Relating Reasoning Methodologies in Linear Logic and Process Algebra*

Yuxin Deng
Carnegie Mellon University
and Shanghai Jiao Tong University
yuxin@cs.cmu.edu

Iliano Cervesato
Carnegie Mellon University
Doha, Qatar
iliano@cmu.edu

Robert J. Simmons
Carnegie Mellon University
Pittsburgh, PA
rjsimmon@cs.cmu.edu

We show that the proof-theoretic notion of logical preorder coincides with the process-theoretic notion of contextual preorder for a CCS-like 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 a labeled transition system and a coinductively defined simulation relation. This result establishes a connection between an approach to reason about process specifications and a method to reason about logic specifications.

1 Introduction

By now, execution-preserving relationships between (fragments of) linear logic and (fragments of) process algebras are well-established (see [5] for an overview). Abramsky observed early on that linear cut elimination resembles reduction in CCS and the π-calculus [15], thereby identifying processes with (some) linear proofs and establishing the *process-as-term* interpretation [1]. The alternative *process-as-formula* encoding, pioneered by Miller around the same time [14], 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. This interpretation has been used extensively in a multitude of domains [3, 4, 5, 14], e.g., programming languages and security.

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 behavioral 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 [14]. A similar characterization was made in [12], 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 behavioral relations: for example, adding definitions allows expressing simulation as the derivability of a linear implication [13], but falls short of bisimulation, for which a nominal logic is instead an adequate formalism [17].

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).

*Partially supported by the Qatar National Research Fund under grant NPRP 09-1107-1-168, the Natural Science Foundation of China under grant 61173033, and the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the Carnegie Mellon Portugal Program under Grant NGN-44.

Submitted to:
LINEARITY 2012

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

$$\frac{\Gamma;\Delta_1\vdash A \quad \Gamma;\Delta_2\vdash B}{\Gamma;\Delta_1,\Delta_2\vdash A\otimes B}\otimes R \qquad \frac{\Gamma;\Delta,A,B\vdash C}{\Gamma;\Delta,A\otimes B\vdash C}\otimes L \qquad \frac{}{\Gamma;\cdot\vdash \mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma;\Delta\vdash C}{\Gamma;\Delta,\mathbf{1}\vdash C}\ \mathbf{1}L$$

$$\frac{\Gamma;\Delta\vdash A \quad \Gamma;\Delta\vdash B}{\Gamma;\Delta\vdash A\,\&\,B}\ \&R \qquad \frac{\Gamma;\Delta,A_i\vdash C}{\Gamma;\Delta,A_1\,\&\,A_2\vdash C}\ \&L_i \qquad \frac{}{\Gamma;\Delta\vdash\top}\ \top R \qquad \textit{(no rule } \top L\textit{)}$$

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

Figure 1: A Fragment of Dual Intuitionistic Linear Logic

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; this preorder is novel in the sense that it is a natural and proof-theoretic way of relating *contexts* to other contexts. On the process-algebraic side, we consider an extensional behavioral relation adapted from the standard coinductive notion of contextual preorder. We prove that, for a fragment of linear logic and a matching process calculus, these notions coincide, and we hope in future work to extend this result to a larger fragment of intuitionistic linear logic. Our proofs rely on other standard process algebraic notions as stepping stones, namely simulation and labeled transition systems.

The rest of the paper is organized as follows. In Section 2, we briefly review the fragment of linear logic we are focusing on and define the logical preorder. Then, in Section 3, we recall its standard process-as-formula interpretation and define the contextual preorder. In Section 4, we prove their equivalence through the intermediary of a simulation preorder defined on the basis of a labeled transition system. Full proofs of all results in this paper can be found in the accompanying technical report [7].

2 Logical Preorder

The fragment of linear logic considered in this paper is given by the following grammar:

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

where a is an atomic formula. This language is propositional and, as often the case in investigations of CCS-like process algebras [3, 4, 5], the antecedent of linear implication is restricted to atomic formulas (see the remarks in Section 5 about lifting these constraints).

Derivability for this language is given in terms of dual intuitionistic linear logic (DILL) sequents [2, 5] of the form $\Gamma;\Delta\vdash A$, where the *unrestricted context* Γ and the *linear context* Δ are multisets of formulas. Formally, they are defined by the productions $\Gamma,\Delta ::= \cdot \mid \Delta,A$ where "$\cdot$" represents the empty context, and "Δ,A" is the context obtained by adding the formula A to the context Δ. As usual, we tacitly treat "," as an associative and commutative context union operator "Δ_1,Δ_2" with "$\cdot$" as its unit.

The fairly standard inference rules defining derivability are given in Figure 1. A DILL sequent $\Gamma;\Delta\vdash A$ corresponds to $!\Gamma,\Delta\vdash A$ in Girard's original presentation [9]. We take 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.

We will be interested in a relation, the logical preorder, that compares specifications on the basis of the properties they satisfy, possibly after the components they describe are plugged into a larger system. This relation, written $\preceq_l$, is given by the following definition.

Definition 1 (Logical preorder) *The* logical preorder *is the smallest relation* $\preceq_l$ *such that* $(\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2)$ *if, for all* Γ', Δ', *and* C, *we have that* $(\Gamma',\Gamma_1);(\Delta',\Delta_1) \vdash C$ *implies* $(\Gamma',\Gamma_2);(\Delta',\Delta_2) \vdash C$. □

This relation is reflexive and transitive, and therefore a preorder [7, Theorem 2.5]; we could define logical equivalence as the symmetric closure of $\preceq_l$. The above definition is extensional in the sense that it refers to all contexts Γ' and Δ' and formulas C. It has also an inductive characterization based on derivability [7, Theorem 2.8]:

Property 2 $(\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2)$ *iff* $\Gamma_2;\Delta_2 \vdash \bigotimes !\Gamma_1 \otimes \bigotimes \Delta_1$

Here, $\bigotimes \Delta_1$ denotes the conjunction of all formulas in Δ_1 (or $\mathbf{1}$ if it is empty) and $!\Gamma_1$ is the linear context obtained by prefixing every formula in Γ_1 with "!".

The logical preorder has other interesting properties, such as *harmony* [7, Proposition 2.6]:

Property 3 (Harmony) $\Gamma;\Delta \vdash A$ *if and only if* $(\cdot;A) \preceq_l (\Gamma;\Delta)$. □

In the accompanying technical report, we show that a deductive system satisfies harmony if and only if the rules of *identity* and *cut* are admissible:

$$\frac{}{\Gamma;A \vdash A}\ \textit{identity} \qquad \frac{\Gamma;\Delta \vdash A \quad \Gamma';\Delta',A \vdash C}{(\Gamma,\Gamma');(\Delta,\Delta') \vdash C}\ \textit{cut}$$

In particular, this means that harmony holds not only for our restricted fragment of DILL, but for full DILL and most other syntactic fragments of DILL as well [7, Section 2.3].

3 Contextual Preorder

The subset of linear logic just introduced has a natural interpretation as a fragment of CCS [15] with CSP-style internal choice [10]. It is shown in Figure 2. We will now switch to this reading, which is known as the *conjunctive process-as-formula interpretation* of linear logic [14, 5]. Therefore, for most of the rest of this section, we understand A as a process.

Under this reading, contexts $(\Gamma;\Delta)$ are *process states*, i.e., systems of parallel processes understood as the parallel composition of each process in Δ and, after restoring the implicit replication, in Γ. In process algebra, parallel composition is considered associative and commutative and has the null process as its unit. This endows process states with the following structural congruences:

$$\begin{array}{rcl} (\Gamma;\Delta,\cdot) & \equiv & (\Gamma;\Delta) \\ (\Gamma;\Delta_1,\Delta_2) & \equiv & (\Gamma;\Delta_2,\Delta_1) \\ (\Gamma;\Delta_1,(\Delta_2,\Delta_3)) & \equiv & (\Gamma;(\Delta_1,\Delta_2),\Delta_3) \end{array} \qquad \begin{array}{rcl} (\Gamma,\cdot;\Delta) & \equiv & (\Gamma;\Delta) \\ (\Gamma_1,\Gamma_2;\Delta) & \equiv & (\Gamma_2,\Gamma_1;\Delta) \\ (\Gamma_1,(\Gamma_2,\Gamma_3);\Delta) & \equiv & ((\Gamma_1,\Gamma_2),\Gamma_3;\Delta) \\ (\Gamma,A,A;\Delta) & \equiv & (\Gamma,A;\Delta) \end{array}$$

which amount to asking that Δ and Γ be commutative monoids. The last equality on the right merges identical replicated processes, making Γ into a set. In the following, we will always consider process states modulo this structural equality, and therefore treat equivalent states as syntactically identical.

a	atomic process that sends a
$A \otimes B$	process that forks into processes A and B
$\mathbf{1}$	null process
$A_1 \,\&\, A_2$	process that can behave either as A_1 or as A_2
$\top$	stuck process
$a \multimap B$	process that receives a and continues as B
$!A$	any number of copies of process A

Figure 2: Process-as-formula Interpretation

$$
\begin{array}{rcl}
(\Gamma;\Delta, A \otimes B) & \rightsquigarrow & (\Gamma;\Delta, A, B) \\
(\Gamma;\Delta, \mathbf{1}) & \rightsquigarrow & (\Gamma;\Delta) \\
(\Gamma;\Delta, A_1 \,\&\, A_2) & \rightsquigarrow & (\Gamma;\Delta, A_i) \\
\multicolumn{3}{c}{\textit{(No rule for } \top\textit{)}} \\
(\Gamma;\Delta, a, a \multimap B) & \rightsquigarrow & (\Gamma;\Delta, B) \\
(\Gamma;\Delta, !A) & \rightsquigarrow & (\Gamma, A;\Delta) \\
(\Gamma, A;\Delta) & \rightsquigarrow & (\Gamma, A;\Delta, A)
\end{array}
$$

Figure 3: Transitions

Opening a parenthesis back into logic, it easy to prove that structurally equivalent context pairs are logically equivalent. In symbols, if $(\Gamma_1;\Delta_1) \equiv (\Gamma_2;\Delta_2)$, then $(\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2)$ and $(\Gamma_2;\Delta_2) \preceq_l (\Gamma_1;\Delta_1)$.

The analogy with CCS above motivates the reduction relation $\rightsquigarrow$ between process states defined in Figure 3. A formula $A \otimes B$ (parallel composition) transitions to the two formulas A and B in parallel, for instance, and a formula $A \,\&\, B$ (choice) either transitions to A or to B. 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. We write $\rightsquigarrow^*$ for the reflexive and transitive closure of $\rightsquigarrow$.

Intuitively, two systems of processes are contextually equivalent if they behave in the same way when composed with any given process. We will be interested in the asymmetric variant of this notion, a relation known as the contextual preorder. We understand "behavior" as the ability to produce the same messages. To model this, 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 $(\Gamma';\Delta')\downarrow_a$. We also define the *composition* of two states $(\Gamma_1;\Delta_1)$ and $(\Gamma_2;\Delta_2)$, written $((\Gamma_1;\Delta_1),(\Gamma_2;\Delta_2))$, as the state $((\Gamma_1,\Gamma_2);(\Delta_1,\Delta_2))$.

In defining our contextual preorder, we will also require that partitions of systems of processes behave congruently — a related notion, Markov simulation, is used in probabilistic process algebras [6]. We formally capture it by defining that a *partition* of a state $(\Gamma;\Delta)$ is any pair of states $(\Gamma_1;\Delta_1)$ and $(\Gamma_2;\Delta_2)$ such that $(\Gamma;\Delta) \equiv ((\Gamma_1;\Delta_1),(\Gamma_2;\Delta_2))$.

Definition 4 (Contextual preorder) *Let $\mathscr{R}$ be a binary relation over states. We say that $\mathscr{R}$ is*

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

reduction-closed *if $(\Gamma_1;\Delta_1)\,\mathscr{R}\,(\Gamma_2;\Delta_2)$ and $(\Gamma_1;\Delta_1) \rightsquigarrow (\Gamma_1';\Delta_1')$ implies $(\Gamma_2;\Delta_2) \rightsquigarrow^* (\Gamma_2';\Delta_2')$ and $(\Gamma_1';\Delta_1')\,\mathscr{R}\,(\Gamma_2';\Delta_2')$ for some $(\Gamma_2';\Delta_2')$.*

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

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

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

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

The contextual preorder is indeed reflexive and transitive, and therefore a preorder [7, Theorem 3.7]. In contrast to the proof of analogous property of the logical preorder, the proof of this result is coinductive.

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

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

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

$$\frac{}{(\Gamma;\Delta,A_1 \,\&\, A_2) \xrightarrow{\tau} (\Gamma;\Delta,A_i)}\ lts\&_i \qquad \text{(No rule for } \top\text{)}$$

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

Figure 4: Labeled Transition System

Contextual equivalence, which is the symmetric closure of the contextual 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 [11, 16, 8, 6].

4 Correspondence via Simulation

In this section, we show that the logical and the contextual preorders are the same relation. A direct proof eluded us, as the inductive reasoning techniques that underlie derivations (on which $\preceq_l$ is based) do not play nicely with the intrinsically coinductive arguments that are natural for $\preceq_c$. Instead, our proof uses a second relation, the simulation preorder, as a stepping stone. This intermediary relation is also coinductive, but it is relatively easy to show that it is equivalent to the logical preorder.

The definition of the simulation preorder relies on the labeled transition system in Figure 4. It defines the transition judgment $(\Gamma_1;\Delta_1) \xrightarrow{\beta} (\Gamma_2;\Delta_2)$ between states $(\Gamma_1;\Delta_1)$ and $(\Gamma_2;\Delta_2)$. Here, β is a label. We distinguish "non-receive" labels, denoted α, as either the silent action τ or a label $!a$ for atomic formula a — it represents a send action of a. Generic labels β extend them with receive actions $?a$. We write $\xRightarrow{\tau}$ for the reflexive and transitive closure of $\xrightarrow{\tau}$, and $(\Gamma_1;\Delta_1) \xRightarrow{\beta} (\Gamma_2;\Delta_2)$ for $(\Gamma_1;\Delta_1) \xRightarrow{\tau} \xrightarrow{\beta} \xRightarrow{\tau} (\Gamma_2;\Delta_2)$, if $\beta \neq \tau$.

Rule $lts!?$ synchronizes a send action (rule $lts!$) with a receive action (rule $lts?$), thereby achieving the same effect as the transition for $\multimap$ in Figure 3. The other τ transitions in Figure 4 correspond directly to reductions (since $\top$ is the stuck process, it has no action to perform). Indeed, the following result holds [7, Lemma 4.1, Lemma 5.2]:

Property 5 $(\Gamma_1;\Delta_1) \xRightarrow{\tau} (\Gamma_2;\Delta_2)$ *if and only if* $(\Gamma_1;\Delta_1) \rightsquigarrow^* (\Gamma_2;\Delta_2)$. □

Based on the labeled transition system in Figure 4, we are in a position to give a coinductive definition of the simulation preorder.

Definition 6 (Simulation preorder) *A relation $\mathscr{R}$ between two processes represented as $(\Gamma_1;\Delta_1)$ and $(\Gamma_2;\Delta_2)$ is a* simulation *if $(\Gamma_1;\Delta_1)\ \mathscr{R}\ (\Gamma_2;\Delta_2)$ implies that*

1. *if* $(\Gamma_1;\Delta_1) \equiv (\Gamma_1';\cdot)$ *then* $(\Gamma_2;\Delta_2) \stackrel{\tau}{\Longrightarrow} (\Gamma_2';\cdot)$ *and* $(\Gamma_1';\cdot)\ \mathscr{R}\ (\Gamma_2';\cdot)$.
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* $(\Gamma_1';\Delta_1')\ \mathscr{R}\ (\Gamma_2';\Delta_2')$ *and* $(\Gamma_1'';\Delta_1'')\ \mathscr{R}\ (\Gamma_2'';\Delta_2'')$.
3. *if* $(\Gamma_1;\Delta_1) \stackrel{\alpha}{\longrightarrow} (\Gamma_1';\Delta_1')$, *there exists* $(\Gamma_2';\Delta_2')$ *such that* $(\Gamma_2;\Delta_2) \stackrel{\alpha}{\Longrightarrow} (\Gamma_2';\Delta_2')$ *and* $(\Gamma_1';\Delta_1')\ \mathscr{R}\ (\Gamma_2';\Delta_2')$.
4. *if* $(\Gamma_1;\Delta_1) \stackrel{?a}{\longrightarrow} (\Gamma_1';\Delta_1')$, *there exists* $(\Gamma_2';\Delta_2')$ *such that* $(\Gamma_2;\ \Delta_2,a) \stackrel{\tau}{\Longrightarrow} (\Gamma_2';\Delta_2')$ *and* $(\Gamma_1';\Delta_1')\ \mathscr{R}\ (\Gamma_2';\Delta_2')$.

We write $(\Gamma_1;\Delta_1) \preceq_s (\Gamma_2;\Delta_2)$ *if there is some simulation* $\mathscr{R}$ *with* $(\Gamma_1;\Delta_1)\ \mathscr{R}\ (\Gamma_2;\Delta_2)$. *We call* $\preceq_s$ *the* simulation preorder. □

The first two points of the definition ensure that a simulation is partition-preserving. The others characterize similarity. The fourth is a key bridge to the logical behavior of implication. The simulation preorder is reflexive, transitive (i.e., a preorder) and compositional [7, Proposition 4.9, Theorem 4.11, Proposition 5.5]:

Property 7 ($\preceq_s$ is a compositional preorder)

- $\preceq_s$ *is a preorder.*
- *If* $(\Gamma_1;\Delta_1) \preceq_s (\Gamma_2;\Delta_2)$, *then* $((\Gamma_1;\Delta_1),(\Gamma;\Delta)) \preceq_s ((\Gamma_2;\Delta_2),(\Gamma;\Delta))$ *for any process state* $(\Gamma;\Delta)$. □

The soundness and completeness of the contextual preorder with respect to the simulation preorder is readily established by coinduction [7, Theorem 4.12, Theorem 4.13, Theorem 5.6]:

Theorem 8 $(\Gamma_1;\Delta_1) \preceq_c (\Gamma_2;\Delta_2)$ *if and only if* $(\Gamma_1;\Delta_1) \preceq_s (\Gamma_2;\Delta_2)$. □

Relating the simulation preorder and the logical preorder is more involved, and is where the inductive approach to reasoning about the former meets the coinductive arguments normally used with the latter. The following set of properties establish the connection: the directions that assume the linear preorder proceed by induction, while those that assume the simulation preorder use coinduction.

Property 9

- $\Gamma;\Delta \vdash A$ *if and only if* $(\cdot;A) \preceq_s (\Gamma;\Delta)$ *[7, Theorem 5.9].*
- $(\Gamma;\Delta) \preceq_l (\cdot;\ !\Gamma,\Delta)$ *and* $(\cdot;\ !\Gamma;\Delta) \preceq_l (\Gamma;\Delta)$ *[7, Lemma 5.14].*
- $(\Gamma;\Delta) \preceq_s (\cdot;!\Gamma,\Delta)$ *and* $(\cdot;!\Gamma,\Delta) \preceq_s (\Gamma;\Delta)$ *[7, Lemma 5.15].* □

We can now establish the correspondence between the two preorders [7, Theorem 4.17, Theorem 5.16].

Theorem 10 $(\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2)$ *if and only if* $(\Gamma_1;\Delta_1) \preceq_s (\Gamma_2;\Delta_2)$. □

Chaining Theorems 8 and 10 yields the main result of the paper, i.e., the equivalence of the logical and contextual preorder [7, Corollary 4.18, Corollary 5.18].

Corollary 11 $(\Gamma_1;\Delta_1) \preceq_l (\Gamma_2;\Delta_2)$ *if and only if* $(\Gamma_1;\Delta_1) \preceq_c (\Gamma_2;\Delta_2)$. □

5 Conclusions and Future Work

Corollary 11 shows that the proof-theoretic notion of logical preorder coincides with an extensional behavioral relation adapted from the process-theoretic notion of contextual preorder. 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.

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 and the π-calculus, 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.

References

[1] Samson Abramsky (1994): *Proofs as Processes*. Theoretical Computer Science 135, pp. 5–9.

[2] Andrew Barber (1996): *Dual Intuitionistic Linear Logic*. Technical Report ECS-LFCS-96-347, Laboratory for Foundations of Computer Sciences, University if Edinburgh.

[3] Iliano Cervesato, Nancy Durgin, Max Kanovich & Andre Scedrov (2000): *Interpreting Strands in Linear Logic*. In: 2000 Workshop on Formal Methods and Computer Security, Chicago, IL.

[4] Iliano Cervesato, Frank Pfenning, David Walker & Kevin Watkins (2002): *A Concurrent Logical Framework II: Examples and Applications*. Technical Report CMU-CS-2002-002, Carnegie Mellon University.

[5] Iliano Cervesato & Andre Scedrov (2009): *Relating state-based and process-based concurrency through linear logic*. Information and Computation 207, pp. 1044–1077.

[6] Yuxin Deng & Matthew Hennessy (2011): *On the Semantics of Markov Automata*. In: Proc. ICALP'11, Springer-Verlag LNCS 6756, pp. 307–318.

[7] Yuxin Deng, Robert J. Simmons & Iliano Cervesato (2011): *Relating Reasoning Methodologies in Linear Logic and Process Algebra*. Technical Report CMU-CS-11-145, Carnegie Mellon University.

[8] Cédric Fournet & Georges Gonthier (2005): *A hierarchy of equivalences for asynchronous calculi*. Journal of Logic and Algebraic Programming 63(1), pp. 131–173.

[9] Jean-Yves Girard (1987): *Linear logic*. Theoretical Computer Science 50, pp. 1–102.

[10] C.A.R. Hoare (1985): *Communicating Sequential Processes*. Prentice Hall.

[11] Kohei Honda & Mario Tokoro (1992): *On Asynchronous Communication Semantics*. In: Proc. of ECOOP'91 Workshop on Object-Based Concurrent Computing, Springer-Verlag LNCS 612.

[12] Patrick Lincoln & Vijay Saraswat (1991): *Proofs as concurrent processes: A logical interpretation for concurrent constraint programming*. Technical Report, Systems Sciences Laboratory, Xerox PARC.

[13] Raymond McDowell, Dale Miller & Catuscia Palamidessi (2003): *Encoding transition systems in sequent calculus*. Theoretical Computer Science 294(3), pp. 411–437.

[14] Dale Miller (1992): *The π-Calculus as a Theory in Linear Logic: Preliminary Results*. In E. Lamma & P. Mello, editors: Proc. ELP, Springer-Verlag LNCS 660, pp. 242–265.

[15] Robin Milner (1989): *Communication and Concurrency*. Prentice Hall.

[16] Julian Rathke & Pawel Sobocinski (2008): *Deriving Structural Labelled Transitions for Mobile Ambients*. In: Proc. CONCUR'08, Springer-Verlag LNCS 5201, pp. 462–476.

[17] Alwen Tiu & Dale Miller (2004): *A Proof Search Specification of the π-Calculus*. In: 3rd Workshop on the Foundations of Global Ubiquitous Computing, ENTCS 138, pp. 79–101.