Familial monads and structural operational semantics

Tom Hirschowitz

Familial monads and structural operational semantics

TOM HIRSCHOWITZ, Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, LAMA, France

We propose an abstract framework for structural operational semantics, in which we prove that under suitable hypotheses bisimilarity is a congruence. We demonstrate the flexibility of our approach by comparing three variants of the π-calculus and recovering known congruence (and non-congruence) results. We then refine the framework to account for soundness of bisimulation up to context, an efficient method for reducing the size of bisimulation relations. We recover soundness of bisimulation up to non-input context in π-calculus, and obtain a new result, soundness of wide open bisimulation up to context.

1 INTRODUCTION

Motivation

Structural operational semantics is a method for specifying the semantics of programming languages by induction on their syntax, introduced by Plotkin around 1980 [12]. Induction here allows one to describe the behaviour of a program in terms of its components.

But this sort of compositionality is not enough: it is often desirable for compositionality to transfer to equational reasoning. Specifically, in structural operational semantics, a programming language is described as a *labelled transition system*, a particular kind of labelled graph, where labels describe the kind of interaction that the considered program is having with its environment. There is a canonical behavioural equivalence between programs in such a labelled transition system: *bisimilarity*. (Well, admittedly, there are quite a few reasonable alternatives to bisimilarity, but let us stick to it as the most widely used.) By compositionality transferring to equational reasoning, we mean that one generally hopes that bisimilarity is a congruence, i.e., that replacing a program fragment with a bisimilar one does not change the bisimilarity class. A bit more formally, if two program fragments P and Q are bisimilar, which we denote by $P \sim Q$, then for any context C, we should have $C[P] \sim C[Q]$.

One may hope that specifying the labelled transition system compositionally ensures that the obtained bisimilarity is a congruence. If compositionality is meant in the sense of structural operational semantics, this is famously known not to hold, in particular for the π-calculus [14, Section 2.2.1]. This problem led to a rich variety of syntactic *formats* [11], which put some constraints on structural operational semantics specifications, offering in exchange useful guarantees like congruence of bisimilarity.

However, formats have grown to be very diverse, and also very low-level, hence they hardly offer a satisfactory explanation of when and why bisimilarity is a congruence. The desire for a higher-level answer to this question has led to *functorial operational semantics* [8, 16], in which structural operational semantics specifications are recast as distributive laws of a comonad over a monad. While deeply developed in the set-based case, functorial operational semantics does not scale well to more complex settings, notably in the presence of variable binding, where it becomes extremely complex, as, e.g., in Fiore and Staton's impressive work [4].

Staton himself argues that functorial operational semantics is too abstract [15], as "one must really squint hard to view a distributive law as a collection of rules, and to view the naturality conditions of a distributive law as the conditions imposed by a concrete rule format."

Author's address: Tom Hirschowitz, Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, LAMA, 73000, Chambéry, France, tom.hirschowitz@univ-smb.fr.

He goes on to develop a simpler approach which makes sense in any $\prod W$-pretopos, typically presheaf toposes. A great benefit is that variable binding is treated almost transparently. However, the notion of structural operational semantics specification remains fairly involved, and the range of potential applications seems limited. Indeed, in ten years of existence, to our knowledge, the format has not really been further developed.

Contributions

In this paper, we propose to attack the question of when bisimilarity is a congruence from the other end: instead of focusing on formats, we introduce a very simple, elementary setting, *transition categories*, in which we are able to prove that bisimilarity is a congruence. Or, rather, we try to define transition categories at just the right level of generality for the standard proof scheme to go through. In our setting, labelled transition systems are simply objects in a category $\mathscr{A}$, and bisimulations are defined by lifting, in the style of [7]. Furthermore, the structural operational semantics specifications are monads $\mathscr{T}$ on $\mathscr{A}$, which account for both syntax and transition rules. Thus, models of the specification are $\mathscr{T}$-algebras. Our main result (Corollary 3.12) then states that whenever

- the monad $\mathscr{T}$ satisfies a certain *familiality* [3, 17] property, and
- the considered $\mathscr{T}$-algebras, say X and Y, are *compositional*, in the sense that both structure maps $\mathscr{T}(X) \to X$ and $\mathscr{T}(Y) \to Y$ are functional bisimulations,

then bisimilarity is a congruence.

This leads us in Section 4 to an analysis of why standard bisimilarity is not a congruence in the π-calculus: familiality holds; what fails is that the structure map $\mathscr{T}(Pi) \to Pi$ is not a functional bisimulation. We interpret this as a lack of compositionality, hidden in the input axiom. We then consider two standard ways of working around this issue. First, we recast in our setting the standard result that bisimilarity is a *non-input congruence* [14]. Secondly, we consider a different presentation of π-calculus, in which bisimilarity is known as *wide open* bisimilarity [15], to which Corollary 3.12 applies.

Finally, we consider in Section 5 a notion of *pre-bisimulation*, which is perhaps closer to the standard notion than the bisimulations of Sections 3–4. Assuming a few additional hypotheses, we define a factorisation system in which factoring any pre-bisimulation gives rise to a bisimulation, thus exhibiting a tight relationship between the two notions. The advantage of pre-bisimulations is that they may be modified to a notion of *pre-bisimulation up to context*, which in examples corresponds to standard bisimulations up to context [14, Section 2.3.2]. Our main result is that any pre-bisimulation up to context embeds into some pre-bisimulation (and hence into some bisimulation), i.e., in standard terms, pre-bisimulation up to context is sound. This is to our knowledge the first general soundness result for bisimulation up to context covering calculi with variable binding.

Related work

Theorem 12 of Staton's paper is very close in spirit to our Corollary 3.12, but in a different axiomatic setting, without any link to familiality, and at a different level of generality – indeed, his functional bisimulations are members of an abstract class of *open* maps; ours are defined concretely through a lifting property, which is key in connecting preservation of functional bisimulations with familiality. Other related work extends the bialgebraic approach to deal with up-to techniques [2], in the set-based case. Finally, our notion of $\mathbf{T}_s$-familial monad is close in spirit to *cellular* analytic functors [5].

Plan

In Section 2, we explain in which sense structural operational semantics specifications are monads, and models are algebras. We proceed by example, considering combinatory logic and CCS. In Section 3, we introduce transition categories and prove our main result, together with a few corollaries. In Section 4, we analyse bisimilarity in the π-calculus through the lens of transition categories. In Section 5, we refine the framework to account for soundness of bisimulation up to context. Finally, we conclude in Section 6.

Notations and preliminaries

We assume basic familiarity with category theory [9]. Furthermore, we will use the notion of a *locally presentable* category [1], though for examples it should be enough to know that presheaf categories are locally presentable.

For any locally small category $\mathbb{C}$, we denote by $\widehat{\mathbb{C}}$ the category of *presheaves* on $\mathbb{C}$, i.e., contravariant functors to sets (**Set**) and natural transformations between them. For any $f : c \to c'$ in $\mathbb{C}$ and $X \in \widehat{\mathbb{C}}$ the action $X(f) : X(c') \to X(c)$ is denoted by $x \mapsto x \cdot f$. The Yoneda embedding is denoted by $\mathbf{y} : \mathbb{C} \to \widehat{\mathbb{C}}$, and often left implicit.

We denote by $el(X)$ the *category of elements* [10] of any presheaf X: it has as objects all pairs (c, x) with $x \in X(c)$, and as morphisms $(c, x) \to (c', x')$ all morphisms $f : c \to c'$ in $\mathbb{C}$ such that $x' \cdot f = x$. We denote the corresponding morphism by $f \upharpoonright x'$.

Furthermore, we often denote by n the finite set $\{1, ..., n\}$.

Finally, we briefly recall the basics of weak factorisation systems [13, Sections 11.1 and 11.2]. In any category $\mathscr{A}$, a *lifting problem* for morphisms f and g is any commuting square as the solid part of

$$\begin{array}{ccc} A & \longrightarrow & C \\ {\scriptstyle f}\downarrow & \nearrow & \downarrow{\scriptstyle g} \\ B & \longrightarrow & D. \end{array}$$

The lifting problem has a solution when there exists a lifting as shown (dashed) making both triangles commute.

Definition 1.1. Let $f \boxtimes g$ iff all lifting problems for f and g have at least one solution.

When $f \boxtimes g$, we say that g has the *right lifting property* w.r.t. f, or alternatively that f has the *left lifting property* w.r.t. g. We moreover let $f^{\boxtimes}$ denote the class of all morphisms that have the right lifting property w.r.t. f. Similarly, for any class $\mathscr{D}$ of morphisms, let $\mathscr{D}^{\boxtimes} = \bigcap_{f \in \mathscr{D}} f^{\boxtimes}$. We define ${}^{\boxtimes}g$ and ${}^{\boxtimes}\mathscr{D}$ symmetrically.

Definition 1.2. A *weak factorisation system* on a category $\mathscr{A}$ consists of two classes of maps $\mathscr{L}$ and $\mathscr{R}$ such that $\mathscr{L}^{\boxtimes} = \mathscr{R}$, $\mathscr{L} = {}^{\boxtimes}\mathscr{R}$, and every map $f : A \to B$ factors as $A \xrightarrow{l} C \xrightarrow{r} B$ with $l \in \mathscr{L}$ and $r \in \mathscr{R}$.

Let us recall some closure properties of $\mathscr{L}$ and $\mathscr{R}$.

Definition 1.3. For any ordinal γ, a *γ-sequence* is a cocontinuous functor from γ viewed as a category, to $\mathscr{A}$. A *transfinite composite* of any γ-sequence $F : \gamma \to \mathscr{A}$ is the component $\lambda_0 : \gamma(0) \to F^\infty$ of any colimiting cocone (where F^∞ is the considered colimit).

Definition 1.4. A *retract* of f: $X \to Y$ is any map $g : A \to B$ for which there exists a retraction $f \to g$ in the arrow category $\mathscr{A}^{\to}$, i.e., morphisms $g \xrightarrow{s} f \xrightarrow{r} g$ such that $r \circ s = id_g$.

Proposition 1.5 ([13, Lemma 11.1.4]). *For any factorisation system $(\mathscr{L}, \mathscr{R})$, $\mathscr{L}$ (resp. $\mathscr{R}$) contains all isomorphisms and is closed under composition, retracts, coproducts (resp. products) of arrows, and pushouts (resp. pullbacks). $\mathscr{L}$ is furthermore closed under transfinite composition.*

Cofibrantly generated weak factorisation systems are those generated from a set of maps by lifting. They enjoy an explicit characterisation of $\mathscr{L}$:

Proposition 1.6 ([13, Theorem 12.2.2 and Corollary 12.2.4]). *For any set $\mathscr{J}$ of maps in a locally presentable category, the classes $({}^{\boxslash}\mathscr{J})^{\boxslash}$ and $\mathscr{J}^{\boxslash}$ form a weak factorisation system, whose left class consists precisely of retracts of transfinite composites of pushouts of maps in $\mathscr{J}$.*

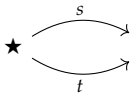
2 STRUCTURAL OPERATIONAL SEMANTICS SPECIFICATIONS AS MONADS

In this section, we explain how standard structural operational semantics, through inductive syntax and transition rules, may be formulated in the language of monads and their algebras, an observation that we would attribute to Staton [15]. The first step (Section 2.1) is to view labelled transition systems as objects in adequate presheaf categories, in which bisimulation may be defined by a lifting property, as in presheaf models [7]. Then (Section 2.2), we show by example how to define monads on such categories that correspond to structural operational semantics specifications. This allows us to abstractly define what it means for bisimilarity to be a congruence.

2.1 Labelled transition systems as objects in transition categories

In this section, we first explain how labelled transition systems may be viewed as presheaves and how bisimulation may be defined by lifting. We then abstract over what we did and define transition categories and bisimulation therein.

2.1.1 Labelled transition systems. The simplest kind of labelled transition system is the one with just one label: this is simply graphs (provided one accepts the slightly unusual generality of allowing distinct, parallel transitions between nodes). Thus, the first category of interest for us is **Gph**, the category of (directed, multi) graphs, viewed as presheaves over the category



.

If one needs to consider labels, it is often enough to introduce them as a particular graph, say A, so that labelled transition systems are graphs over A, i.e., morphisms $G \to A$ for some graph G. But as is well-known, the slice category $\mathbf{Gph}/A$ is equivalent to $\widehat{el(A)}$, presheaves over the category of elements of A.

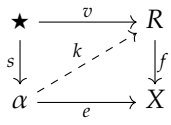
Our running example for this section will be CCS. The labels of CCS are elements of

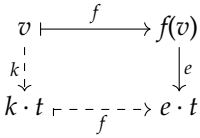
$$\{\tau\} \cup \bigcup_{a \in \mathscr{N}} \{a, \overline{a}\},$$

where $\mathscr{N}$ denotes a fixed, infinite set of *names*, for example the natural numbers $\mathbb{N}$. Viewing this as the edges of a one-vertex graph A, the relevant base category $\mathbb{C}_{CCS} := el(A)$ is the category freely generated by the graph with

- vertices $\star$ and τ, plus vertices a and $\overline{a}$ for all names a,
- edges $\star \xrightarrow{s} \alpha \xleftarrow{t} \star$ for all $\alpha \in \{\tau\} \cup \bigcup_{a \in \mathscr{N}} \{a, \overline{a}\}$.

2.1.2 Bisimulation. Presheaves on $\mathbb{C}_{CCS}$ may thus be viewed as labelled transition systems, and functional bisimulations may be defined by lifting: a morphism $f : R \to X$ in $\widehat{\mathbb{C}_{CCS}}$ is a functional bisimulation iff it has the right lifting property w.r.t. all maps of the form $s : \star \to \alpha$ for α a label. Indeed, a presheaf morphism is automatically a simulation (merely because edges are mapped to edges), and the lifting property says that for any commuting square of the form below left





there is a lifting k as shown that makes both triangles commute. By Yoneda, v denotes a vertex in R, e an α-transition in X, and commutation of the square says that $f(v)$ is the source of e. Existence of k then says that there is an α-transition k in R with source v, mapped by f to e, as above right, which is the standard definition of a functional bisimulation.

General (potentially non-functional) *bisimulations* may be defined as spans $X \overset{l}{\leftarrow} R \overset{r}{\to} Y$ where l and r are functional bisimulations. Such bisimulations may have non-monic pairing $R \to X \times Y$, but if we consider their epi-mono factorisation $R \overset{e}{\twoheadrightarrow} im(R) \overset{m}{\hookrightarrow} X \times Y$, we have:

Proposition 2.1. *The monic factor of any bisimulation is again a bisimulation.*

Proof. Consider any factorisation $m \circ e$ as above. Because $\star$ is representable, its covariant hom-functor preserves epis, i.e., if $f : X \to Y$ is epi, then by Yoneda so is the set map

$$\widehat{\mathbb{C}_{CCS}}(\star, X) \xrightarrow{\widehat{\mathbb{C}_{CCS}}(\star, f)} \widehat{\mathbb{C}_{CCS}}(\star, Y).$$

Thus every lifting problem for any $s : \star \to \alpha$ and m induces a lifting problem for s and the composite $m \circ e$. The latter has a lifting by hypothesis, which yields a lifting for m. □

Monic bisimulations are called *bisimulation relations*. Finally, we have:

Proposition 2.2. *Bisimulation relations are closed under union, and admit a maximum, called* bisimilarity*, which is terminal in the full subcategory of the slice $\widehat{\mathbb{C}_{CCS}}/X \times Y$ with objects all bisimulations.*

Proof. Presheaf categories are cocomplete so arbitrary (small) unions of subobjects exist. Furthermore, because $\star$ is representable, it is *tiny*, i.e., its covariant hom-functor preserves colimits. Thus, every lifting problem for any $s : \star \to \alpha$ and a given union $\bigcup_i R_i$ yields one for some R_i, which has a lifting by hypothesis. Finally, presheaf categories are well-powered, i.e., each object has only a set of subobjects. This set being closed under unions, it has a maximum element. □

2.1.3 Transition categories. Reflecting on what we just did, the main ingredients are a category equipped with a selection **T** of cospans, thought of as *transition types*, whose maps have domains in a certain set **P** of *process types*, and codomains in a certain set **L** of *label types*. We have furthermore used the fact that presheaf categories are cocomplete and well-powered, and that covariant hom-functors of process types preserve all colimits and epis. A reasonable abstraction of this is:

Definition 2.3. A *transition category* consists of a locally presentable category $\mathscr{A}$ equipped with

- two sets $\mathbf{P}$ and $\mathbf{L}$ of objects called *process types* and *label types*, respectively;
- a set $\mathbf{T}$ of cospans $P \xrightarrow{s} L \xleftarrow{t} Q$ called *transition types*, in which $P, Q \in \mathbf{P}$ and $L \in \mathbf{L}$,

such that covariant hom-functors of process types preserve all colimits and epis.

Notation 1. *We generally denote a transition category by just $\mathscr{A}$, leaving* $\mathbf{P}$, $\mathbf{L}$, *and* $\mathbf{T}$ *implicit.*

We may replay the above development in any transition category:

Definition 2.4. For any transition category $\mathscr{A}$, a morphism $f : R \to X$ is a *functional bisimulation* iff it is in $\mathbf{T}_s^{\boxslash}$, where $\mathbf{T}_s$ denotes the class of morphisms appearing as s in some transition type. Given any two objects X and Y, a *bisimulation* is a span $X \xleftarrow{s} R \xrightarrow{t} Y$ such that s and t are both functional bisimulations. When the associated pairing $R \to X \times Y$ is monic, we call R a *bisimulation relation*.

Lemma 2.5. *In any transition category $\mathscr{A}$, bisimulations are stable under images, i.e., if $r \colon R \to X \times Y$ is a bisimulation, then so is the right factor of its epi-mono factorisation $R \xrightarrow{e} im(R) \xrightarrow{m} X \times Y$.*

Proposition 2.6. *In any transition category, bisimulation relations $X \nrightarrow Y$ are stable under union, hence admit a maximum, called* bisimilarity *and denoted by $\sim_{X,Y}$, or simply $\sim$ when X and Y are clear from context. Bisimilarity is also a terminal object in the full subcategory of the slice $\mathscr{A}/X \times Y$ with objects all bisimulations.*

As expected we have:

Example 2.7 (Graphs). Graphs form a transition category with $\mathbf{P} = \{\mathbf{y}_\star\}$, $\mathbf{L} = \{\mathbf{y}_{[1]}\}$, and the cospan $\star \xrightarrow{s} [1] \xleftarrow{t} \star$ as unique transition type (omitting the Yoneda embedding).

Example 2.8 (CCS labels). We consider $\widehat{\mathbb{C}_{CCS}}$ as a transition category by taking $\star$ as only process type, all other objects as label types, and all cospans $\star \xrightarrow{s} \alpha \xleftarrow{t} \star$ as transition types.

Remark 1. *Everything in Section 3 would work with the slightly more general assumption that, without necessarily being locally presentable, $\mathscr{A}$ is regular, complete, and cocomplete, and furthermore all process types are small relative to $\mathbf{T}_s$-cell [6, Definition 2.1.3], where $\mathbf{T}_s$ denotes the class of morphisms appearing as s in some transition type. Section 5 would further require a similar smallness assumption but relative to copairings $[s, t] : P + Q \to L$, for $(s, t) \in \mathbf{T}$. This is all automatic when $\mathscr{A}$ is locally presentable.*

2.2 Structural operational semantics specifications as monads on transition categories

In the previous section, we have seen by example how labelled transition systems can be viewed as objects in adequate presheaf categories, and how bisimulation can be treated in this setting. We have then defined transition categories and bisimulation therein, which abstract over this situation. Let us now further explain how structural operational semantics specifications may be viewed as monads on transition categories, again starting with examples and then abstracting away. We first consider the definition of structural operational semantics specifications, and then investigate congruence of bisimilarity.

2.2.1 First example: combinatory logic. To get a feel for why monads on transition categories are relevant to operational semantics, let us consider the example of combinatory logic, viewed as a labelled transition system on just one label, i.e., a graph.

Definition 2.9. Let $\mathscr{T}_{CL}$ denote the functor on **Gph** mapping any graph $G = (V, E, s, t)$ to the one with

- as vertices all terms generated by the grammar

$$M, N ::= (\!|v|\!) \mid S \mid K \mid MN,$$

  where v ranges over V, S and K are constants, and MN stands for the application of a binary symbol (called "application") to M and N;
- as edges all proofs consructed inductively following the rules

$$\frac{e \in G(x,y)}{(\!|e|\!) : (\!|x|\!) \to (\!|y|\!)} \qquad \frac{}{s_{M,N,P} : SMNP \to (MN)(MP)} \qquad \frac{}{k_{M,N} : KMN \to M}$$

$$\frac{L : M \to M'}{LN : MN \to M'N} \qquad \frac{R : N \to N'}{MR : MN \to MN'}$$

  with the given sources and targets.

The action of $\mathscr{T}_{CL}$ on morphisms is by renaming vertices and edges according to the given graph morphism.

This functor is a monad with multiplication given by substitution, as inductively defined by

$$\begin{array}{rcl} \mu_\star(m) & = & m \\ \mu_\star(K) & = & K \\ \mu_\star(S) & = & S \\ \mu_\star(MN) & = & \mu_\star(M)\mu_\star(N) \end{array} \qquad \begin{array}{rcl} \mu_{[1]}(r) & = & r \\ \mu_{[1]}(k_{M,N}) & = & k_{\mu_\star(M),\mu_\star(N)} \\ \mu_{[1]}(s_{M,N,P}) & = & s_{\mu_\star(M),\mu_\star(N),\mu_\star(P)} \\ \mu_{[1]}(LN) & = & \mu_{[1]}(L)\mu_\star(N) \\ \mu_{[1]}(MR) & = & \mu_\star(M)\mu_{[1]}(R). \end{array}$$

Example 2.10. Let $M = (\!|M_1|\!)(\!|M_2|\!)$, with $M_1, M_2 \in \mathscr{T}_{CL}(H)(\star)$. Then, $\mu(M) = M_1M_2$. Similarly, for $k_{P,Q} \in \mathscr{T}_{CL}(H)(KPQ, P)$, we have $\mu((\!|k_{P,Q}|\!)M) = k_{P,Q}(M_1M_2)$.

Of interest to us is the free algebra $\mathscr{T}_{CL}(\varnothing)$, which is precisely combinatory logic.

2.2.2 Example with labels: CCS. In this section, as a second example useful for illustrating labels, let us deal with CCS. Recalling the base category $\mathbb{C}_{CCS}$ from Section 2.1.1:

Definition 2.11. Let $\mathscr{T}_{CCS}$ denote the functor on $\widehat{\mathbb{C}_{CCS}}$ such that

- $\mathscr{T}_{CCS}(G)(\star)$ is the set of CCS terms (simplified for expository purposes) with 'process constants' in $G(\star)$, i.e., generated by the grammar

$$P, Q ::= (\!|x|\!) \mid 0 \mid a.P \mid \overline{a}.P \mid (P|Q) \mid va.P$$

  with x ranging over $G(\star)$, *not* considered equivalent up to renaming of bound names in $va.P$ (α-equivalence is not necessary for CCS, and avoiding it allows our construction to work over sets; by contrast, α-equivalence is necessary in the π-calculus, which forces us to move to a more complex category);

- for all $\alpha \neq \star$, $\mathscr{T}_{CCS}(G)(\alpha)$ is the set of proofs of transitions $P \xrightarrow{\alpha} Q$, as inductively generated by the rules

$$\frac{e \in G(\alpha)}{(e) : (e \cdot s) \xrightarrow{\alpha} (e \cdot t)} \qquad \frac{}{\overline{a}.P \xrightarrow{\overline{a}} P} \qquad \frac{}{a.P \xrightarrow{a} P} \qquad \frac{p : P \xrightarrow{\overline{a}} P' \qquad q : Q \xrightarrow{a} Q'}{p \bowtie q : (P|Q) \xrightarrow{\tau} (P'|Q')}$$

$$\frac{p : P \xrightarrow{\alpha} P'}{(p|Q) : (P|Q) \xrightarrow{\alpha} (P'|Q)} \qquad \frac{q : Q \xrightarrow{\alpha} Q'}{(P|q) : (P|Q) \xrightarrow{\alpha} (P|Q')} \qquad \frac{p : P \xrightarrow{\alpha} Q}{va.p : va.P \xrightarrow{\alpha} va.Q} \; (a \notin \alpha).$$

Again substitution equips $\mathscr{T}_{CCS}$ with monad structure, and the free algebra $\mathscr{T}_{CCS}(\varnothing)$ is precisely (our simplified variant of) CCS.

2.2.3 Stating congruence of bisimilarity, abstractly. In the previous sections, we have seen two example monads on transition categories, which have the labelled transition systems for combinatory logic and CCS as their free algebras. Let us now review what it means for bisimilarity to be a congruence, and then again give an abstract account in the setting of algebras for a monad on some transition category.

Standardly, given a structural operational semantics specifications X, we say that bisimilarity is a congruence when for all multi-hole contexts C and pairs $(x_1, y_1), ..., (x_n, y_n)$ of pairwise bisimilar processes, we have $C[x_1, ..., x_n] \sim C[y_1, ..., y_n]$.

In the abstract setting, given any monad $\mathscr{T}$ on a transition category $\mathscr{A}$, we may mimick this definition, and even slightly generalise it by considering two different $\mathscr{T}$-algebras:

Definition 2.12. Given a monad $\mathscr{T}$ on a transition category $\mathscr{A}$, and $\mathscr{T}$-algebras $a \colon \mathscr{T}(X) \to X$ and $b \colon \mathscr{T}(Y) \to Y$, we say that bisimilarity (between X and Y) is a *congruence* when the image of

$$\mathscr{T}(\sim_{X,Y}) \to \mathscr{T}(X \times Y) \xrightarrow{\langle \mathscr{T}(\pi), \mathscr{T}(\pi') \rangle} \mathscr{T}(X) \times \mathscr{T}(Y) \xrightarrow{a \times b} X \times Y \tag{1}$$

embeds into $\sim_{X,Y}$.

3 CONGRUENCE OF BISIMILARITY

In the previous sections, we have defined (functional) bisimulation (relations) in transition categories and stated what it means for bisimilarity to be a congruence for a given monad. We now turn to proving it. We first briefly sketch the standard proof method, and then analyse each step in terms of monads on transition categories.

3.1 Standard proof method

The standard way to prove that bisimilarity is a congruence is to show that for any bisimulation R, $\mathscr{T}(R)$ is again a bisimulation, with projection to $X \times Y$ given by (1). This goes roughly as follows:

(i) Any given pair (x, y) in $X \times Y$ related for $\mathscr{T}(R)$ has the form

$$(a(C[x_1, ..., x_n]), b(C[y_1, ..., y_n])),$$

for some context C, with each $(x_i, y_i) \in R$.

(ii) By compositionality, any transition $x \xrightarrow{\alpha} x'$ may be written as $a(C[p_1, ..., p_n])$ with each p_i a transition $x_i \xrightarrow{\alpha_i} x_i'$ in X, for some label type α_i.

(iii) Because R is a bisimulation, we find $(q_1, ..., q_n)$ such that each $(p_i, q_i) \in R$, so that $C[(p_1, q_1), ..., (p_n, q_n)] \in \mathscr{T}(R)$. But the latter is projected to $a(C[p_1, ..., p_n])$ and $b(C[q_1, ..., q_n])$, respectively, which shows that Y matches the given X-transition.

We will show that, making adequate hypotheses on $\mathscr{T}$, X, and Y, this proof method may be reproduced in the abstract setting. We introduce familiality in Section 3.2, then compositionality in Section 3.3, and finally prove congruence of bisimilarity in Section 3.4.

3.2 Familiality

In order for Step *(i)* above to work in the abstract setting, we need to be able to decompose any $x \in \mathscr{T}(X)$ into a context C and a map $(x_1, ..., x_n) : n \to X$ from the holes of C to X. Categorically, thinking of maps $x : P \to \mathscr{T}(Z)$ as contexts of type P with holes filled with processes from Z, we need any such x to canonically factor as a generic context C followed by an assignment of its holes, say A to Z, as in

$$P \xrightarrow{C} \mathscr{T}(A) \xrightarrow{\mathscr{T}(h)} \mathscr{T}(Z). \tag{2}$$

Here, A represents the collection of holes of C. In *(i)* above, we would have $A = n$. Similarly, each transition $t : L \to \mathscr{T}(Z)$ should factor canonically as a generic transition context followed by an assignment of its holes to Z, as in

$$L \xrightarrow{D} \mathscr{T}(R) \xrightarrow{\mathscr{T}(k)} \mathscr{T}(Z).$$

In order to define the involved notion of canonicity, we appeal to Carboni and Johnstone's [3] *familially representable* functors, as developed by Weber [17] under the name of *parametric right adjoints*, here called *familial* functors following [5].

Definition 3.1. Given any functor $\mathscr{F} : \mathscr{A} \to \mathscr{X}$, a morphism $\xi : X \to \mathscr{F}(A)$ is *$\mathscr{F}$-generic*, or *generic* for short, when for all commuting squares of the form below

$$\begin{array}{ccc} X & \xrightarrow{\chi} & \mathscr{F}(B) \\ {\scriptstyle\xi}\downarrow & \overset{\mathscr{F}(h)}{\nearrow} & \downarrow{\scriptstyle\mathscr{F}(g)} \\ \mathscr{F}(A) & \xrightarrow[\mathscr{F}(f)]{} & \mathscr{F}(C) \end{array} \tag{3}$$

there exists a unique h such that $\mathscr{F}(h) \circ \xi = \chi$ and $g \circ h = f$.

Definition 3.2. A functor $\mathscr{F}$ is *familial* when any morphism $X \to \mathscr{F}(A)$ factors into some generic morphism followed by a free one, i.e., one of the form $\mathscr{F}(f)$. A monad $(\mathscr{T}, \eta, \mu)$ is *familial* when the underlying endofunctor is, and furthermore η and μ are *cartesian* natural transformations, i.e., all their naturality squares are pullbacks.

Remark 2. *By the pullback lemma, when the domain category has a terminal object, a natural transformation $\alpha : F \to G$ is cartesian iff its naturality squares of the form below are pullbacks.*

$$\begin{array}{ccc} FA & \xrightarrow{F(!)} & F1 \\ {\scriptstyle\alpha_A}\downarrow & & \downarrow{\scriptstyle\alpha_1} \\ GA & \xrightarrow[G(!)]{} & G1 \end{array}$$

Let us now show the monads $\mathscr{T}_{CL}$ and $\mathscr{T}_{CCS}$ of Sections 2.2.1 and 2.2.2 are familial. The main tool for doing so is the following characterisation of familial endofunctors on presheaf categories:

Lemma 3.3 ([17, Remark 2.12]). *An endofunctor $\mathscr{T}$ on $\widehat{\mathbb{C}}$ is familial iff there is a functor $E : el(\mathscr{T}(1)) \to \widehat{\mathbb{C}}$ and a natural isomorphism (in X and c):*

$$\mathscr{T}(X)(c) \cong \sum_{x \in \mathscr{T}(1)(c)} \widehat{\mathbb{C}}(E(c,x), X).$$

Proof sketch. In presheaf categories, familiality is equivalent to *pointwise familiality*, i.e., existence of a generic-free factorisation for all morphisms of the form $\mathbf{y}_c \to \mathscr{T}(X)$. But pointwise familiality is equivalent to existence of generic-free factorisations for all morphisms of the form $\mathbf{y}_c \to \mathscr{T}(1)$ (this follows directly by unique lifting). By Yoneda, each $x \in \mathscr{T}(X)(c)$ is thus determined up to isomorphism by the choice of a generic-free factorisation

$$\mathbf{y}_c \xrightarrow{\xi} \mathscr{T}(E(c, \mathscr{T}(!) \circ x)) \xrightarrow{\mathscr{T}(!)} \mathscr{T}(1) \qquad \text{of the composite} \qquad \mathbf{y}_c \xrightarrow{x} \mathscr{T}(X) \xrightarrow{\mathscr{T}(!)} \mathscr{T}(1),$$

together with a map $\varphi : E(c, \mathscr{T}(!) \circ x) \to X$. □

Proposition 3.4. *The monad $\mathscr{T}_{CL}$ is familial.*

Proof. By Lemma 3.3, it suffices to exhibit a functor $E : el(\mathscr{T}_{CL}(1)) \to \mathbf{Gph}$ from the category of elements of $\mathscr{T}_{CL}(1)$ to graphs, such that $\mathscr{T}_{CL}(Z)(c) \cong \sum_{x \in \mathscr{T}_{CL}(1)(c)} [E(c,x), Z]$, naturally in c and Z. Now, $\mathscr{T}_{CL}(1)(\star)$ consists of closed terms on a unique free variable, say $\perp$, and we define E to map any such term C to the discrete graph with n_C as vertices, where n_C is the number of occurrences of $\perp$ in C. Similarly, $\mathscr{T}_{CL}[1]$ consists of transition derivations on just one transition axiom, say $\perp\!\!\!\perp : \perp \to \perp$. On such derivations, we define E by induction:

$$\begin{array}{rcl} E(\perp\!\!\!\perp) & = & \mathbf{y}_{[1]} \\ E(s_{M,N,P}) & = & E(M) + E(N) + E(P) \\ E(k_{M,N}) & = & E(M) + E(N) \\ E(LN) & = & E(L) + E(N) \\ E(MR) & = & E(M) + E(R). \end{array}$$

Finally, we need to define E on s and t. We again proceed inductively:

- For the variable case, we have $E(\perp) = \star$ and $E(\perp\!\!\!\perp) = [1]$, so we let $E(s \uparrow \perp\!\!\!\perp)$ be just s and $E(t \uparrow \perp\!\!\!\perp)$ be t.
- For $s_{M,N,P}$, we have $E(MNP) \cong E(M) + E(N) + E(P)$ and $E((MN)(MP)) = E(M) + E(N) + E(M) + E(P)$, so we pick the obvious maps

  $$E(M) + E(N) + E(P) \quad \to \quad E(M) + E(N) + E(P) \quad \leftarrow \quad E(M) + E(N) + E(M) + E(P)$$

  for $E(s \uparrow s_{M,N,P})$ and $E(t \uparrow s_{M,N,P})$.
- We proceed similarly for $k_{M,N}$.
- For LN, we get inductively $E(M) \xrightarrow{E(s \uparrow L)} E(L) \xleftarrow{E(t \uparrow L)} E(M')$, which induces

  $$E(M) + E(N) \xrightarrow{E(s \uparrow L) + E(N)} E(L) + E(N) \xleftarrow{E(t \uparrow L) + E(N)} E(M') + E(N).$$

- We proceed similarly for MR. □

Proposition 3.5. *$\mathscr{T}_{CCS}$ is familial.*

Proof. Similar, using the fact that we do not mod out by α-equivalence. □

3.3 Compositionality

Now that Step *(i)* has been taken care of, we need to express compositionality in the abstract framework:

Definition 3.6. A $\mathscr{T}$-algebra $a\colon \mathscr{T}(X) \to X$ is *compositional* iff $a \in \mathbf{T}_s^{\boxtimes}$.

Concretely, this says that given any square

$$\begin{array}{ccc} P & \xrightarrow{p} & \mathscr{T}(X) \\ {\scriptstyle \mathbf{T}_s \ni s}\downarrow & \overset{k}{\nearrow} & \downarrow{\scriptstyle a} \\ L & \xrightarrow[r]{} & X \end{array}$$

there exists a lifting k making both triangles commute. Thinking as above of p as a process with variables in X, i.e., a context applied to some processes in X, of $a \circ p$ as its evaluation in X, and of r as a transition from $a \circ p$, this says that r may be decomposed as the evaluation $a \circ k$ of some transition context k with domain $k \circ s = p$.

Proposition 3.7. *The free $\mathscr{T}_{CL}$-algebra $\mathscr{T}_{CL}(\varnothing)$ is compositional.*

Proof. Compositionality amounts to showing that for any transition $E : \mu_\varnothing(M) \to Y$, there exists N and

$$R \in \mathscr{T}_{CL}(\mathscr{T}_{CL}(\varnothing))(M, N)$$

such that $E = \mu_\varnothing(R)$ (and $Y = \mu_\varnothing(N)$). Here, M is merely a term with variables in $\mathscr{T}_{CL}(\varnothing)$, otherwise said a pair of a context C with a certain number of free variables occurrences (ordered from left to right), say $x_1, ..., x_n$, together with n terms $M_1, ..., M_n$. We denote this by $M = C[M_1, ..., M_n]$. We proceed by induction on C and case analysis on E:

- If $C = (x_1)$ (and $n = 1$), then we have

  $$\mu_\varnothing(M) = \mu_\varnothing(C[M_1]) = \mu_\varnothing(M_1) = M_1.$$

  Thus, E is in fact a transition $M_1 \to Y$, so that taking $N = (Y)$ and $R = (E)$, we get $\mu_\varnothing(R) = E$ as desired.
- If $C = C_1C_2$, then we may divide $M_1, ..., M_n$ into

  $$M_1^1, ..., M_{n_1}^1, M_1^2, ..., M_{n_2}^2,$$

  so that putting $X = \mu_\varnothing(M)$, $P_i = C_i[M_1^i, ..., M_{n_i}^i]$, and $X_i = \mu_\varnothing(P_i)$, we have $X = X_1X_2$. We then proceed by case analysis.
  - If $E = E_1X_2$, for some $E_1 : X_1 \to Y_1$, then $Y = Y_1X_2$, and by induction hypothesis we get $R_1 : P_1 \to Q_1$ such that $\mu_\varnothing(R_1) = E_1$ and $\mu_\varnothing(Q_1) = Y_1$. Taking $R = R_1P_2$ thus yields $\mu_\varnothing(R) = E_1X_2$ as desired.
  - The case $E = X_1E_2$ is similar.
  - If $X_1 = KX'$ and $E = k_{X',X_2}$, then $C_1 = KC'$ and, putting $P' = C'[M_1^1, ..., M_{n_1}^1]$, we have $X' = \mu_\varnothing(P')$ and $Y = X'$. Taking $R = k_{P',P_2}$ then yields $\mu_\varnothing(R) = k_{X',X_2}$ as desired.
  - The case where $X_1 = SX'X''$ and $E = s_{X',X'',X_2}$ is similar. □

Proposition 3.8. *The $\mathscr{T}_{CCS}$-algebra $\mathscr{T}_{CCS}(\varnothing)$ is compositional.*

Proof. Similar to $\mathscr{T}_{CL}$, using the fact that, because we do not mod out by α-equivalence, the *va* operator may be considered as a unary operator indexed by names. □

3.4 Congruence of bisimilarity and $\mathbf{T}_s$-familiality

Before we state our congruence result, there is a final subtlety. Intuitively, familiality decomposes any process in $\mathscr{T}(X)$ into some context C, and a morphism from the arity of C to X. This induces a similar property on arrows, which we require to behave well. Specifically, we require the arity of all arrows $s \in \mathbf{T}_s$ to lie in ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$.

Definition 3.9. For any familial monad $(\mathscr{T}, \eta, \mu)$ on a transition category $\mathscr{A}$, the class $\overline{\mathbf{T}_s}$ consists of morphisms s' occurring in any generic-free factorisation of the form below, where $s \in \mathbf{T}_s$ and D is $\mathscr{T}$-generic.

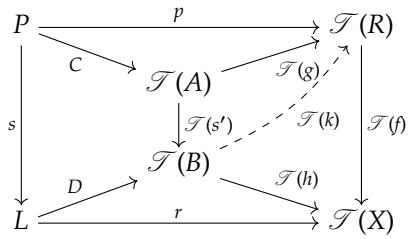
$$\begin{array}{ccc} P & \xrightarrow{\quad s \quad} & L \\ {\scriptstyle C}\downarrow & & \downarrow{\scriptstyle D} \\ \mathscr{T}(A) & \xrightarrow[\mathscr{T}(s')]{} & \mathscr{T}(R) \end{array}$$

We say that the monad is $\mathbf{T}_s$*-familial* when $\overline{\mathbf{T}_s} \subseteq {}^{\boxslash}(\mathbf{T}_s^{\boxslash})$.

Propositions 1.5 and 1.6 may help appreciate how general this can be in examples.

Theorem 3.10. *Consider any $\mathbf{T}_s$-familial monad $\mathscr{T}$ on some transition category $\mathscr{A}$, together with a compositional algebra (X, a). If $f : R \to X$ is a functional bisimulation, then so is $\mathscr{T}(R) \xrightarrow{\mathscr{T}(f)} \mathscr{T}(X) \xrightarrow{a} X$.*

Proof. By Proposition 1.5, functional bisimulations are closed under composition, so it suffices to show that $\mathscr{T}(f)$ is one. We thus need to construct a lifting for any commuting square as the exterior of



We use familiality of $\mathscr{T}$ to factor p as $\mathscr{T}(g) \circ C$ and r as $\mathscr{T}(h) \circ D$ with C and D generic. Genericity of C then yields $s' : A \to B$ as shown, which is in ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$ by $\mathbf{T}_s$-familiality, so we obtain a lifting k as shown, and $\mathscr{T}(k) \circ D$ is a lifting for the whole diagram. □

Corollary 3.11. *For any bisimulation $R : X \nrightarrow Y$ between compositional algebras for a $\mathbf{T}_s$-familial monad $\mathscr{T}$, the induced span $\mathscr{T}(R) : X \nrightarrow Y$ is a bisimulation.*

Corollary 3.12. *Between any two compositional algebras for a $\mathbf{T}_s$-familial monad $\mathscr{T}$, bisimilarity is a congruence.*

In most examples, the considered algebras are both the free algebra $\mathscr{T}(\varnothing)$. They are thus covered by the following

Corollary 3.13. *Consider any $\mathbf{T}_s$-familial monad $\mathscr{T}$ on some transition category $\mathscr{A}$ such that $\mu_1 : \mathscr{T}^2(1) \to \mathscr{T}(1)$ is a functional bisimulation. Then for all X, if $f : R \to \mathscr{T}(X)$ is a functional bisimulation, so is $\mathscr{T}(R) \xrightarrow{\mathscr{T}(f)} \mathscr{T}^2(X) \xrightarrow{\mu_X} \mathscr{T}(X)$, and hence bisimilarity in $\mathscr{T}(X)$ is a congruence.*

Proof. Because $\mathscr{T}$ is familial, all naturality squares for μ are pullbacks, so μ_X is a functional bisimulation by Proposition 1.5. We conclude by the theorem. □

Section 4.3 is an exception: the π-calculus *is* a free algebra for a certain monad $\mathscr{T}_\pi$, but there we consider it as an algebra for a certain submonad of $\mathscr{T}_\pi$, which is hence not free.

4 THREE SHADES OF π-CALCULUS

Let us now consider a more significant example than combinatory logic and CCS: the π-calculus. Unlike in CCS, we have to mod out by α-equivalence, because channel names may be input, hence substituted deep in process terms, which may force renaming. We essentially follow Sangiorgi and Walker's presentation presentation [14, Section 1.3], using a simplified variant for expository purposes.

After explaining in Section 4.1 why we need to move away from the simple base category used for CCS in Section 2.2.2, we make in Section 4.2 a naive attempt at covering π-calculus using our approach. We manage to design a familial monad, $\mathscr{T}_\pi$, over a certain presheaf category $\widehat{\mathbb{B}}$, which faithfully encodes the desired labelled transition system. However, as bisimilarity is known not to be a congruence in π, something is bound to fail. And indeed, we show that the initial $\mathscr{T}_\pi$-algebra $\mathscr{T}_\pi(\varnothing)$ of processes is not compositional. We rectify this in a standard way in Section 4.3, by defining a familial submonad, $\mathscr{T}_\pi^-$, such that the $\mathscr{T}_\pi^-$-algebra $\mathscr{T}_\pi(\varnothing)$ is compositional, thus recovering the known fact that standard bisimilarity is a congruence for all operators but input. We finally consider in Section 4.4 a different, though still standard, way of remedying the non-congruence problem. This consists in restricting attention to relations that are stable under channel renaming. We do this by working over a different base category $\mathbb{F}$, and adapting the definition of $\mathscr{T}_\pi$, yielding a new familial monad $\mathscr{T}_\pi^+$. Bisimilarity for the free $\mathscr{T}_\pi^+$-algebra $\mathscr{T}_\pi^+(\varnothing)$ is *wide open* bisimilarity, in Staton's sense, and we recover the result that it is a congruence.

4.1 Naive approach

Naively adaptating what we did with CCS to the π-calculus, we could try working over the category freely generated by the graph with

- a vertex $\star$,
- vertices in $L = \bigcup_{a,b\in\mathscr{N}} \{\tau, o_{a,b}, \iota_{a,b}, o_a^\nu, \iota_b^\nu\}$ where $\mathscr{N}$ is as in Section 2.2.2 a fixed, countable set of names,
- for each $\alpha \in L$, edges $s, t : \star \to \alpha$.

This category forms a transition category by taking all cospans $\star \xrightarrow{s} \alpha \xleftarrow{t} \star$ as transition types.

However, the π-calculus has one axiom that this setting cannot accomodate, namely the input rule:

$$\frac{}{a(v).P \xrightarrow{a(b)} P[v \mapsto b]}\ \textsc{In}\ .$$

Indeed, this requires being able to rename variables (or, depending on the chosen presentation, to substitute names for variables). We could try to define a monad $\mathscr{T}_\pi^0$ equipped with an inductively defined renaming operation $\mathscr{T}_\pi^0(\star) \xrightarrow{[v\mapsto b]} \mathscr{T}_\pi^0(\star)$, but would stumble upon

the base case of $(x)[v \mapsto b]$, for $x \in X(\star)$. In that case, indeed, replacing v with b does not make any sense. We thus need consider a different base category.

4.2 Basic approach

In this section, we illustrate our approach by examining the failure of congruence for standard bisimilarity in π. We analyse the problem, which allows us to consider two different solutions in the next two sections.

Definition 4.1. Let $\mathbb{B}$ denote the subcategory of $\mathbf{Set}^{op}$ with finite subsets of $\mathscr{N}$ as objects and bijections as morphisms, augmented with

- objects $\tau_\gamma, o_{\gamma,a,b}, o^v_{\gamma,a,c}, \iota_{\gamma,a,b}$, and $\iota^v_{\gamma,a,c}$ for all $\gamma \in \mathscr{P}_f(\mathscr{N})$, $a, b \in \gamma$, and $c \notin \gamma$,
- morphisms giving the type of each transition:

$$\gamma \xrightarrow{s} \tau_\gamma \xleftarrow{t} \gamma \qquad \gamma \xrightarrow{s} o_{\gamma,a,b} \xleftarrow{t} \gamma \qquad \gamma \xrightarrow{s} \iota_{\gamma,a,b} \xleftarrow{t} \gamma \qquad \gamma \xrightarrow{s} o^v_{\gamma,a,c} \xleftarrow{t} \gamma, c \qquad \gamma \xrightarrow{s} \iota^v_{\gamma,a,c} \xleftarrow{t} \gamma, c\,,$$

  denoting by γ, c the (disjoint) union $\gamma \uplus \{c\}$,

- plus, for all $\gamma \xrightarrow{s} \alpha \xleftarrow{t} \gamma, \delta$ and bijections $h : \gamma \rightarrow\!\!\!\!\!\rightarrow \gamma'$ and $k : \delta \rightarrow\!\!\!\!\!\rightarrow \delta'$, a morphism $(h,k) : \alpha[h,k] \to \alpha$, where

$$\begin{array}{rcl} \tau_\gamma[h,k] & = & \tau_{\gamma'} \\ o_{\gamma,a,b}[h,k] & = & o_{\gamma',h(a),h(b)} \\ \iota_{\gamma,a,b}[h,k] & = & \iota_{\gamma',h(a),h(b)} \\ o^v_{\gamma,a,c}[h,k] & = & o_{\gamma',h(a),k(c)} \\ \iota^v_{\gamma,a,c}[h,k] & = & \iota_{\gamma',h(a),k(c)}, \end{array}$$

  satisfying the obvious equations:

$$\begin{array}{ccccc} \gamma & \xrightarrow{\ s\ } & \alpha & \xleftarrow{\ t\ } & \gamma, \delta \\ {\scriptstyle h}\uparrow & & \uparrow{\scriptstyle (h,k)} & & \uparrow{\scriptstyle h+k} \\ \gamma' & \xrightarrow[\ s\ ]{} & \alpha[h,k] & \xleftarrow[\ t\ ]{} & \gamma', \delta'. \end{array} \tag{4}$$

Notation 2. *For general presheaves X, we denote the action $X(c') \xrightarrow{X(f)} X(c)$ of any $f : c \to c'$ in the base by $x \mapsto x \cdot f$. However, for $f : \gamma \rightarrow\!\!\!\!\!\rightarrow \gamma'$ in sets, although f acts contravariantly as a map $\gamma' \to \gamma$ in $\mathbb{B}$, it acts covariantly as a set-map, so we often write $f \cdot x$ instead.*

The category $\widehat{\mathbb{B}}$ clearly forms a transition category, with transition types $\mathbf{T}^\pi$ given by all cospans above.

Let us now define our monad on $\widehat{\mathbb{B}}$. For any $X \in \widehat{\mathbb{B}}$, let $\mathscr{T}_\pi(X)$ denote the presheaf in which

- $\mathscr{T}_\pi(X)(\gamma)$ is the set of all α-equivalence classes of π-calculus terms of the form $\gamma \vdash P$, as defined in the top part of Figure 1;
- *renaming*, the action $\mathscr{T}_\pi(\gamma) \to \mathscr{T}_\pi(\gamma')$ of any set-map $f : \gamma \to \gamma'$, is defined inductively: for any $g : \gamma'' \to \gamma$, we set $f \cdot x(g) = x(fg)$ for the base case and rely on α-equivalence in the case of binders;
- for all $\gamma \xrightarrow{s} \alpha \xleftarrow{t} \gamma'$, $\mathscr{T}_\pi(X)(\alpha)$ denotes the set of α-equivalence classes of transitions

$$P \xrightarrow{\alpha} Q$$

$$\textsc{Const}\ \frac{x \in X(\gamma) \qquad f \in \mathbf{Set}(\gamma, \gamma')}{\gamma' \vdash_X x(f)} \qquad \textsc{Par}\ \frac{\gamma \vdash_X P \qquad \gamma \vdash_X Q}{\gamma \vdash_X P|Q} \qquad \textsc{Zero}\ \frac{}{\gamma \vdash_X 0} \qquad \textsc{Nu}\ \frac{\gamma, a \vdash_X P}{\gamma \vdash_X \nu a.P}$$

$$\textsc{Out}\ \frac{\gamma \vdash_X P \qquad a, b \in \gamma}{\gamma \vdash_X \overline{a}\langle b \rangle.P} \qquad \textsc{In}\ \frac{\gamma, b \vdash_X P \qquad a \in \gamma}{\gamma \vdash_X a(b).P}$$

$$\textsc{DoConst}\ \frac{x \in X(\alpha) \qquad \gamma \xrightarrow{s} \alpha \xleftarrow{t} \gamma, \delta}{\langle\!\langle x \rangle\!\rangle : \gamma \vdash_X (x \cdot s)(id) \xrightarrow{\alpha} \gamma, \delta \vdash_X (x \cdot t)(id)} \qquad \textsc{DoIn}\ \frac{\gamma, b \vdash_X P \qquad a, c \in \gamma}{in^{a,c}_{b.P} : \gamma \vdash_X a(b).P \xrightarrow{\iota_{\gamma,a,c}} \gamma \vdash_X P[b \mapsto c]}$$

$$\textsc{DoOut}\ \frac{\gamma \vdash_X P \qquad a, b \in \gamma}{out^{a,b}_{\gamma \vdash_X P} : \gamma \vdash_X \overline{a}\langle b \rangle.P \xrightarrow{o_{\gamma,a,b}} \gamma \vdash_X P}$$

$$\textsc{Sync}\ \frac{R : \gamma \vdash_X P \xrightarrow{o_{\gamma,a,b}} \gamma \vdash_X P' \qquad S : \gamma \vdash_X Q \xrightarrow{\iota_{\gamma,a,b}} \gamma \vdash_X Q'}{R \bowtie S : \gamma \vdash_X (P|Q) \xrightarrow{\tau_\gamma} \gamma \vdash_X (P'|Q')}\ \text{+ symmetric rule}$$

$$\textsc{BoundIn}\ \frac{\gamma, b \vdash_X P \qquad a \in \gamma}{in^{a,\nu b}_P : \gamma \vdash_X a(b).P \xrightarrow{\iota^\nu_{\gamma,a,b}} \gamma, b \vdash_X P} \qquad \textsc{Open}\ \frac{R : \gamma, b \vdash_X P \xrightarrow{o_{(\gamma,b),a,b}} \gamma, b \vdash_X Q \qquad a \neq b}{\nabla b.R : \gamma \vdash_X \nu b.P \xrightarrow{o^\nu_{\gamma,a,b}} \gamma, b \vdash_X Q}$$

$$\textsc{Close}\ \frac{R : \gamma \vdash_X P \xrightarrow{o^\nu_{\gamma,a,b}} \gamma, b \vdash_X P' \qquad S : \gamma \vdash_X Q \xrightarrow{\iota^\nu_{\gamma,a,b}} \gamma, b \vdash_X Q'}{R \bowtie^{\nu b} S : \gamma \vdash_X P|Q \xrightarrow{\tau_\gamma} \gamma \vdash_X \nu b.(P'|Q')}\ \text{+ symmetric rule}$$

$$\textsc{Res}\ \frac{R : \gamma, b \vdash_X P \xrightarrow{\alpha} \gamma, b, \delta \vdash_X Q \qquad b \notin fv(\alpha)}{\nu b.R : \gamma \vdash_X \nu b.P \xrightarrow{\nu b.\alpha} \gamma, \delta \vdash_X \nu b.Q} \qquad \textsc{Left}\ \frac{L : \gamma \vdash_X P \xrightarrow{\alpha} \gamma, \delta \vdash_X P' \qquad \gamma \vdash_X Q}{L|Q : \gamma \vdash_X P|Q \xrightarrow{\alpha} \gamma, \delta \vdash_X P'|Q}$$

$$\textsc{Right}\ \frac{\gamma \vdash_X P \qquad R : \gamma \vdash_X Q \xrightarrow{\alpha} \gamma, \delta \vdash_X Q'}{P|R : \gamma \vdash_X P|Q \xrightarrow{\alpha} \gamma, \delta \vdash_X P|Q'}$$

Fig. 1. Syntax and labelled transition system for π

with constants in X, for $P \in \mathscr{T}_\pi(X)(\gamma)$ and $Q \in \mathscr{T}_\pi(X)(\gamma')$, as defined in the bottom part of Figure 1, where

– the only binding operations are $in^{a,c}_{b.P}$, $\nu b.R$, and $R \bowtie^{\nu b} S$ – binding b;
– $P[b \mapsto c]$ stands for $f \cdot P$ with $f : \gamma, b \to \gamma$ mapping b to c and the rest of γ to itself;
– weakening is used implicitly in Rules Left and Right;
– in Rule Res, we define

$$\begin{array}{rcl} fv(o_{\gamma,a,b}) & = & \{a,b\} \\ fv(\iota_{\gamma,a,b}) & = & \{a,b\} \\ fv(o^{\nu}_{\gamma,a,b}) & = & \{a,b\} \\ fv(\iota^{\nu}_{\gamma,a,b}) & = & \{a,b\} \\ fv(\tau_\gamma) & = & \varnothing \end{array} \quad \text{and} \quad \begin{array}{rcl} \nu c.o_{(\gamma,c),a,b} & = & o_{\gamma,a,b} \\ \nu c.\iota_{(\gamma,c),a,b} & = & \iota_{\gamma,a,b} \\ \nu c.o^{\nu}_{(\gamma,c),a,b} & = & o^{\nu}_{\gamma,a,b} \\ \nu c.\iota^{\nu}_{(\gamma,c),a,b} & = & \iota^{\nu}_{\gamma,a,b} \\ \nu c.\tau_{\gamma,c} & = & \tau_\gamma \end{array}$$

where $\nu c.\alpha$ is defined iff $c \notin fv(\alpha)$.

Remark 3 (Replication). *The π-calculus standardly features additional operations like guarded sum and replication. Guarded sum may be incorporated readily, but replication may deserve some comment. Indeed, according to the chosen presentation, it is more or less easy to handle. E.g., the compact transition rule*

$$\frac{P|!P \xrightarrow{\alpha} Q}{!P \xrightarrow{\alpha} Q},$$

does not appear to yield a familial monad when considered over the current base category $\mathbb{B}$. Incorporating it appears to force us to consider a locally presentable, but non-presheaf category. Although nothing is wrong with that, it is significantly more complicated. The rules of [14, Section 1.3], however, which would here become

$$\frac{\gamma \vdash P \xrightarrow{\alpha} \gamma' \vdash P'}{\gamma \vdash !P \xrightarrow{\alpha} \gamma' \vdash P'|!P} \qquad \frac{\gamma \vdash P \xrightarrow{o_{\gamma,a,b}} \gamma \vdash P' \qquad \gamma \vdash P \xrightarrow{\iota_{\gamma,a,b}} \gamma \vdash P''}{\gamma \vdash !P \xrightarrow{\tau_\gamma} \gamma \vdash (P'|P'')|!P}$$

$$\frac{\gamma \vdash P \xrightarrow{o^{\nu}_{\gamma,a,b}} \gamma, b \vdash P' \qquad \gamma \vdash P \xrightarrow{\iota^{\nu}_{\gamma,a,b}} \gamma, b \vdash P''}{\gamma \vdash !P \xrightarrow{\tau_\gamma} \gamma \vdash \nu b.(P'|P'')|!P},$$

do yield a familial monad directly.

Proposition 4.2. *$\mathscr{T}_\pi$ forms a $\mathbf{T}^{\pi}_{s}$-familial monad.*

Proof. To see that $\mathscr{T}_\pi$ is a familial functor, by Lemma 3.3, it is enough to define a functor $E : el(\mathscr{T}_\pi(1)) \to \widehat{\mathbb{B}}$ such that

$$\mathscr{T}_\pi(X)(c) \cong \sum_{R \in \mathscr{T}_\pi(1)(c)} [E(c,R), X],$$

naturally in X and $c \in \mathbb{B}$. Elements of $\mathscr{T}_\pi(1)(\gamma)$ are processes of type γ, over exactly one constant process process, say $\bot_{\gamma'}$ of each type γ'. Elements of $\mathscr{T}_\pi(1)(\alpha)$ are transitions with exactly one constant transition of each type $\gamma \xrightarrow{s} \beta \xleftarrow{t} \gamma'$, say $\bot_\beta$, with source $\bot_\beta \cdot s = \bot_\gamma$ and target $\bot_\beta \cdot t = \bot_{\gamma'}$.

On objects, we construct E by induction on the depth of the considered derivation (globally for all objects $c \in \mathbb{B}$), as in the top part of Figure 2. This is in fact only well-defined up to isomorphism. Indeed, we need to make a global choice of coproducts, and furthermore binders are a delicate matter. E.g., although $\nu a.P = \nu b.(P[a \mapsto b])$ for $\gamma, a \vdash_X P$ and $b \notin \gamma$, the above definition of E gives $E(\gamma \vdash \nu a.P) = E(\gamma, a \vdash P)$ and $E(\gamma \vdash \nu b.(P[a \mapsto b])) = E(\gamma, b \vdash P[a \mapsto b])$. We solve this issue by picking a global choice of representatives for α-equivalence classes.

We observe in passing that for any $f : \gamma \to \gamma'$ in sets and $\gamma \vdash P$, we have by induction:

$$E(P[f]) \cong E(P). \tag{5}$$

Let now define E on morphisms:

- For any $f : \gamma \twoheadrightarrow \gamma'$ in sets and $P \in \mathscr{T}_\pi(1)(\gamma)$, we have $f \upharpoonright P : (\gamma', f \cdot P) \to (\gamma, P)$ in $el(\mathscr{T}_\pi(1))$ and define $E(f \upharpoonright P) : E(\gamma', f \cdot P) \to E(\gamma, P)$ by induction on P, as in the middle part of Figure 2.
- For all $\gamma \xrightarrow{s} \alpha \xleftarrow{t} \gamma, \delta$ and $R \in \mathscr{T}_\pi(1)(\alpha)$, we need to pick a cospan

$$E(R \cdot s) \xrightarrow{E(s \upharpoonright R)} E(R) \xleftarrow{E(t \upharpoonright R)} E(R \cdot t),$$

  which extends to a definition of E on all morphisms. We do so as in the bottom part of Figure 2. Clearly, the chosen maps lie in ${}^{\boxtimes}((\mathbf{T}^\pi_s)^{\boxtimes})$, so that $\mathscr{T}^\pi$ is $\mathbf{T}^\pi_s$-familial.
- Finally, for all $(h, k) : \alpha[h, k] \to \alpha$, we pick by induction the obvious isomorphism $E(\alpha[h, k]) \to E(\alpha)$.

This shows that $\mathscr{T}_\pi$ is a $\mathbf{T}^\pi_s$-familial functor. Its unit is given by variables, and multiplication is essentially substitution, which (is different from renaming and) is straightforwardly defined by induction in Figure 3. This clearly satisfies naturality in both arguments, as well as associativity and unitality. Finally, cartesianness of η and μ may be checked pointwise (i.e., relative to representable presheaves), and follows by induction. □

As announced and expected, because bisimilarity is not a congruence in π, something must fail. And it is compositionality:

Proposition 4.3. *The free $\mathscr{T}_\pi$-algebra, $\mathscr{T}_\pi(\varnothing)$, with action given by $\mu_\varnothing$, is not compositional.*

Proof. The problem comes from renaming, as embedded in constant terms $x(f)$. Indeed, consider $p = (\overline{a}\langle a\rangle.0 | b(c).0)$ (for $a \neq b$), a process involved in a standard counterexample to bisimilarity being a congruence [14]. Letting $f : \{a, b\} \to \{a\}$ denote the unique such map, $p(f) \in (\mathscr{T}_\pi)^2(\varnothing)\{a\}$ is mapped by $(\mu_\varnothing)_{\{a\}}$ to $p' = (\overline{a}\langle a\rangle.0 | a(c).0)$. The latter process has a τ-transition to $0|0$, which the former cannot match. □

Remark 4. *The root of the problem here is the DoIn rule, which forces the syntax for processes to feature renaming, even if only at the level of constants $x(f)$. To emphasise that this does make renaming a proper syntactic operation, it may help to realise that $\mathscr{T}_\pi$ could be defined using an explicit renaming operation $P[f]$ together with equations describing how it propagates down towards the leaves, e.g., $(P|Q)[f] = P[f]|Q[f]$, to finally integrate with constants: $x(g)[f] = x(fg)$.*

4.3 Non-input congruence

A first, standard way around a non-congruence of bisimilarity is to elude the problematic case, and prove that bisimilarity is a congruence for all operators but input. Our framework can encompass this by viewing π-calculus as a non-free algebra for a smaller monad $\mathscr{T}^-_\pi$.

Definition of E on objects

Processes

$$\begin{array}{rcl} E(\bot_\gamma(f:\gamma\to\gamma')) &=& \mathbf{y}_\gamma \\ E(P|Q) &=& E(P)+E(Q) \\ E(0) &=& \varnothing \\ E(\nu a.P) &=& E(P) \\ E(\overline{a}\langle b\rangle.P) &=& E(P) \\ E(a(b).P) &=& E(P) \end{array}$$

Transitions

$$\begin{array}{rcl} E\langle\!\langle \bot_\alpha \rangle\!\rangle &=& \mathbf{y}_\alpha \\ E(in^{a,c}_{b.P}) &=& E(P) \\ E(out^{a,c}_{P}) &=& E(P) \\ E(R\bowtie S) &=& E(R)+E(S) \\ E(in^{a,\nu b}_{P}) &=& E(P) \\ E(\nabla b.R) &=& E(R) \\ E(R\bowtie^{\nu b} S) &=& E(R)+E(S) \\ E(L|Q) &=& E(L)+E(Q) \\ E(P|R) &=& E(P)+E(R). \end{array}$$

Definition of E on morphisms: $E(f\restriction P) : E(\gamma', f\cdot P)\to P(\gamma,P)$, for $f:\gamma\nrightarrow\gamma'$ in sets

$$E(f\restriction \bot_{\gamma''}(g:\gamma''\to\gamma)) = \left(E(\bot_{\gamma''}(fg)) = \mathbf{y}_{\gamma''} \xrightarrow{id} \mathbf{y}_{\gamma''} = E(\bot_{\gamma''}(g))\right)$$

$$E(f\restriction P|Q) = \left(E(f\cdot(P|Q)) = E(f\cdot P)+E(f\cdot Q) \xrightarrow{E(f\restriction P)+E(f\restriction Q)} E(P)+E(Q) = E(P|Q)\right)$$

$$E(f\restriction 0) = \left(E(f\cdot 0) = \varnothing \xrightarrow{!} E(0)\right)$$

$$E(f\restriction \nu a.P) = \left(E(f\cdot \nu a.P) = E((f,a)\cdot P) \xrightarrow{E((f,a)\restriction P)} E(P) = E(\nu a.P)\right)$$

$$E(f\restriction \overline{a}\langle b\rangle.P) = \left(E(f\cdot \overline{a}\langle b\rangle.P) = E(f\cdot P) \xrightarrow{E(f\restriction P)} E(P) = E(\overline{a}\langle b\rangle.P)\right)$$

$$E(f\restriction a(b).P) = \left(E(f\cdot a(b).P) = E((f,b)\cdot P) \xrightarrow{E((f,b)\restriction P)} E(P) = E(a(b).P)\right)$$

Definition of E on morphisms: cospans $E(R\cdot s) \xrightarrow{E(s\restriction R)} E(R) \xleftarrow{E(t\restriction R)} E(R\cdot t)$

$$\begin{array}{rcl}
\langle\!\langle \bot_\alpha \rangle\!\rangle : & E(\mathbf{y}_\gamma) \xrightarrow{\mathbf{y}_s} E(\mathbf{y}_\alpha) \xleftarrow{\mathbf{y}_t} & E(\mathbf{y}_{\gamma,\delta}) \\
in^{a,c}_{b.P} : & E(a(b).P) = E(P) \xrightarrow{id} E(P) \xleftarrow{id} & E(P) = E(P[b\mapsto c]) \\
out^{a,c}_{P} : & E(\overline{a}\langle b\rangle.P) = E(P) \xrightarrow{id} E(P) \xleftarrow{id} & E(P) \\
R\bowtie S : & E(P)+E(Q) \xrightarrow{E(s\restriction R)+E(s\restriction S)} E(R)+E(S) \xleftarrow{E(t\restriction R)+E(t\restriction S)} & E(P')+E(Q') \\
in^{a,\nu b}_{P} : & E(P) \xrightarrow{id} E(P) \xleftarrow{id} & E(P) \\
\nabla b.R : & E(\nu b.P) = E(P) \xrightarrow{E(s\restriction R)} E(R) \xleftarrow{E(t\restriction R)} & E(Q) \\
R\bowtie^{\nu} S : & E(P)+E(Q) \xrightarrow{E(s\restriction R)+E(s\restriction S)} E(R)+E(S) \xleftarrow{E(t\restriction R)+E(t\restriction S)} & E(P')+E(Q') \\
\nu c.R : & E(\nu c.P) = E(P) \xrightarrow{E(s\restriction R)} E(R) \xleftarrow{E(t\restriction R)} & E(Q) = E(\nu c.Q) \\
L|Q : & E(P)+E(Q) \xrightarrow{E(s\restriction L)+id} E(L)+E(Q) \xleftarrow{E(t\restriction L)+id} & E(P')+E(Q') \\
P|R : & E(P)+E(Q) \xrightarrow{id+E(s\restriction R)} E(P)+E(Q) \xleftarrow{id+E(t\restriction R)} & E(P)+E(Q')
\end{array}$$

Fig. 2. Definition of E

Processes:

$$
\begin{array}{rcl}
(\mu_X)_\gamma(p(f)) & = & p[f] \\
(\mu_X)_\gamma(P|Q) & = & (\mu_X)_\gamma(P)|(\mu_X)_\gamma(Q) \\
(\mu_X)_\gamma(va.P) & = & va.(\mu_X)_{\gamma,a}(P) \\
(\mu_X)_\gamma(0) & = & 0 \\
(\mu_X)_\gamma(\overline{a}\langle b\rangle.P) & = & \overline{a}\langle b\rangle.(\mu_X)_\gamma(P) \\
(\mu_X)_\gamma(a(b).P) & = & a(b).(\mu_X)_{\gamma,b}(P)
\end{array}
$$

Transitions:

$$
\begin{array}{rcl}
(\mu_X)_\alpha(\!|r|\!) & = & r \\
(\mu_X)_{\iota_{\gamma,a,c}}(in^{a,c}_{b.P}) & = & in^{a,c}_{b.(\mu_X)_{\gamma,b}(P)} \\
(\mu_X)_{o_{\gamma,a,b}}(out^{a,b}_P) & = & out^{a,b}_{(\mu_X)_\gamma(P)} \\
(\mu_X)_{\tau_\gamma}(R \bowtie S) & = & (\mu_X)_{o_{\gamma,a,b}}(R) \bowtie (\mu_X)_{\iota_{\gamma,a,b}}(S) \\
(\mu_X)_{\iota^\nu_{\gamma,a,b}}(in^{a,\nu b}_P) & = & in^{a,\nu b}_{(\mu_X)_{\gamma,b}(P)} \\
(\mu_X)_{o^\nu_{\gamma,a,b}}(\nabla b.R) & = & \nabla b.(\mu_X)_{o_{(\gamma,b),a,b}}(R) \\
(\mu_X)_{\tau_\gamma}(R \bowtie^{\nu b} S) & = & (\mu_X)_{o^\nu_{\gamma,a,b}}(R) \bowtie^{\nu b} (\mu_X)_{\iota^\nu_{\gamma,a,b}}(S) \\
(\mu_X)_{\nu c.\alpha}(\nu c.R) & = & \nu c.(\mu_X)_\alpha(R) \\
(\mu_X)_\alpha(L|Q) & = & (\mu_X)_\alpha(L)|(\mu_X)_\gamma(Q) \\
(\mu_X)_\alpha(P|R) & = & (\mu_X)_\gamma(P)|(\mu_X)_\alpha(R).
\end{array}
$$

Fig. 3. Monad multiplication for $\mathscr{T}_\pi$

Definition 4.4. Let $\mathscr{T}^-_\pi$ denote the sub-functor of $\mathscr{T}_\pi$ obtained by removing rule In from the syntax of Figure 1, replacing rule Const by rule

$$\frac{x \in X(\gamma) \qquad f \in \mathbf{Inj}(\gamma,\gamma')}{\gamma' \vdash_X x(f)}\ \text{Const}',$$

and removing rule DoIn from the labelled transition system of Figure 1.

Remark 5. *We need to retain injective renaming for Rules Left and Right to make sense.*

We clearly obtain:

Proposition 4.5. *$\mathscr{T}^-_\pi$ is a $\mathbf{T}^\pi_s$-familial monad.*

But this time, instead of considering $\mathscr{T}^-_\pi(\varnothing)$, which does not even satisfy the input rule, we observe that $\mathscr{T}_\pi(\varnothing)$ forms a $\mathscr{T}^-_\pi$-algebra (because it satisfies all π-calculus rules, hence in particular those of $\mathscr{T}^-_\pi$). We thus obtain:

Proposition 4.6. *The $\mathscr{T}^-_\pi$-algebra $\mathscr{T}_\pi(\varnothing)$ is compositional.*

Let us now explain how this gives an alternative proof to [14, Theorem 2.2.8(1)], which says that bisimilarity is a *non-input congruence* [14, Definition 2.1.23]. In our language:

Corollary 4.7. *Bisimilarity is a non-input congruence, i.e., bisimilarity for the $\mathscr{T}^-_\pi$-algebra $\mathscr{T}_\pi(\varnothing)$ is a congruence.*

Proof. By Corollary 3.13, $\mathbf{T}^\pi_s$-familiality of $\mathscr{T}^-_\pi$, and compositionality of $\mathscr{T}_\pi(\varnothing)$ *qua* $\mathscr{T}^-_\pi$-algebra. □

4.4 Wide open bisimilarity

Another standard solution to the failure of bisimilarity to be a congruence is to resort to *wide open bisimilarity* [15]. For this,, we need to modify our base category $\mathbb{B}$ to include non-bijective maps $\gamma \to \gamma'$.

Definition 4.8. Let $\mathbb{F}$ denote the subcategory of $\mathbf{Set}^{op}$ with finite subsets of $\mathscr{N}$ as objects and all maps as morphisms, augmented with

- objects $\tau_\gamma, o_{\gamma,a,b}, o^\nu_{\gamma,a,c}, \iota_{\gamma,a,b}$, and $\iota^\nu_{\gamma,a,c}$ for all $\gamma \in \mathscr{P}_f(\mathscr{N})$, $a, b \in \gamma$, and $c \notin \gamma$,
- morphisms giving the type of each transition:

$$\gamma \xrightarrow{s} \tau_\gamma \xleftarrow{t} \gamma \qquad \gamma \xrightarrow{s} o_{\gamma,a,b} \xleftarrow{t} \gamma \qquad \gamma \xrightarrow{s} \iota_{\gamma,a,b} \xleftarrow{t} \gamma \qquad \gamma \xrightarrow{s} o^\nu_{\gamma,a,c} \xleftarrow{t} \gamma, c \qquad \gamma \xrightarrow{s} \iota^\nu_{\gamma,a,c} \xleftarrow{t} \gamma, c\,,$$

- plus, for all $\gamma \xrightarrow{s} \alpha \xleftarrow{t} \gamma, \delta$, maps $h : \gamma \twoheadrightarrow \gamma'$, and bijections $k : \delta \twoheadrightarrow \delta'$, a morphism $(h, k) : \alpha[h, k] \to \alpha$, where

$$\begin{array}{rcl@{\qquad}rcl} \tau_\gamma[h,k] & = & \tau_{\gamma'} & o^\nu_{\gamma,a,c}[h,k] & = & o_{\gamma',h(a),k(c)} \\ o_{\gamma,a,b}[h,k] & = & o_{\gamma',h(a),h(b)} & \iota^\nu_{\gamma,a,c}[h,k] & = & \iota_{\gamma',h(a),k(c)}, \\ \iota_{\gamma,a,b}[h,k] & = & \iota_{\gamma',h(a),h(b)} & & & \end{array}$$

  satisfying equations (4).

Please note that while we include non-bijective morphisms $f : \gamma \to \gamma'$, we do not (need to) do the same for transition objects. The presheaf category $\widehat{\mathbb{F}}$ forms a transition category with transition types given by all cospans $\mathbf{T}^+_\pi$ above.

Let us now adapt our monad on $\widehat{\mathbb{F}}$.

Definition 4.9. Let $\mathscr{T}^+_\pi$ denote the functor defined from the rules of Figure 1, replacing rule Const by rule

$$\frac{x \in X(\gamma)}{\gamma \vdash_X x}\ \textsc{Const}'.$$

It may be helpful to comment a bit on renaming. For $\mathscr{T}_\pi$, in rules DoIn, Left and Right, we made use of renaming, which was defined by induction on processes. So we defined processes first, then renaming, and finally transitions. Here, things are similar, except that the given presheaf X comes equipped with an action of all maps $\gamma \to \gamma'$, not just bijections. Thus, the treatment of constants may be simplified: instead of pairs $x(f)$, mapped by any g to $x(gf)$, we may reduce to just elements x, mapped by g to $g \cdot x$ (i.e., $X(g)(x)$).

It is not entirely trivial that this new monad is also familial.

Proposition 4.10. *$\mathscr{T}^+_\pi$ is a $\mathbf{T}^+_s$-familial monad.*

Proof. Most of the definitions are adapted straightforwardly from $\mathbb{B}$ to $\mathbb{F}$. The most significant change is the action of E on $t \upharpoonright in^{a,c}_{b.P} : P[b \mapsto c] \to in^{a,c}_{b.P}$. Indeed, while before we had $E(P[f]) \cong E(P)$ (see (5)), we now have, calling E^+ the functor $el(\mathscr{T}^+_\pi(1)) \to \widehat{\mathbb{F}}$ making $\mathscr{T}^+_\pi$ familial: $E^+(\bot_\gamma[f]) = E^+(f \cdot \bot_\gamma) = E^+(\bot_{\gamma'}) = \mathbf{y}_{\gamma'}$. So we pick for $E^+(t \upharpoonright in^{a,c}_{b.P})$ the morphism

defined by induction on P as follows:

$$\begin{aligned}
E^+(t \upharpoonright in^{a,c}_{b.\perp_{\gamma,b}}) &= (\mathbf{y}_\gamma \xrightarrow{\mathbf{y}_{[b\mapsto c]}} \mathbf{y}_{\gamma,b}) \\
E^+(t \upharpoonright in^{a,c}_{b.(P|Q)}) &= (E^+(P[f]) + E^+(Q[f]) \xrightarrow{E^+(t\upharpoonright in^{a,c}_{b.P})+E^+(t\upharpoonright in^{a,c}_{b.Q})} E^+(P) + E^+(Q) \\
E^+(t \upharpoonright in^{a,c}_{b.0}) &= (\varnothing \xrightarrow{id} \varnothing) \\
E^+(t \upharpoonright in^{a,c}_{b.\nu d.P}) &= E^+(P[f + id_{\{d\}}]) \xrightarrow{E^+(t\upharpoonright in^{a,c}_{b.P})} E^+(P) \\
E^+(t \upharpoonright in^{a,c}_{b.\bar{d}_1\langle d_2\rangle.P}) &= E^+(P[f]) \xrightarrow{E^+(t\upharpoonright in^{a,c}_{b.P})} E^+(P) \\
E^+(t \upharpoonright in^{a,c}_{b.d_1(d_2).P}) &= E^+(P[f + id_{\{d_2\}}]) \xrightarrow{E^+(t\upharpoonright in^{a,c}_{b.P})} E^+(P).
\end{aligned}$$

Please observe that in the base case, the set-map $[b \mapsto c] : \gamma, b \to \gamma$ yields a morphism in the opposite direction in $\mathbb{F}$ hence in $\widehat{\mathbb{F}}$. □

The compositionality problem created by the presence of renaming in the syntax (as $x(f)$) having disappeared, we get:

Proposition 4.11. *$\mathscr{T}^+_\pi(\varnothing)$ is a compositional algebra.*

Remark 6. *It might be slightly worrying at first to observe that $E^+(t \upharpoonright in^{a,c}_{b.\perp_{\gamma,b}})$, namely $\mathbf{y}_\gamma \xrightarrow{\mathbf{y}_{[b\mapsto c]}} \mathbf{y}_{\gamma,b}$ is definitely not a coproduct of isomorphisms or t-maps. But in fact, any bisimulation is stable under renaming by construction, hence everything works smoothly.*

Calling *wide open* bisimilarity the largest bisimulation relation over $\mathscr{T}^+_\pi(\varnothing)$, we get:

Proposition 4.12. *Wide open bisimilarity is a congruence.*

Proof. By $\mathbf{T}^+_s$-familiality of $\mathscr{T}^+_\pi$ and compositionality of $\mathscr{T}^+_\pi(\varnothing)$. □

Remark 7. *Instead of changing the base category from $\mathbb{B}$ to $\mathbb{F}$, we could simply augment $\mathbb{B}$ with a transition type $\gamma, a \xrightarrow{s} \sigma_{a,b} \xleftarrow{t} \gamma$, for all $b \in \gamma$, and consider a substitution transition rule as follows*

$$\frac{\gamma, a \vdash_X P \qquad b \in \gamma}{Subst_{a,b}(P) : \gamma, a \vdash_X P \xrightarrow{\sigma_{a,b}} \gamma \vdash_X P[a \mapsto b]}.$$

The induced monad is familial again, and we recover wide open bisimilarity.

Remark 8. *Similarly, we could handle open bisimilarity by considering yet another base category, where instead of finite sets of names and (bijective) maps we would have as objects pairs (γ, D) of a finite set γ of names and a* distinction *D, i.e., an irreflexive relation on γ, with maps $(\gamma, D) \to (\gamma', D')$ all maps $f : \gamma \to \gamma'$* respecting *D, i.e., if $(a, b) \in D$ then $f(a) \neq f(b)$. This would lead to* open bisimilarity *[14, Definition 4.6.2]*

5 BISIMULATION UP TO CONTEXT

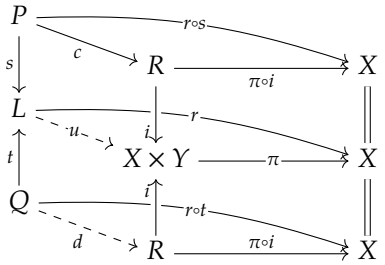
In this section, we consider a different notion of bisimulation in transition categories, *pre-bisimulations*, which we relate to the former by showing that under a mild additional hypothesis any pre-bisimulation embeds into some bisimulation. We then define a notion

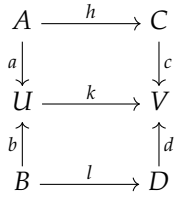
of *pre-bisimulation up to context*, which in examples corresponds to bisimulation up to context. Finally, we prove that pre-bisimulation up to context is sound, in the sense that any pre-bisimulation up to context embeds into some pre-bisimulation (and hence into some bisimulation).

5.1 Pre-bisimulations

In Definition 2.4, we defined functional bisimulations through a lifting property, which is flexible enough to, e.g., not consider all transitions from the considered labelled transition systems X and Y. The following is in fact closer to the ordinary definition of a bisimulation.

Definition 5.1. Consider any transition category. A *pre-bisimulation* $X \nrightarrow Y$ is a map $i : R \to X \times Y$ such that for all transition types $P \xrightarrow{s} L \xleftarrow{t} Q$, processes $c : P \to R$, and transitions $r : L \to X$ making the solid part below left commute, there exist u and d as shown making the whole commute, and symmetrically for Y.

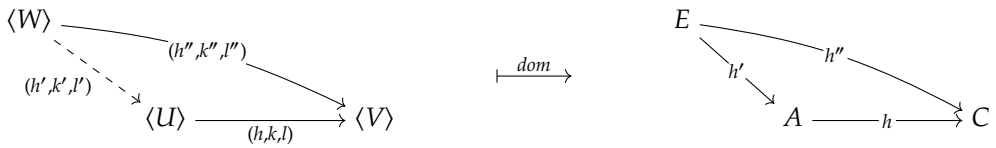




(6)

Equivalently, the pre-bisimulation condition may be formulated as a *weak cartesianness* property of the cospan map formed by the front face above left, relative to transition types. Indeed, consider the category $\mathscr{A}^\vee$ with cospans in $\mathscr{A}$ as objects, and as morphisms all commuting diagrams of the form above right. Taking the top component (i.e., sending (h, k, l) to h) defines a functor $dom : \mathscr{A}^\vee \to \mathscr{A}$.

Definition 5.2. A morphism (6) is *weakly cartesian* relative to some cospan $E \to W \leftarrow F$ iff, denoting by $\langle U \rangle$ the cospan with U as middle component (and inferring the rest from context), for all morphisms $(h'', k'', l'') : \langle W \rangle \to \langle V \rangle$ in $\mathscr{A}^\vee$, and morphisms $h' : E \to A$ such that $h \circ h' = h''$, there exists k' and l' making the diagram below commute.



Remark 9. *This is the cartesianness condition from Grothendieck fibrations, except with weakened universal property: it is restricted to certain cospans, and the mediating arrow need not be unique.*

Intuitively, the value of R over transitions is irrelevant in the definition. But, thinking of R as a relation, one would expect that by completing it with all transitions that exist in $X \times Y$ between pairs of related processes we would get a bisimulation. This completion operation may be performed generically by a factorisation system, up to an additional stratification hypothesis. Let us first define the factorisation system in question, then introduce the needed hypothesis, and finally relate pre-bisimulation to bisimulation (Proposition 5.6).

For any transition category $\mathscr{A}$, as we have seen, the small object argument applies, so we can put:

Definition 5.3. Let $(\mathscr{L}, \mathscr{R})$ denote the weak factorisation system cofibrantly generated by the set $\mathscr{J} = \{[s,t] : P + Q \to L \mid (s,t) \in \mathbf{T}\}$.

Definition 5.4. A transition category $\mathscr{A}$ is *two-level* iff all maps $[s,t] : P + Q \to L$, for $(s,t) \in \mathbf{T}$ are such that any map $f : P' \to L$ with $P' \in \mathbf{P}$ lifts through $[s,t]$, i.e., there exists k making the following triangle commute.

$$\begin{array}{ccc} & & P+Q \\ & \overset{k}{\nearrow} & \downarrow{\scriptstyle [s,t]} \\ P' & \xrightarrow[f]{} & L \end{array} \tag{7}$$

Remark 10. *This is equivalent to all maps $[s,t]$ being in $\mathbf{P}^{\boxslash}$, viewing each $P' \in \mathbf{P}$ as the unique map $0 \to P'$.*

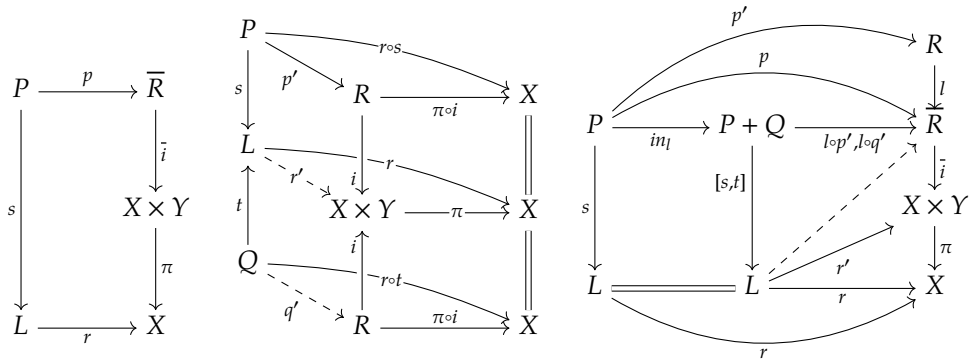
Notation 3. *In accordance with the previous remark, we will denote by $\mathbf{P}^{\boxslash}$ all maps satisfying the right lifting property (7) w.r.t. all $P' \in \mathbf{P}$.*

Lemma 5.5. *In any two-level transition category, we have $\mathscr{L} \subseteq \mathbf{P}^{\boxslash}$.*

Proof. By Proposition 1.6, it suffices to show that $\mathbf{P}^{\boxslash}$ contains $\mathscr{J}$ and is stable under pushout, transfinite composition, and retracts. First of all, $\mathscr{J} \subseteq \mathbf{P}^{\boxslash}$ holds by hypothesis. Stability under retracts is direct, and stability under pushout and transfinite composition follows from tininess of process types (by transfinite induction in the latter case, using existence of transfinite composites). □

Proposition 5.6. *For any pre-bisimulation $i : R \to X \times Y$ in a two-level transition category, the right factor $\bar{i}$ of the $(\mathscr{L}, \mathscr{R})$-factorisation $R \overset{l}{\to} \overline{R} \overset{\bar{i}}{\to} X \times Y$ of i is a bisimulation $X \nrightarrow Y$.*

Proof. Consider any commuting square as below left. By Lemma 5.5, we find a lifting, say $p' : P \to R$ of p through $l : R \to \overline{R}$. Because R is a pre-bisimulation, we get r' and q' making the diagram below center commute. We may thus factor the left-hand square as below right, and hence get the desired (dashed) lifting by $\bar{i} \in \mathscr{R}$.

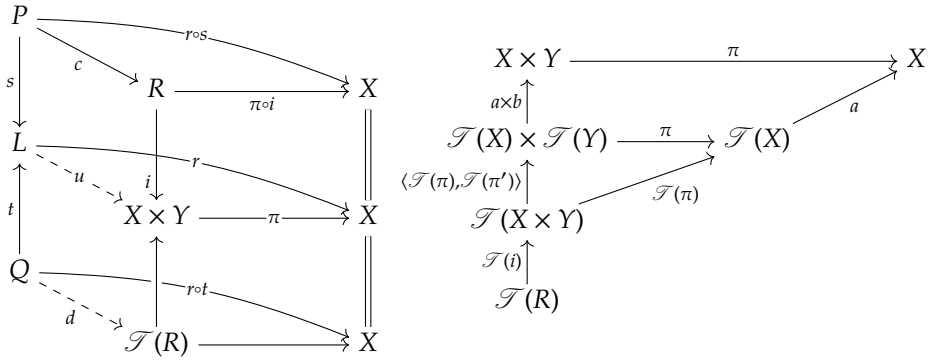


5.2 Pre-bisimulation up to context

After defining pre-bisimulations in the previous section, and showing that they induce bisimulations by factorisation, we now proceed in this section to defining pre-bisimulations up to context, and showing that they induce pre-bisimulations, hence bisimulations, provided an additional saturation hypothesis holds.

Let $\mathscr{T}$ denote a familial monad on a transition category $\mathscr{A}$, and let $a : \mathscr{T}(X) \to X$ and $b : \mathscr{T}(Y) \to Y$ be $\mathscr{T}$-algebras.

Definition 5.7. A map $i : R \to X \times Y$ is a *pre-bisimulation up to* $\mathscr{T}$ iff for all transition types $P \xrightarrow{s} L \xleftarrow{t} Q$, processes $c : P \to R$, and transitions $r : L \to X$ making the solid part below commute, there exist u and d as shown making the whole commute, and symmetrically for Y, where the bottom square, viewed as a triangle, is defined as on the right.

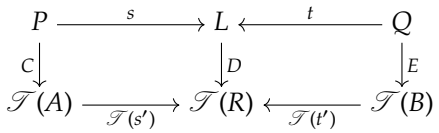


(8)

Following Definition 5.2, this may be expressed as a weak cartesianness condition relative to transition types.

Before proving soundness of pre-bisimulation up to $\mathscr{T}$, we need to find an analogue of $\mathbf{T}_s$-familiality in the previous section. Indeed, in order to find the desired lifting in the proof of Theorem 3.10, we needed to assume that the image by E of any $s \in \mathbf{T}_s$ was in ${}^{\boxtimes}(\mathbf{T}_s^{\boxtimes})$. With pre-bisimulations, things are different, however, because they are not defined through any lifting property. Instead, we use the following notion of saturated pre-bisimulations up to $\mathscr{T}$.

Definition 5.8. Let $\overline{\mathbf{T}}$ denote the class of cospans (s', t') occurring in any pair of generic-free factorisations of the form below, where $(s, t) \in \mathbf{T}$ and D is $\mathscr{T}$-generic.



A pre-bisimulation up to $\mathscr{T}$ is *saturated* iff the front face in (8) is weakly cartesian relative to all cospans in $\overline{\mathbf{T}}$.

Theorem 5.9. *For any saturated pre-bisimulation up to $\mathscr{T}$, say $i : R \to X \times Y$, where X and Y are compositional $\mathscr{T}$-algebras with respective structure maps $a : \mathscr{T}(X) \to X$ and $b : \mathscr{T}(Y) \to Y$, the map*

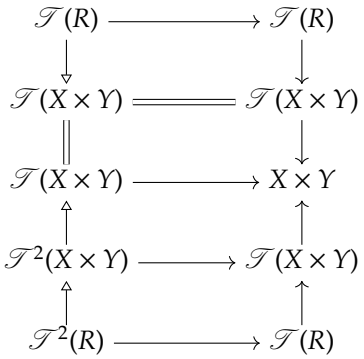
$$\mathscr{T}(R) \xrightarrow{\mathscr{T}(i)} \mathscr{T}(X \times Y) \xrightarrow{\langle \mathscr{T}(\pi), \mathscr{T}(\pi') \rangle} \mathscr{T}(X) \times \mathscr{T}(Y) \xrightarrow{a \times b} X \times Y \tag{9}$$

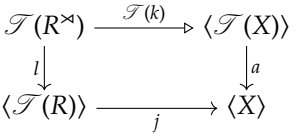
is a pre-bisimulation.

Proof. Let us write

- $\langle X \rangle$ for the cospan (id_X, id_X),
- $\langle \mathscr{T}(R) \rangle$ for the cospan (i', i'), where $i' : \mathscr{T}(R) \to X \times Y$ denotes (9),
- $k : R^{\rtimes} \to \langle X \rangle$ for the cospan map given by the front face of the left diagram in (8),
- $j : \langle \mathscr{T}(R) \rangle \to \langle X \rangle$ for the cospan map given by twice the right-hand diagram in (8).

By symmetry, it suffices to prove that j is weakly cartesian relative to transition types. Let us start by observing that the cospan morphism $l : \mathscr{T}(R^{\rtimes}) \to \langle \mathscr{T}(R) \rangle$ below left makes the diagram below right commute (Please note that from now on we use open arrow heads to denote arrows of the form $\mathscr{T}(-)$, and when we say that a diagram made of such arrows commute, we mean that the underlying arrows, without $\mathscr{T}$, agree).

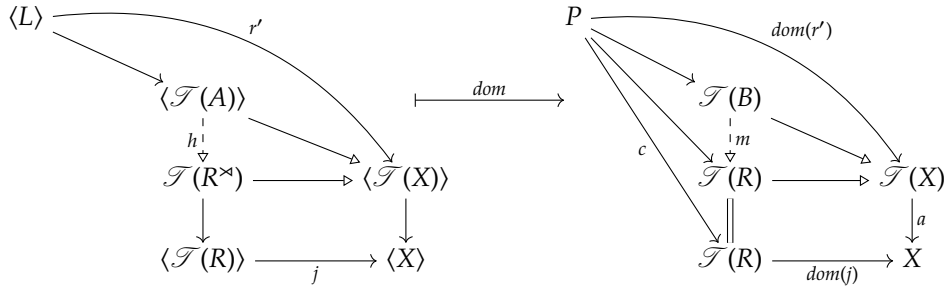




Now, consider a map $r : \langle L \rangle \to \langle X \rangle$, where $\langle L \rangle$ denotes any cospan in $\mathbf{T}$, whose top component x factors through the top component of j, i.e.,

$$\mathscr{T}(R) \xrightarrow{\mathscr{T}(i)} \mathscr{T}(X \times Y) \xrightarrow{\mathscr{T}(\pi)} \mathscr{T}(X) \xrightarrow{a} X,$$

say as $P \xrightarrow{c} \mathscr{T}(R) \to X$. By compositionality of X, r factors as $\langle L \rangle \xrightarrow{r'} \langle \mathscr{T}(X) \rangle \xrightarrow{a} \langle X \rangle$ (because the top component of j does). By familiality of $\mathscr{T}$, we get a componentwise generic-free factorisation of r', say as $\langle L \rangle \to \langle \mathscr{T}(A) \rangle \longrightarrow\!\!\!\!\triangleright \langle \mathscr{T}(X) \rangle$, as below.



By genericity of its top component, say $P \to \langle \mathscr{T}(B) \rangle$, we get a morphism m as shown. Finally, by weak cartesianness of $R^{\rtimes} \to \langle X \rangle$ relative to $\overline{\mathbf{T}}$, we obtain a morphism h as shown, which yields the desired lifting, namely the composite $\langle L \rangle \to \langle \mathscr{T}(A) \rangle \xrightarrow{h} \mathscr{T}(R^{\rtimes}) \xrightarrow{l} \langle \mathscr{T}(R) \rangle$. □

As before, we get a bunch of cheap, yet useful corollaries. E.g., although we do not have any precise characterisation like that of ${}^{\boxtimes}(\mathbf{T}_s^{\boxtimes})$, we do have:

PROPOSITION 5.10. *If* $\overline{\mathbf{T}}$ *consists of coproducts of isomorphisms and cospans in* $\mathbf{T}$*, then all pre-bisimulations up to* $\mathscr{T}$ *are saturated.*

Example 5.11. As an example application of the theorem, bisimulation up to $\mathscr{T}_\pi^-$ is sound in the basic presentation of π (Section 4.2), as is well-known [14, Lemma 2.3.21]. Another example application: in the refined presentation of Section 4.4, bisimulation up to $\mathscr{T}_\pi^+$ is sound, which is new to us.

6 CONCLUSION AND PERSPECTIVES

We presented a simple abstract framework for studying congruence of bisimilarity, based on familial monads and lifting properties. We then refined the framework to account for soundness of bisimulation up to context, using a weak form of cartesianness instead of lifting properties. To my knowledge, this is the first abstract framework handling both congruence of bisimilarity and soundness of bisimulation up to context in the presence of binding.

However, although the framework provides abstract, rather general proofs of non-trivial facts, it does not come with any format, i.e., means to construct instances from more basic data. An obvious next step is thus to look for such formats, which in our case means automatically constructing familial monads satisfying the relevant hypotheses. Another potential direction is to consider questions related to congruence of bisimilarity and soundness of bisimulation up to context, like Howe's method, environmental bisimulation, or solutions of process equations. Finally, familial monads for weak bisimulation and typed calculi might be worth investigating.

REFERENCES

[1] Jiří Adámek and Jiří Rosicky. 1994. *Locally Presentable and Accessible Categories*. Cambridge University Press. https://doi.org/10.1017/CBO9780511600579

[2] Filippo Bonchi, Daniela Petrişan, Damien Pous, and Jurriaan Rot. 2016. A general account of coinduction up-to. *Acta Informatica* (2016), 1–64. https://doi.org/10.1007/s00236-016-0271-4

[3] Aurelio Carboni and Peter Johnstone. 1995. Connected Limits, Familial Representability and Artin Glueing. *Mathematical Structures in Computer Science* 5, 4 (1995), 441–459. https://doi.org/10.1017/S0960129500001183

[4] Marcelo P. Fiore and Sam Staton. 2006. A Congruence Rule Format for Name-Passing Process Calculi from Mathematical Structural Operational Semantics. In *Proc. 21st Symposium on Logic in Computer Science* IEEE, 49–58. https://doi.org/10.1109/LICS.2006.7

[5] Richard H. G. Garner and Tom Hirschowitz. 2018. Shapely monads and analytic functors. *Journal of Logic and Computation* 28, 1 (2018), 33–83. https://doi.org/10.1093/logcom/exx029

[6] Mark Hovey. 1999. *Model Categories*. Mathematical Surveys and Monographs, Volume 63, AMS (1999), Vol. 63. American Mathematical Society.

[7] André Joyal, Mogens Nielsen, and Glynn Winskel. 1993. Bisimulation and open maps. In *Proc. 8th Symposium on Logic in Computer Science* IEEE, 418–427. https://doi.org/10.1109/LICS.1993.287566

[8] Bartek Klin. 2011. Bialgebras for structural operational semantics: An introduction. *Theoretical Computer Science* 412, 38 (2011), 5043–5069. https://doi.org/10.1016/j.tcs.2011.03.023

[9] Saunders Mac Lane. 1998. *Categories for the Working Mathematician* (2nd ed.). Number 5 in Graduate Texts in Mathematics. Springer.

[10] Saunders Mac Lane and Ieke Moerdijk. 1992. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer.

[11] MohammadReza Mousavi, Michel A Reniers, and Jan Friso Groote. 2007. SOS formats and meta-theory: 20 years after. *Theoretical Computer Science* 373, 3 (2007), 238–272.

[12] Gordon D. Plotkin. 1981. *A structural approach to operational semantics*. DAIMI Report FN-19. Computer Science Department, Aarhus University.

[13] Emily Riehl. 2014. *Categorical Homotopy Theory*. Number 24 in New Mathematical Monographs. Cambridge University Press.

[14] Davide Sangiorgi and David Walker. 2001. *The π-calculus - a theory of mobile processes*. Cambridge University Press.

[15] Sam Staton. 2008. General Structural Operational Semantics through Categorical Logic. In *Proc. 23rd Symposium on Logic in Computer Science* 166–177. https://doi.org/10.1109/LICS.2008.43

[16] Daniele Turi and Gordon D. Plotkin. 1997. Towards a Mathematical Operational Semantics. In *Proc. 12th Symposium on Logic in Computer Science* 280–291. https://doi.org/10.1109/LICS.1997.614955

[17] Mark Weber. 2007. Familial 2-functors and parametric right adjoints. *Theory and Applications of Categories* 18, 22 (2007), 665–732.