vdash A$ corresponds to $!\Gamma, \Delta \vdash A$ in Girard's original presentation (Girard 1987). In the first part of this paper, we will be working with the sublanguage of linear logic where the unrestricted context Γ is empty and there are no occurrences of the formula $!A$. We will then abbreviate the purely linear sequent $\cdot; \Delta \vdash A$ as $\Delta \vdash A$.

The remainder of the section is as follows: in Section 2.1 we discuss the fundamental metatheoretical results for DILL – the core of the 'proof theorist's toolkit'. In Section 2.2, we introduce the *logical preorder*, a natural semantic relationship between contexts that will be our connection to the exploration of the 'process calculist's toolkit' in future sections. In Section 2.3, we reveal the nice connection between these two approaches. In Section 2.4, we give a purely inductive characterization of the logical preorder.

2.1. *Metatheory of linear logic*

DILL enjoys the same metatheoretical properties as traditional logic. In particular, it supports inlining lemmas (which is formalized as a property called the 'admissibility of

$$\frac{}{\Gamma; a \vdash a}\ \textit{init} \qquad \frac{\Gamma, A; \Delta, A \vdash C}{\Gamma, A; \Delta \vdash C}\ \textit{clone}$$

$$\frac{}{\Gamma; \cdot \vdash \mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma; \Delta \vdash C}{\Gamma; \Delta, \mathbf{1} \vdash C}\ \mathbf{1}L$$

$$\frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2 \vdash B}{\Gamma; \Delta_1, \Delta_2 \vdash A \otimes B}\ \otimes R \qquad \frac{\Gamma; \Delta, A, B \vdash C}{\Gamma; \Delta, A \otimes B \vdash C}\ \otimes L$$

$$\frac{\Gamma; \Delta, A \vdash B}{\Gamma; \Delta \vdash A \multimap B}\ \multimap R \qquad \frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2, B \vdash C}{\Gamma; \Delta_1, \Delta_2, A \multimap B \vdash C}\ \multimap L$$

$$\frac{}{\Gamma; \Delta \vdash \top}\ \top R \qquad \textit{(no rule } \top L\textit{)}$$

$$\frac{\Gamma; \Delta \vdash A \quad \Gamma; \Delta \vdash B}{\Gamma; \Delta \vdash A \,\&\, B}\ \& R \qquad \frac{\Gamma; \Delta, A_i \vdash C}{\Gamma; \Delta, A_1 \,\&\, A_2 \vdash C}\ \& L_i$$

$$\frac{\Gamma; \cdot \vdash A}{\Gamma; \cdot \vdash\, !A}\ !R \qquad \frac{\Gamma, A; \Delta \vdash C}{\Gamma; \Delta, !A \vdash C}\ !L$$

Fig. 1. Dual intuitionistic linear logic.

cut') and proving any formula on the basis of its own assumption (this 'identity' lemma generalizes rule *init* to arbitrary formulas). These properties are proved using Gentzen's methodology, the aforementioned standard toolkit of a working proof theorist (Gentzen 1935; Pfenning 2000). The relevant theorems are these:

Proposition 2.1 (weakening). If $\Gamma; \Delta \vdash A$, then $\Gamma, \Gamma'; \Delta \vdash A$ for any Γ'.

Proof. By induction on the structure of the given derivation. □

Proposition 2.2 (identity). $\Gamma; A \vdash A$ for all formulas A and unrestricted contexts Γ.

Proof. By induction on the structure of the formula A. □

This proposition entails that the following 'identity' rule is admissible:

$$\frac{}{\Gamma; A \vdash A}\ \textit{id}.$$

Proposition 2.3 (cut admissibility). If $\Gamma; \Delta \vdash A$ and $\Gamma'; \Delta', A \vdash C$, then we have $\Gamma', \Gamma; \Delta', \Delta \vdash C$.

Proof. We generalize the induction hypothesis by proving the statement of the theorem together with the statement '*if* $\Gamma; \cdot \vdash A$ *and* $\Gamma', A; \Delta' \vdash C$, *then* $\Gamma', \Gamma; \Delta' \vdash C$.' Then the proof proceeds by mutual, lexicographic induction, first on the structure of the formula A, and second by induction on the structure of the two given derivations (Pfenning 2000). □

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

Proposition 2.3 is called the 'cut admissibility' theorem because the proof indicates that these two 'cut' rules are admissible:

$$\frac{\Gamma;\Delta \vdash A \quad \Gamma';\Delta',A \vdash C}{\Gamma',\Gamma;\Delta',\Delta \vdash C}\,cut \qquad \frac{\Gamma;\cdot \vdash A \quad \Gamma',A;\Delta' \vdash C}{\Gamma',\Gamma;\Delta' \vdash C}\,cut\,!$$

From the proof theory perspective, Propositions 2.2 and 2.3 are canonical and establish fundamental properties of the logic; any system of rules not validating the identity and cut admissibility theorems can hardly be called a logic. One relevant point about the *proofs* of Propositions 2.2 and 2.3 is that they are quite modular with respect to various sublanguages of linear logic. Essentially any syntactic restriction of the language of formulas, and certainly every restriction considered in this paper, preserves the validity of the identity and cut properties. They also hold for larger fragments of linear logic, for example with additive disjunction or quantifiers (the latter case requires upgrading the derivability judgment with a context Σ of first-order variables and therefore the statements of the last two propositions above).

2.2. *The logical preorder*

Girard's seminal paper on linear logic (Girard 1987) already hinted at a relationship between the then-new formalism and concurrency. This intuition was later fleshed out into several interpretations of process algebras into linear logic (Abramsky 1994; Miller 1992). In this paper, we will follow the *process-as-formula* approach (Miller 1992), which understand each process constructor as one of the connectives or quantifiers of linear logic. A process is therefore mapped to a linear logic formula and a system of processes to the context $\Gamma;\Delta$ of a sequent $\Gamma;\Delta \vdash A$. The formula A on the right-hand side of this sequent is viewed as an observation or property about these processes. We will formalize and further discuss these notions in Section 3.

The study of process calculi is primarily concerned with the relationships between different processes, but the only judgment that we have so far for linear logic is derivability, $\Gamma;\Delta \vdash A$, which expresses that the *process state* $(\Gamma;\Delta)$ satisfies the formula or specification A.

In the formulation of logics for process calculi, a central judgment is $P \models \phi$, which states that the process P satisfies some formula ϕ. This leads to a natural definition of a *logical preorder*, where $P \preceq_l Q$ if $P \models \phi$ implies $Q \models \phi$ for all ϕ; in other words, if the set of formulas satisfied by the former is included in the set of formulas satisfied by the latter. See e.g. Hennessy and Milner (1985) for an example in classical process algebra and Deng and Du (2011) and Deng *et al.* (2007) in probabilistic process algebra. Our goal is to give a definition of $P \preceq_l Q$ where P and Q are process states in their incarnation as linear logic formulas.

If we choose to look at the derivability judgment $\Gamma;\Delta \vdash A$ as analogous to the judgment $P \models \phi$, then it gives us an obvious way of relating process states: a specific process state $(\Gamma_{specific};\Delta_{specific})$ is generalized by a general process state $(\Gamma_{general};\Delta_{general})$ if all formulas A satisfied by the specific process are also satisfied by the general process. In other words,

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

we can define a preorder $(\Gamma_{specific}; \Delta_{specific}) \preceq_l (\Gamma_{general}; \Delta_{general})$, which we will also call the logical preorder, which holds if $\Gamma_{specific}; \Delta_{specific} \vdash C$ implies $\Gamma_{general}; \Delta_{general} \vdash C$ for all C.

This is an intuitively reasonable definition, but, as we explain below, that definition requires us to assume some specific properties of the logic. By giving a slightly more general (but ultimately equivalent) definition of the logical preorder, we can avoid making *a priori* assumptions about the relationship between logical derivability and the logical preorder.

Definition 2.4 (logical preorder). The *logical preorder* is the smallest relation $\preceq_l$ on states such that $(\Gamma_1; \Delta_1) \preceq_l (\Gamma_2; \Delta_2)$ if, for all Γ', Δ', and C, we have $\Gamma', \Gamma_1; \Delta', \Delta_1 \vdash C$ implies $\Gamma', \Gamma_2; \Delta', \Delta_2 \vdash C$.

The only difference between our informal motivation and Definition 2.4 is that, in the latter, the specific and general process states must both satisfy the same formula C after being *extended* with the same unrestricted context Γ' and linear context Δ'.

Because the $\multimap R$ and $!L$ rules are invertible (for each rule, the conclusion implies the premise), this definition is equivalent to the definition that does not use extended contexts. This is because, if $\Gamma' = A_1, \ldots, A_n$ and $\Delta' = B_1, \ldots, B_m$, then the judgment $\Gamma', \Gamma; \Delta', \Delta \vdash C$ can be derived if and only if the judgment

$$\Gamma; \Delta \vdash\ !A_1 \multimap \cdots \multimap !A_n \multimap B_1 \multimap \cdots \multimap B_m \multimap C$$

can be derived. However, the proof of invertibility relies on the metatheory of linear logic as presented in the previous section. The more 'contextual' version in Definition 2.4 allows us to talk about the relationship between the derivability judgment and the logical preorder without baking in any assumptions about invertibility.

Altogether, we are taking the view, common in practice, that the context part of the sequent $(\Gamma; \Delta)$ represents the state of some system component and that the consequent A corresponds to some property satisfied by this system. Then, the logical preorder compares specifications on the basis of the properties they satisfy, possibly after the components they describe are plugged into a larger system.

As a sanity check, we can verify that the relation $\preceq_l$ is indeed a preorder. Note that this is independent of the actual rules defining derivability. It follows only from the definition of the relation and the fact that informal implication is itself reflexive and transitive. This means that $\preceq_l$ is a preorder for any fragment of linear logic we may care to study.

Theorem 2.5. The logical preorder $\preceq_l$ is a preorder.

Proof. We need to show that $\preceq_l$ is reflexive and transitive.

Reflexivity: Expanding the definition, the reflexivity statement, $(\Gamma; \Delta) \preceq_l (\Gamma; \Delta)$ for all A assumes the form '*for all* Γ', Δ' *and* A *if* $\Gamma, \Gamma'; \Delta, \Delta' \vdash A$, *then* $\Gamma, \Gamma'; \Delta, \Delta' \vdash A$'. This holds trivially.

Transitivity: We want to prove that if $(\Delta_1; \Gamma_1) \preceq_l (\Delta_2; \Gamma_2)$ and $(\Delta_2; \Gamma_2) \preceq_l (\Delta_3; \Gamma_3)$ then $(\Delta_1; \Gamma_1) \preceq_l (\Delta_3; \Gamma_3)$. By the definition of $\preceq_l$, we know that for any Γ', Δ' and A', if $\Gamma_1, \Gamma'; \Delta_1, \Delta' \vdash A'$ then $\Gamma_2, \Gamma'; \Delta_2, \Delta' \vdash A'$. Similarly, for any Γ'', Δ'' and A'', if $\Gamma_2, \Gamma''; \Delta_2, \Delta'' \vdash A''$ then $\Gamma_3, \Gamma''; \Delta_3, \Delta'' \vdash A''$. If we choose Γ'', Δ'' and A'' to be precisely

Γ', Δ' and A' respectively, we can chain these two implications, obtaining that for any Γ', Δ' and A', if $\Gamma_1, \Gamma'; \Delta_1, \Delta' \vdash A'$ then $\Gamma_3, \Gamma'; \Delta_3, \Delta' \vdash A'$. This is however exactly the definition of logical preorder: therefore $(\Delta_1; \Gamma_1) \preceq_l (\Delta_3; \Gamma_3)$.

Therefore $\preceq_l$ is indeed a preorder. □

2.3. *Relating cut, identity, and the logical preorder*

We have now defined two judgments. The derivability judgment of linear logic, written as $\Gamma; \Delta \vdash A$, declares that the process state $(\Gamma; \Delta)$ meets the specification set by A. The logical preorder $(\Gamma_1; \Delta_1) \preceq_l (\Gamma_2; \Delta_2)$ says that the process state $(\Gamma_1; \Delta_1)$ can act as a specific instance of the more general process state $(\Gamma_2; \Delta_2)$. Recall from the introduction that linear logic formulas have two natures: they are both (1) the specifications that a process state can satisfy and (2) the atomic constituents of a process state. By the convention that $(\cdot; \Delta)$ can be written as Δ, we can also think of the formula A as synonymous with the singleton process state $(\cdot; A)$.

The derivability judgment $\Gamma; \Delta \vdash A$ says that A is one specification that the process state $(\Gamma; \Delta)$ satisfies; of course it may satisfy many other specifications as well – *interpreted as a specification*, the formula A is specific, and the process state $(\Gamma; \Delta)$ is general. We relate a specific and general process states with the logical preorder: $(\Gamma_{specific}; \Delta_{specific}) \preceq_l (\Gamma_{general}; \Delta_{general})$. This suggests that, if the two natures of a formula are in harmony, we can expect that $\Gamma; \Delta \vdash A$ exactly when $(\cdot; A) \preceq_l (\Gamma; \Delta)$, a suggestion that is captured by the following proposition:

Proposition 2.6 (harmony for the logical preorder). $\Gamma; \Delta \vdash A$ if and only if $A \preceq_l (\Gamma; \Delta)$.

Proof. This will be a simple corollary of Theorem 2.7 below – read on. □

Proposition 2.6 should be seen as a sort of sanity check that the logical preorder's notion of 'more specific' and 'more general' makes sense relative to the derivability judgment's notion. A result that initially surprised us is that this sanity check is exactly equivalent to the classic sanity checks of the proof theorists: identity and cut admissibility.

Theorem 2.7. Proposition 2.6 holds if and only if Propositions 2.2 and 2.3 hold.

Proof. This theorem establishes the equivalence between two propositions; both directions can be established independently.

Assuming harmony, prove identity and cut. For the identity theorem, we must show $A \vdash A$ for some arbitrary formula A. By harmony, it suffices to show $A \preceq_l A$, and $\preceq_l$ is reflexive (Theorem 2.5).

For the cut admissibility theorem, we are given $\Gamma; \Delta \vdash A$ and $\Gamma'; \Delta', A \vdash C$ and must prove $\Gamma', \Gamma; \Delta', \Delta \vdash C$. By harmony and the first given derivation, we have $A \preceq_l (\Gamma; \Delta)$. Expanding Definition 2.4, this means that, for all $\overline{\Gamma'}$, $\overline{\Delta'}$, and $\overline{C}$, we have $\overline{\Gamma'}; \overline{\Delta'}, A \vdash \overline{C}$ implies $\overline{\Gamma'}, \Gamma; \overline{\Delta'}, \Delta \vdash \overline{C}$. So by letting $\overline{\Gamma'} = \Gamma'$, $\overline{\Delta'} = \Delta'$, and $\overline{C} = C$ and applying the second given derivation $\Gamma'; \Delta', A \vdash C$, we get $\Gamma', \Gamma; \Delta', \Delta \vdash C$, which was what we needed to prove.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

Assuming identity and cut, prove harmony. For the forward implication, we are given $\Gamma;\Delta \vdash A$ and we need to prove $A \preceq_l (\Gamma;\Delta)$. Expanding Definition 2.4, this means that, for an arbitrary Γ', Δ' and C, we are given $\Gamma';\Delta',A \vdash C$ and need to prove $\Gamma',\Gamma;\Delta',\Delta \vdash C$. The statement of cut admissibility, Proposition 2.3, is that if $\Gamma;\Delta \vdash A$ and $\Gamma';\Delta',A \vdash C$, then $\Gamma',\Gamma;\Delta',\Delta \vdash C$. We have the two premises, and the conclusion is what we needed to prove.

For the reverse implication, we are given $A \preceq_l (\Gamma;\Delta)$ and we need to prove $\Gamma;\Delta \vdash A$. Expanding Definition 2.4, this means that, for all $\overline{\Gamma'}$, $\overline{\Delta'}$, and $\overline{C}$, we have that $\overline{\Gamma'};\overline{\Delta'},A \vdash \overline{C}$ implies $\overline{\Gamma'},\Gamma;\overline{\Delta'},\Delta \vdash \overline{C}$. So by letting $\overline{\Gamma'} = \cdot$, $\overline{\Delta'} = \cdot$, and $\overline{C} = A$, we have that $A \vdash A$ implies $\Gamma;\Delta \vdash A$. The conclusion is what we needed to show, and the premise is exactly the statement of identity, Proposition 2.2, so we are done. □

It is critical to note that Theorem 2.7 holds *entirely independently* of the actual definition of the derivability judgment $\Gamma;\Delta \vdash A$. This means, in particular, that it holds independently of the actual validity of Propositions 2.2 and 2.3 as they were presented here, and that it holds for any alternative definition of the derivability judgment that we might present. Furthermore, because Propositions 2.2 and 2.3 do, of course, hold for DILL, Proposition 2.6 holds as a simple corollary of Theorem 2.7.

The proof theorist's sanity checks are motivated by arguments that are somewhat philosophical. In Girard's *Proofs and Types*, cut and identity are motivated by an observation that hypotheses and conclusions should have equivalent epistemic strength Girard *et al.* (1989). Martin-Löf's *judgmental methodology* gives a weaker sanity check, *local soundness* Martin-Löf (1996), which was augmented by Pfenning and Davies with a sanity check of *local completeness* (Pfenning and Davies 2001). Local soundness and completeness take the verificationist viewpoint that the meaning of a logical connective is given by its introduction rules in a natural deduction presentation of the logic. This means that the elimination rules must be justified as neither too strong (soundness) nor too weak (completeness) relative to the introduction rules. The surprising (at least, initially, to us) equivalence of Proposition 2.6 to the critical sanity checks of sequent calculi suggests that the process state interpretation of linear logic can actually be treated as fundamental, that is, as a philosophical organizing principle for sequent calculus presentations of logic. This connection between the logical preorder and the core reasoning methodologies for linear logic grounds our subsequent exploration, which will rely more heavily on the reasoning methodologies used in the study of process algebra.

2.4. *Re-characterizing the logical preorder*

In the previous section, we have argued for the canonicity of the logical preorder by motivating harmony and showing that harmony is equivalent to the canonical properties of cut and identity. However, it is not obvious, given our discussion so far that it is easy or even possible to prove interesting properties of the logical preorder. In this section, we show that there is an alternate characterization of the logical preorder for DILL directly in terms of the existing derivability judgment $\Gamma;\Delta \vdash A$. This re-characterization makes it more obvious that the logical preorder is a fundamentally inductive concept. However, it

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

relies on the provability rules for a specific logic, here DILL, while our original definition is fully general.

This re-characterization depends on the auxiliary concepts of the *tensorial product* of a linear context Δ, written $\bigotimes\Delta$, and *exponential linearization* of an unrestricted context Γ, written $!\Gamma$, which are defined as follows:

$$\begin{cases} \bigotimes(\cdot) & = \mathbf{1} \\ \bigotimes(\Delta, A) & = (\bigotimes\Delta)\otimes A \end{cases} \qquad \begin{cases} !(\cdot) & = \cdot \\ !(\Gamma, A) & = (!\Gamma), !A \end{cases}$$

The main result of this section is given by the following theorem.

Theorem 2.8. $(\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2)$ iff $\Gamma_2;\Delta_2 \vdash \bigotimes\Delta_1 \otimes \bigotimes !\Gamma_1$.

Proof. ($\Rightarrow$) Assume that $(\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2)$. Then, by definition, for every Γ, Δ and A such that $\Gamma_1,\Gamma;\Delta_1,\Delta \vdash A$ is derivable, there is a derivation of $\Gamma_2,\Gamma;\Delta_2,\Delta \vdash A$. Take $\Gamma = \Delta = \cdot$ and $A = \bigotimes\Delta_1 \otimes \bigotimes !\Gamma_1$. If we can produce a derivation of $\Gamma_1;\Delta_1 \vdash \bigotimes\Delta_1 \otimes \bigotimes !\Gamma_1$, then our desired result follows. Here is a schematic derivation of this sequent:

$$\dfrac{\dfrac{}{\Gamma_1;\Delta_1 \vdash \bigotimes\Delta_1}\otimes R, \mathbf{1}R, id \qquad \dfrac{\cdots \quad \dfrac{\dfrac{\dfrac{}{\Gamma_1;C_i \vdash C_i}\,id}{\Gamma_1;\cdot \vdash C_i}\,clone}{\Gamma_1;\cdot \vdash !C_i}\,!R}{\Gamma_1;\cdot \vdash \bigotimes !\Gamma_1}\otimes R, \mathbf{1}R}{\Gamma_1;\Delta_1 \vdash \bigotimes\Delta_1 \otimes \bigotimes !\Gamma_1}\otimes R.$$

Here, we assume that Γ_1 expands to $C_1,\ldots,C_i,\ldots,C_n$ for an appropriate $n \geqslant 0$.

($\Leftarrow$) Assume now that $\Gamma_2;\Delta_2 \vdash \bigotimes\Delta_1 \otimes \bigotimes !\Gamma_1$ with derivation $\mathcal{D}$. To show that $(\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2)$, we need to show that, given Γ, Δ, A and a derivation $\mathcal{D}_1$ of $\Gamma_1,\Gamma;\Delta_1,\Delta \vdash A$, we can construct a derivation of $\Gamma_2,\Gamma;\Delta_2,\Delta \vdash A$.

To do so, we start by weakening $\mathcal{D}$ with Γ and $\mathcal{D}_1$ with Γ_2 by means of Proposition 2.1. Let $\mathcal{D}'$ and $\mathcal{D}_1'$ be the resulting derivations of $\Gamma_2,\Gamma;\Delta_2 \vdash \bigotimes\Delta_1 \otimes \bigotimes !\Gamma_1$ and $\Gamma_2,\Gamma_1,\Gamma;\Delta_1,\Delta \vdash A$ respectively. We then build the desired derivation schematically as follows:

$$\dfrac{\begin{matrix}\mathcal{D}' \\ \Gamma_2,\Gamma;\Delta_2 \vdash \bigotimes\Delta_1 \otimes \bigotimes !\Gamma_1\end{matrix} \qquad \dfrac{\dfrac{\begin{matrix}\mathcal{D}_1' \\ \Gamma_2,\Gamma_1,\Gamma;\Delta_1,\Delta \vdash A\end{matrix}}{\Gamma_2,\Gamma;\Delta_1,!\Gamma_1,\Delta \vdash A}\,!L}{\Gamma_2,\Gamma;\bigotimes\Delta_1 \otimes \bigotimes !\Gamma_1,\Delta \vdash A}\otimes L, \mathbf{1}L}{\Gamma_2,\Gamma;\Delta_2,\Delta \vdash A}\,cut.$$

□

This replaces an extensional test that considers arbitrary contexts and goal formulas with an existential test that only requires exhibiting a single derivation. This makes it evident that the logical preorder is a semi-decidable relation. It is interesting to specialize this result to the fragment of our language without exponentials nor permanent contexts. We obtain:

Corollary 2.9. $\Delta_1 \preceq_l \Delta_2$ iff $\Delta_2 \vdash \bigotimes \Delta_1$.

For this language fragment, it is decidable whether $\Delta \vdash A$ has a derivation. Therefore, the logical preorder relation is decidable too: given Δ_1 and Δ_2, it is decidable whether $\Delta_1 \preceq_l \Delta_2$.

3. Process interpretation and the barbed preorder

The remainder of this paper will explore the relationship between the logical preorder just introduced and a second relation, the barbed preorder, that emerges from trying to directly understand transitions on linear logic process states. We will do so gradually. Indeed, this section and the next will concentrate on a fragment of intuitionistic linear logic as presented in Section 2. This language is given by the following grammar:

$$\textit{Formulas} \qquad A, B, C ::= a \mid \mathbf{1} \mid A \otimes B \mid a \multimap B \mid \top \mid A \,\&\, B.$$

This fragment is purely linear (there is no exponential $!A$) and the antecedents of linear implications are required to be atomic. Our derivability judgment will always have an empty unrestricted context. Therefore, we will write it simply $\Delta \vdash A$. As a consequence, the definition of logical preorder simplifies to '$\Delta_1 \preceq_l \Delta_2$ iff, for all Δ and A, if $\Delta_1, \Delta \vdash A$ then $\Delta_2, \Delta \vdash A$'.

This section is written from a process calculus perspective; we will forget, for now, that we presented a sequent calculus in Figure 1 that assigns meaning to the propositions of this fragment of linear logic. In that story, the meaning of propositions is given by the definition of derivability $\Delta \vdash A$ which treats A as a specification that the process state Δ satisfies. In this story, we will try to understand propositions directly by describing their behaviour as constituents of a process state directly. The result is somewhat analogous to a fragment of CCS (Milner 1989) with CSP-style internal choice (Hoare 1985):

a	atomic process that sends a
$\mathbf{1}$	null process
$A \otimes B$	process that forks into processes A and B
$a \multimap B$	process that receives a and continues as B
$\top$	stuck process
$A \,\&\, B$	process that can behave either as A or as B

According to this interpretation, a process state Δ is a system of interacting processes represented by formulas. Then, the empty context '$\cdot$' is interpreted as a null process while the context constructor ',' is a top-level parallel composition. This structure, which makes contexts commutative monoids, is interpreted as imposing a structural equivalence among the corresponding systems of processes. We write this equivalence as $\equiv$ when stressing this interpretation. It is the least relation satisfying the following equations:

$$\Delta, \cdot \equiv \Delta,$$
$$\Delta_1, \Delta_2 \equiv \Delta_2, \Delta_1,$$
$$\Delta_1, (\Delta_2, \Delta_3) \equiv (\Delta_1, \Delta_2), \Delta_3.$$

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

$$\begin{array}{rcll}
(\Delta, \mathbf{1}) & \rightsquigarrow & \Delta & (\rightsquigarrow\mathbf{1}) \\
(\Delta, A \otimes B) & \rightsquigarrow & (\Delta, A, B) & (\rightsquigarrow\otimes) \\
(\Delta, A \,\&\, B) & \rightsquigarrow & (\Delta, A) & (\rightsquigarrow\&_1) \\
(\Delta, A \,\&\, B) & \rightsquigarrow & (\Delta, B) & (\rightsquigarrow\&_2) \\
(\Delta, a, a \multimap B) & \rightsquigarrow & (\Delta, B) & (\rightsquigarrow\multimap) \\
\multicolumn{4}{c}{\textit{(No rule for } \top\textit{)}}
\end{array}$$

Fig. 2. The transition formulation of a fragment of linear logic.

We will always consider contexts modulo this structural equality, and therefore treat equivalent contexts as syntactically identical.

The analogy with CCS above motivates the reduction relation between process states defined in Figure 2. A formula $A \otimes B$ (parallel composition) transitions to the two formulas A and B in parallel (rule $\rightsquigarrow\otimes$), for instance, and a formula $A \,\&\, B$ (choice) either transitions to A or to B (rules $\rightsquigarrow\&_1$ and $\rightsquigarrow\&_2$). The rule corresponding to implication is also worth noting: a formula $a \multimap B$ can interact with an atomic formula a to produce the formula B; we think of the atomic formula a as sending a message asynchronously and $a \multimap B$ as receiving that message.

3.1. *Proof theory interlude*

A proof theorist will recognize the strong relationship between the rules in Figure 2 and the left rules from the sequent calculus presentation in Figure 1. This relationship has been studied in details in Cervesato and Scedrov (2009). It is made explicit in the current setting by the following proposition:

Proposition 3.1. If $\Delta \rightsquigarrow \Delta'$ and $\Delta' \vdash C$, then $\Delta \vdash C$.

Proof. This proof proceeds by case analysis on the reduction $\Delta \rightsquigarrow \Delta'$. Most of the cases are straightforward, we give two. If we have a reduction by $\rightsquigarrow\&_2$, then we have $\Delta, A \vdash C$ and must prove $\Delta, A \,\&\, B \vdash C$, which we can do by rule $\&_{L2}$. Alternatively, if we have a reduction by $\rightsquigarrow\multimap$, then we have $\Delta \vdash B$ and must prove $\Delta, a, a \multimap B \vdash C$. We can prove $a \vdash a$ by *init*, and therefore the result follows by $\multimap_L$. □

This theorem can also be understood as demonstrating the admissibility of the following rule:

$$\frac{\Delta \rightsquigarrow \Delta' \qquad \Delta' \vdash A}{\Delta \vdash A}\; \textit{transition}.$$

This single rule essentially replaces all the left rules in Figure 1. This includes rule $\multimap L$ in the instance studied here (and larger fragment), a derivation of whose left premise can always be inlined so the context Δ_1 degenerates to just a. A detailed study, inclusive of all proofs, can be found in Cervesato and Scedrov (2009).

Let $\rightsquigarrow^*$ be the reflexive and transitive closure of $\rightsquigarrow$. Then an easy induction proves a variant of Proposition 3.1 that strengthens $\rightsquigarrow$ to $\rightsquigarrow^*$. It yields the following admissible rule.

$$\frac{\Delta \rightsquigarrow^* \Delta' \qquad \Delta' \vdash A}{\Delta \vdash A}\ \textit{transition}^*$$

Another statement that we can prove is that given the *transition* rule or the *transition** rule, all the left rules of the sequent calculus are admissible. Such a proof is straightforward; the only interesting case is proving $\multimap L$ admissible. However, we will not discuss this property any further in this paper. (This concludes our proof theory interlude.)

We will now define the barbed preorder in Section 3.2 and explore some of its properties in Section 3.3.

3.2. *The barbed preorder*

As we once again forget that we know anything about the proof-theoretic notion of derivability, we turn to the following question: what does it mean for one process state to behave like another process state?[†] Any relation $\mathcal{R}$ that declares $\Delta_1\ \mathcal{R}\ \Delta_2$ when Δ_1's behaviour can be imitated by Δ_2 will be called a *behavioural preorder*,[‡] and we will describe a set of desiderata for what makes a good behavioural preorder.

The most fundamental thing a process can do is to be observed; observations are called 'barbs' in the language of the process calculists. We start with the idea that we can observe only the presence of an atomic proposition in a context. Therefore, if we want to claim (Δ_1, a) is imitated by Δ_2 (that is, if $(\Delta_1, a)\ \mathcal{R}\ \Delta_2$ for some behavioural preorder $\mathcal{R}$), then we should be able to observe the presence of a in Δ_2. But it is not quite right to require that $\Delta_2 \equiv (\Delta_2', a)$; we need to give the process state Δ_2 some time to compute before it is forced to cough up an observation. For this reason we introduce the auxiliary notation $\Delta \Downarrow_a$, which says that $\Delta \rightsquigarrow^* (\Delta', a)$ for some Δ'. This auxiliary notion lets us define our first desideratum: *behavioural preorders should be barb-preserving*. That is, if we say that (Δ_1, a) can be imitated by Δ_2, then it had better be the case that $\Delta_2 \Downarrow_a$.

The next two desiderata do not require any auxiliary concepts. If a process Δ_1 is imitated by Δ_2, and $\Delta_1 \rightsquigarrow \Delta_1'$, then we should expect Δ_2 to be able to similarly evolve to a state that imitates Δ_1'. In other words, *behavioural preorders should be reduction-closed*. The third desideratum has to do with surrounding contexts. If Δ_1 is imitated by Δ_2, then if we put both Δ_1 and Δ_2 in parallel with some other process state Δ', then (Δ_2, Δ') should still imitate (Δ_1, Δ'). In other words, *behavioural preorders should be compositional*.

These three desiderata – barb-preservation, reduction-closure, and compositionality – are standard ideas in what we have been calling the 'process calculist's toolkit,' and it would be possible to define behavioural preorders entirely in terms of these three

[†] The following discussion is intended to be a gentle introduction to some core ideas in process calculus. We want to stress that, with the exception of partition-preservation, this section discusses standard and elementary concepts in the process calculus literature.

[‡] This is a convenient misnomer: while the largest behavioural preorder (which we will name the *barbed preorder*) will turn out to be a true preorder, we will not stipulate that every behavioural preorder is a proper preorder, and indeed many of them are not reflexive – the empty relation will satisfy all of our desiderata for being a behavioural 'preorder.'

desiderata. To foreshadow the developments of the next section, we will eventually want to show that the logical preorder $\Delta_1 \preceq_l \Delta_2$ is *sound* – that it is a behavioural preorder according to these desiderata. This would be provable: the logical preorder *is* barb-preserving, reduction-closed, and compositional. However, the logical preorder is *incomplete* with respect to these three desiderata. Here is a sketch of the reason why: there is a barb-preserving, reduction-closed, and compositional behavioural preorder which says that the process state $(a \multimap \mathbf{1}, b \multimap \mathbf{1})$ is imitated by the process state $(a \multimap b \multimap \mathbf{1})$.[†] However, $(a \multimap \mathbf{1}, b \multimap \mathbf{1}) \not\preceq_l (a \multimap b \multimap \mathbf{1})$ because $a \multimap \mathbf{1}, b \multimap \mathbf{1} \vdash (a \multimap \mathbf{1}) \otimes (b \multimap \mathbf{1})$ but $a \multimap b \multimap \mathbf{1} \nvdash (a \multimap \mathbf{1}) \otimes (b \multimap \mathbf{1})$.

The logical preorder is, to a degree, fixed and canonical due to its relationship with the standard metatheory of the sequent calculus, so if we want the logical preorder to be complete with respect to our desiderata, we have to add additional desideratum. The culprit for incompleteness, as we identified it, was the derivation rule for $A \otimes B$, which requires us to *partition* a process state into two parts and observe those parts independently. The nullary version of this is $\mathbf{1}$, the unit of $\otimes$, which requires us to split a process state into zero pieces; that is only possible if the process state's linear context is empty. Motivated by this possibility, we add a fourth desideratum, that *behavioural preorders should be partition-preserving*. In the binary case, this means that, if $(\Delta_{1a}, \Delta_{1b})$ is imitated by Δ_2, then it must be the case that $\Delta_2 \leadsto^* (\Delta_{2a}, \Delta_{2b})$ where Δ_{1a} is imitated by Δ_{2a} and Δ_{1b} is imitated by Δ_{2b}. In the nullary case, this means that, if $\cdot$ is imitated by Δ, then it must be the case that $\Delta_2 \leadsto^* \cdot$.[‡]

Formally, these desiderata are captured by the following definition. In this definition, we use the more traditional notation from process calculus and say that $\Delta \downarrow_a$ if $\Delta = (\Delta', a)$ for some Δ'.

Definition 3.2. Let $\mathcal{R}$ be a binary relation over states. We say that $\mathcal{R}$ is

— *barb-preserving* if, whenever $\Delta_1 \mathcal{R} \Delta_2$ and $\Delta_1 \downarrow_a$ for any atomic proposition a, we have that $\Delta_2 \Downarrow_a$.
— *reduction-closed* if $\Delta_1 \mathcal{R} \Delta_2$ implies that whenever $\Delta_1 \leadsto \Delta_1'$, there exists Δ_2' such that $\Delta_2 \leadsto^* \Delta_2'$ and $\Delta_1' \mathcal{R} \Delta_2'$.
— *compositional* if $\Delta_1 \mathcal{R} \Delta_2$ implies $(\Delta', \Delta_1) \mathcal{R} (\Delta', \Delta_2)$ for all Δ'.
— *partition-preserving* if $\Delta_1 \mathcal{R} \Delta_2$ implies that

1. if $\Delta_1 \equiv \cdot$, then $\Delta_2 \leadsto^* \cdot$,

[†] The proof that there is a barb-preserving, reduction-closed, and compositional relation $\mathcal{R}$ such that $(a \multimap \mathbf{1}, b \multimap \mathbf{1}) \mathcal{R} (a \multimap b \multimap \mathbf{1})$ is complex. The simplest way we know how to establish this is to define a labelled transition system which is equivalent to the largest barb-preserving, reduction-closed, and compositional relation. We can then show that, according to this labelled transition system, $(a \multimap \mathbf{1}, b \multimap \mathbf{1})$ is related to $(a \multimap b \multimap \mathbf{1})$ in a simulation for *asynchronous* communicating processes. This is essentially the same development we will perform in Section 4 for $\preceq_b$, the largest barb-preserving, reduction-closed, compositional, *and partition-preserving* relation, that we are about to define.

[‡] A concept reminiscent of this notion of partition-preservation has been given in probabilistic process algebras (Deng and Hennessy 2011); the similar concept goes by the name *Markov bisimulation* in that setting. The idea is that, in order to compare two distributions, we decompose them into an equal number of point distributions on states and then compare states pointwise. In the literature there are also formalisms with locality-aware semantics, such as the synchronous CCS and spatial logic (Caires and Cardelli 2003).

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

2. for all Δ_{1a} and Δ_{1b}, if $\Delta_1 \equiv (\Delta_{1a}, \Delta_{1b})$ then there exist Δ_{2a} and Δ_{2b} such that $\Delta_2 \leadsto^* (\Delta_{2a}, \Delta_{2b})$ and furthermore $\Delta_{1a} \mathcal{R} \Delta_{2a}$ and $\Delta_{1b} \mathcal{R} \Delta_{2b}$.

These desiderata are useful in letting us conclude that one process *does not* imitate another. Barb-preservation lets us conclude that a is not imitated by (b, b, b), and reduction-closure furthermore lets us conclude that $(b, b \multimap a)$ is not imitated by (b, b, b). Partition-preservation lets us conclude that $(a \multimap b, a \multimap c)$ is not imitated by $a \multimap a \multimap (b \otimes c)$. Compositionality lets us conclude that $(a \multimap b, b \multimap c)$ is not imitated by $(a \multimap c)$, because if we put both process states in parallel with the process state b, then we would be able to step, in the former case, to $(a \multimap b, c)$ and then observe that $(a \multimap b, c) \downarrow_c$; the process $(a \multimap c, b)$ is unable to keep up.

Concluding that $(a \multimap \mathbf{1}, b \multimap \mathbf{1})$ is not imitated by $(a \multimap b \multimap \mathbf{1})$ – as is required by our goal of completeness relative to the logical preorder – requires compositionality and partition-preservation. If $(a \multimap \mathbf{1}, b \multimap \mathbf{1})$ is imitated by $(a \multimap b \multimap \mathbf{1})$, then by partition preservation (and the fact that $(a \multimap b \multimap \mathbf{1})$ can make no transitions), we must be able to split $(a \multimap b \multimap \mathbf{1})$ into two parts, Δ and Δ' so that $a \multimap \mathbf{1}$ is imitated by Δ and $b \multimap \mathbf{1}$ is imitated by Δ'. That necessarily means that $\Delta = \cdot$ and $\Delta' = (a \multimap b \multimap \mathbf{1})$ or vice versa. Because of compositionality, if $(a \multimap \mathbf{1})$ were imitated by $\cdot$ then $(a, a \multimap \mathbf{1})$ would be imitated by a, but $(a, a \multimap \mathbf{1}) \leadsto^* \cdot$ and $a \not\leadsto^* \cdot$. Therefore, $(a \multimap \mathbf{1})$ cannot be imitated by $\cdot$, and by a similar argument $(b \multimap \mathbf{1})$ cannot be imitated by $\cdot$. This furthermore refutes the proposition that $(a \multimap \mathbf{1}, b \multimap \mathbf{1})$ is imitated by $(a \multimap b \multimap \mathbf{1})$, as we had hoped. Partition-preservation is discussed further in Remark 4.17.

Our desiderata allow us to define the most generous behavioural preorder by coinduction; we call this relation the *barbed preorder*. The definition has an innocent-until-proven-guilty flavour: unless there is some reason, arising from the desiderata, that one process cannot imitate another, then the barbed preorder declares the two process states to be in relation. This coinductive definition is standard from process algebra (modulo our additional requirement of partition-preservation).

Definition 3.3 (barbed preorder). The *barbed preorder*, denoted by $\preceq_b$, is the largest relation over process states which is barb-preserving, reduction-closed, compositional and partition-preserving.

Barbed equivalence, which is the symmetric closure of the barbed preorder, has been widely studied in concurrency theory, though its appearance in linear logic seems to be new. It is also known as *reduction barbed congruence* and used in a variety of process calculi (Honda and Yoshida 1995; Rathke and Sobocinski 2008; Fournet and Gonthier 2005; Deng and Hennessy 2011).

3.3. *Properties of the barbed preorder*

Having defined the barbed preorder as the largest barb-preserving, reduction-closed, compositional, and partition-preserving binary relation over process states, we will close out this section by proving a few technical lemmas.

We start with three small 'multistep' lemmas: the first lets us act as if reduction closure was defined exclusively in terms of $\leadsto^*$, the second lets us act as if barb-preservation was defined exclusively in terms of $\Downarrow_a$, and the third lets us do something similar for partition preservation. These will be used later on, and also help us prove that $\preceq_b$ is actually a preorder. Theorem 3.7 establishes that $\preceq_b$ is indeed a preorder. Differently from the common phrase 'behavioural preorders' discusses above, which are not necessarily preorders, our barbed preorder is a genuine preorder. Finally, Lemma 3.9 is a technical lemma about observations that we need later on in the proof of Theorem 4.12, and the atom renaming lemma (Lemma 3.8) is needed to prove this technical lemma.

Lemma 3.4 (multistep reduction closure). Suppose $\Delta_1 \preceq_b \Delta_2$. If $\Delta_1 \leadsto^* \Delta_1'$, then there exists a Δ_2' such that $\Delta_2 \leadsto^* \Delta_2'$ and $\Delta_1' \preceq_b \Delta_2'$.

Proof. We proceed by induction on the number of steps in $\Delta_1 \leadsto^* \Delta_1'$.

— Suppose $\Delta_1 \leadsto^* \Delta_1'$ in zero steps (that is, $\Delta_1 \equiv \Delta_1'$). Then $\Delta_2 \leadsto^* \Delta_2$ in zero steps and $\Delta_1' \preceq_b \Delta_2$ by assumption.
— Suppose $\Delta_1 \leadsto \Delta_1'' \leadsto^* \Delta_1'$. Since $\Delta_1 \preceq_b \Delta_2$, there exists some Δ_2'' such that $\Delta_2 \leadsto^* \Delta_2''$ and $\Delta_1'' \preceq_b \Delta_2''$. The induction hypothesis then implies the existence of some Δ_2' such that $\Delta_2'' \leadsto^* \Delta_2'$ and $\Delta_1' \preceq_b \Delta_2'$. Since the relation $\leadsto^*$ is transitive, we have $\Delta_2 \leadsto^* \Delta_2'$ as required.

□

Lemma 3.5 (multistep barb-preservation). Suppose $\Delta_1 \preceq_b \Delta_2$. If $\Delta_1 \Downarrow_a$ then $\Delta_2 \Downarrow_a$.

Proof. If $\Delta_1 \Downarrow_a$, then there exists Δ_1' with $\Delta_1 \leadsto^* \Delta_1'$ and $\Delta_1' \downarrow_a$. By multistep reduction closure (Lemma 3.4), there exists Δ_2' such that $\Delta_2 \leadsto^* \Delta_2'$ and $\Delta_1' \preceq_b \Delta_2'$. The latter and $\Delta_1' \downarrow_a$ together imply $\Delta_2' \Downarrow_a$, i.e. there exists some Δ_2'' such that $\Delta_2' \leadsto^* \Delta_2''$ and $\Delta_2'' \downarrow_a$. The transitivity of $\leadsto^*$ then yields $\Delta_2 \leadsto^* \Delta_2''$. It follows that $\Delta_2 \Downarrow_a$. □

Lemma 3.6 (multistep partition-preservation). Suppose $\Delta_1 \preceq_b \Delta_2$. If $\Delta_1 \leadsto^* \cdot$, then $\Delta_2 \leadsto^* \cdot$, and if $\Delta_1 \leadsto^* (\Delta_{1a}, \Delta_{1b})$, then there exist Δ_{2a} and Δ_{2b} such that $\Delta_2 \leadsto^* (\Delta_{2a}, \Delta_{2b})$, $\Delta_{1a} \preceq_b \Delta_{2a}$, and $\Delta_{1b} \preceq_b \Delta_{2b}$.

Proof. In the first case, we assume $\Delta_1 \leadsto^* \cdot$ and have by multistep reduction closure (Lemma 3.4) that $\Delta_2 \leadsto^* \Delta_2'$ such that $\cdot \preceq_b \Delta_2'$. Then, by partition-preservation, we have that $\Delta_2' \leadsto^* \cdot$. The result then follows by the transitivity of $\leadsto^*$.

In the second case, we assume $\Delta_1 \leadsto^* (\Delta_{1a}, \Delta_{1b})$ and have by multistep reduction closure (Lemma 3.4) that $\Delta_2 \leadsto^* \Delta_2'$ such that $(\Delta_{1a}, \Delta_{1b}) \preceq_b \Delta_2'$. Then, by partition-preservation, we have that $\Delta_2' \leadsto^* (\Delta_{2a}, \Delta_{2b})$ such that $\Delta_{1a} \preceq_b \Delta_{2a}$ and $\Delta_{1b} \preceq_b \Delta_{2b}$. The result then follows by the transitivity of $\leadsto^*$. □

The following three proofs, of Theorem 3.7, Lemma 3.8, and Lemma 3.9, proceed by coinduction. We prove a property $\mathcal{P}$ of the barbed preorder $\preceq_b$ by defining some relation $\mathcal{R}$ in such a way that it is barb-preserving, reduction-closed, compositional, and partition-preserving, which establishes that $\mathcal{R} \subseteq \preceq_b$. Below we only provide a detailed proof for Lemma 3.9.

Theorem 3.7. $\preceq_b$ is a preorder.

Proof. A preorder is a reflexive and transitive relation. It is straightforward to show that the identity relation, $\mathcal{R}_{id}$ such that $\Delta\ \mathcal{R}_{id}\ \Delta$ for all Δ, is barb-preserving, reduction-closed, compositional, and partition-preserving. Since $\preceq_b$ is the largest relation with these properties, we have that $\mathcal{R}_{id} \subseteq \preceq_b$ and therefore $\preceq_b$ is reflexive.

For transitivity we can define the relation

$$\mathcal{R} \quad := \quad \{(\Delta_1, \Delta_3) \mid \text{ there is some } \Delta_2 \text{ with } \Delta_1 \preceq_b \Delta_2 \text{ and } \Delta_2 \preceq_b \Delta_3\}$$

and show that $\mathcal{R} \subseteq \preceq_b$.

□

Lemma 3.8 (atom renaming). If ρ is a bijective function substituting atomic propositions for atomic propositions, then $\rho\Delta_1 \preceq_b \rho\Delta_2$ implies $\Delta_1 \preceq_b \Delta_2$.

Proof. Two extra lemmas are needed to prove this property. The first, the *transition renaming property*, is that $\Delta \rightsquigarrow \Delta'$ iff $\rho\Delta \rightsquigarrow \rho\Delta'$. This is shown by case analysis on the given reduction. The second, the *multistep transition renaming property*, is that $\Delta \rightsquigarrow^* \Delta'$ iff $\rho\Delta \rightsquigarrow^* \rho\Delta'$. This is shown by induction on the structure of the given reduction and the transition renaming property. Then we define the relation $\mathcal{R} \ := \ \{(\Delta_1, \Delta_2) \mid \rho\Delta_1 \preceq_b \rho\Delta_2\}$ and show that $\mathcal{R} \subseteq \preceq_b$.

□

Lemma 3.9 (fresh atom removal). If $(\Delta_1, a) \preceq_b (\Delta_2, a)$, where a occurs neither in Δ_1 nor in Δ_2, then $\Delta_1 \preceq_b \Delta_2$.

Proof. Consider the relation

$$\mathcal{R} \quad := \quad \{(\Delta_1, \Delta_2) \mid \text{there exists } a \notin (\Delta_1 \cup \Delta_2) \text{ with } (\Delta_1, a) \preceq_b (\Delta_2, a)\}.$$

It will suffice to show that $\mathcal{R} \subseteq \preceq_b$, because in that case, given $(\Delta_1, a) \preceq_b (\Delta_2, a)$ for some a that occurs neither in Δ_1 or Δ_2, we will know that $\Delta_1\ \mathcal{R}\ \Delta_2$, which will in turn imply $\Delta_1 \preceq_b \Delta_2$. We can show that $\mathcal{R}$ is barb-preserving, reduction-closed, compositional and partition-preserving. Suppose $\Delta_1\ \mathcal{R}\ \Delta_2$, that is that $(\Delta_1, a) \preceq_b (\Delta_2, a)$ for some arbitrary a such that $a \notin \Delta_1$ and $a \notin \Delta_2$.

Barb-preserving. We assume $\Delta_1 \downarrow_b$ for some arbitrary b and must show that $\Delta_2 \Downarrow_b$. It cannot be the case that $\Delta_1 \downarrow_a$, so we have $a \neq b$. Pick another fresh atomic proposition $c \notin (\Delta_1 \cup \Delta_2)$. By the definition of $\downarrow_b$ and the assumption $\Delta_1 \downarrow_b$, we know that $\Delta_1 \equiv \Delta_1', b$. By the compositionality on $(\Delta_1, a) \preceq_b (\Delta_2, a)$ we get $(b \multimap a \multimap c, \Delta_1, a) \preceq_b (b \multimap a \multimap c, \Delta_2, a)$. Using rule $\rightsquigarrow\multimap$ we infer that

$$(b \multimap a \multimap c, \Delta_1, a) \rightsquigarrow (a \multimap c, \Delta_1', a) \rightsquigarrow (\Delta_1', c),$$

which means $(b \multimap a \multimap c, \Delta_1, a) \Downarrow_c$. Then we have $(b \multimap a \multimap c, \Delta_2, a) \Downarrow_c$ by multistep barb-preservation (Lemma 3.5). That is, $(b \multimap a \multimap c, \Delta_2, a) \rightsquigarrow^* (\Delta', c)$. By induction on the structure of (8), $b \multimap a \multimap c$ must be consumed to produce c, meaning that $\Delta_2 \rightsquigarrow^* (\Delta_2', b)$ and, consequently, $\Delta_2 \Downarrow_b$.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

Reduction-closed. We assume $\Delta_1 \rightsquigarrow \Delta_1'$ for some arbitrary Δ_1'. Clearly, we have $(\Delta_1, a) \rightsquigarrow (\Delta_1', a)$. By the definition of reduction-closure on $(\Delta_1, a) \preceq_b (\Delta_2, a)$, we obtain $(\Delta_2, a) \rightsquigarrow^* \Delta^*$ and $(\Delta_1', a) \preceq_b \Delta^*$. Moreover, the former implies $\Delta^* \equiv (\Delta_2', a)$ with $a \notin \Delta_2'$ for some Δ_2', by induction on the multistep reduction. So we can rewrite the latter as $(\Delta_1', a) \preceq_b (\Delta_2', a)$, which means that $\Delta_1' \mathcal{R} \Delta_2'$.

Compositional. We must show $(\Delta, \Delta_1) \mathcal{R} (\Delta, \Delta_2)$ for arbitrary Δ. Pick another fresh atomic proposition $b \notin (\Delta_1 \cup \Delta_2 \cup \Delta)$. By atom renaming (Lemma 3.8) on $(\Delta_1, a) \preceq_b (\Delta_2, a)$ we get $(\Delta_1, b) \preceq_b (\Delta_2, b)$. Then $(\Delta, \Delta_1, b) \preceq_b (\Delta, \Delta_2, b)$ for any Δ by compositionality. It follows that $(\Delta, \Delta_1) \mathcal{R} (\Delta, \Delta_2)$.

Partition-preserving. We first assume $\Delta_1 \equiv \cdot$. Since $(\Delta_1, a) \preceq_b (\Delta_2, a)$ we have $a \preceq_b (\Delta_2, a)$. Then $(a, a \multimap \mathbf{1}) \preceq_b (\Delta_2, a, a \multimap \mathbf{1})$ by compositionality. By rule $\rightsquigarrow\multimap$, rule $\rightsquigarrow\mathbf{1}$, and the transitivity of $\rightsquigarrow^*$, we see that $(a, a \multimap \mathbf{1}) \rightsquigarrow^* \cdot$. Then $(\Delta_2, a, a \multimap \mathbf{1}) \rightsquigarrow^* \cdot$ by multistep partition-preservation. By structural induction on Δ_2, we obtain that $\Delta_2 \rightsquigarrow^* \cdot$.

We next assume $\Delta_1 \equiv (\Delta_{1a}, \Delta_{1b})$ and must show that $\Delta_2 \rightsquigarrow^* (\Delta_{2a}, \Delta_{2b})$ such that $\Delta_{1a} \mathcal{R} \Delta_{2a}$ and $\Delta_{1b} \mathcal{R} \Delta_{2b}$. Pick another fresh atomic proposition $b \notin (\Delta_1 \cup \Delta_2)$. By the compositionality on $(\Delta_1, a) \preceq_b (\Delta_2, a)$, we have $(\Delta_{1a}, \Delta_{1b}, a, b) \preceq_b (\Delta_2, a, b)$. By the definition of partition-preserving, we get $(\Delta_2, a, b) \rightsquigarrow^* (\Delta_{2a}', \Delta_{2b}')$ with $(\Delta_{1a}, a) \preceq_b \Delta_{2a}'$ and $(\Delta_{1b}, b) \preceq_b \Delta_{2b}'$. By barb-preserving we have $\Delta_{2a}' \Downarrow_a$, which means $\Delta_{2a}' \equiv (\Delta_{2a}, a)$ for some Δ_{2a} with $a \notin \Delta_{2a}$ by induction on the reduction steps. It follows that $\Delta_{1a} \mathcal{R} \Delta_{2a}$. Similarly, we can obtain $\Delta_{2b}' \equiv (\Delta_{2b}, b)$ for some Δ_{2b} with $\Delta_{1b} \mathcal{R} \Delta_{2b}$.

This suffices to show that $\mathcal{R} \subseteq \preceq_b$, which concludes the proof. □

Note that Lemma 3.9 is invalid if a is not fresh with respect to Δ_1 and Δ_2. For example, we have

$$a \preceq_b (a \multimap a, a) \quad \text{but} \quad (\cdot) \not\preceq_b (a \multimap a).$$

This will be easy to check when using the results in Section 4.5.

4. Labelled transitions and the simulation preorder

We will now show that, for the restricted fragment of linear logic given in Section 3, the logical preorder and the barbed preorder coincide. Observe that it is hard to directly use the barbed preorder to show that one process actually imitates another. For example, in order to check the compositionality of a relation, all possible surrounding contexts need to be considered due to the presence of a universal quantifier. To overcome the problem, the key idea here is to coinductively define a third preorder, which is a labelled simulation relation based on labelled transition systems, so as to act as a stepping stone between the logical and barbed preorders.

The overall structure of the relevant proofs up to and including this section is shown in Figure 3. In Sections 4.1–4.3, we will present our stepping stones: a labelled transition system and the simulation preorder, and in Section 4.4 we will prove some properties of this preorder. Then, in Section 4.5 we show that the simulation and barbed preorders coincide, and in Section 4.6 we show that the simulation and logical preorders coincide; it is an obvious corollary that the logical and barbed preorders coincide.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

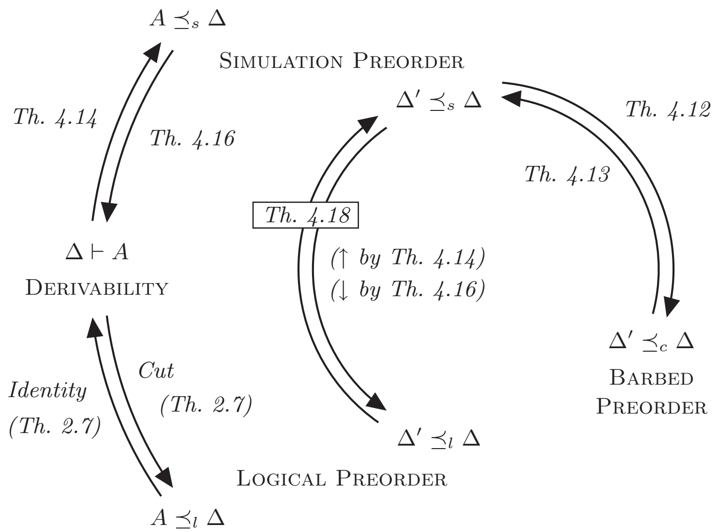

Fig. 3. Visual summary of equivalence proofs in Sections 3 and 4.

4.1. *Labelled transitions*

We will characterize the barbed preorder as a coinductively defined relation. For that purpose, we give a labelled transition semantics for states. Labels, or actions, are defined by the following grammar:

Non-receive actions	$\alpha ::= \tau \mid !a$
Generic actions	$\beta ::= \alpha \mid ?a$

We distinguish 'non-receive' labels, denoted α, as either the silent action τ or a label $!a$ representing a send action. Generic labels β extend them with receive actions $?a$.

The labelled transition semantics for our states, written using the judgment $\Delta \xrightarrow{\beta} \Delta'$, is defined by the rules in Figure 4. Since $\top$ is a process that is stuck, it has no action to perform. We write $\overset{\tau}{\Longrightarrow}$ for the reflexive transitive closure of $\xrightarrow{\tau}$, and $\Delta \overset{\beta}{\Longrightarrow} \Delta'$ for $\Delta \overset{\tau}{\Longrightarrow}\xrightarrow{\beta}\overset{\tau}{\Longrightarrow} \Delta'$, if $\beta \neq \tau$. Note that τ transitions correspond to reductions, as expressed by Lemma 4.1:

Lemma 4.1. $\Delta_1 \xrightarrow{\tau} \Delta_2$ if and only if $\Delta_1 \rightsquigarrow \Delta_2$, and $\Delta_1 \overset{\tau}{\Longrightarrow} \Delta_2$ if and only if $\Delta_1 \rightsquigarrow^* \Delta_2$.

Proof. The proof of the first statement is by case analysis on the τ transition rules in Figure 4 in one direction and by case analysis on the reduction rules in Figure 2 in the other.

The second statement follows from the first by induction on the number of steps taken. □

Now we can use the labelled transition system to define a simulation relation; simulation is defined by coinduction and echoes the definition of the barbed preorder in many ways.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

$$\frac{}{(\Delta, a) \xrightarrow{!a} \Delta}\ lts{-}atom \qquad \frac{}{(\Delta, a \multimap B) \xrightarrow{?a} (\Delta, B)}\ lts{\multimap}$$

$$\frac{}{(\Delta, \mathbf{1}) \xrightarrow{\tau} \Delta}\ lts\mathbf{1} \qquad \frac{}{(\Delta, A \otimes B) \xrightarrow{\tau} (\Delta, A, B)}\ lts\otimes$$

$$\frac{}{(\Delta, A \,\&\, B) \xrightarrow{\tau} (\Delta, A)}\ lts\,\&_1 \qquad \frac{}{(\Delta, A \,\&\, B) \xrightarrow{\tau} (\Delta, B)}\ lts\,\&_2$$

$$\text{(No rule for } \top) \qquad \frac{\Delta_1 \xrightarrow{!a} \Delta_1' \quad \Delta_2 \xrightarrow{?a} \Delta_2'}{(\Delta_1, \Delta_2) \xrightarrow{\tau} (\Delta_1', \Delta_2')}\ lts{-}com$$

Fig. 4. Labelled transition system.

However, it critically lacks the compositionality requirement that appears in the definition of the barbed preorder.

Definition 4.2 (simulation). A relation $\mathcal{R}$ between two processes represented as Δ_1 and Δ_2 is a *simulation* if $\Delta_1 \mathcal{R} \Delta_2$ implies

1. if $\Delta_1 \equiv \cdot$, then $\Delta_2 \stackrel{\tau}{\Longrightarrow} \cdot$;
2. if $\Delta_1 \equiv (\Delta_1', \Delta_1'')$, then $\Delta_2 \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$ for some Δ_2', Δ_2'' such that $\Delta_1' \mathcal{R} \Delta_2'$ and $\Delta_1'' \mathcal{R} \Delta_2''$;
3. whenever $\Delta_1 \xrightarrow{\alpha} \Delta_1'$, there exists Δ_2' such that $\Delta_2 \stackrel{\alpha}{\Longrightarrow} \Delta_2'$ and $\Delta_1' \mathcal{R} \Delta_2'$;
4. whenever $\Delta_1 \xrightarrow{?a} \Delta_1'$, there exists Δ_2' such that $(\Delta_2, a) \stackrel{\tau}{\Longrightarrow} \Delta_2'$ and $\Delta_1' \mathcal{R} \Delta_2'$;

We write $\Delta_1 \preceq_s \Delta_2$ if there is some simulation $\mathcal{R}$ such that $\Delta_1 \mathcal{R} \Delta_2$.

The last clause in the above definition is inspired by relevant notions of bisimulation for asynchronous process calculi e.g. Amadio *et al.* (1998). The 'natural' answer for Δ_2 to perform the action $?a$ is not compulsory. If $\Delta_2 \stackrel{?a}{\Longrightarrow} \Delta_2'$, then we can also obtain Δ_2' by the transition $(\Delta_2, a) \stackrel{\tau}{\Longrightarrow} \Delta_2'$. In the asynchronous setting, the sender of a message does not know when the message will be consumed by a receiver because an asynchronous observer cannot directly detect the receive actions of the observed process.

4.2. *Examples*

The barbed preorder was good at showing that one process was *not* imitated by another, but the simulation preorder is useful for showing that a process state *is* simulated by another. We will give a few examples, most of which implicitly utilize the following property:

Remark 4.3. Given a formula A and a context Δ, to check if $A \preceq_s \Delta$ holds, there is no need to consider clause (2) in Definition 4.2 because it holds vacuously.

Proof. We are given that $A \equiv (\Delta_1', \Delta_1'')$ and we have to pick a Δ_2' and a Δ_2'' such that $\Delta_2 \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$, $\Delta_1' \preceq_s \Delta_2'$, and $\Delta_1'' \preceq_s \Delta_2''$. Without loss of generality we can say that $\Delta_1' \equiv A$ and $\Delta_1'' \equiv \cdot$ – the other case, where $\Delta_1' \equiv \cdot$ and $\Delta_1'' \equiv A$, is symmetric. We pick Δ_2' to be Δ_2, pick Δ_2'' to be $\cdot$, and we must show

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

— $\Delta_2 \stackrel{\tau}{\Longrightarrow} (\Delta_2, \cdot)$ – this is immediate from the reflexivity of $\stackrel{\tau}{\Longrightarrow}$ and the definition of barbed equivalence.
— $A \preceq_s \Delta_1$ – this is what we initially set out to prove, so it follows immediately from the coinduction hypothesis.
— $\cdot \preceq_s \cdot$ – this follows from condition 1, none of the other conditions are applicable.

□

Given the remark above, we can show that $\top \preceq_s \Delta$ for any Δ, because $\top$ is neither empty nor able to perform any actions. Therefore, conditions 1, 3, and 4 are met vacuously. However, $\mathbf{1} \not\preceq_s \top$, because $\top$ cannot be reduced to $\cdot$ and $\mathbf{1}$ can (condition 1).

As another example, we have that $(a \multimap a) \preceq_s (\cdot)$. Ignoring condition 2 as before, the only possible transition for $(a \multimap a)$ is $(a \multimap a) \xrightarrow{?a} a$. We match this action, according to condition 4, by letting $a \stackrel{\tau}{\Longrightarrow} a$; we then must show $a \preceq_s a$, which again follows by reflexivity (Lemma 4.8, which we will be proving momentarily). This example is reminiscent of the law $a.\bar{a} = \mathbf{0}$ in the asynchronous π-calculus (Amadio *et al.* 1998). If a process receives a message and then sends it out, it does not exhibit any visible action to an observer who communicates asynchronously with the process. Here we also notice that simulation is incomparable with trace inclusion relation, with the usual definition of traces on labelled transition systems. On one hand, the process $(a \multimap a)$ has the trace $?a!a$ while $(\cdot)$ does not. On the other hand, the process $(a \multimap \mathbf{1}) \otimes (b \multimap \mathbf{1})$ is not simulated by $(a \multimap b \multimap \mathbf{1}) \& (b \multimap a \multimap \mathbf{1})$ but the traces of the former also appear in the latter.

Finally, we show that $(a \multimap b \multimap A) \preceq_s (b \multimap a \multimap A)$. The only transition possible for the (purportedly) simulated process state is $(a \multimap b \multimap A) \xrightarrow{?a} (b \multimap A)$, which means that we can proceed by showing that $(b \multimap a \multimap A, a) \stackrel{\tau}{\Longrightarrow} (b \multimap a \multimap A, a)$ (immediate from the reflexivity of $\stackrel{\tau}{\Longrightarrow}$) and that $(b \multimap A) \preceq_s (b \multimap a \multimap A, a)$. To prove this, we observe that the only transition possible for the (purportedly) simulated process is $(b \multimap A) \xrightarrow{?b} A$, which means that we can proceed by showing that $(b \multimap a \multimap A, a, b) \stackrel{\tau}{\Longrightarrow} A$ (which can be done in two steps) and that $A \preceq_s A$, which again follows from reflexivity (Lemma 4.8 again).

Another way of looking at this last example is that, in the case where A is $\mathbf{1}$, we have proved that the binary relation

$$\{(a \multimap b \multimap \mathbf{1}, b \multimap a \multimap \mathbf{1}), \quad (b \multimap \mathbf{1}, (b \multimap a \multimap \mathbf{1}, a)), \quad (\mathbf{1}, \mathbf{1}), \quad (\cdot, \cdot)\}$$

is a simulation. The simulation also works in the other direction – $(b \multimap a \multimap A) \preceq_s (a \multimap b \multimap A)$. Again in the case where A is $\mathbf{1}$, this is the same as proving that the binary relation

$$\{(b \multimap a \multimap \mathbf{1}, a \multimap b \multimap \mathbf{1}), \quad (a \multimap \mathbf{1}, (a \multimap b \multimap \mathbf{1}, b)), \quad (\mathbf{1}, \mathbf{1}), \quad (\cdot, \cdot)\}$$

is a simulation.

However, the two process states are *not* bisimilar according to the usual definition of bisimilarity: there is no *single* relation that simultaneously establishes that $b \multimap a \multimap \mathbf{1}$ is simulated by $a \multimap b \multimap \mathbf{1}$ and, if we flip the relation around, establishes that $a \multimap b \multimap \mathbf{1}$ is simulated by $b \multimap a \multimap \mathbf{1}$. To see why, consider the pair $(b \multimap \mathbf{1}, (b \multimap a \multimap \mathbf{1}, a))$ in the first of the two simulation relations above. The two process states in the pair are *not* similar in both directions: $b \multimap \mathbf{1} \preceq_s (b \multimap a \multimap \mathbf{1}, a)$ but $(b \multimap a \multimap \mathbf{1}, a) \not\preceq_s b \multimap \mathbf{1}$, because

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

$(b \multimap a \multimap \mathbf{1}, a) \xrightarrow{!a} b \multimap a \multimap \mathbf{1}$ while $b \multimap \mathbf{1} \not\xRightarrow{!a}$ (Condition 3). Thus, there is no way to construct a bisimulation.

4.3. *Properties of labelled transitions*

Towards the ultimate end of proving that the largest simulation, $\preceq_s$, is actually a preorder, we will need a few facts about labelled transitions.

Lemma 4.4 (compositionality of labelled transitions).

1. If $\Delta_1 \xrightarrow{\beta} \Delta_2$, then $(\Delta, \Delta_1) \xrightarrow{\beta} (\Delta, \Delta_2)$.
2. If $\Delta_1 \xRightarrow{\beta} \Delta_2$, then $(\Delta, \Delta_1) \xRightarrow{\beta} (\Delta, \Delta_2)$.

Proof. To prove the first statement, we proceed by induction on the derivation of $\Delta_1 \xrightarrow{\beta} \Delta_2$. There are 7 cases according to Figure 4. All of them are immediate, except the case of *lts−com*, which we expand. Suppose $\Delta_1 \equiv (\Delta_1', \Delta_1'')$, $\Delta_2 \equiv (\Delta_2', \Delta_2'')$, $\Delta_1' \xrightarrow{!a} \Delta_2'$ and $\Delta_1'' \xrightarrow{?a} \Delta_2''$. By induction, we have $(\Delta, \Delta_1') \xrightarrow{!a} (\Delta, \Delta_2')$ for any Δ. Using rule *lts−com* on this derivation and $\Delta_1'' \xrightarrow{?a} \Delta_2''$, we obtain $(\Delta, \Delta_1', \Delta_1'') \xrightarrow{\tau} (\Delta, \Delta_2', \Delta_2'')$, i.e. $(\Delta, \Delta_1) \xrightarrow{\tau} (\Delta, \Delta_2)$.

The second statement follows from the first by induction on the number of steps taken. □

Lemma 4.5 (partitioning). If $(\Delta_1, \Delta_2) \xrightarrow{\beta} \Delta^*$ then we must be in one of the following four cases:

1. $\Delta_1 \xrightarrow{\beta} \Delta_1'$ and $\Delta^* \equiv (\Delta_1', \Delta_2)$;
2. $\Delta_2 \xrightarrow{\beta} \Delta_2'$ and $\Delta^* \equiv (\Delta_1, \Delta_2')$;
3. $\Delta_1 \xrightarrow{?a} \Delta_1'$ and $\Delta_2 \xrightarrow{!a} \Delta_2'$ for some a, such that β is τ and $\Delta^* \equiv (\Delta_1', \Delta_2')$;
4. $\Delta_1 \xrightarrow{!a} \Delta_1'$ and $\Delta_2 \xrightarrow{?a} \Delta_2'$ for some a, such that β is τ and $\Delta^* \equiv (\Delta_1', \Delta_2')$.

Proof. There are three possibilities, depending on the forms of β.

— $\beta \equiv\ !a$ for some a. Then the last rule used to derive the transition $(\Delta_1, \Delta_2) \xrightarrow{\beta} \Delta^*$ must be *lts–atom* in Figure 4. So a is a member either in Δ_1 or in Δ_2. Correspondingly, we are in case 1 or 2.

— $\beta \equiv\ ?a$ for some a. Then the last rule used to derive the transition $(\Delta_1, \Delta_2) \xrightarrow{\beta} \Delta^*$ must be *lts*$\multimap$. So $a \multimap B$ is a member either in Δ_1 or in Δ_2. Correspondingly, we are in case 1 or 2.

— $\beta \equiv\ \tau$. If the last rule used to derive the transition $(\Delta_1, \Delta_2) \xrightarrow{\beta} \Delta^*$ is not *lts−com*, then the transition is given rise by a particular formula, and we are in case 1 or 2. If *lts–com* is the last rule used, there is an input action and an output action happening at the same time. Either both of them come from Δ_1, or both of them come from Δ_2, or one action from Δ_1 and the other from Δ_2. Consequently we are in one of the four cases above.

□

In the next lemma, we write $\xLeftarrow{\tau}$ for the reciprocal of $\xRightarrow{\tau}$.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

Lemma 4.6. $\stackrel{\tau}{\Longleftarrow}$ is a simulation (and, consequently, $\Delta_1 \stackrel{\tau}{\Longrightarrow} \Delta_2$ implies $\Delta_2 \preceq_s \Delta_1$).

Proof. We will show that the four conditions in Definition 4.2 are satisfied by $\stackrel{\tau}{\Longleftarrow}$, which proves that the relation is a simulation; the corollary follows immediately by virtue of $\preceq_s$ being the largest simulation. Suppose $\Delta_2 \stackrel{\tau}{\Longrightarrow} \Delta_1$.

1. If $\Delta_2 \equiv \cdot$, then obviously $\Delta_1 \stackrel{\tau}{\Longrightarrow} \Delta_2 \equiv \cdot$.
2. If $\Delta_2 \equiv \Delta_2', \Delta_2''$, then $\Delta_1 \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$. Taking zero steps we have $\Delta_2' \stackrel{\tau}{\Longleftarrow} \Delta_2'$ and $\Delta_2'' \stackrel{\tau}{\Longleftarrow} \Delta_2''$ as required.
3. If $\Delta_2 \stackrel{\alpha}{\longrightarrow} \Delta_2'$, then $\Delta_1 \stackrel{\tau}{\Longrightarrow} \Delta_2 \stackrel{\alpha}{\longrightarrow} \Delta_2'$ and therefore $\Delta_2 \stackrel{\alpha}{\Longrightarrow} \Delta_2'$. Taking zero steps we have $\Delta_2' \stackrel{\tau}{\Longleftarrow} \Delta_2'$ as required.
4. Now suppose $\Delta_2 \stackrel{?a}{\longrightarrow} \Delta_2'$. Then there are Δ_2'' and A such that $\Delta_2 \equiv (\Delta_2'', a \multimap A)$ and $\Delta_2' \equiv (\Delta_2'', A)$. From (Δ_1, a) we have the following matching transition:

$$\begin{aligned} (\Delta_1, a) &\stackrel{\tau}{\Longrightarrow} (\Delta_2, a) && \text{by compositionality (Lemma 4.4)} \\ &\equiv (\Delta_2'', a \multimap A, a) \\ &\stackrel{\tau}{\longrightarrow} (\Delta_2'', A) && \text{by rule } \mathit{lts-com} \text{ in Figure 4} \\ &\equiv \Delta_2' \end{aligned}$$

Taking zero steps, we have $\Delta_2' \stackrel{\tau}{\Longleftarrow} \Delta_2'$ as required.

□

4.4. *Properties of the simulation preorder*

It was relatively simple to prove that the barbed preorder was, in fact, a preorder. It is a bit more difficult to do so for the simulation preorder; our goal in this section is to prove Theorem 4.11, that the simulation preorder $\preceq_s$ is a proper preorder.

The structure of this section mirrors the structure of Section 3.3 (Properties of the barbed preorder). First we will prove a technical lemma that lets us act as if simulation was defined exclusively in terms of $\stackrel{\beta}{\Longrightarrow}$ rather than $\stackrel{\beta}{\longrightarrow}$, and then we prove that simulation is reflexive (Lemma 4.8), compositional (Lemma 4.9), and transitive (Lemma 4.10), from which Theorem 4.11 is an immediate result.

Lemma 4.7. If $\Delta_1 \preceq_s \Delta_2$ then

1. whenever $\Delta_1 \stackrel{\tau}{\Longrightarrow} \cdot$, then $\Delta_2 \stackrel{\tau}{\Longrightarrow} \cdot$;
2. whenever $\Delta_1 \stackrel{\tau}{\Longrightarrow} (\Delta_1', \Delta_1'')$, there exist Δ_2' and Δ_2'' such that $\Delta_2 \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$ and $\Delta_1' \preceq_s \Delta_2'$ and furthermore $\Delta_1'' \preceq_s \Delta_2''$;
3. whenever $\Delta_1 \stackrel{\alpha}{\Longrightarrow} \Delta_1'$, there exists Δ_2' such that $\Delta_2 \stackrel{\alpha}{\Longrightarrow} \Delta_2'$ and $\Delta_1' \preceq_s \Delta_2'$.

Proof. We first prove a particular case of the third statement:

$$\text{If } \Delta_1 \stackrel{\tau}{\Longrightarrow} \Delta_1' \text{ then there exists some } \Delta_2' \text{ such that } \Delta_2 \stackrel{\tau}{\Longrightarrow} \Delta_2' \text{ and } \Delta_1' \preceq_s \Delta_2'. \qquad (1)$$

We proceed by induction on the length of the transition $\Delta_1 \stackrel{\tau}{\Longrightarrow} \Delta_1'$.

— If $\Delta_1 \equiv \Delta_1'$, then $\Delta_2 \stackrel{\tau}{\Longrightarrow} \Delta_2$ by the reflexivity of $\stackrel{\tau}{\Longrightarrow}$, and $\Delta_1 \preceq_s \Delta_2$ by assumption.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

— Suppose $\Delta_1 \stackrel{\tau}{\Longrightarrow} \Delta_1'' \stackrel{\tau}{\longrightarrow} \Delta_1'$ for some Δ_1''. Since $\Delta_1 \preceq_s \Delta_2$, by the induction hypothesis there exists Δ_2'' such that $\Delta_2 \stackrel{\tau}{\Longrightarrow} \Delta_2''$ and $\Delta_1'' \preceq_s \Delta_2''$. The latter implies the existence of some Δ_2' such that $\Delta_2'' \stackrel{\tau}{\Longrightarrow} \Delta_2'$ and $\Delta_1' \preceq_s \Delta_2'$. The transitivity of $\stackrel{\tau}{\Longrightarrow}$ entails that $\Delta_2 \stackrel{\tau}{\Longrightarrow} \Delta_2'$.

We are now ready to prove the lemma.

1. Suppose $\Delta_1 \preceq_s \Delta_2$ and $\Delta_1 \stackrel{\tau}{\Longrightarrow} \cdot$. By (1), there exists Δ_2' such that $\Delta_2 \stackrel{\tau}{\Longrightarrow} \Delta_2'$ and $\cdot \preceq_s \Delta_2'$. By Definition 4.2, we have $\Delta_2' \stackrel{\tau}{\Longrightarrow} \cdot$. The transitivity of $\stackrel{\tau}{\Longrightarrow}$ entails $\Delta_2 \stackrel{\tau}{\Longrightarrow} \cdot$.
2. Suppose $\Delta_1 \preceq_s \Delta_2$ and $\Delta_1 \stackrel{\tau}{\Longrightarrow} (\Delta_1', \Delta_1'')$. By (1), there exists Δ_2''' such that $\Delta_2 \stackrel{\tau}{\Longrightarrow} \Delta_2'''$ and $(\Delta_1', \Delta_1'') \preceq_s \Delta_2'''$. By Definition 4.2, there exist Δ_2' and Δ_2'' such that $\Delta_2''' \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$, $\Delta_1' \preceq_s \Delta_2'$ and $\Delta_1'' \preceq_s \Delta_2''$. By the transitivity of $\stackrel{\tau}{\Longrightarrow}$, we have $\Delta_2 \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$.
3. Suppose $\Delta_1 \stackrel{\alpha}{\Longrightarrow} \Delta_1'$ where $\alpha \neq \tau$. Then there are Δ_{11} and Δ_{12} with $\Delta_1 \stackrel{\tau}{\Longrightarrow} \Delta_{11} \stackrel{\alpha}{\longrightarrow} \Delta_{12} \stackrel{\tau}{\Longrightarrow} \Delta_1'$. By (1), there is some Δ_{21} such that $\Delta_2 \stackrel{\tau}{\Longrightarrow} \Delta_{21}$ and $\Delta_{11} \preceq_s \Delta_{21}$. By Definition 4.2, there is Δ_{22} such that $\Delta_{21} \stackrel{\alpha}{\Longrightarrow} \Delta_{22}$ and $\Delta_{12} \preceq_s \Delta_{22}$. By (1) again, there is Δ_2' with $\Delta_{22} \stackrel{\tau}{\Longrightarrow} \Delta_2'$ with $\Delta_1' \preceq_s \Delta_2'$. Note that we also have $\Delta_2 \stackrel{\alpha}{\Longrightarrow} \Delta_2'$.

□

Lemma 4.8 (reflexivity of $\preceq_s$). For all contexts Δ, we have $\Delta \preceq_s \Delta$.

Proof. Consider the identity relation $\mathcal{R}_{id}$, which is the set of all pairs (Δ, Δ). It will suffice to show that $\mathcal{R}_{id} \subseteq \preceq_s$, because $\Delta \; \mathcal{R}_{id} \; \Delta$ always holds, which will in turn imply $\Delta \preceq_s \Delta$. We can show that $\mathcal{R}_{id}$ meets the four criteria for simulation. Let us pick up any pair (Δ, Δ) from $\mathcal{R}_{id}$.

1. Assume that $\Delta \equiv \cdot$. Then, in particular, $\Delta \stackrel{\tau}{\Longrightarrow} \cdot$.
2. Assume that $\Delta \equiv (\Delta', \Delta'')$. Then, in particular, $\Delta \stackrel{\tau}{\Longrightarrow} (\Delta', \Delta'')$, and we also have $\Delta' \; \mathcal{R}_{id} \; \Delta'$ and $\Delta'' \; \mathcal{R}_{id} \; \Delta''$.
3. Assume that $\Delta \stackrel{\alpha}{\longrightarrow} \Delta'$. Then, in particular, $\Delta \stackrel{\alpha}{\Longrightarrow} \Delta'$ and we have $\Delta' \; \mathcal{R}_{id} \; \Delta'$.
4. Assume that $\Delta \stackrel{?a}{\longrightarrow} \Delta'$, which entails that $\Delta \equiv (a \multimap B, \Delta'')$ and $\Delta' \equiv (B, \Delta'')$. Because $\Delta' \; \mathcal{R}_{id} \; \Delta'$ (it is the identity relation!), it is sufficient to show that $(\Delta, a) \stackrel{\tau}{\Longrightarrow} \Delta'$, i.e. that $(a, a \multimap B, \Delta'') \stackrel{\tau}{\Longrightarrow} (B, \Delta'')$. However, we know that $a \stackrel{!a}{\longrightarrow} \cdot$ and $(a \multimap B, \Delta'') \stackrel{?a}{\longrightarrow} (B, \Delta'')$ by the rules *lts−atom* and *lts* $\multimap$ in Figure 4, respectively. We can now combine them using rule *lts − com* in the desired reduction to obtain $(a, a \multimap B, \Delta'') \stackrel{\tau}{\longrightarrow} (B, \Delta'')$.

□

Proposition 4.9 (compositionality of $\preceq_s$). If $\Delta_1 \preceq_s \Delta_2$, then $(\Delta, \Delta_1) \preceq_s (\Delta, \Delta_2)$.

Proof. Consider the relation

$$\mathcal{R} \;\; := \;\; \{((\Delta, \Delta_1), (\Delta, \Delta_2)) \mid \Delta_1 \preceq_s \Delta_2\}.$$

It will suffice to show that $\mathcal{R} \subseteq \preceq_s$, because in that case, given $\Delta_1 \preceq_s \Delta_2$, we have that $(\Delta, \Delta_1) \; \mathcal{R} \; (\Delta, \Delta_2)$ for any Δ, which will in turn imply $(\Delta, \Delta_1) \preceq_s (\Delta, \Delta_2)$. We can show that $\mathcal{R}$ meets the four criteria for simulation. Suppose $(\Delta, \Delta_1) \; \mathcal{R} \; (\Delta, \Delta_2)$, which means that we also have $\Delta_1 \preceq_s \Delta_2$.

1. Let us show that if $(\Delta, \Delta_1)\ \mathcal{R}\ (\Delta, \Delta_2)$ with $(\Delta, \Delta_1) \equiv \cdot$, then $(\Delta, \Delta_2) \stackrel{\tau}{\Longrightarrow} \cdot$.
   If $(\Delta, \Delta_1) \equiv \cdot$, then $\Delta \equiv \cdot$. Moreover, by the definition of $\mathcal{R}$, we have that $\cdot \precsim_s \Delta_2$. Now, because $\precsim_s$ is a simulation, we have that $\Delta_2 \stackrel{\tau}{\Longrightarrow} \cdot$. Since $(\Delta, \Delta_2) \equiv \Delta_2$, we conclude that $(\Delta, \Delta_2) \stackrel{\tau}{\Longrightarrow} \cdot$, as desired.
2. Let us prove that if $(\Delta, \Delta_1) \equiv (\Delta_1', \Delta_1'')$, then $(\Delta, \Delta_2) \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$ such that $\Delta_1'\ \mathcal{R}\ \Delta_2'$ and $\Delta_1''\ \mathcal{R}\ \Delta_2''$.
   If (Δ, Δ_1) can be decomposed into (Δ_1', Δ_1'') for some Δ_1' and Δ_1'', we need to find some Δ_2' and Δ_2'' such that $(\Delta, \Delta_2) \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$ with $\Delta_1'\ \mathcal{R}\ \Delta_2'$ and $\Delta_1''\ \mathcal{R}\ \Delta_2''$. Without loss of generality, assume that we have the decomposition of (Δ, Δ_1) with $\Delta = (\Delta^a, \Delta^b)$ and $\Delta_1 = (\Delta_1^a, \Delta_1^b)$ such that $\Delta_1' = (\Delta^a, \Delta_1^a)$ and $\Delta_1'' = (\Delta^b, \Delta_1^b)$. Since $\Delta_1 \precsim_s \Delta_2$, there exists some transition $\Delta_2 \stackrel{\tau}{\Longrightarrow} (\Delta_2^a, \Delta_2^b)$ such that $\Delta_1^a \precsim_s \Delta_2^a$ and $\Delta_1^b \precsim_s \Delta_2^b$. It follows by compositionality (Lemma 4.4) that

   $$(\Delta, \Delta_2) \stackrel{\tau}{\Longrightarrow} (\Delta^a, \Delta^b, \Delta_2^a, \Delta_2^b) \equiv (\Delta^a, \Delta_2^a, \Delta^b, \Delta_2^b)$$

   Let $\Delta_2' = (\Delta^a, \Delta_2^a)$ and $\Delta_2'' = (\Delta^b, \Delta_2^b)$. We observe that $\Delta_1'\ \mathcal{R}\ \Delta_2'$ and $\Delta_1''\ \mathcal{R}\ \Delta_2''$, as required.
3. Let us show that if $(\Delta, \Delta_1) \stackrel{\alpha}{\longrightarrow} \Delta_1'$ then there is Δ_2' such that $(\Delta, \Delta_2) \stackrel{\alpha}{\Longrightarrow} \Delta_2'$ and $\Delta_1'\ \mathcal{R}\ \Delta_2'$, and if $(\Delta, \Delta_1) \stackrel{?a}{\longrightarrow} \Delta_1'$ then there is Δ_2' such that $(\Delta, \Delta_2, a) \stackrel{\tau}{\Longrightarrow} \Delta_2'$ and $\Delta_1'\ \mathcal{R}\ \Delta_2'$. It is convenient to prove both these parts of Definition 4.2 together.
   Assume that $(\Delta, \Delta_1) \stackrel{\beta}{\longrightarrow} \Delta^*$. There are four cases, according to Lemma 4.5.

   a. $(\Delta, \Delta_1) \stackrel{\beta}{\longrightarrow} (\Delta', \Delta_1)$ because of the transition $\Delta \stackrel{\beta}{\longrightarrow} \Delta'$. If $\beta = \alpha$, then by Lemma 4.4 (1) we also have $(\Delta, \Delta_2) \stackrel{\alpha}{\longrightarrow} (\Delta', \Delta_2)$ and clearly $(\Delta', \Delta_1)\ \mathcal{R}\ (\Delta', \Delta_2)$. If $\beta = ?a$, then $(\Delta, a) \stackrel{\tau}{\longrightarrow} \Delta'$ and thus $(\Delta, \Delta_2, a) \stackrel{\tau}{\longrightarrow} (\Delta', \Delta_2)$. Again, we have $(\Delta', \Delta_1)\ \mathcal{R}\ (\Delta', \Delta_2)$.

   b. If $(\Delta, \Delta_1) \stackrel{\beta}{\longrightarrow} (\Delta, \Delta_1')$ because of the transition $\Delta_1 \stackrel{\beta}{\longrightarrow} \Delta_1'$, since $\Delta_1 \precsim_s \Delta_2$ there are two possibilities. If $\beta = \alpha$, then there is a matching transition $\Delta_2 \stackrel{\alpha}{\Longrightarrow} \Delta_2'$ and $\Delta_1' \precsim_s \Delta_2'$. It follows that $(\Delta, \Delta_1') \stackrel{\alpha}{\Longrightarrow} (\Delta, \Delta_2')$ by Lemma 4.4 (2) and $(\Delta, \Delta_1')\ \mathcal{R}\ (\Delta, \Delta_2')$. If $\beta = ?a$, then $\Delta_2, a \stackrel{\tau}{\Longrightarrow} \Delta_2'$ for some Δ_2' with $\Delta_1' \precsim_s \Delta_2'$. We also have $(\Delta, \Delta_2, a) \stackrel{\tau}{\Longrightarrow} (\Delta, \Delta_2')$ by Lemma 4.4 (2) and $(\Delta, \Delta_1')\ \mathcal{R}\ (\Delta, \Delta_2')$.

   c. If $(\Delta, \Delta_1) \stackrel{\tau}{\longrightarrow} (\Delta', \Delta_1')$ because of the transitions $\Delta \stackrel{?a}{\longrightarrow} \Delta'$ and $\Delta_1 \stackrel{!a}{\longrightarrow} \Delta_1'$, since $\Delta_1 \precsim_s \Delta_2$ there is a transition $\Delta_2 \stackrel{!a}{\Longrightarrow} \Delta_2'$ with $\Delta_1' \precsim_s \Delta_2'$. It follows that $(\Delta, \Delta_2) \stackrel{\tau}{\Longrightarrow} (\Delta', \Delta_2')$ and we have $(\Delta', \Delta_1')\ \mathcal{R}\ (\Delta', \Delta_2')$.

   d. If $(\Delta, \Delta_1) \stackrel{\tau}{\longrightarrow} (\Delta', \Delta_1')$ because of the transitions $\Delta \stackrel{!a}{\longrightarrow} \Delta'$ and $\Delta_1 \stackrel{?a}{\longrightarrow} \Delta_1'$, then this can be simulated by a transition from (Δ, Δ_2). The reason is as follows. In order for Δ to enable the transition $\Delta \stackrel{!a}{\longrightarrow} \Delta'$, it must be the case that $\Delta \equiv \Delta', a$. Since $\Delta_1 \precsim_s \Delta_2$ we know that $(\Delta_2, a) \stackrel{\tau}{\Longrightarrow} \Delta_2'$ for some Δ_2' with $\Delta_1' \precsim_s \Delta_2'$. Therefore, we obtain that $(\Delta, \Delta_2) \equiv (\Delta', a, \Delta_2) \stackrel{\tau}{\Longrightarrow} (\Delta', \Delta_2')$ and $(\Delta', \Delta_1')\ \mathcal{R}\ (\Delta', \Delta_2')$.

In summary, we have verified that $\mathcal{R}$ is a simulation. □

Lemma 4.10 (transitivity of $\precsim_s$). If $\Delta_1 \precsim_s \Delta_2$ and $\Delta_2 \precsim_s \Delta_3$, then $\Delta_1 \precsim_s \Delta_3$.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

Proof. Consider the relation

$$\mathcal{R} \ := \ \{(\Delta_1, \Delta_3) \mid \text{there exists } \Delta_2 \text{ with } \Delta_1 \preceq_s \Delta_2 \text{ and } \Delta_2 \preceq_s \Delta_3\}$$

It will suffice to show that $\mathcal{R} \subseteq \preceq_s$, because in that case, given $\Delta_1 \preceq_s \Delta_2$ and $\Delta_2 \preceq_s \Delta_3$, we will know that $\Delta_1 \ \mathcal{R} \ \Delta_3$, which will in turn imply $\Delta_1 \preceq_s \Delta_3$. We can show that $\mathcal{R}$ meets the four criteria for simulation. Suppose $\Delta_1 \ \mathcal{R} \ \Delta_3$, that is $\Delta_1 \preceq_s \Delta_2$ and $\Delta_2 \preceq_s \Delta_3$ for some Δ_2.

1. If $\Delta_1 \equiv \cdot$, then $\Delta_2 \stackrel{\tau}{\Longrightarrow} \cdot$. By Lemma 4.7, we have $\Delta_3 \stackrel{\tau}{\Longrightarrow} \cdot$.
2. If $\Delta_1 \equiv (\Delta_1', \Delta_1'')$, then $\Delta_2 \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$ for some Δ_2' and Δ_2'' such that $\Delta_1' \preceq_s \Delta_2'$ and $\Delta_1'' \preceq_s \Delta_2''$. By Lemma 4.7, there exist Δ_3' and Δ_3'' such that $\Delta_3 \stackrel{\tau}{\Longrightarrow} (\Delta_3', \Delta_3'')$, $\Delta_2' \preceq_s \Delta_3'$ and $\Delta_2'' \preceq_s \Delta_3''$. Therefore, $\Delta_1' \ \mathcal{R} \ \Delta_3'$ and $\Delta_1'' \ \mathcal{R} \ \Delta_3''$.
3. If $\Delta_1 \stackrel{\alpha}{\longrightarrow} \Delta_1'$, there exists Δ_2' such that $\Delta_2 \stackrel{\alpha}{\Longrightarrow} \Delta_2'$ and $\Delta_1' \preceq_s \Delta_2'$. By Lemma 4.7, there exists Δ_3' such that $\Delta_3 \stackrel{\alpha}{\Longrightarrow} \Delta_3'$ and $\Delta_2' \preceq_s \Delta_3'$, thus $\Delta_1' \ \mathcal{R} \ \Delta_3'$.
4. If $\Delta_1 \stackrel{?a}{\longrightarrow} \Delta_1'$, there exists Δ_2' such that $(\Delta_2, a) \stackrel{\tau}{\Longrightarrow} \Delta_2'$ and $\Delta_1' \preceq_s \Delta_2'$. By Proposition 4.9 we have $(\Delta_2, a) \preceq_s (\Delta_3, a)$. By Lemma 4.7, there exists Δ_3' with $(\Delta_3, a) \stackrel{\tau}{\Longrightarrow} \Delta_3'$ and $\Delta_2' \preceq_s \Delta_3'$. It follows that $\Delta_1' \ \mathcal{R} \ \Delta_3'$.

□

Theorem 4.11. $\preceq_s$ is a preorder.

Proof. $\preceq_s$ is reflexive (Lemma 4.8) and transitive (Lemma 4.10). □

4.5. *Equivalence of the barbed and simulation preorders*

We are now in a position to fill in the rightmost portion of the proof map (Figure 4) from the beginning of this section. In this section, we show the equivalence of the largest simulation $\preceq_s$ and the barbed preorder. With the compositionality of $\preceq_s$ (Lemma 4.9) at hand, the soundness proof is straightforward. For the completeness proof, we crucially rely on fresh atom removal (Lemma 3.9).

Theorem 4.12 (soundness). If $\Delta_1 \preceq_s \Delta_2$, then $\Delta_1 \preceq_b \Delta_2$.

Proof. We show that all aspects of the definition of the barbed preorder are satisfied.

Barb-preserving. We show that $\preceq_s$ is barb-preserving. Suppose $\Delta_1 \preceq_s \Delta_2$ and $\Delta_1 \downarrow_a$. Then $\Delta_1 \equiv (\Delta_1', a)$ for some Δ_1', thus $\Delta_1 \stackrel{!a}{\longrightarrow} \Delta_1'$. Since $\Delta_1 \preceq_s \Delta_2$ there exists Δ_2' such that $\Delta_2 \stackrel{!a}{\Longrightarrow} \Delta_2'$, i.e. $\Delta_2 \stackrel{\tau}{\Longrightarrow} \Delta_2'' \stackrel{!a}{\longrightarrow} \Delta_2'$. Note that Δ_2'' must be the form (Δ_2''', a) for some Δ_2'''. It follows that $\Delta_2 \Downarrow_a$.

Compositional. By Lemma 4.9 $\preceq_s$ is compositional.

Reduction-closed. If $\Delta_1 \leadsto \Delta_1'$, then $\Delta_1 \stackrel{\tau}{\longrightarrow} \Delta_1'$ by Lemma 4.1. By the third condition of Definition 4.2 there exists Δ_2' such that $\Delta_2 \stackrel{\tau}{\Longrightarrow} \Delta_2'$ and $\Delta_1' \preceq_s \Delta_2'$. By Lemma 4.1 again, we have $\Delta_2 \leadsto^* \Delta_2'$.

Partition-preserving. If $\Delta_1 \equiv \cdot$, by the first condition of Definition 4.2 we see that $\Delta_2 \stackrel{\tau}{\Longrightarrow} \cdot$. By Lemma 4.1 this means $\Delta_2 \leadsto^* \cdot$.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

If $\Delta_1 \equiv (\Delta_1', \Delta_1'')$, by the second condition of Definition 4.2 there are Δ_2' and Δ_2'' with $\Delta_2 \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$, $\Delta_1' \preceq_s \Delta_2'$ and $\Delta_1'' \preceq_s \Delta_2''$. By Lemma 4.1 this means $\Delta_2 \rightsquigarrow^* (\Delta_2', \Delta_2'')$.

□

Theorem 4.13 (completeness). If $\Delta_1 \preceq_b \Delta_2$ then $\Delta_1 \preceq_s \Delta_2$.

Proof. We need to show that $\preceq_b$ is a simulation. By Definition 4.2, this decomposes into four parts.

1. Let us show that if $\Delta_1 \preceq_b \Delta_2$ and $\Delta_1 \equiv \cdot$ then $\Delta_2 \stackrel{\tau}{\Longrightarrow} \cdot$. Suppose $\Delta_1 \preceq_b \Delta_2$ and $\Delta_1 \equiv \cdot$, then $\Delta_2 \rightsquigarrow^* \cdot$ as $\preceq_b$ is reduction closed. By Lemma 4.1 we have $\Delta_2 \stackrel{\tau}{\Longrightarrow} \cdot$.
2. Let us show that if $\Delta_1 \preceq_b \Delta_2$ and $\Delta_1 \equiv (\Delta_1', \Delta_2')$ then $\Delta_2 \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$ for some Δ_2' and Δ_2'' such that $\Delta_1' \preceq_b \Delta_2'$ and $\Delta_1'' \preceq_b \Delta_2''$.
   Suppose $\Delta_1 \preceq_b \Delta_2$ and $\Delta_1 \equiv (\Delta_1', \Delta_1'')$, then $\Delta_2 \rightsquigarrow^* (\Delta_2', \Delta_2'')$ such that $\Delta_1' \preceq_b \Delta_2'$ and $\Delta_1'' \preceq_b \Delta_2''$ as $\preceq_b$ is reduction-closed. By Lemma 4.1 we have $\Delta_2 \stackrel{\tau}{\Longrightarrow} (\Delta_2', \Delta_2'')$.
3. Let us show that if $\Delta_1 \preceq_b \Delta_2$ and $\Delta_1 \stackrel{\alpha}{\longrightarrow} \Delta_1'$ then there exists Δ_2' such that $\Delta_2 \stackrel{\alpha}{\Longrightarrow} \Delta_2'$ and $\Delta_1' \preceq_b \Delta_2'$. Suppose $\Delta_1 \preceq_b \Delta_2$ and $\Delta_1 \stackrel{\alpha}{\longrightarrow} \Delta_1'$.
   — $\alpha \equiv \tau$. By Lemma 4.1 this means $\Delta_1 \rightsquigarrow \Delta_1'$. Since $\preceq_b$ is reduction-closed, there exists Δ_2' such that $\Delta_2 \rightsquigarrow^* \Delta_2'$ and $\Delta_1' \preceq_b \Delta_2'$. By Lemma 4.1 again, we have $\Delta_2 \stackrel{\tau}{\Longrightarrow} \Delta_2'$.
   — $\alpha \equiv\, !a$ for some a. Note that Δ_1 must be in the form (Δ_1', a). Since $\preceq_b$ is partition-preserving, there exist Δ_2' and Δ_a such that
   $$\Delta_2 \rightsquigarrow^* (\Delta_2', \Delta_a) \tag{2}$$
   with $\Delta_1' \preceq_b \Delta_2'$ and $a \preceq_b \Delta_a$. Then $(a \multimap \mathbf{1}, a) \preceq_b (a \multimap \mathbf{1}, \Delta_a)$ by the compositionality of $\preceq_b$. Since $(a \multimap \mathbf{1}, a) \rightsquigarrow^* \cdot$, by Lemma 3.6 we have $(a \multimap \mathbf{1}, \Delta_a) \rightsquigarrow^* \cdot$. Then there exists some Δ_a' such that $\Delta_a \rightsquigarrow^* (\Delta_a', a)$ and $\Delta_a' \rightsquigarrow^* \cdot$, thus
   $$\Delta_a \rightsquigarrow^* a \tag{3}$$
   by transitivity. It follows from (2) and (3) that $\Delta_2 \rightsquigarrow^* (\Delta_2', a)$. By Lemma 4.1 this means $\Delta_2 \stackrel{!a}{\Longrightarrow} \Delta_2'$, which is the desired transition.
4. Let us show that if $\Delta_1 \preceq_b \Delta_2$ and $\Delta_1 \stackrel{?a}{\longrightarrow} \Delta_1'$ then there exists Δ_2' such that $(\Delta_2, a) \stackrel{\tau}{\Longrightarrow} \Delta_2'$ and $\Delta_1' \preceq_b \Delta_2'$.
   Suppose $\Delta_1 \preceq_b \Delta_2$ and $\Delta_1 \stackrel{?a}{\longrightarrow} \Delta_1'$. Then $(\Delta_1, a) \stackrel{\tau}{\longrightarrow} \Delta_1'$, and thus $(\Delta_1, a) \rightsquigarrow \Delta_1'$ by Lemma 4.1. Since $\Delta_1 \preceq_b \Delta_2$ we know $(\Delta_1, a) \preceq_b (\Delta_2, a)$ by the compositionality of $\preceq_b$. So there exists some Δ_2' such that $(\Delta_2, a) \rightsquigarrow^* \Delta_2'$ and $\Delta_1' \preceq_b \Delta_2'$. By Lemma 4.1 we also have $(\Delta_2, a) \stackrel{\tau}{\Longrightarrow} \Delta_2'$.

□

4.6. *Equivalence of the logical and simulation preorders*

We will now start to fill in the remaining portions of the proof map (Figure 4) from the beginning of this section. First, we prove the soundness and completeness of *derivability* relative to simulation, and then we use this to prove the soundness and completeness of the barbed preorder relative to the logical preorder.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

Theorem 4.14. If $\Delta \vdash A$, then $A \preceq_s \Delta$.

Proof. We proceed by rule induction, where the rules are given in Figure 1.

— (rule $\top R$). If $\Delta \vdash \top$ then we have $\top \preceq_s \Delta$ vacuously, as $\top$ is a nonempty process state that can make no transitions.

— (rule $\mathbf{1}R$). If $\cdot \vdash \mathbf{1}$ then it is trivial to see that $\mathbf{1} \preceq_s \cdot$.

— (rule *init*). If $a \vdash a$ then $a \preceq_s a$ follows from the reflexivity of $\preceq_s$.

— (rule $\multimap R$). Suppose $\Delta \vdash a \multimap A$ is derived from $\Delta, a \vdash A$. By induction, we have

$$A \preceq_s (\Delta, a). \tag{4}$$

The only transition from $a \multimap A$ is $(a \multimap A) \xrightarrow{?a} A$. It is matched by the trivial transition $(\Delta, a) \overset{\tau}{\Longrightarrow} (\Delta, a)$ in view of (4).

— (rule $\multimap L$). Suppose $(\Delta_1, \Delta_2, a \multimap A) \vdash B$ is derived from $\Delta_1 \vdash a$ and $(\Delta_2, A) \vdash B$. By induction, we have

$$a \preceq_s \Delta_1 \quad \text{and} \quad B \preceq_s (\Delta_2, A). \tag{5}$$

By the first part of (5) we know that there is some Δ_1' such that $\Delta_1 \overset{!a}{\Longrightarrow} \Delta_1'$ and $\cdot \preceq_s \Delta_1'$. It is easy to see that $\Delta_1 \overset{\tau}{\Longrightarrow} (\Delta_1', a)$ and $\Delta_1' \overset{\tau}{\Longrightarrow} \cdot$. Then

$$\begin{aligned} (\Delta_1, \Delta_2, a \multimap A) &\overset{\tau}{\Longrightarrow} (\Delta_1', a, \Delta_2, a \multimap A) \\ &\xrightarrow{\tau} (\Delta_1', \Delta_2, A) \\ &\overset{\tau}{\Longrightarrow} (\Delta_2, A). \end{aligned}$$

In other words, we have $(\Delta_1, \Delta_2, a \multimap A) \overset{\tau}{\Longrightarrow} (\Delta_2, A)$. By Lemma 4.6 it follows that $(\Delta_2, A) \preceq_s (\Delta_1, \Delta_2, a \multimap A)$. By transitivity ($\preceq_s$ is a preorder, Theorem 4.11), we can combine this with the second part of (5), yielding

$$B \;\preceq_s\; (\Delta_1, \Delta_2, a \multimap A).$$

— (rule $\otimes R$). Suppose $(\Delta_1, \Delta_2) \vdash A \otimes B$ is derived from $\Delta_1 \vdash A$ and $\Delta_2 \vdash B$. By induction, we have

$$A \preceq_s \Delta_1 \quad \text{and} \quad B \preceq_s \Delta_2. \tag{6}$$

Now, the only transition from $A \otimes B$ is $A \otimes B \xrightarrow{\tau} (A, B)$. It can be matched by the trivial transition $(\Delta_1, \Delta_2) \overset{\tau}{\Longrightarrow} (\Delta_1, \Delta_2)$ because by compositionality of $\preceq_s$ (Lemma 4.9 and (6) we know that

$$(A, B) \preceq_s (\Delta_1, B) \preceq_s (\Delta_1, \Delta_2).$$

Now it is immediate that $(A, B) \preceq_s (\Delta_1, \Delta_2)$ by transitivity ($\preceq_s$ is a preorder, Theorem 4.11).

— (rule $\otimes L$). Suppose $(\Delta, A \otimes B) \vdash C$ is derived from $(\Delta, A, B) \vdash C$. By induction we have

$$C \preceq_s (\Delta, A, B). \tag{7}$$

Since $(\Delta, A \otimes B) \xrightarrow{\tau} (\Delta, A, B)$, we apply Lemma 4.6 and obtain

$$(\Delta, A, B) \;\preceq_s\; (\Delta, A \otimes B). \tag{8}$$

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

By (7), (8) and the transitivity of $\preceq_s$, we have $C \preceq_s (\Delta, A \otimes B)$.

— (rules $\mathbf{1}L$, $\& L_1$ and $\& L_2$). Similar.

— (rule $\& R$). Suppose $\Delta \vdash (A \& B)$ is derived from $\Delta \vdash A$ and $\Delta \vdash B$. By induction we have

$$A \preceq_s \Delta \quad \text{and} \quad B \preceq_s \Delta. \tag{9}$$

The only transitions from $A \& B$ are $(A \& B) \xrightarrow{\tau} A$ and $(A \& B) \xrightarrow{\tau} B$. Both of them can be matched by the trivial transition $\Delta \stackrel{\tau}{\Longrightarrow} \Delta$ in view of (9).

□

Proposition 4.15. If $\Delta_1 \stackrel{\tau}{\Longrightarrow} \Delta_2$ and $\Delta_2 \vdash A$, then $\Delta_1 \vdash A$.

Proof. Immediate by Proposition 3.1 and Lemma 4.1. □

Theorem 4.16. If $A \preceq_s \Delta$, then $\Delta \vdash A$.

Proof. We proceed by induction on the structure of A.

— $A \equiv \top$. By rule $\top R$ we have $\Delta \vdash \top$.

— $A \equiv \mathbf{1}$. Then we also have $A \equiv \cdot$. Since $A \preceq_s \Delta$, it must be the case that $\Delta \stackrel{\tau}{\Longrightarrow} \cdot$. By rule $\mathbf{1}R$, we have $\cdot \vdash \mathbf{1}$. By Proposition 4.15 it follows that $\Delta \vdash \mathbf{1}$.

— $A \equiv a$. Since $A \preceq_s \Delta$ and $A \xrightarrow{!a} \cdot$, there is Δ' such that

$$\Delta \stackrel{!a}{\Longrightarrow} \Delta' \qquad \text{and} \qquad \cdot \preceq_s \Delta'. \tag{10}$$

From the first part of (10), we obtain $\Delta \stackrel{\tau}{\Longrightarrow} (\Delta'', a)$ for some Δ'' with $\Delta'' \stackrel{\tau}{\Longrightarrow} \Delta'$. From the second part, we have $\Delta' \stackrel{\tau}{\Longrightarrow} \cdot$. Combining them together yields $\Delta \stackrel{\tau}{\Longrightarrow} a$. By rule *init* we can infer $a \vdash a$. Then it follows from Proposition 4.15 that $\Delta \vdash a$.

— $A \equiv a \multimap A'$. Since $A \preceq_s \Delta$ and $A \xrightarrow{?a} A'$, there is Δ' such that $(\Delta, a) \stackrel{\tau}{\Longrightarrow} \Delta'$ and $A' \preceq_s \Delta'$. By induction, we know that $\Delta' \vdash A'$. By Proposition 4.15 it follows that $(\Delta, a) \vdash A'$. Now use rule $\multimap R$ we obtain $\Delta \vdash (a \multimap A')$.

— $A \equiv A_1 \& A_2$. Since $A \preceq_s \Delta$ and $A \xrightarrow{\tau} A_1$, there is Δ_1 such that $\Delta \stackrel{\tau}{\Longrightarrow} \Delta_1$ and $A_1 \preceq_s \Delta_1$. By induction, we have $\Delta_1 \vdash A_1$. By Proposition 4.15 it follows that $\Delta \vdash A_1$. By a similar argument, we see that $\Delta \vdash A_2$. Hence, it follows from rule $\& R$ that $\Delta \vdash (A_1 \& A_2)$.

— $A \equiv A_1 \otimes A_2$. Since $A \preceq_s \Delta$ and $A \xrightarrow{\tau} (A_1, A_2)$, we apply Lemma 4.7 and derive some transition $\Delta \stackrel{\tau}{\Longrightarrow} (\Delta_1, \Delta_2)$ such that $A_1 \preceq_s \Delta_1$ and $A_2 \preceq_s \Delta_2$. By induction, we obtain $\Delta_1 \vdash A_1$ and $\Delta_2 \vdash A_2$. It follows from rule $\otimes R$ that $\Delta_1, \Delta_2 \vdash A_1 \otimes A_2$, that is $\Delta \vdash A$.

□

Remark 4.17. Note that Theorem 4.16 would fail if we used the standard barbed preorder on processes, without adding the condition partition-preservation in Definition 3.2. In that case, the process state $(a \multimap \mathbf{1}) \otimes (b \multimap \mathbf{1})$ would be related to $a \multimap b \multimap \mathbf{1}$ by barbed preorder but

$$a \multimap b \multimap \mathbf{1} \;\nvdash\; (a \multimap \mathbf{1}) \otimes (b \multimap \mathbf{1}).$$

The next property is obtained mostly by applying Theorems 4.14 and 4.16.

Theorem 4.18. $\Delta_1 \preceq_l \Delta_2$ if and only if $\Delta_1 \preceq_s \Delta_2$.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

Proof.

($\Rightarrow$) Suppose $\Delta_1 \preceq_l \Delta_2$. It is trivial to see that $\Delta_1 \vdash \bigotimes \Delta_1$. By the definition of logical preorder, it follows that $\Delta_2 \vdash \bigotimes \Delta_1$. By Theorem 4.14 we have

$$\bigotimes \Delta_1 \ \preceq_s \ \Delta_2. \tag{11}$$

Considering the formula $\bigotimes \Delta_1$ as a context, we have $\bigotimes \Delta_1 \stackrel{\tau}{\Longrightarrow} \Delta_1$ according to our reduction semantics. By Lemma 4.6, it follows that

$$\Delta_1 \ \preceq_s \ \bigotimes \Delta_1. \tag{12}$$

By combining (11) and (12), we obtain that $\Delta_1 \preceq_s \Delta_2$ because $\preceq_s$ is transitive by Theorem 4.11.

($\Leftarrow$) Suppose that $\Delta_1 \preceq_s \Delta_2$. For any Δ and A, assume that $(\Delta, \Delta_1) \vdash A$. By Theorem 4.14 we have

$$A \preceq_s (\Delta, \Delta_1). \tag{13}$$

Since $\Delta_1 \preceq_s \Delta_2$ and $\preceq_s$ is compositional (Lemma 4.9), we obtain

$$(\Delta, \Delta_1) \ \preceq_s \ (\Delta, \Delta_2). \tag{14}$$

By (13), (14) and the transitivity of $\preceq_s$, we see that $A \preceq_s (\Delta, \Delta_2)$. Then, Theorem 4.16 yields $(\Delta, \Delta_2) \vdash A$. Therefore, we have shown that $\Delta_1 \preceq_l \Delta_2$.

This concludes the proof of this result. □

Finally, we arrive at the main result of the section.

Corollary 4.19 (soundness and completeness). $\Delta_1 \preceq_l \Delta_2$ if and only if $\Delta_1 \preceq_b \Delta_2$.

Proof. By Theorems 4.12 and 4.13 we know that $\preceq_b$ coincides with $\preceq_s$. Theorem 4.18 tells us that $\preceq_s$ coincides with $\preceq_l$. Hence, the required result follows. □

5. Exponentials

In this section, we extend the investigation by adding the exponential modality '!' from intuitionistic linear logic, which will closely correspond to the replication operator of the π-calculus. Our extension refers to the propositional linear language seen in Sections 3–4. Specifically, the language we will be working on is:

$$\textit{Formulas} \qquad A, B, C ::= a \mid \mathbf{1} \mid A \otimes B \mid a \multimap B \mid \top \mid A \& B \mid !A.$$

Observe that this language still limits the antecedent of linear implications to be an atomic proposition – this is a common restriction when investigating fragments of linear logic that correspond to CCS-like process algebras (Cervesato *et al.* 2000, 2002; Cervesato and Scedrov 2009)

The structure of this section is similar to our development for the language without exponentials: we first present the extended language and a notion of states in our process interpretation in Section 5.1, then we connect barbed preorder with logical preorder by making use of simulation in Section 5.2.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

5.1. *Process interpretation and barbed preorder*

The π-calculus reading of the language with exponentials is extended by interpreting the exponential ! as the replication operator !.

......

$!A$any number of copies of process A.

The structural equivalences seen in Section 3 are updated by adding an inert unrestricted context Γ and the following rules:

$$(\Gamma, \cdot;\ \Delta) \equiv (\Gamma;\ \Delta),$$
$$(\Gamma_1, \Gamma_2;\ \Delta) \equiv (\Gamma_2, \Gamma_1;\ \Delta),$$
$$(\Gamma_1, (\Gamma_2, \Gamma_3);\ \Delta) \equiv ((\Gamma_1, \Gamma_2), \Gamma_3;\ \Delta),$$
$$(\Gamma, A, A;\ \Delta) \equiv (\Gamma, A;\ \Delta).$$

These rule entail that the unrestricted context behaves like a set.

The reductions in Figure 2 are upgraded with an inert context Γ, and the following two reductions are added:

$$(\Gamma, A;\ \Delta) \rightsquigarrow (\Gamma, A;\ \Delta, A) \qquad (\rightsquigarrow \text{clone})$$
$$(\Gamma;\ \Delta, !A) \rightsquigarrow (\Gamma, A;\ \Delta) \qquad (\rightsquigarrow !)$$

The two rules are intended to reflect the derivations entailed by rules *clone* and $!L$ in Figure 1. Instead of introducing the two reduction rules, we could turn them into two equations and add to the structural equivalence $\equiv$ defined above. But we prefer to mimic one step of derivation in linear logic by one step of reduction in our process interpretation of the logic.

The *composition* of two states $(\Gamma_1; \Delta_1)$ and $(\Gamma_2; \Delta_2)$, written $((\Gamma_1; \Delta_1), (\Gamma_2; \Delta_2))$, is defined as the state $((\Gamma_1, \Gamma_2); (\Delta_1, \Delta_2))$. Recall that unrestricted contexts are set so that Γ_1, Γ_2 may collapse identical formulas occurring in both Γ_1 and Γ_2 (while linear contexts can contain duplicates).

A *partition* of a state $(\Gamma; \Delta)$ is any pair of states $(\Gamma_1; \Delta_1)$ and $(\Gamma_2; \Delta_2)$ such that $(\Gamma; \Delta) = ((\Gamma_1; \Delta_1), (\Gamma_2; \Delta_2))$.

We write $(\Gamma; \Delta) \downarrow_a$ whenever $a \in \Delta$, and $\Delta \Downarrow_a$ whenever $(\Gamma; \Delta) \rightsquigarrow^* (\Gamma'; \Delta')$ for some $(\Gamma'; \Delta')$ with $(\Gamma'; \Delta') \downarrow_a$. The definition of barbed preorder given in Definition 3.3 now takes the following form.

Definition 5.1 (barbed preorder). Let $\mathcal{R}$ be a binary relation over states. We say that $\mathcal{R}$ is

— *barb-preserving* if, whenever $(\Gamma_1; \Delta_1)\ \mathcal{R}\ (\Gamma_2; \Delta_2)$ and $(\Gamma_1; \Delta_1) \downarrow_a$, we have that $(\Gamma_2; \Delta_2) \Downarrow_a$ for any a.

— *reduction-closed* if $(\Gamma_1; \Delta_1)\ \mathcal{R}\ (\Gamma_2; \Delta_2)$ and $(\Gamma_1; \Delta_1) \rightsquigarrow (\Gamma'_1; \Delta'_1)$ implies $(\Gamma_2; \Delta_2) \rightsquigarrow^* (\Gamma'_2; \Delta'_2)$ and $(\Gamma'_1; \Delta'_1)\ \mathcal{R}\ (\Gamma'_2; \Delta'_2)$ for some $(\Gamma'_2; \Delta'_2)$.

— *compositional* if $(\Gamma_1; \Delta_1)\ \mathcal{R}\ (\Gamma_2; \Delta_2)$ implies $((\Gamma_1; \Delta_1), (\Gamma; \Delta))\ \mathcal{R}\ ((\Gamma_2; \Delta_2), (\Gamma; \Delta))$ for all $(\Gamma; \Delta)$.

— *partition-preserving* if $(\Gamma_1; \Delta_1)\ \mathcal{R}\ (\Gamma_2; \Delta_2)$ implies that

1. if $\Delta_1 = \cdot$, then $(\Gamma_2; \Delta_2) \rightsquigarrow^* (\Gamma'_2; \cdot)$ and $(\Gamma_1; \cdot)\ \mathcal{R}\ (\Gamma'_2; \cdot)$,

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

$$\frac{}{(\Gamma;\Delta,a) \xrightarrow{!a} (\Gamma;\Delta)}\ lts{-}atom \qquad \frac{}{(\Gamma;\Delta,a \multimap B) \xrightarrow{?a} (\Gamma;\Delta,B)}\ lts \multimap$$

$$\frac{}{(\Gamma;\Delta,\mathbf{1}) \xrightarrow{\tau} (\Gamma;\Delta)}\ lts\mathbf{1} \qquad \frac{}{(\Gamma;\Delta,A\otimes B) \xrightarrow{\tau} (\Gamma;\Delta,A,B)}\ lts\otimes$$

$$\frac{}{(\Gamma;\Delta,A\,\&\,B) \xrightarrow{\tau} (\Gamma;\Delta,A)}\ lts\,\&_1 \qquad \frac{}{(\Gamma;\Delta,A\,\&\,B) \xrightarrow{\tau} (\Gamma;\Delta,B)}\ lts\,\&_2$$

$$\frac{}{(\Gamma;\Delta,!A) \xrightarrow{\tau} (\Gamma,A;\Delta)}\ lts!A \qquad \text{(No rule for } \top\text{)}$$

$$\frac{}{(\Gamma,A;\Delta) \xrightarrow{\tau} (\Gamma,A;\Delta,A)}\ ltsClone$$

$$\frac{(\Gamma_1;\Delta_1) \xrightarrow{!a} (\Gamma_1';\Delta_1') \quad (\Gamma_2;\Delta_2) \xrightarrow{?a} (\Gamma_2';\Delta_2')}{(\Gamma_1,\Gamma_2;\Delta_1,\Delta_2) \xrightarrow{\tau} (\Gamma_1',\Gamma_2';\Delta_1',\Delta_2')}\ lts{-}com$$

Fig. 5. Labelled transition system with exponentials.

2. for all $(\Gamma_1';\Delta_1')$ and $(\Gamma_1'';\Delta_1'')$, if $(\Gamma_1;\Delta_1) = ((\Gamma_1';\Delta_1'),(\Gamma_1'';\Delta_1''))$ then there exists $(\Gamma_2';\Delta_2')$ and $(\Gamma_2'';\Delta_2'')$ such that $(\Gamma_2;\Delta_2) \rightsquigarrow^* ((\Gamma_2';\Delta_2'),(\Gamma_2'';\Delta_2''))$ and furthermore $(\Gamma_1';\Delta_1')\ \mathcal{R}\ (\Gamma_2';\Delta_2')$ and $(\Gamma_1'';\Delta_1'')\ \mathcal{R}\ (\Gamma_2'';\Delta_2'')$.

The *barbed preorder*, denoted by $\preceq_b$, is the largest relation over processes which is barb-preserving, reduction-closed, compositional and partition-preserving.

Observe that this definition is structurally identical to our original notion of barbed preorder (Definitions 3.2 and 3.3). Indeed, we have simply expressed the notions of composing and partitioning states as explicit operations while our original definition relied on context composition, which is what these notion specialize to when we only have linear contexts. The present definition appears to be quite robust and we have used it in extensions of this work to larger languages.

5.2. *Logical preorder and barbed preorder*

The labelled transition semantics for our language with replicated formulas is given in Figure 5. The following is an updated version of Lemma 4.1.

Lemma 5.2. $(\Gamma_1,\Delta_1) \stackrel{\tau}{\Longrightarrow} (\Gamma_2,\Delta_2)$ if and only if $(\Gamma_1,\Delta_1) \rightsquigarrow^* (\Gamma_2,\Delta_2)$.

In the new semantics, our definition of simulation is in the following form.

Definition 5.3 (simulation). A relation $\mathcal{R}$ between two processes represented as $(\Gamma_1;\Delta_1)$ and $(\Gamma_2;\Delta_2)$ is a *simulation* if $(\Gamma_1;\Delta_1)\ \mathcal{R}\ (\Gamma_2;\Delta_2)$ implies

1. if $(\Gamma_1;\Delta_1) \equiv (\Gamma_1';\cdot)$ then $(\Gamma_2;\Delta_2) \stackrel{\tau}{\Longrightarrow} (\Gamma_2';\cdot)$ and $(\Gamma_1';\cdot)\ \mathcal{R}\ (\Gamma_2';\cdot)$;
2. if $(\Gamma_1;\Delta_1) \equiv ((\Gamma_1';\Delta_1'),(\Gamma_1'';\Delta_1''))$ then $(\Gamma_2;\Delta_2) \stackrel{\tau}{\Longrightarrow} ((\Gamma_2';\Delta_2'),(\Gamma_2'';\Delta_2''))$ for some $(\Gamma_2';\Delta_2')$ and $(\Gamma_2'';\Delta_2'')$ such that $(\Gamma_1';\Delta_1')\ \mathcal{R}\ (\Gamma_2';\Delta_2')$ and $(\Gamma_1'';\Delta_1'')\ \mathcal{R}\ (\Gamma_2'';\Delta_2'')$;

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

3. whenever $(\Gamma_1;\Delta_1) \xrightarrow{\alpha} (\Gamma_1';\Delta_1')$, there exists $(\Gamma_2';\Delta_2')$ such that $(\Gamma_2;\Delta_2) \overset{\alpha}{\Longrightarrow} (\Gamma_2';\Delta_2')$ and $(\Gamma_1';\Delta_1') \mathcal{R} (\Gamma_2';\Delta_2')$;
4. whenever $(\Gamma_1;\Delta_1) \xrightarrow{?a} (\Gamma_1';\Delta_1')$, there exists $(\Gamma_2';\Delta_2')$ such that $(\Gamma_2;\ \Delta_2, a) \overset{\tau}{\Longrightarrow} (\Gamma_2';\Delta_2')$ and $(\Gamma_1';\Delta_1') \mathcal{R} (\Gamma_2';\Delta_2')$.

We write $(\Gamma_1;\Delta_1) \preceq_s (\Gamma_2;\Delta_2)$ if there is some simulation $\mathcal{R}$ with $(\Gamma_1;\Delta_1) \mathcal{R} (\Gamma_2;\Delta_2)$.

Example 5.4. We have mentioned before that $!A$ intuitively represents any number of copies of A. Then it is natural to identify $!!A$ and $!A$, as in some presentations of the π-calculus. For instance, we have that

$$(\cdot;\,!!a) \preceq_s (\cdot;\,!a) \qquad \text{and} \qquad (\cdot;\,!a) \preceq_s (\cdot;\,!!a). \tag{15}$$

To prove the first inequality, consider the two sets

$$S_1 = \{(\cdot;\,!!a)\} \cup \{(!a;(!a)^n) \mid n \geqslant 0\} \cup \{(!a, a;\ (!a)^n, a^m) \mid n \geqslant 0,\ m \geqslant 0\}$$
$$S_2 = \{(\cdot;\,!a)\} \cup \{a; a^n) \mid n \geqslant 0\}$$

where we write A^0 for '$\cdot$' and A^n for n copies of A where $n > 0$. Let $\mathcal{R}$ be $S_1 \times S_2$, the Cartesian product of S_1 and S_2. It can be checked that $\mathcal{R}$ is a simulation relation. In the same way, one can see that $S_2 \times S_1$ is also a simulation relation, which implies the second inequality in (15).

By adapting the proof of Proposition 4.9 to the case with exponentials, we have the compositionality of $\preceq_s$.

Proposition 5.5. If $(\Gamma_1;\Delta_1) \preceq_s (\Gamma_2, \Delta_2)$ then $(\Gamma_1, \Gamma;\Delta_1, \Delta) \preceq_s (\Gamma_2, \Gamma;\Delta_2, \Delta)$ for any process state (Γ, Δ).

Similar to Theorems 4.12 and 4.13, it can be shown that the following coincidence result holds.

Theorem 5.6. $(\Gamma_1;\Delta_1) \preceq_b (\Gamma_2;\Delta_2)$ if and only if $(\Gamma_1;\Delta_1) \preceq_s (\Gamma_2;\Delta_2)$.

The rest of this subsection is devoted to showing the coincidence of $\preceq_l$ and $\preceq_s$, by following the schema in Section 4.6. We first need two technical lemmas whose proofs are simple and thus omitted.

Lemma 5.7 (weakening). $(\Gamma;\Delta) \preceq_s ((\Gamma, \Gamma');\Delta)$ for any Γ'.

Lemma 5.8. If $(\Gamma_1;\Delta_1) \overset{\tau}{\Longrightarrow} (\Gamma_2;\Delta_2)$ then $(\Gamma_2;\Delta_2) \preceq_s (\Gamma_1;\Delta_1)$.

We are now in a position to connect simulation with provability. First, we state a result akin to Theorem 4.14. If $\Gamma;\Delta \vdash A$, then the process $(\Gamma;A)$ can be simulated by the process $(\Gamma;\Delta)$. Note that the same Γ is used in both processes, which makes it easy to verify the soundness of the rule !R with respect to simulation relation.

Theorem 5.9. If $\Gamma;\Delta \vdash A$ then $(\Gamma;A) \preceq_s (\Gamma;\Delta)$.

Proof. As in Theorem 4.14, we proceed by rule induction. Here we consider the three new rules.

— (rule clone). Suppose $\Gamma, B; \Delta \vdash A$ is derived from $\Gamma, B; \Delta, B \vdash A$. By induction, we have

$$(\Gamma, B;\ A) \preceq_s (\Gamma, B;\ \Delta, B). \tag{16}$$

From $(\Gamma, B; \Delta)$ we have the transition $(\Gamma, B; \Delta) \stackrel{\tau}{\longrightarrow} (\Gamma, B;\ \Delta, B)$. By Lemma 5.8, we know that

$$(\Gamma, B;\ \Delta, B) \preceq_s (\Gamma, B; \Delta). \tag{17}$$

Combining (16), (17) and the transitivity of similarity, we obtain $(\Gamma, B;\ A) \preceq_s (\Gamma, B;\ \Delta)$.

— (rule !L). Suppose $\Gamma; \Delta, !B \vdash A$ is derived from $\Gamma, B; \Delta \vdash A$. By induction, we have

$$(\Gamma, B;\ A) \preceq_s (\Gamma, B;\ \Delta). \tag{18}$$

From $(\Gamma;\ \Delta, !B)$ we have the transition $(\Gamma;\ \Delta, !B) \stackrel{\tau}{\longrightarrow} (\Gamma, B;\ \Delta)$. By Lemma 5.8, we know that

$$(\Gamma, B;\ \Delta) \preceq_s (\Gamma;\ \Delta, !B). \tag{19}$$

By Lemma 5.7 we have

$$(\Gamma;\ A) \preceq_s (\Gamma, B;\ A). \tag{20}$$

Combining (18)–(20), and the transitivity of similarity, we obtain $(\Gamma;\ A) \preceq_s (\Gamma;\ \Delta, !B)$.

— (rule !R) Suppose $\Gamma; \cdot \vdash !A$ is derived from $\Gamma; \cdot \vdash A$. By induction we have

$$(\Gamma; A) \preceq_s (\Gamma; \cdot). \tag{21}$$

We now construct a relation $\mathcal{R}$ based on (21).

$$\begin{aligned}\mathcal{R} = &\{((\Gamma; \Delta, !A),\ (\Gamma; \Delta)) \mid \text{for any } \Delta\}\\ &\cup\{((\Gamma, A; \Delta),\ (\Gamma'; \Delta')) \mid \text{for any } \Delta, \Delta' \text{ and } \Gamma' \text{with } (\Gamma; \Delta) \preceq_s (\Gamma'; \Delta')\}\\ &\cup \preceq_s\end{aligned}$$

The relation is composed of three sets. The pairs in the first set come directly from (21) extended by some linear contexts Δ. After performing some matching transitions, the pairs in the first set may evolve into those in the second or the third set. So the last two sets are included to make $\mathcal{R}$ a closed set with respect to simulation relation. Below we show that $\mathcal{R}$ is indeed a simulation, thus $\mathcal{R} \subseteq \preceq_s$. Since $(\Gamma;\ !A)\ \mathcal{R}\ (\Gamma; \cdot)$, it follows that $(\Gamma;\ !A) \preceq_s (\Gamma; \cdot)$.

Let us pick any pair of states from $\mathcal{R}$. It suffices to consider the elements from the first two subsets of $\mathcal{R}$:

- The two states are $(\Gamma;\ \Delta, !A)$ and $(\Gamma; \Delta)$ respectively. Let us consider any transition from the first state.
  - The transition is $(\Gamma; \Delta, !A) \stackrel{\tau}{\longrightarrow} (\Gamma, A; \Delta)$. This is matched by the trivial transition $(\Gamma; \Delta) \stackrel{\tau}{\Longrightarrow} (\Gamma; \Delta)$ because $(\Gamma; \Delta) \preceq_s (\Gamma; \Delta)$ and thus we have $(\Gamma, A; \Delta)\ \mathcal{R}\ (\Gamma; \Delta)$.
  - The transition is $(\Gamma;\ \Delta, !A) \stackrel{\alpha}{\longrightarrow} (\Gamma';\ \Delta', !A)$ because of $(\Gamma; \Delta) \stackrel{\alpha}{\longrightarrow} (\Gamma'; \Delta')$. Then the latter transition can match the former because $(\Gamma'; \Delta', !A)\ \mathcal{R}\ (\Gamma'; \Delta')$.
  - The transition is $(\Gamma; \Delta, !A) \stackrel{?a}{\longrightarrow} (\Gamma; \Delta', !A)$ because of $(\Gamma; \Delta) \stackrel{?a}{\longrightarrow} (\Gamma; \Delta')$. Then we have $(\Gamma; \Delta, a) \stackrel{\tau}{\longrightarrow} (\Gamma; \Delta')$, which is a matching transition because we have $(\Gamma; \Delta', !A)\ \mathcal{R}\ (\Gamma; \Delta')$.

- If $(\Gamma;\Delta,!A)$ can be split as $((\Gamma_1;\Delta_1),(\Gamma_2;\Delta_2))$, then $!A$ occurs in either Δ_1 or Δ_2. Without loss of generality, we assume that $!A$ occurs in Δ_1. That is, there is some Δ_1' such that $\Delta_1 \equiv \Delta_1', !A$. Then $(\Gamma;\Delta) \equiv ((\Gamma_1;\Delta_1'),(\Gamma_2;\Delta_2))$. It is easy to see that $(\Gamma_1;\Delta_1)\ \mathcal{R}\ (\Gamma_1;\Delta_1')$ and $(\Gamma_2;\Delta_2)\ \mathcal{R}\ (\Gamma_2;\Delta_2)$.

– The two states are $(\Gamma,A;\Delta)$ and $(\Gamma';\Delta')$ respectively with

$$(\Gamma;\Delta) \lesssim_s (\Gamma';\Delta'). \tag{22}$$

Let us consider any transition from the first state.

- If $\Delta \equiv \cdot$, then $(\Gamma;\cdot) \lesssim_s (\Gamma';\Delta')$. So there exists some Γ'' such that $(\Gamma';\Delta') \stackrel{\tau}{\Longrightarrow} (\Gamma'';\cdot)$ and $(\Gamma;\cdot) \lesssim_s (\Gamma'';\cdot)$. It follows that $(\Gamma,A;\cdot)\ \mathcal{R}\ (\Gamma'';\cdot)$ as required.
- The transition is $(\Gamma,A;\Delta) \stackrel{\tau}{\longrightarrow} (\Gamma,A;\Delta,A)$. We argue that it is matched by the trivial transition $(\Gamma';\Delta') \stackrel{\tau}{\Longrightarrow} (\Gamma';\Delta')$. By (21) and the compositionality of $\lesssim_s$, we obtain

$$(\Gamma;\Delta,A) \lesssim_s (\Gamma;\Delta). \tag{23}$$

  By (22) and (23), together with the transitivity of similarity, it can be seen that $(\Gamma;\Delta,A) \lesssim_s (\Gamma';\Delta')$, which implies $(\Gamma,A;\Delta,A)\ \mathcal{R}\ (\Gamma';\Delta')$.
- The transition is $(\Gamma,A;\Delta) \stackrel{\alpha}{\longrightarrow} (\Gamma,A;\Delta'')$ because of $(\Gamma;\Delta) \stackrel{\alpha}{\longrightarrow} (\Gamma;\Delta'')$. By (22) there exist some Γ''',Δ''' such that $(\Gamma';\Delta') \stackrel{\alpha}{\Longrightarrow} (\Gamma''';\Delta''')$ and $(\Gamma;\Delta'') \lesssim_s (\Gamma''';\Delta''')$. Therefore, $(\Gamma,A;\Delta'')\ \mathcal{R}\ (\Gamma''';\Delta''')$ and we have found the matching transition from $(\Gamma';\Delta')$.
- The transition is $(\Gamma,A;\Delta) \stackrel{?a}{\longrightarrow} (\Gamma,A;\Delta'')$ because of $(\Gamma;\Delta) \stackrel{?a}{\longrightarrow} (\Gamma;\Delta'')$. By (22) there exist some Γ''',Δ''' such that $(\Gamma';\Delta',a) \stackrel{\alpha}{\Longrightarrow} (\Gamma''';\Delta''')$ and $(\Gamma;\Delta'') \lesssim_s (\Gamma''';\Delta''')$. Therefore, $(\Gamma,A;\Delta'')\ \mathcal{R}\ (\Gamma''';\Delta''')$ and we have found the matching transition from $(\Gamma';\Delta',a)$.
- If $(\Gamma,A;\Delta)$ can be split as $((\Gamma_1;\Delta_1),(\Gamma_2;\Delta_2))$, then A occurs in either Γ_1 or Γ_2. Without loss of generality, we assume that A occurs in Γ_1. That is, there is some Γ_1' such that $\Gamma_1 \equiv \Gamma_1',A$. Then $(\Gamma;\Delta) \equiv ((\Gamma_1';\Delta_1),(\Gamma_2;\Delta_2))$. By (22) we have the transition $(\Gamma';\Delta') \stackrel{\tau}{\Longrightarrow} ((\Gamma_3;\Delta_3),(\Gamma_4;\Delta_4))$ for some $(\Gamma_3;\Delta_3)$ and $(\Gamma_4;\Delta_4)$ such that $(\Gamma_1';\Delta_1) \lesssim_s (\Gamma_3;\Delta_3)$ and $(\Gamma_2;\Delta_2) \lesssim_s (\Gamma_4;\Delta_4)$. It follows that $(\Gamma_1;\Delta_1)\ \mathcal{R}\ (\Gamma_3;\Delta_3)$ and $(\Gamma_2;\Delta_2)\ \mathcal{R}\ (\Gamma_4;\Delta_4)$. □

Corollary 5.10. If $\Gamma;\Delta \vdash A$, then $(\cdot;A) \lesssim_s (\Gamma;\Delta)$.

Proof. By Lemma 5.7, Theorem 5.9 and the transitivity of $\lesssim_s$. □

Proposition 5.11. If $(\Gamma_1;\Delta_1) \stackrel{\tau}{\Longrightarrow} (\Gamma_2;\Delta_2)$ and $\Gamma_2;\Delta_2 \vdash A$ then $\Gamma_1;\Delta_1 \vdash A$.

Proof. Similar to the proof of Proposition 4.15. We now have two more cases:

— (rule lts!A). Suppose $(\Gamma;\Delta,!A) \stackrel{\tau}{\longrightarrow} (\Gamma,A;\Delta)$ and $\Gamma,A;\Delta \vdash B$. By rule !L, we infer that $\Gamma;\Delta,!A \vdash B$.

— (rule ltsClone). Suppose $(\Gamma,A;\Delta) \stackrel{\tau}{\longrightarrow} (\Gamma,A;\Delta,A)$ and $\Gamma,A;\Delta,A \vdash B$. By rule clone, we infer that $\Gamma,A;\Delta \vdash B$.

□

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

Our next goal is to prove Theorem 5.16, the coincidence of logical preorder with simulation. For that purpose, a series of intermediate results are in order.

Theorem 5.12. If $(\Gamma_1; A) \preceq_s (\Gamma_2; \Delta)$ then $\Gamma_2; \Delta \vdash A$.

Proof. As in Theorem 4.16, the proof is by induction on the structure of A. We now have one more case.

— $A \equiv\, !A'$. By rule lts!A we have the transition $(\Gamma_1; A) \xrightarrow{\tau} (\Gamma_1, A'; \cdot)$. Since $(\Gamma_1; A) \preceq_s (\Gamma_2; \Delta)$ there is some Γ_2' such that $(\Gamma_2; \Delta) \stackrel{\tau}{\Longrightarrow} (\Gamma_2'; \cdot)$ and

$$(\Gamma_1, A'; \cdot) \preceq_s (\Gamma_2'; \cdot). \tag{24}$$

From $(\Gamma_1, A'; \cdot)$ we have the transition $(\Gamma_1, A'; \cdot) \xrightarrow{\tau} (\Gamma_1, A'; A')$ by rule ltsClone. By Lemma 5.8 we have

$$(\Gamma_1, A'; A') \preceq_s (\Gamma_1, A'; \cdot). \tag{25}$$

It follows from (24), (25), and the transitivity of similarity that

$$(\Gamma_1, A'; A') \preceq_s (\Gamma_2'; \cdot). \tag{26}$$

Now by induction hypothesis, we obtain $\Gamma_2'; \cdot \vdash A'$ because A' has a smaller structure than A. By rule !R we infer that $\Gamma_2'; \cdot \vdash A$. Using Proposition 5.11 we conclude that $\Gamma_2; \Delta \vdash A$.

□

We now have the counterpart of Theorem 4.18.

Theorem 5.13. $(\Gamma; \Delta_1) \preceq_l (\Gamma; \Delta_2)$ if and only if $(\Gamma; \Delta_1) \preceq_s (\Gamma; \Delta_2)$.

Proof.

($\Rightarrow$) Suppose $(\Gamma; \Delta_1) \preceq_l (\Gamma; \Delta_2)$. It is trivial to see that $\Gamma; \Delta_1 \vdash \bigotimes \Delta_1$. By the definition of logical preorder, it follows that $\Gamma; \Delta_2 \vdash \bigotimes \Delta_1$. By Theorem 5.9 we have

$$(\Gamma; \bigotimes \Delta_1) \;\; \preceq_s \;\; (\Gamma; \Delta_2). \tag{27}$$

According to our reduction semantics, we have $(\Gamma; \bigotimes \Delta_1) \stackrel{\tau}{\Longrightarrow} (\Gamma; \Delta_1)$. By Lemma 5.8, it follows that

$$(\Gamma; \Delta_1) \;\; \preceq_s \;\; (\Gamma; \bigotimes \Delta_1). \tag{28}$$

By combining (27) and (28), we obtain that $(\Gamma; \Delta_1) \preceq_s (\Gamma; \Delta_2)$ because $\preceq_s$ is transitive.

($\Leftarrow$) Suppose that $(\Gamma; \Delta_1) \preceq_s (\Gamma; \Delta_2)$. If $(\Gamma', \Gamma; \Delta, \Delta_1) \vdash A$ for some Γ', Δ and A, then by Theorem 5.9 we have

$$(\Gamma', \Gamma; A) \preceq_s (\Gamma', \Gamma; \Delta, \Delta_1). \tag{29}$$

Since $(\Gamma; \Delta_1) \preceq_s (\Gamma; \Delta_2)$ and $\preceq_s$ is compositional, we obtain

$$(\Gamma', \Gamma; \Delta, \Delta_1) \;\; \preceq_s \;\; (\Gamma', \Gamma; \Delta, \Delta_2). \tag{30}$$

By (29), (30) and the transitivity of $\preceq_s$, we see that $(\Gamma', \Gamma; A) \preceq_s (\Gamma', \Gamma; \Delta, \Delta_2)$. Then, Theorem 5.12 yields $\Gamma', \Gamma; \Delta, \Delta_2 \vdash A$. Therefore, we have shown that $(\Gamma; \Delta_1) \preceq_l (\Gamma; \Delta_2)$.

□

In Theorem 5.13 we compare two states with exactly the same unrestricted resource Γ. The theorem can be relaxed so that the two states can have different unrestricted resources. In order to prove that result, we first need two lemmas.

Lemma 5.14. $(\Gamma;\Delta) \preceq_l (\cdot;\ !\Gamma,\Delta)$ and $(\cdot;\ !\Gamma;\Delta) \preceq_l (\Gamma;\Delta)$.

Proof. For any Γ' and Δ', if $\Gamma',\Gamma;\Delta',\Delta \vdash A$ then $\Gamma';\Delta',!\Gamma,\Delta \vdash A$, for any formula A, by using rule $!L$. In other words, $(\Gamma;\Delta) \preceq_l (\cdot;\ !\Gamma,\Delta)$.

Suppose $\Gamma';\Delta',!\Gamma,\Delta \vdash A$ for any Γ',Δ' and A. By rule induction on the derivation of $\Gamma';\Delta',!\Gamma,\Delta \vdash A$ it can be shown that $\Gamma',\Gamma;\Delta',\Delta \vdash A$, thus $(\cdot;\ !\Gamma,\Delta) \preceq_l (\Gamma;\Delta)$. □

Lemma 5.15. $(\Gamma;\Delta) \preceq_s (\cdot;!\Gamma,\Delta)$ and $(\cdot;!\Gamma,\Delta) \preceq_s (\Gamma;\Delta)$.

Proof. Since $(\cdot;\ !\Gamma,\Delta) \stackrel{\tau}{\Longrightarrow} (\Gamma;\Delta)$, we apply Lemma 5.8 and conclude that $(\Gamma,\Delta) \preceq_s (\cdot;\ !\Gamma,\Delta)$.

To show that $(\cdot;!\Gamma,\Delta) \preceq_s (\Gamma;\Delta)$, we let $\mathcal{R}$ be the relation that relates any state $(\Gamma;!A_1,\ldots,!A_n,\Delta)$ with the state $(\Gamma,A_1,\ldots,A_n;\Delta)$. The relation $\mathcal{R}$ is a simulation. Consider any transition from $(\Gamma;!A_1,\ldots,!A_n,\Delta)$.

— If $(\Gamma;!A_1,\ldots,!A_n,\Delta) \stackrel{\alpha}{\longrightarrow} (\Gamma';!A_1,\ldots,!A_n,\Delta')$ because of $(\Gamma;\Delta) \stackrel{\alpha}{\longrightarrow} (\Gamma';\Delta')$, the transition can be matched by $(\Gamma,A_1,\ldots,A_n;\Delta) \stackrel{\alpha}{\longrightarrow} (\Gamma',A_1,\ldots,A_n;\Delta')$.
— If $(\Gamma;!A_1,\ldots,!A_n;\Delta) \stackrel{\tau}{\longrightarrow} (\Gamma,A_1;!A_2,\ldots,!A_n,\Delta)$ then the transition can be matched by the trivial transition $(\Gamma,A_1,\ldots,A_n;\Delta) \stackrel{\tau}{\Longrightarrow} (\Gamma,A_1,\ldots,A_n;\Delta)$.
— If $(\Gamma;!A_1,\ldots,!A_n,\Delta)$ performs an input action, it must be given by an input action from Δ. Obviously, this can be mimicked by $(\Gamma,A_1,\ldots,A_n;\Delta)$.
— It is easy to see that for any splitting of $(\Gamma;!A_1,\ldots,!A_n,\Delta)$ there is a corresponding splitting of $(\Gamma,A_1,\ldots,A_n;\Delta)$.

We have shown that $\mathcal{R}$ is a simulation. Therefore, $(\Gamma;!A_1,\ldots,!A_n,\Delta) \preceq_s (\Gamma,A_1,\ldots,A_n;\Delta)$, and as a special case $(\cdot;!\Gamma,\Delta) \preceq_s (\Gamma;\Delta)$. □

Theorem 5.16. $(\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2)$ if and only if $(\Gamma_1;\Delta_1) \preceq_s (\Gamma_2;\Delta_2)$.

Proof. Suppose $(\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2)$. By Lemma 5.14 we infer that

$$(\cdot;!\Gamma_1,\Delta_1) \preceq_l (\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2) \preceq_l (\cdot;!\Gamma_2,\Delta_2).$$

Since $\preceq_l$ is a preorder, its transitivity gives $(\cdot;!\Gamma_1,\Delta_1) \preceq_l (\cdot;!\Gamma_2,\Delta_2)$. By Theorem 5.13, we have $(\cdot;!\Gamma_1,\Delta_1) \preceq_s (\cdot;!\Gamma_2,\Delta_2)$. Then by Lemma 5.15 we infer that

$$(\Gamma_1;\Delta_1) \preceq_s (\cdot;!\Gamma_1,\Delta_1) \preceq_s (\cdot;!\Gamma_2,\Delta_2) \preceq_s (\Gamma_2;\Delta_2).$$

By the transitivity of $\preceq_s$, we obtain that $(\Gamma_1;\Delta_1) \preceq_s (\Gamma_2;\Delta_2)$.

In a similar manner, we can show that $(\Gamma_1;\Delta_1) \preceq_s (\Gamma_2;\Delta_2)$ implies $(\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2)$. □

With Theorem 5.16 we can slightly generalize Theorem 5.12.

Corollary 5.17. If $(\Gamma_1;A) \preceq_s (\Gamma_2;\Delta)$ then $\Gamma_2;\Delta \vdash A \otimes \bigotimes !\Gamma_1$.

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

Proof. Suppose $(\Gamma_1; A) \preceq_s (\Gamma_2; \Delta)$. By Theorem 5.16 this means that

$$(\Gamma_1; A) \preceq_l (\Gamma_2; \Delta). \tag{31}$$

By Theorem 2.8, we have that $\Gamma_1; A \vdash A \otimes \bigotimes !\Gamma_1$. Now, by applying the definition of logical equivalence, (31) yields $\Gamma_2; \Delta \vdash A \otimes \bigotimes !\Gamma_1$. □

From Theorems 5.6 and 5.16, we obtain the main result of this subsection.

Corollary 5.18. $(\Gamma_1; \Delta_1) \preceq_l (\Gamma_2; \Delta_2)$ if and only if $(\Gamma_1; \Delta_1) \preceq_b (\Gamma_2; \Delta_2)$.

6. Concluding remarks

In this paper, we have shown that the proof-theoretic notion of logical preorder coincides with an extensional behavioural relation adapted from the process-theoretic notion of barbed preorder (Deng and Hennessy 2011). The former is defined exclusively in terms of traditional derivability, and the latter is defined in terms of a CCS-like process algebra inspired by the formula-as-process interpretation of a fragment of linear logic. In order to establish the connection, a key ingredient is to introduce a coinductively defined simulation as a stepping stone. It is interesting to see that coinduction, a central proof technique in process algebras, is playing an important role in this study of linear logic. This topic definitely deserves further investigation so that useful ideas developed in one field can benefit the other, and vice versa.

In the current work, we have interpreted a fragment of linear logic into asynchronous CCS. It is not fully clear how to satisfactorily represent synchronous process algebras in linear logic. In addition, while the & connective can be naturally translated into internal choice in process algebra, the $\oplus$ connective does not seem to match external choice very well. We hope to tackle these problems in future work.

We have started expanding the results in this paper by examining general implication (i.e. formulas of the form $A \multimap B$ rather than $a \multimap B$) and the usual quantifiers. While special cases are naturally interpreted into constructs found in the join calculus (Fournet and Gonthier 2000) and the π-calculus (Milner 1989; Sangiorgi and Walker 2001), the resulting language appears to extend well beyond them. If successful, this effort may lead to more expressive process algebras. We are also interested in understanding better the interplay of the proof techniques used in the present work. This may develop into an approach to employ coinduction effectively in logical frameworks so as to facilitate formal reasoning and verification of concurrent systems.

Acknowledgment

We thank the anonymous referees for their insightful comments which helped us to improve the current work. Support for this research was provided by the Qatar National Research Fund under NPRP grant 09-1107-1-168, by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the Carnegie Mellon Portugal Program under Grant NGN-44, and by an X10 Innovation Award from IBM. Yuxin Deng would also like to acknowledge the support of the Natural Science

https://doi.org/10.1017/S0960129514000413 Published online by Cambridge University Press

Foundation of China (61173033, 61033002, 61261130589) and ANR 12IS02001 'PACE". The statements made herein are solely the responsibility of the authors.

References

Abramsky, S. (1994) Proofs as processes. *Theoretical Computer Science* **135** 5–9.

Amadio, R., Castellani, I. and Sangiorgi, D. (1998). On bisimulation for the asynchronous pi-calculus. *Theoretical Computer Science* **195** (2) 291–324.

Barber, A. (1996) Dual intuitionistic linear logic. *Technical Report* ECS-LFCS-96-347, Laboratory for Foundations of Computer Sciences, University of Edinburgh.

Caires, L. and Cardelli, L. (2003) A spatial logic for concurrency (part I). *Information and Computation* **186** (2) 194–235.

Caires, L. and Pfenning, F. (2010) Session types as intuitionistic linear propositions. In: Proceedings of the 21st International Conference on Concurrency Theory. *Springer Lecture Notes in Computer Science* **6269** 222–236.

Caires, L., Pfenning, F. and Toninho, B. (2012) Towards concurrent type theory. In: *Proceedings of the Seventh ACM SIGPLAN Workshop on Types in Languages Design and Implementation*, ACM 1–12.

Cervesato, I., Durgin, N., Kanovich, M. I. and Scedrov, A. (2000) Interpreting strands in linear logic. In: Veith, H., Heintze, N. and Clark, E. (eds.) *Workshop on Formal Methods and Computer Security – FMCS'00*, Chicago, IL.

Cervesato, I., Pfenning, F., Walker, D. and Watkins, K. (2002) A concurrent logical framework II: Examples and applications. *Technical Report* CMU-CS-2002-002, Department of Computer Science, Carnegie Mellon University. Revised May 2003.

Cervesato, I. and Scedrov, A. (2009) Relating state-based and process-based concurrency through linear logic. *Information and Computation* **207** 1044–1077.

Dam, M. (1994) Process-algebraic interpretations of positive linear and relevant logics. *Journal of Logic and Computation* **4** (6) 939–973.

Deng, Y. and Du, W. (2011) Logical, metric, and algorithmic characterisations of probabilistic bisimulation. *Technical Report* CMU-CS-11-110, Carnegie Mellon University.

Deng, Y. and Hennessy, M. (2011) On the semantics of markov automata. In: Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP'11) *Springer Lecture Notes in Computer Science* **6756** 307–318.

Deng, Y., van Glabbeek, R., Hennessy, M., Morgan, C. and Zhang, C. (2007) Characterising testing preorders for finite probabilistic processes. In: *Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society 313–325.

Ehrhard, T. and Laurent, O. (2010) Interpreting a finitary pi-calculus in differential interaction nets. *Information and Computation* **208** (6) 606–633.

Fournet, C. and Gonthier, G. (2000) The join calculus: A language for distributed mobile programming. Applied Semantics Summer School – APPSEM'00, Caminha. Available at http://research.microsoft.com/ fournet.

Fournet, C. and Gonthier, G. (2005) A hierarchy of equivalences for asynchronous calculi. *Journal of Logic and Algebraic Programming* **63** (1) 131–173.

Gentzen, G. (1935) Untersuchungen über das logische schließen. I. *Mathematische Zeitschrift* **39** (1) 176–210.

Girard, J.-Y. (1987) Linear logic. *Theoretical Computer Science* **50** 1–102.

Girard, J.-Y., Taylor, P. and Lafont, Y. (1989) *Proofs and Types*, Cambridge University Press.

Hennessy, M. and Milner, R. (1985) Algebraic laws for nondeterminism and concurrency. *Journal of the ACM* **32** (1) 137–161.

Hoare, C. (1985) *Communicating Sequential Processes*, Prentice Hall.

Honda, K. and Yoshida, N. (1995) On reduction-based process semantics. *Theoretical Computer Science* **151** (2) 437–486.

Lincoln, P. and Saraswat, V. (1991) Proofs as concurrent processes: A logical interpretation for concurrent constraint programming. *Technical report*, Systems Sciences Laboratory, Xerox PARC.

Martin-Löf, P. (1996) On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic* **1** (1) 11–60.

McDowell, R., Miller, D. and Palamidessi, C. (2003) Encoding transition systems in sequent calculus. *Theoretical Computer Science* **294** (3) 411–437.

Miller, D. (1992) The π-calculus as a theory in linear logic: Preliminary results. In: Lamma, E. and P. Mello (eds.) Proceedings of the Workshop on Extensions to Logic Programming – ELP'92. *Springer-Verlag Lecture Notes in Computer Science* **660** 242–265.

Milner, R. (1989) *Communication and Concurrency*, Prentice Hall.

Pfenning, F. (2000) Structural cut elimination I. Intuitionistic and classical logic. *Information and Computation* **157** (1/2) 84–141.

Pfenning, F. and Davies, R. (2001) A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science* **11** (4) 511–540. Notes to an invited talk at the *Workshop on Intuitionistic Modal Logics and Applications (IMLA'99)*, Trento, Italy, July 1999.

Rathke, J. and Sobocinski, P. (2008) Deriving structural labelled transitions for mobile ambients. In: Proceedings of the 19th International Conference on Concurrency Theory. *Springer Lecture Notes in Computer Science* **5201** 462–476.

Sangiorgi, D. and Walker, D. (2001) *The π-Calculus: A Theory of Mobile Processes*, Cambridge University Press.

Tiu, A. and Miller, D. (2004) A proof search specification of the π-calculus. In: 3rd Workshop on the Foundations of Global Ubiquitous Computing. *Electronic Notes in Theoretical Computer Science* **138** 79–101.

Wadler, P. (2012) Propositions as sessions. In: *Proceedings of the 17th ACM SIGPLAN International Conference on Functional Programming*, ACM 273–286.