# Constructing Process Categories

J.R.B. Cockett D.A. Spooner

Corresponding Author:

David Spooner

University of Calgary
Department of Computer Science
2500 University Drive N.W.
Calgary, Canada T2N 1N4

Phone: 403 220 6780

email: `spoonerd@cpsc.ucalgary.ca`

**Abstract**

We present a general construction of process categories and show how Abramsky's category **SProc** is obtained in a variety of instances. The process category of a category of models is a quotiented span category: the congruence is induced by a chosen cover system on the model category, and corresponds to process bisimulation. The construction is 2-functorial, allowing properties of process categories to be established in the model categories. A general theory of trace objects within model categories is also developed: it is shown to be sufficient to construct processes upon the full subcategory of trace objects as it yields the same process category.

## 1 Introduction

*Interaction Categories* have been proposed by Abramsky [Abr93] as a new paradigm in the semantics of computation. In contrast with the categories of denotational semantics, interaction categories have *specifications* as objects, *concurrent processes* as morphisms, and composition given by *process interaction*.

| Denotational Semantics | Interaction Categories |
|---|---|
| domains | interface specifications |
| continuous functions | communicating processes |
| functional composition | interaction |

A significant consequence of this paradigm shift has been the discovery of fully abstract models for PCF [AJM93]: game theoretic interaction categories [AJ92] can be constructed in a straightforward manner to capture the key operational aspects of computation. Perhaps more significant is, in providing a process based semantics for programming Abramsky raised the possibility of providing a unifying typed framework for concurrent and functional programming.

Prior to Interaction Categories, the concurrency community had regarded processes primarily as algebras. This made inaccessible the type-based verification techniques so fundamental to functional programming. In the Interaction Category paradigm, a process is a map between interface specifications; as such, the fact that a composite process adheres to its specification is a consequence of its construction from well-typed components.

To realize the benefits of typed concurrent programming, Abramsky introduced the interaction category **SProc** of synchronous processes. It's objects are safety specifications, given as sets of permissible traces; morphisms are transition systems modulo strong bisimulation, and composition is given by restricted parallel composition in the sense of Milner's calculus SCCS [Mil83]. Abramsky has shown how SCCS is faithfully interpreted in **SProc**, allowing the typed reconstruction of much of process algebra.

A common feature amongst examples of Interaction Categories is that the maps are chosen to ensure a canonical representative of each. In **SProc**, for example, Aczel's synchronization trees [Acz88] serve to represent strong bisimulation classes of transition systems. It is often desirable, however, to present processes using a state-based notation. Equality of processes then becomes an issue as such presentations inevitably allow many expressions of the "same" process.

Joyal, Nielsen and Winskel [JNW93] have recently presented a categorical treatment of bisimulation: objects in a category of models are *bisimilar* when related by a span of *open maps*, where these are defined appropriately in each model category. This view yields accepted notions of bisimulation in a variety of settings: transition systems, labelled event structures and transition systems with independence.

For example, the category $\mathbf{T}$ of labelled transition systems presented by Winskel [Win87] has as objects structures

$$(\Sigma,\ S,\ i\in S,\ R\subseteq S\times\Sigma\times S),$$

with $\Sigma$ a set of actions, $S$ a set of states, $i$ an initial state, and $R$ a transition relation. The morphisms $f : A \longrightarrow B$ of this category are pairs of functions

$$(f_S : S_A \longrightarrow S_B,\ f_\Sigma : \Sigma_A \longrightarrow \Sigma_B)$$

which preserve the initial state (i.e. $f(i_A) = i_B$) and preserve transitions (i.e. $(s, a, t) \in R_A$ implies $(f(s), f(a), f(t)) \in R_B$). It is the nature of any map $f : A \longrightarrow B$ of $\mathbf{T}$ that finite sequences of actions (or *paths*) available in $A$ correspond (via $f$) to paths in $B$.

$$\begin{tikzcd}
i_A \arrow[r, bend left, "a_1"] & \cdots \arrow[r, bend left, "a_n"] \arrow[d, squiggly, "f"] & \cdot & & \\
i_B \arrow[r, bend left, "f(a_1)"] & \cdots \arrow[r, bend left, "f(a_n)"] & \cdot \arrow[r, bend left, "b_1"] & \cdots \arrow[r, bend left, "b_m"] & \cdot
\end{tikzcd}$$

Intuitively, one takes the open maps of this category to be those for which the *extensions* of any such path in $B$ are reflected in $A$. Such maps can be identified categorically as having a *path lifting* property. In a fibre $\mathbf{T}_\Sigma$ over some alphabet $\Sigma$, transition systems $A$ and $B$ are strongly bisimilar if and only if they are related by a span of such open maps [JNW93]:

$$\begin{tikzcd}
& R \arrow[dl, "f"'] \arrow[dr, "g"] & \\
A & & B
\end{tikzcd}$$

Our work, we hope, serves to integrate the two paradigms discussed above, making the categorical treatment of process equivalence central to the construction of Interaction Categories. We believe that the general techniques used in our construction clarify the nature of Interaction Categories and will simplify their construction in other settings.

A process category is constructed upon a model category (such as $\mathbf{T}$) as a quotiented span category. Spans are a natural generalization of functions: they are symmetric and correspond to the intuition of processes being "relations extended in time" [Abr93]. The congruence used to quotient spans is a generalized bisimulation induced by a chosen *cover system* (or class of open maps) in the model category. The entire construction is 2-functorial, allowing properties of (and amongst) process categories to be established in the model categories.

**SProc** arises as the process category constructed upon the subcategory of traces within the category of deterministic transition systems. We develop a general characterization of *trace objects* within model categories, which is linked to the existence of partial inductive datatypes. It is shown to be sufficient to consider the full subcategory of trace objects when constructing processes, as it yields an equivalent process category.

Thus the original presentation of **SProc** in terms of traces is unnecessary. Indeed, **SProc** is obtained from a variety of related model categories — this is due to relationships amongst the model categories. An important observation is that the choice of model category affects the effort required to demonstrate properties of the resulting process category.

The rest of the paper is structured as follows:

· section 2 presents the model categories upon which we will construct processes and identifies some relationships between them;

· section 3 describes the theory of cover systems which are used subsequently to express process bisimulation and to demonstrate equivalence of process categories;

· section 4 presents the construction of process categories and discusses some consequences of its 2-functorial nature;

· section 5 introduces the theory of trace objects and shows how their construction in the category of deterministic transition systems is related to the existence of partial list construction.

## 2 Models for Processes

This section describes three categories of *models* upon which we will (later) construct *process* categories. Each represents a variety of transition system: two of these, deterministic and nondeterministic transition systems, will be familiar; the other, which we call *source machines*, are deterministic transition systems with a separated initial state. The category of source machines turns out to have more structure than its counterparts, and this structure will prove useful for establishing properties of the associated process category. Although the model categories are distinct, due to certain relationships amongst them, each will give rise to the same process categories.

Each model category is presented as the category of models of a finite limit sketch (see Barr and Wells [BW90]) — this will prove valuable in obtaining cover systems for each setting. As the base category for each construction, we will use the elementary setting of a lextensive category (see Carboni, Lack and Walters [CLW92] or Cockett [Coc93]). A lextensive category has finite limits and finite coproducts with the property that in the following diagram

$$\begin{tikzcd}
X \arrow[r, "x"] \arrow[d] \arrow[dr, phantom, "(1)"] & Z \arrow[d] & Y \arrow[l, "y"'] \arrow[d] \arrow[dl, phantom, "(2)"] \\
A \arrow[r, "b_0"'] & A+B & B \arrow[l, "b_1"]
\end{tikzcd}$$

(1) and (2) are pullbacks if and only if the top row is a coproduct. [1]

### 2.1 Source Machines

Let $\mathbf{C}$ be a lextensive category. We define the category of source machines and their homomorphisms in $\mathbf{C}$ as follows:

**Definition 2.1** $\mathrm{Src}(\mathbf{C})$ *is the category of models in* $\mathbf{C}$ *of the following sketch:*

$$\begin{tikzcd}
 & P \arrow[dl, "m"'] \arrow[dr, "\alpha"] & \\
(S+1)\times\Sigma & & S
\end{tikzcd}$$

Here $S$ is a state space, $\Sigma$ is an alphabet of actions, $m$ indicates the actions permitted at each state, and $\alpha$ determines the state change upon action. Note that the initial state (denoted by 1) is not amongst the state space and thus not (otherwise) reachable.

[1]We take $\times$ and $+$ as left associative operators, with $\times$ binding tighter than $+$ and $+$ binding tighter than ;.

As **C** is lextensive, one can separate the behaviour at the initial and non-initial states to obtain an equivalent view of a source machine:

$$\begin{tikzcd}
 & S & \\
P_0 \arrow[ur, dashed, "\alpha_0"] \arrow[r] \arrow[d, "m_0"'] \arrow[dr, phantom, "\lrcorner", very near start] & P \arrow[u, "\alpha"'] \arrow[d, "m"] & P_1 \arrow[ul, dashed, "\alpha_1"'] \arrow[l] \arrow[d, "m_1"] \arrow[dl, phantom, "\llcorner", very near start] \\
1\times\Sigma \arrow[r, "b_1\times 1"'] & (S+1)\times\Sigma & S\times\Sigma \arrow[l, "b_0\times 1"]
\end{tikzcd}$$

Note that $P$ is a coproduct of $P_0$ and $P_1$, and that $(m_0, \alpha_0)$ and $(m_1, \alpha_1)$ give the behavior at the initial and non-initial states respectively.

**Lemma 2.2** Src(**C**) *is a lextensive category.*

**Proof**. Finite limits in Src(**C**) are given componentwise — the final object is $1 \times (1+1) \xleftarrow{=} 1 \times (1+1) \longrightarrow 1$ and, using the fact that pullbacks in **C** are preserved by products and coproducts, the pullback of maps $f$ and $g$ is as follows:

$$\begin{tikzcd}
 & P_Q \arrow[rrrr, "q_P'"] \arrow[dr, "\alpha_Q"] \arrow[ddl, dashed, "m_Q"'] \arrow[ddd, "q_P"] & & & & P_B \arrow[dr, "\alpha_B"] \arrow[ddl, "m_B"'] \arrow[ddd, "g_P"] & \\
 & & S_Q \arrow[rrrr, "q_S'"] \arrow[ddd, "q_S"] & & & & S_B \arrow[ddd, "g_S"] \\
(S_Q+1)\times\Sigma_Q \arrow[rrrr, "(q_S'+1)\times q_\Sigma'"'] \arrow[ddd, "(q_S+1)\times q_\Sigma"] & & & & (S_B+1)\times\Sigma_B \arrow[ddd, "(g_S+1)\times g_\Sigma"'] & & \\
 & P_A \arrow[rrrr, "f_P"] \arrow[dr, "\alpha_A"] \arrow[ddl, "m_A"'] & & & & P_C \arrow[dr, "\alpha_C"] \arrow[ddl, "m_C"'] & \\
 & & S_A \arrow[rrrr, "f_S"] & & & & S_C \\
(S_A+1)\times\Sigma_A \arrow[rrrr, "(f_S+1)\times f_\Sigma"'] & & & & (S_C+1)\times\Sigma_C & &
\end{tikzcd}$$

Finite coproducts are also obtained componentwise: the initial object is given by $0 \times (1+0) \longleftarrow 0 \xrightarrow{=} 0$ and the coproduct of $A$ and $B$ is shown below.

$$\begin{tikzcd}
 & P_A \arrow[rrr, "b_0"] \arrow[dr, "\alpha_A"] \arrow[dl, "m_A"'] & & & P_A+P_B \arrow[dr, "\alpha_A+\alpha_B"] \arrow[dl, "m_A+m_B"'] & & & P_B \arrow[lll, "b_1"'] \arrow[dr, "\alpha_B"] \arrow[dl, "m_B"'] & \\
 & & S_A \arrow[rrr, "b_0"] & & & S_A+S_B & & & S_B \arrow[lll, "b_1"'] \\
(S_A+1)\times\Sigma_A \arrow[rrr, "b_0"] \arrow[drrr, "(b_0+1)\times b_0"'] & & & (S_A+1)\times\Sigma_A+(S_B+1)\times\Sigma_B \arrow[d, dashed] & & & (S_B+1)\times\Sigma_B \arrow[lll, "b_1"'] \arrow[dlll, "(b_1+1)\times b_1"] & & \\
 & & & (S_A+S_B+1)\times(\Sigma_A+\Sigma_B) & & & & &
\end{tikzcd}$$

That the coproduct is extensive follows easily from the componentwise construction of coproducts and pullbacks: Recalling the diagramatic property of lextensive categories, if (1) and (2) are pullbacks then the components (e.g. $(f_S, x_S)$ and $(y_S, g_S)$) are pullbacks, making the components of $(Z, x, y)$ coproducts in $\mathbf{C}$ and thus $(Z, x, y)$ itself a coproduct in $\mathrm{Src}(\mathbf{C})$. Conversely, if the top row is a coproduct then its components (e.g. $(S_Z, x_S, y_S)$) are coproducts, making the components of (1) and (2) pullbacks in $\mathbf{C}$ and thus (1) and (2) pullbacks in $\mathrm{Src}(\mathbf{C})$. $\square$

## 2.2 Deterministic Transition Systems

We now consider the category of deterministic transition systems and their homomorphisms. Let $\mathbf{C}$ remain a lextensive category,

**Definition 2.3** $\mathrm{DTran}(\mathbf{C})$ *is the category of models of the following sketch:*

$$\begin{tikzcd}
& P \arrow[dl, "m"'] \arrow[dr, "\alpha"] & & \\
S\times\Sigma & & S & 1 \arrow[l, "i"]
\end{tikzcd}$$

The objects of $\mathrm{DTran}(\mathbf{C})$ differ from those of $\mathrm{Src}(\mathbf{C})$ only in that the initial state is included in the state space. The present category has finite limits, again given componentwise, but does not have coproducts — a coproduct would have the initial states of its summands coalesced, requiring $\mathbf{C}$ has pushouts along "elements". If $\mathbf{C}$ had pushouts, the induced coproduct on $\mathrm{DTran}(\mathbf{C})$ would not be lextensive as such coproducts are not disjoint.

**Lemma 2.4** $\mathrm{DTran}(\mathbf{C})$ *has finite limits.*

There is an adjunction relating the categories $\mathrm{Src}(\mathbf{C})$ and $\mathrm{DTran}(\mathbf{C})$. To obtain a deterministic transition system from a source machine, one simply forgets the distinguished status of the initial state.

$$\begin{tikzcd}
& P \arrow[dl, "m"'] \arrow[dr, "\alpha"] & & & & P \arrow[dl, "m"'] \arrow[dr, "\alpha"] & & \\
(S+1)\times\Sigma & & S \arrow[r, mapsto, "F"] & \, & (S+1)\times\Sigma & & S \arrow[dr, "b_0"] & \\
& & & & & & & S+1 & 1 \arrow[l, "b_1"]
\end{tikzcd}$$

Conversely, given a deterministic transition system $(m, \alpha, i)$, one can isolate the behavior $(m', \alpha')$ at the initial state as follows:

$$\begin{tikzcd}
1\times\Sigma \arrow[d, "i\times 1"'] & P' \arrow[l, "m'"'] \arrow[d, "p"] \arrow[dr, dashed, "\alpha'"] \arrow[dl, phantom, "\lrcorner", very near start] & \\
S\times\Sigma & P \arrow[l, "m"] \arrow[r, "\alpha"'] & S
\end{tikzcd}$$

A source machine is then obtained by introducing a separate initial state with this behaviour[2]:

$$\begin{tikzcd}
 & P \arrow[dl, "m"'] \arrow[dr, "\alpha"] & & & & & & P+P' \arrow[dl, "m+m'"'] \arrow[ddr, "\langle\alpha|\alpha'\rangle"] & \\
S\times\Sigma & & S & 1 \arrow[l, "i"] & {} \arrow[r, mapsto, "G"] & {} & S\times\Sigma+1\times\Sigma \arrow[dl, "\sim"'] & & \\
 & & & & & (S+1)\times\Sigma & & & S
\end{tikzcd}$$

The unit $\eta$ of the adjunction simply injects a source machine $A$ into the larger $GFA$. Note that this requires taking the initial state of $A$ to the new initial state of $GFA$, so it is convenient to use the alternate view of source machines discussed previously.

$$\begin{tikzcd}
 & P \arrow[rrr, "b_0+1"] \arrow[ddl, "m"'] \arrow[dr, "\alpha"] & & & P+P_0 \arrow[ddl, "m+m_0;\sim"'] \arrow[dr, "\langle\alpha|\alpha_0\rangle;b_0"] & \\
 & & S \arrow[rrr, "b_0"] & & & S+1 \\
(S+1)\times\Sigma \arrow[rrr, "(b_0+1)\times 1"'] & & & (S+1+1)\times\Sigma & &
\end{tikzcd}$$

The counit $\epsilon$ coalesces the initial state introduced by $G$ in $FGB$ with the original initial state of $B$:

$$\begin{tikzcd}
 & P+P' \arrow[rrr, "\langle 1|p\rangle"] \arrow[ddl, "m+m';\sim"'] \arrow[dr, "\langle\alpha|\alpha'\rangle;b_0"] & & & P \arrow[ddl, "m"'] \arrow[dr, "\alpha"] & \\
 & & S+1 \arrow[rrr, "\langle 1|i\rangle"] & & & S \\
(S+1)\times\Sigma \arrow[rrr, "\langle 1|i\rangle\times 1"'] & & & S\times\Sigma & & \\
 & & 1 \arrow[uu, "b_0"] \arrow[rrr, dash] & & & 1 \arrow[uu, "i"']
\end{tikzcd}$$

**Lemma 2.5** $(\eta, \epsilon) : F \dashv G : \mathrm{Src}(\mathbf{C}) \longrightarrow \mathrm{DTran}(\mathbf{C})$.

[2]A lextensive category is distributive, and thus the canonical map $A \times B + A \times C \longrightarrow A \times (B + C)$ is an isomorphism.

**Proof.** $F$ and $G$ are functors which, since $\mathbf{C}$ is lextensive, are stable. That $\eta$ and $\epsilon$ are maps is seen in the following:

$$\begin{tikzcd}
S \arrow[r, "b_0"] \arrow[dr, dash, "="] & S+1 \\
 & S \arrow[u, "b_0"'] \\
P_1+P_0 \arrow[uu, "\langle\alpha_0|\alpha_1\rangle"] \arrow[r, "b_0+1"] \arrow[d, "m_1+m_0"'] \arrow[dr, phantom, "\lrcorner", very near start] & P_1+P_0+P_0 \arrow[u, "\langle\langle\alpha_0|\alpha_1\rangle|\alpha_1\rangle"'] \arrow[d, "m_1+m_0+m_0"] \\
S\times\Sigma+1\times\Sigma \arrow[r, "b_0+1"] \arrow[d, dash, "="'] \arrow[dr, phantom, "\lrcorner", very near start] & S\times\Sigma+1\times\Sigma+1\times\Sigma \arrow[d, "\sim+1"] \\
S\times\Sigma+1\times\Sigma \arrow[r, "b_0\times 1+1"] \arrow[d, "\sim"'] \arrow[dr, phantom, "\lrcorner", very near start] & (S+1)\times\Sigma+1\times\Sigma \arrow[d, "\sim"] \\
(S+1)\times\Sigma \arrow[r, "(b_0+1)\times 1"'] & (S+1+1)\times\Sigma
\end{tikzcd}
\qquad
\begin{tikzcd}
S+1 \arrow[r, "\langle 1|i\rangle"] & S \\
S \arrow[u, "b_0"] \arrow[ur, dash, "="'] & \\
P+P' \arrow[u, "\langle\alpha|\alpha'\rangle"] \arrow[r, "\langle 1|p\rangle"] \arrow[d, "m+m'"'] \arrow[dr, phantom, "\lrcorner", very near start] & P \arrow[uu, "\alpha"'] \arrow[d, "m"] \\
S\times\Sigma+1\times\Sigma \arrow[r, "\langle 1|i\times 1\rangle"] \arrow[d, "\sim"'] \arrow[dr, phantom, "\lrcorner", very near start] & S\times\Sigma \arrow[d, dash, "="] \\
(S+1)\times\Sigma \arrow[r, "\langle 1|i\rangle\times 1"'] & S\times\Sigma
\end{tikzcd}$$

Naturality is given by their construction.

Checking the triangle identities componentwise amounts to checking the following equations: for $F\eta;\epsilon_F = 1_F$, $b_0+1;\langle 1|b_0\rangle = 1_P$ and $b_0+1;\langle 1|b_0\rangle = 1_{S+1}$; for $\eta_G;G\epsilon = 1_G$, $b_0+1;\langle 1|p\rangle+1 = 1_P$ and $b_0;\langle 1|i\rangle = 1_S$. □

The fact that the indicated squares are pullbacks will be important later when we consider transmitting this adjunction through the process construction.

## 2.3 Nondeterministic Transition Systems

Let $\mathbf{C}$ be a lextensive category,

**Definition 2.6** Tran($\mathbf{C}$) *is the category of models in* $\mathbf{C}$ *of the following (finite limit) sketch:*

$$\begin{tikzcd}
R \arrow[d, "r"'] & 1 \arrow[d, "i"] \\
S\times\Sigma\times S & S
\end{tikzcd}$$

Tran(**Set**) is the category of transition systems as presented by Winskel [Win87]. For lextensive $\mathbf{C}$, as is the case for DTran($\mathbf{C}$), the category Tran($\mathbf{C}$) has finite limits, but not coproducts.

**Lemma 2.7** Tran($\mathbf{C}$) *has finite limits.*

A deterministic transition is naturally a nondeterministic transition system as well:

$$\begin{tikzcd}
& P \arrow[dl, "m"'] \arrow[dr, "\alpha"] & & & {} \arrow[r, mapsto, "G"] & {} & P \arrow[d, "\langle m,\alpha\rangle"'] & 1 \arrow[d, "i"] \\
S\times\Sigma & & S & 1 \arrow[l, "i"] & & & S\times\Sigma\times S & S
\end{tikzcd}$$

One can make a nondeterministic transition system deterministic by adding (target) state information to the labels, thus distinguishing the actions leaving each state:

$$\begin{tikzcd}
& & & & R \arrow[d, "r"] & & \\
R \arrow[d, "r"'] & 1 \arrow[d, "i"] & {} \arrow[r, mapsto, "F"] & {} & S\times\Sigma\times S \arrow[dl, "a"'] \arrow[dr, "p_1"] & & \\
S\times\Sigma\times S & S & & S\times(\Sigma\times S) & & S & 1 \arrow[l, "i"]
\end{tikzcd}$$

These are easily shown to be stable functors which preserve $\exists^{\mathcal{C}}$.

Although one might expect an adjunction between DTran(**C**) and Tran(**C**), there is no apparent unit. There does, however, appear to be two natural choices for a counit: $\epsilon : GFA \Longrightarrow A$

$$\begin{tikzcd}
R \arrow[rr, no head, "="] \arrow[d, "r"'] & & R \arrow[dd, "r"] \\
S\times\Sigma\times S \arrow[d, "\langle a,p_1\rangle"'] & & \\
S\times(\Sigma\times S)\times S \arrow[rr, "1\times p_0\times 1"'] & & S\times\Sigma\times S
\end{tikzcd}$$

and $\epsilon' : FGB \Longrightarrow B$

$$\begin{tikzcd}
& P \arrow[rrr, no head, "="] \arrow[d, "\langle m,\alpha\rangle"'] & & & P \arrow[ddl, "m"'] \arrow[dr, "\alpha"] & \\
& S\times\Sigma\times S \arrow[dl, "a"'] \arrow[r, "p_1"] & S \arrow[rrr, no head, "="] & & & S \\
S\times(\Sigma\times S) \arrow[rrr, "1\times p_0"'] & & & S\times\Sigma & &
\end{tikzcd}$$

Each of these simply eliminates the state information introduced in the labels.

**Lemma 2.8** *As described above, $F$ and $G$ are stable functors and $\epsilon$ and $\epsilon'$ are natural transformations.*

We will see later that these transformations do indeed give rise to an equivalence between the respective process categories.

## 3 Cover Systems

This section describes the basic theory of cover systems, which play a central role in the subsequent development: cover systems will be used to express bisimulation equivalence of processes — or more precisely, to obtain congruences on the morphisms of span categories. We will describe cover systems for each of the model categories of the previous section.

Let $\mathbf{X}$ be a category with pullbacks,

**Definition 3.1** *A collection $\mathcal{X}$ of the maps of* $\mathbf{X}$ *is a* **cover system** *provided it contains all isomorphisms, is closed to composition, and is closed to pulling back along arbitrary maps — i.e. if $x$ is in $\mathcal{X}$ and the following is a pullback then $y$ is in $\mathcal{X}$:*

$$\begin{tikzcd}
\cdot \arrow[r] \arrow[d, "y"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \cdot \arrow[d, "x"] \\
\cdot \arrow[r] & \cdot
\end{tikzcd}$$

Cover systems will provide a weak notion of when two objects are the same. Examples of cover systems in any category are the isomorphisms $\mathcal{I}$, the retractions $\mathcal{R}$, and the monics $\mathcal{M}$. In a regular category, the regular epimorphisms $\mathcal{E}$ form a cover system. We say that a cover system $\mathcal{X}$ is *left-factor closed* if $f$ is in $\mathcal{X}$ whenever both $g$ and $f;g$ are in $\mathcal{X}$. Thus $\mathcal{I}$ and $\mathcal{M}$ are left-factor closed cover systems, while $\mathcal{R}$ and $\mathcal{E}$ are not. Note that the classes of cover systems and left-factor closed cover systems are closed to intersection.

Let $\mathcal{X}$ be a cover system on $\mathbf{X}$,

**Definition 3.2** *A commuting square in* $\mathbf{X}$ *is an* $\mathcal{X}$**-pullback** *if the induced map to the inscribed pullback is in $\mathcal{X}$:*

$$\begin{tikzcd}
\cdot \arrow[rr] \arrow[dd] \arrow[dr, dashed, "x"] & & \cdot \arrow[dd] \\
& \cdot \arrow[ur] \arrow[dl] \arrow[dr, phantom, very near start, "\lrcorner"] & \\
\cdot \arrow[rr] & & \cdot
\end{tikzcd}$$

A pullback is an $\mathcal{X}$-pullback for any $\mathcal{X}$. Furthermore, a map $f$ is an $\mathcal{X}$-map if and only if the square $f;1 = f;1$ is an $\mathcal{X}$-pullback.

Cover-pullbacks are important for obtaining equivalence on the maps of process categories. The following shows that $\mathcal{X}$-pullbacks satisfy much the same properties as pullbacks.

**Lemma 3.3** *In a category with a cover system* $(\mathbf{X}, \mathcal{X})$*:*

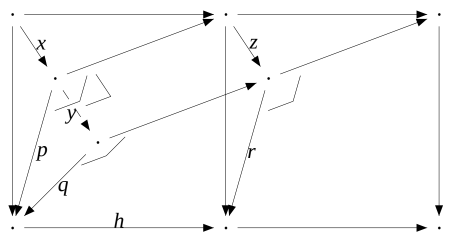
$$\begin{tikzcd}
\cdot \arrow[r] \arrow[d] \arrow[dr, phantom, "(1)"] & \cdot \arrow[r] \arrow[d] \arrow[dr, phantom, "(2)"] & \cdot \arrow[d] \\
\cdot \arrow[r] & \cdot \arrow[r] & \cdot
\end{tikzcd}$$

*(i) the outer square is an* $\mathcal{X}$*-pullback whenever (1) and (2) are* $\mathcal{X}$*-pullbacks;*

*(ii) (1) is an* $\mathcal{X}$*-pullback whenever (2) is a pullback and the outer square is an* $\mathcal{X}$*-pullback;*

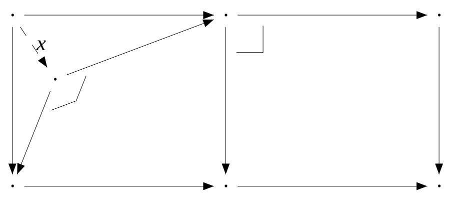
*(iii)* $\mathcal{X}$ *is left-factor closed if and only if for all diagrams of this shape (1) is an* $\mathcal{X}$*-pullback whenever the outer square and (2) are* $\mathcal{X}$*-pullbacks.*

**Proof**. For (i), let $x$ and $z$ witness that (1) and (2) are $\mathcal{X}$-pullbacks.

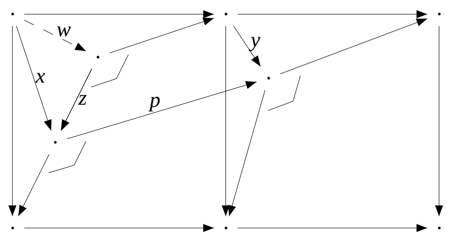


Forming the pullback of $r$ and $h$, one obtains the pullback inscribed in the entire square; the induced map to this pullback is $x;y$, where $y$ is the unique factoring of $p$ through $q$. As $y$ is an $\mathcal{X}$-map, $x;y$ is also.

For (ii), the induced map to the pullback inscribed in (1) is the induced map to the pullback inscribed in the outer square.



For (iii), suppose $\mathcal{X}$ is left-factor closed and let $x$ and $y$ witness that the outer square and (2) are $\mathcal{X}$-pullbacks.



Forming the pullback of $y$ along $p$ gives a pullback inscribed in (1); the induced map $w$ is an $\mathcal{X}$-map since $x$ and $z$ are $\mathcal{X}$-maps. Conversely, if $f;y$ and $y$ are $\mathcal{X}$-maps then, using parts (i) and (ii) above, the top left square of

$$\begin{tikzcd}
\cdot \arrow[r, "f"] \arrow[d, "f"'] & \cdot \arrow[r, "y"] \arrow[d, equal] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d, equal] \\
\cdot \arrow[r, equal] \arrow[d, "y"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r, "y"] \arrow[d, "y"'] & \cdot \arrow[d, equal] \\
\cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot
\end{tikzcd}$$

is an $\mathcal{X}$-pullback and thus $f$ is an $\mathcal{X}$-map. □

Consequently, one can obtain a cover system in a category of functors and natural transformations by considering the transformations for which chosen naturality squares are cover-pullbacks. If $\alpha$ is a transformation of functors $\mathbf{Y} \longrightarrow \mathbf{X}$ and $\mathcal{A}$ is a collection of some arrows of $\mathbf{Y}$, we say that $\alpha$ is $\mathcal{X}$-*cartesian for* $\mathcal{A}$ when the naturality squares associated with each arrow of $\mathcal{A}$ are $\mathcal{X}$-pullbacks.

**Proposition 3.4** *Let $F, G : \mathbf{Y} \longrightarrow \mathbf{X}$, $\mathcal{X}$ a cover system on $\mathbf{X}$, and $\mathcal{A}$ a collection of arrows of $\mathbf{Y}$. The transformations $\alpha : F \Longrightarrow G$ which are $\mathcal{X}$-cartesian for $\mathcal{A}$ form a cover system in the functor category $Fun(\mathbf{Y}, \mathbf{X})$; this cover system is left-factor closed if and only if $\mathcal{X}$ is left-factor closed.*

**Proof**. The proposed cover system contains natural isomorphisms, as these are cartesian, and is closed to composition by part (i) of lemma 3.3. Parts (i) and (ii) of the same lemma show that the rear face of

$$\begin{tikzcd}
\cdot \arrow[rr, "\gamma_A"] \arrow[dr] \arrow[dd, dashed, "Pf"'] & & \cdot \arrow[dr, "\beta_A"] \arrow[dd, "Hf" near start] & \\
& \cdot \arrow[rr, "\alpha_A" near start] \arrow[dd, "Ff" near start] & & \cdot \arrow[dd, "Gf"] \\
\cdot \arrow[rr, "\gamma_B" near end] \arrow[dr] & & \cdot \arrow[dr, "\beta_B"] & \\
& \cdot \arrow[rr, "\alpha_B"'] & & \cdot
\end{tikzcd}$$

is an $\mathcal{X}$-pullback whenever the front face is an $\mathcal{X}$-pullback. Thus if $\gamma$ results from pulling back $\alpha$ along $\beta$, then $\gamma$ is $\mathcal{X}$-cartesian for $\mathcal{A}$ whenever $\alpha$ is.

The rest is immediate from part (iii) of lemma 3.3. □

This gives a method of obtaining the cover systems on model categories — if $\mathcal{S}$ is a sketch and $a$ an arrow of $\mathcal{S}$, the model morphisms of $\mathcal{S}(\mathbf{X})$ which are

$\mathcal{X}$-cartesian for $a$ form a cover system; this cover system is left-factor closed whenever $\mathcal{X}$ is left-factor closed.

Let $\mathcal{C}$ be a cover system on a lextensive category $\mathbf{C}$. The class of maps of Tran($\mathbf{C}$) which are $\mathcal{C}$-cartesian for the arrow $r;p_0;p_0$ will be referred to as $\exists^{\mathcal{C}}$. Each map of $\exists^{\mathcal{C}}$ thus has the following square a $\mathcal{C}$-pullback in $\mathbf{C}$:

$$\begin{tikzcd}
R_A \arrow[r, "r"] \arrow[d, "f_R"'] & S_A{\times}\Sigma_A{\times}S_A \arrow[r, "p_0"] & S_A{\times}\Sigma_A \arrow[r, "p_0"] & S_A \arrow[d, "f_S"] \\
R_B \arrow[r, "r"] & S_B{\times}\Sigma_B{\times}S_B \arrow[r, "p_0"] & S_B{\times}\Sigma_B \arrow[r, "p_0"] & S_B
\end{tikzcd}$$

For instance, in Tran($\mathbf{Set}$) a map $f : A \longrightarrow B$ of $\exists^{\mathcal{I}}$ has the square above a pullback which means that each transition from a state $f(s)$ of $B$ is the image (via $f$) of a unique transition from state $s$ of $A$. A map $f : A \longrightarrow B$ of $\exists^{\mathcal{R}}$ has the weaker property that each transition from $f(s)$ is the image of at least one transition from $s$.

Within a fibre Tran($\mathbf{Set}$)$_\Sigma$ over alphabet $\Sigma$, objects $A$ and $B$ are bisimilar whenever they are related by a span of (vertical) $\exists^{\mathcal{R}}$-maps:

$$\begin{tikzcd}
 & R \arrow[dl, "f"'] \arrow[dr, "g"] & \\
A & & B
\end{tikzcd}$$

To see this, choose any state $r$ of the "relation" $R$; if $f(r)\xrightarrow{x}s'$ in $A$ then there exists $r'$ such that $f(r') = s'$ and $r\xrightarrow{x}r'$ in $R$, so $g(r)\xrightarrow{x}g(r')$ in $B$. As maps $f$ and $g$ preserve initial states, all reachable states of $A$ and $B$ "appear" in $R$. Indeed, for Tran($\mathbf{Set}$), $\exists^{\mathcal{R}}$ is just another view of the cover system given in [JNW93].

One obtains cover systems for Src($\mathbf{C}$) and DTran($\mathbf{C}$) similarly, as the classes of maps which are $\mathcal{C}$-cartesian for the respective arrow $m;p_0$. We will refer to each of these cover systems as $\exists^{\mathcal{C}}$ also, relying on context to disambiguate. Note that in each case $\exists^{\mathcal{I}}$ is left-factor closed, while $\exists^{\mathcal{R}}$ is not.

There is one more important method of generating cover systems which we will require. This involves taking the preimage of an "almost" stable functor:

**Lemma 3.5** *If $\mathcal{X}$ is a cover system on $\mathbf{X}$ and $F : \mathbf{Y} \longrightarrow \mathbf{X}$ takes pullbacks to $\mathcal{X}$-pullbacks, then $F^{-1}(\mathcal{X})$ is a cover system on $\mathbf{Y}$ which is left-factor closed whenever $\mathcal{X}$ is left-factor closed.*

**Proof.** $F^{-1}(\mathcal{X})$ contains all isomorphisms and is closed to composition since $F$ is a functor. To see it is closed to pullback, let $Ff \in \mathcal{X}$ and $p; f = q; g$ be a pullback in $\mathbf{Y}$.

$$\begin{tikzcd}
\cdot \arrow[rr, "Fp"] \arrow[dr, "x"] \arrow[dd, "Fq"'] & & \cdot \arrow[dd, "Ff"] \\
& \cdot \arrow[ur] \arrow[dl, "z"] & \\
\cdot \arrow[rr, "Fg"'] & & \cdot
\end{tikzcd}$$

The diagram above then has both $x$ and $z$ in $\mathcal{X}$, and thus $Fq \in \mathcal{X}$. That $F^{-1}(\mathcal{X})$ inherits left-factor closure is obvious. $\square$

Thus, for instance, given any stable functor $F : \mathbf{Y} \longrightarrow \mathbf{Z}$ one has a (left-factor closed) cover system $F^{-1}(\mathcal{I})$ on $\mathbf{Y}$ consisting of all the maps which are taken to isomorphisms by $F$.

Cover systems play a central role in the construction of process categories. In fact, the construction will be performed in a 2-category $\mathbf{Cov}$ defined as follows:

· 0-cells $(\mathbf{X}, \mathcal{X})$ are categories with cover systems;

· 1-cells $F : (\mathbf{X}, \mathcal{X}) \longrightarrow (\mathbf{Y}, \mathcal{Y})$ are functors $\mathbf{X} \longrightarrow \mathbf{Y}$ which are *$\mathcal{X}$-stable* (or *cover-stable*) in that $\mathcal{X}$-pullbacks are taken to $\mathcal{Y}$-pullbacks;

· 2-cells $\alpha : F \Longrightarrow G : (\mathbf{X}, \mathcal{X}) \longrightarrow (\mathbf{Y}, \mathcal{Y})$ are natural transformations $F \Longrightarrow G$ which are *$\mathcal{Y}$-cartesian* (or *cover-cartesian*) in that all naturality squares are $\mathcal{Y}$-pullbacks.

The following results will be useful for establishing when functors and natural transformations belong to $\mathbf{Cov}$:

**Lemma 3.6** *Let $(\mathbf{X}, \mathcal{X})$ and $(\mathbf{Y}, \mathcal{Y})$ be categories with cover systems, and $F : \mathbf{X} \longrightarrow \mathbf{Y}$. Then $F : (\mathbf{X}, \mathcal{X}) \longrightarrow (\mathbf{Y}, \mathcal{Y})$ if and only if $F$ preserves $\mathcal{X}$ and takes pullbacks to $\mathcal{Y}$-pullbacks.*

**Proof.** If $F$ is $\mathcal{X}$-stable and $c$ is an $\mathcal{X}$-map then, since $c;1 = c;1$ is an $\mathcal{X}$-pullback, $Fc;1 = Fc;1$ is a $\mathcal{Y}$-pullback and so $Fc$ is a $\mathcal{Y}$-map.

Conversely, if $c$ is witness that $h;f = k;g$ is an $\mathcal{X}$-pullback then $Fc;d$ is witness that the outer square below is a $\mathcal{Y}$-pullback.

$$\begin{tikzcd}
\cdot \arrow[rrr, "Fh"] \arrow[dr, "Fc"] \arrow[ddd, "Fk"'] & & & \cdot \arrow[ddd, "Ff"] \\
& \cdot \arrow[urr] \arrow[ddl] \arrow[dr, "d"] & & \\
& & \cdot \arrow[ur] \arrow[dll] & \\
\cdot \arrow[rrr, "Fg"'] & & & \cdot
\end{tikzcd}$$

$\square$

The following shows that natural transformations whose components belong to a left-factor closed cover system are automatically cover-cartesian.

**Lemma 3.7** *Let $F$ and $G$ be functors $(\mathbf{Y}, \mathcal{Y}) \longrightarrow (\mathbf{X}, \mathcal{X})$ with $\mathcal{X}$ left-factor closed. If $\alpha : F \Longrightarrow G$ is such that $\alpha_A \in \mathcal{X}$ for all $A \in \mathbf{Y}$, then $\alpha$ is $\mathcal{X}$-cartesian.*

**Proof.** This is easily seen in the diagram of definition 3.2. $\square$

Transformation which pointwise belong to left-factor closed cover systems will play an important role in demonstrating equivalence of process categories.

## 4 Process Categories

We regard a *process* on a model category such as DTran($\mathbf{C}$) as a *span*:

$$\begin{tikzcd} & P \arrow[dl, "f"'] \arrow[dr, "g"] & \\ A & & B \end{tikzcd}$$

The endpoints $A$ and $B$ act as interface specifications, and the legs $f$ and $g$ indicate how $P$ adheres to the specification — in the case of DTran($\mathbf{C}$), how the actions of $P$ determine visible actions simultaneously at each interface.

In general, the process category of a model category with a cover system is a span category whose maps are quotiented by the cover system. This construction is 2-functorial, allowing one to determine properties of the process categories in terms of the model categories upon which they are constructed. This is illustrated on the model categories of section 2, and it is shown that each yields the same process categories.

From any category $\mathbf{X}$ with pullbacks one can form the bicategory of spans in $\mathbf{X}$ (see Bénabou [Ben67]): the objects are those of $\mathbf{X}$; 1-cells $A \longrightarrow B$ are spans $(f, g)$ in $\mathbf{X}$; and 2-cells $(f, g) \longrightarrow (f', g')$ are given by maps $h$ of $\mathbf{X}$ such that

$$\begin{tikzcd} & P \arrow[dl, "f"'] \arrow[dr, "g"] \arrow[dd, "h"] & \\ A & & B \\ & P' \arrow[ul, "f'"] \arrow[ur, "g'"'] & \end{tikzcd}$$

commutes in $\mathbf{X}$. 1-cell composition is given by pullback, and is thus determined only to isomorphism — i.e. $(f, g);(h, k)$ is $(p; f,\ q; k)$, where:

$$\begin{tikzcd} & & R \arrow[dl, "p"'] \arrow[dr, "q"] \arrow[dd, phantom, "\lrcorner" rotate=-45, very near start] & & \\ & P \arrow[dl, "f"'] \arrow[dr, "g"] & & Q \arrow[dl, "h"'] \arrow[dr, "k"] & \\ A & & B & & C \end{tikzcd}$$

We say that spans $(f, g)$ and $(h, k)$ are $\mathcal{X}$*-bisimilar*, written $\sim_{\mathcal{X}}$, when there exist $\mathcal{X}$-maps $x$ and $y$ such that

$$\begin{tikzcd} & P \arrow[dl, "f"'] \arrow[dr, "g"] & \\ A & R \arrow[u, "x"] \arrow[d, "y"'] & B \\ & Q \arrow[ul, "h"] \arrow[ur, "k"'] & \end{tikzcd}$$

commutes in $\mathbf{X}$.

**Lemma 4.1** *$\sim_{\mathcal{X}}$ is a congruence on the spans of $\mathbf{X}$ (with respect to span composition).*

**Proof**. That $\sim_{\mathcal{X}}$ is an equivalence relation follows easily from the axioms of a cover system: reflexivity holds as identities are covers; symmetry is obvious; and transitivity holds as covers are closed to composition and pullback.

Suppose that $(x, x')$ is a bisimulation of $(f, g)$ and $(f', g')$, and that $(h, k)$ is of the appropriate type for composition. Forming the pullback of $h$ and $x; g$ induces maps $z$ and $z'$ to the "composites";

$$\begin{tikzcd}
 & \cdot \arrow[dl] \arrow[drrrrr] \arrow[dr, phantom, "\lrcorner", very near start] & & \cdot \arrow[ll, dashed, "z"'] \arrow[rr, dashed, "z'"] \arrow[dl] \arrow[drrr] \arrow[d, phantom, "\lrcorner", very near start] & & \cdot \arrow[dl] \arrow[dr] \arrow[d, phantom, "\lrcorner", very near start] & \\
P \arrow[drrrr, "g"'] & & R \arrow[ll, "x"'] \arrow[rr, "x'"] \arrow[drr, dashed] & & P' \arrow[d, "g'"] & & Q \arrow[dll, "h"] \\
 & & & & B & &
\end{tikzcd}$$

these maps are covers as the rear faces are pullbacks. Thus $(z, z')$ is a bisimulation of $(f, g); (h, k)$ and $(f', g'); (h, k)$. $\square$

Thus quotienting 1-cells by $\sim_{\mathcal{X}}$, and ignoring the extant 2-cell structure, one obtains a category $\mathrm{Proc}(\mathbf{X}, \mathcal{X})$ of processes up-to the specified equivalence. Certainly the simplest examples of this construction are span categories and categories of relations: for $\mathbf{X}$ with pullbacks, $\mathrm{Proc}(\mathbf{X}, \mathcal{I})$ is written $\mathrm{Span}(\mathbf{X})$; and for $\mathbf{E}$ a regular category, $\mathrm{Proc}(\mathbf{E}, \mathcal{E})$ is written $\mathrm{Rel}(\mathbf{E})$.

As in the paradigm of interaction categories, composition here corresponds clearly to process interaction, or restricted "parallel composition" in terms of CCS.

The construction of process categories is in fact a 2-functor from the 2-category **Cov** discussed previously: $\mathrm{Proc}(F)$ applies functor $F$ to each leg of a span (i.e. $(f, g) \mapsto (Ff, Fg)$); for $\alpha : F \Longrightarrow G$ a natural transformation, the component of $\mathrm{Proc}(\alpha)$ at $A$ is the trivial span $(1_{GA}, \alpha_A)$.

**Proposition 4.2** $\mathrm{Proc} : \mathbf{Cov} \longrightarrow \mathbf{Cat}$ *is a 2-functor which preserves products.*

**Proof**. To see that $\mathrm{Proc}(F)$ is a functor: it preserves equivalence of spans as $F$ preserves cover maps; it preserves identities as $F$ is a functor; and it preserves composition as $F$ takes pullbacks to cover-pullbacks.

$$\begin{tikzcd}
 & & FR \arrow[dl] \arrow[dr] \arrow[d, dashed] & & \\
 & FP \arrow[dl] \arrow[dr] & \cdot \arrow[l] \arrow[r] \arrow[d, phantom, "\vee", very near start] & FQ \arrow[dl] \arrow[dr] & \\
FA & & FB & & FC
\end{tikzcd}$$

Proc($\alpha$) is natural as $\alpha$ is cover-cartesian:

$$\begin{tikzcd}
GA & GA \arrow[l, "="'] \arrow[rr, "\alpha_A"] & & HA \\
GP \arrow[u, "Gf"] \arrow[d, "Gg"'] & GP \arrow[l, "="] \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[rr, bend right, "\alpha_P"'] & \cdot \arrow[ul] \arrow[r] & HP \arrow[u, "Hf"'] \arrow[d, "Hg"] \\
GB & GB \arrow[l, "="] \arrow[rr, "\alpha_B"'] & & HB
\end{tikzcd}$$

□

Thus any functorial structure on $\mathbf{X}$ will occur also in Proc($\mathbf{X}, \mathcal{X}$) provided the functors and natural transformations involved are cover-stable and cover-cartesian respectively.

Concerning products, these are unlikely to pass through the process construction as the projections and copy transformation are not cartesian. However, the product in $\mathbf{X}$ will produce a symmetric tensor in Proc($\mathbf{X}, \mathcal{X}$) provided $\times$ is $\mathcal{X}$-stable (which it is — see below). Any category with products provides a trivial example of a weakly distributive category (see Cockett and Seely [CS92]), by identifying the tensor and cotensor with $\times$ and taking the distributions as the natural maps given by associativity and symmetry. This weakly distributive structure is transmitted onto the process category by the 2-functor Proc. In fact,

**Proposition 4.3** *If* $\mathbf{X}$ *has finite products then* Proc($\mathbf{X}, \mathcal{X}$) *is compact-closed for any cover system* $\mathcal{X}$*.*

**Proof.** The functor $\times$ is stable and furthermore preserves covers as the following are always pullbacks:

$$\begin{tikzcd}
A\times C \arrow[r, "f\times 1"] \arrow[d, "p_0"'] \arrow[dr, phantom, "\lrcorner", very near start] & B\times C \arrow[d, "p_0"] & B\times C \arrow[r, "1\times g"] \arrow[d, "p_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & B\times D \arrow[d, "p_1"] \\
A \arrow[r, "f"'] & B & C \arrow[r, "g"'] & D
\end{tikzcd}$$

Being isomorphisms, the weak distribution maps are cover-cartesian and therefore $\mathrm{Proc}(\mathbf{X}, \mathcal{X})$ is a symmetric weakly distributive category as well.

To obtain the *-autonomous structure, take the involution to be the span reversal (which has identity effect on objects). The complementation map $\tau_A$ is the following span

$$\begin{tikzcd}
 & A \arrow[dl, "!"'] \arrow[dr, "\Delta"] & \\
1 & & A\times A
\end{tikzcd}$$

and $\gamma_A$ is it's involution. As the tensor and cotensor coincide, checking the complementation diagram reduces to verifying that $1_{A\times 1} = 1\times\tau;\ \delta^L_R;\ 1\times\gamma$. This is shown in the diagram below,

$$\begin{tikzcd}
 & & & A \arrow[dl, "\Delta"'] \arrow[rrrddd, "\Delta"] & & & & & \\
 & & \cdot \arrow[dl, "="'] \arrow[ddr] \arrow[dddrr, "1\times\Delta"] & & & & & & \\
 & \cdot \arrow[ddl, "1\times !"'] \arrow[ddr, "1\times\Delta"] & & & (1) & & & & \\
 & & & \cdot \arrow[dl, "="'] \arrow[dr, "1\times s"'] & & & \cdot \arrow[ddl, "\Delta\times 1"'] \arrow[dr, "s"] & & \\
A\times 1 & & \cdot & & \cdot \arrow[dr, "a"'] & & & \cdot \arrow[ddl, "1\times\Delta"'] \arrow[ddr, "1\times !"] & \\
 & & & & & \cdot \arrow[dr, "s"'] & & & \\
 & & & & & & \cdot & & A\times 1
\end{tikzcd}$$

where we note that (1) is always a pullback and that $\delta^L_R$ is defined in $\mathbf{X}$ as $1\times s;\ a;\ s$. $\square$

Thus, for instance, the categories $\mathrm{Span}(\mathbf{X})$ for a category $\mathbf{X}$ with products and $\mathrm{Rel}(\mathbf{E})$ for a regular category $\mathbf{E}$ with products are compact-closed. Moreover, the process categories of each of the model categories of section 2 are compact-closed as well.

Regarding coproducts, note that if they are given by a **Cov**-adjunction then they will exist in $\mathrm{Proc}(\mathbf{X}, \mathcal{X})$. Furthermore, they must be simultaneously products and coproducts (since the process category is self dual) and are thus biproducts. In a lextensive category, $+$ is stable and the associated transformations are cartesian. Thus,

**Proposition 4.4** *If* $\mathbf{C}$ *is lextensive with a cover system* $\mathcal{C}$ *and* $+$ *preserves* $\mathcal{C}$*, then* $\mathrm{Proc}(\mathbf{C}, \mathcal{C})$ *has finite biproducts.*

As $\mathrm{Src}(\mathbf{C})$ is lextensive, for its process category to have finite biproducts it is sufficient that coproducts in $\mathrm{Src}(\mathbf{C})$ preserve the cover system $\exists^{\mathcal{C}}$.

**Lemma 4.5** *Coproducts in* $\mathrm{Src}(\mathbf{C})$ *preserve* $\exists^{\mathcal{C}}$ *whenever coproducts in* $\mathbf{C}$ *preserve* $\mathcal{C}$*.*

**Proof.** Suppose $f \in \exists^{\mathcal{C}}$, with $c \in \mathcal{C}$ its witness and $q$ and $q'$ the projections of the associated pullback — we show that $f+1 \in \exists^{\mathcal{C}}$. As coproducts in $\mathbf{C}$ preserve pullbacks and $\mathcal{C}$-maps, the outer square below is a $\mathcal{C}$-pullback:

$$\begin{tikzcd}
P_A+P \arrow[rr, "f_P+1"] \arrow[dr, "c+1"] \arrow[dd, "m_A+m"'] & & P_B+P \arrow[dd, "m_B+m"] & \\
& Q+P \arrow[ur, "q'+1"'] \arrow[dddd, "q+(m;p_0)"] \arrow[dddd, phantom, very near start, "\lrcorner"] & & \\
(S_A+1)\times\Sigma_A+(S+1)\times\Sigma \arrow[d, "\gamma"'] & & (S_B+1)\times\Sigma_B+(S+1)\times\Sigma \arrow[ddd, "p_0+p_0"'] \arrow[dr, "\gamma"] & \\
(S_A+S+1)\times(\Sigma_A+\Sigma) \arrow[ddd, "p_0"'] & & & (S_B+S+1)\times(\Sigma_B+\Sigma) \arrow[dddl, "p_0"] \\
& & & \\
& S_A+1+(S+1) \arrow[r, "f_S+1+1"'] \arrow[dl, "\gamma_1"] \arrow[dl, phantom, very near start, "\lrcorner"] & S_B+1+(S+1) \arrow[d, "\gamma_1"'] & \\
S_A+S+1 \arrow[rr, "f_S+1+1"'] & & S_B+S+1 &
\end{tikzcd}$$

Here $\gamma$ and $\gamma_1$ are the transformations $\langle(b_0+1)\times b_0|(b_1+1)\times b_1\rangle$ and $\langle b_0+1|b_1+1\rangle$ which are cartesian. Similarly $g \in \exists^{\mathcal{C}}$ implies $1+g \in \exists^{\mathcal{C}}$ and thus coproducts preserve $\exists^{\mathcal{C}}$. □

Since isomorphisms and retractions are preserved by any functor,

**Corollary 4.6** Proc(Src($\mathbf{C}$), $\exists^{\mathcal{I}}$) *and* Proc(Src($\mathbf{C}$), $\exists^{\mathcal{R}}$) *are compact-closed with finite biproducts.*

Although having $\mathbf{C}$ lextensive is equivalent to Span($\mathbf{C}$) having finite biproducts, being lextensive is not necessary for the process category to have biproducts — in fact the base category need not have coproducts at all. The following results will reveal a considerable latitude in the choice of base category when constructing a specific process category.

**Lemma 4.7** *If $\mathcal{X}$ is a left-factor closed cover system on* $\mathbf{X}$, *then spans of $\mathcal{X}$-maps are isomorphisms in* Proc($\mathbf{X}, \mathcal{X}$).

**Proof.** The inverse of a span $(x,y)$ of $\mathcal{X}$-maps is given by its span reversal.

$$\begin{tikzcd}
& & A \arrow[ddddll, dash, "="'] \arrow[ddddrr, dash, "="] & & \\
& & P \arrow[u, "x"] \arrow[ddl, dash, "="'] \arrow[ddr, dash, "="] \arrow[d, dashed, "z"] & & \\
& & \cdot \arrow[dl, "y_1"] \arrow[dr, "y_2"'] \arrow[d, phantom, very near start, "\lrcorner"] & & \\
& P \arrow[dl, "x"'] \arrow[dr, "y"] & & P \arrow[dl, "y"'] \arrow[dr, "x"] & \\
A & & B & & A
\end{tikzcd}$$

As $\mathcal{X}$ is left-factor closed, the map $z$ induced to the pullback of $y$ and $y$ is an $\mathcal{X}$-map. Thus $(x, z)$ is a bisimulation of $(1_A, 1_A)$ and $(x, y); (y, x)$. Similarly, $(1_B, 1_B) = (y, x); (x, y)$. $\square$

Consequently, natural transformations all of whose components belong to a left-factor closed cover system become natural isomorphisms via the 2-functor Proc.

**Corollary 4.8** *If* $\alpha : F \Longrightarrow G : (\mathbf{Y}, \mathcal{Y}) \longrightarrow (\mathbf{X}, \mathcal{X})$ *with* $\mathcal{X}' \subseteq \mathcal{X}$ *left-factor closed and* $\alpha_A \in \mathcal{X}'$ *for all* $A \in \mathbf{Y}$, *then* $\mathrm{Proc}(\alpha)$ *is a natural isomorphism.*

Thus an adjoint equivalence can be obtained from an adjunction which has its unit and counit belonging to left-factor closed cover systems at each component. Such is the adjunction between $\mathrm{Src}(\mathbf{C})$ and $\mathrm{DTran}(\mathbf{C})$ described in section 2. For any cover system $\mathcal{C}$ on $\mathbf{C}$,

**Proposition 4.9** $\mathrm{Proc}(\mathrm{Src}(\mathbf{C}), \exists^{\mathcal{C}})$ *is equivalent to* $\mathrm{Proc}(\mathrm{DTran}(\mathbf{C}), \exists^{\mathcal{C}})$.

**Proof.** We need only show that the adjunction is in **Cov**, and that the unit and counit give rise to isomorphisms.

$F$ and $G$ are stable functors since $\mathbf{C}$ is lextensive. $F$ trivially preserves $\exists^{\mathcal{C}}$-maps, while $G$ preserves $\exists^{\mathcal{C}}$ whenever coproducts in $\mathbf{C}$ preserve $\mathcal{C}$. To see the latter, suppose $f \in \exists^{\mathcal{C}}$. Note first that the rear face of

$$\begin{tikzcd}
P_A' \arrow[rr, "m_A'"] \arrow[dr] \arrow[dd, dashed, "f_P'"'] & & 1\times\Sigma_A \arrow[rr, "p_0"] \arrow[dr, "1\times i_A"] & & 1 \arrow[dr, "i_A"] \arrow[dd, dash] & \\
& P_A \arrow[rr, "m_A"'] \arrow[dd, "f_P"] & & S_A\times\Sigma_A \arrow[rr, "p_0"'] & & S_A \arrow[dd, "f_S"] \\
P_B' \arrow[rr, "m_B'"] \arrow[dr] & & 1\times\Sigma_B \arrow[rr, "p_0"] \arrow[dr, "1\times i_B"] & & 1 \arrow[dr, "i_B"] & \\
& P_B \arrow[rr, "m_B"'] & & S_B\times\Sigma_B \arrow[rr, "p_0"'] & & S_B
\end{tikzcd}$$

is a $\mathcal{C}$-pullback whenever the front face is. As $+$ is $\mathcal{C}$-stable the following

$$\begin{tikzcd}
& P_A+P_A' \arrow[rr, "f_P+f_{P'}"] \arrow[d, "m_A+m_A'"'] & & P_B+P_B' \arrow[d, "m_B+m_B'"] & \\
& S_A\times\Sigma_A+1\times\Sigma_A \arrow[dl, "\sim"'] \arrow[dd, "p_0+p_0"] & & S_B\times\Sigma_B+1\times\Sigma_B \arrow[dr, "\sim"] \arrow[dd, "p_0+p_0"'] & \\
(S_A+1)\times\Sigma_A \arrow[dr, "p_0"'] & & & & (S_B+1)\times\Sigma_B \arrow[dl, "p_0"] \\
& S_A+1 \arrow[rr, "f_S+1"'] & & S_B+1 &
\end{tikzcd}$$

is a $\mathcal{C}$-pullback, showing that $G$ is $\exists^{\mathcal{C}}$-stable.

It is easily seen in the proof of lemma 2.5 that $\eta$ and $\epsilon$ are $\exists^{\mathcal{I}}$-maps at each component, which implies that they are $\exists^{\mathcal{C}}$-cartesian as well. $\square$

More generally, one need not have an adjunction of model categories to obtain equivalent process categories:

**Corollary 4.10** *Let* $(\mathbf{X},\mathcal{X}) \underset{G}{\overset{F}{\rightleftarrows}} (\mathbf{Y},\mathcal{Y})$ *with* $\mathcal{X}' \subseteq \mathcal{X}$ *and* $\mathcal{Y}' \subseteq \mathcal{Y}$ *left-factor closed. If* $\phi$ *is an* $\mathcal{X}'$*-transformation of the form* $FG \Longrightarrow 1$ *or* $1 \Longrightarrow FG$*, and* $\psi$ *is a* $\mathcal{Y}'$*-transformation of the form* $GF \Longrightarrow 1$ *or* $1 \Longrightarrow GF$*, then* $\mathrm{Proc}(\mathbf{X},\mathcal{X})$ *is equivalent to* $\mathrm{Proc}(\mathbf{Y},\mathcal{Y})$.

This is the case for the model categories $\mathrm{Tran}(\mathbf{C})$ and $\mathrm{DTran}(\mathbf{C})$. For any cover system $\mathcal{C}$ on $\mathbf{C}$,

**Proposition 4.11** $\mathrm{Proc}(\mathrm{DTran}(\mathbf{C}),\exists^{\mathcal{C}})$ *is equivalent to* $\mathrm{Proc}(\mathrm{Tran}(\mathbf{C}),\exists^{\mathcal{C}})$.

**Proof.** Both $F$ and $G$ are easily seen to be stable, and to preserve $\exists^{\mathcal{C}}$-maps for any $\mathcal{C}$. That the transformations there are $\exists^{\mathcal{I}}$-maps is shown by the following:

$$\begin{tikzcd}
R \arrow[r, no head, "="] \arrow[d, "r"'] \arrow[dr, phantom, "\lrcorner", very near start] & R \arrow[dd, "r"] \\
S\times\Sigma\times S \arrow[d, "\langle a,p_1\rangle"'] & \\
S\times(\Sigma\times S)\times S \arrow[d, "p_0"'] & S\times\Sigma\times S \arrow[d, "p_0"] \\
S\times(\Sigma\times S) \arrow[d, "p_0"'] & S\times\Sigma \arrow[d, "p_0"] \\
S \arrow[r, no head, "="] & S
\end{tikzcd}
\qquad
\begin{tikzcd}
P \arrow[r, no head, "="] \arrow[d, "\langle m,a\rangle"'] \arrow[dr, phantom, "\lrcorner", very near start] & P \arrow[dd, "m"] \\
S\times\Sigma\times S \arrow[d, "a"'] & \\
S\times(\Sigma\times S) \arrow[d, "p_0"'] & S\times\Sigma \arrow[d, "p_0"] \\
S \arrow[r, no head, "="] & S
\end{tikzcd}$$

□

In summary, one constructs the same process category beginning with either of the model categories of section 2. However, the properties of the process category may be more easily seen from one model category than from another: as source machines provide a lextensive model category, it is easily shown that the associated process category has finite biproducts.

## 5 Trace-based Processes

This section develops a general theory of traces within model categories such as DTran($\mathbf{C}$). It is shown to be sufficient to consider the full subcategory of such trace objects when constructing a process category. **SProc** arises in precisely this way: **SProc** is (isomorphic to) the process category constructed upon the traces of DTran($\mathbf{Set}$). Although the traces in this setting are exactly as one would expect, we argue that their construction can be performed in arbitrary locos — i.e. lextensive category with list construction (see Cockett [Coc90]). More precisely, we show how the existence of trace objects and partial inductive datatypes are connected. A direct consequence of this is the fact that in a fibre which fixes the "external labels", the trace objects form a lattice.

### 5.1 Trace Objects

Let $\mathcal{X}$ be a left-factor closed cover system on $\mathbf{X}$,

**Definition 5.1** *An object $Z$ of $\mathbf{X}$ is an $\mathcal{X}$-**trace object** if for all $x : A \longrightarrow B$ of $\mathcal{X}$, any $f : Z \longrightarrow B$ factors uniquely through $x$:*

$$\begin{tikzcd}
A \arrow[dd, "x"'] & \\
& Z \arrow[ul, dashed] \arrow[dl, "f"] \\
B &
\end{tikzcd}$$

Intuitively, an $\mathcal{X}$-map $x : A \longrightarrow B$ indicates when objects $A$ and $B$ are the same — $Z$ is an $\mathcal{X}$-trace object if it cannot distinguish objects which are the same with respect to $\mathcal{X}$.

One can think of a trace object $Z$ with an $\mathcal{X}$-map $x : Z \longrightarrow A$ as a trace object *for* $A$. Fortunately, all trace objects for a specific $A$ are isomorphic — if $x : Z \longrightarrow A$ and $x' : Z' \longrightarrow A$ are $\mathcal{X}$ maps with $Z$ and $Z'$ $\mathcal{X}$-trace objects then there exist unique $h$ and $h'$ such that:

$$\begin{tikzcd}
Z \arrow[dr, "x"'] & & Z' \arrow[ll, dashed, "h'"'] \arrow[dl, "x'"] & \\
& A & & Z \arrow[ll, "x"] \arrow[ul, dashed, "h"']
\end{tikzcd}$$

Thus $h; h' = 1_Z$, and similarly $h'; h = 1_{Z'}$.

We say that $\mathbf{X}$ has *enough $\mathcal{X}$-trace objects* if for each object $A$ of $\mathbf{X}$ one can construct an $\mathcal{X}$-trace object $TA$ and an $\mathcal{X}$-map $\epsilon_A : TA \longrightarrow A$.

**Proposition 5.2** *If* $\mathbf{X}$ *has enough $\mathcal{X}$-trace objects then $T$ is an idempotent comonad in* **Cov** *with counit $\epsilon$.*

**Proof**. The effect of $T$ on maps is given by the projective property:

$$\begin{tikzcd}
TA \arrow[r, dashed, "Tf"] \arrow[d, "\varepsilon_A"'] & TB \arrow[d, "\varepsilon_A"] \\
A \arrow[r, "f"'] & B
\end{tikzcd}$$

from which it is clear that $T$ is a functor and $\epsilon$ is an $\mathcal{X}$-cartesian natural transformation. To see that $T$ is $\mathcal{X}$-stable, note that if the bottom face of the following cube is an $\mathcal{X}$-pullback then, since all sides are $\mathcal{X}$-pullbacks and $\mathcal{X}$ is left-factor closed, the top face is an $\mathcal{X}$-pullback also.

$$\begin{tikzcd}
& TP \arrow[rr, dashed] \arrow[dl, dashed] \arrow[dd, "\varepsilon"] & & TA \arrow[dl, dashed] \arrow[dd, "\varepsilon"] \\
TB \arrow[rr, dashed] \arrow[dd, "\varepsilon"'] & & TC \arrow[dd, "\varepsilon"] & \\
& P \arrow[rr] \arrow[dl] & & A \arrow[dl] \\
B \arrow[rr] & & C &
\end{tikzcd}$$

As $\epsilon$ is an $\mathcal{X}$-map, the comultiplication $\delta$ is the unique left inverse of $\epsilon_T$ given by the projective property: $1_{TA} = \delta_A; \epsilon_{TA}$. As $\mathcal{X}$ is left factor closed, $\delta$ is an $\mathcal{X}$-map and thus has a unique left inverse given by $1_{TTA} = \epsilon_{TA}; \delta_A$. This makes $\delta$ a natural isomorphism.

Regarding the comonad laws, the first is given in the definition of $\delta$ and the other two are easy consequences:

$$\begin{tikzcd}
TA \arrow[d, "\varepsilon"'] & TTA \arrow[l, "\varepsilon"'] \arrow[d, dashed, "T\varepsilon"'] & \\
& & TA \arrow[ul, "\delta"'] \arrow[dl, "="] \\
A & TA \arrow[l, "\varepsilon"] &
\end{tikzcd}
\qquad
\begin{tikzcd}
TA \arrow[r, "\delta"] \arrow[d, "\delta"'] & TTA \arrow[r, "\varepsilon"] \arrow[d, dashed, "T\delta"] & TA \arrow[d, "\delta"] \\
TTA \arrow[r, "\delta"'] & TTTA \arrow[r, "\varepsilon"'] & TTA
\end{tikzcd}$$

□

As $T$ is idempotent the associated category of coalgebras $\mathbf{X}^T$ is just the full coreflective subcategory of trace objects in $\mathbf{X}$ as determined by $T$. As the counit of this coreflection is a left-factor closed cover map at each component, the coreflection induces an equivalence between the respective process categories.

**Corollary 5.3** *If* $\mathbf{X}$ *has enough* $\mathcal{X}$*-trace objects then* $\mathrm{Proc}(\mathbf{X}, \mathcal{X})$ *is equivalent to* $\mathrm{Proc}(\mathbf{X}^T, \mathcal{X}^T)$.

Here $\mathcal{X}^T$ is the restriction of $\mathcal{X}$ to the subcategory $\mathbf{X}^T$.

On the other hand, if $(T, \epsilon, \delta)$ is an idempotent comonad then $T$ is stable and so one obtains a left-factor closed cover system $T^{-1}(\mathcal{I})$ by pulling the isomorphisms back through $T$. In fact,

**Proposition 5.4** *If $(T, \epsilon, \delta)$ is an idempotent comonad then* $\mathbf{X}$ *has enough $T^{-1}(\mathcal{I})$-trace objects given by $\epsilon_A : TA \longrightarrow A$.*

**Proof.** $\epsilon$ is a cover map as $T\epsilon$ is an isomorphism. To see the projective property, suppose $x : C \longrightarrow B \in T^{-1}(\mathcal{I})$ and $f : TA \longrightarrow B$. As $Tx$ and $\epsilon_{TA}$ are isomorphisms, there is a map $\ell : TA \longrightarrow C$ — namely $\epsilon_{TA}^{-1}; Tf; Tx^{-1}; \epsilon_C$ — for which $f = \ell; x$. Furthermore, any map $g : TA \longrightarrow C$ for which $f = g; x$ is equal to $\ell$ as shown below:

$$\begin{tikzcd}
TC \arrow[rr, "\varepsilon"] \arrow[d, "Tx"'] & & C \arrow[d, "x"'] & \\
TB \arrow[rr, "\varepsilon"'] & & B & \\
& TTA \arrow[lu, "Tf"] \arrow[luu, "Tg"'] \arrow[rr, "\varepsilon"'] & & TA \arrow[lu, "f"'] \arrow[luu, dashed, "g"']
\end{tikzcd}$$

□

## 5.2 Partial Inductive Datatypes

We now show how he existence of trace objects is related to the existence of partial inductive datatypes. Let $\mathbf{C}$ be a lextensive category with list construction given by functor $L$ and transformations *nil* and *cons* — such a category is called a *locos* (see Cockett [Coc90]). In the category of partial maps on $\mathbf{C}$ one has natural maps $pnil : B \rightharpoonup LA \times B$ and $pcons : A \times (LA \times B) \rightharpoonup LA \times B$ for all $A$ and $B$:

$$\begin{tikzcd}
& 1{\times}B \arrow[dl, "\sim"'] \arrow[dr, "nil{\times}1"] & & & (A{\times}LA){\times}B \arrow[dl, "\sim"'] \arrow[dr, "cons{\times}1"] & \\
B & & LA{\times}B & A{\times}(LA{\times}B) & & LA{\times}B
\end{tikzcd}$$

**Definition 5.5** $\mathbf{C}$ *has partial list construction if for all partial maps $f : B \rightharpoonup C$ and $g : A \times C \rightharpoonup C$ of* $\mathrm{Par}(\mathbf{C})$ *there exists $r : LA \times B \rightharpoonup C$ such that*

$$\begin{tikzcd}
A{\times}(LA{\times}B) \arrow[r, "pcons"] \arrow[d, "1{\times}r"'] & LA{\times}B \arrow[d, "r"'] & B \arrow[l, "pnil"'] \arrow[dl, "f"] \\
A{\times}C \arrow[r, "g"'] & C &
\end{tikzcd}$$

*commutes in* $\mathrm{Par}(\mathbf{C})$. *Furthermore, for all $q : LA \times B \rightharpoonup C$ such that*

$$\begin{tikzcd}
A{\times}(LA{\times}B) \arrow[r, "pcons"] \arrow[d, "1{\times}q"'] \arrow[dr, phantom, "\geq"] & LA{\times}B \arrow[d, "q"'] & B \arrow[l, "pnil"'] \arrow[dl, "f"] \\
A{\times}C \arrow[r, "g"'] & C &
\end{tikzcd}$$

*there exists a unique $\delta : q \Longrightarrow r$ which equates the following 2-cells:*

$$\begin{tikzcd}
 & A\times(LA\times B) \arrow[r, "pcons"] \arrow[dl, phantom] \arrow[d, "1\times q"] \arrow[d, phantom, shift right=20, "\geq"] \arrow[dr, phantom, "\geq"] & LA\times B \arrow[d, "q"] \\
 & A\times C \arrow[r, "g"'] & C
\end{tikzcd}
\qquad
\begin{tikzcd}
A\times(LA\times B) \arrow[r, "pcons"] \arrow[d, "1\times r"'] \arrow[dr, phantom, "="] & LA\times B \arrow[d, "r"', bend right] \arrow[d, "q"] \arrow[d, phantom, "\geq"] \\
A\times C \arrow[r, "g"'] & C
\end{tikzcd}$$

The question now is: when does a category have partial list construction? Certainly **Set** does. However, we conjecture that partial list construction already exists in a locos.

**Conjecture 5.6** *Locoi have partial list construction.*

## 5.3 Trace Machines

We now turn to the construction of trace objects for deterministic transition systems. It is important to note that DTran($\mathbf{C}$) is fibred over $\mathbf{C}$. The fibration functor $\delta$ extracts the alphabet of a transition system. $\delta$ is stable as pullbacks are given pointwise in the overlying category, and thus we can modify any cover system on DTran($\mathbf{C}$) to lie within the fibration:

**Definition 5.7** *Let $\exists^{\mathcal{I}}_F$ be the cover system $\exists^{\mathcal{I}} \cap \delta^{-1}(\mathcal{I})$.*

Recall that $\exists^{\mathcal{I}}$ is the (left-factor closed) class of maps which are cartesian for $m; p_0$. We will regard objects $A$ and $B$ of DTran($\mathbf{C}$) as having the same labels when related by an $\exists^{\mathcal{I}}_F$-map.

For $f : \Sigma \longrightarrow \Sigma'$, the substitution functor $f^* : \mathrm{DTran}(\mathbf{C})_{\Sigma'} \longrightarrow \mathrm{DTran}(\mathbf{C})_{\Sigma}$ has the following effect:

$$\begin{tikzcd}
 & P_{f^*A} \arrow[rr] \arrow[dl, "m_{f^*A}"'] \arrow[dd, dashed] & & P_A \arrow[r, "\alpha_A"] \arrow[dl, "m_A"] \arrow[dd] & S_A \arrow[dd] \\
\Sigma\times S_A \arrow[rr, "f\times 1"'] \arrow[dd] & & \Sigma'\times S_A \arrow[dd] & & \\
 & P_{f^*A} \arrow[rr] \arrow[dl, "m_{f^*B}"'] & & P_B \arrow[r, "\alpha_B"] \arrow[dl, "m_B"] & S_B \\
\Sigma\times S_B \arrow[rr, "f\times 1"'] & & \Sigma'\times S_B & &
\end{tikzcd}$$

it preserves pullbacks and also preserves $\exists^{\mathcal{I}}_F$-maps. So $\delta$ is in fact a **Cov**-fibration $(\mathrm{DTran}(\mathbf{C}), \exists^{\mathcal{I}}_F) \longrightarrow (\mathbf{C}, \mathcal{I})$.

Let **C** be a locos which has partial list construction. Then for any object $A$ of DTran(**C**), one has the following diagram in **C**:

$$\begin{tikzcd}
\Sigma\times L\Sigma \arrow[r, dash, "="] & \Sigma\times L\Sigma \arrow[r, "rcons"] & L\Sigma & 1 \arrow[l, "rnil"'] \\
\Sigma\times S_{TA} \arrow[u, tail] \arrow[d] & P_{TA} \arrow[l, tail, "m_{TA}"'] \arrow[u, tail] \arrow[d] \arrow[r, "\alpha_{TA}"] \arrow[ur, phantom, "\urcorner", very near start] \arrow[dl, phantom, "\llcorner", very near start] & S_{TA} \arrow[u, tail] \arrow[d] & 1 \arrow[l, "i_{TA}"'] \arrow[u, dash, "="'] \arrow[dl, "i_A"] \arrow[ul, phantom, "\ulcorner", very near start] \\
\Sigma\times S_A & P_A \arrow[l, tail, "m_A"] \arrow[r, "\alpha_A"'] & S_A &
\end{tikzcd}$$

We take the *trace machine* of $A$ to be $TA = (m_{TA}, \alpha_{TA})$, and will refer to the maps $TA \longrightarrow A$ and $TA \longrightarrow L\Sigma$ as $\epsilon_A$ and $\omega_A$ respectively.

**Lemma 5.8** *If $c : A \longrightarrow B \in \exists_F^{\mathcal{I}}$ then any trace machine of $A$ is a trace machine of $B$.*

**Proof**. A $\exists_F^{\mathcal{I}}$-map has the square associated with $m$ a pullback. Thus,

$$\begin{tikzcd}
\Sigma\times L\Sigma \arrow[r, dash, "="] & \Sigma\times L\Sigma \arrow[r, "rcons"] & L\Sigma & 1 \arrow[l, "rnil"'] \\
\Sigma\times S_{TA} \arrow[u, tail] \arrow[d] & P_{TA} \arrow[l, tail, "m_{TA}"] \arrow[u, tail] \arrow[d] \arrow[r, "\alpha_{TA}"'] \arrow[ur, phantom, "\urcorner", very near start] \arrow[dl, phantom, "\llcorner", very near start] & S_{TA} \arrow[u, tail] \arrow[d] & 1 \arrow[l, "i_{TA}"] \arrow[u, dash, "="'] \arrow[dl, "i_A"] \arrow[ddl, "i_B"] \arrow[ul, phantom, "\ulcorner", very near start] \\
\Sigma\times S_A \arrow[d, "1\times c_S"'] & P_A \arrow[l, tail, "m_A"] \arrow[r, "\alpha_A"'] \arrow[d, "c_P"] \arrow[dl, phantom, "\llcorner", very near start] & S_A \arrow[d, "c_S"] & \\
\Sigma\times S_B & P_B \arrow[l, tail, "m_B"] \arrow[r, "\alpha_B"'] & S_B &
\end{tikzcd}$$

$\square$

We now show that this construction above actually yields trace objects as described in the previous section:

**Proposition 5.9** DTran(**C**) *has enough $\exists_F^{\mathcal{I}}$-trace objects given by the preceeding construction.*

**Proof**. Let $c : A \longrightarrow B \in \exists_F^{\mathcal{I}}$, and $f : TZ \longrightarrow B$. We will show that there is a unique $h : TZ \longrightarrow A$ such that $f = h; c$ in two steps.

First, suppose $\Sigma_Z = \Sigma_A$. Then $(\omega_Z, f)$ determines a square with a two-cell in Par($\mathbf{C}$):

$$\begin{tikzcd}
\Sigma\times L\Sigma \arrow[rr, no head, "="] & & \Sigma\times L\Sigma \arrow[r, "rcons"] & L\Sigma & 1 \arrow[l, "rnil"'] \arrow[d, equal] \\
\Sigma\times S_{TZ} \arrow[u] \arrow[ddd, "1\times f_S"'] & & P_{TZ} \arrow[ll, "m_{TZ}"'] \arrow[u] \arrow[r, "\alpha_{TZ}"] \arrow[ddd, "f_P"] \arrow[ur, phantom, "\lrcorner"] & S_{TZ} \arrow[u] \arrow[ddd, "f_S"] & 1 \arrow[l, "i_{TZ}"'] \arrow[dddl, "i_A"] \arrow[ul, phantom, "\llcorner"] \\
& \cdot \arrow[ul] \arrow[ur, dashed] \arrow[ddr] & & & \\
& & & & \\
\Sigma\times S_A & & P_A \arrow[ll, "m_A"] \arrow[r, "\alpha_A"'] & S_A &
\end{tikzcd}$$

By the universal property of partial lists, there exists a unique $h' : TZ \longrightarrow TA$ such that

$$\begin{tikzcd}
& & L\Sigma & \\
TA \arrow[urr, "\omega_A"] \arrow[dr, "\varepsilon_A"'] & & & TZ \arrow[ul, "\omega_Z"'] \arrow[lll, dashed, "l"] \arrow[dl, "f"] \\
& A \arrow[r, "c"'] & B &
\end{tikzcd}$$

commutes in DTran($\mathbf{C}$). Let $h = h'; \epsilon_A$, and suppose that $k$ also satisfies $f = k; c$. Then both $h$ and $k$ determine squares with two-cells in Par($\mathbf{C}$) and so there exist unique $h'$ and $k'$ such that

$$\begin{tikzcd}
& L\Sigma & \\
TZ \arrow[ur, "\omega_Z"] \arrow[r, dashed, "h'"] \arrow[dr, "h"'] & TA \arrow[u, "\omega_A"] \arrow[d, "\varepsilon_A"] & TZ \arrow[ul, "\omega_Z"'] \arrow[l, dashed, "k'"'] \arrow[dl, "k"] \\
& A &
\end{tikzcd}$$

commutes in DTran($\mathbf{C}$). But then $h' = k'$ as $\omega_Z$ is monic, and thus $h = k$.

If $\Sigma_Z \neq \Sigma_A$ then we can use the substitution functor to obtain an $\exists^{\mathcal{I}}_F$-map $f^*c$ over $\Sigma_Z$. As $\lambda_B$ is cartesian there exists a unique $f'$ such that $f = f'; \lambda_B$ and thus (by the preceeding argument) a unique $\ell$ such that $f' = \ell; f^*c$:

$$\begin{tikzcd}
& f^*A \arrow[dl, "\lambda_A"'] \arrow[d, "f^*c"] & \\
A \arrow[d, "c"'] & f^*B \arrow[dl, "\lambda_B"'] & \\
B & & TZ \arrow[ll, "f"] \arrow[ul, dashed, "f'"'] \arrow[uul, dashed, "h'"']
\end{tikzcd}$$

Then $\ell; \lambda_A$ serves as a lifting of $f$ along $c$, and uniqueness is given by the cartesian nature of $\lambda_A$. □

These trace objects of DTran(**C**) can also be characterized directly:

**Definition 5.10** Trace(**C**) *is the category of models of the following (higher-order) sketch:*

$$\begin{tikzcd}
 & \Sigma\times L\Sigma \arrow[r, "cons"] & L\Sigma & 1 \arrow[l, "nil"'] \arrow[dl] \\
\Sigma\times S \arrow[ur, tail, "1\times s"] & S_1 \arrow[l, "q"] \arrow[u, tail] \arrow[r] \arrow[ur, phantom, "\lrcorner", very near start] & S \arrow[u, tail, "s"] &
\end{tikzcd}$$

In **Set**, the models of this sketch are exactly the object of **SProc** — i.e. nonempty, prefix-closed trace specification. In fact,

**Proposition 5.11** **SProc** *is isomorphic to* Proc(Trace(**Set**), $\exists^{\mathcal{R}}$)*.*

Where $\exists^{\mathcal{R}}$ in Trace(**C**) is the class of maps which are $\mathcal{R}$-cartesian for $q; p_1$. The details concerning this result can be found in Cockett and Spooner [CS94]. As $\exists^{\mathcal{I}}_{F}$ is contained in $\exists^{\mathcal{R}}$, we have the following equivalent characterizations of **SProc**:

**Theorem 5.12** *The process categories constructed upon the following model categories are equivalent to* **SProc***:*

*i) source machines,* Src(**Set**)*;*

*ii) deterministic transition systems,* DTran(**Set**)*;*

*iii) nondeterministic transition systems,* Tran(**Set**)

## 6 Conclusion

We have presented a general construction of process categories which yields **SProc** in a variety of instances. One might reasonably ask whether Abramsky's category **ASProc** of *asynchronous processes* can be obtained using these techniques. The answer is yes, however the significant additional machinery is required to handle asynchrony.

Not suprisingly, the introduction of asynchrony into a synchronous setting involves monads of delay — these correspond to the $\delta$ and $\Delta$ operators of SCCS. The composite of these monads, say $D$, occurs at the level of the model category $\mathrm{DTran}(\mathbf{C})$ (or any of the other model categories), and is special in that the associated Kleisli category has finite limits. This allows the construction of a compact-closed category of asynchonous processes up-to strong equivalence. [3]

The method of obtaining weak equivalence is analogous to the means by which one would compute weak bisimulation in terms of strong bisimulation: construct "macro" transition systems whose actions are the sequences of non-idle actions of the originals; then check strong bisimilarity of these. The construction can be performed in any category with the partial list datatype, and yields a cover-stable functor $F : (\mathrm{DTran}(\mathbf{C})_D, \mathcal{I}) \longrightarrow (\mathrm{DTran}(\mathbf{C}), \exists^{\mathcal{R}})$. The cover system $F^{-1}(\exists^{\mathcal{R}})$ then provides weak equivalence of asynchonous processes.

The techniques for asynchrony play an important role in constructing categories of Games and Strategies (see Abramsky and Jagadeesen [AJ92]). One begins with a category of games and homomorphisms, but limits the maps of the model category used to construct spans to come from a pair of cover systems — say $\mathcal{L}_0$ for the left legs and $\mathcal{L}_1$ for the right legs. This has the effect that spans correspond to strategies in the sense of [AJ92].

The category described [AJ92] is implicitly asynchronous. A reconstruction in our framework naturally makes a progression from synchrony to asynchrony. Interestingly, the category of synchronous strategies is weakly distributive (see Cockett and Seely [CS92]): The introduction of asynchrony provides the complementation maps necessary to obtain $\star$-autonomy.

Like the construction of **SProc**, one can construct games and strategies beginning with state-based model categories. The traces used in Abramsky and Jagadeesen's formulation arise also as trace objects for an appropriate partial inductive datatype.

[3] The free functor $F$ of the Kleisli construction induces a cover system $F(\mathcal{C})$ on $\mathbf{C}_D$, which lifts the synchronous equivalence unchanged into the asynchronous setting.

## References

[Abr93] Abramsky, S., *Interaction Categories (Extended Abstract).* Theory and Formal Methods Workshop, Springer Verlag, 1993.

[AJ92] Abramsky, S., and R. Jagadeesan, *Games and Full Completeness for Multiplicative Linear Logic.* Imperial College Technical Report DoC 92/24, 1992.

[AJM93] Abramsky, S., Jagadeesan, R., and Malacaria, P, *Games and full abstraction for PCF: preliminary annnouncement.* Unpublished.

[Acz88] Aczel, P., *Non-well-founded sets.* CLSI Lecture Notes 14, Center for the Study of Language and Information, 1988.

[BW90] Barr, M. and C. Wells, *Category Theory for Computing Science.* Prentice Hall, 1990.

[Ben67] Bénabou J., *Introduction to bicategories.* Lecture Notes in Mathematics 47, Springer Verlag, 1967.

[CLW92] Carboni, A., S. Lack and R.F.C. Walters, *Introduction to extensive and distributive categories.* Manuscript, March 1992.

[Coc90] Cockett, J.R.B., *List-arithmetic open categories: Locoi.* Journal of Pure and Applied Algebra, 66:1–29, 1990.

[Coc93] Cockett, J.R.B., *Introduction to distributive categories.* Mathematical Structures in Computer Science, 3:277–307, 1993.

[CS92] Cockett J.R.B. and R.A.G. Seely, *Weakly distributive categories.* London Mathematical Society Lecture Notes, 177:45–65, Cambridge University Press, 1992.

[CS94] Cockett J.R.B. and D.A. Spooner, *SProc Categorically.* Proceedings of CONCUR '94, Springer Verlag, 1994.

[JNW93] Joyal, A., M. Nielsen and G. Winskel, *Bisimulation and open maps.* Proceedings of the Eighth Symposium on Logic in Computer Science, IEEE, 1993.

[Mil83] Milner, R., *Calculi for synchrony and asynchrony.* Theoretical Computer Science, 25:267–310, 1983.

[Win87] Winskel, G., *A Compositional Proof System on a Category of Labelled Transition Systems.* Information and Computation 87:2–57.