Electronic Notes in Computer Science 1 (1995)

Categories for synchrony and asynchrony

J.R.B. Cockett [1] and D.A. Spooner [1]

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

Abstract

The purpose of this paper is to show how one may construct from a synchronous interaction category, such as **SProc**, a corresponding asynchronous version. Significantly, it is not a simple Kleisli construction, but rather arises due to particular properties of a monad combined with the existence of a certain type of distributive law.

Following earlier work we consider those synchronous interaction categories which arise from model categories through a quotiented span construction: **SProc** arises in this way from labelled transition systems. The quotienting is determined by a cover system which expresses bisimulation. Asynchrony is introduced into a model category by a monad which, in the case of transition systems, adds the ability to idle. To form a process category atop this two further ingredients are required: pullbacks in the Kleisli category, and a cover system to express (weak) bisimulation.

The technical results of the paper provide necessary and sufficient conditions for a Kleisli category to have finite limits. Furthermore, they show how distributive laws can be used to induce cover systems on such Kleisli categories. These provide the ingredients for the construction of asynchronous settings.

1 Introduction

The Interaction Categories of Abramsky [Abr93] promise to provide a unified semantic framework for concurrent and functional programming together with a useful type discipline for concurrent programming. The key example **SProc**, a category of synchronous processes, was shown in [CS94a] to arise as a span category quotiented by a cover system. This paper develops the general categorical machinery for introducing asynchrony in such process categories, and

[1] Research partially supported by NSERC Canada.

illustrates these techniques through a reconstruction of Abramsky's **ASProc**, a category of asynchronous processes [AGN93,Abr94].

An asynchronous process category is constructed in the same manner as a synchronous process category, and thus its construction involves two steps: identifying an appropriate model category; and selecting a cover system to express bisimulation. As the first step, an asynchronous model category arises from a synchronous model category as the Kleisli category of a monad which adds the ability to idle. As Kleisli categories do not generally have pullbacks, the description of those monads which support the process construction constitutes the main technical result of the paper. As the second step, cover systems on asynchronous model categories arise from cover systems on synchronous model categories through certain distributive laws — in the case of **ASProc**, the distributive law elides idling. Although we illustrate the theory through the construction of **ASProc**, the method can be applied more generally: in a sequel we will show how it is applied in the game theoretic interaction categories of [AJ92,AJM94].

Synchronous Processes

To construct the process category **SProc**, one can begin with the model category **Tran** of deterministic transition systems. The objects of **Tran** are structures $(R \subseteq S \times \Sigma \times S,\ i \in S)$ such that[2] $s \xrightarrow{x} t_1 \in R$ and $s \xrightarrow{x} t_2 \in R$ implies $t_1 = t_2$; the maps $f : A \longrightarrow B$ are pairs of functions $(f_S : S_A \longrightarrow S_B,\ f_\Sigma : \Sigma_A \longrightarrow \Sigma_B)$ such that[3] $f(i_A) = i_B$ and that $s \xrightarrow{x} t \in R_A$ implies $fs \xrightarrow{fx} ft \in R_B$.

As put forth in [CS94a,CS94b] one can view a span of maps in **Tran**

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

as a process $A \longleftrightarrow B$: the endpoints A and B serve as interface specifications, and the legs f and g determine the visible effect of transitions in the apex P. Such a process corresponds to a (nondeterministic) transition system

$$(\{s \xrightarrow{fx,gx} t \mid s \xrightarrow{x} t \in R_P\},\ i_P)$$

and thus to a (proto) morphism of **SProc**. Span composition (given by pullback) implements the composition of morphisms in **SProc**, which is given by restricted parallel composition in the sense of SCCS[Mil83]. Finite products in **Tran** induce a tensor on **SProc** which corresponds to the synchronous product (without communication) of SCCS.

[2] We use $s \xrightarrow{x} t$ to abbreviate (s, x, t), and $s \xrightarrow{x}$ to indicate $\exists t.\, s \xrightarrow{x} t$.

[3] We will drop the subscripts S and Σ on component maps when unambiguous.

Finally, bisimulation equivalence of processes is given by the class of maps used by [JNW93] to characterize bisimulation equivalence of transition systems.

Asynchronous Processes

The model category used to construct **ASProc** arises from a monad D on **Tran**, which corresponds to the combination of the monads δ and Δ of [Mil83,Abr93]. D introduces a new action $*$ which provides an "idle" transition at each state:

$$DA \stackrel{def}{=} (R_A \cup \{s \xrightarrow{*} s \mid s \in S_A\},\ i_A)$$

A span in the Kleisli category $\mathbf{Tran}_D$ corresponds to a span of the following form in **Tran**

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

and is viewed as an asynchronous process $A \longleftrightarrow B$: each action of the apex P may correspond to a silent action at either or both interfaces. When viewed through the underlying functor of the Kleisli construction, such processes are *idle* in the sense of Milner[Mil83]. Furthermore, pullbacks and products in $\mathbf{Tran}_D$ yield the notions of composition and tensor product that one expects for asynchronous processes.

Given a transition system A, the transition system MA has actions corresponding to (possibly empty) sequences of actions of A and is constructed in the standard way:

$$MA \stackrel{def}{=} (\{s_0 \xrightarrow{a_1..a_n} s_n \mid s_0 \xrightarrow{a_1} \ldots \xrightarrow{a_n} s_n \in R_A\},\ i_A)$$

M is an endofunctor on **Tran**, and the operation of removing idle actions is a natural transformation $\theta : MD \Longrightarrow M$. In fact θ is a distributive law and thus induces a functor $M_\theta : \mathbf{Tran}_D \longrightarrow \mathbf{Tran}$. The preimage of M_θ on the cover system for strong bisimulation is the cover system for weak bisimulation. Furthermore, this functor extends via the process construction to an embedding of **ASProc** in **SProc**.

As much of the structure of **SProc** can be identified in the model category **Tran**, the question of what synchronous structure passes to the asynchronous setting is answered through the general theory of lifting functorial structure developed in section 3. Unfortunately, very little structure does lift: neither the product nor the coproduct of **Tran** induce a functor (of the appropriate sort) on $\mathbf{Tran}_D$. The latter is the functorial analogue of the fact that weak bisimulation is not a congruence with respect to summation.

Overview

Section 2 desribes the basic construction of process categories as span categories quotiented by cover systems. Section 3 is concerned with obtaining asynchronous model categories, and begins by reviewing the Kleisli construction on a monad. We then characterize a class of monads whose Kleisli categories admit the construction of processes. Afterwards we consider when synchronous constructions are inherited by an asynchronous model category by extending the standard results about lifting functorial structure over monads. Section 4 summarizes the construction of **ASProc** and shows how weak bisimulation is obtained from the machinery of the preceeding section.

2 Preliminaries

This section reviews the techniques used to construct a category of processes as a span category quotiented by a cover system. It describes the model category **Tran** of transition systems used to construct **SProc** as well as the functorial structure of **Tran** used later to construct **ASProc**.

2.1 Notation

For generality, we describe the category of transition systems and its functorial structure in a *lextensive* category (see [CLW92] or [Coc93]). Such categories have 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 if and only if the top row is a coproduct. Although the path construction on transition systems is described in **Set**, we conjecture that it can be performed in any *locos* — a lextensive category with list arithmetic (see [Coc90]).

We write p_0 and p_1 for the product projections and b_0 and b_1 for the coproduct injections. We write Δ for the diagonal map of the product, ∇ as the codiagonal map of the coproduct, and a and s as the associativity and symmetry maps of either. We assume that both $\times$ and $+$ associate to the left, with $\times$ having binding precedence over $+$.

A *club* is a monad whose functor is stable (i.e. preserves pullbacks) and whose unit and multiplication natural transformations are cartesian (i.e. all naturality squares are pullbacks) — see Kelly[Kel91]. Finally, a *double pullback* is a limit of the following diagram:

$$\begin{tikzcd}[column sep=small]
\cdot \arrow[dr] & & \cdot \arrow[dl] \arrow[dr] & & \cdot \arrow[dl] \\
& \cdot & & \cdot &
\end{tikzcd}$$

2.2 Model categories

Let $\mathbf{C}$ be a lextensive category. The category of deterministic transition systems outlined in the introduction is constructed in $\mathbf{C}$ as follows:

Definition 2.1 *$Tran(\boldsymbol{C})$ is the category of models in $\boldsymbol{C}$ of the sketch:*

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

An equivalent view is as the category of models of the sketch below:

$$\begin{tikzcd}
& R \arrow[dl, dashed] \arrow[d, "r"] & 1 \arrow[d, "i"] \\
S\times\Sigma & S\times\Sigma\times S \arrow[l, "p_0"] & S
\end{tikzcd}$$

We will use these two views interchangably as convenient when defining the functorial structure of $Tran(\mathbf{C})$.

Note that to check commutivity of a diagram in $Tran(\mathbf{C})$, once it is established that the maps involved are in $Tran(\mathbf{C})$, it suffices to check commutivity of the state and label components. Thus, for instance, to show a transformation is natural it suffices to show that its state and label components are natural.

$Tran(\mathbf{C})$ has finite limits, with pullbacks and a final object given componentwise, and an initial object which has a single (initial) state and no labels.

2.2.1 A delay monad

The functor $D : Tran(\mathbf{C}) \longrightarrow Tran(\mathbf{C})$ gives a transition system the ability to delay by adding a new label which provides an "idle" action at each state:

$$\begin{tikzcd}
& R \arrow[dl, "r"'] \arrow[dr, "\alpha"] & & & & R+S \arrow[dl, "r+1"'] \arrow[ddr, "\langle\alpha|1\rangle"] & \\
S\times\Sigma & & S & \arrow[r, mapsto, "D"] & S\times\Sigma+S \arrow[d, "\sim"'] & & \\
& & & & S\times(\Sigma+1) & & S
\end{tikzcd}$$

D is given componentwise by the identity and exception monads, and is itself a monad: the unit η_A injects a transition system A into the more premissive DA, and the multiplication μ_A unifies in DA the two separate idle actions of DDA.

Proposition 2.2 *D is a club on $Tran(\boldsymbol{C})$.*

Proof. It is sufficient to show that D is well-defined on maps (i.e. $Df : DA \longrightarrow DB$ whenever $f : A \longrightarrow B$) and that η_A and μ_A are maps. These facts are seen in the following diagrams:

$$
\begin{tikzcd}
S_A\times(\Sigma_A+1) \arrow[r, phantom, "\sim"] \arrow[d, "f_S\times(f_\Sigma+1)"'] & S_A\times\Sigma_A+S_A \arrow[d, "f_S\times f_\Sigma+1"] & R_A+S_A \arrow[l, "r_A+1"'] \arrow[r, "\langle\alpha_A|1\rangle"] \arrow[d, "f_R+f_S"] & S_A \arrow[d, dash, "f_S"] \\
S_B\times(\Sigma_B+1) \arrow[r, phantom, "\sim"] & S_B\times\Sigma_B+S_B & R_B+S_B \arrow[l, "r+1"] \arrow[r, "\langle\alpha_B|1\rangle"'] & S_B
\end{tikzcd}
$$

$$
\begin{tikzcd}
S\times\Sigma \arrow[d, "1\times b_0"'] & & R \arrow[ll, "r"'] \arrow[r, "\alpha"] \arrow[d, "b_0"] & S \arrow[d, equal] \\
S\times(\Sigma+1) \arrow[r, phantom, "\sim"] & S\times\Sigma+S & R+S \arrow[l, "r+1"] \arrow[r, "\alpha"'] & S
\end{tikzcd}
$$

$$
\begin{tikzcd}
S\times(\Sigma+1+1) \arrow[r, phantom, "\sim"] \arrow[d, "1\times(a;1+\nabla)"'] & S\times\Sigma+S+S & R+S+S \arrow[l, "r+1+1"'] \arrow[r, "\langle\alpha|1|1\rangle"] \arrow[d, "a;1+\nabla"] & S \arrow[d, equal] \\
S\times(\Sigma+1) \arrow[r, phantom, "\sim"] & S\times\Sigma+S & R+S \arrow[l, "r+1"] \arrow[r, "\langle\alpha|1\rangle"'] & S
\end{tikzcd}
$$

$\square$

Note that the functor D turns initial objects into final objects, the significance being that 0 will be final in the Kleisli category of D:

Proposition 2.3 *$D(0)$ is final in $Tran(\boldsymbol{C})$.*

2.2.2 A path monad

The functor M : **Tran** $\longrightarrow$ **Tran** constructs a transition system whose states are the same as the original, but whose actions correspond to sequences of actions of the original:

Definition 2.4 *For A an object of **Tran** define $MA = (i \in S, \bigcup_{i<\omega} R^i \subseteq S\times\Sigma^*\times S)$, where*

$$
\begin{aligned}
R^0 &= \{(s, [], s) \mid s \in S\} \\
R^{i+1} &= \{(s, a :: \ell, t) \mid \exists u.\ (s, a, u) \in R \ \wedge\ (u, \ell, t) \in R^i\}
\end{aligned}
$$

The state and label components of M are given by the identity and list monads respectively: the label components of the unit and multiplication are thus:

$$inj : A \longrightarrow A^* \,; a \mapsto [a]$$
$$flatten : A^{**} \longrightarrow A^* \,; \begin{cases} [] \mapsto [] \\ \ell :: \ell s \mapsto append(\ell, flatten(\ell s)) \end{cases}$$

Proposition 2.5 *M is a club on **Tran**.*

Proof. MA is a deterministic transition system as all R_i are deterministic and involve distinct labels. The effect of M on maps is given componentwise by the identity and list monads, and a simple induction on the structure of the labels shows this is well-defined.

To see M is stable, let P be the pullback [4] of f and g and consider the induced map h to the pullback of Mf and Mg:

$$\begin{tikzcd} MP \arrow[rr, "M\pi"] \arrow[dr, dashed, "h"] \arrow[dd, "M\pi'"'] & & MA \arrow[dd, "Mf"] \\ & Q \arrow[ur, "\pi"'] \arrow[dl, "\pi'"] & \\ MB \arrow[rr, "Mg"'] & & MC \end{tikzcd}$$

Define $h' : Q \longrightarrow MP$ such that $(s,t) \mapsto (s,t)$ and $(l,m) \mapsto zip(l,m)$. An induction on the structure of the labels of Q shows h' is well-defined. To see that h' is the inverse of h it is sufficient to consider the label component and note that $unzip; zip$ is the identity on $(A{\times}B)^*$, and $zip; unzip$ is the identity on $\{(\ell, m) \mid length(\ell) = length(m)\} \subseteq A^*{\times}B^*$. □

As with D, M turns initial objects into final objects:

Proposition 2.6 *$M(0)$ is final in **Tran**.*

Consider an object MDA which results by performing the path construction upon a transition system with delays. There is a natural map $\theta_A : MDA \longrightarrow MA$ which strips idle components from the actions sequences of MDA. This map has the identity effect on states, and the following effect on labels:

$$\begin{tikzcd} (A+1)^* \arrow[r, "(inj+nil)^*"] \arrow[drr, dashed, "\theta"'] & (A^*+A^*)^* \arrow[r, "\nabla^*"] & A^{**} \arrow[d, "flatten"] \\ & & A^* \end{tikzcd}$$

[4] We take the pullback of $f : A \longrightarrow C$ and $g : B \longrightarrow C$ in **Set** to be $\{(a,b) \mid f(a) = g(b)\} \subseteq A{\times}B$ together with the appropriate restrictions of the projections and pairing maps.

Proposition 2.7 *$\theta : MD \Longrightarrow M$ is a cartesian natural transformation.*

Proof. θ_A is seen to be a map by induction on the structure of the labels of MDA. Suppose $(s, \ell, t) \in MDA$:

i) $\ell = []$ implies $s = t$ and $(s, \theta(\ell) = [], t) \in MA$.
ii) $\ell = a :: m$ implies there exists $u \in S_A$ such that $(s, a, u) \in DA$ and $(u, m, t) \in MDA$. Furthermore, $(u, \theta(m), t) \in MA$ by inductive hypothesis. If $a \in \Sigma_A$ then $(s, a, u) \in A$ and so $(s, \theta(\ell) = a :: \theta(m), t) \in MA$. Otherwise $u = s$ and so $(s, \theta(\ell) = \theta(m), t) \in MA$.

To see θ is cartesian, consider the induced map h to the pullback in the diagram below:

$$\begin{tikzcd}
MDA \arrow[rr, "\theta"] \arrow[dr, dashed, "h"] \arrow[dd, "MDf"'] & & MA \arrow[dd, "Mf"] \\
& Q \arrow[ur, "\pi"'] \arrow[dl, "\pi'"] \arrow[dr, phantom, "\lrcorner", very near start] & \\
MDB \arrow[rr, "\theta"'] & & MB
\end{tikzcd}$$

Define $k : Q \longrightarrow MDA$ such that for states $(s, s') \mapsto s$ and for labels:

$$\begin{aligned}
k([], []) &= [] \\
k(\ell, * :: m) &= * :: k(\ell, m) \\
k(a :: \ell, f(a) :: m) &= a :: k(\ell, m)
\end{aligned}$$

k is seen to be a map by induction on the structure of the labels of Q, and is then seen to be the inverse of h componentwise. $\square$

We will see later that θ is a distribution which allows us to obtain weak equivalence of asynchronous processes.

2.3 Cover systems

Cover systems capture the properties required of a class of maps to induce a congruence on a span category, and thus to provide a compositional notion of equivalence on processes. A detailed development of the results sketched here can be found in [CS94b].

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

Definition 2.8 *A collection $\mathcal{X}$ of the maps of $\mathbf{X}$ is a **cover system** provided it contains all isomorphisms, is closed under composition, and is closed under pulling back along arbitrary maps — i.e. if x is in $\mathcal{X}$ and the following is a* *pullback then y is in $\mathcal{X}$:*

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

Examples of cover systems in any category are the isomorphisms $\mathcal{I}$, the retractions $\mathcal{R}$, and the monics $\mathcal{M}$. 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. The cartesian maps of any fibration also form a left-factor closed cover system.

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

Definition 2.9 *A commuting square in $\boldsymbol{X}$ is an **$\mathcal{X}$-pullback** if the induced map to the inscribed pullback is in $\mathcal{X}$:*

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

A pullback is an $\mathcal{X}$-pullback for any $\mathcal{X}$, and a map f in $\mathbf{X}$ is an $\mathcal{X}$-map if and only if the square $f;1 = f;1$ is an $\mathcal{X}$-pullback. Cover-pullbacks satisfy some of the same properties as pullbacks. Specifically, in the following diagram:

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

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

As the unit η of a club (T, η, μ) is cartesian, the functor T reflects covers and reflects pullbacks, and consequently reflects cover pullbacks. These facts are useful to establish the results of section 3.

One method of obtaining cover systems on model categories is as follows. If $\mathcal{S}$ is a sketch and a an arrow of $\mathcal{S}$, then the morphisms of $Mod(\mathcal{S}, \mathbf{X})$ for which the naturality square corresponding to a is an $\mathcal{X}$-pullback form a cover system on $Mod(\mathcal{S}, \mathbf{X})$. Furthermore, this cover system is left-factor closed whenever $\mathcal{X}$ is left-factor closed.

Thus any cover system $\mathcal{C}$ on $\mathbf{C}$ yields the following cover system on $Tran(\mathbf{C})$:

Definition 2.10 $\exists^{\mathcal{C}}$ *is the class of maps $f : A \longrightarrow B$ of $Tran(\boldsymbol{C})$ for which* *the following square is a $\mathcal{C}$-pullback in* $\boldsymbol{C}$*:*

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

In **Tran**, for instance, 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 property that each transition from a state $f(s)$ is the image of at least one transition from s. Note that $\exists^{\mathcal{M}}$ does not provide a particularly useful cover system as it contains the map $0 \longrightarrow A$ for all objects A of $Tran(\mathbf{C})$.

Later in the paper we show how weak bisimulation arises. If $\mathcal{X}$ is a cover system on $\mathbf{X}$ and $G : \mathbf{Y} \longrightarrow \mathbf{X}$ takes pullbacks to $\mathcal{X}$-pullbacks, then $G^{-1}(\mathcal{X})$ is a cover system on $\mathbf{Y}$ which is left-factor closed whenever $\mathcal{X}$ is left-factor closed. The cover system for weak bisimulation is obtained by constructing a stable functor from the Kleisli category $\mathbf{Tran}_D$ back to **Tran** and taking the preimage of $\exists^{\mathcal{R}}$.

2.4 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 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 $\mathbf{X}$. Span composition is given by pullback — i.e. $(f, g); (h, k)$ is $(p; f,\ q; k)$, where:

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

A cover system $\mathcal{X}$ on $\mathbf{X}$ induces a congruence on spans: (f, g) and (h, k) are

$\mathcal{X}$-*bisimilar* 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 $\mathbf{X}$. Quotienting 1-cells by this congruence one obtains a category $Proc(\mathbf{X}, \mathcal{X})$ — processes on $\mathbf{X}$ upto $\mathcal{X}$-bisimulation.

Certainly the simplest examples of this construction are span categories and categories of relations: for $\mathbf{X}$ with pullbacks, $Proc(\mathbf{X}, \mathcal{I})$ is written $Span(\mathbf{X})$; and for $\mathbf{E}$ a regular category, $Proc(\mathbf{E}, \mathcal{E})$ is written $Rel(\mathbf{E})$. It is shown in [CS94b] that $Proc(\mathbf{Tran}, \exists^{\mathcal{R}})$ is equivalent to $\mathbf{SProc}$, and furthermore that $\mathbf{SProc}$ arises as the process category on a variety of related model categories.

The construction of process categories can be viewed as a 2-functor $Proc$: The domain of the construction $Proc$ is the 2-category $\underline{\mathbf{Cov}}$ whose 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 *cover-stable* (or $\mathcal{X}$-stable in that $\mathcal{X}$-pullbacks are taken to $\mathcal{Y}$-pullbacks), and 2-cells $\alpha : F \Longrightarrow G : (\mathbf{X}, \mathcal{X}) \longrightarrow (\mathbf{Y}, \mathcal{Y})$ are natural transformations $F \Longrightarrow G$ which are *cover-cartesian* (or $\mathcal{Y}$-cartesian in that all naturality squares are $\mathcal{Y}$-pullbacks). Thus any functorial structure on $\mathbf{X}$ will occur also in $Proc(\mathbf{X}, \mathcal{X})$ provided the functors and natural transformations involved exist in $\underline{\mathbf{Cov}}$. For functors F, $Proc(F)$ applies F to each leg of a span; for natural transformations $\alpha : F \Longrightarrow G$, $Proc(\alpha)$ at A is the trivial span (id_{GA}, α_A). It is shown in [CS94b] that $Proc(\mathbf{X}, \mathcal{X})$ is compact closed for any $\mathcal{X}$ when $\mathbf{X}$ has products, and that $Proc(\mathbf{X}, \mathcal{X})$ has biproducts whenever coproducts in $\mathbf{X}$ are given by a $\underline{\mathbf{Cov}}$-adjunction.

Note that a functor $G : \mathbf{X} \longrightarrow \mathbf{Y}$ is a cover-stable functor $(\mathbf{X}, \mathcal{X}) \longrightarrow (\mathbf{Y}, \mathcal{Y})$ if and only if G preserves covers and takes pullbacks to $\mathcal{Y}$-pullbacks. Thus any $G : (\mathbf{X}, \mathcal{X}) \longrightarrow (\mathbf{Y}, \mathcal{Y})$ can be factored into a *cover-increasing* map $(\mathbf{X}, \mathcal{X}) \hookrightarrow (\mathbf{X}, G^{-1}(\mathcal{Y}))$ followed by a *cover-reflecting* map $G : (\mathbf{X}, G^{-1}(\mathcal{Y})) \longrightarrow (\mathbf{Y}, \mathcal{Y})$. Taking $\mathcal{C}_I$ to consist of those functors $I : (\mathbf{X}, \mathcal{X}) \longrightarrow (\mathbf{Y}, \mathcal{Y})$ for which $I : \mathbf{X} \longrightarrow \mathbf{Y}$ is an isomorphism and $\mathcal{C}_R$ to consist of the functors which reflect covers,

Proposition 2.11 *$(\mathcal{C}_I, \mathcal{C}_R)$ is an $(\mathcal{E}, \mathcal{M})$-factorization system on $\underline{\mathbf{Cov}}$.*

Proof. Each class clearly contains all isomorphisms and is closed to composition. Suppose the following commutes with $I \in \mathcal{C}_I$ and $R \in \mathcal{C}_R$:

$$\begin{tikzcd}
(\mathbf{X}, \mathcal{X}) \arrow[r, "I"] \arrow[d, "G"'] & (\mathbf{Y}, \mathcal{Y}) \arrow[d, "H"] \\
(\mathbf{Z}, \mathcal{Z}) \arrow[r, "R"'] & (\mathbf{W}, \mathcal{W})
\end{tikzcd}$$

The required fill-in is given by GI^{-1}. I^{-1} need not preserve covers, but if $y \in \mathcal{Y}$ then $Hy = RGI^{-1}y \in \mathcal{W}$ and thus $GI^{-1}y \in \mathcal{Z}$. □

3 Asynchronous model categories

This section is concerned with introducing asynchrony into model categories via monads. We begin by reviewing the Kleisli construction and the standard results relevant to the subsequent development. We then identify a class of monads whose Kleisli categories admit the construction of processes — in particular, monads whose Kleisli categories have pullbacks. A cover system on the underlying category induces a canonical cover system on the Kleisli category, and we characterize abstractly the conditions for lifting functorial structure in <u>**Cov**</u> over these monads.

3.1 Review of the Kleisli construction

A monad on a category $\mathbf{X}$ is a functor $T : \mathbf{X} \longrightarrow \mathbf{X}$ together with natural transformations $\eta : Id \Longrightarrow T$ (the *unit*) and $\mu : TT \Longrightarrow T$ (the *multiplication*) such that the following commute for all A in $\mathbf{X}$:

$$\begin{tikzcd}
 & TA \arrow[dl, "T\eta"'] \arrow[d, equal] \arrow[dr, "\eta_T"] & \\
T^2A \arrow[r, "\mu"'] & TA & T^2A \arrow[l, "\mu"]
\end{tikzcd}
\qquad
\begin{tikzcd}
T^3A \arrow[r, "T\mu"] \arrow[d, "\mu"'] & T^2A \arrow[d, "\mu"] \\
T^2A \arrow[r, "\mu"'] & TA
\end{tikzcd}$$

A club is a monad whose functor is stable and whose natural transformations are cartesian.

Example 3.1 For X an object of a lextensive category $\mathbf{C}$, the monad of exceptions is a club on $\mathbf{C}$: the functor is $(\_{+}X)$, and the unit and multiplication are given by the transformations b_0 and $a; 1{+}\nabla$ of the coproduct. □

Example 3.2 In a locos, the list monad is a club: the functor is $(\_)^*$, and the unit and multiplication are $inj : A \longrightarrow A^*$ and $flatten : A^{**} \longrightarrow A^*$. □

Given any monad (T, η, μ) on $\mathbf{X}$ one forms the Kleisli category $\mathbf{X}_T$ as follows: The objects are those of $\mathbf{X}$, while maps $A \longrightarrow B$ are given by maps $f : A \longrightarrow TB$ in $\mathbf{X}$. Identities id_A are given by η_A in $\mathbf{X}$, and the composition of maps $f : A \longrightarrow B$ and $g : B \longrightarrow C$ is given in $\mathbf{X}$ by $f; Tg; \mu$.

There is an adjunction $(\eta, \epsilon) : F_T \dashv U_T : \mathbf{X} \longrightarrow \mathbf{X}_T$ associated with the Kleisli construction: the free functor F_T post-composes η to the maps of $\mathbf{X}$, and the underlying functor U_T takes a map g in $\mathbf{X}_T$ to $Tg; \mu$. The counit ϵ at A is given in $\mathbf{X}$ by the identity on TA.

Given monads (T, η, μ) and (S, ι, ν) on $\mathbf{X}$ and $\mathbf{Y}$ respectively, there are precise conditions for lifting functors $G : \mathbf{X} \longrightarrow \mathbf{Y}$ through the Kleisli construction. A natural transformation $\lambda : GT \Longrightarrow SG$ allows one to turn a map $f : A \longrightarrow TB$ in $\mathbf{X}$ into a map $Gf; \lambda : GA \longrightarrow SGA$ in $\mathbf{Y}$. This operation is a functor $G_\lambda : \mathbf{X}_T \longrightarrow \mathbf{Y}_S$ if and only if

$$
\begin{tikzcd}
 & GA \arrow[dl, "G\eta"'] \arrow[dr, "\iota"] & \\
GTA \arrow[rr, "\lambda"'] & & SGA
\end{tikzcd}
\qquad
\begin{tikzcd}
GTTA \arrow[r, "\lambda"] \arrow[d, "G\mu"'] & SGTA \arrow[r, "S\lambda"] & SSGA \arrow[d, "\nu"] \\
GTA \arrow[rr, "\lambda"'] & & SGA
\end{tikzcd}
$$

commute for all A in $\mathbf{X}$. A natural transformation λ with this property is called a *distribution* for G.

Example 3.3 In a lextensive category $\mathbf{C}$, distributions for both the product and coproduct functors are given by a combination of symmetry, multiplication and tensorial strength $\tau^\bullet : T\_\bullet B \Longrightarrow T(\_\bullet B)$. In each case, τ is a distribution for the functor $(\_\bullet B)$. $\square$

An important special case of a distribution α for G is when S is the identity monad. In this case $G_\alpha : \mathbf{X}_T \longrightarrow \mathbf{Y}$ if and only if $G\eta; \alpha = Id_G$ and $G\mu; \alpha = \alpha; \alpha$ commute for all A in $\mathbf{X}$. Such a distribution is called a *T-action* for G.

Example 3.4 For any monad T, the multiplication $\mu : TT \longrightarrow T$ is a T-action which lifts the functor T to the underlying functor U_T. $\square$

Example 3.5 In a locos $\mathbf{C}$, the transformation $\theta : (\_+1)^* \Longrightarrow (\_)^*$ is an action for the exception monad. Furthermore, $\theta : MD \Longrightarrow M$ is an action for the delay monad as it is given componentwise by the identity and θ above; thus $M_\theta : \mathbf{Tran}_D \longrightarrow \mathbf{Tran}$ is a functor. $\square$

Turning now to the lifting of natural transformations, let G and H be functors $\mathbf{X} \longrightarrow \mathbf{Y}$ with distributions λ and κ respectively. A natural transformation $\alpha : G \Longrightarrow H$ lifts to a natural transformation $\tilde{\alpha} : G_\lambda \Longrightarrow H_\kappa$, where $\tilde{\alpha}_A$ is given by $\alpha_A; \iota_{HA}$, if and only if

$$\begin{tikzcd} GTA \arrow[r, "\alpha_{TA}"] \arrow[d, "\lambda_A"'] & HTA \arrow[d, "\kappa_A"] \\ SGA \arrow[r, "S\alpha_A"'] & SHA \end{tikzcd}$$

commutes for all A in $\mathbf{X}$. Such a transformation α is said to *respect* the distributions λ and κ.

Example 3.6 In a lextensive category, the injections b_i and co-copy map ∇ of the coproduct and the projections p_i of the product respect the associated distributions; the copy map Δ of the product, however, does not. $\Box$

The Kleisli construction on a 2-category $\underline{\mathbf{X}}$ can be seen as a 2-functor into $\underline{\mathbf{X}}$. The domain is the 2-category $Dist(\underline{\mathbf{X}})$ whose 0-cells are monads T in $\underline{\mathbf{X}}$, 1-cells $T \longrightarrow S$ are given by distributions $\lambda : GT \Longrightarrow SG$ of $\underline{\mathbf{X}}$, and 2-cells $\lambda \Longrightarrow \kappa$ are given by natural transformations $\alpha : G \Longrightarrow H$ of $\underline{\mathbf{X}}$ which respect the distributions as described above. There is a related 2-category of arrows $Lift(\underline{\mathbf{X}})$ whose 0-cells are again monads in $\underline{\mathbf{X}}$, 1-cells $T \longrightarrow S$ are "liftings", or pairs (G, G') such that $F_T; G' = G; F_S$, and 2-cells are "pillows", or pairs (α, α') such that $F_T; ; \alpha' = \alpha; ; F_S$. The following result appears to be folklore:

Theorem 3.7 *$Dist(\underline{\mathbf{X}})$ is isomorphic to $Lift(\underline{\mathbf{X}})$.*

The proof is based on the fact that the 1-cells of $Lift(\underline{\mathbf{X}})$ correspond exactly to distributions (see [Mul93]).

3.2 Covered Kleisli categories

Here we consider how to obtain a Kleisli category which lies in the domain of the process construction. The first step is to identify those clubs whose Kleisli categories have pullbacks. We then show how additional restrictions allow cover systems in the underlying category to be lifted.

Finitely complete Kleisli categories

It is not difficult to show that a square $p; f = q; g$ is a pullback in a Kleisli category $\mathbf{X}_T$ if and only if it's image via the underlying functor U_T is a pullback in $\mathbf{X}$ — so U_T reflects as well as preserves pullbacks. However, this provides little guidance for constructing pullbacks in $\mathbf{X}_T$.

Let (T, η, μ) be a club on a category $\mathbf{X}$ with pullbacks: We say that T is a *stable monad* when the associated Kleisli category has pullbacks.

Proposition 3.8 *(T, η, μ) is stable if and only if there exists a stable functor $P : \boldsymbol{X} \longrightarrow \boldsymbol{X}$ and cartesian natural transformations $\alpha, \beta : P \Longrightarrow TT$ such that*

$$\begin{tikzcd}
TPA \arrow[r, "T\alpha"] \arrow[d, "T\beta"'] \arrow[dr, phantom, "\lrcorner", very near start] & T^3A \arrow[r, "\mu"] & T^2A \arrow[dd, "\mu"] \\
T^3A \arrow[d, "\mu"'] & & \\
T^2A \arrow[rr, "\mu"'] & & TA
\end{tikzcd}$$

is a pullback for all A in $\boldsymbol{X}$.

Proof. Note that ϵ_A, the counit of the Kleisli adjunction, is taken by U to μ_A in $\mathbf{X}$. So if $\mathbf{X}_T$ has pullbacks, the pullback of ϵ_A along itself is taken by U to the diagram above. It is not difficult to show that P is a stable functor and that α and β are cartesian natural transformations.

Conversely, if P, α and β are as stated, a pullback of f and g in $\mathbf{X}_T$ is given by a pullback of Uf and Ug in $\mathbf{X}$ which lies in the image of U: forming the double pullback (x, y) of (Tg, β, α, Tf) makes

$$\begin{tikzcd}
TZ \arrow[rr, "Tx"] \arrow[dd, "Ty"'] \arrow[dr, phantom, "\lrcorner", very near start] & & T^2A \arrow[r, "\mu"] \arrow[d, "T^2f"'] & TA \arrow[d, "Tf"] \\
& TPC \arrow[r, "T\alpha"] \arrow[d, "T\beta"'] & T^3C \arrow[r, "\mu"] & T^2C \arrow[dd, "\mu"] \\
T^2B \arrow[r, "T^2g"] \arrow[d, "\mu"'] & T^3C \arrow[d, "\mu"] & & \\
TB \arrow[r, "Tg"'] & T^2C \arrow[rr, "\mu"'] & & TC
\end{tikzcd}$$

a pullback in $\mathbf{X}$ (as T is stable and μ is cartesian), and thus (x, y) is a pullback of f and g in $\mathbf{X}_T$. $\square$

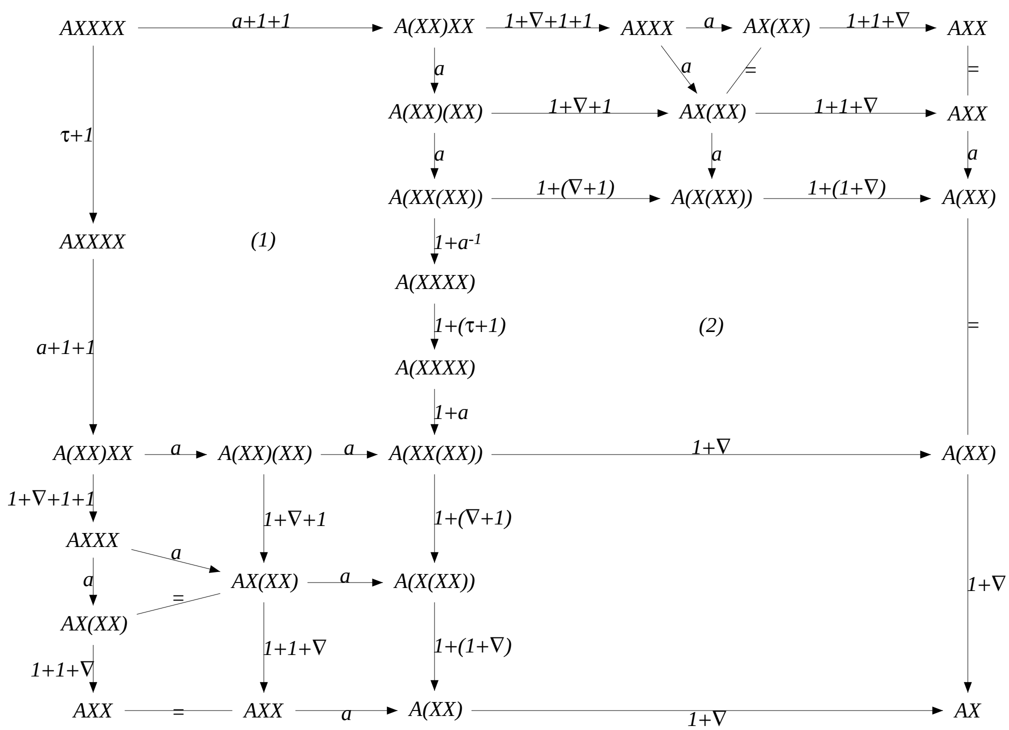
We will use the naming convention of P_T, μ_0 and μ_1 when referring to the additional components of a stable club (T, η, μ).

Given pullbacks, one secures finite limits in the presence of a final object. It is easily seen that

Proposition 3.9 *Z is final in $\boldsymbol{X}_T$ if and only if TZ is final in $\boldsymbol{X}$.*

Example 3.10 *In a lextensive category, any exception monad $(\_+X)$ is stable. Furthermore, if X is final then the Kleisli category of $(\_+X)$ has finite limits.*

Proof. Clearly $0+1 \cong 1$. The additional structure of a stable monad arises from the isomorphism $\tau^+ : A + X + X \longrightarrow A + X + X$ which serves to swap the order of exceptions. The required pullback is constructed in the following diagram, where the coproduct of objects is written as juxtaposition:



(1) and (2) are easily shown to commute and are thus pullbacks as opposing sides are isomorphisms. □

Example 3.11 The delay monad D on $Tran(\mathbf{C})$ is a stable monad as it is given componentwise by the exception and identity monads. In addition $D0 \cong 1$, so $Tran(\mathbf{C})_D$ has finite limits. □

Lifting cover systems

We are interested in monads which exist in the 2-category <u>**Cov**</u>, so in addition to preserving pullbacks the functors must also preserve the chosen cover system. If T is a stable monad on a category $\mathbf{X}$ with a cover system $\mathcal{X}$, then we refer to T as $\mathcal{X}$*-stable* (or *cover-stable*) provided T preserves $\mathcal{X}$ and has the property that every isomorphism j of $\mathbf{X}_T$ is $F(i)$ for some isomorphism i of $\mathbf{X}$. For T an $\mathcal{X}$-stable monad on $\mathbf{X}$,

Definition 3.12 *$\mathcal{X}_T$ is the class of maps $F_T(\mathcal{X})$ in $\boldsymbol{X}_T$.*

Proposition 3.13 *If T is an $\mathcal{X}$-stable monad on $\boldsymbol{X}$ then: i) $\mathcal{X}_T$ is a cover system on $\boldsymbol{X}_T$; ii) $\mathcal{X}_T$ is left-factor closed if and only if $\mathcal{X}$ is left-factor closed.*

Proof. We show only i), as ii) is straightforward. $\mathcal{X}_T$ contains all isomorphisms by definition, and is closed to composition as F_T is a functor. To see that $\mathcal{X}_T$ is closed to pullback, suppose $x \in \mathcal{X}$ and $f \in \mathbf{X}_T$. The following pullback in

$\mathbf{X}$ corresponds to a pullback $g; Fx = Fy; f$ in $\mathbf{X}_T$:

$$\begin{tikzcd} \cdot \arrow[r, "Tg"] \arrow[d, "Ty"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r, "\mu"] \arrow[d, "T^2x"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d, "Tx"] \\ \cdot \arrow[r, "Tf"'] & \cdot \arrow[r, "\mu"'] & \cdot \end{tikzcd}$$

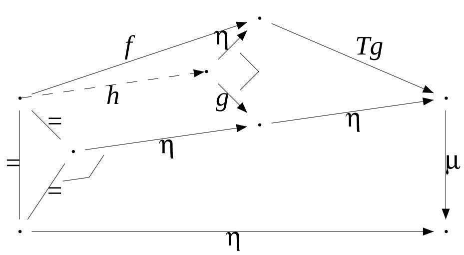
$y \in \mathcal{X}$ as T preserves and reflects $\mathcal{X}$, and thus $Fy \in \mathcal{X}_T$. □

Even if all isomorphisms of $\mathbf{X}_T$ do not arise by lifting isomorphisms of $\mathbf{X}$, one can obtain a cover system by adding all isomorphisms to $F_T(\mathcal{X})$ and then closing to composition. Left-factor closure, however, is not preserved by this construction.

Lemma 3.14 *$\mathcal{I}_{X_T} = F_T(\mathcal{I}_X)$ whenever η is monic and the following is a pullback for all A in $\boldsymbol{X}$:*

$$\begin{tikzcd} A \arrow[r, "\eta"] \arrow[d, equal] \arrow[drr, phantom, "\lrcorner", very near start] & TA \arrow[r, "\eta"] & T^2A \arrow[d, "\mu"] \\ A \arrow[rr, "\eta"'] & & TA \end{tikzcd}$$

Proof. Suppose f is an isomorphism with inverse g in $\mathbf{X}_T$. Then $f; Tg; \mu = \eta$ in $\mathbf{X}$ and as η is cartesian there is a map h such that $f = F_T(h)$:



Similarly, there exists k such that $g = F_T(k)$. Furthermore, $h; k = 1$ and $k; h = 1$ as $\eta = b_0$ is monic. □

Example 3.15 *In a lextensive category $\boldsymbol{C}$ with cover system $\mathcal{C}$, the exception monad is $\mathcal{C}$-stable provided coproducts preserve $\mathcal{C}$.*

Proof. Coproduct injections are monic in a lextensive category, and the pullback diagram of lemma 3.14 appears below:

$$\begin{tikzcd} A \arrow[r, "b_0"] \arrow[d, equal] \arrow[drr, phantom, "\lrcorner", very near start] & A+X \arrow[r, "b_0"] & A+X+X \arrow[d, "a"] \\ A \arrow[rr, "b_0"] \arrow[d, equal] \arrow[drr, phantom, "\lrcorner", very near start] & & A+(X+X) \arrow[d, "1+\nabla"] \\ A \arrow[rr, "b_0"'] & & A+X \end{tikzcd}$$

□

Example 3.16 *The delay monad D is $\exists^{\mathcal{C}}$-stable on $Tran(\boldsymbol{C})$ provided coproducts in $\boldsymbol{C}$ preserve $\mathcal{C}$.*

Proof. To see that D preserves $\exists^{\mathcal{C}}$-maps, suppose $f : A \longrightarrow B \in \exists^{\mathcal{C}}$. Then the following is a $\mathcal{C}$-pullback as $+$ is $\mathcal{C}$-stable and ∇ is cartesian:

$$\begin{tikzcd}
& P_A+S_A \arrow[rr, "f_P+f_S"] \arrow[d, "p_A+1"'] & & P_B+S_B \arrow[d, "p_B+1"] \\
& S_A\times\Sigma_A+S_A \arrow[dl, "\pi+1"'] \arrow[dd, dash, "\sim"] & & S_B\times\Sigma_B+S_B \arrow[dl, "\pi+1"'] \arrow[dd, dash, "\sim"] \\
S_A+S_A \arrow[rr, "f_S+f_S"] \arrow[ddr, "\nabla"'] & & S_B+S_B \arrow[ddr, "\nabla"'] & \\
& S_A\times(\Sigma_A+1) \arrow[d, "\pi"] & & S_B\times(\Sigma_B+1) \arrow[d, "\pi"] \\
& S_A \arrow[rr, "f_S"'] & & S_B
\end{tikzcd}$$

□

3.3 Lifting functorial structure

We now extend the standard conditions for lifting functorial structure above monads to conditions for lifting functors and natural transformations of <u>**Cov**</u> above cover-stable monads.

Let (T, η, μ) and (S, ι, ν) be cover-stable monads on $(\mathbf{X}, \mathcal{X})$ and $(\mathbf{Y}, \mathcal{Y})$ respectively, $G : (\mathbf{X}, \mathcal{X}) \longrightarrow (\mathbf{Y}, \mathcal{Y})$, and $\lambda : GT \Longrightarrow SG$ a $\mathcal{Y}$-cartesian distribution for $G : \mathbf{X} \longrightarrow \mathbf{Y}$. We say λ is a *cover-stable distribution* if and only if $G_\lambda : (\mathbf{X}_T, \mathcal{X}_T) \longrightarrow (\mathbf{Y}_T, \mathcal{Y}_T)$.

Proposition 3.17 *λ is a cover-stable distribution if and only if*

$$\begin{tikzcd}
GP_TA \arrow[r, "G\mu_0"] \arrow[d, "G\mu_1"'] & GT^2A \arrow[r, "\lambda"] & SGTA \arrow[d, "S\lambda"] \\
GT^2A \arrow[d, "\lambda"'] & P_SGA \arrow[r, "\nu_0"] \arrow[d, "\nu_1"'] & S^2GA \\
SGTA \arrow[r, "S\lambda"'] & S^2GA &
\end{tikzcd}$$

is a $\mathcal{Y}$-double pullback for all A in $\boldsymbol{X}$.

Proof. $(\Rightarrow)$ The pullback of ϵ_A along itself is taken by the composite $U_S G_\lambda$

to the following $\mathcal{Y}$-pullback:

$$\begin{tikzcd}
SGP_TA \arrow[r, "SG\mu_0"] \arrow[d, "SG\mu_1"'] & SGT^2A \arrow[r, "S\lambda"] & S^2GTA \arrow[r, "\nu"] \arrow[d, "S^2\lambda"'] \arrow[dr, phantom, "\lrcorner", very near start] & SGTA \arrow[d, "S\lambda"] \\
SGT^2A \arrow[d, "S\lambda"'] & SP_SGA \arrow[r, "S\nu_0"] \arrow[d, "S\nu_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & S^3GA \arrow[r, "\nu"] & S^2GA \arrow[dd, "\nu"] \\
S^2GTA \arrow[r, "S^2\lambda"] \arrow[d, "\nu"'] \arrow[dr, phantom, "\lrcorner", very near start] & S^3GA \arrow[d, "\nu"] & & \\
SGTA \arrow[r, "S\lambda"'] & S^2GA \arrow[rr, "\nu"'] & & SGA
\end{tikzcd}$$

Since S reflects cover-pullbacks, the diagram in question is a $\mathcal{Y}$-double pullback.

($\Leftarrow$) If $h; f = k; g$ is an $\mathcal{X}_T$-pullback then as U_T preserves and T reflects cover pullbacks the following is an $\mathcal{X}$-double pullback:

$$\begin{tikzcd}
Z \arrow[rr, "h"] \arrow[dd, "k"'] & & TA \arrow[d, "Tf"] \\
& P_TC \arrow[r, "\mu_0"] \arrow[d, "\mu_1"'] & T^2C \\
TB \arrow[r, "Tg"'] & T^2C &
\end{tikzcd}$$

Let y be the induced $\mathcal{Y}$-map the the double pullback (p, q) inscribed in the following:

$$\begin{tikzcd}
GZ \arrow[rr, "Gh"] \arrow[dd, "Gk"'] & & GTA \arrow[r, "\lambda"] \arrow[d, "GTf"'] & SGA \arrow[d, "SGf"] \\
& GP_TC \arrow[r, "G\mu_0"] \arrow[d, "G\mu_1"'] & GT^2C \arrow[r, "\lambda"] & SGTC \arrow[d, "S\lambda"] \\
GTB \arrow[r, "GTg"] \arrow[d, "\lambda"'] & GT^2C \arrow[d, "\lambda"] & P_SGC \arrow[r, "\nu_0"] \arrow[d, "\nu_1"'] & S^2GC \\
SGB \arrow[r, "SGg"'] & SGTC \arrow[r, "S\lambda"'] & S^2GC &
\end{tikzcd}$$

The Sy is the induced $\mathcal{Y}$-map to the pullback inscribe in the following:

$$
\begin{tikzcd}
SGZ \arrow[rrr, "SGh"] \arrow[dr, "Sy"] \arrow[dd, "SGk"'] & & & SGTA \arrow[r, "S\lambda"] & S^2GA \arrow[r, "\nu"] \arrow[d, "S^2Gf"'] \arrow[dr, phantom, "\lrcorner", very near start] & SGA \arrow[d, "SGf"] \\
& SP \arrow[urrr, "Sp"] \arrow[dl, "Sq"] \arrow[u, phantom, "\lrcorner", very near start] & & & S^2GTC \arrow[r, "\nu"] \arrow[d, "S^2\lambda"'] \arrow[dr, phantom, "\lrcorner", very near start] & SGTC \arrow[d, "S\lambda"] \\
SGTB \arrow[d, "S\lambda"'] & & & SP_SGC \arrow[r, "S\nu_0"] \arrow[d, "S\nu_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & S^3GC \arrow[r, "\nu"] & S^2GC \arrow[dd, "\nu"] \\
S^2GB \arrow[rr, "S^2Gg"] \arrow[d, "\nu"'] \arrow[drr, phantom, "\lrcorner", very near start] & & S^2GTC \arrow[r, "S^2\lambda"] \arrow[d, "\nu"'] \arrow[dr, phantom, "\lrcorner", very near start] & S^3GC \arrow[d, "\nu"'] & & \\
SGB \arrow[rr, "SGg"'] & & SGTC \arrow[r, "S\lambda"'] & S^2GC \arrow[rr, "\nu"'] & & SGC
\end{tikzcd}
$$

Note that $p; G_\lambda f = q; G_\lambda g$ is a pullback in $\mathbf{Y}_S$; as the following equations hold in $\mathbf{Y}$,

$$
\begin{aligned}
&Gh; \lambda = y; p = y; \iota; Sp; \nu \\
&Gk; \lambda = y; q = y; \iota; Sq; \nu
\end{aligned}
$$

the image of $h; f = k; g$ under G_λ is a $\mathcal{Y}_S$-pullback. $\square$

Unfortunately, rather few examples of distributions involving the exception monad are cover-stable. In particular, the distributions for $+$ and $\times$ are neither stable nor $\mathcal{R}$-stable. To see why, suppose T and S are "exception" monads in that the stable monad structure comes from twist maps t and s respectively. Then λ is a cover-stable distribution for G whenever (1) and (2) below are cover-pullbacks:

$$
\begin{tikzcd}
GT^3A \arrow[rr, "Gt"] \arrow[ddd, "GT\mu"'] \arrow[ddddr, "\lambda"] & & GT^3A \arrow[rr, "GT\mu"] \arrow[drrr, "\lambda"'] & & GT^2A \arrow[rr, "\lambda"] & & SGTA \arrow[dddd, "S\lambda"] \\
& & & & & SGT^2A \arrow[ur, "SG\mu"'] \arrow[d, "S\lambda"] & \\
& & (1) & & & S^2GTA \arrow[dd, "S^2\lambda"] \arrow[r, phantom, "(2)"] & \\
GT^2A \arrow[dd, "\lambda"'] & & & & & & \\
& SGT^2A \arrow[r, "S\lambda"] \arrow[dl, "SG\mu"] & S^2GTA \arrow[r, "S^2\lambda"] & S^3GA \arrow[rr, "s"] \arrow[d, "S\nu"] & & S^3GA \arrow[r, "S\nu"'] & S^2GA \\
SGTA \arrow[rrr, "S\lambda"'] & & & S^2GA & & &
\end{tikzcd}
$$

For both $+$ and $\times$ (2) is pullback, but (1) is neither a pullback nor an $\mathcal{R}$-pullback as the inscribed pullback is "larger".

Example 3.18 *In a lextensive category, τ^+ (of example 3.1) is a stable distribution for $(\_{+}B)$ over the exception monad $(\_{+}X)$.*

Proof. Since τ is an isomorphism, (2) is a pullback and (1) will be a pullback provided it commutes. This is given by coherence for symmetric monoidal categories:

$$\begin{tikzcd}
A+W+X+Y+Z \arrow[r, "\tau+1"] \arrow[dd, "\tau"] & A+W+Y+X+Z \arrow[rr, "\tau"] & & A+W+Y+Z+X \arrow[d, "\tau+1"] \\
 & & & A+W+Z+Y+X \arrow[d, "\tau+1+1"] \\
A+W+X+Z+Y \arrow[r, "\tau+1"'] & A+W+Z+X+Y \arrow[r, "\tau+1+1"'] & A+Z+W+X+Y \arrow[r, "\tau"'] & A+Z+W+Y+X
\end{tikzcd}$$

□

Example 3.19 Consequently, the distribution for the delay monad functor (and the unit delay functor) on $Tran(\mathbf{C})$ is cover-stable. □

As monad actions will provide another means of obtaining cover systems on Kleisli categories, we note the following instance of proposition 3.17.

Corollary 3.20 *If $\alpha : GT \Longrightarrow G$ is a T-action for $G : (\boldsymbol{X}, \mathcal{X}) \longrightarrow (\boldsymbol{Y}, \mathcal{Y})$ then $G_\theta : (\boldsymbol{X}_T, \mathcal{X}_T) \longrightarrow (\boldsymbol{Y}, \mathcal{Y})$ if and only if the following is a $\mathcal{Y}$-pullback for all A in $\boldsymbol{X}$:*

$$\begin{tikzcd}
GP_TA \arrow[r, "G\mu_0"] \arrow[d, "G\mu_1"'] & GT^2A \arrow[r, "\theta"] & GTA \arrow[dd, "\theta"] \\
GT^2A \arrow[d, "\theta"'] & & \\
GTA \arrow[rr, "\theta"'] & & GA
\end{tikzcd}$$

Obvious examples are again given by identity transformations and multiplications for cover-stable monads.

Example 3.21 $\theta : (\_ + 1)^* \Longrightarrow (\_)^*$ *is a stable distribution, and thus* $(\_)^*_\theta$: $\boldsymbol{Set}_{(\_+1)} \longrightarrow \boldsymbol{Set}$ *is a stable functor.*

Proof. The square required to be a pullback is as follows:

$$\begin{tikzcd}
(A+1+1+1)^* \arrow[r, dash, "s^*"] \arrow[d, "(\mu+1)^*"'] & (A+1+1+1)^* \arrow[r, "(\mu+1)^*"] & (A+1+1)^* \arrow[r, "\theta"] & (A+1)^* \arrow[dd, "\theta"] \\
(A+1+1)^* \arrow[d, "\theta"'] & & & \\
(A+1)^* \arrow[rrr, "\theta"'] & & & A^*
\end{tikzcd}$$

It is easily seen to commute as each route simply strips the three distinct exceptions from each element of A+1+1+1. To see that it is a pullback, let h

be the map to the pullback Q of θ_A and θ_A. Define $k : Q \longrightarrow A{+}1{+}1{+}1$ as follows:

$$\begin{aligned}
k([],[]) &= [] \\
k(* :: \ell, []) &= *_1 :: k(\ell, []) \\
k(* :: \ell, a :: m) &= *_1 :: k(\ell, a :: m) \\
k([], * :: m) &= *_2 :: k([], m) \\
k(a :: \ell, * :: m) &= *_2 :: k(a :: \ell, m) \\
k(* :: \ell, * :: m) &= *_3 :: k(\ell, m) \\
k(a :: \ell, a :: m) &= a :: k(\ell, m)
\end{aligned}$$

where $a \in \Sigma_A$. An induction on the structure of Q shows that k is the inverse of h. $\Box$

It is then an easy consequence that

Example 3.22 *$\theta : MD \Longrightarrow M$ is a stable distribution, and thus M_θ : $\boldsymbol{Tran}_D \longrightarrow \boldsymbol{Tran}$ is a stable functor.*

Proof. Let h be the map to the pullback Q of θ and θ, and define $k : Q \longrightarrow MD^3A$ such that for states $(s, s) \mapsto s$ and for labels $(\ell, m) \mapsto \theta(\ell, m)$. To see that k is the inverse of h it suffices to check that k is a map. This is seen by induction on the structure of the labels of Q. $\Box$

We now turn to lifting the natural transformations of <u>**Cov**</u>. Let G and H be functors $(\mathbf{X}, \mathcal{X}) \longrightarrow (\mathbf{Y}, \mathcal{Y})$ with cover-stable distributions λ and κ, respectively, and $\alpha : F \Longrightarrow G$:

Proposition 3.23 *$\tilde{\alpha} : F_\lambda \Longrightarrow G_\kappa : (\boldsymbol{X}_T, \mathcal{X}_T) \longrightarrow (\boldsymbol{Y}_S, \mathcal{Y}_S)$ if and only if the following is a $\mathcal{Y}$-pullback for all A in $\boldsymbol{X}$.*

$$\begin{tikzcd}
GTA \arrow[r, "\alpha_{TA}"] \arrow[d, "\lambda_A"'] & HTA \arrow[d, "\kappa_A"] \\
SGA \arrow[r, "S\alpha_A"'] & SHA
\end{tikzcd}$$

Proof. $(\Rightarrow)$ The naturality square of α associated with ϵ_A in $\mathbf{Y}_S$ is taken by U_S to the following $\mathcal{Y}$-pullback:

$$\begin{tikzcd}
SGTA \arrow[r, "S\lambda"] \arrow[d, "S\alpha"'] \arrow[dr, phantom, "(*)"] & S^2GA \arrow[r, "\nu"] \arrow[d, "S^2\alpha"'] \arrow[dr, phantom, "\lrcorner", very near start] & SGA \arrow[d, "S\alpha"] \\
SHTA \arrow[r, "S\kappa"'] & S^2HA \arrow[r, "\nu"'] & SHA
\end{tikzcd}$$

As ν is cartesian, (*) is a $\mathcal{Y}$-pullback and thus the required square is a $\mathcal{Y}$-pullback as S reflects $\mathcal{Y}$-pullbacks.

($\Leftarrow$) Let y be the $\mathcal{Y}$-map induced to the pullback (p,q) inscribed in:

$$\begin{tikzcd}
GA \arrow[r, "Gh"] \arrow[d, "\alpha"'] & GTB \arrow[r, "\lambda"] \arrow[d, "\alpha"'] & SGB \arrow[d, "S\alpha"] \\
HA \arrow[r, "Hh"'] & HTB \arrow[r, "\kappa"'] & SHB
\end{tikzcd}$$

Then Sy serves as the map induced to the pullback inscribed in the following,

$$\begin{tikzcd}
SGA \arrow[r, "SGh"] \arrow[dr, "Sy"] \arrow[dd, "S\alpha"'] & SGTB \arrow[r, "S\lambda"] & S^2GB \arrow[r, "\nu"] \arrow[dd, "S^2\alpha"'] \arrow[ddr, phantom, "\lrcorner"] & SGB \arrow[dd, "S\alpha"] \\
 & \cdot \arrow[ur, "Sp"'] \arrow[dl, "Sq"] & & \\
SHA \arrow[r, "SHh"'] & SHTB \arrow[r, "S\kappa"'] & S^2HB \arrow[r, "\nu"'] & SHB
\end{tikzcd}$$

which implies $\tilde{\alpha}$ is $\mathcal{Y}_S$-cartesian. $\square$

Example 3.24 *The unit and multiplication of the exception monad respect the associated distributions, and thus one exception monad can be lifted over another.*

Proof. The required pullbacks are constructed below:

$$\begin{tikzcd}
A+X \arrow[r, "b_0"] \arrow[d, equal] \arrow[dr, phantom, "\lrcorner", very near start] & A+X+B \arrow[d, "\tau"] \\
A+X \arrow[r, "b_0+1"'] & A+B+X
\end{tikzcd}
\qquad
\begin{tikzcd}
A+X+B+B \arrow[r, "\mu"] \arrow[d, "\tau+1"'] \arrow[ddr, phantom, "\lrcorner", very near start] & A+X+B \arrow[dd, "\tau"] \\
A+B+X+B \arrow[d, "\tau"'] & \\
A+B+B+X \arrow[r, "\mu+1"'] & A+B+X
\end{tikzcd}$$

$\square$

3.4 2-categorical aspects

One may ask whether the Kleisli construction (in the 2-categorical sense) exists in **Cov**. Unfortunately, the answer is no. Although the functors F and U are cover-stable and the unit η is cartesian, the counit ϵ is not cartesian. For ϵ to be cartesian would require the image under U of a naturality square $\epsilon_A; f = FUf; \epsilon_B$ to be a pullback:

$$\begin{tikzcd}
T^2A \arrow[r, "T^2f"] \arrow[d, "\mu"'] & T^3B \arrow[r, "T\mu"] & T^2B \arrow[d, "\mu"] \\
TA \arrow[r, "Tf"'] & T^2B \arrow[r, "\mu"'] & TB
\end{tikzcd}$$

Given proposition 3.8, this is certainly not the case.

Although it is not the Kleisli construction in $\underline{\mathbf{Cov}}$, the construction of asynchronous model categories can be characterized by an analogue of theorem 3.7. Let $CSDist(\underline{\mathbf{X}})$ be the subcategory of $Dist(\underline{\mathbf{X}})$ whose 0-cells are cover-stable monads, 1-cells are cover-stable distributions and 2-cells respect the distributions in the sense of proposition 3.23.

Proposition 3.25 *$CSDist(\underline{\boldsymbol{X}})$ is a 2-category.*

Proof. It is sufficient to show that composition of the 1-cells is well-defined. Suppose $F_\lambda : (\mathbf{X}_T, \mathcal{X}_T) \longrightarrow (\mathbf{Y}_S, \mathcal{Y}_S)$ and $G_\kappa : (\mathbf{Y}_S, \mathcal{Y}S) \longrightarrow (\mathbf{Z}_R, \mathcal{Z}_R)$ are cover-stable functors. The following shows that $G\lambda;\kappa$ is a cover-stable distribution for the composite GF:

$$\begin{tikzcd}
\cdot \arrow[r, "GF\mu_0"] \arrow[d, "GF\mu_1"'] & \cdot \arrow[r, "G\lambda"] & \cdot \arrow[r, "\kappa"] \arrow[d, "GS\lambda"'] & \cdot \arrow[d, "RG\lambda"] \\
\cdot \arrow[d, "G\lambda"'] & \cdot \arrow[r, "G\nu_0"] \arrow[d, "G\nu_1"'] & \cdot \arrow[r, "\kappa"] & \cdot \arrow[d, "R\kappa"] \\
\cdot \arrow[r, "GS\lambda"] \arrow[d, "\kappa"'] & \cdot \arrow[d, "\kappa"'] & \cdot \arrow[r, "\sigma_0"'] \arrow[d, "\sigma_1"] & \cdot \\
\cdot \arrow[r, "RG\lambda"'] & \cdot \arrow[r, "R\kappa"'] & \cdot &
\end{tikzcd}$$

□

Let $CSLift(\underline{\mathbf{X}})$ be the subcategory of $Lift(\underline{\mathbf{X}})$ whose 0-cells are those monads which are cover-stable, 1-cells are liftings whose components are cover-stable, and 2-cells are pillows whose components are cover-cartesian. We can now state the analogue of theorem 3.7 which characterizes the construction of asynchronous model categories presented in this section.

Theorem 3.26 *$CSDist(\underline{\boldsymbol{X}})$ is isomorphic to $CSLift(\underline{\boldsymbol{X}})$.*

Proof. Any cover-stable lifting corresponds to a distribution, and thus the result is immediate from propositions 3.17 and 3.23. □

4 Asynchronous Processes

This section summarizes the construction of **ASProc** using the techniques of the previous section. First we examine the notions of asynchronous composition and tensor product given by the pullback and product in the Kleisli category. We then show how weak bisimulation equivalence arises from a stable functor from the asynchronous to the synchronous model category. This functor also provides an embedding of **ASProc** into **SProc**.

The Kleisli category $\mathbf{Tran}_D$, as shown in the previous section, has finite limits and thus admits for any cover system $\mathcal{X}$ the construction of a compact-closed process category:

$$\mathbf{ASProc}(\mathcal{X}) \stackrel{def}{=} Proc(\mathbf{Tran}_D, \mathcal{X})$$

Given maps $g : P \longrightarrow B$ and $h : Q \longrightarrow B$ in $\mathbf{Tran}_D$, their pullback consists of the following transition system

$$\begin{aligned} &\{(s,t)\xrightarrow{x,y}(s',t') \mid s\xrightarrow{x}s',\ t\xrightarrow{y}t',\ gx = hy\} \\ &\cup \{(s,t)\xrightarrow{x,*}(s',t) \mid s\xrightarrow{x}s',\ gx = *\} \\ &\cup \{(s,t)\xrightarrow{*,y}(s,t') \mid t\xrightarrow{y}t',\ hy = *\} \end{aligned}$$

together with the obvious projections. Thus, at their shared interface, the actions of a composite process $P;Q$ correspond either to synchronization of each process on a common action, or to silent actions by either process independently. The product of transition systems A and B is given by the transition system

$$\begin{aligned} &\{(s,t)\xrightarrow{x,y}(s',t') \mid s\xrightarrow{x}s',\ t\xrightarrow{y}t'\} \\ &\cup \{(s,t)\xrightarrow{x,*}(s',t) \mid s\xrightarrow{x}s'\} \\ &\cup \{(s,t)\xrightarrow{*,y}(s,t') \mid t\xrightarrow{y}t'\} \end{aligned}$$

which allows transitions by both components simultaneously as well as independent transitions by either component. Thus the general machinery of the process construction yields exactly what one expects for composition and tensor product of asynchronous processes.

The canonical cover system $\exists^{\mathcal{R}}_D$ given by the free functor of the Kleisli construction serves to lift the strong bisimulation equivalence of **SProc** into the asynchronous setting. Under this equivalence, asynchronous processes $A \longleftrightarrow B$ are related exactly when related as processes $DA \longleftrightarrow DB$ of **SProc**. Thus processes such as x and $x.\tau$ with different internal structure are distinguished. One can use the factorization system of the 2-category $\underline{\mathbf{Cov}}$ to understand how weaker cover systems are obtained: A monad action $\alpha : GD \Longrightarrow G$, as in corollary 3.20, induces a cover system $\mathcal{X}_\alpha \equiv G_\alpha^{-1}(\mathcal{X})$ on the asynchronous model category. The monad multiplication $\mu : DD \Longrightarrow D$ is a D-action, giving rise to the underlying functor U, and induces a cover system which is only slightly weaker than $\exists^{\mathcal{R}}_D$. It can equate processes which differ in their internal actions, but still requires related processes to be strongly bisimilar with respect to visible actions. So although processes such as x and $x.\tau$ are equated, processes such as $x.y$ and $x.\tau.y$ are not.

Weak bisimulation equivalence is obtained from the action $\theta : MD \Longrightarrow M$: a map $f : A \longrightarrow B$ of $Tran_D$ is in $\exists^{\mathcal{R}}_\theta$ if and only if $Mf;\theta : MA \longrightarrow MB$ is in $\exists^{\mathcal{R}}$. To see how the induced equivalence on processes corresponds to weak bisimulation, note that any span $A \longleftrightarrow B$ is equivalently specified as a map $P \longrightarrow A{\times}B$ of $Tran_D$. Two such spans $f : P \longrightarrow A{\times}B$ and $g : Q \longrightarrow A{\times}B$

are weakly bisimilar in the presence of a symmetric relation S on the states of P and Q for which $(p,q) \in S$ and $p \xrightarrow{\ell} p' \in MP$ implies that there exists q' and m such that $q \xrightarrow{m} q' \in MQ$ and $g(\theta(m)) = f(\theta(\ell))$.

Proposition 4.1 *Two spans $A \longleftrightarrow B$ are $\exists^{\mathcal{R}}_{\theta}$-bisimilar if and only if they are weakly bisimilar.*

Proof. The implication is easily seen. To see the converse, suppose $\mathcal{S}$ is a weak bisimulation of the spans. Form a transition system R whose states are $\{(p,q) \in \mathcal{S} \mid f(p) = g(q)\}$, labels are $\{(x,y) \mid f(x) = g(y)\}$ and transitions are $\{(p,q) \xrightarrow{(x,y)} (p',q') \mid p \xrightarrow{x} p' \wedge q \xrightarrow{y} q'\}$. Note that R is a subobject of the pullback of f and g which may ignore unreachable states. It is straightforward to shown that the projections from R are $\exists^{\mathcal{R}}_{\theta}$-maps. $\square$

Note that the formulation of weak bisimulation in this setting corresponds very closely to the first definition given by Milner in [Mil83] rather than the description (given there as proposition 8.4) which has now become standard [Mil89].

As $M_{\theta} : (\mathbf{Tran}_D, \exists^{\mathcal{R}}_{\theta}) \longrightarrow (\mathbf{Tran}, \exists^{\mathcal{R}})$ preserves finite limits, it induces a functor $Proc(M_{\theta}) : \mathbf{ASProc}(\exists^{\mathcal{R}}_{\theta}) \longrightarrow \mathbf{SProc}$ which preserves the tensor product. Although it is not the generally case that a faithful functor G which reflects covers yields a faithful functor $Proc(G)$, it is the case that $Proc(M_{\theta})$ is faithful.

5 Conclusion

The motivation of this work was to understand the construction of asynchronous processes using the categorical formulation of bisimulation advocated in [JNW93] and the view of processes proposed in [CS94a]. Once the technical dust settles, what emerges is a direct categorical interpretation of Milner's original description of asynchrony.

In the view of process algebra provided by **SProc** and **ASProc**, asynchrony arises through a well known categorical construction: a distributive law. The theory developed in the paper is quite general and suggests that one should look for such structure in other settings. We are already aware that these techniques can be used to describe the game theoretic interaction categories of [AJ92] as well as examples not considered in the literature, such as a non-interleaving version of **SProc** built upon event structures or transition systems with independence [JNW93].

References

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

[Abr94] Abramsky, S., *Interaction Categories and communicating sequential processes*, In A. W. Roscoe (ed), *A Classical Mind: Essays in Honour of C. A. R. Hoare*, Prentice Hall International, 1994.

[AGN93] Abramsky, S., S. Gay and R. Nagarajan, *Constructing and verifying typed processes.* Unpublished, 1993.

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

[AJM94] Abramsky, S., Jagadeesan, R., and Malacaria, P., *Full abstraction for PCF (extended abstract)*, In M. Hagiya and J. C. Mitchel (eds), *Theoretical Aspects of Computer Software. International Symposium TACS'94*, Lecture Notes in Computer Science 789, Springer-Verlag, 1994.

[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.

[CS94a] Cockett J.R.B. and D.A. Spooner, *SProc Categorically.* Proceedings of CONCUR '94, Springer Verlag, 1994.

[CS94b] Cockett J.R.B. and D.A. Spooner, *Constructing Process Categories.* Available by ftp as `/pub/papers/categories/interaction/cpc.ps.Z` on `ftp.cpsc.ucalgary.ca`.

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

[Kel91] Kelly, G.M., *On clubs and data-type constructors.* Applications of Categories in Computer Science, LMSLNS 177, Cambridge University Press, 1992.

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

[Mil89] Milner, R., **Communication and Concurrency**, Prentice Hall, 1989.

[Mul93] Mulry, P.S., *Lifting Theorems for Kleisli Categories.* Mathematical Foundations of Programming Semantics, LNCS 802, Springer Verlag, 1994.