SProc Categorically

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

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

Abstract. We provide a systematic reconstruction of Abramsky's category **SProc** of synchronous processes [Abr93]: **SProc** is isomorphic to a span category on a category of traces. The significance of the work is twofold: It shows that the original presentation of **SProc** in mixed formulations is unnecessary — a simple categorical description exists. Furthermore, the techniques employed in the reconstruction suggest a general method of obtaining process categories with structure similar to **SProc**. In particular, the method of obtaining bisimulation equivalence in our setting, which represents an extension of the work of Joyal, Nielsen and Winskel [JNW93], has natural application in many settings.

1 Introduction

In [Abr93], Abramsky proposed a new paradigm for the semantics of computation, *interaction categories*, where the following substitutions are made:

Denotational semantics	Categories	Interaction categories
Domains	objects	Interface specifications
Continuous functions	maps	Communicating Processes
Functional composition	composition	Interaction

This paradigm shift, away from the classical approach to modeling computation, provided by domain theory, had an immediate effect. The problem of finding fully abstract models for programming languages — a problem that had dogged the semantics community for two decades — suddenly became tractable [AJM93]: game theoretic interaction categories could be constructed in a relatively straightforward manner to capture the key operational aspects of computation. It was a construction that had eluded the classical methods and was brought into reach by the simple trick of passing to the much richer semantic settings provided by interaction categories.

In effect, Abramsky had used ideas from concurrency to solve a problem in programming semantics. Furthermore, in providing a "process" based semantics for programming, he raised the possibility that these two disparate aspects of computing could be unified. Such a unified view, he realized, would facilitate the transfer of ideas between the two areas and enhance their mutual development.

Up to that time the concurrency community had regarded processes to be structured objects whose morphisms were simulations. This view made inaccessible a fundamental and very basic verification technique of modern programming: type checking. In the interaction category paradigm, a processes is a map between interface specifications and type checking is the demonstration that a process respects the constraints imposed by its interfaces.

To realize the benefits of a having type system for concurrency, Abramsky [Abr93] introduced the interaction category of *synchronous processes*, **SProc**, and its sister the interaction category of *asynchronous processes*, **ASProc**. In describing **SProc** Abramsky employed a wide range of methodologies: Aczel's theory of non-well-founded sets, linear type theory, category theory, and concurrency theory. While this helped to focus the ideas of a wide community, it did not provide a clear understanding of the required categorical foundation for the construction. In this paper we present a categorical reconstruction of **SProc** and, more importantly, introduce the techniques employed.

The main conceptual tool in this reconstruction, that of systems of *open maps*, which we call *cover systems*, is not new. The recognition of their relationship to bisimulation, however, is recent (see Joyal, Nielsen, and Winskel [JNW93]). A cover system on a category is a compositional class of maps, including all isomorphisms, which is closed to pulling back along arbitrary maps. A span of cover maps from one object to another can be viewed as a witness to a bisimulation. This view provides a unified framework for describing bisimulation in a number of categorical models of concurrency, among which are labelled transition systems, labelled event structures, and transition systems with independence.

Cover systems can also be used to induce a congruence on span categories. From a category with a cover system $(\mathbf{X}, \mathcal{X})$, we construct the quotiented span category $Proc(\mathbf{X}, \mathcal{X})$, which we call the category of processes: this construction is 2-functorial. The main results of the paper concern the manner in which **SProc** arises as such a process category. We exhibit, in fact, three different (but related) categories and their cover systems for which **SProc** is the resulting process category.

The relationship of interaction categories to span categories is already implicit in Abramsky's work when he refers to the bicategories of Carboni and Walters [CW87]. We maintain that the explicit reconstruction of this aspect of **SProc** is of considerable interest as it indicates much more clearly the categorical foundation required.

2 SProc Recounted

Abramsky presents **SProc** as an example of an interaction category [Abr93]. The objects of this category are safety specifications for concurrent systems, given as sets of permissible traces, maps are given by processes (i.e. transition systems modulo strong bisimulation), and composition is given by *interaction*.

The category is presented in the style of a linear type theory emphasizing the similarity to other examples of interaction categories (e.g. Games and Strategies [AJ92]). Linear negation $\perp$ and multiplicative conjunction $\otimes$ are defined on the objects (types) of **SProc**, inducing multiplicative disjunction $A \oplus B \overset{\text{def}}{=} (A^\perp \otimes B^\perp)^\perp$ and linear implication $A \multimap B \overset{\text{def}}{=} A^\perp \oplus B$. A satisfaction relation $\models$ is introduced between processes and types, allowing a morphism $p : A \longrightarrow B$ to be defined by a process p for which $p \models A \multimap B$.

As we are not concerned here with the linear type structure of **SProc**, we take a more direct description of the category. We will, however, follow Abramsky in choosing synchronization trees, in terms of non-well-founded sets, as canonical representatives of processes. Thus, the synchronization trees over a label set Σ are the largest solution to the (non-well-founded) set equation:

$$ST_\Sigma = \mathcal{P}(\Sigma \times ST_\Sigma).$$

We will use the process notation $p \xrightarrow{a} q$ to represent $(a, q) \in p$.

For any set X and function $f : X \longrightarrow Y$, let $(X^*, \epsilon, \cdot)$ denote the free monoid on X and $f^* : X^* \longrightarrow Y^*$ the unique homomorphic extension of f. Given any process p on the label set Σ we can construct a macro process with labels in Σ^* whose transitions correspond to sequences of transitions in p:

$$M(p) = \{(\epsilon, p)\} \cup \{(x \cdot s, q) \mid p \xrightarrow{x} p' \wedge (s, q) \in M(p')\}.$$

We shall write $p \overset{s}{\Longrightarrow} q$ to mean $(s, q) \in M(p)$. The finite sequences of visible actions (or traces) of a synchronization tree are:

$$traces(p) = \{s \in \Sigma^* \mid \exists q.\, p \overset{s}{\Longrightarrow} q\}.$$

SProc can now be defined as follows:

- The objects A are pairs
$$(\Sigma_A,\, S_A \subseteq \Sigma_A^*),$$
where S_A is a non-empty and prefix closed set of traces over alphabet Σ_A.
- The morphisms $p : A \longrightarrow B$ are synchronization trees $p \in ST_{\Sigma_A \times \Sigma_B}$ such that:
$$fst^*(traces(p)) \subseteq S_A \;\wedge\; snd^*(traces(p)) \subseteq S_B.$$
- Composition is given by a combination of synchronous product and restriction, following SCCS [Mil83]:
$$p; q = \{((a, c),\, p'; q') \mid \exists b.\, p \xrightarrow{(a,b)} p' \;\wedge\; q \xrightarrow{(b,c)} q'\}.$$
- Identities are synchronous buffers, or wires, through which information flows unchanged instantaneously:
$$1_A = \{((a, a),\, 1_{A\backslash a}) \mid a \in S_A\}.$$

Where $A\backslash s$ is given by

$$A\backslash s = (\Sigma_A,\ \{t \mid s \cdot t \in S_A\}), \quad \text{if } s \in S_A.$$

That is the permitted behaviors after (performance of) the trace s.

Although parallel composition combined with restriction is not an associative operation in process algebra, the typed framework provided by **SProc** is enough to make it so. **SProc** provides a model of full second-order linear logic (it is a compact closed category with additional structure); it also supports *delay* monads, which allow asynchrony to be built upon synchrony following Milner[Mil83].

3 A Category of Traces

The first step in reconstructing **SProc** is to construct a category whose objects are trace specifications and whose maps are simply homomorphisms. We show how this can be done for lextensive categories with list arithmetic (see Cockett [Coc90]). A lextensive category has finite limits, finite coproducts, and the property that in the following diagram

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

(1) and (2) are pullbacks exactly when the top row is a coproduct (see Carboni, Lack and Walters [CLW92], or Cockett [Coc93]). It is important to realize that, foundationally, this is all the structure that is required: although we shall need some extra ingredients to carry through the subsequent construction of processes, the remaining ingredients are a matter of choice rather than axiomatization.

Let $\mathbf{C}$ be a lextensive category with a list constructor $((\_)^*, \epsilon, cons)$. A subobject $s : S \longrightarrow \Sigma^*$ in $\mathbf{C}$ is a *trace specification* provided maps q and e exists such that

$$\begin{tikzcd}
 & Q \arrow[r] \arrow[d] \arrow[dl, dashed, "q"'] \arrow[dr, phantom, "\lrcorner", very near start] & S \arrow[d, "s"] & 1 \arrow[l, dashed, "e"'] \arrow[dl, "\varepsilon"] \\
S\times\Sigma \arrow[r, "s\times id"'] & \Sigma^*\times\Sigma \arrow[r, "cons"'] & \Sigma^* &
\end{tikzcd} \tag{1}$$

commutes. The object Σ^* is understood as the finite sequences over "alphabet" Σ — this is described in [Coc90], where it is also shown that the functor $(\_)^*$ preserves pullbacks. The object Q corresponds to the non-ϵ sequences of S, and the existence of q means that S is prefix-closed; the existence of e means S contains ϵ.

Definition 3.1 *$Trace(\boldsymbol{C})$ is the category whose objects are trace specifications and whose maps $f : A \longrightarrow B$ are pairs (f_Σ, f_S) for which*

$$\begin{tikzcd}
S_A \arrow[r, dashed, "f_S"] \arrow[d] & S_B \arrow[d] \\
\Sigma_A{}^* \arrow[r, "f_\Sigma{}^*"'] & \Sigma_B{}^*
\end{tikzcd}$$

commutes.

$Trace(\mathbf{C})$ may be viewed as the category of models of a higher-order sketch (see Power and Wells [PW92]) involving the list functor in $\mathbf{C}$. It is perhaps unexpected to find that:

Proposition 3.2 *$Trace(\boldsymbol{C})$ is a lextensive category.*

Proof. Initial and final objects are given by $\epsilon_0 : 1 \longrightarrow 0^*$ and $id_1 : 1^* \longrightarrow 1^*$, respectively, where 0 is initial and 1 is final in $\mathbf{C}$.

The pullback of $f : A \longrightarrow B$ and $g : C \longrightarrow B$ is given componentwise in $\mathbf{C}$, using the fact that $(\_)^*$ preserves pullbacks:

$$\begin{tikzcd}
S_P \arrow[rr] \arrow[dr] \arrow[dd, dashed, "s_P"'] & & S_C \arrow[dr, "g_S"] \arrow[dd] & \\
& S_A \arrow[rr, "f_S"] \arrow[dd] & & S_B \arrow[dd] \\
\Sigma_P{}^* \arrow[rr] \arrow[dr] & & \Sigma_C{}^* \arrow[dr, "g_\Sigma{}^*"] & \\
& \Sigma_A{}^* \arrow[rr, "f_\Sigma{}^*"'] & & \Sigma_B{}^*
\end{tikzcd}$$

For coproducts it is worth shifting attention to the arrow $q : Q \longrightarrow S \times \Sigma$ by which a trace can be equivalently specified. The coproduct is then given by:

$$\begin{tikzcd}
Q_A \arrow[r, "b_0"] \arrow[d, "q;s\times id;cons"'] & Q_A+Q_B \arrow[d, dashed] & Q_B \arrow[l, "b_1"'] \arrow[d, "q;s\times id;cons"] \\
\Sigma_A{}^* \arrow[r, "b_0{}^*"'] & (\Sigma_A+\Sigma_B)^* & \Sigma_B{}^* \arrow[l, "b_1{}^*"]
\end{tikzcd}$$

from which the fact that it is extensive follows immediately. □

It is worth noting that the definition given by Abramsky for $A \otimes B$ in **SProc** is precisely the product in $Trace(\mathbf{Set})$. Clearly, the objects of **SProc** are the objects of this category. That the maps in Abramsky's formulation of **SProc** come from another source is unnecessary, as we shall see.

4 Cover Systems

Let **X** be a category with pullbacks. A collection $\mathcal{X}$ of the maps of **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 $x \in \mathcal{X}$ and the following is a pullback, then $y \in \mathcal{X}$:

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

In any category the class of isomorphisms $\mathcal{I}$ and the class of retractions $\mathcal{R}$ are cover systems which are preserved by all functors. In a regular category, the class $\mathcal{E}$ of regular epimorphisms is a cover system.

A commuting square $h; f = k; g$ in a category with a cover system $\mathcal{X}$ is an $\mathcal{X}$*-pullback* provided the induced map x to the inscribed pullback is in $\mathcal{X}$:

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

In the category of models of a sketch, one obtains a cover system by considering those model morphisms for which a chosen naturality square is an $\mathcal{X}$-pullback.

The latter technique will be motivated through a series of related examples. The first demonstrates how one obtains the standard notion of strong bisimulation on labelled transition systems via spans of cover maps. The subsequent examples of deterministic transition systems and trace specifications will be used later in the reconstruction of **SProc**.

4.1 Transition Systems

Given a lextensive category **C**, we define the category $Tran(\mathbf{C})$ as the category of models in **C** of the following sketch:

$$\begin{tikzcd}
R \arrow[d, tail, "r"'] & 1 \arrow[d, "s_0"] \\
S \times \Sigma \times S & S
\end{tikzcd}$$

$Tran(\mathbf{Set})$ is the category of labelled transition systems presented by Winskel [Win87]: the objects are structures

$$(\Sigma,\ S,\ s_0 \in S,\ R \subseteq S \times \Sigma \times S)$$

and morphisms $A \longrightarrow B$ are pairs of functions

$$(\sigma : S_A \longrightarrow S_B,\ \lambda : \Sigma_A \longrightarrow \Sigma_B)$$

which preserve transitions and initial states.

If $\mathcal{C}$ is a cover system on $\mathbf{C}$ then the maps $f : A \longrightarrow B$ of $Tran(\mathbf{C})$ for which

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

is a $\mathcal{C}$-pullback form a cover system we will call $\exists^{(Tran(\mathbf{C}),\mathcal{C})}$, or simply $\exists^{\mathcal{C}}$ if $Tran(\mathbf{C})$ is understood.

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

In [JNW93], the cover system $\exists^{\mathcal{E}}$ is expressed via a path lifting property, and is shown to yeild bisimulation equivalence on the objects of $Tran(\mathbf{Set})$: taking the fibre category $Tran(\mathbf{Set})_{\Sigma}$ over any label set Σ and the restriction $\exists^{\mathcal{E}}_{\Sigma}$ of $\exists^{\mathcal{E}}$ to the maps of the fibre, transition systems A and B are strongly bisimilar if and only if there exists a span of $\exists^{\mathcal{E}}_{\Sigma}$ maps with endpoints A and B:

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

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 the relation R.

4.2 Deterministic Transition Systems

A closely related, yet simpler, example is given by deterministic labelled transition systems. Let $DTran(\mathbf{C})$ be the category of models in $\mathbf{C}$ of the following sketch:

$$\begin{tikzcd}
 & P \arrow[dl, tail, "m"'] \arrow[dr, "\alpha"] & & \\
S\times\Sigma & & S & 1 \arrow[l, "s_0"']
\end{tikzcd}$$

m is a permission relation, indicating the actions available at each state, and α determines the state change upon action. Of course $DTran(\mathbf{C})$ is just the full subcategory of deterministic machines in $Tran(\mathbf{C})$.

A cover system $\exists^{\mathcal{C}}$ is similarly obtained on $DTran(\mathbf{C})$ by considering those maps for which the square associated with $m; p_0$ is a $\mathcal{C}$-pullback. Considering a fibre $DTran(\mathbf{C})_{\Sigma}$, we find that $\exists^{\mathcal{R}}_{\Sigma}$ and $\exists^{\mathcal{I}}_{\Sigma}$ coincide, which is not unexpected as bisimulation and language equivalence coincide for such deterministic objects.

4.3 Trace Specifications

Recall the map $q : Q \longrightarrow S \times \Sigma$ of diagram 1 which represents the non-ϵ traces of an object of $Trace(\mathbf{C})$. Corresponding to each $f : A \longrightarrow B$ is an induced map $f_Q : Q_A \longrightarrow Q_B$, giving the following naturality square in $\mathbf{C}$:

$$\begin{tikzcd}
Q_A \arrow[r, "q_A"] \arrow[d, "f_Q"'] & S_A \times \Sigma_A \arrow[r, "p_0"] & S_A \arrow[d, "f_S"] \\
Q_B \arrow[r, "q_B"'] & S_B \times \Sigma_B \arrow[r, "p_0"'] & S_B
\end{tikzcd}$$

If $\mathcal{C}$ is a cover system on $\mathbf{C}$ then $\exists^{\mathcal{C}}$ is the cover system on $Trace(\mathbf{C})$ consisting of those $f : A \longrightarrow B$ for which this square is a $\mathcal{C}$-pullback.

In $Trace(\mathbf{Set})$, a map $f : A \longrightarrow B$ of $\exists^{\mathcal{I}}$ has the property that each extension of a trace $f(t)$ in B is the image of a unique extension of trace t in A, while $\exists^{\mathcal{E}}$ has the more generous property that for each extension of $f(t)$ there exists a corresponding extension of t. We will see that $\exists^{\mathcal{E}}$ yields the notion of bisimulation required to reconstruct **SProc**

It is interesting to note that when one considers only a fibre $Trace(\mathbf{C})_{\Sigma}$, both $\exists^{\mathcal{R}}_{\Sigma}$ and $\exists^{\mathcal{I}}_{\Sigma}$ coincide with the isomorphisms $\mathcal{I}_{\Sigma}$ — a consequence of the properties of trace specifications. Later we will see that nondeterminism is recaptured in such deterministic settings through the construction of "processes".

5 Processes as Spans

A fruitful way of regarding a process on a category such as $Trace(\mathbf{C})$ is as a *span*, that is a pair of maps with a common domain:

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

One should regard the endpoints A and B as interface specifications, and thus the legs f and g deterimine how actions in the apex P determine (visible) actions simultaneously at each interface.

From any category $\mathbf{X}$ with pullbacks one can form the bicategory of spans in $\mathbf{X}$: the objects are those of $\mathbf{X}$; the 1-cells $A \longrightarrow B$ are spans in $\mathbf{X}$, which we will write as $P_{(f,g)}$; and 2-cells $P_{(f,g)} \longrightarrow P'_{(f',g')}$ are given by maps $h : P \longrightarrow P'$ 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. $P_{(f,g)}; Q_{(h,k)}$ is $R_{(p;f,\, q;k)}$, where:

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

A cover system $\mathcal{X}$ on $\mathbf{X}$ induces a congruence $\sim_{\mathcal{X}}$ (or $\mathcal{X}$-bisimulation) on spans, taking $P_{(f,g)}$ and $Q_{(h,k)}$ as equal when there exist maps x and y of $\mathcal{X}$ 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 $\mathbf{X}$. Quotienting 1-cells by $\sim_{\mathcal{X}}$ gives a category $Proc(\mathbf{X}, \mathcal{X})$ of processes up-to the specified equivalence.

Certainly the simplest example of this construction is a category of relations: for $\mathbf{E}$ a regular category, $Proc(\mathbf{E}, \mathcal{E})$ is usually written $Rel(\mathbf{E})$.

Just as relations constructed from the functions of $\mathbf{Set}$, the processes constructed from the traces of $Trace(\mathbf{Set})$ may exhibit nondeterminism: the CCS agents $a.(b.0 + c.0)$ and $a.b.0 + a.c.0$ can be seen as maps $1 \longrightarrow A$ of $\mathbf{SProc}$, with $A = \{\epsilon, a, a \cdot b, a \cdot c\} \subseteq \{a, b, c\}^*$. These are represented respectively by the following spans:

$$\begin{tikzcd}
 & A \arrow[dl, "!"'] \arrow[dr, "="] & \\
1 & & A
\end{tikzcd}
\qquad
\begin{tikzcd}
 & A' \arrow[dl, "!"'] \arrow[dr, "f"] & \\
1 & & A
\end{tikzcd}$$

where $A' = \{\epsilon, a_1, a_2, a_1 \cdot b, a_2 \cdot c\}$ distinguishes the a-action which preceeds b from the a-action which preceeds c. Clearly the latter process alone commits to performing either b or c upon performance of an a.

Although the cover system on $Trace(\mathbf{C})$ required to reconstruct $\mathbf{SProc}$ is $\exists^{\mathcal{R}}$ (or $\exists^{\mathcal{E}}$ when $\mathbf{C}$ is regular), $\exists^{\mathcal{I}}$ plays an important role as well and it is worth commenting on the processes which result. For example, with $A = \{\epsilon, a\} \subseteq \{a\}^*$ and $A' = \{\epsilon, a_1, a_2\} \subseteq \{a_1, a_2\}^*$, the following spans

$$\begin{tikzcd}
 & A \arrow[dl, "!"'] \arrow[dr, "="] & \\
1 & & A
\end{tikzcd}
\qquad
\begin{tikzcd}
 & A' \arrow[dl, "!"'] \arrow[dr, "f"] & \\
1 & & A
\end{tikzcd}$$

represent distinct processes. In fact, $\exists^{\mathcal{I}}$ distinguishes processes with different amounts of nondeterminism — even when that nondeterminism is not visible.

The application of cover systems to morphisms in span categories reveals a subtlety in the relationship between bisimulation and cover maps, as bisimulation is traditionally viewed as being specific to a label set while cover maps are independent of labelling. As we have seen the category $Trace(\mathbf{C})$ is fibred over $\mathbf{C}$, so it is possible to recapture the notion of (labelled) bisimulation implied by the cover system. However, that information may not be enough to reconstruct the cover system: in particular, for $Trace(\mathbf{C})$ with the cover system $\exists^{\mathcal{R}}$, traces over the same label set are bisimilar only if they are isomorphic.

The construction of processes can in fact be viewed as a 2-functor $Proc : \mathbf{Cov} \longrightarrow \mathbf{Cat}$, where $\mathbf{Cov}$ is the 2-category for which:

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

· 1-cells $F : (\mathbf{X}, \mathcal{X}) \longrightarrow (\mathbf{Y}, \mathcal{Y})$ are functors $F : \mathbf{X} \longrightarrow \mathbf{Y}$ which take $\mathcal{X}$-pullbacks to $\mathcal{Y}$-pullbacks;

· and 2-cells $\alpha : F \Longrightarrow G : (\mathbf{X}, \mathcal{X}) \longrightarrow (\mathbf{Y}, \mathcal{Y})$ are natural transformations $\alpha : F \Longrightarrow G$ whose naturality squares are $\mathcal{Y}$-pullbacks.

For a functor F of $\mathbf{Cov}$, $Proc(F)$ applies F to each leg of a span,

$$\begin{tikzcd} & FP \arrow[dl, "Ff"'] \arrow[dr, "Fg"] & \\ FA & & FB \end{tikzcd}$$

and for $\alpha : F \Longrightarrow G$, $Proc(\alpha)_A$ is as follows:

$$\begin{tikzcd} & FA \arrow[dl, "="'] \arrow[dr, "\alpha_A"] & \\ FA & & GA \end{tikzcd}$$

Consequently, one can show that:

Proposition 5.1 *If $\boldsymbol{C}$ is lextensive with cover system $\mathcal{C}$ then $Proc(\boldsymbol{C}, \mathcal{C})$ is compact-closed. Furthermore, if coproducts in $\boldsymbol{C}$ preserve $\mathcal{C}$ then $Proc(\boldsymbol{C}, \mathcal{C})$ has finite biproducts.*

Coproducts in $Trace(\mathbf{C})$ preserve $\exists^{\mathcal{C}}$ whenever coproducts in $\mathbf{C}$ preserve $\mathcal{C}$, so in particular $Proc(Trace(\mathbf{C}), \exists^{\mathcal{I}})$ and $Proc(Trace(\mathbf{C}), \exists^{\mathcal{R}})$ are compact-closed with finite biproducts. It is also easy to see that if $\mathbf{C}$ is regular then $\exists^{\mathcal{E}}$ is preserved by coproducts (as coequalizers and coproducts commute), so $Proc(Trace(\mathbf{C}), \exists^{\mathcal{E}})$ is compact-closed with biproducts.

Note that the choice of $\mathbf{Cat}$ as the endpoint of $Proc$ is certainly not optimal. Of course choosing an appropriate 2-category will involve an abstract characterization of a "process category": if done correctly, one would hope that $Proc$ was part of a 2-adjunction.

6 SProc Reformulated

This section demonstrates that **SProc** can be obtained as an instance of the general construction of the previous section. Specifically,

Proposition 6.1 ***SProc*** *is isomorphic to $Proc(Trace(\boldsymbol{Set}), \exists^{\mathcal{E}})$.*

The proof of this statement makes use of the coinduction principle for non-well-founded sets: two synchronization trees are equal exactly when they are bisimilar. First note that the construction $A\backslash s$ on the objects of **SProc** can be extended to a construction on the spans of $Trace(\mathbf{Set})$ as follows: [1]

$$\begin{tikzcd}
 & P \arrow[dl, "f"'] \arrow[dr, "g"] & & & & P\backslash s \arrow[dl, "f\backslash s"'] \arrow[dr, "g\backslash s"] & \\
A & & B \arrow[r, phantom, "\longmapsto"] & & A\backslash f^*(s) & & B\backslash g^*(s)
\end{tikzcd}$$

The isomorphism is given by a functor F which constructs the synchronization tree of a span. It is defined to have identity effect on objects, and has the following effect on arrows:

$$P_{(f,g)} \qquad \overset{F}{\longmapsto} \qquad \{((f(x), g(x)),\ F(P_{(f,g)}\backslash x)) \mid x \in S_P\}$$

It is easily seen that $P_{(f,g)} : A \longrightarrow B$ implies $F(P_{(f,g)}) : A \longrightarrow B$. That identities and composition are preserved is witnessed by the following bisimulations:

$$\{(id_{A\backslash s},\ F(A_{(id,id)}\backslash s)) \mid s \in S_A\}$$
$$\{(F(P_{(f,g)}\backslash r_0^*(s)); F(Q_{(h,k)}\backslash r_1^*(s)),\ F((P_{(f,g)}; Q_{(h,k)})\backslash s)) \mid s \in S_R\}$$

where $(R,\ r_0,\ r_1)$ is a pullback of g and h in $Trace(\mathbf{Set})$ (i.e. $R_{(r_0;f,\ r_1;k)}$ is the composite of $P_{(f,g)}$ and $Q_{(h,k)}$). Finally, if $C_{(c,d)}$ is a span of cover maps equating $P_{(f,g)}$ and $Q_{(h,k)}$, then the following is a bisimulation equating $F(P_{(f,g)})$ and $F(Q_{(h,k)})$:

$$\{(F(P_{(f,g)}\backslash c^*(s)),\ F(Q_{(h,k)}\backslash d^*(s)) \mid s \in S_C\}$$

As the categories in question have the same objects and F has identity effect on objects, it is enough to show that F is full and faithful.

To show that F is full we construct a span $G(p)$ from a map p in **SProc**, and then show $p = F(G(p))$. First form the traces of a synchronization tree, but with the labels separated:

$$G_S(p) = \{\epsilon\} \cup \{(a, q) \cdot s \mid p \xrightarrow{a} q \ \wedge\ s \in G_S(q)\}.$$

The span $G(p)$ associated to process $p : A \longrightarrow B$ is then:

$$\begin{tikzcd}
 & G_S(p) \arrow[dl, dashed] \arrow[dr, dashed] \arrow[d] & \\
S_A \arrow[dd] & (\Sigma_{A\times B} \times ST)^* \arrow[dl] \arrow[dr] & S_B \arrow[dd] \\
 & & \\
{\Sigma_A}^* & & {\Sigma_B}^*
\end{tikzcd}$$

where the lower components are $(fst; fst)^*$ and $(fst; snd)^*$, respectively, and the upper components are induced as $fst^*(G_S(p)) = traces(p)$. It follows from

[1] We will abuse notation by referring to the components f_Σ of a map f of $Trace(\mathbf{Set})$ also as f, relying on context to remove ambiguity.

this definition that $F(G(p)\backslash(a,q)) = F(G(q))$, consequently the relation below is a bisimulation.

$$\{(p,\ F(G(p))) \mid p \in \mathit{arrows}(\mathbf{SProc})\}$$

To see that F is faithful, suppose $F(P_{(f,g)}) = F(Q_{(h,k)})$. The span $C_{(c,d)}$ below,

$$\begin{tikzcd}
 & S_C \arrow[dr, tail] & & & \\
 & & S_{P\times Q} \arrow[dll, dashed] \arrow[drr, dashed] \arrow[dd] & & \\
S_P \arrow[dd] & & & & S_Q \arrow[dd] \\
 & & \Sigma_{P\times Q}{}^* \arrow[dll, "fst^*"'] \arrow[drr, "snd^*"] & & \\
\Sigma_P{}^* & & & & \Sigma_Q{}^*
\end{tikzcd}$$

where $S_C = \{s \mid \langle f,g\rangle^*(fst^*(s)) = \langle h,k\rangle^*(snd^*(s))\}$, equates $P_{(f,g)}$ and $Q_{(h,k)}$. Clearly $c;f = d;h$ and $c;g = d;k$. To see that $c \in \exists_{\mathcal{E}}$, suppose $s \cdot x \in S_P$. Then $F(P)\overset{s}{\Longrightarrow}F(P\backslash s)$ and $F(P\backslash s)\xrightarrow{f(x),g(x)}F(P\backslash s\cdot x)$. An induction on s shows that

$$\exists t \in S_Q.\ \langle f,g\rangle^*(s) = \langle h,k\rangle^*(t)\ \wedge\ F(Q)\overset{t}{\Longrightarrow}F(Q\backslash t)\ \wedge\ F(P\backslash s) = F(Q\backslash t)$$

So $y \in \Sigma_Q$ exists such that $\langle f,g\rangle(x) = \langle h,k\rangle(y)$ and $F(Q\backslash t)\xrightarrow{h(y),k(y)}F(Q\backslash t\cdot y)$. Then, as $S_C \subseteq S_{P\times Q}$, there exists $u \in S_C$ for which $fst^*(u) = s\cdot x$, as required.

7 SProc Revisited

This section outlines how **SProc** is obtained through our construction starting with either $DTran(\mathbf{C})$ or $Tran(\mathbf{C})$ as base categories. Although simpler than the formulation presented above, each reformulation is most easily seen with the previous one in hand. We begin by developing a general result concerning the 2-functor $Proc$.

We say that a cover system $\mathcal{X}$ is *left-factor closed* if f is an $\mathcal{X}$-map whenever both $f;g$ and g are $\mathcal{X}$-maps. Examples of left-factor closed cover systems are the isomorphisms $\mathcal{I}$ of any category and $\exists^{\mathcal{I}}$ as defined for any of the categories $Trace(\mathbf{C})$, $DTran(\mathbf{C})$ or $Tran(\mathbf{C})$.

A span whose legs belong to a left-factor closed cover system $\mathcal{X}$ is an isomorphism in $Proc(\mathbf{X},\mathcal{X}')$ for any cover system $\mathcal{X}'$ containing $\mathcal{X}$:

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

Consequently,

Lemma 7.1 *If* $(\eta, \epsilon) : F \Longrightarrow G : (\boldsymbol{X}, \mathcal{X}) \longrightarrow (\boldsymbol{Y}, \mathcal{Y})$ *is a* ***Cov****-adjunction*[2] *such that* $\eta_X \in \mathcal{X}'$ *and* $\epsilon_Y \in \mathcal{Y}'$ *for left-factor closed cover systems* $\mathcal{X}' \subseteq \mathcal{X}$ *and* $\mathcal{Y}' \subseteq \mathcal{Y}$*, then* $Proc(\boldsymbol{X}, \mathcal{X})$ *is equivalent to* $Proc(\boldsymbol{Y}, \mathcal{Y})$*.*

An adjunction (actually a coreflection) of the require form exists between $Trace(\mathbf{C})$ and $DTran(\mathbf{C})$: the right adjoint constructs the traces of a transition system in the standard way, and the counit is an $\exists^{\mathcal{I}}$-map at each component. As $\exists^{\mathcal{I}} \subseteq \exists^{\mathcal{C}}$ for any $\mathcal{C}$, $Proc(DTran(\mathbf{C}), \exists^{\mathcal{C}})$ is equivalent to $Proc(Trace(\mathbf{C}), \exists^{\mathcal{C}})$. Thus,

Proposition 7.2 ***SProc*** *is equivalent to* $Proc(DTran(\boldsymbol{Set}), \exists^{\mathcal{E}})$*.*

Although there is no similar adjunction between $DTran(\mathbf{C})$ and $Tran(\mathbf{C})$, there are two functors which appear to be natural candidates for an adjunction: a deterministic transition system is easily a transition system as well, and one makes a transition system deterministic by adding (target) state information to the labels. These functors give rise to an equivalence between the respective process categories.

Proposition 7.3 ***SProc*** *is equivalent to* $Proc(Tran(\boldsymbol{Set}), \exists^{\mathcal{E}})$*.*

Each of these alternate formulations has the advantage of admitting finite presentation of many infinite trace specifications. As specifications of allowable interaction, however, the objects of $DTran(\mathbf{C})$ — being deterministic — are clearly preferable to those of $Tran(\mathbf{C})$.

8 Conclusion

The fact that **SProc** arises as the processes of trace specifications, deterministic transitions systems, and transition systems, is not altogether surprising: these afterall are very standard models for interleaved concurrency. It does, however, illustrate a considerable latitude in the choice of starting point when constructing a specific process category.

Abramsky's category **ASProc** [AGN93] is similarly obtained. In this setting, the monads of delay occur at the level of the base category $Trace(\mathbf{C})$, $DTran(\mathbf{C})$, or $Tran(\mathbf{C})$. These monads are special in that the associated Kleisli categories have finite limits, allowing the construction of compact-closed process categories. It is interesting to note that pullbacks and products in these Kleisli categories give the notions of composition and parallel "product" that one expects for asynchronous processes. The cover systems which provide weak equivalence are obtained indirectly, related to the way one would compute weak bisimulation in terms of strong bisimulation on a modified transition system.

[2] The functors and natural transformations involved belong to **Cov**.

The methods described in this paper may also be applied to obtain rather different interaction categories [CS93]. For example, categories of games and strategies [AJ92] can be obtained as a restricted process category of a basic category of games and homomorpisms — the restriction concerns the legs of the spans which must satisfy certain conditions. This construction can also be performed from a variety of starting points rather than, as originally presented, via traces.

Acknowledgements

The second author is grateful for the support of his former supervisor Graham Birtwistle, and both authors are grateful for the support of NSERC. Many thanks to Rick Blute and Tom Fukushima, who provided valuable comments on an earlier draft.

References

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

[AGN93] Abramsky, S., S. Gay and R. Nagarajan, *Constructing and verifying typed processes.* Available by FTP, 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.

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

[CW87] Carboni, A. and R.F.C. Walters, *Cartesian bicategories I.* Journal of Pure and Applied Algebra, 49:11–32, 1987.

[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.

[CS93] Cockett J.R.B. and D. Spooner, *The category of Protocols.* Presented at Dägstuhl, 1993.

[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.

[PW92] Power, A.J. and C.Wells, *A formalism for the specification of essentially algebraic structures in 2-categories.* Mathematical Structures in Computer Science, 2:1-28, 1992.

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