HAL open science

Familial Monads and Structural Operational Semantics

Tom Hirschowitz

▶ To cite this version:

Tom Hirschowitz. Familial Monads and Structural Operational Semantics. Proceedings of the ACM on Programming Languages, 2019, 3 (POPL), pp.1-28. 10.1145/3290334 . hal-01815328v4

HAL Id: hal-01815328

https://hal.science/hal-01815328v4

Submitted on 13 Nov 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Familial Monads and Structural Operational Semantics

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

We propose a categorical framework for structural operational semantics, in which we prove that under suitable hypotheses bisimilarity is a congruence. We then refine the framework to prove soundness of bisimulation up to context, an efficient method for reducing the size of bisimulation relations. Finally, we demonstrate the flexibility of our approach by reproving known results in three variants of the π-calculus.

CCS Concepts: • **Theory of computation** → **Denotational semantics**; **Operational semantics**; *Process calculi*; *Categorical semantics*;

Additional Key Words and Phrases: Structural operational semantics, category theory, familial monads

ACM Reference Format:
Tom Hirschowitz. 2019. Familial Monads and Structural Operational Semantics. *Proc. ACM Program. Lang.* 3, POPL, Article 21 (January 2019), 28 pages. https://doi.org/10.1145/3290334

1 INTRODUCTION

1.1 Motivation

Structural operational semantics [Plotkin 1981] is a method for specifying the dynamics of programming languages by induction on their syntax. This means that one describes the behaviour of a program in terms of its components, a feature often called *compositionality*. An important issue in structural operational semantics is the extent to which compositionality entails good behaviour of the generated labelled transition system. In this paper, we consider two particular questions: *congruence of bisimilarity* and *soundness of bisimulation up to context*.

The former is a long-standing problem in structural operational semantics. Bisimilarity is one of the most widely used behavioural equivalences, and congruence of bisimilarity essentially amounts to substitutivity: given two bisimilar program fragments P and Q (which we denote by $P \sim Q$), do we have $C[P] \sim C[Q]$ for any context C? Bisimilarity is famously known not to be a congruence in general, e.g., in the π-calculus [Sangiorgi and Walker 2001, §2.2.1].

Our second object of study is bisimulation up to context [Pous and Sangiorgi 2011; Sangiorgi and Walker 2001], an efficient variant of bisimulation, which often produces the same results using simpler relations. However, it is sometimes unsound, in the sense that bisimilarity up to context may not entail bisimilarity. Just like congruence of bisimilarity, soundness of bisimulation up to context has proved to be a subtle matter.

The difficulty of such questions, particularly the former, led to a rich variety of syntactic *formats* [Mousavi et al. 2007], which ensure good behaviour of the generated labelled transition system, up to some constraints on the considered specification. Despite their diversity, these formats have a lot in common, both in definitions and in proof schemes.

This commonality motivated *functorial operational semantics* [Klin 2011; Turi and Plotkin 1997], a unifying theory of formats, in which specifications are recast as distributive laws of a comonad

Author's address: Tom Hirschowitz, Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, LAMA, Chambéry, France.

This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2019 Copyright held by the owner/author(s).
2475-1421/2019/1-ART21
https://doi.org/10.1145/3290334

over a monad. The approach has been deeply developed in the set-based case, particularly for congruence of bisimilarity [Mousavi et al. 2007], but also, among other questions, for soundness of bisimulation up to context [Bonchi et al. 2016]. A few attempts have also been made to cover languages with variable binding [Fiore and Staton 2006; Fiore and Turi 2001; Staton 2008].

However, despite its generality and power, functorial operational semantics, in particular in its variants with variable binding, has not caught on as widely as one might have hoped among more practical operational semanticists. A first reason, already noticed by Staton, is that it is "too abstract". Specifically, although it is very powerful for proving general results, "one must really squint hard to view a distributive law as a collection of rules" [Staton 2008]. Otherwise said, functorial operational semantics is better at reasoning in the large than in the small. Another possible reason for functorial operational semantics not being more widely adopted might be that we should distinguish between two different goals, both of which are crucial for a theory of structural operational semantics to be adopted by the community: (1) the first goal is a high-level abstract language for *reasoning* on structural operational semantics, as close as possible to concrete intuitions; (2) the second is a generic toolbox for *producing* well-behaved structural operational semantics from more basic data. It seems fair to say that, until now, most work on the first goal was done with a view to the second one, i.e., with formats in mind. In this paper, instead, our aim is to find the right level of generality, by proposing a new theory of structural operational semantics which attempts to get closer to operational intuitions, while still proving useful abstract results – specifically, congruence of bisimilarity and soundness of bisimulation up to context.

1.2 Contributions

Our first contribution is the introduction of *transition categories*, which are categories equipped with a distinguished class of cospans, thought of as the set of transition labels. Transition categories support a notion of bisimulation defined by lifting, much as in Joyal et al. [1993]. A structural operational semantics specification is then a monad $\mathscr{T}$ on the considered transition category, which embodies the syntax and proof rules for transitions. Models of the specification $\mathscr{T}$ are thus simply $\mathscr{T}$-algebras. The Kleisli category of $\mathscr{T}$ is a high-level, flexible environment for reasoning about terms and (partial) transition proofs, where composition models plugging partial proofs (and terms) into one another. The approach thus lends itself to reasoning in the small. Our first main result (Corollary 4.30) is about reasoning in the large: it states that whenever $\mathscr{T}$ satisfies a certain *familiality* property [Carboni and Johnstone 1995; Diers 1978; Garner and Hirschowitz 2018; Weber 2007] 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 (between states of X and Y) is a congruence.

We then turn to soundness of bisimulation up to context in §5. Bisimulation up to context requires the introduction of an intermediate notion, *pre-bisimulation*, which, instead of relying on lifting, paraphrases the standard definition of bisimulation. We relate pre-bisimulation to bisimulation by showing that it also may be defined by lifting, though in the cospan category of the ambient transition category. Furthermore, under a mild additional hypothesis and using a suitable weak factorisation system, we show that any pre-bisimulation embeds into some bisimulation. We then define *pre-bisimulation up to context*, which agrees with standard bisimulation up to context in examples. Our second main result (Corollary 5.15) is that, up to a refinement of the familiality hypothesis, any pre-bisimulation up to context embeds into some bisimulation, i.e., pre-bisimulation up to context is sound. This is, to our knowledge, the first categorical soundness result for bisimulation up to context covering calculi with variable binding.

Finally, we demonstrate the flexibility of our framework in §6 by analysing the failure of bisimilarity to be a congruence in the π-calculus. We first pinpoint where the hypotheses of Corollary 4.30 fail, namely: the structure map $\mathscr{T}(Pi) \to Pi$ is not a functional bisimulation. We then recast in our setting two standard ways of working around this issue: (1) prove the weaker claim that bisimilarity is a *non-input congruence* [Sangiorgi and Walker 2001]; (2) restrict attention to a more constrained notion, *wide-open* bisimulation [Fiore and Staton 2006; Sangiorgi and Walker 2001; Staton 2008], to which Corollary 4.30 applies.

1.3 Related Work

We know of only two other abstract accounts of structural operational semantics covering both syntax and models ($\approx$ categorical accounts), and proving congruence of bisimilarity: functorial operational semantics and Staton [2008]. The clearest novelty of our approach compared to them is that, to our knowledge, they do not cover soundness of bisimulation up to context in the presence of binding. Another distinctive feature is the crucial role of familiality. In particular, Staton [2008, Theorem 12] is very close in spirit to our Corollary 4.30, but beyond the fact that it lives in a different setting, it merely assumes that the considered monad preserves functional bisimulations, while familiality allows us to prove it. Less closely related work includes presheaf models and their generalisations [Cattani et al. 1998; Joyal et al. 1993], which emphasise semantical, rather than operational aspects. Furthermore, their representation of labelled transition systems as presheaves markedly differs from ours, in that, e.g., any finite but cyclic labelled transition system may be finitely represented in our approach, while it has to be represented by an infinite presheaf in theirs. Our approach is also more economical for defining bisimulation, by only lifting against one morphism per label, instead of one morphism per trace extension. Finally, our $\mathbf{T}_s$-familial monads are related to *cellular* analytic functors [Garner and Hirschowitz 2018], and the relationship between our pre-bisimulations and bisimulations (§5.2) is closely related to Dubut et al. [2016].

1.4 Plan

We start in §2 with a non-technical overview of the new framework, trying to show that it is more practical and intuitive than previous approaches. In §3, relying on the examples of combinatory logic and CCS, we recast structural operational semantics and congruence in the setting of monads on transition categories. In §4, we prove that compositionality and familiality entail congruence of bisimilarity. In §5, we investigate soundness of bisimulation up to context. In §6, we analyse bisimilarity in the π-calculus using our framework. Finally, we conclude in §7.

1.5 Notation and Preliminaries

We assume basic familiarity with category theory [Mac Lane 1998]. For any 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 $X(f) \colon X(c') \to X(c)$ is denoted by $x \mapsto x \cdot f$. The Yoneda embedding is denoted by $\mathbf{y} \colon \mathbb{C} \to \widehat{\mathbb{C}}$, and often left implicit.

We denote by $el(X)$ the *category of elements* [Mac Lane and Moerdijk 1992] 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 $f \restriction x'$. Furthermore, we often abbreviate (c, x) to x.

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

2 OVERVIEW

2.1 Bisimulation by Lifting

Before delving into details, let us give a non-technical overview. A transition category is much like the category of labelled transition systems (over a fixed set of labels), with functional simulations as morphisms. Sticking to the untyped case for simplicity, there is thus an object, say $\star$, consisting of just one state. Similarly, for each label α, there is an object consisting of just one α-transition between two states. Thus, giving a morphism $\star \to X$ to any object X is equivalent to choosing a state in X, and likewise giving a morphism $e\colon \alpha \to X$ is equivalent to choosing an α-transition in X. Furthermore, taking the source of e is just pre-composing with the source morphism $s\colon \star \to \alpha$. So the source of e is $e \circ s$. Symmetrically, its target is $\star \xrightarrow{t} \alpha \xrightarrow{e} X$.

Now, we may define bisimulation diagrammatically, as follows. A relation $R \hookrightarrow X \times Y$ is a bisimulation iff both projections $p\colon R \to X$ and $q\colon R \to Y$ are *functional bisimulations*, where, e.g., p is a functional bisimulation iff for all labels α and commuting squares as below left, there is a lifting k that makes both triangles commute.

$$\begin{tikzcd} \star \arrow[r, "v"] \arrow[d, "s"'] & R \arrow[d, "p"] \\ \alpha \arrow[r, "e"'] \arrow[ur, dashed, "k"] & X \end{tikzcd} \qquad \begin{tikzcd} v = (x,y) \arrow[r, mapsto, "p"] \arrow[d, dashed, "k"'] & x \arrow[d, "e"] \\ (x',y') \arrow[r, dashed, "p"'] & x' \end{tikzcd} \tag{1}$$

Indeed, as we saw, v denotes a state in R, i.e., a related pair (x,y), e an α-transition in X, and commutation of the square says that $p(v) = x$ is the source of e, so that $e\colon x \xrightarrow{\alpha} x'$. We are thus in a situation like above right. Existence of k then says that there is an α-transition k in R with source v, mapped by p to e, i.e., a pair $(e,f)\colon (x,y) \xrightarrow{\alpha} (x',y')$ of related transitions as above right, just as in the standard definition of bisimulation.

2.2 Structural Operational Semantics Specifications as Monads

The next step is to view structural operational semantics specifications as monads $\mathscr{T}$ on the considered transition category. A specification consists of proof rules for inductively constructing transitions. Intuitively, $\mathscr{T}(X)$ is obtained by saturating its argument X by the considered proof rules, i.e., it augments X with new, formal transitions constructed using the rules. A model for the considered structural operational semantics specification is thus merely a $\mathscr{T}$*-algebra*, i.e., an object X respecting the rules, formally a map $\mathscr{T}(X) \to X$ satisfying certain conditions.

The Kleisli category of $\mathscr{T}$ provides a high-level, flexible environment for manipulating terms and (partial) proofs, including plugging them into one another, at a fine level which is not directly available in functorial operational semantics.

Remark 2.1. This is not really surprising: monads are a generalisation of algebraic theories, and the Kleisli category (or rather its opposite) is the corresponding generalisation [Berger et al. 2012] of Lawvere's [1963] syntactic category.

Example 2.2. In the case of labelled transition systems, if $\mathscr{T}$ is the monad for CCS terms and rules: morphisms $\star \to \mathscr{T}(X)$ are in one-to-one correspondence with CCS terms with free variables (a.k.a. constants, or indeterminates) in X. Similarly, for any label α, morphisms $\alpha \to \mathscr{T}(X)$ correspond to proofs of α-transitions with variables in X, meaning that transitions in X are treated as axioms in the proof system. E.g., consider a labelled transition system $X \cong \overline{a} + a$ consisting of

just two transitions $e\colon x \xrightarrow{\bar{a}} x'$ and $f : y \xrightarrow{a} y'$. Then, the left-hand proof below yields a morphism $\tau \to \mathscr{T}(X)$, and the right-hand one yields a morphism $\bar{a} \to \mathscr{T}(X)$.

$$\dfrac{\dfrac{}{x \xrightarrow{\bar{a}} x'}\,e \qquad \dfrac{}{y \xrightarrow{a} y'}\,f}{(x|y) \xrightarrow{\tau} (x'|y')} \qquad\qquad \dfrac{\dfrac{\dfrac{}{x \xrightarrow{\bar{a}} x'}\,e}{(x|y) \xrightarrow{\bar{a}} (x'|y)}}{(x|y|y) \xrightarrow{\bar{a}} (x'|y|y)} \tag{2}$$

The left-hand proof is *linear*, i.e., each bit of X is used exactly once. In such cases, the identity of these bits does not really matter, so we may represent proofs a bit more abstractly as *transition contexts*, i.e., partial proof trees, as below left.

$$\dfrac{\square_1 \xrightarrow{\bar{a}} \square_1' \qquad \square_2 \xrightarrow{a} \square_2'}{(\square_1|\square_2) \xrightarrow{\tau} (\square_1'|\square_2')} \qquad\qquad \dfrac{\dfrac{\square}{\bar{a}.\square \xrightarrow{\bar{a}} \square} \qquad \square_2 \xrightarrow{a} \square_2'}{(\bar{a}.\square|\square_2) \xrightarrow{\tau} (\square|\square_2')} \tag{3}$$

Such a transition context $E\colon \tau \to \mathscr{T}(\bar{a} + a)$ may be thought of as a term of type τ with two free variables of respective types $\bar{a}$ and a. Let us now assume that one wants to instantiate, say, the first variable in E with the output rule $\bar{a}.\square \xrightarrow{\bar{a}} \square$ (which still depends on a process variable, $\square$). Then they may compose E in the Kleisli category with the morphism

$$\bar{a} + a \xrightarrow{output+\eta} \mathscr{T}(\star) + \mathscr{T}(a) \xrightarrow{[\mathscr{T}(in_l),\mathscr{T}(in_r)]} \mathscr{T}(\star + a),$$

where $\eta\colon a \to \mathscr{T}(a)$ just picks the given variable, *output* is the map $\bar{a} \to \mathscr{T}(\star)$ corresponding to the output axiom, and in_l and in_r denote the coproduct injections $\star \hookrightarrow \star + a \hookleftarrow a$. Concretely, the composite in the Kleisli category is the transition context on the right in (3), where we explicitly leave $\square$ as an open-ended branch to emphasise that it might be further instantiated with some (potentially open) process $P\colon \star \to \mathscr{T}(Y)$.

2.3 Congruence of Bisimilarity, Contexts, and Familiality

Now that we have explained why monads on transition categories model structural operational semantics specifications and how terms and partial transition proofs may be manipulated in the Kleisli category, let us consider congruence of bisimilarity. Essentially, this amounts to the fact that given any bisimulation R relating the states of labelled transition systems X and Y, the relation consisting of all pairs of the form $(C[x_1, ..., x_n], C[y_1, ..., y_n])$, for any context C with $(x_i, y_i) \in R$ for all i, is again a bisimulation. The standard, operational approach for proving this [Bernstein 1998; Bloom et al. 1995; Bol and Groote 1996; Middelburg 2001; Mousavi et al. 2005; Sangiorgi 1994; Sangiorgi and Walker 2001] goes by induction on C, but the key underlying intuition is that any transition proof $e\colon C[x_1, ..., x_n] \xrightarrow{\alpha} x'$ decomposes into a partial transition proof by the context C and transition proofs by some x_is. The categorical framework provides a high-level language to describe this, which leads to a more algebraic proof (by computation instead of by induction).

In our case, the key lemma will state that given algebras $a\colon \mathscr{T}(X) \to X$ and $b\colon \mathscr{T}(Y) \to Y$, together with a bisimulation R as above, $\mathscr{T}(R)$ is again a bisimulation, in the sense that the composite $\mathscr{T}(R) \xrightarrow{\mathscr{T}(p)} \mathscr{T}(X) \xrightarrow{a} X$ and its symmetric variant with Y satisfy the same lifting property (1)

as p above. First of all, it suffices to prove that both a and $\mathscr{T}(p)$ are functional bisimulations, so we consider them separately. The general framework does not have much to say about the former: as we will see in §4.1, it intuitively corresponds to the considered algebra being compositional.

For $\mathscr{T}(p)$, the first step is to be able to talk about *contexts*, i.e., terms which make linear use of their free variables. Among morphisms in the Kleisli category, contexts will be those satisfying a universal property called *genericness*, described at the beginning of §4.2. The monad $\mathscr{T}$ is *familial* just when any morphism $X \to \mathscr{T}(Y)$ canonically factors as

$$X \xrightarrow{C} \mathscr{T}(A) \xrightarrow{\mathscr{T}(f)} \mathscr{T}(Y)$$

with C generic, which corresponds to the standard fact that any term with free variables in Y decomposes as a context, here C, equipped with a map from its holes to Y, here f. The morphism $\mathscr{T}(A) \to \mathscr{T}(Y)$, being in the image of $\mathscr{T}$, is called *free*, and we generally refer to such factorisations as *generic-free*. The middle object A is called the *arity* of C.

Example 2.3. In the case of CCS, for any variable $x\colon \star \to X$, the term $(x|x)$ factors as

$$\star \xrightarrow{C} \mathscr{T}(\star + \star) \xrightarrow{\mathscr{T}[x,x]} \mathscr{T}(X),$$

where C is the context $(\square_1|\square_2)$. General generic transitions $\alpha \to \mathscr{T}(A)$ are essentially the transition contexts of Example 2.2.

The factorisation property of familial monads applies uniformly to terms, transitions, and more (i.e., any object). An important feature of generic-free factorisations in our examples is that they also prescribe how to compute the source and target of transition contexts. The source of a transition, i.e., of a (not necessarily generic) morphism $E\colon \alpha \to \mathscr{T}(Y)$, is just the composite $\star \xrightarrow{s} \alpha \xrightarrow{E} \mathscr{T}(Y)$. So if we start with a generic E, and take the generic-free factorisation of $E \circ s$, we get a commuting square

$$\begin{tikzcd} \star \arrow[r, "s"] \arrow[d, "C"'] & \alpha \arrow[d, "E"] \\ \mathscr{T}(A) \arrow[r, "\mathscr{T}(f)"'] & \mathscr{T}(Y). \end{tikzcd}$$

Example 2.4. When E is as on the left of (3), factoring $E \circ s$ yields $C := (\square_1|\square_2)\colon \star \to \mathscr{T}(\star + \star)$ with $f = s + s\colon \star + \star \to \bar{a} + a$. Please note that when E is instantiated, say with $h\colon \bar{a} + a \to Z$, f gives the desired instantiation of C, namely the composite $\star + \star \xrightarrow{s+s} \bar{a} + a \xrightarrow{h} Z$, which correctly models the fact that the source of any transition $z_1|z_2 \xrightarrow{\tau} z_1'|z_2'$ of this form is indeed $z_1|z_2$.

Returning to the proof that $\mathscr{T}(p)$ is a functional bisimulation when p is, we need to prove that any commuting square as the exterior below admits a lifting. Using familiality of $\mathscr{T}$, we take generic-free factorisations of both e and r (morally: the given transition and pair of terms, respectively), as in the solid part of

$$\begin{tikzcd} \star \arrow[rr, "r"] \arrow[dr, "C"] \arrow[dd, "s"'] & & \mathscr{T}(R) \arrow[dd, "\mathscr{T}(p)"] \\ & \mathscr{T}(A) \arrow[ur, "\mathscr{T}(k)"'] \arrow[d, dashed, "\mathscr{T}(l)"'] & \\ & \mathscr{T}(Y) \arrow[uur, dashed, "\mathscr{T}(j)"'] \arrow[dr, "\mathscr{T}(h)"] & \\ \alpha \arrow[ur, "E"] \arrow[rr, "e"'] & & \mathscr{T}(X). \end{tikzcd} \tag{4}$$

Genericness of C (see §4.2 for details) then yields l making both squares commute (with $p \circ k = h \circ l$). Intuitively, the left-hand square should be just as in Example 2.4, so l should merely be a coproduct of source maps. Because p is a functional bisimulation, we should thus find a lifting for each one of them, hence j by cotupling, which yields the desired lifting $\mathscr{T}(j) \circ E$. In order to enforce this property that the obtained l still admits liftings w.r.t. functional bisimulations, we need to impose a further hypothesis on $\mathscr{T}$, which is essentially a specialisation of *cellularity* [Garner and Hirschowitz 2018]. This is called $\mathbf{T}_s$-familiality and explained in detail in §4.3.

2.4 Summary

We hope that the present overview demonstrates that the Kleisli category of a monad on a transition category is a high-level, flexible algebraic framework for reasoning about structural operational semantics. It is finer-grained than functorial operational semantics, in the sense that it directly models terms and (partial) transition proofs. Yet, it is a convenient language for proving properties like congruence of bisimilarity. Notably, the language of familial monads brings us closer to intuitions underlying the standard proof by induction over context, and proceed algebraically instead of logically. Similar benefits are observed for bisimulation up to context in §5, using much the same language except that lifting is done in the more complex category of cospans in the ambient transition category. This makes the development a bit more verbose, so we refrain from exposing it here, and move on to the technical development.

3 STRUCTURAL OPERATIONAL SEMANTICS SPECIFICATIONS AS MONADS

In this section, we essentially fill in the technical details of §2, up to the abstract statement of congruence of bisimilarity. We first explain how labelled transition systems may be viewed as presheaves, and how bisimulation may be defined by lifting (§3.1). We then abstract over this to define transition categories and bisimulation therein. We then show how structural operational semantics specifications naturally yield monads (§3.2), an observation originally due to Staton [2008]. We finally define congruence in this setting.

3.1 Labelled Transition Systems as Objects in Transition Categories

Let us first view labelled transition systems as presheaves in concrete examples, and then abstract over what we did and define transition categories and bisimulation therein.

3.1.1 Labelled Transition Systems. The simplest kind of labelled transition system, the one with just one label, is suitably modelled by **Gph**, the category of (directed, multi) graphs, i.e., presheaves over the category $\bigstar \underset{t}{\overset{s}{\rightrightarrows}} [1]$ (provided one accepts the extra generality of allowing distinct, parallel transitions between nodes). More generally, let us show that transition systems labelled in a set L may also be viewed as presheaves. As a first step, they may be defined as graph morphisms to the graph A_L with one vertex and one endo-edge for each label $l \in L$. So the relevant category is the slice $\mathbf{Gph}/A_L$. But as is well known, for any A, $\mathbf{Gph}/A$ is equivalent to $\widehat{el(A)}$, presheaves over the category of elements of A.

Example 3.1. In CCS, labels are elements of $L_{CCS} = \{\tau\} \cup \bigcup_{a \in \mathscr{N}} \{a, \overline{a}\}$, where $\mathscr{N}$ denotes a fixed, infinite set of *names*, for example the natural numbers $\mathbb{N}$. The relevant base category $\mathbb{C}_{CCS} := el(A_{L_{CCS}})$ is freely generated by the graph with vertices in $\{\bigstar\} \uplus L_{CCS}$, plus, for all $\alpha \in L_{CCS}$, two edges $\bigstar \xrightarrow{s \upharpoonright \alpha} \alpha \xleftarrow{t \upharpoonright \alpha} \bigstar$, which we often abbreviate to s and t. E.g., the labelled transition system $x \xleftarrow{\overline{a}} y \underset{b}{\overset{b}{\rightrightarrows}} z \circlearrowleft a$ is modelled by the presheaf X with

$$X(\star) = \{x, y, z\} \qquad \begin{array}{c} X(\overline{a}) = \{e\} \\ X(b) = \{f, f'\} \\ X(a) = \{g\} \end{array} \qquad \begin{array}{c} x = e \cdot t \\ y = e \cdot s = f \cdot s = f' \cdot s \\ z = f \cdot t = f' \cdot t = g \cdot s = g \cdot t. \end{array}$$

3.1.2 Bisimulation. Having seen how presheaves on $\mathbb{C}_{CCS}$ generalise labelled transition systems, let us now define functional bisimulation by lifting, as sketched in the overview. We will use the following efficient, standard notation [Riehl 2014]. In any category $\mathscr{A}$, a *lifting problem* for morphisms f and g is a commuting square as the solid part below. A *solution* to the lifting problem is a lifting as shown (dashed) making both triangles commute.

$$\begin{tikzcd} A \arrow[r] \arrow[d, "f"'] & C \arrow[d, "g"] \\ B \arrow[r] \arrow[ur, dashed] & D \end{tikzcd}$$

Definition 3.2. Let $f \boxslash g$ iff all lifting problems for f and g have at least one solution.

When $f \boxslash g$, we say that g has the *right lifting property* w.r.t. f, or symmetrically that f has the *left lifting property* w.r.t. g. We moreover let $f^{\boxslash}$ 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 $\mathscr{D}^{\boxslash} = \bigcap_{f \in \mathscr{D}} f^{\boxslash}$. We define ${}^{\boxslash}g$ and ${}^{\boxslash}\mathscr{D}$ symmetrically.

Proposition 3.3. *A morphism $p\colon R \to X$ in $\widehat{\mathbb{C}_{CCS}}$ is a functional bisimulation (according to the standard definition) iff it has the right lifting property w.r.t. all maps of the form $s\colon \star \to \alpha$ for α a label, i.e., iff $p \in \mathbf{T}_s^{\boxslash}$, where $\mathbf{T}_s$ denotes the set of all maps $s\colon \star \to \alpha$.*

Indeed, this is just as in §2.1, which should now make technical sense.

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

Proposition 3.4. *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 $f : A \to B$ is epi, so is the set map $\widehat{\mathbb{C}_{CCS}}(\star, A) \xrightarrow{\widehat{\mathbb{C}_{CCS}}(\star, f)} \widehat{\mathbb{C}_{CCS}}(\star, B)$. Thus every lifting problem for any $s\colon \star \to \alpha$ and $im(R) \overset{m}{\hookrightarrow} X \times Y \xrightarrow{\pi} X$ induces a lifting problem for s and the composite $\pi \circ m \circ e$. The latter has a lifting by hypothesis, which yields a lifting for $\pi \circ m$. Everything works symmetrically for the projection to Y. □

Definition 3.5. Isomorphism classes of monic bisimulations, i.e., subobjects of $X \times Y$ that are bisimulations, are called *bisimulation relations.*

Bisimulation relations are ordered by inclusion and for any set I we may define the union of relations $R_i \hookrightarrow X \times Y$, for $i \in I$, as the image of their (wide) cotupling $\sum_i R_i \to X \times Y$. We have:

Proposition 3.6. *Bisimulation relations are closed under union, and admit a maximum, called* bisimilarity.

Proof. As before the covariant hom-functor of $\star$ preserves epis. Thus, every lifting problem for any $s\colon \star \to \alpha$ and a given union $\bigcup_i R_i \hookrightarrow X \times Y \xrightarrow{\pi} X$ yields one for some $R_i \hookrightarrow X \times Y \to X$, which has a lifting by hypothesis. Finally, presheaf categories are *well-powered*, i.e., each object has only a set of subobjects. We thus in particular only have a set of bisimulation relations. This set being closed under unions, it has a maximum. □

3.1.3 Transition Categories. Let us now abstract away from the particular example of $\widehat{\mathbb{C}_{CCS}}$. The structure and properties we need for deriving abstract analogues of Propositions 3.4 and 3.6 are:

Definition 3.7. A *transition category* is a cocomplete, finitely complete, well-powered category $\mathscr{A}$ with images (i.e., initial factorisations through some mono), equipped with

- two sets $\mathbf{P}$ and $\mathbf{L}$ of objects called *process types* and *label types*, respectively, and
- 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 process types are *tiny*, i.e., their covariant hom-functors preserve colimits.

Notation 3.8. We generally denote a transition category by just $\mathscr{A}$, leaving $\mathbf{P}$, $\mathbf{L}$, and $\mathbf{T}$ implicit.

Remark 3.9. In any initial factorisation $A \xrightarrow{e} B \overset{m}{\hookrightarrow} C$ with m mono, e is epi, because if $f \circ e = g \circ e$ then e factors through the equaliser of f and g, which thus has to be an iso [Johnstone 2002, p20].

Remark 3.10. Tininess entails that covariant hom-functors of process types preserve epis.

In examples, tininess follows from:

Remark 3.11. If $\mathscr{A}$ is a presheaf category, then all representable presheaves are tiny.

Let us now abstractly replay the above development of bisimulation and bisimilarity.

Definition 3.12. 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 $X \xleftarrow{s} R \xrightarrow{t} Y$, or equivalently a map $R \xrightarrow{\langle s,t \rangle} X \times Y$ such that s and t are both functional bisimulations. Isomorphism classes of monic bisimulations are called *bisimulation relations*.

Proposition 3.13. *In any transition category $\mathscr{A}$, bisimulations are closed under images, i.e., if $r : R \to X \times Y$ is a bisimulation, then so is m in its epi-mono factorisation $R \overset{e}{\twoheadrightarrow} im(R) \overset{m}{\hookrightarrow} X \times Y$.*

Proposition 3.14. *In any transition category, bisimulation relations $R \hookrightarrow X \times Y$ are closed under unions, hence admit a maximum, called* bisimilarity *and denoted by $\sim_{X,Y}$, or simply $\sim$ when X and Y are clear from context.*

As expected we have:

Example 3.15 (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 3.16 (CCS labels). The category $\widehat{\mathbb{C}_{CCS}}$ forms a transition category with $\star$ as only process type, all other representables as label types, and all cospans $\star \xrightarrow{s} \alpha \xleftarrow{t} \star$ as transition types.

3.2 Specifications as Monads on Transition Categories

In the previous section, we have defined transition categories and bisimulation therein, which abstract over standard labelled transition systems. 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.

3.2.1 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.

Let $\mathscr{T}_{CL}$ denote the functor on **Gph** mapping any graph G to the one with

- as vertices all terms generated by the grammar $M, N ::= (\!|x|\!) \mid S \mid K \mid M\,N$, where x ranges over $G(\star)$, S and K are constants, and $M\,N$ stands for the application of a binary function symbol (called "application") to M and N;
- edges inductively defined by the following rules, with the given sources and targets.

$$\frac{e \in G(x,y)}{(\!|e|\!) : (\!|x|\!) \to (\!|y|\!)} \qquad \frac{}{s_{M,N,P} : S\,M\,N\,P \to (M\,N)\,(M\,P)} \qquad \frac{}{k_{M,N} : K\,M\,N \to M}$$

$$\frac{L : M \to M'}{L\,N : M\,N \to M'\,N} \qquad \frac{R : N \to N'}{M\,R : M\,N \to M\,N'}$$

The action of $\mathscr{T}_{CL}$ on morphisms is by applying the given graph morphism to vertices $(\!|x|\!)$ and edges $(\!|e|\!)$. This functor is a monad whose multiplication (a.k.a. substitution) merely removes the top layer of $(\!|-|\!)$s. It is inductively defined by

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

Example 3.17. Let $M = (\!|M_1|\!)\,(\!|M_2|\!)$, with $M_1, M_2 \in \mathscr{T}_{CL}(0)(\star)$. Then, $\mu_{0,\star}(M) = M_1\,M_2$. Similarly, for $k_{P,Q} \in \mathscr{T}_{CL}(0)(K\,P\,Q, P)$, we have $\mu_{0,[1]}((\!|k_{P,Q}|\!)\,M) = k_{P,Q}\,(M_1\,M_2)$.

The free algebra $\mathscr{T}_{CL}(0)$ is a proof-relevant variant of combinatory logic.

3.2.2 Example with Labels: CCS. As a second example, useful for illustrating labels, let us consider CCS, recalling the base category $\mathbb{C}_{CCS}$ from §3.1.1.

Definition 3.18. 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 variables in $G(\star)$, i.e., generated by the grammar $P, Q ::= (\!|x|\!) \mid 0 \mid a.P \mid \bar{a}.P \mid (P|Q) \mid \nu a.P$, with x ranging over $G(\star)$, *not* considered equivalent up to renaming of bound names in $\nu a.P$ (α-equivalence is not necessary for CCS; by contrast, it is necessary for π-calculus, which forces us to use a more complex base category);
- for all $\alpha \neq \star$, $\mathscr{T}_{CCS}(G)(\alpha)$ is the set of transition proofs $P \xrightarrow{\alpha} Q$, much as in Boudol and Castellani [1988], inductively generated by the following rules.

$$\frac{e \in G(\alpha)}{(\!|e|\!) : (\!|e \cdot s|\!) \xrightarrow{\alpha} (\!|e \cdot t|\!)} \qquad \frac{}{out^a_P : \bar{a}.P \xrightarrow{\bar{a}} P} \qquad \frac{}{in^a_P : a.P \xrightarrow{a} P} \qquad \frac{R : P \xrightarrow{\alpha} Q \qquad \alpha \notin \{a, \bar{a}\}}{\nu a.R : \nu a.P \xrightarrow{\alpha} \nu a.Q}$$

$$\frac{L : P \xrightarrow{\alpha} P'}{(L|Q) : (P|Q) \xrightarrow{\alpha} (P'|Q)}(+\text{ symmetric } Q|L) \qquad \frac{L : P \xrightarrow{\bar{a}} P' \qquad R : Q \xrightarrow{a} Q'}{L \rhd R : (P|Q) \xrightarrow{\tau} (P'|Q')}(+\text{ sym. } R \lhd L)$$

Again substitution equips $\mathscr{T}_{CCS}$ with monad structure, and the free algebra $\mathscr{T}_{CCS}(0)$ is a proof-relevant variant of CCS.

3.2.3 Abstract Congruence. 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 to be a congruence, and then give an abstract definition in the setting of algebras for a monad on a transition category.

Standardly, given a structural operational semantics specification X, we say that a relation R is a congruence when for all multi-hole contexts C and pairs $(x_1, y_1), ..., (x_n, y_n)$ of related processes, $C[x_1, ..., x_n]$ and $C[y_1, ..., y_n]$ are again related. 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 3.19. Given a monad $\mathscr{T}$ on a transition category $\mathscr{A}$, $\mathscr{T}$-algebras $a\colon \mathscr{T}(X) \to X$ and $b\colon \mathscr{T}(Y) \to Y$, and a relation $R \hookrightarrow X \times Y$, we say that R is a *congruence* when the map $\mathscr{T}(R) \to \mathscr{T}(X \times Y) \xrightarrow{\langle \mathscr{T}(\pi), \mathscr{T}(\pi') \rangle} \mathscr{T}(X) \times \mathscr{T}(Y) \xrightarrow{a \times b} X \times Y$ factors through $R \hookrightarrow X \times Y$.

Proposition 3.20. *When $R \hookrightarrow X \times Y$ is a congruence, R is a $\mathscr{T}$-algebra and $R \hookrightarrow X \times Y$ is a morphism of $\mathscr{T}$-algebras.*

Remark 3.21. What does factorisation through R have to do with being a congruence? Intuitively, an element in $\mathscr{T}(R)$ has the form $C[(x_1, y_1), \dots, (x_n, y_n)]$, with $(x_i, y_i) \in R$ for all i. It is mapped by $\langle \mathscr{T}(\pi), \mathscr{T}(\pi') \rangle$ to the pair $(C[x_1, \dots, x_n], C[y_1, \dots, y_n]) \in \mathscr{T}(X) \times \mathscr{T}(Y)$, and $a \times b$ evaluates this back to $X \times Y$. Thus, the composite factoring through R precisely means that $a(C[x_1, \dots, x_n])$ and $b(C[y_1, ..., y_n])$ are related, i.e., R is a congruence.

4 CONGRUENCE OF BISIMILARITY

In the previous sections, we have explained in which sense monads on transition categories model structural operational semantics specifications, and defined congruence in this setting. We now turn to proving congruence of bisimilarity, by filling in the details missing from the overview. We consider compositionality in §4.1, familiality in §4.2, and $\mathbf{T}_s$-familiality in §4.3. Relying on the theory of weak factorisation systems, we are then able to prove congruence of bisimilarity.

4.1 Compositionality

In the overview, our proof sketch for congruence of bisimilarity started by assuming that the structure map $a\colon \mathscr{T}(X) \to X$ of the considered algebra was a functional bisimulation. Concretely, for any square as below, there exists a lifting k making both triangles commute.

$$\begin{tikzcd} P \arrow[r, "v"] \arrow[d, "\mathbf{T}_s \ni s"'] & \mathscr{T}(X) \arrow[d, "a"] \\ L \arrow[r, "r"'] \arrow[ur, dashed, "k"] & X \end{tikzcd}$$

Thinking of v as a process with variables in X, i.e., a context applied to some processes in X, of $a \circ v$ as its evaluation in X, and of r as a transition from $a \circ v$, this says that r may be decomposed as the evaluation $a \circ k$ of some transition proof k with variables in X and domain $k \circ s = v$. This leads us to:

Definition 4.1. A $\mathscr{T}$-algebra $a\colon \mathscr{T}(X) \to X$ is *compositional* iff $a \in \mathbf{T}_s^{\boxslash}$.

Notation 4.2. We often denote algebras by their carriers, e.g., X is compositional.

Example 4.3. It might be instructive to exhibit a simple example of a non-compositional algebra. Consider the monad $\mathscr{T}$ on **Gph** whose terms are freely generated by a constant c and two unary operations f and g, and whose transitions are freely generated by the axiom below left.

$$\overline{f(g(x)) \to x} \qquad\qquad \begin{array}{ccc} f[g(c)] & \longmapsto & f(g(c)) \\ & & \downarrow \\ & & c \end{array}$$

Clearly, the transition above right cannot be matched by $f[g(c)]$ because no rule has a conclusion starting from f alone. The free $\mathscr{T}$-algebra $\mathscr{T}(0)$ is thus not compositional, i.e., the monad multiplication μ is not in $\mathbf{T}_s^{\boxslash}$.

Proposition 4.4. *The free $\mathscr{T}_{CL}$-algebra $\mathscr{T}_{CL}(0)$ is compositional, i.e., $\mu_0 \in \mathbf{T}_s^{\boxslash}$.*

Proof sketch. (More detail in [Hirschowitz 2018].) Any $M \in \mathscr{T}_{CL}^2(0)$ is a process with variables in $\mathscr{T}_{CL}(0)$, i.e., it has the shape $C[M_1, ..., M_n]$. We need to be able to decompose any transition $R\colon \mu_{0,\star}(M) \to N$ into $E[e_1, ..., e_n]$ such that $\mu_{0,[1]}(E[e_1, ..., e_n]) = R$. We proceed by induction on C. E.g., if $C = C_1\ C_2$ and $R = L\ P$, then there exists $i \in n$ such that $L \cdot s = \mu_{0,\star}(C_1[M_1, ..., M_{i-1}])$ and $P = \mu_{0,\star}(C_2[M_i, ..., M_n])$. Then, by induction hypothesis, we find $E'[e'_1, ..., e'_{i-1}]$ such that $\mu_{0,[1]}(E'[e'_1, ..., e'_{i-1}]) = L$, so we can pick $E = E'\ C_2$. □

Proposition 4.5. *The $\mathscr{T}_{CCS}$-algebra $\mathscr{T}_{CCS}(0)$ is compositional.*

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

4.2 Familiality

Let us now turn to proving that $\mathscr{T}(f) \in \mathbf{T}_s^{\boxslash}$, starting with familiality. Here is the long-awaited definition of genericness:

Definition 4.6. Given any functor $\mathscr{F} : \mathscr{A} \to \mathscr{X}$, a morphism $\xi\colon X \to \mathscr{F}(A)$ is $\mathscr{F}$*-generic*, or *generic* for short, when for all commuting squares of the form

$$\begin{tikzcd} X \arrow[r, "\chi"] \arrow[d, "\xi"'] & \mathscr{F}(B) \arrow[d, "\mathscr{F}(g)"] \\ \mathscr{F}(A) \arrow[r, "\mathscr{F}(f)"'] \arrow[ur, dashed, "\mathscr{F}(h)"] & \mathscr{F}(C) \end{tikzcd} \tag{5}$$

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

Example 4.7. Let $\mathscr{T}$ denote the free monoid monad on sets, which associates to any set X the set $\sum_n X^n$ of sequences of elements of X. The generic-free factorisation associated to any sequence $l = (x_1, ..., x_n)$, viewed as a morphism $1 \to \mathscr{T}(X)$, is

$$1 \xrightarrow{(1,...,n)} \mathscr{T}(n) \xrightarrow{\mathscr{T}[x_i]_{i \in n}} \mathscr{T}(X),$$

where we recall that n denotes $\{1, ..., n\}$. Intuitively, the generic part retains only the 'shape' of l, or otherwise said the linear term from which l may be obtained by substitution.

Example 4.8. In the same setting, a non-generic morphism is the sequence $(1, 1)$, viewed as a morphism $1 \to \mathscr{T}(1)$. Indeed, the square

$$\begin{tikzcd} 1 \arrow[r, "(1,2)"] \arrow[d, "(1,1)"'] & \mathscr{T}(2) \arrow[d, "\mathscr{T}(!)"] \\ \mathscr{T}(1) \arrow[r, "\mathscr{T}(!)"'] & \mathscr{T}(1) \end{tikzcd}$$

commutes, with diagonal the sequence $(1,1)$, but no renaming will send $(1,1)$ to $(1,2)$. This example shows the tight connection between genericity and linearity.

Definition 4.9. A functor $\mathscr{F} : \mathscr{A} \to \mathscr{X}$ is *familial* when any morphism $X \to \mathscr{F}(A)$ factors as the composite of 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 4.10. By the pullback lemma, when the domain category has a terminal object, a natural transformation $\alpha\colon F \to G$ is cartesian iff its naturality squares of the form below are pullbacks.

$$\begin{tikzcd} FA \arrow[r, "F(!)"] \arrow[d, "\alpha_A"'] & F1 \arrow[d, "\alpha_1"] \\ GA \arrow[r, "G(!)"'] & G1 \end{tikzcd}$$

It should now be clear that the mediating arrow l in (4) follows from genericity of C.

Example 4.11. On **Set**, if the considered functor $\mathscr{T}$ is finitary, then arities A of generic operations $C\colon 1 \to \mathscr{T}(A)$ may be proved to be finite, so we may simply choose them to be the ordinal corresponding to the number n_C of holes in C. Familial functors thus coincide with standard polynomial functors [Kock 2011], as we have $\mathscr{T}(X) \cong \sum_C X^{n_C}$.

There is a slight generalisation of the formula of Example 4.11 to presheaf categories, which will be useful for showing that the monads $\mathscr{T}_{CL}$ and $\mathscr{T}_{CCS}$ of §3.2.1 and §3.2.2 are familial:

Lemma 4.12 ([Weber 2007, Remark 2.12]). *An endofunctor $\mathscr{T}$ on any presheaf category $\widehat{\mathbb{C}}$ is familial iff there is a functor $E\colon el(\mathscr{T}(1)) \to \widehat{\mathbb{C}}$ and a natural isomorphism (in X and c):*

$$\mathscr{T}(X)(c) \cong \sum_{x \in \mathscr{T}(1)(c)} \widehat{\mathbb{C}}(E(c,x), X). \tag{6}$$

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 $\mathbf{y}_c \to \mathscr{T}(X)$. Any functor of the form (6) is clearly pointwise familial: any map $(x, \varphi)\colon \mathbf{y}_c \to \mathscr{T}(X)$, with $\varphi\colon E(c,x) \to X$, factors as $\mathbf{y}_c \xrightarrow{(x,id)} \mathscr{T}(E(c,x)) \xrightarrow{\mathscr{T}(\varphi)} \mathscr{T}(X)$. Conversely, if $\mathscr{T}$ is pointwise familial, define $E(c,x)$ for any $x\colon \mathbf{y}_c \to \mathscr{T}(1)$ to be given by (any global choice of) generic-free factorisation of x: $\mathbf{y}_c \to \mathscr{T}(E(c,x)) \to \mathscr{T}(1)$. □

Proposition 4.13. *The monad $\mathscr{T}_{CL}$ is familial.*

Proof sketch. (More detail in [Hirschowitz 2018].) By Lemma 4.12, it suffices to exhibit a functor $E\colon el(\mathscr{T}_{CL}(1)) \to \mathbf{Gph}$, 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 terms on a unique free variable, say $\top$, and we define E to map any such term C to the discrete graph with vertices in the ordinal n_C, where n_C is the number of occurrences of $\top$ in C. Similarly, $\mathscr{T}_{CL}(1)[1]$ consists of transition proofs on just one transition axiom, say $\pi\colon \top \to \top$. On such proofs, we define E by induction:

$$E(\!|\pi|\!) = \mathbf{y}_{[1]}, \qquad E(k_{M,N}) = E(M) + E(N), \qquad E(L\,N) = E(L) + E(N), \ldots$$

E is then defined on $s \upharpoonright R$ and $t \upharpoonright R$ (recall §1.5), by straightforward induction. □

Proposition 4.14. *$\mathscr{T}_{CCS}$ is familial.*

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

4.3 Congruence of Bisimilarity And $\mathbf{T}_s$-Familiality

Let us now get to $\mathbf{T}_s$-familiality. In the overview, we obtained by familiality of $\mathscr{T}$ the mediating morphism l, which we argued should intuitively be a coproduct of source maps. This allowed us to find a lifting: we found one for each source map individually, and then took the cotupling. In the general case, however, there is *a priori* no guarantee that the mediating morphism will have such a nice form. Now, the only thing we need in order to conclude is that the mediating morphism be in ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$, so it is tempting to take this as an additional hypothesis. But the question is then whether this will be expressive enough to cover our examples. E.g., will ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$ always contain coproducts of maps in $\mathbf{T}_s$ and isomorphisms? This is where basic results from weak factorisation systems [Hovey 1999; Riehl 2014] come to the rescue.

4.3.1 Weak Factorisation Systems and ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$. Indeed, the point is that the classes of maps ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$ and $\mathbf{T}_s^{\boxslash}$ will form a *cofibrantly generated weak factorisation system.* Let us start with the most general notion:

Definition 4.15. A *weak factorisation system* on a category $\mathscr{A}$ consists of two classes of maps $\mathscr{L}$ and $\mathscr{R}$ such that $\mathscr{L}^{\boxslash} = \mathscr{R}$, $\mathscr{L} = {}^{\boxslash}\mathscr{R}$, and every map $f : A \to B$ factors as $A \xrightarrow{l} C \xrightarrow{r} B$ with $l \in \mathscr{L}$ and $r \in \mathscr{R}$.

Cofibrantly generated weak factorisation systems are those generated from a set of maps by lifting (this is the so-called *small object* argument). We introduce them in Proposition 4.19 below, which requires us to first define *transfinite* composition, *small* objects, and *relative cell complexes*:

Definition 4.16. For any ordinal λ, a *λ-sequence* is a cocontinuous functor from λ viewed as a category, to $\mathscr{A}$. A *transfinite composite* of any λ-sequence $X : \lambda \to \mathscr{A}$ is the component $\rho_0 : \lambda(0) \to colim_{\beta<\lambda} X(\beta)$ of any colimiting cocone ρ.

Definition 4.17. Let $\mathscr{J}$ denote any class of morphisms in a cocomplete category $\mathscr{A}$, and let κ denote any cardinal. An object $A \in \mathscr{A}$ is *κ-small* relative to $\mathscr{J}$ iff, for all κ-filtered [Hovey 1999, Definition 2.1.12] ordinals λ and λ-sequences $X : \lambda \to \mathscr{A}$ such that $X(\beta) \to X(\beta+1)$ is in $\mathscr{J}$ for all $\beta + 1 < \lambda$, the canonical map $colim_{\beta<\lambda}\mathscr{A}(A, X(\beta)) \to \mathscr{A}(A, colim_{\beta<\lambda} X(\beta))$ is bijective. We say that A is *small relative to* $\mathscr{J}$ iff it is κ-small relative to $\mathscr{J}$ for some κ.

Definition 4.18. For any class $\mathscr{J}$ of maps in a cocomplete category $\mathscr{A}$, let *$\mathscr{J}$-cell* denote the class of transfinite composites of pushouts of maps in $\mathscr{J}$, which we call *relative $\mathscr{J}$-cell complexes.*

Proposition 4.19 ([Hovey 1999, Theorem 2.1.14]). *For any set $\mathscr{J}$ of maps in a cocomplete category, if the domains of maps in $\mathscr{J}$ are small relative to $\mathscr{J}$-cell, then ${}^{\boxslash}(\mathscr{J}^{\boxslash})$ and $\mathscr{J}^{\boxslash}$ form a weak factorisation system. Any so obtained weak factorisation system is called* cofibrantly generated.

This applies to our abstract setting:

Proposition 4.20. *In any transition category, process types are small relative to $\mathbf{T}_s$-cell.*

Proof. Smallness of an object A means precisely that its covariant hom-functor preserves certain transfinite compositions, which holds for process types by tininess. □

We thus have by Proposition 4.19:

Corollary 4.21. *In any transition category,* ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$ *and* $\mathbf{T}_s^{\boxslash}$ *form a weak factorisation system.*

Cofibrantly generated weak factorisation systems enjoy an explicit characterisation of $\mathscr{L}$:

Definition 4.22. A *retract* of $f : X \to Y$ is any map $g\colon A \to B$ 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 4.23 ([Hovey 1999, Corollary 2.1.15]). *In the setting of Proposition 4.19, the left class* ${}^{\boxslash}(\mathscr{J}^{\boxslash})$ *consists precisely of retracts of relative* $\mathscr{J}$*-cell complexes.*

We thus have:

Corollary 4.24. *In any transition category, the classes of maps* $({}^{\boxslash}(\mathbf{T}_s^{\boxslash}), \mathbf{T}_s^{\boxslash})$ *form a weak factorisation system whose left class consists precisely of retracts of relative* $\mathbf{T}_s$*-cell complexes. In particular,* ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$ *contains coproducts of maps in* $\mathbf{T}_s$ *and of isomorphisms.*

4.3.2 Congruence of Bisimilarity. Being assured that ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$ is large enough for our purposes, let us now return to congruence of bisimilarity.

Definition 4.25. A monad $(\mathscr{T}, \eta, \mu)$ on a transition category $\mathscr{A}$ is $\mathbf{T}_s$*-familial* when it is familial and furthermore for any commuting diagram

$$\begin{tikzcd} P \arrow[r, "s"] \arrow[d, "C"'] & L \arrow[d, "D"] \\ \mathscr{T}(A) \arrow[r, "\mathscr{T}(s')"'] & \mathscr{T}(R) \end{tikzcd}$$

where $s \in \mathbf{T}_s$ and C and D are generic, we have $s' \in {}^{\boxslash}(\mathbf{T}_s^{\boxslash})$.

Remark 4.26. By Lemma 4.12, when $\mathscr{A}$ is a presheaf category and P and L are representable, this is equivalent to requiring that all maps of the form $s' \cong E(s \upharpoonright x)$ (for some $x \in \mathscr{T}(1)(L)$) are in ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$. Indeed, above, take for x the composite $L \xrightarrow{D} \mathscr{T}(R) \xrightarrow{\mathscr{T}(!)} \mathscr{T}(1)$: we then have $R \cong E(L, x)$, $A \cong E(P, x \cdot s)$, and $s' \cong E(s \upharpoonright x)$.

Theorem 4.27. *For any* $\mathbf{T}_s$*-familial monad* $\mathscr{T}$ *on a transition category, if* $f : R \to X$ *is a functional bisimulation, so is* $\mathscr{T}(R) \xrightarrow{\mathscr{T}(f)} \mathscr{T}(X)$.

Proof. We need to construct a lifting for any commuting square as the exterior of

$$\begin{tikzcd} P \arrow[rr, "p"] \arrow[dr, "C"'] \arrow[dd, "s"'] & & \mathscr{T}(R) \arrow[dd, "\mathscr{T}(f)"] \\ & \mathscr{T}(A) \arrow[ur, "\mathscr{T}(g)"'] \arrow[d, "\mathscr{T}(s')"] & \\ & \mathscr{T}(B) \arrow[uur, dashed, "\mathscr{T}(k)"'] \arrow[dr, "\mathscr{T}(h)"] & \\ L \arrow[ur, "D"] \arrow[rr, "r"'] & & \mathscr{T}(X). \end{tikzcd}$$

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 $s'\colon A \to B$ as shown, which is in ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$ by $\mathbf{T}_s$-familiality, so we obtain a lifting $k\colon B \to R$, and $\mathscr{T}(k) \circ D$ is a lifting for the whole diagram. □

We may now prove that bisimilarity is a congruence, based on the following well-known stability properties of factorisation systems.

Lemma 4.28 ([Riehl 2014, Lemma 11.1.4]). *For any weak 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.*

Corollary 4.29. *For any bisimulation $R \to X \times Y$ between compositional algebras for a $\mathbf{T}_s$-familial monad $\mathscr{T}$, the induced map $\mathscr{T}(R) \to X \times Y$ is a bisimulation.*

Proof. By compositionality, Theorem 4.27, and Lemma 4.28. □

Corollary 4.30. *Between any two compositional algebras for a $\mathbf{T}_s$-familial monad $\mathscr{T}$, bisimilarity is a congruence.*

In most examples, the considered algebra is the free one $\mathscr{T}(0)$. This is thus covered by:

Corollary 4.31. *Consider any $\mathbf{T}_s$-familial monad $\mathscr{T}$ on some transition category $\mathscr{A}$ such that $\mu_1 \colon \mathscr{T}^2(1) \to \mathscr{T}(1)$ 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 Lemma 4.28. We conclude by Corollary 4.30. □

Example 4.32. Corollary 4.31 applies to $\mathscr{T}_{CL}(0)$, $\mathscr{T}_{CCS}(0)$, and $\mathscr{T}^+_\pi(0)$ (see §6.3 below). Indeed, $\mathscr{T}_{CL}$ and $\mathscr{T}_{CCS}$ are both $\mathbf{T}_s$-familial (for different $\mathbf{T}_s$), because any s' obtained as in Definition 4.25 is a coproduct (in the arrow category) of isomorphisms and maps in $\mathbf{T}_s$, hence in ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$ by Lemma 4.28. Similarly, $\mathbf{T}_s$-familiality of $\mathscr{T}^+_\pi$ is Proposition 6.16 below.

However, Corollary 4.31 does not apply to either of the other two variants of the π-calculus that we consider in §6. Indeed, for $\mathscr{T}_\pi(0)$ (§6.1), μ_1 is not a functional bisimulation (this is the standard fact that bisimilarity is not a congruence), and in §6.2 the π-calculus is considered as an algebra for a certain submonad $\mathscr{T}^-_\pi$ of $\mathscr{T}_\pi$, not of the relevant form $\mathscr{T}^-_\pi(X)$, so we need to resort to the theorem rather than the corollary.

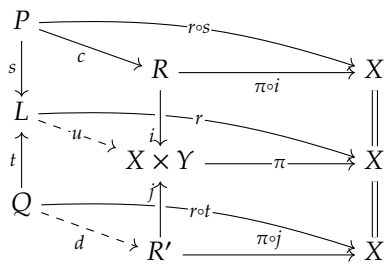
5 BISIMULATION UP TO CONTEXT

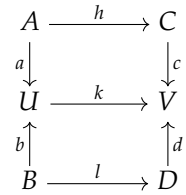
In this section, we consider an alternative notion of bisimulation in transition categories, *pre-bisimulation*, which we relate to bisimulation by showing that under a mild additional hypothesis any pre-bisimulation embeds into some bisimulation. We then define a notion of *pre-bisimulation up to context*, 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 Progression and Pre-Bisimulation

In Definition 3.12, we defined functional bisimulation through a lifting property, which is flexible enough to, e.g., exclude some transitions from the considered labelled transition systems X and Y. We will now define a slightly different notion which is in fact closer to the ordinary definition of bisimulation. A standard way to define bisimulation and bisimulation up to context is through the notion of progression [Sangiorgi and Rutten 2011, Definition 6.2.1]: a relation R *progresses* to R' when for all $(x, y) \in R$ and transitions $x \xrightarrow{\alpha} x'$, there exists $y \xrightarrow{\alpha} y'$ such that $(x', y') \in R'$ (and symmetrically). So in particular, R is a bisimulation iff it progresses to itself. This may be defined in the abstract setting, as follows.

Definition 5.1. Consider maps $i\colon R \to X \times Y$ and $j\colon R' \to X \times Y$ in any transition category. We say that R *progresses* to R', notation $R \rightsquigarrow R'$, iff for all transition types $P \xrightarrow{s} L \xleftarrow{t} Q$, processes $c\colon P \to R$, and transitions $r\colon L \to X$ making the solid part below left commute, there exist u and d as shown making the whole commute, and symmetrically for Y.





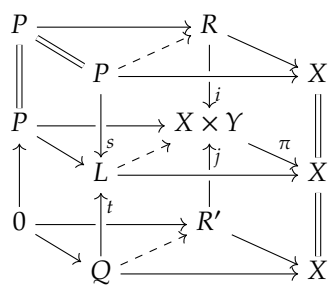
(7)

A relation R is a *pre-bisimulation* iff $R \rightsquigarrow R$.

Let us mention an equivalent presentation of progression, which uses lifting analogously to bisimulation, but in the category of cospans:

Definition 5.2. Let $\mathscr{A}^{\vee}$ have cospans in $\mathscr{A}$ as objects, and as morphisms $(a, b) \to (c, d)$ all commuting diagrams as on the right of (7).

Proposition 5.3. *Given $R \to X \times Y$ and $R' \to X \times Y$, R progresses to R' iff every commuting diagram of the following form admits a dashed lifting as shown, and symmetrically for Y.*



(8)

5.2 From Pre-Bisimulation to Bisimulation

Let us now relate pre-bisimulation to bisimulation. Intuitively, the value of R over transitions is irrelevant in the definition of pre-bisimulation. But, thinking of R as a relation over processes, 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 weak factorisation system, up to an additional stratification hypothesis. Let us first define the weak factorisation system in question, then introduce the relevant hypothesis, and finally relate pre-bisimulation to bisimulation (Proposition 5.8).

Proposition 5.4. *In any transition category $\mathscr{A}$, the set $\mathscr{J} = \{[s, t]\colon P + Q \to L \mid (s, t) \in \mathbf{T}\}$ generates a weak factorisation system, say $(\mathscr{L}, \mathscr{R})$.*

Proof. By Proposition 4.19 and tininess of process types. □

Definition 5.5. A transition category $\mathscr{A}$ is *two-level* iff $\mathscr{J} \subseteq \mathbf{P}^{\boxslash}$, viewing each $P' \in \mathbf{P}$ as the unique map $0 \to P'$.

Explicitly, $\mathscr{A}$ is two-level when any map $f : P' \to L$ with $P' \in \mathbf{P}$ lifts through any $[s,t]$ with codomain L, i.e., there exists k making the following triangle commute.

$$\begin{tikzcd} & P+Q \arrow[d, "{[s,t]}"] \\ P' \arrow[ur, dashed, "k"] \arrow[r, "f"'] & L \end{tikzcd} \tag{9}$$

Remark 5.6. All our example transition categories are two-level. Indeed, the involved base categories feature a notion of dimension, given by the only non-trivial functors to $\mathbb{2}$, the two-element ordinal viewed as a category. E.g., the functor $\mathbb{C}_{CCS} \to \mathbb{2}$ maps $\star$ to 0 and all labels to 1. Dimension extends to presheaves by decreeing that a presheaf has dimension 0 when it is empty over all objects of dimension 1, and dimension 1 otherwise. The point is that in examples, all morphisms $[s,t] : P + Q \to L$ are in fact bijective over objects of dimension 0, which directly ensures two-levelness.

Lemma 5.7. *In any two-level transition category, we have $\mathscr{L} \subseteq \mathbf{P}^{\boxslash}$.*

Proof. By Proposition 4.23, 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). □

Proposition 5.8. *For any pre-bisimulation $i : R \to X \times Y$ in a two-level transition category, the factor $\bar{i}$ of the $(\mathscr{L}, \mathscr{R})$-factorisation $R \xrightarrow{l} \overline{R} \xrightarrow{\bar{i}} X \times Y$ of i is a bisimulation.*

Intuitively, $\overline{R}$ coincides with R on states, but adds in all the transitions between pairs of related vertices that exist in $X \times Y$.

Proof. Consider any commuting square as below left. By Lemma 5.7, 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 original square as below right, and hence get the desired (dashed) lifting by $\bar{i} \in \mathscr{R}$.

$$\begin{tikzcd} P \arrow[r, "p"] \arrow[d, "s"'] & \overline{R} \arrow[d, "\bar{i}"] \\ L \arrow[r, "r"'] & X \times Y \arrow[d, "\pi"] \\ & X \end{tikzcd}$$

$$\begin{tikzcd} P \arrow[d, "s"'] \arrow[dr, "p'"] \arrow[drr, bend left, "r\circ s"] & & \\ L \arrow[dr, dashed, "r'"] \arrow[drr, bend left, "r"] & R \arrow[r, "\pi\circ i"] \arrow[d, "i"] & X \arrow[d, equal] \\ & X \times Y \arrow[r, "\pi"] & X \\ Q \arrow[u, "t"] \arrow[dr, dashed, "q'"'] \arrow[drr, bend left, "r\circ t"] & & \\ & R \arrow[r, "\pi\circ i"] \arrow[uu, "i"] & X \arrow[uu, equal] \end{tikzcd}$$

$$\begin{tikzcd} & & R \arrow[d, "l"] \\ P \arrow[r, "in_l"] \arrow[d, "s"'] \arrow[urr, bend left, "p'"] \arrow[urr, bend left, "p"'] & P+Q \arrow[r, "{[l\circ p', l\circ q']}"] \arrow[d, "{[s,t]}"] & \overline{R} \arrow[d, "\bar{i}"] \\ & & X \times Y \arrow[d, "\pi"] \\ L \arrow[r, equal] \arrow[urr, bend left, "r"'] & L \arrow[r, "r"'] \arrow[uur, dashed] \arrow[ur, "r'"'] & X \\ & & \end{tikzcd}$$

□

5.3 Pre-bisimulation up to Context

After defining pre-bisimulations in the previous section, and showing that they embed into bisimulations by factorisation, we now proceed in this section to defining pre-bisimulations up to context,

and showing that they embed into pre-bisimulations, hence into bisimulations, under suitable hypotheses.

Let us consider any familial monad $\mathscr{T}$ on a transition category $\mathscr{A}$, and $\mathscr{T}$-algebras $a\colon \mathscr{T}(X) \to X$ and $b\colon \mathscr{T}(Y) \to Y$.

Definition 5.9. A map $i\colon R \to X \times Y$ is a *pre-bisimulation up to* $\mathscr{T}$ iff $R \rightsquigarrow \mathscr{T}(R)$, where $\mathscr{T}(R)$ is equipped with the map $\mathscr{T}(R) \xrightarrow{\mathscr{T}(i)} \mathscr{T}(X \times Y) \xrightarrow{\langle \mathscr{T}(\pi), \mathscr{T}(\pi') \rangle} \mathscr{T}(X) \times \mathscr{T}(Y) \xrightarrow{a \times b} X \times Y$.

Before proving soundness of pre-bisimulation up to $\mathscr{T}$, we need to refine $\mathbf{T}_s$-familiality. Indeed, in order to find the desired lifting in the proof of Theorem 4.27, we needed to assume that the image by E of any $s \in \mathbf{T}_s$ was in ${}^{\boxslash}(\mathbf{T}_s^{\boxslash})$. For pre-bisimulations, we will proceed analogously, but in the cospan category.

Proposition 5.10. *The cospan category $\mathscr{A}^\vee$ is again a transition category with transition types, say $\mathbf{T}^\vee$, given by*

$$\begin{tikzcd}
P \arrow[r, equal] \arrow[d, equal] & P \arrow[d, "s"] & 0 \arrow[l] \arrow[d] \\
P \arrow[r, "s"'] & L & Q \arrow[l, "t"'] \arrow[d, equal] \\
0 \arrow[u] \arrow[r] & Q \arrow[u, "t"'] \arrow[r, equal] & Q.
\end{tikzcd}$$

Proposition 5.11. *Any familial monad $\mathscr{T}$ on $\mathscr{A}$ lifts to a familial monad $\mathscr{T}^\vee$ on the cospan category $\mathscr{A}^\vee$, whose generics are given by componentwise generics.*

This leads us to define:

Definition 5.12. A familial monad $\mathscr{T}$ on a transition category $\mathscr{A}$ is $\mathbf{T}_s^\vee$*-familial* iff $\mathscr{T}^\vee$ is (as a monad on $\mathscr{A}^\vee$).

Theorem 5.13. *For any $\mathbf{T}_s^\vee$-familial monad $\mathscr{T}$, compositional $\mathscr{T}$-algebras $a\colon \mathscr{T}(X) \to X$ and $b\colon \mathscr{T}(Y) \to Y$, and pre-bisimulation $i\colon R \to X \times Y$ up to $\mathscr{T}$, the following map is a pre-bisimulation:*

$$\mathscr{T}(R) \xrightarrow{\mathscr{T}(i)} \mathscr{T}(X \times Y) \xrightarrow{\langle \mathscr{T}(\pi), \mathscr{T}(\pi') \rangle} \mathscr{T}(X) \times \mathscr{T}(Y) \xrightarrow{a \times b} X \times Y. \tag{10}$$

Proof. Let i' denote the map (10), and define a', b', i'_X, and i'_Y by composition as in

$$\begin{tikzcd}
\mathscr{T}(R) \arrow[r, "\mathscr{T}(i)"] \arrow[rrr, bend left=40, "i'_X"] \arrow[rrr, bend right=40, "i'_Y"'] & \mathscr{T}(X \times Y) \arrow[r, "\langle \mathscr{T}(\pi), \mathscr{T}(\pi') \rangle"] \arrow[rr, bend left=40, "a'"] \arrow[rr, bend right=40, "b'"'] & \mathscr{T}(X) \times \mathscr{T}(Y) \arrow[r, "a \times b"] & X \times Y \arrow[r, shift left, "\pi"] \arrow[r, shift right, "\pi'"'] & X \\
 & & & & Y.
\end{tikzcd}$$

The cube on the left of Figure 1 commutes (notably by the monad algebra laws), which means that it forms a square as on the right in $\mathscr{A}^\vee$, whose top arrow is by hypothesis the image by $\mathscr{T}^\vee$ of a map in $(\mathbf{T}_s^\vee)^{\boxslash}$.

Notation 5.14. Open arrow heads denote arrows of the form $\mathscr{T}^\vee(-)$, and commutation of a diagram of such arrows means that the underlying arrows, without $\mathscr{T}$, agree.

By symmetry, it suffices to prove that the bottom map (i'_X, π, i'_X) is in $(\mathbf{T}_s^\vee)^{\boxslash}$. For this, consider any commuting square as the perimeter of

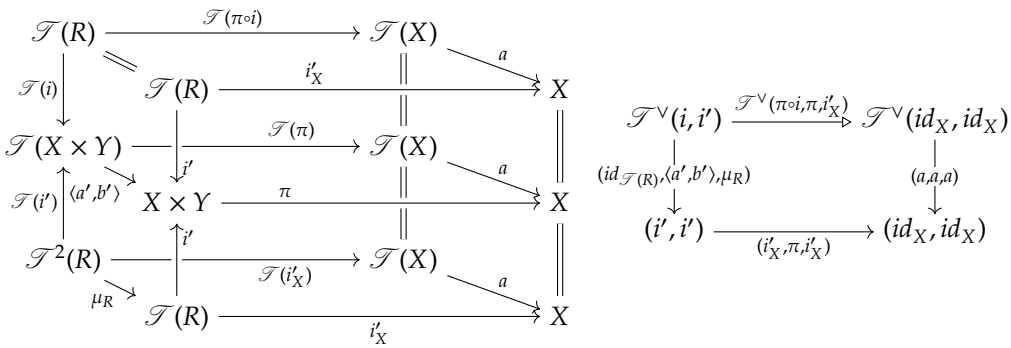
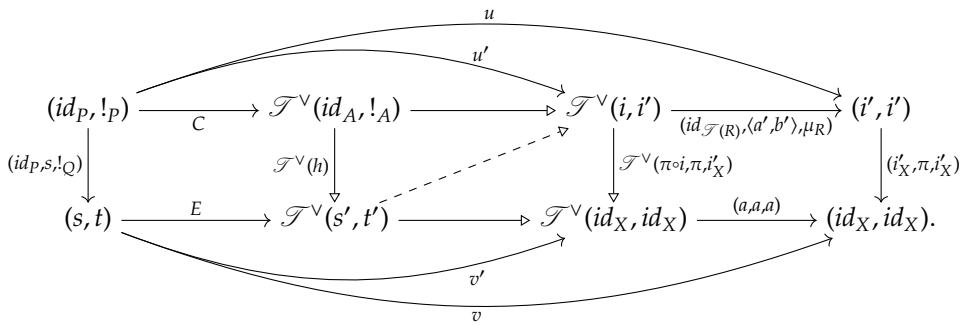


Fig. 1. Cube for the proof of Theorem 5.13



Because $(id_{\mathscr{T}(R)}, \langle a', b'\rangle, \mu_R)$ has identity first component and $(id_P, !_P)$ has identity top leg, u factors through the former, say as u'. But by compositionality of X, we have $(a, a, a) \in (\mathbf{T}_s^\vee)^{\boxslash}$, so by lifting we find a v' as shown making both parts of the diagram commute. Now, by familiality of $\mathscr{T}^\vee$, we find generic-free factorisations of u' and v', say with generics C and E. Next, by genericity of C we find a lifting $\mathscr{T}^\vee(h)$ as shown, with $h \in {}^{\boxslash}((\mathbf{T}_s^\vee)^{\boxslash})$ by $\mathbf{T}_s^\vee$-familiality. We thus find a dashed lifting as shown, which at last yields a lifting for the whole diagram. □

Corollary 5.15. *For any $\mathbf{T}_s^\vee$-familial monad $\mathscr{T}$ and compositional $\mathscr{T}$-algebras $a\colon \mathscr{T}(X) \to X$ and $b\colon \mathscr{T}(Y) \to Y$, any pre-bisimulation up to $\mathscr{T}$ embeds into some bisimulation.*

Proof. By Theorem 5.13 and Proposition 5.8. □

Example 5.16. In CCS, the cospans morphisms involved in $\mathbf{T}_s^\vee$-familiality are coproducts (in $(\mathscr{A}^\vee)^{\to}$) of maps in $\mathbf{T}_s^\vee$ and isomorphisms, so the corollary applies and bisimulation up to context is sound.

6 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 the presentation of Sangiorgi and Walker [2001, §1.3], using a simplified variant for expository purposes.

In §6.1, we make a first attempt at covering the 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 $\mathscr{T}_\pi(0)$ of processes is not compositional. We rectify this in a standard way in §6.2, by defining a familial submonad, $\mathscr{T}_\pi^-$, such that the $\mathscr{T}_\pi^-$-algebra $\mathscr{T}_\pi(0)$ is compositional, thus recovering the known facts that (1) standard bisimilarity is a congruence for all operators but input, and (2) bisimulation up to non-input context is sound. We finally consider in §6.3 a different, though still standard, way of remedying the non-congruence problem. This consists in restricting attention to relations, called *wide open* [Fiore and Staton 2006; Sangiorgi and Walker 2001; Staton 2008], 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 $\mathscr{T}_\pi^+$. We recover the known results that wide-open bisimilarity is a congruence and that wide-open bisimulation up to context is sound.

6.1 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.

Naively adapting what we did with CCS to the π-calculus, we could try considering a similar base category with CCS labels replaced with π-calculus labels, $\tau, o_{a,b}, \iota_{a,b}$, etc. However, this setting cannot accomodate the input axiom: $a(c).P \xrightarrow{\iota_{a,b}} P[c \mapsto b]$. Indeed, this requires a definition of renaming, for which a standard, inductive definition stumbles upon the base case: it does not make any sense to replace c with b in $(\!|x|\!)$. We thus consider a different base category.

Definition 6.1. Recalling $\mathscr{N}$ from Example 3.1, 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 $\tau_\gamma, o_{\gamma,a,b}, o^\nu_{\gamma,a,c}, \iota_{\gamma,a,b}$, and $\iota^\nu_{\gamma,a,c}$ for all $\gamma \in \mathscr{P}_f(\mathscr{N})$, $a, b \in \gamma$, and $c \notin \gamma$,
- morphisms, arranged by transition types (denoting by γ, c the (disjoint) union $\gamma \uplus \{c\}$ – here but not, e.g., in $o_{\gamma,a,b}$: input and output labels have three arguments, a γ and two names; we rely on context to disambiguate):

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

- plus, for all transition types $\gamma \xrightarrow{s} \alpha \xleftarrow{t} \gamma, \delta$ and bijections $h\colon \gamma \xrightarrow{\sim} \gamma'$ and $k : \delta \xrightarrow{\sim} \delta'$ such that $\gamma' \cap \delta' = \varnothing$, a morphism $(h,k)\colon (h,k) \cdot \alpha \to \alpha$, where

$$(h,k) \cdot \tau_\gamma = \tau_{\gamma'} \qquad \text{and} \qquad (h,k) \cdot \alpha_{\gamma,a,b} = \alpha_{\gamma',h(a),(h+k)(b)} \text{ for } \alpha \in \{o, \iota, o^\nu, \iota^\nu\},$$

satisfying the obvious equations:

$$\begin{tikzcd} \gamma \arrow[r, "s"] & \alpha & \gamma, \delta \arrow[l, "t"'] \\ \gamma' \arrow[u, "h"] \arrow[r, "s"'] & (h,k) \cdot \alpha \arrow[u, "(h,k)"'] & \gamma', \delta'. \arrow[u, "h+k"'] \arrow[l, "t"] \end{tikzcd} \tag{11}$$

Notation 6.2. 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 $f : \gamma \xrightarrow{\sim} \gamma'$ 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. We use the same convention for arbitrary (potentially non-bijective) maps below.

Proposition 6.3. *The category $\widehat{\mathbb{B}}$ forms a transition category, with transition types $\mathbf{T}^\pi$ as above.*

Let us now define our monad on $\widehat{\mathbb{B}}$, in three stages. We first define $\mathscr{T}_\pi(X)$ on process types. We then observe that not only bijections act on $\mathscr{T}_\pi(X)$, but also arbitrary maps. The presheaf $\mathscr{T}_\pi(X)$

that we defined on process types and bijections thus extends to a functor on the category of finite subsets of $\mathscr{N}$ and all maps. We finally use this to define $\mathscr{T}_\pi(X)$ on transitions.

For any $X \in \widehat{\mathbb{B}}$, let $\mathscr{T}_\pi(X)(\gamma)$ denote 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 2. Let us then define *renaming*, the action

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

$$\frac{\gamma \vdash_X P \qquad a,b \in \gamma}{\gamma \vdash_X \bar{a}\langle b\rangle.P} \qquad \frac{\gamma, b \vdash_X P \qquad a \in \gamma}{\gamma \vdash_X a(b).P}$$

$$\frac{e \in X(\alpha) \qquad \gamma \xrightarrow{s} \alpha \xleftarrow{t} \gamma,\delta \qquad h\colon \gamma \rightarrowtail \gamma' \qquad k\colon \delta \rightarrowtail \delta' \qquad \gamma' \cap \delta' = \varnothing}{\langle\!\langle e\rangle\!\rangle(h,k) : \gamma' \vdash_X (\!|e\cdot s|\!)(h) \xrightarrow{(h,k)\cdot\alpha} \gamma',\delta' \vdash_X (\!|e\cdot t|\!)(h+k)}$$

$$\frac{\gamma, b \vdash_X P \qquad a,c \in \gamma}{in^{a,c}_{b.P} : \gamma \vdash_X a(b).P \xrightarrow{\iota_{\gamma,a,c}} \gamma \vdash_X [b \mapsto c]\cdot P} \qquad \frac{\gamma \vdash_X P \qquad a,b \in \gamma}{out^{a,b}_{\gamma\vdash_X P} : \gamma \vdash_X \bar{a}\langle b\rangle.P \xrightarrow{o_{\gamma,a,b}} \gamma \vdash_X P}$$

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

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

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

$$\frac{R : \gamma, b \vdash_X P \xrightarrow{\alpha} \gamma, b, \delta \vdash_X Q \qquad b \notin fv(\alpha)}{\nu b.R : \gamma \vdash_X \nu b.P \xrightarrow{\nu b.\alpha} \gamma, \delta \vdash_X \nu b.Q}$$

$$\frac{L : \gamma \vdash_X P \xrightarrow{\alpha} \gamma, \delta \vdash_X P' \qquad \gamma \vdash_X Q}{L|Q : \gamma \vdash_X P|Q \xrightarrow{\alpha} \gamma, \delta \vdash_X P'|Q} + \text{symmetric rule } Q|L$$

Fig. 2. Syntax and labelled transition system for π

$\mathscr{T}_\pi(\gamma) \to \mathscr{T}_\pi(\gamma')$ of any set-map $f : \gamma \to \gamma'$, denoted by $P \mapsto f \cdot P$, by straightforward induction; notably, for the base case, if $g\colon \gamma'' \to \gamma$, we set $f \cdot (\!|x|\!)(g) = (\!|x|\!)(f \circ g)$ (see full definition in [Hirschowitz 2018]).

Finally, let us define $\mathscr{T}_\pi(X)$ on transitions. For all $\gamma \xrightarrow{s} \alpha \xleftarrow{t} \gamma'$, $\mathscr{T}_\pi(X)(\alpha)$ denotes the set of α-equivalence classes of transitions $P \xrightarrow{\alpha} Q$ 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 2, where

- the only binding operations are $in^{a,c}_{b.P}$, $\nu b.R$, and $R \rhd^{\nu b} S$ – binding b;
- a peculiarity is that in $(\!|x|\!)(f)$, the status of names in γ is analogous to that of 'binding' names in contexts, e.g., as b in $a(b).\square$; in particular they are neither free, nor bound, nor α-convertible;
- $[b \mapsto c]$ stands for the map $(\gamma, b) \to \gamma$ mapping b to c and the rest of γ to itself;
- weakening (i.e., renaming along an injective map) is used implicitly in $L|Q$ and $Q|L$;
- for $\nu b.R$, we define

$$\begin{array}{lll} fv(o_{\gamma,a,b}) = fv(\iota_{\gamma,a,b}) = fv(o^\nu_{\gamma,a,b}) = fv(\iota^\nu_{\gamma,a,b}) = \{a,b\} & \text{and} & fv(\tau_\gamma) = \varnothing \\ \text{and} \quad \nu c.\tau_{\gamma,c} = \tau_\gamma & & \nu c.\alpha_{(\gamma,c),a,b} = \alpha_{\gamma,a,b}, \text{ for } \alpha \in \{o, \iota, o^\nu, \iota^\nu\}, \end{array}$$

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

This almost defines an assignment $ob(\widehat{\mathbb{B}}) \to ob(\widehat{\mathbb{B}})$: it remains to define the action of maps in $\mathbb{B}$ on $\mathscr{T}_\pi(X)$ and show that it is functorial. For maps of the form $\gamma \to \gamma'$, we merely use renaming; for s and t, we use the sources and targets specified in transitions. For $(h,k)\colon (h,k)\cdot\alpha \to \alpha$, we define $(h,k)\cdot R$ by induction on R, e.g.,

$$\begin{array}{rcl} (h,k)\cdot(\langle\!\langle e\rangle\!\rangle(h',k')) & = & \langle\!\langle e\rangle\!\rangle(h\circ h', k\circ k') \\ (h, id_\varnothing)\cdot in^{a,c}_{b.P} & = & in^{h(a),h(c)}_{a_{\gamma'}.((h+(b\mapsto a_{\gamma'}))\cdot P)} \\ (h, id_\varnothing)\cdot(R \rhd^{\nu b} S) & = & ((h,(b\mapsto a_{\gamma'}))\cdot R) \rhd^{\nu a_{\gamma'}} ((h,(b\mapsto a_{\gamma'}))\cdot S), \ldots \end{array} \tag{12}$$

where $(b \mapsto c)$ is the unique map $\{b\} \to \{c\}$, and a_γ denotes some globally chosen name not in γ.

Finally, we prove that this is functorial, again by induction. This defines an assignment $ob(\widehat{\mathbb{B}}) \to ob(\widehat{\mathbb{B}})$, which easily extends to a functor $\mathscr{T}_\pi\colon \widehat{\mathbb{B}} \to \widehat{\mathbb{B}}$: the action of a morphism $F\colon X \to Y$ in $\widehat{\mathbb{B}}$ is obtained by renaming x, resp. e, according to F in $(\!|x|\!)(f)$, resp. $\langle\!\langle e\rangle\!\rangle(h,k)$.

Remark 6.4 (Replication). The π-calculus standardly features additional operations like guarded sum and replication. Guarded sum may be incorporated readily, but, depending on presentation, replication may be less easy. E.g., the compact transition rule $\dfrac{P|!P \xrightarrow{\alpha} Q}{!P \xrightarrow{\alpha} Q}$ does not obviously yield a familial monad. However, the rules of Sangiorgi and Walker [2001, §1.3], a variant of which is reproduced below (without the γ's for readability), do yield a familial monad directly.

$$\frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'|!P} \qquad \frac{P \xrightarrow{o_{\gamma,a,b}} P' \qquad P \xrightarrow{\iota_{\gamma,a,b}} P''}{!P \xrightarrow{\tau_\gamma} (P'|P'')|!P} \qquad \frac{P \xrightarrow{o^\nu_{\gamma,a,b}} P' \qquad P \xrightarrow{\iota^\nu_{\gamma,a,b}} P''}{!P \xrightarrow{\tau_\gamma} \nu b.(P'|P'')|!P}$$

Proposition 6.5. *$\mathscr{T}_\pi$ forms a $\mathbf{T}^\pi_s$-familial monad.*

Proof sketch. (More detail in [Hirschowitz 2018].) To see that $\mathscr{T}_\pi$ is a familial functor, by Lemma 4.12, it is enough to exhibit a functor $E\colon el(\mathscr{T}_\pi(1)) \to \widehat{\mathbb{B}}$ 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, say $\top_{\gamma'}$ 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 $\top_\beta$, with source $\top_\beta \cdot s = \top_\gamma$ and target $\top_\beta \cdot t = \top_{\gamma'}$.

On objects, we construct E by induction on the depth of the considered proof (globally for all objects $c \in \mathbb{B}$), as in Figure 3, relying on some global choice of binary coproducts and fresh name a_γ, for all $\gamma \in \mathscr{P}_f(\mathscr{N})$, as in the definition of renaming. The choice of a_γ is in fact completely irrelevant, because for all maps $h : \gamma \rightharpoonup \gamma'$, $k\colon \delta \rightharpoonup \delta'$ in sets, processes $\gamma \vdash P$, and transitions $R\colon (\gamma \vdash P) \xrightarrow{\alpha} (\gamma, \delta \vdash Q)$, we have by induction:

$$E(h \cdot P) = E(P) \qquad \text{and} \qquad E((h,k) \cdot R) = E(R) \qquad \text{(yes, equality!).} \tag{13}$$

Typically, if $f : \gamma \to \gamma'$, we have by definition $E(f \cdot (\!|\top_\gamma|\!)(id)) = E((\!|\top_\gamma|\!)(f)) = \mathbf{y}_\gamma$: f is not taken into account.

Processes

$$\begin{array}{rcl} E((\!|\top_\gamma|\!)(f)) & = & \mathbf{y}_\gamma \\ E(P|Q) & = & E(P) + E(Q) \\ E(0) & = & 0 \\ E(\nu a_\gamma.P) & = & E(P) \\ E(\overline{a}\langle b\rangle.P) & = & E(P) \\ E(a(a_\gamma).P) & = & E(P) \end{array}$$

Transitions

$$\begin{array}{rcl} E(\langle\!\langle \top_\alpha \rangle\!\rangle(h,k)) & = & \mathbf{y}_\alpha \\ E(in^{a,c}_{a_\gamma.P}) & = & E(P) \\ E(out^{a,c}_P) & = & E(P) \\ E(R \rhd S) & = & E(R) + E(S) \\ E(in^{a,\nu b}_P) & = & E(P) \end{array} \qquad \begin{array}{rcl} E(\nabla b.R) & = & E(R) \\ E(R \rhd^{\nu a_\gamma} S) & = & E(R) + E(S) \\ E(\nu a_\gamma.R) & = & E(R) \\ E(L|Q) & = & E(L) + E(Q) \\ E(P|R) & = & E(P) + E(R) \end{array}$$

Fig. 3. Definition of E on objects

Let us now define E on morphisms:

- For any $f : \gamma \rightharpoonup \gamma'$ in sets and $P \in \mathscr{T}_\pi(1)(\gamma)$, we have $f \upharpoonright P\colon (\gamma', f \cdot P) \to (\gamma, P)$ in $el(\mathscr{T}_\pi(1))$ and define $E(f \upharpoonright P)\colon E(\gamma', f \cdot P) \to E(\gamma, P)$ to be just the identity (which makes sense by (13)).
- Similarly, for all $(h,k)\colon (h,k) \cdot \alpha \to \alpha$, let $E((h,k) \upharpoonright R) = id_{E(R)}$.
- Finally, for all $\gamma \xrightarrow{s} \alpha \xleftarrow{t} \gamma, \delta$ and $R \in \mathscr{T}_\pi(1)(\alpha)$, we define $E(s \upharpoonright R)$ and $E(t \upharpoonright R)$ straightforwardly by induction, thanks to (13). An example key case is $E(R \rhd^{\nu a_\gamma} S)$, for which we define $E(s \upharpoonright R \rhd^{\nu a_\gamma} S)$ and $E(t \upharpoonright R \rhd^{\nu a_\gamma} S)$ to be

$$E(P) + E(Q) \xrightarrow{E(s\upharpoonright R)+E(s\upharpoonright S)} E(R) + E(S) \xleftarrow{E(t\upharpoonright R)+E(t\upharpoonright S)} E(P') + E(Q').$$

Clearly, by induction, each $E(s \upharpoonright R)$ lies in ${}^{\boxtimes}((\mathbf{T}^\pi_s)^{\boxtimes})$, so that $\mathscr{T}_\pi$ is $\mathbf{T}^\pi_s$-familial by Remark 4.26. The unit is given by variables, and multiplication is essentially substitution, which is straightforwardly defined by induction. Key cases are:

$$\mu_{X,\alpha}(\langle\!\langle r \rangle\!\rangle(h,k)) = (h,k) \cdot r \qquad\qquad \mu_{X,\tau_\gamma}(R \rhd^{\nu b} S) = \mu_{X,o^\nu_{\gamma,a,b}}(R) \rhd^{\nu b} \mu_{X,\iota^\nu_{\gamma,a,b}}(S).$$

Unit and multiplication are natural and satisfy the monad laws. 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:

Proposition 6.6. *The free $\mathscr{T}_\pi$-algebra, $\mathscr{T}_\pi(0)$, with action given by μ_0, 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 [Sangiorgi and Walker 2001]. Letting $f : \{a,b\} \to \{a\}$ denote the unique such

map, $(\!|p|\!)(f) \in (\mathscr{T}_\pi)^2(0)\{a\}$ is mapped by $\mu_{0,\{a\}}$ 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 6.7. The root of the problem here is $in^{a,c}_{b.P}$, 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 presented 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: $(\!|x|\!)(g)[f] = (\!|x|\!)(f \circ g)$.

6.2 Non-Input Congruence

A first, standard way around 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 cover this by viewing π-calculus as a non-free algebra for a smaller monad $\mathscr{T}^-_\pi$.

Definition 6.8. Let $\mathscr{T}^-_\pi$ denote the sub-functor of $\mathscr{T}_\pi$ obtained by removing $a(b).P$ from the syntax of Figure 2, replacing the rule for $(\!|x|\!)(f)$ by

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

where $\mathbf{Inj}(\gamma, \gamma')$ denotes the set of injective maps $\gamma \hookrightarrow \gamma'$, and removing $in^{a,c}_{b.P}$ and $in^{a,\nu b}_P$ from the labelled transition system of Figure 2.

Remark 6.9. We need to retain injective renaming for $L|Q$ and $P|R$ to make sense.

We clearly obtain:

Proposition 6.10. *$\mathscr{T}^-_\pi$ is a $\mathbf{T}^\pi_s$-familial monad.*

But this time, instead of considering $\mathscr{T}^-_\pi(0)$, which does not even satisfy the input rule, we observe that $\mathscr{T}_\pi(0)$ 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 6.11. *The $\mathscr{T}^-_\pi$-algebra $\mathscr{T}_\pi(0)$ is compositional.*

This gives an alternative proof of [Sangiorgi and Walker 2001, Theorem 2.2.8(1)], which says that bisimilarity is a *non-input congruence* [Sangiorgi and Walker 2001, Definition 2.1.23]. In our language:

Corollary 6.12. *Bisimilarity for the $\mathscr{T}^-_\pi$-algebra $\mathscr{T}_\pi(0)$ is a congruence.*

Proof. By Corollary 4.30 and Propositions 6.10 and 6.11. □

Similarly, $\mathscr{T}^-_\pi$ is $(\mathbf{T}^\pi_s)^\vee$-familial, so we recover Sangiorgi and Walker [2001, Lemma 2.3.21]:

Corollary 6.13. *Bisimulation up to non-input context is sound.*

6.3 Wide-Open Bisimilarity

Another standard solution to the failure of bisimilarity to be a congruence is to resort to wide-open bisimilarity. For this, following Staton [2008], we need to modify our base category $\mathbb{B}$ to include non-bijective maps $\gamma \to \gamma'$.

Definition 6.14. Let $\mathbb{F}$ be defined just as $\mathbb{B}$ (Definition 6.1), but including all morphisms from $\mathbf{Set}^{op}$ (between relevant objects), instead of just bijections.

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

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

Proposition 6.15. *Replacing the rules for $(\!|x|\!)(f)$ and $\langle\!\langle e\rangle\!\rangle(h,k)$ in Figure 2 by*

$$\frac{x \in X(\gamma)}{\gamma \vdash_X (\!|x|\!)} \qquad\qquad \frac{e \in X(\alpha)}{\gamma \vdash_X \langle\!\langle e\rangle\!\rangle},$$

and changing the base cases for renaming to $f \cdot (\!|x|\!) = (\!|f \cdot x|\!)$ and $(h,k) \cdot \langle\!\langle e\rangle\!\rangle = \langle\!\langle (h,k) \cdot e\rangle\!\rangle$ yields a functor $\mathscr{T}^+_\pi : \widehat{\mathbb{F}} \to \widehat{\mathbb{F}}$.

It is not entirely trivial that this again forms a familial monad.

Proposition 6.16. *$\mathscr{T}^+_\pi$ is a $\mathbf{T}^+_s$-familial monad.*

Proof sketch. (More detail in [Hirschowitz 2018].) Most of the definitions are adapted from $\mathbb{B}$ to $\mathbb{F}$, modulo the following subtlety. On objects, fresh names a_γ are chosen globally for all γ as in the proof of Proposition 6.5, and only the base cases change to $E(\!|x|\!) = \mathbf{y}_\gamma$ (for $x \in X(\gamma)$) and $E\langle\!\langle e\rangle\!\rangle = \mathbf{y}_\alpha$ (for $e \in X(\alpha)$). However, over $\mathbb{B}$, we had $E(f \cdot P) = E(P)$, which considerably eased the definition of E on morphisms. But because of the new treatment of renaming, this no longer holds over $\mathbb{F}$. E.g., consider $P = (\!|\top_{\{a\}}|\!)$. We have $E(P) = \mathbf{y}_{\{a\}}$, but $E((a \mapsto b) \cdot P) = \mathbf{y}_{\{b\}}$ (where, we recall from (12), $(a \mapsto b)$ denotes the unique map $\{a\} \to \{b\}$), so we only get $E(P) \cong E((a \mapsto b) \cdot P)$. The treatment of morphisms thus needs adjustment: for $h : \gamma \rightarrowtail \gamma'$ and $k : \delta \rightarrowtail \delta'$ as in (13), we get inductively-defined, functorial assignments

$$E(h \upharpoonright P) : E(h \cdot P) \to E(P) \qquad \text{and} \qquad E((h,k) \upharpoonright R) : E((h,k) \cdot R) \to E(R).$$

Finally, the definitions of $E(s \upharpoonright R)$ and $E(t \upharpoonright R)$ also need adjustment for scope-changing constructors $in^{a,vb}_P$, $\nabla b.R$, $vb.R$, and $R \rhd^{vb} S$. E.g., consider the morphism $t \upharpoonright in^{a,c}_{b.P} : [b \mapsto c] \cdot P \to in^{a,c}_{b.P}$. We have by definition $E(in^{a,c}_{b.P}) = E((b \mapsto a_\gamma) \cdot P)$ so we define $E(t \upharpoonright in^{a,c}_{b.P})$ to be

$$E([b \mapsto c] \cdot P) \xrightarrow{E([a_\gamma \mapsto c] \upharpoonright (b \mapsto a_\gamma) \cdot P)} E((b \mapsto a_\gamma) \cdot P).$$

We then show by induction that this assignment on morphisms is functorial, which entails that $\mathscr{T}^+_\pi$ is familial. Finally, by definition, morphisms $s \upharpoonright R$ are mapped to coproducts of isomorphisms and maps in $\mathbf{T}^+_s$, up to isomorphism in the arrow category, hence $\mathscr{T}^+_\pi$ is indeed $\mathbf{T}^+_s$-familial. □

The compositionality problem created by $x(\!|f|\!)$ having disappeared, we get:

Proposition 6.17. *$\mathscr{T}^+_\pi(0)$ is a compositional algebra.*

Calling *wide-open* bisimilarity the largest bisimulation relation over $\mathscr{T}^+_\pi(0)$, we get:

Proposition 6.18. *Wide-open bisimilarity is a congruence.*

Proof. By Corollary 4.31 and Propositions 6.16 and 6.17. □

Similarly, we recover Sangiorgi and Walker [2001, Corollary 2.3.25]:

Corollary 6.19. *Wide-open bisimulation up to context is sound.*

Remark 6.20. A different presentation of wide-open bisimilarity may be obtained by augmenting $\mathbb{B}$ with a transition type $\gamma, a \xrightarrow{s} \sigma_{\gamma,a,b} \xleftarrow{t} \gamma$ (for $b \in \gamma$), and adding the rule $P \xrightarrow{\sigma_{\gamma,a,b}} [a \mapsto b] \cdot P$.

Remark 6.21. Similarly, *open* bisimilarity [Sangiorgi and Walker 2001, Definition 4.6.2] may be obtained 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 $f(a) \neq f(b)$.

7 CONCLUSION AND PERSPECTIVES

We presented a categorical 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 lifting in the cospan category. We finally showed that the framework flexibly accounts for most known results about congruence of bisimilarity and soundness of bisimulation up to context in the π-calculus. To our knowledge, this is the first categorical account of congruence of bisimilarity and soundness of bisimulation up to context in the presence of binding. We furthermore hope to have demonstrated that the approach is close to operational intuitions.

However, although the framework provides abstract, rather general proofs of non-trivial facts, it does not yet 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. An intermediate step would be to show that relevant formats (e.g., Middelburg [2001]; Mousavi et al. [2005]; Sands [1997]) give rise to familial monads with compositional initial algebras. Another potential direction is to consider questions related to congruence of bisimilarity and soundness of bisimulation up to context, e.g., weak variants of these results, environmental bisimulation, or solutions of process equations. Finally, we could consider adapting *pointwise analytic* monads [Garner and Hirschowitz 2018] to transition categories, which could accomodate *structural congruence*, hence possibly provide a new language to study the derivation of labelled transition systems from reduction rules [Sewell 1998].

ACKNOWLEDGMENTS

Thanks to Uli Fahrenberg, André Hirschowitz, Paul-André Melliès, David Sands, Sam Staton, and the reviewers for useful comments, and to Paweł Sobociński for bringing my attention to this question some 10 years ago. And eternal categorical thanks to Richard Garner.

REFERENCES

C. Berger, P.-A. Melliès, and M. Weber. 2012. Monads with arities and their associated theories. *Journal of Pure and Applied Algebra* 216, 8 (2012), 2029 – 2048. https://doi.org/10.1016/j.jpaa.2012.02.039

Karen L. Bernstein. 1998. A Congruence Theorem for Structured Operational Semantics of Higher-Order Languages. In *Proc. 13th Symposium on Logic in Computer Science* IEEE Computer Society, 153–164. https://doi.org/10.1109/LICS.1998.705652

Bard Bloom, Sorin Istrail, and Albert R. Meyer. 1995. Bisimulation Can't be Traced. *Journal of the ACM* 42, 1 (1995), 232–268. https://doi.org/10.1145/200836.200876

Roland N. Bol and Jan Friso Groote. 1996. The Meaning of Negative Premises in Transition System Specifications. *Journal of the ACM* 43, 5 (1996), 863–914. https://doi.org/10.1145/234752.234756

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

Gérard Boudol and Ilaria Castellani. 1988. A non-interleaving semantics for CCS based on proved transitions. *Fundamenta Informaticae* XI (1988).

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

Gian Luca Cattani, John Power, and Glynn Winskel. 1998. A Categorical Axiomatics for Bisimulation, See [Sangiorgi and de Simone 1998], 581–596. https://doi.org/10.1007/BFb0055649

Yves Diers. 1978. Spectres et localisations relatifs à un foncteur. *Comptes rendus hebdomadaires des séances de l'Académie des sciences* 287, 15 (1978), 985–988.

Jérémy Dubut, Eric Goubault, and Jean Goubault-Larrecq. 2016. Bisimulations and unfolding in $\mathscr{P}$-accessible categorical models. In *Proc. 27th International Conference on Concurrency Theory (LIPIcs)*, Josée Desharnais and Radha Jagadeesan (Eds.). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. https://doi.org/10.4230/LIPIcs.CONCUR.2016.25

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

Marcelo P. Fiore and Daniele Turi. 2001. Semantics of Name and Value Passing. In *Proc. 16th Symposium on Logic in Computer Science* IEEE, 93–104. https://doi.org/10.1109/LICS.2001.932486

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

Tom Hirschowitz. 2018. Familial monads and structural operational semantics. (2018). https://hal.archives-ouvertes.fr/hal-01815328 Preprint.

Mark Hovey. 1999. *Model Categories.* Mathematical Surveys and Monographs, Volume 63, AMS (1999), Vol. 63. American Mathematical Society. https://doi.org/10.1090/surv/063

Peter T. Johnstone. 2002. *Sketches of an Elephant: A Topos Theory Compendium - Volume 1.* Oxford University Press.

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

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

Joachim Kock. 2011. Polynomial functors and trees. *International Mathematics Research Notices* 2011, 3 (2011), 609–673. https://doi.org/10.1093/imrn/rnq068

F. W. Lawvere. 1963. *Functorial semantics of algebraic theories.* Ph.D. Dissertation. Columbia University.

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

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

C. A. Middelburg. 2001. Variable binding operators in transition system specifications. *Journal of Logic and Algebraic Programming* 47, 1 (2001), 15–45. https://doi.org/10.1016/S1567-8326(00)00003-5

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. https://doi.org/10.1016/j.tcs.2006.12.019

Mohammad Reza Mousavi, Murdoch Gabbay, and Michel A. Reniers. 2005. SOS for Higher Order Processes. In *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings (Lecture Notes in Computer Science)*, Martín Abadi and Luca de Alfaro (Eds.), Vol. 3653. Springer, 308–322. https://doi.org/10.1007/11539452_25

Gordon D. Plotkin. 1981. *A Structural Approach to Operational Semantics.* DAIMI Report FN-19. Computer Science Department, Aarhus University.

Damien Pous and Davide Sangiorgi. 2011. *Enhancements of the bisimulation proof method*, Chapter 6. In Sangiorgi and Rutten [Sangiorgi and Rutten 2011].

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

David Sands. 1997. From SOS Rules to Proof Principles: An Operational Metatheory for Functional Languages. In *Proc. 24th International Symposium on Principles of Programming Languages*, Peter Lee, Fritz Henglein, and Neil D. Jones (Eds.). ACM Press, 428–441. https://doi.org/10.1145/263699.263760

Davide Sangiorgi. 1994. The Lazy Lambda Calculus in a Concurrency Scenario. *Information and Computation* 111, 1 (1994), 120–153. https://doi.org/10.1006/inco.1994.1042

Davide Sangiorgi and Robert de Simone (Eds.). 1998. *Proc. 9th International Conference on Concurrency Theory.* LNCS, Vol. 1466. Springer.

Davide Sangiorgi and Jan Rutten (Eds.). 2011. *Advanced Topics in Bisimulation and Coinduction.* Number 52 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Davide Sangiorgi and David Walker. 2001. *The π-calculus – A Theory of Mobile Processes.* Cambridge University Press.

Peter Sewell. 1998. From Rewrite Rules to Bisimulation Congruences, See [Sangiorgi and de Simone 1998], 269–284. https://doi.org/10.1007/BFb0055628

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

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

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