Electronic Notes in Theoretical Computer Science 20 (1999)
URL: http://www.elsevier.nl/locate/entcs/volume20.html 23 pages

Categorical Structure of Asynchrony

Peter Selinger

Department of Mathematics
University of Michigan
Ann Arbor, MI 48109-1109, U.S.A.

Abstract

We investigate a categorical framework for the semantics of asynchronous communication in networks of parallel processes. Abstracting from a category of asynchronous labeled transition systems, we formulate the notion of a categorical model of asynchrony as a uniformly traced monoidal category with diagonals, such that every morphism is total and the focus is equivalent to a category of complete partial orders. We present a simple, non-deterministic, cpo-based model that satisfies these requirements, and we discuss how to refine this model by an observational congruence. We also present a general construction of passing from deterministic to non-deterministic models, and more generally, from non-linear to linear structure on a category.

Introduction

In this paper, we investigate the categorical semantics of asynchronous communication in networks of parallel processes. Informally, communication is said to be synchronous if messages are sent and received simultaneously, requiring a 'handshake' between sender and receiver. It is asynchronous if messages travel through a communication medium with possible delay, such that the sender does not have to wait for the receiver to be available. Asynchronous communication is a common assumption in the design of distributed and concurrent systems, and particularly in the design of large-scale networks. It is embodied in such process calculi as the asynchronous π-calculus [5], the programming language PICT [12], the join calculus [6], and the actors model [3].

There is no shortage of semantic models for networks of communicating processes. But it seems fair to say that there is a lack of unifying principles behind the multitude of models that have been suggested. The goal of the present paper is to use category theory to isolate some general, and hopefully useful, properties of models of asynchronous communication. These properties might be taken as a basis for classifying and relating some of the existing models. The axioms that we suggest are neither claimed to be complete, nor in their ultimate form. Rather, they are intended as a working model for what constitutes a category of asynchronous processes, a notion to be refined and hopefully improved in the future.

To motivate our categorical treatment, we begin by reviewing a particular, concrete model of asynchronous communication which was presented in [14]. In this model, which is based on labeled transition systems, asynchronous channels are modeled explicitly as non-blocking, unbounded capacity buffers. A process is called asynchronous if its input and output behave as though they are passing through such a buffer. These asynchronous processes form the morphisms of a traced monoidal category **Buf**. We take this category as a prototype, and by abstracting from its properties, we arrive at our notion of a categorical model of asynchrony as a traced monoidal category with certain additional structure.

One feature of our semantics is that it accounts for non-deterministic as well as for deterministic processes. The classical interpretation of *deterministic* asynchronous processes is due to Kahn, who has shown that such processes can be adequately modeled as Scott-continuous maps between certain complete partial orders [11]. There is no such canonical semantics in the non-deterministic case. Nevertheless, Kahn's ideas play a central role here, because in any model of non-determinism, the deterministic processes form an important subcategory, which we call the *focus* of the category. We adopt a version of Kahn's principle by requiring that the focus of a categorical model of asynchrony is equivalent to a category of complete partial orders.

1 A category of asynchronous processes

In this section, we describe a category of asynchronous labeled transition systems. This category, which was first introduced in [14], will serve us to motivate the more abstract categorical definitions of the later sections.

1.1 Labeled transition systems and bisimulation

Throughout, we will write RS for the composition of binary relations, *i.e.*, $xRSz$ if for some y, xRy and ySz.

Definition 1.1 A ***labeled transition system*** is a tuple $\mathbf{S} = \langle S, A, \to, s_0\rangle$, where S is a set of ***states***, A is a set of ***actions***, $\to \subseteq S \times A \times S$ is a ***transition relation***, and $s_0 \in S$ is the ***initial state***.

We also write $|\mathbf{S}| = S$ for the set of states. The set of actions A is also called the ***type*** of the labeled transition system $\mathbf{S}$, and we write $\mathbf{S} : A$. For the transition relation, we write $s \xrightarrow{\alpha} t$ instead of $\langle s, \alpha, t\rangle \in \to$, and the intended interpretation is that the system can get from state s to state t by performing an action α.

Sometimes we will consider the set A to contain a special action $\tau \in A$, called the ***silent*** or ***unobservable*** action. Let $\xrightarrow{\tau}{}^{*}$ denote the reflexive, transitive closure of $\xrightarrow{\tau}$. Then we write $s \overset{\alpha}{\Rightarrow} t$ for $s \xrightarrow{\tau}{}^{*} \xrightarrow{\alpha} \xrightarrow{\tau}{}^{*} t$, if $\alpha \neq \tau$, and $s \overset{\tau}{\Rightarrow} t$ for $s \xrightarrow{\tau}{}^{*} t$.

Definition 1.2 Let $\mathbf{S}$ and $\mathbf{T}$ be labeled transition systems of type A. A relation $R \subseteq |\mathbf{S}| \times |\mathbf{T}|$ is called a ***strong bisimulation*** between $\mathbf{S}$ and $\mathbf{T}$ if for all $\alpha \in A$, one has $R\xrightarrow{\alpha} \subseteq \xrightarrow{\alpha}R$ and $R^{-1}\xrightarrow{\alpha} \subseteq \xrightarrow{\alpha}R^{-1}$. In diagrams:

$$\begin{array}{ccc} s & R & t \\ & & \downarrow{\scriptstyle\alpha} \\ & & t' \end{array} \Rightarrow \exists s'.\ \begin{array}{ccc} s & R & t \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\alpha} \\ s' & R & t' \end{array} \qquad \text{and} \qquad \begin{array}{ccc} s & R & t \\ {\scriptstyle\alpha}\downarrow & & \\ s' & & \end{array} \Rightarrow \exists t'.\ \begin{array}{ccc} s & R & t \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\alpha} \\ s' & R & t'. \end{array}$$

A relation $R \subseteq |\mathbf{S}| \times |\mathbf{T}|$ is a ***weak bisimulation*** between $\mathbf{S}$ and $\mathbf{T}$ if for all $\alpha \in A$, one has $R\xrightarrow{\alpha} \subseteq \overset{\alpha}{\Rightarrow}R$ and $R^{-1}\xrightarrow{\alpha} \subseteq \overset{\alpha}{\Rightarrow}R^{-1}$. In diagrams:

$$\begin{array}{ccc} s & R & t \\ & & \downarrow{\scriptstyle\alpha} \\ & & t' \end{array} \Rightarrow \exists s'.\ \begin{array}{ccc} s & R & t \\ {\scriptstyle\alpha}\Downarrow & & \downarrow{\scriptstyle\alpha} \\ s' & R & t' \end{array} \qquad \text{and} \qquad \begin{array}{ccc} s & R & t \\ {\scriptstyle\alpha}\downarrow & & \\ s' & & \end{array} \Rightarrow \exists t'.\ \begin{array}{ccc} s & R & t \\ {\scriptstyle\alpha}\downarrow & & \Downarrow{\scriptstyle\alpha} \\ s' & R & t' \end{array}$$

It is well-known that there exists a maximal strong bisimulation between $\mathbf{S}$ and $\mathbf{T}$, which we denote by $\sim_{\mathbf{S},\mathbf{T}}$, and a maximal weak bisimulation, which we denote by $\approx_{\mathbf{S},\mathbf{T}}$. We often omit the subscripts. Recall that $\sim_{\mathbf{S},\mathbf{S}}$ and $\approx_{\mathbf{S},\mathbf{S}}$ are equivalence relations on $|\mathbf{S}|$. We say that two states $s \in |\mathbf{S}|$ and $t \in |\mathbf{T}|$ are ***strongly (weakly) bisimilar*** if $s \sim t$ ($s \approx t$). Finally, labeled transition systems $\mathbf{S}$ and $\mathbf{T}$ are said to be strongly (weakly) bisimilar if their initial states are strongly (weakly) bisimilar. Strong and weak bisimulation each form an equivalence relation on the class of all labeled transition systems.

Remark 1.3 Note that $R \subseteq |\mathbf{S}| \times |\mathbf{T}|$ is a weak bisimulation if and only if for all $\alpha \in A$, $R\overset{\alpha}{\Rightarrow} \subseteq \overset{\alpha}{\Rightarrow}R$ and $R^{-1}\overset{\alpha}{\Rightarrow} \subseteq \overset{\alpha}{\Rightarrow}R^{-1}$.

1.2 Input, output, and sequential composition

We want to use labeled transition system to model processes which interact with some environment through input and output. To this end, we put some additional structure on the set of actions A. Fix two distinct constants *in* and *out*. If X and Y are any sets, we define a set of actions $X \to Y$ as the following disjoint union of sets:

$$X \to Y := \{\mathit{in}\} \times X + \{\mathit{out}\} \times Y + \{\tau\}.$$

A labeled transition system of type $X \to Y$ is also called an ***agent*** of this type. Thus, an agent may have three kinds of transitions: ***input transitions***, written $s \xrightarrow{\mathit{in}\,x} s'$; ***output transitions***, written $s \xrightarrow{\mathit{out}\,y} s'$, and ***silent transitions*** $s \xrightarrow{\tau} s'$. An agent which contains no silent transitions is called ***τ-free***.

Example 1.4 Let $X = \{a\}$. We define a one-place buffer $\mathcal{I} : X \to X$. The states are $|\mathcal{I}| = \{\bot, a\}$, the initial state is $\bot$, and the transitions are $\bot \xrightarrow{in\,a} a$ and $a \xrightarrow{out\,a} \bot$. The one-place buffer can be represented by the following state graph, where the initial state is circled:

$$\mathcal{I} = \begin{tikzcd} \bot \arrow[r, bend left, "in\,a"] & a \arrow[l, bend left, "out\,a"] \end{tikzcd}.$$

Example 1.5 For any X, we can define an unbounded buffer $\mathcal{B}_X : X \to X$. The set of states is X^{**}, the free commutative monoid generated by X. The elements of X^{**} are finite multisets in X, which we denote by u, v, w. The initial state is the empty word ϵ, and the transitions are

$$u \xrightarrow{in\,x} ux \qquad ux \xrightarrow{out\,x} u.$$

Note that this buffer does not preserve the order of input and output.

As suggested by the arrow notation for types, agents can be composed. If $\mathbf{S} : X \to Y$ and $\mathbf{T} : Y \to Z$, then the sequential composition $\mathbf{S};\mathbf{T} : X \to Z$ is defined by the following data: It has states $|\mathbf{S}| \times |\mathbf{T}|$ and initial state $\langle s_0, t_0 \rangle$. The transitions are given by the following rules:

$$\frac{s \xrightarrow{\alpha}_{\mathbf{S}} s' \qquad \alpha \text{ not output}}{\langle s, t\rangle \xrightarrow{\alpha}_{\mathbf{S};\mathbf{T}} \langle s', t\rangle}, \qquad \frac{t \xrightarrow{\alpha}_{\mathbf{T}} t' \qquad \alpha \text{ not input}}{\langle s, t\rangle \xrightarrow{\alpha}_{\mathbf{S};\mathbf{T}} \langle s, t'\rangle},$$

$$\frac{s \xrightarrow{out\,y}_{\mathbf{S}} s' \qquad t \xrightarrow{in\,y}_{\mathbf{T}} t'}{\langle s, t\rangle \xrightarrow{\tau}_{\mathbf{S};\mathbf{T}} \langle s', t'\rangle}.$$

Lemma 1.6 (i) *Sequential composition is associative up to isomorphism.*

(ii) $\mathbf{S} \sim \mathbf{S}'$ *and* $\mathbf{T} \sim \mathbf{T}' \Rightarrow \mathbf{S};\mathbf{T} \sim \mathbf{S}';\mathbf{T}'$.

(iii) $\mathbf{S} \approx \mathbf{S}'$ *and* $\mathbf{T} \approx \mathbf{T}' \Rightarrow \mathbf{S};\mathbf{T} \approx \mathbf{S}';\mathbf{T}'$.

Example 1.4, continued. If $\mathcal{I}$ is the one-place buffer from before, then $\mathcal{I};\mathcal{I}$ is a two-place buffer:

$$\mathcal{I};\mathcal{I} \quad = \quad \begin{tikzcd} \langle\bot,\bot\rangle \arrow[d, "in\,a"'] & \langle\bot,a\rangle \arrow[l, "out\,a"'] \arrow[d, "in\,a"] \\ \langle a,\bot\rangle \arrow[ur, "\tau"'] & \langle a,a\rangle \arrow[l, "out\,a"] \end{tikzcd} \quad \approx \quad \begin{tikzcd} \bullet \arrow[r, bend left, "in\,a"] & \bullet \arrow[l, bend left, "out\,a"] \arrow[r, bend left, "in\,a"] & \bullet \arrow[l, bend left, "out\,a"] \end{tikzcd}$$

Example 1.5, continued. The unbounded buffer $\mathcal{B}_X$ is idempotent up to weak bisimulation, *i.e.*, $\mathcal{B}_X;\mathcal{B}_X \approx \mathcal{B}_X$.

We remark that agents, as described in this section, do not form a category under sequential composition. The problem is that there are no identity morphisms. However, we will obtain a category of asynchronous agents in the next section.

1.3 Asynchronous agents

So how are we going to model asynchrony? The idea is to use unbounded buffers to model the asynchronous behavior of a process. From the viewpoint of an outside observer, it makes no difference whether we think of asynchrony as a property of the *medium* through which we communicate with a process, or whether we think of it as a property of the process itself. In the latter case, we simply think of the medium as a part of the process that we are trying to model.

Definition 1.7 An agent $\mathbf{S} : X \to Y$ is ***out-buffered*** if $\mathbf{S} \approx \mathbf{S}; \mathcal{B}_Y$. It is ***in-buffered*** if $\mathbf{S} \approx \mathcal{B}_X; \mathbf{S}$. We will call an agent ***asynchronous*** if it is in- and out-buffered.

Note that, because of the idempotency of the buffer, an agent $\mathbf{S}$ is asynchronous iff $\mathbf{S} \approx \mathcal{B}_X; \mathbf{S}; \mathcal{B}_Y$. Also, asynchronous agents are closed under composition. Moreover, the buffers $\mathcal{B}_X : X \to X$ act as identity morphisms for this composition, up to weak bisimulation. Thus, the weak bisimulation classes of asynchronous agents form the morphisms of a category **Buf**.

Definition 1.8 The category **Buf** has as its objects sets, and as its morphisms weak bisimulation classes of asynchronous agents $\mathbf{S} : X \to Y$.

1.4 Axioms for asynchrony

The in- and out-buffered agents have an interesting direct characterization in terms of properties of labeled transition systems. Up to weak bisimulation, the out-buffered, respectively, in-buffered agents are precisely those that satisfy the axioms in Table 1, respectively, Table 2. In stating these axioms, we use the convention that variables are implicitly existentially quantified if they occur only on the right-hand-side of an implication, and all other variables are implicitly universally quantified.

Theorem 1.9 (Characterization of in- and out-buffered agents [14])

(i) *An agent* $\mathbf{S}$ *is out-buffered if and only if* $\mathbf{S} \approx \mathbf{T}$ *for some* $\mathbf{T}$ *satisfying the axioms in Table 1.*

(ii) *An agent* $\mathbf{S}$ *is in-buffered if and only if* $\mathbf{S} \approx \mathbf{T}$ *for some* $\mathbf{T}$ *satisfying the axioms in Table 2.*

Remark 1.10 A standard technique for giving denotational semantics to agents up to weak bisimulation is to work with saturated agents. An agent is ***saturated*** if for any transition $s \stackrel{\alpha}{\Rightarrow} t$, there already exists a transition $s \xrightarrow{\alpha} t$. Obviously, for saturated agents weak and strong bisimulation coincide. However, saturation is not a useful technique in our framework of asynchrony. It is not difficult to see that any saturated agent that satisfies the properties in Tables 1 and 2 must be weakly bisimilar to a τ-free agent.

Table 1
First-order axioms for out-buffered agents

$$\begin{tikzcd} s \arrow[r, "out\,y"] & s' \arrow[d, "\alpha"] \\ & t \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "out\,y"] \arrow[d, "\alpha"'] & s' \arrow[d, "\alpha"] \\ s'' \arrow[r, "out\,y"] & t \end{tikzcd}$$

output-commutativity (ob1)

$$\begin{tikzcd} s \arrow[r, "out\,y"] \arrow[d, "\alpha"'] & s' \\ s'' \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "out\,y"] \arrow[d, "\alpha"'] & s' \arrow[d, "\alpha"] \\ s'' \arrow[r, "out\,y"] & t \end{tikzcd}$$

where $\alpha \neq out\,y$

output-confluence (ob2)

$$\begin{tikzcd} s \arrow[r, "out\,y"] \arrow[d, "out\,y"'] & s' \\ s'' \end{tikzcd} \Rightarrow s' = s''$$

output-determinacy (ob3)

Table 2
First-order axioms for in-buffered agents

$$\begin{tikzcd} s \arrow[r, "\alpha"] & s' \arrow[d, "in\,x"] \\ & t \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "\alpha"] \arrow[d, "in\,x"'] & s' \arrow[d, "in\,x"] \\ s'' \arrow[r, "\alpha"] & t \end{tikzcd}$$

input-commutativity (ib1)

$$\begin{tikzcd} s \arrow[r, "in\,x"] \arrow[d, "\alpha"'] & s' \\ s'' \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "in\,x"] \arrow[d, "\alpha"'] & s' \arrow[d, "\alpha"] \\ s'' \arrow[r, "in\,x"] & t \end{tikzcd}$$

input-confluence (ib2)

$$\begin{tikzcd} s \arrow[r, "in\,x"] \arrow[d, "in\,x"'] & s' \\ s'' \end{tikzcd} \Rightarrow s' = s''$$

input-determinacy (ib3)

$$s \quad \Rightarrow \quad s \xrightarrow{in\,x} t$$

input-receptivity (ib4)

1.5 Parallel composition and feedback

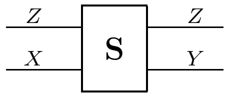
What other operations, besides sequential composition, do we have on agents? To begin with, there is an obvious notion of parallel composition (without interaction). Let $X + X'$ denote the disjoint union of sets X and X'. Then given two agents $\mathbf{S} : X \to Y$ and $\mathbf{T} : X' \to Y'$, we can define an agent $\mathbf{S}{+}\mathbf{T} : X{+}X' \to Y{+}Y'$ as follows: The states are given by $|\mathbf{S}{+}\mathbf{T}| = |\mathbf{S}| \times |\mathbf{T}|$, with initial state $\langle s_0, t_0 \rangle$. The transitions are given by the following rules:

$$\frac{s \xrightarrow{\alpha}_{\mathbf{S}} s'}{\langle s, t \rangle \xrightarrow{\alpha}_{\mathbf{S}+\mathbf{T}} \langle s', t \rangle}, \qquad \frac{t \xrightarrow{\alpha}_{\mathbf{T}} t'}{\langle s, t \rangle \xrightarrow{\alpha}_{\mathbf{S}+\mathbf{T}} \langle s, t' \rangle}.$$

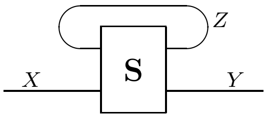
This operation defines a symmetric monoidal structure on the category **Buf**.

There is also a natural operation of feedback for agents, *i.e.*, of feeding the output of an agent back into it as input. This operation allows us to construct networks with loops.

An agent $\mathbf{S} : X + Z \to Y + Z$ can be pictured as a box with input and output wires as in the following diagram:



We want to define another agent $\mathrm{Tr}_Z\,\mathbf{S} : X \to Y$ by connecting the output wire of type Z of $\mathbf{S}$ to the corresponding input wire like this:



Formally, the agent $\mathrm{Tr}_Z\,\mathbf{S}$ is defined as follows: its states and initial state are the same as those of $\mathbf{S}$. The transitions are given by the two rules:

$$\frac{s \xrightarrow{\alpha}_{\mathbf{S}} s' \qquad \alpha \neq \mathit{in}\,z, \mathit{out}\,z}{s \xrightarrow{\alpha}_{\mathrm{Tr}_Z\,\mathbf{S}} s'}, \qquad \frac{s \xrightarrow{\mathit{out}\,z}_{\mathbf{S}} s' \qquad s' \xrightarrow{\mathit{in}\,z}_{\mathbf{S}} s''}{s \xrightarrow{\tau}_{\mathrm{Tr}_Z\,\mathbf{S}} s''}.$$

Asynchronous agents are closed under this feedback operation. In fact, as we will see in the next section, the feedback operation defines a traced monoidal structure on the category **Buf**.

Sequential composition, parallel composition, and feedback are the basic operations required for building arbitrary compositional networks. Of course, there are other useful operations on agents that one might wish to consider. However, many of them can be expressed in terms of primitive agents and the above basic operations.

2 Traced monoidal categories

Traced monoidal categories, introduced by Joyal, Street, and Verity [10], are an extension of symmetric monoidal categories with a notion of loops. Traces arise in many different areas of mathematics, where they go by names such as contraction, Markov trace, or braid closure. In computer science, traced monoidal categories have been applied to model feedback in process algebra [1] and cyclic data flow graphs [7,9]. As usual, we denote by $\mathbf{C}(A, B)$ the hom-set (or more generally the hom-class) of morphisms from A to B in a category $\mathbf{C}$. In a symmetric monoidal category, we denote the symmetry morphism by $c_{XY} : X \otimes Y \to Y \otimes X$. The unit of the tensor is denoted by I.

Definition 2.1 (Joyal, Street, Verity [10]) A ***traced monoidal category*** $\langle \mathbf{C}, \otimes, \mathrm{Tr} \rangle$ is a symmetric monoidal category $\langle \mathbf{C}, \otimes \rangle$, together with a family of operations

$$\mathrm{Tr}_X : \mathbf{C}(A \otimes X, B \otimes X) \to \mathbf{C}(A, B),$$

satisfying the following four axioms:

(i) Naturality. $\mathrm{Tr}_X(g \otimes \mathrm{id}_X; f; h \otimes \mathrm{id}_X) = g; \mathrm{Tr}_X\, f; h$.

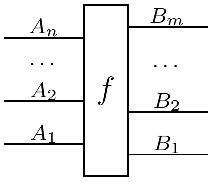
(ii) Strength. $\mathrm{Tr}_X(g \otimes f) = g \otimes \mathrm{Tr}_X\, f$.

(iii) Symmetry sliding. $\mathrm{Tr}_Y(\mathrm{Tr}_X(f; \mathrm{id}_B \otimes c_{XY})) = \mathrm{Tr}_X(\mathrm{Tr}_Y(\mathrm{id}_A \otimes c_{XY}; f))$.

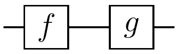
(iv) Yanking. $\mathrm{Tr}_X(c_{XX}) = \mathrm{id}_X$.

Note: in the original paper on traced monoidal categories, Joyal, Street, and Verity list additional axioms, such as Vanishing and Superposing. These additional axioms have been found to be redundant; see *e.g.* [9].

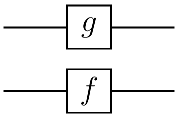
The axioms for a traced monoidal category, in the above equational form, are not very palatable. The meaning of the axioms is more easily understood in the following graphical notation. We write a morphism $f : A_1 \otimes \ldots \otimes A_n \to B_1 \otimes \ldots \otimes B_m$ as a box with wires as follows:



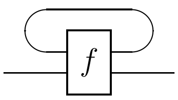
The intuition behind this notation is to think of f as a process with input channels $A_1, \ldots, A_n$ and output channels $B_1, \ldots, B_m$. We usually omit the labels on wires, and sometimes we combine several parallel wires into one. The operations of composition, tensor, and trace, along with the identity and symmetry morphisms, are written as follows:

Composition: $f; g =$ 

Identity: $\mathrm{id} =$

Tensor: $f \otimes g =$ 

Symmetry: $c =$ 

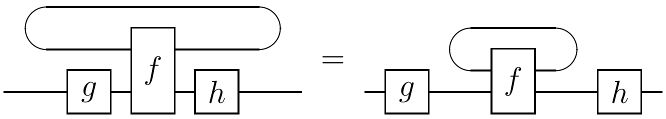
Trace: $\mathrm{Tr}\, f =$ 

In this notation, the four axioms of traced monoidal categories are shown in Table 3. Sometimes, as in the axiom of strength, we use a dashed box to emphasize a particular subgraph. These dashed boxes are not really part of the graphical language; they just serve as an illustration.

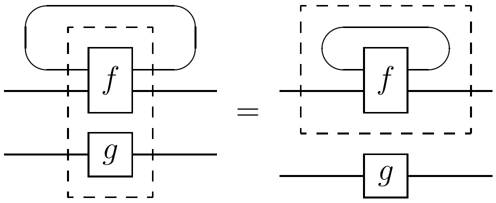
The fundamental property of traced monoidal categories is that the axioms of traced monoidal categories are sound and complete for graph isomor-

Table 3
The axioms of traced monoidal categories

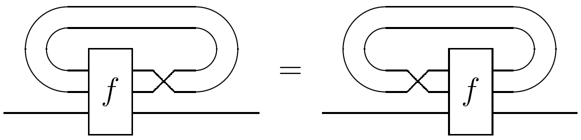
Naturality:



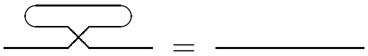
Strength:



Symmetry sliding:

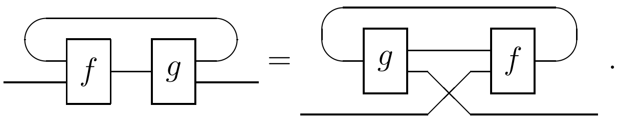


Yanking:



---

phism in this graphical language. In other words, to demonstrate that some equation holds in all traced monoidal categories, it suffices to manipulate the corresponding graphs. For example, to show that for all $f : A \otimes B \to C$ and $g : C \to D \otimes B$, the equation $\mathrm{Tr}_B(f;g) = \mathrm{Tr}_C(\mathrm{id}_A \otimes g; c_{AD} \otimes \mathrm{id}_B; \mathrm{id}_D \otimes f)$ holds, it suffices to see that the following two graphs are isomorphic:



2.1 Some examples of traced monoidal categories

1. The category **Buf** from Section 1 is traced monoidal.

2. Consider the category **Rel** of sets and relations, with the tensor $X + Y$ that is given on objects by disjoint union of sets. This tensor is actually a categorical product on the category **Rel**. We can define a trace as follows: for

$R : X + Z \to Y + Z$, define $\operatorname{Tr} R : X \to Y$ via

$$x(\operatorname{Tr} R)y :\iff xRz_1Rz_2R \dots Rz_nRy$$

for some $z_1, \dots, z_n \in Z$, where $n \geqslant 0$. Then $\langle \mathbf{Rel}, +, \operatorname{Tr}\rangle$ is a traced monoidal category.

3. Consider again the category **Rel**, but this time consider the tensor $X \times Y$ that is given on objects by the cartesian product of sets, and on morphisms by $\langle x, y\rangle(R \times Q)\langle x', y'\rangle$ iff xRy and $x'Qy'$. Note that this is not a categorical product in **Rel**. For $R : X \times Z \to Y \times Z$, define $\operatorname{Tr}' R : X \to Y$ via

$$x(\operatorname{Tr}' R)y :\iff \exists z \in Z. \langle x, z\rangle R \langle y, z\rangle.$$

With this data, $\langle \mathbf{Rel}, \times, \operatorname{Tr}'\rangle$ is traced monoidal.

4. Consider the category **Cpo** of pointed complete partial orders with Scott-continuous maps. For the symmetric monoidal structure, take the usual categorical product on **Cpo**. We can define a trace on this category via a fixpoint construction: if $f : A \times X \to B \times X$, then define $\operatorname{Tr} f : A \to B$ via

$$(\operatorname{Tr} f)(a) := \pi_1(f\langle a, \mu x.\pi_2(f\langle a, x\rangle)\rangle),$$

where $\mu x.\phi(x)$ denotes the least fixpoint of the function $\lambda x.\phi(x)$. More concretely, $(\operatorname{Tr} f)(a)$ is the limit of the sequence $b_1, b_2, b_3, \dots$, where

$$f\langle a, \bot\rangle = \langle b_1, z_1\rangle, \quad f\langle a, z_1\rangle = \langle b_2, z_2\rangle, \quad f\langle a, z_2\rangle = \langle b_3, z_3\rangle, \quad \text{etc.}$$

Other examples of traced monoidal categories arise in many areas of mathematics and computer science, for instance from finite dimensional vector spaces, dataflow networks, predicate transformers, the theory of knots and tangles, game semantics, deterministic Kahn processes, and in many other areas.

3 A classification of traced monoidal categories

The language of traced monoidal categories provides a set of combinators for arranging nodes with fixed input and output arities into cyclic directed graphs. Nothing in the language of traced monoidal categories suggests how these graphs are to be interpreted. As a result, some traced monoidal categories seem to capture operational intuitions about dataflow in networks of communicating processes, while others do not. For this reason, it seems useful to further classify traced monoidal categories, that is, to identify additional properties that allow us to distinguish some of their major features.

Ultimately, it would be desirable to identify enough abstract properties to be able to prove categorical representation theorems for some classes of traced monoidal categories. Such theorems would state, for instance, that any small traced monoidal category with certain properties can be embedded in a given universal one. At present, no such general representation theorems are known for traced monoidal categories. One would expect that the situation is somewhat analogous to that of axiomatic domain theory.

In this section, we will examine two criteria for classifying traced monoidal categories: whether or not the category admits non-determinism, and whether or not the trace is "loop-like".

3.1 First classification: deterministic vs. non-deterministic models

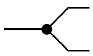
Definition 3.1 A ***monoidal category with diagonals*** is a symmetric monoidal category together with two families of morphisms, indexed by objects, but not necessarily natural:

$$\Delta_A : A \to A \otimes A,$$
$$\Diamond_A : A \to I,$$

such that $\langle A, \Delta_A, \Diamond_A\rangle$ is a symmetric comonoid for each A, and such that these morphisms are compatible with the symmetric monoidal structure in the following sense:

$$\Diamond_I = \mathrm{id}_I : I \to I, \qquad \begin{tikzcd} & I \otimes I \arrow[d, "\cong"] \\ A \otimes B \arrow[ur, "\Diamond_A \otimes \Diamond_B"] \arrow[r, "\Diamond_{A\otimes B}"'] & I, \end{tikzcd} \qquad \begin{tikzcd} & A \otimes A \otimes B \otimes B \arrow[d, "A\otimes c \otimes B"] \\ A \otimes B \arrow[ur, "\Delta_A \otimes \Delta_B"] \arrow[r, "\Delta_{A\otimes B}"'] & A \otimes B \otimes A \otimes B. \end{tikzcd}$$

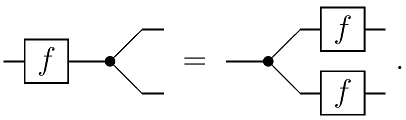
The morphisms Δ_A and $\Diamond_A$ are called, respectively, the ***diagonal*** and the ***weak terminal morphism*** (which serves as the nullary diagonal) at A. In the graphical notation, we write these two morphisms respectively as

 and .

Definition 3.2 In a monoidal category with diagonals, we call a morphism $f : A \to B$ ***copyable*** or ***deterministic*** if

$$\begin{tikzcd} A \arrow[r, "\Delta"] \arrow[d, "f"'] & A \otimes A \arrow[d, "f\otimes f"] \\ B \arrow[r, "\Delta"] & B \otimes B, \end{tikzcd}$$

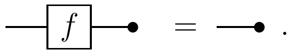
or graphically,



We call a morphism $f : A \to B$ ***discardable*** or ***total*** if

$$\begin{tikzcd} A \arrow[dr, "\Diamond"] \arrow[dd, "f"'] & \\ & I, \\ B \arrow[ur, "\Diamond"'] & \end{tikzcd}$$

or graphically,



The generic terms *copyable* and *discardable* are taken from Thielecke [15]. The terms *deterministic* and *total* are more specific to our particular setting: the intuition behind these terms in the context of communicating processes is as follows. We think of the diagonal morphism as duplicating messages on a channel. Thus, a copyable morphism corresponds to a process with the property that if two independent copies of this process are presented with the same input history, then they produce the same output history. From an extensional point of view, this is the case just in case the process is deterministic. Here, we are assuming that processes interact only via their channels and have no other side-effects.

Further, we think of the weak terminal morphism as a process which accepts, but ignores, all input. Thus, a discardable morphism corresponds to a process which is defined for all possible inputs, *i.e.*, which can never refuse an input. For this reason, we call such a morphism total.

The copyable morphisms of a category $\mathbf{C}$ form a symmetric monoidal subcategory. The same is true for the discardable morphisms. We define the ***focus*** of $\mathbf{C}$ to be the intersection of these two subcategories, and we denote it by $\mathbf{C}^\sharp$. Notice that a morphism $f : A \to B$ is in the focus iff it is a *comonoid morphism*. The following observation is category-theoretical folklore:

Lemma 3.3 *The focus is the largest subcategory on which the tensor product, together with Δ and $\diamond$, restricts to a cartesian product.* □

In particular, it follows that $\mathbf{C} = \mathbf{C}^\sharp$ iff the monoidal structure on $\mathbf{C}$, with its diagonals, is given by a cartesian product. In this case, we also say $\mathbf{C}$ is ***cartesian***.

Examples Each of the four traced monoidal categories from Section 2.1 has an obvious diagonal structure. However, only two of these categories have non-deterministic morphisms:

$\langle\mathbf{Rel}, +\rangle$: Since the tensor is a categorical product in this category, all morphisms are deterministic and total. The focus is the whole category.

$\langle\mathbf{Rel}, \times\rangle$: The total morphisms are the total relations. The deterministic morphisms are the partial functions. Thus the focus is the category $\mathscr{S}$ of sets and functions.

Cpo: Since the tensor is a categorical product in this category, all morphisms are deterministic and total. The focus is the whole category.

Buf: The deterministic and total morphisms in **Buf** are characterized in the following theorem.

Recall that an agent is called τ-free if it contains no silent transitions.

Theorem 3.4 *(a) In* **Buf***, all morphisms are total.*

(b) In **Buf***, a morphisms is deterministic if and only if it is weakly bisimilar to a τ-free agent.* □

Part (a) is an easy consequence of the input-receptivity property from Table 2. To motivate part (b), notice that asynchronous agents, in light of the properties from Tables 1 and 2, satisfy so many confluence properties that τ-actions remain as the only possible source of non-deterministic choices.

Because of their special shape, the τ-free asynchronous agents possess a simple extensional description. As the next theorem shows, they can be represented as continuous maps between certain domains. Thus, one finds that in the deterministic case, our asynchronous agents have no genuine branching-time behavior.

Let ω be the vertical domain of natural numbers. For any set X, ω^X denotes the X-fold power of ω, with the pointwise order. An element of ω^X can be thought of as a multiset in X of countable multiplicity, *i.e.*, where each element occurs at most countably many times.

Theorem 3.5 *Bisimulation equivalence classes of τ-free asynchronous agents $\mathbf{S} : X \to Y$ are in one-to-one correspondence with Scott-continuous functions $f : \omega^X \to \omega^Y$. In particular, $\mathbf{Buf}^\sharp$ is equivalent to a full subcategory of* **Cpo**.$\Box$

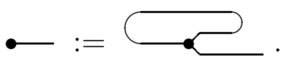
Remark 3.6 The idea of using Scott-continuous maps to model cyclic networks of deterministic, asynchronous processes first appears in an influential paper by Kahn [11]. The previous theorem shows that the focus of the category **Buf** is equivalent to a special case of deterministic Kahn networks, namely the case where each port carries only data of unit type, or equivalently, where all messages commute.

The framework from Section 1 can be naturally extended to include first-in-first-out channels. In this case, one obtains a larger category of asynchronous agents, whose focus is precisely the category of deterministic Kahn processes. An adaption of Theorem 1.9 to the case of first-in-first-out channels was given in [14].

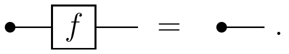
3.2 Second classification: "loop-like" vs. "existential" trace

Consider once again the examples of traced monoidal categories from Section 2.1. The trace operators in **Buf**, $\langle \mathbf{Rel}, + \rangle$, and **Cpo** carry an intuition of "looping" or "iteration": the definition of trace in each of these categories reflects the idea of tokens passing around a loop. On the other hand, the trace operator in $\langle \mathbf{Rel}, \times \rangle$, which is defined by an existential quantifier, carries no such operational intuition. We argue that the intuitive idea of iteration is captured more abstractly by Hasegawa's ***uniformity property*** of traces.

Definition 3.7 For any object A in a traced monoidal category with diagonals, we define a ***weak initial morphism*** $\Box_A : I \to A$ by $\Box_A = \mathrm{Tr}_A(\Delta_A)$. In the graphical notation, we write



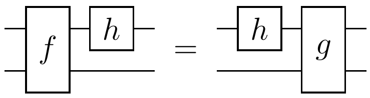
A morphism $f : A \to B$ is called ***strict*** if $\square_A; f = \square_B$, or graphically,



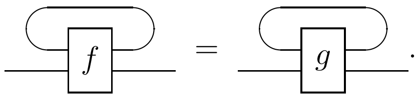
Note: In **Cpo**, our notion of strictness coincides with the usual one: the strict maps are precisely the $\bot$-preserving maps.

For a given traced monoidal category **C**, fix a monoidal subcategory of ***uniform*** morphisms. This subcategory should include at least all isomorphisms, the diagonals, and the weak initial and terminal morphisms. Uniformity of the trace operator is defined with respect to this notion of uniform morphism:

Definition 3.8 (Hasegawa [7]) Given a traced monoidal category with diagonals, we say that the trace is ***uniform*** if for all morphisms $f : A \otimes X \to B \otimes X$, $g : A \otimes Y \to B \otimes Y$, and for all uniform morphisms $h : X \to Y$,



implies



Hasegawa, and independently Hyland, have shown that on a *cartesian* category, giving a traced structure is equivalent to giving a fixpoint structure. Moreover, under this correspondence, uniform traces correspond to uniform fixpoint operators. This is remarkable, because in the usual formulation of uniformity for fixpoint operators, the uniform morphism h appears both in the hypothesis and in the conclusion, whereas in the traced formulation, it appears only in the hypothesis.

Uniformity of traces thus gives rise to a proof principle which is reminiscent of induction: To prove $\mathrm{Tr}_X f = \mathrm{Tr}_Y g$, one chooses a suitable uniform morphism h, which plays the role of an induction hypothesis. The induction step consists of proving $f; (\mathrm{id} \otimes h) = (\mathrm{id} \otimes h); g$. Note that the induction hypothesis does not appear in the conclusion $\mathrm{Tr}_X f = \mathrm{Tr}_Y g$. In some situations, one can also think of h as a kind of loop invariant.

It is because of this inductive flavor of the uniformity principle that we associate it with the intuitive notion of "loop-like" traces. The examples confirm this point of view:

Examples

$\langle$**Rel**$, +\rangle$: The trace is uniform. One can take the class of uniform morphisms to be the entire category.

$\langle$**Rel**$, \times\rangle$: The trace is not uniform. As a matter of fact, the uniformity principle holds precisely if h is an isomorphism.

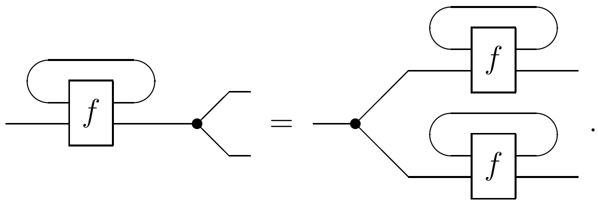
Cpo: The trace is uniform. We can take the uniform morphisms to be the strict maps. Uniformity follows by Hasegawa's result from the fact that **Cpo** has uniform fixpoint operators.

Buf: The trace is uniform. We can take the uniform morphisms to be the strict, deterministic agents which preserve finiteness. The latter requirement means that h produces only finite output on finite input. It is an open problem whether this last hypothesis can be dropped.

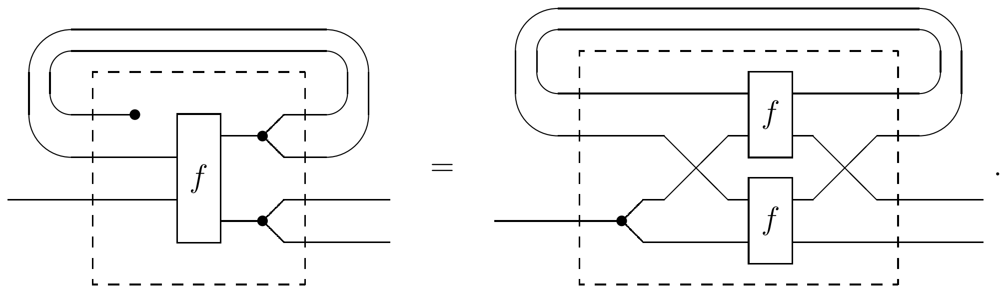
We finish this section on uniform traces by giving an example of how the uniformity property can be applied to mimic an induction proof in a traced monoidal category.

Theorem 3.9 *In a uniformly traced monoidal category, the deterministic morphisms are closed under trace.*

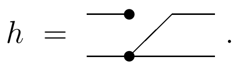
Proof. Suppose that $f : A \otimes X \to B \otimes X$ is deterministic. We want to show that $\mathrm{Tr}_X\, f$ is deterministic as well. Thus, we want to show that



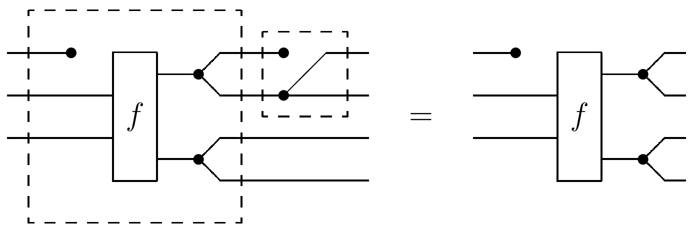
By using the properties of diagonals and graph isomorphism, the claim is equivalent to

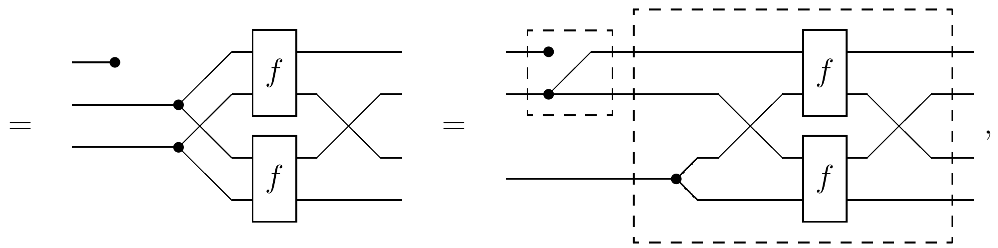


Intuitively, if the two copies of f in the right-hand graph receive identical input, then the two feedback wires will carry identical output, which in turns causes the two copies of f to see identical input, and so on. This kind of situation calls for an inductive argument, which we formalize by using uniformity. As the loop invariant, we choose



Notice that h is uniform. We have





where the middle step uses the assumption that f is deterministic. Now the claim follows by uniformity. □

Corollary 3.10 *Let* $\mathbf{C}$ *be a traced monoidal category with diagonals. If all morphisms in* $\mathbf{C}$ *are total, then* $\mathbf{C}^\sharp$ *is traced monoidal.* □

4 Categorical models of asynchrony

4.1 What is a categorical model of asynchrony?

We have found the category **Buf** of asynchronous agents to have the following structure:

- it is traced monoidal with diagonals,
- all morphisms are total,
- trace is uniform, and
- the focus is equivalent to a category of cpo's.

Notice that some of these properties relate specifically to asynchronous, as opposed to synchronous, communication. The diagonal structure causes an explicit asymmetry between input and output, as is typical for asynchronous systems. By contrast, properties of synchronous communication are generally self-dual. The fact that all morphisms are total corresponds to the receptivity property of asynchronous processes. The last property is a version of Kahn's principle, which roughly states that deterministic asynchronous processes can be modeled by Scott-continuous maps.

The above properties seem to be a reasonably general setup for categories of asynchronous processes. Thus, we define a ***categorical model of asynchrony*** to be a category with the above structure. The question immediately arises whether there are other interesting examples of such categories, and whether there are general constructions available on them. We will see next that the answer to both questions is positive.

4.2 From deterministic to non-deterministic models

We have seen in Section 3.1 how to pass from a non-deterministic to a deterministic model by taking the focus. Is there a construction in the opposite direction, passing from a deterministic model to a non-deterministic one? In fact, there are many such constructions, and the following is one of them.

Definition 4.1 Let $\mathbf{C} = \langle \mathbf{C}, \otimes, \mathrm{Tr} \rangle$ be a traced monoidal category with diagonals. We define a category $\mathbf{C}^{\mathrm{nd}}$ as follows: The objects of $\mathbf{C}^{\mathrm{nd}}$ are those of $\mathbf{C}$. A morphism $F : X \to Y$ in $\mathbf{C}^{\mathrm{nd}}$ is a non-empty set of morphisms $f : X \to Y$ in $\mathbf{C}$. The operations on $\mathbf{C}^{\mathrm{nd}}$, which we write in boldface, are defined pointwise:

$$\begin{aligned} F \mathbin{\boldsymbol{;}} G &:= \{f; g \mid f \in F, g \in G\}, \\ F \mathbin{\boldsymbol{\otimes}} G &:= \{f \otimes g \mid f \in F, g \in G\}, \\ \mathbf{Tr}\, F &:= \{\mathrm{Tr}\, f \mid f \in F\}, \end{aligned}$$

and similarly for constants, such as $\mathbf{id}_X = \{\mathrm{id}_X\}$, $\boldsymbol{\Delta}_X = \{\Delta_X\}$, etc.

Surprisingly, this simple-minded construction actually preserves the structure of $\mathbf{C}$.

Lemma 4.2 *If $\mathbf{C}$ is traced monoidal with diagonals, then so is $\mathbf{C}^{\mathrm{nd}}$.*

Proof. The lemma is a consequence of a general meta-theorem, which we present in Section 5. In the present context, the meta-theorem states that the $(-)^{\mathrm{nd}}$-construction preserves all structure that is given by affine equations. An equation is said to be ***linear*** if each variable occurs precisely once on each side, and ***affine*** if each variable occurs at most once on each side of the equation. Notice that all the defining equations of traced monoidal categories with diagonals, from Definitions 2.1 and 3.1, are linear. As a matter of fact, Definition 3.1 contains only closed (*i.e.*, variable-free) equations, which are trivially linear. Thus, the $(-)^{\mathrm{nd}}$-construction preserves this structure. □

Lemma 4.3 *The focus of $\mathbf{C}^{\mathrm{nd}}$ consists precisely of the singletons $\{f\}$, where f is focal in $\mathbf{C}$. Thus, $(\mathbf{C}^{\mathrm{nd}})^\sharp \cong \mathbf{C}^\sharp$.* □

Let us now concentrate on the case where $\mathbf{C}$ is a cartesian category, that is, all morphisms in $\mathbf{C}$ are deterministic and total.

Lemma 4.4 *Let $\mathbf{C}$ be cartesian. Then*

(i) $(\mathbf{C}^{\mathrm{nd}})^\sharp \cong \mathbf{C}$.

(ii) *All morphisms in $\mathbf{C}^{\mathrm{nd}}$ are total.*

(iii) *If $\mathbf{C}$ is uniformly traced, then so is $\mathbf{C}^{\mathrm{nd}}$. We can take the uniform morphisms in $\mathbf{C}^{\mathrm{nd}}$ to be the singletons of uniform morphisms in $\mathbf{C}$.* □

4.3 Application: A simple cpo-based model

Recall that the category $\mathbf{Cpo}$ is a traced cartesian category with diagonals. We apply the $(-)^{\mathrm{nd}}$-construction from the last section. By Lemmas 4.2 and 4.4, the category $\mathbf{Cpo}^{\mathrm{nd}}$ is traced monoidal with diagonals, all morphisms are total, trace is uniform, and the focus is isomorphic to $\mathbf{Cpo}$. Thus, $\mathbf{Cpo}^{\mathrm{nd}}$ satisfies all our requirements from Section 4.1 to be a model of non-deterministic, asynchronous communication. We remark on two particular features of this model:

No branching-time behavior. A morphism in $\mathbf{Cpo}^{\mathrm{nd}}$ is just a set of continuous maps. Thus, a non-deterministic process in this model has no branching-time behavior: At each run, a process makes a single non-deterministic choice and then behaves like a deterministic process. In the asynchronous world, the absence of true branching-time behavior is not an unreasonable assumption: since the environment in an asynchronous setting can neither influence the outcome of non-deterministic choices, nor detect the time at which such a choice is made, one may as well assume that all such choices are made ahead of time.

Intentionality. The model $\mathbf{Cpo}^{\mathrm{nd}}$ is not very extensional, in the sense that it distinguishes many processes which are intuitively indistinguishable. For example, let $B = \{T, F, \bot\}$ be the domain of flat booleans, and let *id*, *not*, *true*, and *false* be the four obvious strict total maps $B \to B$. Then the two morphisms

$$\{id, not\}, \{true, false\} : B \to B$$

are different in $\mathbf{Cpo}^{\mathrm{nd}}$, but intuitively they represent the same behavior: each of the processes produces one of the answers T or F non-deterministically, regardless of the input. The only way these processes differ is in how their non-deterministic choices are correlated across different, mutually exclusive branches of control.

Depending on what one is trying to achieve with a model, intentionality need not be a problem. Notice, for instance, that the model **Buf** is also highly intentional. However, in some cases, one might want to obtain a model which is less intentional. This can be done, for instance, by putting an observational equivalence on processes, as we will illustrate in the next section.

4.4 A cpo-based model with observational congruence

The preceding discussion motivates us to refine the model $\mathbf{C}^{\mathrm{nd}}$ by equipping it with a suitable equivalence relation on morphisms. On the one hand, this equivalence should capture our intuition about what it means for two processes to be observationally equivalent. On the other hand, it should be compatible with the operations of a traced monoidal category, in other words, it should be a *congruence*. This ensures that equivalent processes can be substituted for each other in any context without changing the global behavior, a property of the semantics which is also known as *compositionality*.

But what is a good notion of observation? This question does not have a single answer, since there are many different intuitions about observations, and many possible formalizations of them. One choice to be made is whether one wants to observe only positive information, such as output, or also some negative information, such as the absence of output, termination, etc.

Here, we give an example of an observational congruence for a model based on continuous domains, in which we observe only positive information. Let $\mathbf{D}$ be a category of continuous domains [2], closed under finite products. The reader who prefers it may assume that $\mathbf{D}$ is a category of algebraic domains.

Definition 4.5 Let A, B be objects of $\mathbf{D}$. An ***observation*** of type $A \to B$ is a pair $\langle u, v\rangle$ of continuous functions $u : \omega \to A$ and $v : B \to \omega$. We say that a morphism $f : A \to B$ ***satisfies*** the observation $\langle u, v\rangle$ if $\mathrm{id}_\omega \leqslant u; f; v$. We write $\mathrm{obs}\, f$ for the set of all observations satisfied by f. If $F : A \to B$ is a morphism in $\mathbf{D}^{\mathrm{nd}}$, then we write $\mathrm{obs}\, F = \bigcup\{\mathrm{obs}\, f \mid f \in F\}$. Finally, we say that $F, G : A \to B$ are ***observationally equivalent***, in symbols $F =_{\mathrm{obs}} G$, if $\mathrm{obs}\, F = \mathrm{obs}\, G$.

We can picture an observation $\langle u, v\rangle$ as a "diagram with a hole":

$$\begin{tikzcd} & A \arrow[r, dotted] & B \arrow[dr, "v"] & \\ \omega \arrow[ur, "u"] \arrow[rrr, "\mathrm{id}"'] & & \arrow[phantom, l, "\geqslant"] & \omega. \end{tikzcd}$$

A morphism $f : A \to B$ satisfies the observation if it fits the hole. Thus, an observation of this kind allows us to test a morphism f on an infinite *increasing* sequence of inputs, and observe whether the output at each stage lies in a given open set. Two sets of morphisms are equivalent if their elements collectively have the same observations. Since the sequence of inputs in each individual run is increasing, no backtracking is possible.

Theorem 4.6 *Observational equivalence is respected by composition, tensor, and trace. Thus, it is a congruence on the traced monoidal category* $\mathbf{D}^{\mathrm{nd}}$. $\square$

In the proof that trace preserves observational equivalence, the domain ω plays an important role. If we had replaced ω in the definition of observations by, say, 1, then the theorem would not be true. Because $\mathrm{Tr}\, f$ is defined as a limit, an observation at a single point at $\mathrm{Tr}\, f$ translates into an increasing sequence of observations at f.

Remark 4.7 The semantics given in this section is a domain-theoretic generalization of *trace semantics*. If one restricts attention to domains of the form ω^X, then two processes of type $\omega^X \to \omega^Y$ are observationally equivalent iff they are *trace equivalent*, *i.e.*, if they generate the same set of combined input-output sequences.

5 The T-construction

In Section 4.2, we have seen that the class of traced monoidal categories with diagonals is closed under a certain set-theoretic operation on hom-sets, namely taking non-empty power sets, and extending the operations pointwise. In this section, we will outline the general principles behind this construction.

Definition 5.1 A ***linear functor*** is a functor $T : \mathscr{S} \to \mathscr{S}$ from the category of sets to itself, together with a morphism $\epsilon : 1 \to T1$ and a natural transformation $\rho_{A,B} : TA \times TB \to T(A \times B)$, satisfying

Left unit:

$$\begin{tikzcd} 1 \times TA \arrow[r, "\cong"] \arrow[d, "\epsilon \times TA"'] & TA \arrow[d, "\cong"] \\ T1 \times TA \arrow[r, "\rho_{1,A}"] & T(1 \times A) \end{tikzcd}$$

Right unit:

$$\begin{tikzcd} TA \times 1 \arrow[r, "\cong"] \arrow[d, "TA \times \epsilon"'] & TA \arrow[d, "\cong"] \\ TA \times T1 \arrow[r, "\rho_{A,1}"] & T(A \times 1) \end{tikzcd}$$

Associativity:

$$\begin{tikzcd} TA \times TB \times TC \arrow[r, "TA \times \rho_{B,C}"] \arrow[d, "\rho_{A,B} \times TC"'] & TA \times T(B \times C) \arrow[d, "\rho_{A,B \times C}"] \\ T(A \times B) \times TC \arrow[r, "\rho_{A \times B,C}"] & T(A \times B \times C) \end{tikzcd}$$

Symmetry:

$$\begin{tikzcd} TA \times TB \arrow[r, "\cong"] \arrow[d, "\rho_{A,B}"'] & TB \times TA \arrow[d, "\rho_{B,A}"] \\ T(A \times B) \arrow[r, "\cong"] & T(B \times A) \end{tikzcd}$$

Of course, one of the unit laws is redundant in the presence of symmetry. We say that a linear functor T is ***affine*** if $T1 \cong 1$, and ***relevant*** if

$$\begin{tikzcd} TA \arrow[r, "\Delta_{TA}"] \arrow[dr, "T\Delta_A"'] & TA \times TA \arrow[d, "\rho_{A,A}"] \\ & T(A \times A). \end{tikzcd}$$

Example 5.2 (i) The covariant power set functor $\mathscr{P}$ is linear.

(ii) The covariant non-empty power set functor $\mathscr{P}^+$, which sends A to the set of non-empty subsets of A, is affine.

(iii) The lifting functor $\mathscr{P}^-$, which sends A to the set of subsets of A of at most one element, is relevant.

(iv) The functor λ, which sends a set X to a vector space with basis X, is linear.

(v) The functor λ^+, which sends X to the subset of λX of convex linear combinations of basis elements, is affine.

(vi) A linear functor can be obtained from any commutative strong monad $\langle T, \mu, \eta \rangle$ on $\mathscr{S}$, where $\epsilon = \eta_1$ and

$$\rho_{A,B} = TA \times TB \to T(TA \times B) \to TT(A \times B) \xrightarrow{\mu} T(A \times B).$$

Here, the commutativity of the strength of the monad is equivalent to the symmetry of the linear functor.

Given a linear functor T, we define an operation on categories, called the ***T-construction***, as follows: If $\mathbf{C}$ is any category, then the objects of $\mathbf{C}^T$ are given by $|\mathbf{C}^T| = |\mathbf{C}|$, and the morphisms by $\mathbf{C}^T(X,Y) = T(\mathbf{C}(X,Y))$. Operations are defined pointwise, using ϵ and ρ. For instance, composition is a binary operation on $\mathbf{C}$:

$$;_{X,Y,Z} : \mathbf{C}(X,Y) \times \mathbf{C}(Y,Z) \to \mathbf{C}(X,Z).$$

This gives rise to a composition operation on $\mathbf{C}^T$ via

$$\boldsymbol{;}_{X,Y,Z} = T(\mathbf{C}(X,Y)) \times T(\mathbf{C}(Y,Z)) \xrightarrow{\rho} T(\mathbf{C}(X,Y) \times \mathbf{C}(Y,Z)) \xrightarrow{T(;)} T(\mathbf{C}(X,Z)).$$

The identity morphisms of $\mathbf{C}$ are given as constants

$$\mathrm{id}_X : 1 \to \mathbf{C}(X,X),$$

from which we define the identity morphisms of $\mathbf{C}^T$ via

$$\mathbf{id}_X = 1 \xrightarrow{\epsilon} T1 \xrightarrow{T(\mathrm{id})} T(\mathbf{C}(X,X)).$$

Any other algebraic operations which are part of the structure of $\mathbf{C}$ may be transferred to $\mathbf{C}^T$ in the same way.

Remark 5.3 The T-construction is a special case of a more general construction in the theory of enriched categories. A linear functor is a lax functor for the symmetric monoidal structure given by products on $\mathscr{S}$. The category $\mathbf{C}^T$ is called the ***direct image of* $\mathbf{C}$ *by* $\boldsymbol{T}$** in Bénabou [4, pp. 53f].

Definition 5.4 An equation is called ***linear*** (respectively ***affine***, respectively ***relevant***) if each variable occurs exactly once (respectively at most once, respectively at least once) on each the left-hand side and the right-hand side of the equation.

Theorem 5.5 (Meta-Theorem) (i) *If T is a linear functor, then any linear equation that holds in $\mathbf{C}$ also holds in $\mathbf{C}^T$.*

(ii) *If T is affine, then any affine equation that holds in $\mathbf{C}$ also holds in $\mathbf{C}^T$.*

(iii) *If T is relevant, then any relevant equation that holds in $\mathbf{C}$ also holds in $\mathbf{C}^T$.*

Some consequences:

- The defining equations of a category, namely the associativity and identity laws, are linear. Thus, if T is linear, then $\mathbf{C}^T$ is a category.
- The defining equations of traced monoidal categories with diagonals are linear. Thus, this structure is preserved by the linear T-construction.
- Monoidal closed structure is given by linear equations, and is thus preserved by the linear T-construction.
- The structure of a terminal object is given by an affine equation. Thus, if T is affine, then any terminal object in $\mathbf{C}$ is also terminal in $\mathbf{C}^T$.

Moreover, if one applies the T-construction to a category with non-linear structure, then at least the linear part of that structure is preserved. For instance, if $\mathbf{C}$ has finite products, then $\mathbf{C}^T$ has a symmetric monoidal structure with diagonals.

Remark 5.6 The $(-)^{\mathrm{nd}}$-construction from Section 4.2 is of course the T-construction with $T = \mathscr{P}^+$, the non-empty power set functor. We can call the kind of non-determinism that is introduced by this construction *set-based non-determinism*. By choosing a different affine functor T, we can model other kinds of non-determinism. For instance, if we let $T = \lambda^+$, the convex linear combinations functor from Example 5.2(v), then we get a model of *probabilistic non-determinism*. Here, a probabilistically non-deterministic process is modeled as a convex linear combination of deterministic processes. Note that the resulting category still satisfies the axioms of Section 4.1, so our notion of categorical models of asynchrony is general enough to allow for different kinds of non-determinism.

A sometimes useful generalization of the T-construction is obtained by dropping the symmetry requirement in the definition of a linear functor. This allows us to consider structures that are defined by equations with a fixed order among the variables. We say that a linear equation is ***rigid*** or ***non-commutative*** if the variables occur *in the same left-to-right order* on both sides of the equation. Of course, this presupposes a choice of concrete syntax for the basic operations. A ***rigid functor*** is defined like a linear functor, except that symmetry is not required. Such functors arise, for example, from non-commutative strong monads such as the continuations monad.

The T-construction for rigid functors is well-defined, and it preserves rigid equations. An example of a structure that is given by rigid equations is the structure of a *premonoidal category*, which was introduced by Power and Robinson [13]. Premonoidal structure is precisely the part of monoidal structure which is given by rigid equations; thus, if T is a rigid functor and $\mathbf{C}$ is a monoidal category, then $\mathbf{C}^T$ is premonoidal.

Acknowledgements

Part of this research was done while I was visiting BRICS, Basic Research in Computer Science, the Centre of the Danish National Research Foundation, in the Spring of 1998. Special thanks to Thomas Hildebrandt for stimulating discussions that have influenced this work. I would also like to thank the organizers of MFPS for inviting me to give this talk in New Orleans.

References

[1] S. Abramsky. Retracing some paths in process algebra. In *Proceedings of the Seventh International Conference on Concurrency Theory*, Springer LNCS 1119, 1996.

[2] S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Clarendon Press, 1994.

[3] G. A. Agha. *Actors: a Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, Massachusetts, 1986.

[4] J. Bénabou. Introduction to bicategories. In *Reports of the Midwest Category Seminar*, Springer Lecture Notes in Mathematics 47, pages 1–77, 1967.

[5] G. Boudol. Asynchrony and the π-calculus. Technical Report 1702, INRIA, Sophia-Antipolis, 1992.

[6] C. Fournet and G. Gonthier. The reflexive cham and the join-calculus. In *POPL '96: Proceedings of the 23rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1996.

[7] M. Hasegawa. *Models of Sharing Graphs: A Categorical Semantics of let and letrec*. PhD thesis, Department of Computer Science, University of Edinburgh, July 1997.

[8] T. T. Hildebrandt, P. Panangaden, and G. Winskel. Relational semantics of non-deterministic dataflow. BRICS Report Series RS-97-36, 1997.

[9] A. Jeffrey. Premonoidal categories and a graphical view of programs. Preprint, Dec. 1997.

[10] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119:447–468, 1996.

[11] G. Kahn. The semantics of a simple language for parallel programming. In *Information Processing*, volume 74, pages 471–475, 1974.

[12] B. C. Pierce and D. N. Turner. *PICT: A Programming Language Based on the π-Calculus*. U. Cambridge, 1996.

[13] J. Power and E. Robinson. Premonoidal categories and notions of computation. *Math. Struct. in Computer Science*, 7(5):445–452, 1997.

[14] P. Selinger. First-order axioms for asynchrony. In *Proceedings of the Eighth International Conference on Concurrency Theory, Warsaw, Poland*, Springer LNCS 1243, pages 376–390, 1997.

[15] H. Thielecke. *Categorical Structure of Continuation Passing Style*. PhD thesis, University of Edinburgh, 1997.