A Logical View of Composition

Martín Abadi[*] Gordon D. Plotkin[†]

Abstract

We define two logics of safety specifications for reactive systems. The logics provide a setting for the study of composition rules. The two logics arise naturally from extant specification approaches; one of the logics is intuitionistic, while the other one is linear.

1 Introduction

Modular, hierarchical methods for specifying reactive systems [HP85a] include rules for composing and refining specifications (*e.g.*, [dBdRR90]). The form of the rules suggests a possible specification logic. In it, the propositions would be system specifications; the notations for combining specifications would become logical connectives; and the rules for composition and refinement would be formulated as sound inference rules. The logic would thereby provide a setting for the study of composition and refinement rules. It should also provide a framework for writing specifications and for verifying them using these rules.

In this paper, we define and develop such a logic for composition. We intend to treat refinement in a second paper, and thereby complete a framework for the use of the modular specification methods that composition and refinement rules underpin. At that point it will be natural and useful to consider a formal logic; in this paper we prefer to work at the semantical level. (The treatment of refinement and the formal logic were sketched in a preliminary version of this paper [AP91].)

In fact two logics of composition arise naturally. One of the logics is an intuitionistic logic, while the other one is linear [Gir87]. In the intuitionistic

[*]Digital Equipment Corporation, Systems Research Center, 130 Lytton Avenue, Palo Alto, California 94301, USA.

[†]Dept. of Computer Science, University of Edinburgh, King's Buildings, Edinburgh EH9 3JZ, UK. This work was completed while at Digital Equipment Corporation.

logic, a specification is a set of allowed behaviors, as in [Lam83a, BKP84]. In the linear logic, a specification is a set of allowed processes, much as in the sense of Abrahamson [Abr79].

Composition rules rules typically apply to safety properties, and also, sometimes with significant complication, to certain liveness properties. Here we treat only safety properties. With this restriction, the logics provide a new understanding of some current specification methods, and suggest extensions. They are intended as a basis for Lamport's transition-axiom method for reactive systems [Lam89].

A reactive system can be expected to operate correctly only when its environment operates correctly. For example, a concurrent program module can be expected to exhibit desirable behavior only when its inputs are of the proper types. The environment cannot be required to operate correctly, but the system's obligations are void when the environment operates incorrectly. An assumption-guarantee specification states that a reactive system satisfies a specification M if it operates in an environment that satisfies an assumption E; this specification is sometimes written $E \Rightarrow M$.

A Composition Principle gives a way of combining assumption-guarantee specifications while discharging their assumptions [MC81, Pnu84, Sta85, AL90]. A simple version of the principle, applied to two reactive systems p_1 and p_2, says:

> If p_1 satisfies $M_2 \Rightarrow M_1$
> and p_2 satisfies $M_1 \Rightarrow M_2$,
> then when they are run in parallel
> p_1 satisfies M_1 and p_2 satisfies M_2.

As stated, the Composition Principle is not sound in general. The underlying propositional reasoning is obviously (and intriguingly) circular.

However, the principle is sound when M_1 and M_2 are safety properties, and under some additional hypotheses. For instance, consider two processes p_1 and p_2 that communicate by the distributed integer variables x_1 and x_2; it is assumed that only p_1 writes x_1 and that only p_2 writes x_2. Let M_1 be "x_1 never decreases" and M_2 be the corresponding assertion for x_2, and suppose that p_1 and p_2 satisfy $M_2 \Rightarrow M_1$ and $M_1 \Rightarrow M_2$, respectively. Then it is sound to conclude that M_1 and M_2 both hold, that is, that neither x_1 nor x_2 ever decreases.

An important test for a logic of specifications is whether it can be used to express and to illuminate the Composition Principle. Both of our logics are designed to satisfy this criterion. For example, the intuitionistic formulation of the principle just given is:

$$(M_2 \to M_1) \wedge (M_1 \to M_2) \vdash M_1 \wedge M_2$$

with a proviso to guarantee that M_1 and M_2 are specifications of separate processes. The logics can express also other variants of the Composition Principle; they serve in comparing these variants and, occasionally, in discovering new ones.

As we consider only safety properties, which are closed sets, we obtain an intuitionistic logic. In this we follow Hennessy and Plotkin [HP87] and, less directly, Abramsky with his proposal of a general logic of open sets [Abr89]. Parallel composition can be represented by conjunction, as in works of Lamport and Pnueli. Both Dam [Dam88] and Abramsky [Vic88] pointed out that in general parallelism will give extra, quantalic structure. This indeed happens when we take specifications to be sets of processes, and then the logic of specifications is linear. Our work may yield some evidence for the relevance of linear logic to concurrency. Other evidence can be found in work on Petri Nets (*e.g.*, [MOM89]) and testing equivalence [AV90].

We introduce our logics in the following overview. Sections 3, 4, and 5 give formal definitions and develop the material further.The reader may wish to consult [DP91, Joh82, Ros90] , for information on partial orders, cpos (complete partial orders), complete Heyting algebras, and quantales.

2 Overview

We review the basic propositional intuitionistic and linear calculi. We describe the usual connectives, and motivate the addition of new constructs, which are needed in order to support the assumption-guarantee specification style.

2.1 A calculus of sets of behaviors

The intuitionistic logic is inspired by the work of Lamport, Pnueli, and others, where the specification of a system is a set of allowed behaviors. In turn, a behavior is a sequence of state transitions, and a state is an assignment of values to state components, or variables. Each state transition is attributed to an agent, the environment process or system process that caused the state change. Thus, a behavior is a sequence

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \ldots$$

where each s_i is a state and each a_i is an agent, and the sequence is either infinite or else ends in a state s_m for some $m \geq 0$.

The use of agents is motivated by the obvious need to distinguish between actions performed by the environment and those performed by the system. In any particular specification, it suffices to consider two agents: the environment and the system. However, it is preferable to allow arbitrary sets of agents, in order to ease the composition of specifications. Agents are taken as a primitive notion below, but this can be avoided, for example as in [Pnu84].

Since we are concerned only with safety properties, we restrict attention to finite behaviors. A safety property is then a prefix-closed set of behaviors. In the logic, the propositions denote safety properties, and $\vdash$ simply stands for $\subseteq$. The collection of safety properties forms a complete Heyting algebra [Joh82] and so the intuitionistic logical operations $\wedge$, $\vee$, and $\rightarrow$ are available. The first two are intersection and union.

Conjunction serves its usual logical role: a process p satisfies $M_1 \wedge M_2$ if and only if it satisfies both M_1 and M_2. Further, conjunction represents parallel composition: if p_1 satisfies M_1 and p_2 satisfies M_2 then p_1 and p_2 in parallel satisfy $M_1 \wedge M_2$. Disjunction corresponds to nondeterministic choice: if p_1 satisfies M_1 and p_2 satisfies M_2 then a processes that acts like either p_1 or p_2 satisfies $M_1 \vee M_2$.

Implication turns out to be a familiar and handy operation: $E \rightarrow M$ is the set of all behaviors that satisfy M at least as long as they satisfy E. The connective $\rightarrow$ has arisen in works on the Composition Principle (in [AL90], and implicitly in [MC81] and [Pnu84]). Under reasonable hypotheses, the specifications $E \Rightarrow M$ and $E \rightarrow M$ have the same implementations, and hence $\Rightarrow$ can be replaced with $\rightarrow$. It is encouraging that the logical formulation naturally yields this connective.

The specification of a system cannot require the environment to work properly, and so any environment action should be allowed. More precisely, if a property M is intended to specify the process represented by an agent (or set of agents) μ, then any minimal behavior not in M should end with a μ state change. When this condition holds, we say that M constrains at most μ, and write $M \triangleleft \mu$.

With this notation, the Composition Principle reads: for any M_1 and M_2,

$$(M_2 \rightarrow M_1) \wedge (M_1 \rightarrow M_2) \vdash M_1 \wedge M_2$$

provided $M_1 \triangleleft \mu_1$, $M_2 \triangleleft \mu_2$, and the sets μ_1 and μ_2 are disjoint. The proviso expresses the requirement that M_1 and M_2 describe different processes. (The principle is not sound otherwise, for example if M_1 and M_2 are the same.) Note how the logical approach obviates the need for explicit reference either to processes (as in [MC81, Pnu84]) or to the realizable parts of properties (as in [AL90]).

Many variants of the Composition Principle can be treated in this framework; for example, we easily obtain:

$$\frac{E \wedge M_2 \vdash E_1 \qquad E \wedge M_1 \vdash E_2}{(E_1 \rightarrow M_1) \wedge (E_2 \rightarrow M_2) \vdash (E \rightarrow M_1 \wedge M_2)}$$

where $M_1 \lhd \mu_1$ and $M_2 \lhd \mu_2$. Some of these variants are well known, while others seem to be new. All of them can be proved equivalent using propositional reasoning and a few rules about the constrains relation.

2.2 A calculus of sets of processes

In the linear calculus, a proposition denotes a set of processes. We take a process to be a set of sequences of state pairs. Intuitively, a process that contains $(s_1, t_1)\,(s_2, t_2)\,(s_3, t_3)\,\ldots$ can change the state from s_1 to t_1, and later from s_2 to t_2, and later yet from s_3 to t_3, ...

In the study of safety, it suffices to consider finite sequences of state pairs. We require also that processes be prefix-closed. It turns out that the set of safety properties is isomorphic to the set of processes; thus, we may identify safety properties and processes.

The logical operations $\wedge$, $\vee$, and $\rightarrow$ are still meaningful. They arise as before from the complete Heyting algebra structure of the partial order of safety properties.

The property $M_1 \wedge M_2$ allows the processes that are allowed both by M_1 and M_2; conjunction does not have any particular relation with concurrency. Disjunction corresponds to nondeterministic choice, as before. Finally, $M_1 \rightarrow M_2$ includes the processes that behave like a process in M_2 at least as long as they behave like a process in M_1.

Intuitionistic linear logic arises when we consider the parallel composition of two processes. The parallel composition of p_1 and p_2 is the set of shuffles of p_1 sequences with p_2 sequences. At the level of specifications, this gives rise to a new logical operation, $\otimes$, which is the multiplicative conjunction in linear logic. A process satisfies $M_1 \otimes M_2$ if it is the parallel composition of an M_1 process with an M_2 process. Thus, if p_1 satisfies M_1 and p_2 satisfies M_2 then the parallel composition of p_1 and p_2 satisfies $M_1 \otimes M_2$.

Associated with the connective $\otimes$ is a linear implication operation, $\multimap$. The property $M_1 \multimap M_2$ is the largest N such that $M_1 \otimes N$ is a subset of M_2. Thus, $p \in M_1 \multimap M_2$ if and only if the parallel composition of p with any $q \in M_1$ satisfies M_2.

Conjunction and disjunction are then the additive connectives of linear logic. The exponential operator ! is trivial, but a nontrivial $(\cdot)^*$ construct can be added to represent the parallel composition of a number of like processes. In the next subsection, we propose an interpretation of the classical constructs.

The standard intuitionistic linear connectives do not suffice as a basis for assumption-guarantee specifications. In particular, $p \in E \multimap M$ is not equivalent to the desired "p satisfies M in any environment that satisfies E." The assertion $p \in E \multimap M$ means only that the composition of p with any E process q is an M process. It is possible that q is not the whole environment of p—there could be a third process running in parallel; it is also possible that p does not satisfy M in this environment—the parallel composition of p and q does.

To remedy this deficiency, we introduce a connective $-\!\!\diamond$. The property $M_1 -\!\!\diamond M_2$ consists of the processes that, when run in parallel with an M_1 process (and with nothing else), behave like M_2 processes. The special case of $M_1 -\!\!\diamond M_2$ where M_1 contains only the null process 1 is of particular interest; $\{1\} -\!\!\diamond M$ is the set of all processes that behave like a process in M when run by themselves, with no interference from the environment. We denote this property by $M^\diamond$.

Now the Composition Principle goes:

$$(M_2 -\!\!\diamond M_1) \otimes (M_1 -\!\!\diamond M_2) \vdash (M_1 \otimes M_2)^\diamond$$

This formula is valid in our model, without any additional proviso. As in the intuitionistic case, a number of variants of the Composition Principle are available, and for example we have also the more general:

$$\frac{E \otimes M_2 \vdash E_1 \qquad E \otimes M_1 \vdash E_2}{(E_1 -\!\!\diamond M_1) \otimes (E_2 -\!\!\diamond M_2) \vdash (E -\!\!\diamond M_1 \otimes M_2)}$$

2.3 Testing

The linear logic described so far is an intuitionistic one. It does not include a constant $\perp$ that resembles falsehood, or a negation-like involution $(\cdot)^\perp$. The notion of testing suggests useful $\perp$ and $(\cdot)^\perp$ constructs, and gives rise to a different account of assumption-guarantee specifications. We can view the environment of a process as a tester for the process. Tests start from a distinguished state α; and another state β is distinguished to represent the result of successful tests. A process p passes the test of q if p and q may yield the state β when they run in parallel, starting from α, and q fails p otherwise. A process succeeds if it may yield β when it runs in isolation, starting from α, and it fails otherwise. Thus, p passes the test of q if the parallel composition of p and q succeeds.

Failure is a safety property, and we write $\perp$ for the set of all processes that fail. A sort of negation can also be defined: $M^\perp$ is the set of all processes that fail M processes. Naturally, we are particularly interested in the propositions M such that $M = (M^\perp)^\perp$, which are called facts. These are the specifications that have sound and complete testers; they can be characterized explicitly with a simple set of closure conditions.

Certain expressions in this classical linear logic are reminiscent of assumption-guarantee specifications. In particular, $(E \wedge M^\perp)^\perp$ is the set of processes that fail all of the tests that M processes fail, provided these tests are from E. In other words, $(E \wedge M^\perp)^\perp$ includes all of the processes that cannot be distinguished from M processes in E environments (by E tests). It is analogous to the assumption-guarantee specification $E \Rightarrow M$, but the obvious analogues of the Composition Principle do not hold.

A small correction solves this problem. Let

$$E^+ = E \cup \{u(s, \beta) \mid u \in E,\ s \text{ a state}\}$$

The processes in E^+ behave like processes in E, except that they may pass the testee at any point. If E and M are facts, then

$$E \multimap\!\!\!\!\diamond\, M = (E^+ \cap M^\perp)^\perp$$

and the expected Composition Principle follows.

3 Intuitionistic Logic

The model that underlies the intuitionistic logic is a small variant of that in [AL90]; we refer the reader to this and previous works for additional motivation.

We assume given a nonempty set of states, **S**, and a nonempty set of agents, **A**. These sets are disjoint. A *behavior* is an alternating finite sequence of states and agents that both begins and ends with a state. It can be pictured as:

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \ldots s_{n-1} \xrightarrow{a_n} s_n$$

where each s_i is a state and each a_i is an agent. We identify states with the corresponding one-element sequences. If σ is a sequence, a an agent, and s a state, then $\sigma \xrightarrow{a} s$ denotes the concatenation σas. The set of all behaviors is denoted by $\mathcal{B}$.

A *safety property* is a set of behaviors closed under prefixes. The set of all safety properties is denoted by $\mathcal{S}_b$, and ordered by subset. It will be convenient to use the turnstile symbol $\vdash$ to denote the subset ordering. Safety properties, as we have defined them, are isomorphic to the safety properties of [AL90], for example, with the caveat that we have not yet treated invariance under stuttering. It is quite natural, and desirable, to add a straightforward condition of invariance under stuttering to our definitions, as first advocated by Lamport [Lam83b]. For simplicity, we do not do so at this point, but do give a full discussion below.

The *length* $|\sigma|$ of a behavior σ is the number of agents that occur in σ. If $0 \leq m \leq |\sigma|$ then $\sigma|_m$ is the prefix of σ of length m; if $m > |\sigma|$, then $\sigma|_m = \sigma$.

Proposition 1 *$\mathcal{S}_b$ is a complete Heyting algebra, where $\wedge$ is $\cap$, $\bigvee$ is $\bigcup$, and the associated implication is*

$$M_1 \rightarrow M_2 = \{\sigma \mid \forall n \geq 0.\text{ if } \sigma|_n \in M_1 \text{ then } \sigma|_n \in M_2\}$$

Proof As $\mathcal{S}_b$ is closed under finite intersections and arbitrary unions, the set-theoretic operations are the lattice-theoretic ones. For implication, note that

$$M_1 \rightarrow M_2 = \{\sigma \mid \forall n \geq 0.\, \sigma|_n \in (\mathcal{B} \backslash \mathcal{M}_\infty) \cup \mathcal{M}_\in\}$$

and so it is the greatest safety property contained in the Boolean implication. ∎

Hence, the algebra of safety properties is a model for intuitionistic logic. The next subsection discusses composition in this intuitionistic setting, and the following one adds the treatment of stuttering.

3.1 Composition

We say that the safety property M *constrains at most* the set of agents μ, and write $M \lhd \mu$, if: (i) if $s \in \mathbf{S}$ then $s \in M$; and (ii) if $\sigma \in M$, $s \in \mathbf{S}$, and

$a \in \bar{\mu}$, then $\sigma \xrightarrow{a} s \in M$. Note that if $M \lhd \mu$ then $(N \to M) \lhd \mu$ for every N, and that if $\mu \subseteq \nu$ and $M \lhd \mu$ then $M \lhd \nu$. The collection of safety properties that constrain at most μ is closed under non-empty joins and finite meets.

Further, let M_μ be the smallest superset of M that constrains at most μ. The definition of "constrains at most," in the form of a monotone closure condition, guarantees that such an M_μ exists. In fact, a behavior in M_μ is either a behavior in M extended with arbitrary $\bar{\mu}$ steps, or simply a behavior that consists exclusively of $\bar{\mu}$ steps. So $(\cdot)_\mu$ is a monotone closure operation. It commutes with arbitrary non-empty joins, and also with finite meets.

We are now in a position to formulate a version of the Composition Principle of [AL90] specialized to safety properties. If I is a set of states, we write $\hat{I}$ for the safety property $\{\sigma \mid \sigma$ begins with an element of $I\}$; such a safety property is an initial condition.

Theorem 1 (Composition Principle) *Let μ_i be sets of agents, let I_i and I be sets of states, and let $M_i \lhd \mu_i$ and $E_i \lhd \bar{\mu}_i$ (for $i = 1, n$). Suppose that $\hat{I} \vdash \bigwedge_i \hat{I}_i$ and $E \wedge \bigwedge_i M_i \vdash \bigwedge_i E_i$. Then*

$$\bigwedge_i (\hat{I}_i \wedge E_i \to M_i) \vdash \hat{I} \wedge E \to \bigwedge_i M_i \tag{1}$$

Proof We show by induction on the length of σ that if it is in the set on the left-hand side and also in $\hat{I} \wedge E$ then it is in M_i $(i = 1, n)$. In case the length is zero, this is immediate as $M_i \lhd \mu_i$. Otherwise, σ has the form $\sigma' \xrightarrow{a} s$. By induction hypothesis, σ' is in M_i. So if $a \notin \mu_i$, we get $\sigma \in M_i$ as $M_i \lhd \mu_i$. We are left with the case where $a \in \mu_i$. As $\sigma \in \hat{I} \wedge E$ we get $\sigma \in \hat{I}_i$. As $\sigma \in E \wedge \bigwedge_i M_i$, we get $\sigma' \in E \wedge \bigwedge_i M_i$, and hence $\sigma' \in E_i$ and $\sigma \in E_i$ (as $E_i \lhd \bar{\mu}_i$). So, finally, as we now have $\sigma \in \hat{I}_i \wedge E_i \to M_i$ and $\sigma \in \hat{I}_i \wedge E_i$, we get $\sigma \in M_i$ as required. ▮

The Composition Principle corresponds to that of [AL90] restricted to safety properties (once stuttering is taken into account). The principle is designed to be of direct use in applications. As such, it is rather complex, and we turn to finding simpler but equivalent versions. An immediate simplification is obtained by removing the initial conditions to obtain that if $M_i \lhd \mu_i$, $E_i \lhd \bar{\mu}_i$, and $E \wedge \bigwedge_i M_i \vdash \bigwedge_i E_i$, then

$$\bigwedge_i (E_i \to M_i) \vdash E \to \bigwedge_i M_i \tag{2}$$

This is evidently a special case of the principle. It also implies the principle as (1) follows from (2) and $\hat{I} \vdash \bigwedge_i \hat{I}_i$ by propositional reasoning. (By that we mean that if we treat (1), (2), and $\hat{I} \vdash \bigwedge_i \hat{I}_i$ as sequents in a suitable intuitionistic calculus, regarding the E, E_i, M_i, $\hat{I}$, $\hat{I}_i$ as propositional symbols, and $\wedge$ and $\rightarrow$ as logical connectives, then (1) can be derived from (2) and $\hat{I} \vdash \bigwedge_i \hat{I}_i$.)

It is instructive to consider the case $n = 1$ which amounts to the fact that if $E \wedge M_1 \vdash E_1$ then $(E_1 \rightarrow M_1) \vdash (E \rightarrow M_1)$. By propositional reasoning this is equivalent to the case where $E = (M_1 \rightarrow E)$, which can be written as:

$$(E_1 \rightarrow M_1) \wedge (M_1 \rightarrow E_1) \vdash M_1 \qquad (M_1 \lhd \mu, E_1 \lhd \bar{\mu}) \tag{3}$$

It turns out that the whole Composition Principle can be reduced to this case just using propositional reasoning. To show this, let us assume (3) and demonstrate the special case of the Composition Principle not involving initial conditions. We proceed by induction on n, with the base case having already been considered. For $n > 1$, assume that $E \wedge \bigwedge_i M_i \vdash \bigwedge_i E_i$. Then for any j (where $1 \leq j \leq n$) we have:

$$\begin{aligned}
\textstyle\bigwedge_i (E_i \rightarrow M_i) \wedge E \;\; & \vdash (E_j \rightarrow M_j) \wedge \textstyle\bigwedge_{i \neq j} (E_i \rightarrow M_i) \wedge E \\
& \vdash (E_j \rightarrow M_j) \wedge (E \wedge M_j \rightarrow \textstyle\bigwedge_{i \neq j} M_i) \wedge E \\
& \quad\;\; \text{(by induction hypothesis, since} \\
& \quad\;\; E \wedge M_j \wedge \textstyle\bigwedge_{i \neq j} M_i \vdash \textstyle\bigwedge_{i \neq j} E_i) \\
& \vdash (E_j \rightarrow M_j) \wedge (E \wedge M_j \rightarrow E_j) \wedge E \\
& \quad\;\; \text{(since by assumption } E \wedge M_j \wedge \textstyle\bigwedge_{i \neq j} M_i \vdash E_j) \\
& \vdash (E_j \rightarrow M_j) \wedge (M_j \rightarrow E_j) \\
& \vdash M_j \\
& \quad\;\; \text{(by (3))}
\end{aligned}$$

In short, we get $\bigwedge_i (E_i \rightarrow M_i) \wedge E \vdash M_j$ (for $j = 1, n$), and hence also $\bigwedge_i (E_i \rightarrow M_i) \vdash E \rightarrow \bigwedge_i M_i$ as desired.

If we allow the $(\cdot)_\mu$ operator in our statements, (3) can be further reduced to:

$$(M_{\bar{\mu}} \rightarrow M) \vdash M \qquad (M \lhd \mu) \tag{4}$$

This formula follows by propositional reasoning from (3) (taking $M = M_1$ and $M_{\bar{\mu}} = E_1$) and the fact that $M \vdash M_{\bar{\mu}}$. But (4) also implies (3), once we add to our propositional reasoning a fact about the $(\cdot)_\mu$ operator given by Lemma 1:

Lemma 1 *If M and E are safety properties and ν is a set of agents, then $M \rightarrow E \vdash M_\nu \rightarrow E_\nu$.*

Proof The proof is a simple chain of implications:

$$\begin{array}{lll} (M \rightarrow E) \wedge M_\nu & \vdash (M \rightarrow E)_\nu \wedge M_\nu & \\ & \vdash ((M \rightarrow E) \wedge M)_\nu & \text{(as } (\cdot)_\nu \text{ preserves intersections)} \\ & \vdash E_\nu & \text{(as } (\cdot)_\nu \text{ is monotone)} \end{array}$$

∎

Now to see that (3) follows from (4), suppose that $M \lhd \mu$, $E \lhd \bar{\mu}$, and calculate:

$$\begin{array}{lll} (E \rightarrow M) \wedge (M \rightarrow E) & \vdash (E \rightarrow M) \wedge (M_{\bar{\mu}} \rightarrow E_{\bar{\mu}}) & \text{(by Lemma 1)} \\ & \vdash M_{\bar{\mu}} \rightarrow M & \text{(since } E \lhd \bar{\mu}) \\ & \vdash M & \text{(by (4))} \end{array}$$

3.2 Stuttering

Two behaviors are stuttering equivalent if they differ only as regards the presence or absence of steps of the form $s \xrightarrow{a} s$. Formally, define stuttering equivalence as the least equivalence relation $\simeq$ on behaviors such that:

$$usasv \simeq usv \tag{5}$$

Orienting this equation from left to right we evidently obtain a strongly normalising Church-Rosser reduction system. The normal forms are the behaviors containing no stuttering steps. Write $\natural\sigma$ for the normal form of σ; it is the shortest behavior stuttering equivalent to σ.

Following [AL90] we concern ourselves with properties closed under $\simeq$. Let $\mathcal{S}t_b$ be the collection of safety properties closed under stuttering, and order it by inclusion. It turns out that $\mathcal{S}t_b$ is again a complete Heyting algebra with finite meets and arbitrary joins given set theoretically and the associated implication is the restriction of that for $\mathcal{S}_b$. The first part of these assertions is obvious; for the second we need to examine the relationship between the prefix ordering $\leq$ on behaviors and stuttering equivalence.

Lemma 2 *Suppose that $\sigma' \leq \sigma \simeq \tau$. Then there exists a τ' such that $\sigma' \simeq \tau' \leq \tau$.*

Proof Since $\sigma \simeq \tau, \tau$ can be obtained from σ by a sequence of steps of the form (5) or the converse. We prove the result for the case of one such step; an evident inductive argument then completes the proof. So first suppose that σ, τ have the forms $usasv$ and usv. Since $\sigma' \leq \sigma = usasv$ either $\sigma' \leq us$ or $us < \sigma'$. In the first case we have $\sigma' \leq \tau$ and so we can take $\tau' = \sigma'$. In the second case σ' must have the form $usasv'$ where $v' \leq v$ and we can take $\tau' = usv'$. It remains to consider the situation where σ, τ have the forms usv and $usasv$. Since $\sigma' \leq usv$ we have either $\sigma' \leq u$ (when we can take $\tau' = \sigma'$) or that σ' has the form usv' with $v' \leq v$ (when we can take $\sigma' = usasv'$). ∎

We can now check that if M_1 and M_2 are in $\mathcal{S}t_b$ then so is $M_1 \rightarrow M_2$ (where $\rightarrow$ is as defined above)—it follows that $\rightarrow$ is the intuitionistic implication in $\mathcal{S}t_b$. For this, suppose that $\sigma \simeq \tau \in M_1 \rightarrow M_2$. Suppose further that $\sigma\ |_n \in M_1$ for some $n \geq 0$. Then, by the Lemma, for some $\tau' \leq \tau$, $\sigma\ |_n \simeq \tau'$. We now have successively that: $\tau' \in M_1$ (as M_1 is closed under $\simeq$), $\tau' \in M_2$ (as $\tau \in M_1 \rightarrow M_2$), and $\sigma\ |_n \in M_2$ (as M_2 is also closed under $\simeq$). Hence, $\sigma \in M_1 \rightarrow M_2$.

The relation between $\mathcal{S}_b$ and $\mathcal{S}t_b$ is best explained by the map $\varphi : \mathcal{S}_b \rightarrow \mathcal{S}_b$ where $\varphi(M)$ is defined to be the least safety property containing M and closed under stuttering.

Proposition 2 *1. $\varphi(M) = \{\tau \mid \exists \sigma \in M. \tau \simeq \sigma\}$.*

2. φ is a monotone closure operation preserving all joins;

$\mathcal{S}t_b$ is its partial order of fixed-points.

Proof

1. It suffices to show that the right-hand side is a safety property and this is immediate from Lemma 2.

2. Obvious.

∎

As the lattice-theoretic operations in $\mathcal{S}t_b$ are the set-theoretic ones, the collection of stuttering-closed safety properties that constrain at most μ is closed under non-empty joins and finite meets; and we also know that if M is such a property then so is $N \rightarrow M$, for any N in $\mathcal{S}t_b$. For M in $\mathcal{S}t_b$, let M^μ be the least superset of M in $\mathcal{S}t_b$ which constrains at most μ.

Proposition 3 *1. $M^\mu = \varphi(M_\mu)$.*

2. $(\cdot)^\mu$ is a monotone closure operation that preserves non-empty joins and finite meets.

Proof

1. It suffices to show that $\varphi(M_\mu)$ constrains at most μ. First $S \subseteq M_\mu \subseteq \varphi(M_\mu)$. Second, suppose that $\sigma \in \varphi(M_\mu), a \notin \mu$, and $s \in S$. Then $\sigma \simeq$ some τ in M_μ. But now we have that $\sigma \xrightarrow{a} s \simeq \tau \xrightarrow{a} s \in M_\mu$ and so $\sigma \xrightarrow{a} s \in \varphi(M_\mu)$.

2. Evidently $(\cdot)^\mu$ is a monotone closure operation. It preserves non-empty joins as both φ and $(\cdot)_\mu$ do. All closure operations preserve the top element. For binary meets, we just prove $\varphi(M_\mu) \cap \varphi(N_\mu) \subseteq \varphi((M \cap N)_\mu)$, the other direction being a trivial consequence of monotonicity. So suppose that $\sigma \simeq \tau$ in M_μ and $\sigma \simeq \gamma$ in N_μ. It is straightforward to show, for any M in $\mathcal{S}t_b$, that if $\sigma \in M_\mu$ then $\natural\sigma \in M_\mu$. So we get that $\sigma \simeq \natural\sigma \in (M_\mu \cap N_\mu)$, as $\natural\sigma = \natural\tau = \natural\gamma$. But $(M_\mu \cap N_\mu) = (M \cap N)_\mu$ as $(\cdot)_\mu$ preserves binary intersections, and so we have $\sigma \in \varphi((M \cap N)_\mu)$, as required.

∎

The Composition Principle goes through with stuttering-invariance exactly as it did before. One need only note that $\hat{I}$ is in $\mathcal{S}t_b$, and that meet, join, and implication for $\mathcal{S}t_b$ are the restrictions of the corresponding $\mathcal{S}_b$ operations. All the reductions of the principle to simpler ones also go through exactly as before, as they are either propositional or use the expected corresponding facts for $(\cdot)^\mu$, *viz.* that $M \vdash M^\mu$ and $M \to E \vdash M^\mu \to E^\mu$—the proof of the latter being perfectly analogous to that of Lemma 1.

4 Intuitionistic Linear Logic

In this section we develop the intuitionistic linear logic proposed in the overview. The study of classical linear logic is postponed to the next section.

We assume given only a set of states $\mathbf{S}$; there is no notion of agent in this calculus. A *transition* is a pair of states. A *process* is a prefix-closed set of sequences of transitions. (Note that the empty sequence ε is allowed.) The set of all processes is denoted by $\mathcal{P}$. It is partially ordered by $\subseteq$ and as such it is a complete semilattice, which is to say that it has least upper bounds of all subsets . For two given complete semilattices L and M, we write $f : L \to_l M$, and say that f is *linear*, meaning that f preserves all least upper bounds, that is $f(\bigvee X) = \bigvee_{x \in X} f(x)$ for all subsets X of L. The set

$L \to_l M$of linear functions from L to M itself forms a complete semi-lattice under the so-called *pointwise* ordering: $f \leq g$ iff $f(x) \leq g(x)$ for all x in X.[1]

Complete semilattices L can be viewed as cpos (partial orders with a least element and least upper bounds (lubs) of directed sets) endowed with a continuous semilattice operation, $+$, such that $x \leq x+y$. (Note that $x+y$ must be $x \vee y$, the lub in the partial order.) This kind of algebra was found in [HP85b] to be appropriate to the study of lower powerdomains, which are just free algebras of that kind. Following ideas in [HP87], we now define a *safety property* on such a structure as a non-empty Scott-closed subset closed under the semilattice operation. The idea is that safety properties correspond to "nothing going wrong" and so: first, nothing can go wrong with $\perp$, the least element, as that corresponds to nothing happening; second, if nothing can go wrong with each element of a directed set X then nothing can go wrong with $\bigvee X$ either, as "going wrong" is continuous; third, if nothing can go wrong with x or y then nothing can go wrong with $x+y$ as all that can happen with $x+y$ is whatever happens with x or whatever happens with y. This can be formalised by taking as a way of going wrong a linear map $f : L \to_l I$ where I is the two-point complete semilattice, $\{\perp, \top\}$, with $\perp \leq \top$. The collection of elements of L where f does not "go wrong" is $f^{-1}(\perp)$ and this yields an isomorphism

$$\mathcal{S}(L) \cong (L \to_l I)^{op}$$

where we order the collection of safety properties $\mathcal{S}(L)$ by subset. Considering again our desire to work with elementary means, note that every safety property $X \subseteq L$ has a largest element, *viz.* $m(X) =_{def} \bigvee X$.

Proposition 4 *The function $m : \mathcal{S}(L) \to L$ is an isomorphism of partial orders.*

[1]It is possible to view $\mathcal{P}$ also as the solution to a domain equation, by choosing a category of domains tailored to nondeterminism, in the fashion of [HP85b]. Specifically, working in the category of complete semilattices, one finds that $\mathcal{P}$ is the initial solution to the equation:

$$\mathcal{P} \cong (\wp(S) \to_l \wp(S) \otimes \mathcal{P})_\perp$$

where the lifting operator $(\cdot)_\perp$ adds a new least element, and the tensor product is defined by a universal property: there is a universal bilinear map $L \times M \xrightarrow{\otimes} L \otimes M$. Thus $\mathcal{P}$ can be obtained by the methods available in domain theory, and as such it provides a kind of resumption useful for the semantics of non-terminating processes. Its simple representation as the prefix-closed sets of transition sequences allows us to work with it using very elementary mathematical means.

Proof The function is clearly monotone. Its inverse is $m^{-1}(x) = \{y \mid y \leq x\}$ which is also monotone. ▮

This isomorphism, together with the above remarks, yields an isomorphism $L^{op} \cong (L \rightarrow_l I)$ which is part of the well-known self-duality of the category of complete semilattices [Joh82]. We say the process p *satisfies* a safety property X, and write $p \models X$, if and only if $p \in X$. Under the isomorphism this is the case iff $p \subseteq m(X)$.

We will work with $\mathcal{P}$ rather than the more complex $\mathcal{S}(\mathcal{P})$. First, $\mathcal{P}$ is again a complete Heyting algebra with the lattice-theoretic operations being the set-theoretic ones and the associated implication being

$$M_1 \rightarrow M_2 = \{u \mid \forall n \geq 0. \text{ if } u|_n \in M_1 \text{ then } u|_n \in M_2\}$$

where the prefix $u|_n$ is defined as usual for sequences. The empty set (falsehood) is written 0, and the set of all transition sequences (truth) is written $\top$.

If p_1 and p_2 are two processes, their parallel composition is $p_1 || p_2$, where $||$ is the language shuffle operator. Conjunction is no longer the logical correlate of parallelism, however. If $p \models X$ and $q \models Y$ it is not true in general that $p \parallel q \models X \wedge Y$. Rather, in order to treat parallelism, we define a new operator on safety properties by:

$$X \otimes Y = \{p \parallel q \mid p \models X, q \models Y\}^s$$

where $(A)^s$ is the least safety property containing A.

Proposition 5 $m(X \otimes Y) = m(X) \parallel m(Y)$.

Proof If $p \models X$ and $q \models Y$ then $p \subseteq m(X), q \subseteq m(Y)$, and so $X \otimes Y = \{r \mid r \subseteq m(X) \parallel m(Y)\}$. ▮

Working with $\mathcal{P}$ in place of $\mathcal{S}(\mathcal{P})$ we take $\otimes$ on $\mathcal{P}$ to be $\parallel$. One sees that $\otimes$ commutes with arbitrary joins in $\mathcal{P}$ and gives a commutative monoid, with unit the null process, $1 = \{\varepsilon\}$. In other words, we have:

Proposition 6 $(\mathcal{P}, \bigcup, 1, \otimes)$ *is a commutative quantale, where* $1 = \{\varepsilon\}$.

The associated quantalic implication is then given by

$$M_1 \multimap M_2 = \{u \mid (\{u\} \ || \ M_1) \subseteq M_2\}$$

It follows immediately that the algebra of safety specifications provides a model of intuitionistic linear logic [Yet90, Ros90]. Parallel composition is the multiplicative conjunction operation, while $\wedge$ and $\vee$ are the additives.

The exponential operator ! is uniquely, but trivially, determined. If $1 \subseteq M$ then $1 \subseteq !M$, and in addition $!M \subseteq 1$, by the general properties of !, so we get $!M = 1$. On the other hand, if $1 \subseteq M$ is false, the only possibility is $M = 0$, and $!M = 0$, as in every model $!M \subseteq M$.

Instead, a nontrivial $(\cdot)^*$ operation is available: M^* is defined as $\bigvee_i M^i$, where M^i is the i-fold parallel composition of M with itself, and it represents an arbitrary number of M processes running in parallel.

Composition

A transition sequence is *chained* if it is of the form

$$(s_1, s_2)\,(s_2, s_3)\ \ldots\ (s_{n-2}, s_{n-1})\,(s_{n-1}, s_n)$$

(The sequences ε and (s_1, s_2) are chained.) Intuitively, chained transition sequences correspond to runs of a system by itself, with no interference from the environment. We write $u \smile_I v$ if u and v have a chained shuffle, beginning with an element of I.

Assumption-guarantee specifications are made possible by a new connective $\multimap\!\!\!\diamond_I$. We set:

$$(M)^\dagger_I = \{u \mid \exists v \in M.\ u \smile_I v\}$$

and

$$M_1 \multimap\!\!\!\diamond_I M_2 = (M_1)^\dagger_I \rightarrow M_2$$

The definition says that if a prefix u of a sequence in $M_1 \multimap\!\!\!\diamond_I M_2$ has a chained shuffle beginning in I with a sequence in M_1, then u is in M_2. Hence, the sequences in $M_1 \multimap\!\!\!\diamond_I M_2$ cannot be distinguished from sequences in M_2 by an M_1 environment as regards computations beginning in I.

It seems rather unfortunate to have to introduce a ternary connective where, furthermore, one of the arguments comes from a set of propositions different from the other two. We are missing a principled explanation of this connective arising from the nature of processes. In Section 5 we give one account of it, relating it to the work using intuitionistic logic.

We can now formulate a version of the Composition Principle in intuitionistic linear logic.

Theorem 2 (Composition Principle) *Let $M_i, E_i \in \mathcal{P}$, and let I_i and I be sets of states (for $i = 1, n$). Set $M'_i = \bigotimes_{j \neq i} M_j$. Suppose that $I \subseteq \bigcap I_i$ and $E \otimes M'_i \vdash M_i \multimap\!\!\diamond_{I_i} E_i$. Then*

$$\bigotimes_i (E_i \multimap\!\!\diamond_{I_i} M_i) \vdash E \multimap\!\!\diamond_I \bigotimes_i M_i$$

Rather than prove the soundness of this rule directly, we will progressively reduce it to simpler principles, and prove the simplest. First, since $\multimap\!\!\diamond_I$ is antimonotone in I the principle is equivalent to the case where $I_i = I$, for $i = 1, n$. We now keep I fixed and often omit it, and write, for example, $E \multimap\!\!\diamond M$.

It is straightforward to reduce the principle to the binary case. The unary case follows from the binary case by taking $M_2 = 1$, $E_2 = E \otimes M_1$, and using the fact that $N \vdash M \multimap\!\!\diamond N$, for all M, N. For $n \geq 2$ we proceed by induction. The base case is given, so suppose $n \geq 3$ and $E \otimes M'_i \vdash M_i \multimap\!\!\diamond E_i$ for $i = 1, n$. So for $i = 2, n$ we have $(E \otimes M_1) \otimes \bigotimes_{j \geq 2, j \neq i} M_j \vdash M_i \multimap\!\!\diamond E_i$ and, by induction hypothesis, we get that $\bigotimes_{i \geq 2} (E_i \multimap\!\!\diamond M_i) \vdash E \otimes M_1 \multimap\!\!\diamond \bigotimes_{i \geq 2} M_i$. In order to prove $\bigotimes_i (E_i \multimap\!\!\diamond M_i) \vdash (E \multimap\!\!\diamond \bigotimes_i M_i)$ it is now enough to prove $(E_1 \multimap\!\!\diamond M_1) \otimes (E \otimes M_1 \multimap\!\!\diamond M'_1) \vdash E \multimap\!\!\diamond M_1 \otimes M'_1$. But, since we have $E \otimes M'_1 \vdash M_1 \multimap\!\!\diamond E_1$, this follows from the binary case, taking M_2 to be M'_1 and E_2 to be $E \otimes M_1$ (and using again the fact that $N \vdash M \multimap\!\!\diamond N$).

More surprisingly, the general case reduces further to the unary case, which is:

$$\frac{E \vdash M_1 \multimap\!\!\diamond E_1}{(E_1 \multimap\!\!\diamond M_1) \vdash (E \multimap\!\!\diamond M_1)}$$

Note that this is equivalent simply to:

$$(E_1 \multimap\!\!\diamond M_1) \vdash (M_1 \multimap\!\!\diamond E_1) \multimap\!\!\diamond M_1 \tag{6}$$

using the antimonotonicity of $M \multimap\!\!\diamond N$ in its first argument.

The proof that the binary case reduces to the unary case has two parts. The first part applies not only to the binary case but also to the general case; it consists in reducing the general case to its instance where $E_i = E \otimes M'_i$:

$$\bigotimes_i (E \otimes M'_i \multimap\!\!\diamond M_i) \vdash E \multimap\!\!\diamond \bigotimes_i M_i$$

In the second part, this instance is derived from the unary case for $n = 2$:

$$(E \otimes M_2 \multimap\!\!\diamond M_1) \otimes (E \otimes M_1 \multimap\!\!\diamond M_2) \vdash E \multimap\!\!\diamond (M_1 \otimes M_2)$$

For the first part of the proof, assume that $E \otimes M_i' \vdash M_i \multimap\!\!\diamond\, E_i$. The antimonotonicity of $\multimap\!\!\diamond$ then gives:

$$\bigotimes_i ((M_i \multimap\!\!\diamond\, E_i) \multimap\!\!\diamond\, M_i) \vdash E \multimap\!\!\diamond \bigotimes_i M_i$$

and (6) gives:

$$\bigotimes_i (E_i \multimap\!\!\diamond\, M_i) \vdash \bigotimes_i ((M_i \multimap\!\!\diamond\, E_i) \multimap\!\!\diamond\, M_i)$$

The general principle follows by transitivity.

We need first a little more about the logic of $\multimap\!\!\diamond$ for the second part of the proof:

Lemma 3 *Let $A, B, E \in \mathcal{P}$. Then*

$$A \otimes (A \otimes E \multimap\!\!\diamond_I B) \vdash E \multimap\!\!\diamond_I A \otimes B$$

Proof It is enough to take w in $A \otimes (A \otimes E \multimap\!\!\diamond_I B)$ and x in E such that $w \smile_I x$ and show that w is in $A \otimes B$. So taking such a w and x, we get first that w is a shuffle of an element u of A with an element v of $A \otimes E \multimap\!\!\diamond\, B$. Next, u and x must have a shuffle, y, say, such that $v \smile_I y$. But then y is in $A \otimes E$ and so as v is in $A \otimes E \multimap\!\!\diamond_I B$, we get that v is in B. So as w is a shuffle of u (in A) with v (in B) we get w in $A \otimes B$ as required. ▮

We may now calculate that:

$$(E \otimes M_2 \multimap\!\!\diamond\, M_1) \otimes (E \otimes M_1 \multimap\!\!\diamond\, M_2)$$

$$\vdash (E \otimes M_2 \multimap\!\!\diamond\, M_1) \otimes ((M_2 \multimap\!\!\diamond\, E \otimes M_1) \multimap\!\!\diamond\, M_2)$$

(by the unary case)

$$\vdash (E \otimes M_2 \multimap\!\!\diamond\, M_1) \otimes (((E \otimes M_2 \multimap\!\!\diamond\, M_1) \otimes E) \multimap\!\!\diamond\, M_2)$$

(as $(E \otimes M_2 \multimap\!\!\diamond\, M_1) \otimes E \vdash M_2 \multimap\!\!\diamond\, E \otimes M_1$ by Proposition 3)

$$\vdash E \multimap\!\!\diamond\, ((E \otimes M_2 \multimap\!\!\diamond\, M_1) \otimes M_2)$$

(by Proposition 3)

$$\vdash E \multimap\!\!\diamond\, (E \multimap\!\!\diamond\, M_1 \otimes M_2)$$

(by Proposition 3)

$$\vdash E \multimap\!\!\diamond\, M_1 \otimes M_2$$

We are left with the task of proving the unary case. The proof requires an induction on the length of transition sequences and it is noteworthy that no other truth of the logic we have so far shown (such as Proposition 3) has done so. Thus all the induction is, as it were, concentrated into this one case.

Proof We have to show that $(E \multimap\!\!\diamond_I M) \vdash (M \multimap\!\!\diamond_I E) \multimap\!\!\diamond_I M$ for any E and M in $\mathcal{P}$. In case $M = 0$ the result follows immediately as $0 \multimap\!\!\diamond_I E = \top$. Otherwise it is enough to show that if u is in $(E \multimap\!\!\diamond_I M)$ and v is in $(M \multimap\!\!\diamond_I E)$ and $u \smile_I v$ then u is in M; we show this by induction on $\mid u \mid + \mid v \mid$. If this is 0 then $u = \varepsilon \in M$. Otherwise let w be a complete shuffle of u and v beginning in I.

There are two cases. In the first, $w = w_1(s,t)$, $v = v_1(s,t)$, and w_1 is a complete shuffle of u and v_1. By induction hypothesis, as $\mid u \mid + \mid v_1 \mid < \mid u \mid + \mid v \mid$, we then get u in M. In the second case, $w = w_1(s,t)$, $u = u_1(s,t)$, and w_1 is a complete shuffle of u_1 and v. By induction hypothesis as $\mid u_1 \mid + \mid v \mid < \mid u \mid + \mid v \mid$, we get u_1 in M. But as v is in $(M \multimap\!\!\diamond_I E)$ and $u_1 \smile_I v$, we get v in E. But then as u is in $(E \multimap\!\!\diamond_I M)$ and $u \smile_I v$, we get u in M. ∎

It also seems possible to obtain variants of the principle that apply to the composition of an arbitrary number of like processes that depend on one another, in an environment E. For example, one can show:

$$\frac{(E \otimes M^* \multimap\!\!\diamond_I M)^*}{E \multimap\!\!\diamond_I M^*}$$

There does not seem to be an analogous rule in the intuitionistc framework of the previous section. check this is true!

5 Classical Linear Logic

Once one has a quantale, there is a well-known and straightforward way to interpret classical linear logic; one chooses an element $\perp$ and, setting $x^\perp = x \multimap \perp$, one works with the $(\cdot)^{\perp\perp}$-closed elements [Ros90]. Here we show that by an appropriate choice of $\perp$ we can also find a Composition Principle within the framework of classical linear logic. Abramsky [Vic88] has suggested that the choice of $\perp$ could depend on a notion of testing, and could be taken to be the set of processes that, when run by themselves, can be seen as failing (*i.e.*, not passing the test). In this way one would have an internalised notion of testing where tests were represented by processes: a process p would pass a test q iff $(p \mid\mid q) \notin \perp$.

Here we will make this suggestion concrete for safety properties; every test q will yield a safety property $q \multimap \bot$ so that p does not pass q iff $p \in q \multimap \bot$. One may think of the safety property yielded by $q \multimap \bot$ as being the failure to pass q. Once one focuses on the $(\cdot)^{\bot\bot}$-closed subsets, all safety properties will be of this kind as then $M = M^{\bot} \multimap \bot$ holds.

It is instructive to begin with an external approach to testing and for this we provide a semantical analogue to some of the testing ideas of de Nicola and Hennessy [Hen88], adapted to the present context of processes and safety specifications. Let α, β be two distinct entities not in $\mathbf{S}$, and put $\mathbf{S}' = \mathbf{S} \cup \{\alpha, \beta\}$. One can think of α and β as being starting and stopping states for an external test scenario. Let $\mathcal{P}'$ be the processes over $\mathbf{S}'$; these will be the tests. Clearly notions and results applying to $\mathbf{S}$ and $\mathcal{P}$ extend to $\mathbf{S}'$ and $\mathcal{P}'$. For p in $\mathcal{P}$ and r in $\mathcal{P}'$, we say that p *passes* r iff there are $u \in p, v \in r$ such that $u \frown v$, meaning that there are prefixes u', v' of u and v that have a chained shuffle starting in α and ending in β. Note the element of possibility here: only the existence of such a pair u, v is required; p will not pass r iff there is no such possibility.

Now one has a natural testing preorder on processes in $\mathcal{P}$:

$$p \leq_{\mathcal{P}} q \ \text{ iff } \ \forall r \in \mathcal{P}'.(p \text{ passes } r \ \supset \ q \text{ passes } r)$$

In order to characterize this preorder some definitions are needed. Let $\sqsupseteq$ be the least preorder on transition sequences over $\mathbf{S}$ such that:

$$uv \sqsupseteq u$$
$$u(r,s)(s,t)v \sqsupseteq u(r,t)v$$
$$uv \sqsupseteq u(s,s)v$$

and for $u = (s_1, t_1) \dots (s_n, t_n)$ (with $n \geq 0$) such a transition sequence, set $u^{\#} = (\alpha, s_1)(t_1, s_2) \dots (t_{n-1}, s_n)(t_n, \beta)$, and set $\varepsilon^{\#} = (\alpha, \beta)$.

Proposition 7 *1. $u \frown u^{\#}$.*

2. Suppose $v \sqsupseteq u \frown w$. Then $v \frown w$.

3. Suppose $v \frown u^{\#}$. Then $v \sqsupseteq u$.

Proof Parts 1 and 2 are easy to prove and we just consider part 3. If $u = \varepsilon$ then (trivially) $v \sqsupseteq u$. Otherwise u has the form $(s_1, t_1) \dots (s_n, t_n)$ with $n > 0$ and since $v \frown u^{\#}$, v must have the form $v_1 \dots v_n v'$ where, for $i = 1, n$, either $v_i = \varepsilon$ and $s_i = t_i$ or v_i is a chained transition sequence beginning in s_i and ending in t_i. In either case $v_i \sqsupseteq (s_i, t_i)$ and so $v \sqsupseteq u$. ∎

Theorem 3 *$p \leq_{\mathcal{P}} q$ iff $\forall u \in p. \exists v \in q. u \sqsubseteq v$.*

Proof First suppose that $p \leq_{\mathcal{P}} q$ and $u \in p$. Let $r = \{w \mid w \leq u^{\#}\} \in \mathcal{P}'$. Then as $u \frown u^{\#}$, by the Proposition, we get that p passes r, and since $p \leq_{\mathcal{P}} q$ so does q. Hence $v \frown w$ for some v in q and some $w \leq u^{\#}$. But then $v \frown u^{\#}$ and so $v \sqsupseteq u$, by the Proposition. Conversely, suppose that $\forall u \in p. \exists v \in q. u \sqsubseteq v$ and that p passes r. Then $u \frown w$ for some $u \in p, w \in r$; taking a $v \in q$ such that $u \sqsubseteq v$, we get $v \frown w$ by the Proposition, and so q passes r. ▮

Note that it follows from the last part of the Proposition that the largest process $\leq_{\mathcal{P}}$-equivalent to a given process p is $\{u \mid \exists v \in p. u \sqsubseteq v\}$.

To internalise, we simply work with $\mathcal{P}'$ rather than with $\mathcal{P}$ and extend the above notions, taking $u \frown v$ for u, v transition sequences over $\mathbf{S}'$ to mean, as before, that there are prefixes u', v' of u, v which have a chained shuffle from α to β and writing p *passes* r also for p in $\mathcal{P}'$ and correspondingly extending the testing preorder — the extension is written as: $\leq_{\mathcal{P}'}$. To pass to classical linear logic, we take $\perp$ to be the safety property of those processes that do not contain a chained transition sequence from α to β and so one has indeed that:

$$p \text{ does not pass } r \text{ iff } (p \mid\mid r) \subseteq \perp$$

Under the isomorphism of processes and safety properties, $\perp$ becomes

$$\{w \mid \text{no prefix of } w \text{ is a chained transition sequence from } \alpha \text{ to } \beta\}$$

and we get for any safety property (under the isomorphism):

$$M^{\perp} = \{u \mid \forall v \in M. \neg(u \frown v)\}$$

Note that p does not pass r iff $r \mid\mid p \vdash \perp$ iff $r \vdash p^{\perp}$, so $p^{\perp}$ is the largest test p does not pass. The internal and external views are linked up as follows:

Proposition 8 *1. For any p, q in $\mathcal{P}'$, $p \leq_{\mathcal{P}'} q$ iff $q^{\perp} \vdash p^{\perp}$.*

2. The largest process $\leq_{\mathcal{P}'}$-equivalent to p is $p^{\perp\perp}$.

Proof

1. Suppose $p \leq_{\mathcal{P}'} q$. Then as q does not pass $q^{\perp}$, neither does p and so $p \mid\mid q^{\perp} \vdash \perp$. Therefore, $q^{\perp} \vdash p^{\perp}$. Conversely, suppose $q^{\perp} \vdash p^{\perp}$ and q does not pass r, so $q \mid\mid r \vdash \perp$. Then $r \vdash q^{\perp} \vdash p^{\perp}$ and so $p \mid\mid r \vdash \perp$.

2. By the first part, p is $\leq_{\mathcal{P}'}$-equivalent to q iff $p^{\perp} = q^{\perp}$. But then p and $p^{\perp\perp}$ are equivalent (as we always have, for any choice of $\perp$, that $p^{\perp} = p^{\perp\perp\perp}$) and if p and q are $\leq_{\mathcal{P}'}$-equivalent then $q \subseteq q^{\perp\perp}$ (true for any choice of $\perp$) $= p^{\perp\perp}$.

∎

The next task is to extend the characterization of the testing preorder to the whole of $\mathcal{P}'$. We extend $\sqsupseteq$ to a relation $\sqsupseteq'$ which is the least preorder on $\mathbf{S}'$-transition sequences such that:

$$uv \sqsupseteq' u$$

$$u(r,s)(s,t)v \sqsupseteq' u(r,t)v$$

$$uv \sqsupseteq' u(s,s)v$$

$$(\alpha,\alpha)u \sqsupseteq' u$$

$$u(s,\beta) \sqsupseteq' u(s,t)v$$

and $(\cdot)^{\#}$ is defined exactly as before. Note that $u^{\#\#} = (\alpha,\alpha)u(\beta,\beta) \equiv' u$ (where we take $\equiv'$ to be the equivalence relation induced by $\sqsupseteq'$).

The analogue of Proposition 7 holds, with $\sqsupseteq'$ replacing $\sqsupseteq$:

Proof As before, parts 1 and 2 are easy and we concentrate on part 3. So suppose that $v \frown u^{\#}$. The case $u = \varepsilon$ is trivial and so we can take u to have the form $(s_1,t_1)(s_2,t_2)\dots(s_n,t_n)$ (with $n > 0$). Then $u^{\#}$ is $(\alpha,s_1)(t_1,s_2),(t_2,s_3)\dots(t_{n-1},s_n)(t_n,\beta)$. Some prefixes v', w of $v, u^{\#}$ have a chained transition sequence from α to β; we take w and then v' to be as short as possible. Then β is either the last state in w or the last state in v'.

In the first case, either $w = u^{\#}$ or $w = (\alpha,s_1)(t_1,s_2)\dots(t_m,s_{m+1})$ with $0 \leq m < n$ and $s_{m+1} = \beta$. In the first of these cases v' will have the form $v_0v_1\dots v_n$ where v_0 is ε or is a chained transition sequence from α to α, and for $i = 1,n$ each v_i is ε and $s_i = t_i$ or v_i is a chained transition sequence from s_i to t_i. But then $v \sqsupseteq' v' \sqsupseteq' (\alpha,\alpha)(s_1,t_1)\dots(s_n,t_n) \sqsupseteq' u$. In the second of these cases v' has the form $v_0v_1\dots v_m$ with v_0 and the v_i as before. Then, $v \sqsupseteq' v' \sqsupseteq' (\alpha,\alpha)(s_1,t_1)\dots(s_m,t_m) \sqsupseteq' (s_1,t_1)\dots(s_m,t_m) \sqsupseteq' (s_1,t_1)\dots(s_m,t_m)(\beta,\beta) \sqsupseteq' (s_1,t_1)\dots(s_m,t_m)(\beta,t_{m+1})\dots(s_n,t_n) = u$.

In the second case, since we chose first w and then v' as short as possible, w has the form $(\alpha,s_1)(t_1,s_2)\dots(t_n,s_{n+1})$ with $0 \leq m < n$ and v' has the form $v_0v_1\dots v_mv_{m+1}$ with v_0 and the v_i as before (for $i = 1,m$) and with v_{m+1} a chained transition sequence from s_{m+1} to β. But then $v \sqsupseteq' (\alpha,\alpha)(s_1,t_1)\dots(s_m,t_m)(s_{m+1},\beta) \sqsupseteq' u$. ∎

The symmetry of testers and testees in the $\frown$ relation enables a pleasing reformulation of the first three parts of the analogue of Proposition 7:

Proposition 9 *$v \frown u$ iff $v \sqsupseteq' u^{\#}$.*

Proof If $v \frown u$ then as $u^{\#\#} \equiv' u$ we get by part 2 of the analogue of Proposition 7, and the symmetry of $\frown$ that $v \frown u^{\#\#}$. So by part 3, $v \sqsupseteq' u^{\#}$.

Conversely if $v \sqsupseteq' u^{\#}$ then as $u \frown u^{\#}$ by part 1, we get $u^{\#} \frown u$ by symmetry and then $v \frown u$ by part 2. ∎

The analogue of Theorem 3 holds, with the analogous proof:

$$p \leq_{\mathcal{P}'} q \text{ iff } \forall u \in p.\exists v \in q.u \sqsubseteq' v$$

and so the facts, being the maximal $\leq_{\mathcal{P}'}$-equivalence classes by Proposition 8, are exactly the $\sqsubseteq'$-downwards closed sets. It follows that the lattice-theoretic operations are the set-theoretic ones. We can rewrite the above formula for $M^{\perp}$ (when M is a fact) using Proposition 9:

Proposition 10 *$M^{\perp} = \{u \mid u^{\#} \notin M\}$.*

Proof Taking negations we see that $\exists v \in M.u \frown v$ iff $\exists v \in M.v \sqsupseteq' u^{\#}$ iff $u^{\#} \in M$ (as M is a fact). ∎

The preorder $\sqsupseteq'$ and the map $(\cdot)^{\#}$ interact in a natural way:

$$u^{\#\#} \equiv' u$$

$$u \sqsupseteq' v \text{ iff } v^{\#} \sqsupseteq' u^{\#}$$

(For the last, note that if $u \sqsupseteq' v$ then $u \frown v^{\#}$ and so $v^{\#} \sqsupseteq' u^{\#}$, by Proposition 9). We call any such map on a preorder an *involution*. The case where the preorder is a set, say U, is well-known to the relevance logicians who instead of quantales considered quasi-fields of subsets of U closed under the quasi-complement operation:

$$\neg X = U \backslash g(X)$$

If we divide out by the equivalence relation $\equiv'$ we obtain a quasi-field of sets $(g([u]_{\equiv'}) = [u^{\#}]_{\equiv'})$ over $U = \{[u]_{\equiv'}\}$ isomorphic to our lattice of facts. The sets in the quasi-field are the subsets of U downwards closed in the partial order $\sqsubseteq' / \equiv'$.

We have already noted that the facts are closed under the set-theoretic operations and so the additives $\wedge, \vee, \top, 0$ retain their set theoretic definitions. However $\otimes$ and 1 must be redefined, and $M \otimes N$ is now $(M \parallel N)^{\perp\perp}$ and 1 is $\{\varepsilon\}^{\perp\perp}$. At the level of transition-sequences one can make a further connection to relevance logic, this time considering R-frames ([Dun86] p.47). Taking U to be the collection of equivalence classes as above one obtains a structure $(U, R, [\varepsilon], g)$ where $R([u],[v],[w])$ iff there are $u' \sqsubseteq' u, v' \sqsubseteq' v$, and a shuffle, x, of u' and v' such that $w \sqsubseteq' x$. This satisfies all the requirements to be an R-frame, except for (the undesired) idempotence. Given any such structure $(U, R, 0, g)$ one obtains a quantale $(Q, \otimes, 1)$ for classical linear logic where Q is the collection of $\leq$-downwards closed subsets of U. One takes $u \leq v$ iff $R(1, v, u)$ and $A \otimes B = \{z \mid \exists x \in A, y \in R.R(x, y, z)\}$, $1 = \{x \mid x \leq 0\}$, and $\perp = \{x \mid x \leq g(0)\}$. Starting from the $(U, R, [\varepsilon], g)$ as above one obtains the quantale for classical linear logic considered in this paper.

Composition

To be consistent with the testing idea of starting computations from α, we fix the set I to be $\{\alpha\}$, and write $M \multimap\!\!\diamond\, N$ for $M \multimap\!\!\diamond_I N$. As suggested in subsection 2.2, let

$$E^+ = E \cup \{u(s, \beta) \mid u \in E,\ s \in \mathbf{S}'\}$$

Note that E^+ is not a fact in general, even when E is a fact.

Lemma 4 *Let E be a fact. Suppose $w \in E$ and $v \smile_{\{\alpha\}} w$.Then $v^\sharp \in E^+$*

Proof First suppose that $v = \varepsilon$. Then $v^\sharp$ is (α, β) which is in E^+ as ε is in E (since w is). Suppose now instead that $w = \varepsilon$. Then v is a chained transition sequence from α to some state t, and so $v^\sharp$ has the form $u(t, \beta)$, where is u a sequence of *stutters*, that is transition pairs of the form (s, s). But then: $w \sqsupseteq' \varepsilon \sqsupseteq' u$ and so u is in E, and $v^\sharp$ is in E^+.

We may now therefore suppose that neither v nor w are ε. There are two cases according as to whether the chained shuffle of v and w starts with a transition from w, or one from v. In the first case there is a prefix w' of w, states $s_0, \ldots, s_{n+1}$ (with $s_0 = \alpha$) and $t_0, \ldots, t_n$, and also $v_0, \ldots, v_n$ and $w_0, \ldots, w_n$ such that $v = v_0 \cdots v_n$, $w' = w_0 \cdots w_n$, and for $i = 0, n$, v_i is a chained transition sequence from t_i to s_{i+1}, and w_i is a chained transition sequence from s_i to t_i.Now $v^\sharp$ has the form $(s_0, t_0)u_0 \cdots (s_n, t_n)u_n(s_{n+1}, \beta)$ where for $i = 0, n$, u_i is a sequence of stutters. But then

$$w \sqsupseteq' w' \sqsupseteq' (s_0, t_0)u_0 \cdots (s_n, t_n)u_n$$

as w_i is a chained transition sequence from s_i to t_i, and so $v^\sharp$ is in E^+.

The last case last case is similar. Here there is a prefix w' of w, states $s_0, \ldots, s_{n+1}$ (with $s_0 = \alpha$) and $t_0, \ldots, t_{n+1}$, and also $v_0, \ldots, v_{n+1}$ and $w_0, \ldots, w_n$ such that $v = v_0 \cdots v_{n+1}$, $w' = w_0 \cdots w_n$, and for $i = 0, n+1$, v_i is a chained transition sequence from s_i to t_i, and for $i = 0, n$, w_i is a chained transition sequence from t_i to s_{i+1}. Now $v^\sharp$ has the form

$$(s_0, s_0)u_0(t_0, s_1) \cdots u_n(t_n, s_{n+1})u_{n+1}(t_{n+1}, \beta)$$

where for $i = 0, n$, u_i is a sequence of stutters. But then

$$w \sqsupseteq' w' \sqsupseteq' (s_0, s_0)u_0(t_0, s_1) \cdots u_n(t_n, s_{n+1})u_{n+1}$$

as w_i is a chained transition sequence from t_i to s_{i+1}, and so $v^\sharp$ is in E^+, concluding the proof. ▮

We may now obtain:

Proposition 11 *If E and M are facts then*

$$E \multimap\!\!\diamond\, M = (E^+ \cap M^\perp)^\perp$$

Proof It is fairly straightforward to show that $E \multimap\!\!\diamond\, M \subseteq (E^+ \cap M^\perp)^\perp$, directly from the definitions. Suppose that $u \in E \multimap\!\!\diamond M$, that $v \in (E^+ \cap M^\perp)$, to prove that it is not the case that $u \frown v$. If $u \frown v$ then some prefix u' of u has a chained shuffle from α to β with some prefix v' of v. Choose such a u' and v' with u'as short as possible.

Since $v \in E^+$, $v' \in E^+$ and so either $v' \in E$ or $v' = v''(t, \beta)$ for some $v'' \in E$ and some state t. In the first case, $v' \in E$ and so $u' \in M$, using the assumption that $u \in E \multimap\!\!\diamond M$. But as we also have that $u' \frown v$ and $v \in M^\perp$, this is a contradiction.

In the second case, u' and v'' have a complete shuffle starting from α, by the choice of u' and v'. This again gives us that $u' \in M$, and we have a contradiction as before.

For the converse, assume that $u \in (E^+ \cap M^\perp)^\perp$, that v is a prefix of u, and that $v \smile_{\{\alpha\}} w$ for some $w \in E$, to show that $v \in M$. Then

$v \in (E^+ \cap M^\perp)^\perp$ and by Lemma 4, $v^\sharp \in E^+$. Now assume for the sake of contradiction that $v \notin M$. Then $v^\sharp \in M^\perp$, by Proposition 10. But now $v \frown v^\sharp$ is in contradiction with $v \in (E^+ \cap M^\perp)^\perp$. ▮

So it is not necessary to redefine $\multimap\!\!\diamond$ in the classical logic. The direct analogue of the Composition Principle for the intuitionistic case holds, *viz.*:

Theorem 4 (Composition Principle) *Let M_i and E_i be facts (for $i = 1, n$). Set $M_i' = \bigotimes_{j \neq i} M_j$. Suppose that $E \otimes M_i' \vdash M_i \multimap\!\!\diamond E_i$. Then*

$$\bigotimes_i (E_i \multimap\!\!\diamond M_i) \vdash E \multimap\!\!\diamond \bigotimes_i M_i$$

where we are taking the classical interpretation of the tensor products. This version of the Composition Principle follows straightforwardly from the intuitionistic one using Proposition 11 and propositional reasoning. Further, the propositional reasoning used in the discussion of the intuitionistic case remains valid here, including the analogue of Proposition 3.

6 Comparisons

The intuitionistic logic and the linear logic are based on different connectives, and on different semantic models, yet there is a fairly straightforward translation between them. Let σ be a behavior

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots s_{n-1} \xrightarrow{a_n} s_n$$

Let $t_\mu(\sigma)$ be the subsequence of $(s_0, s_1) \ldots (s_{n-1}, s_n)$ such that the transition (s_{i-1}, s_i) appears in $t_\mu(\sigma)$ if and only if $a_i \in \mu$. The *runs of an element p of $\mathcal{P}$ with identity μ* are the behaviors σ such that $t_\mu(\sigma) \in p$. This yields a map $r_\mu : \mathcal{P} \to \mathcal{S}_b$. It has a left-inverse $s_\mu : \mathcal{S}_b \to \mathcal{P}$, which maps a set of behaviors to the most general process that implements this set of behaviors. The operations of the two calculi can then be related, and for example

$$M_1 \otimes M_2 = s_{\mu \cup \nu}(r_\mu(M_1) \cap r_\nu(M_2))$$

and again,

$$M_1 \multimap\!\!\diamond M_2 = s_\mu(r_\nu(M_1) \to r_\mu(M_2))$$

where μ, ν are nonempty, disjoint sets of agents.

The intuitionistic logic captures an external view of processes, via their behaviors. The notation $M \triangleleft \mu$ makes it possible to express who is the subject of a specification. Linear logic specifications describe a process at a time, and hence the notion of "constrains at most" is unnecessary. On the other hand, it becomes more difficult to express that one process is the complete environment of another, and that the system that they form is closed. Such closed systems are essential in the notion of testing, which then helps in the analysis of assumption-guarantee specifications.

Acknowledgements

Luca Cardelli encouraged this work in its initial phase, and tried some of the ideas in a CCS setting. David Israel suggested the connection with relevance logic. We enjoyed discussions about testing with Rocco de Nicola. We derived understanding and inspiration from Leslie Lamport and Vaughan Pratt.Cynthia Hibbard provided editorial help.

References

[Abr79] K. Abrahamson. Modal logic of concurrent nondeterministic programs. In *International Symposium on Semantics of Concurrent Computation*, Evian-les-Baines, July 1979.

[Abr89] Samson Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 1989.

[AL90] Martín Abadi and Leslie Lamport. Composing specifications. Research Report 66, Digital Equipment Corporation Systems Research Center, 1990. A preliminary version appeared in [dBdRR90].

[AP91] Martín Abadi and Gordon Plotkin. A logical view of composition and refinement. In *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Programming Languages*, pages 323–332, January 1991.

[AV90] Samson Abramsky and Steve Vickers. Quantales, observational logic, and process semantics. Technical report, Imperial College, January 1990.

[BKP84] Howard Barringer, Ruurd Kuiper, and Amir Pnueli. Now you may compose temporal logic specifications. In *Sixteenth Annual*

ACM Symposium on Theory of Computing, pages 51–63. ACM, April 1984.

[Dam88] Mads Dam. Relevance logic and concurrent computation. In *Proceedings of the Third Symposium on Logic in Computer Science*, pages 178–185. IEEE, July 1988.

[dBdRR90] J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors. *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, volume 430 of *Lecture Notes in Computer Science*, Berlin, 1990. Springer-Verlag.

[DP91] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, 1991.

[Dun86] J. Michael Dunn. Relevance logic and entailment. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 3, pages 117–224. D. Reidel Publishing Co., 1986.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Hen88] Matthew Hennessy. *Algebraic Theory of Processes*. MIT Press, Cambridge, Massachusetts, 1988.

[HP85a] David Harel and Amir Pnueli. On the development of reactive systems. In K. R. Apt, editor, *Logics and models of concurrent systems*, volume F13 of *NATO ASI Series*, pages 477–498. Springer-Verlag, 1985.

[HP85b] Matthew Hennessy and Gordon Plotkin. Full abstraction for a simple parallel programming language. In J. Becvar, editor, *Proceedings of 8th Mathematical Foundations of Computer Science Conference, Olomouc, Czechoslovakia*, volume 74 of *Lecture Notes in Computer Science*, Berlin, 1985. Springer-Verlag.

[HP87] Matthew Hennessy and Gordon Plotkin. Finite conjunctive nondeterminism. In K. Voss, H. J. Genrich, and G. Rozenberg, editors, *Concurrency and Nets*, pages 233–244, Berlin, 1987. Springer-Verlag.

[Joh82] P. T. Johnstone. *Stone Spaces*. Cambridge University Press, Cambridge, 1982.

[Lam83a] Leslie Lamport. Specifying concurrent program modules. *ACM Transactions on Programming Languages and Systems*, 5(2):190–222, April 1983.

[Lam83b] Leslie Lamport. What good is temporal logic? In R. E. A. Mason, editor, *Information Processing 83: Proceedings of the IFIP 9th World Congress*, Paris, September 1983. IFIP, North Holland.

[Lam89] Leslie Lamport. A simple approach to specifying concurrent systems. *Communications of the ACM*, 32(1):32–45, January 1989.

[MC81] Jayadev Misra and K. Mani Chandy. Proofs of networks of processes. *IEEE Transactions on Software Engineering*, SE-7(4):417–426, July 1981.

[MOM89] Narciso Martí-Oliet and José Meseguer. From Petri nets to linear logic. Technical Report SRI-CSL-89-4R2, SRI International, December 1989.

[Pnu84] Amir Pnueli. In transition from global to modular temporal reasoning about programs. In Krzysztof R. Apt, editor, *Logics and Models of Concurrent Systems*, NATO ASI Series, pages 123–144, Berlin, October 1984. Springer-Verlag.

[Ros90] K.I. Rosenthal. *Quantales and their applications*, volume 234 of *Pitman Research Notes in Mathematics*. Longman, Harlow, 1990.

[Sta85] Eugene W. Stark. A proof technique for rely/guarantee properties. In S. N. Maheshwari, editor, *Foundations of Software Technology and Theoretical Computer Science*, volume 206 of *Lecture Notes in Computer Science*, pages 369–391, Berlin, 1985. Springer-Verlag.

[Vic88] Steve Vickers. Samson Abramsky on linear process logics. Foundation Workshop Notes, October-November 1988.

[Yet90] David N. Yetter. Quantales and (noncommutative) linear logic. *Journal of Symbolic Logic*, 55(1):41–64, March 1990.