Categorical simulations

J.R.B. Cockett *
Department of Computer Science, University of Calgary,
Calgary, T2N 1N4, Alberta, Canada
email : robin@cpsc.ucalgary.ca

Pieter J. W. Hofstra †
Dept. of Mathematics and Statistics, University of Ottawa,
Ottawa, K1N 6N5, Ontario, Canada
email : phofstra@uottawa.ca

August 8, 2008

Abstract

We investigate notions of simulation between categories over a base, inspired by and directly relevant for the study of categories arising in computability and realizability, but applicable to other settings as well. Such simulations admit a conceptual description in terms of the free fibration monad; this relates them closely to fibrations of (partitioned) assemblies. Our main application is in the area of abstract computability, where we show that the category of Turing categories over a fixed base and simulations between them is 2-equivalent to the category of relative PCAs in the base.

*Partially supported by NSERC, Canada.
†Supported by NSERC, Canada

Contents

1 Introduction

Both in computer science and mathematics there are situations when one wishes to compare structures, but where homomorphisms are too restrictive and a weaker notion of "simulation" is required. To motivate this weaker notion of structural comparison let us begin with some examples.

1.1 Examples of simulations

Partial combinatory algebras Much of this work arose from the study of *partial combinatory algebras* (PCAs) and various other structures related to computability. Informally, a PCA is an object which embodies an abstract notion of computation; the paradigmatic example is Kleene's first model $\mathcal{K}_1$ which consists of the natural numbers with the recursion theoretic partial application operation $n \bullet m$ in which n is regarded as the index of a Turing machine and m as the input for that machine. More formally a PCA $\mathbb{A} = (A, \bullet)$ is a set A together with a partial binary application operation $\bullet$. The partial functions $a \bullet - : A \longrightarrow A$ are then regarded as the $\mathbb{A}$-computable functions; this is generalized to n-ary partial functions by $(x_1, ..., x_n) \mapsto (..((a \bullet x_1) \bullet x_2)...) \bullet x_n$. (There is a technical condition here that the map is total in its first $n-1$ arguments, but that will not concern us here.) The special requirement of a PCA is *combinatory completeness*, that is that *every* polynomial over $\mathbb{A}$ (in variables, elements of A, and the application) can be expressed as an $\mathbb{A}$-computable map.

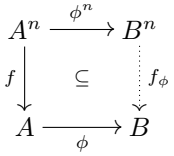
PCA are often used as the basis for providing constructive (or *realizablility*) interpretations of logical systems: in particular, from a PCA one can construct a *realizability topos* which is a model of higher-order intuitionistic logic (see [17] for a modern exposition and bibliography).

From $\mathbb{A}$ one may also build a category $\mathsf{Comp}(A)$ with the formal finite powers of $\mathbb{A}$ as objects, and the (tuples of) partial n-ary computable functions as maps. Such a category is an example of a so-called *Turing category*, which is a partial map category with finite products and a weakly universal object (see [4] for details, as well as an explanation of how such categories may be used to give an abstract presentation of computability theory).

Between PCAs one is usually not so interested in homomorphisms, rather one considers a much weaker notion due to Longley [11], which he called an *applicative morphism*). An applicative morphism between two PCAs $\mathbb{A} = (A, \bullet_A)$ and $\mathbb{B} = (B, \bullet_B)$, $\phi : \mathbb{A} \longrightarrow \mathbb{B}$, is a total relation from A to B such that there exists an element $u \in B$ with the property that for all $a, a' \in A$ with $a \bullet a'$ defined, $x \in \phi(a)$, and $x' \in \phi(a')$ we have $(u \bullet x) \bullet y \in \phi(a \bullet a')$.

Reformulating the above condition, one may show that it is equivalent to asking that for each $\mathbb{A}$-computable $f : A^n \longrightarrow A$ there is a $\mathbb{B}$-computable function $f_\phi : B^n \longrightarrow B$ such that the following diagram commutes up to inclusion of

relations:

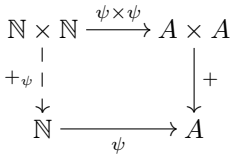


This may be interpreted as requiring that under ϕ the PCA $\mathbb{B}$ can simulate every $\mathbb{A}$-computable function.

There are a number of variations on this basic pattern. For example one could ask that ϕ be functional or that the containment in the above diagram be an equality – in other words that it be a *strict* rather than *lax* simulation. In this paper the case when ϕ is a function shall be studied further. Whether this is appropriate depends on the setting and application one has in mind. However, in [4] it was argued that the functional version is, in some sense, more primitive as not only does the relational version generally require more structure of the ambient setting (e.g. to be a regular category), but also it is often possible (see [9]) to describe relational morphisms as a Kleisli category for the functional ones anyway.

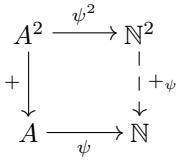
Whether one uses lax or strict simulations, of course, also depends on the application. Realizability uses the lax variant. However, for computability, as better control of the partiality is needed, it can be argued [4] that the strict variant is appropriate.

Computable rings In constructive algebra [15] the notion of a computable ring is defined to be a ring A with a partial surjection $\psi : \mathbb{N} \longrightarrow A$ such that the ring operations can be recursively tracked. For addition this means that there is a partial recursive function $+_\psi : \mathbb{N} \times \mathbb{N} \longrightarrow N$ such that



commutes. In modern terms this may be succinctly expressed by saying that A is a ring object in the category of modest sets.

To make this look more like our earlier example note that the converse of ψ, written $\psi^\circ : A \longrightarrow \mathbb{N}$ is is a total relation which, additionally, satisfies a discreteness requirement: $\psi^\circ(a) \cap \psi^\circ(b) = \emptyset$ if and only if $a \neq b$. To say that addition is tracked amounts to requiring that there is a partial recursive function $+_\psi$ such that



commutes up to containment of relations.

Gödel numbering Our second example comes from logic: the process of encoding formal systems (structures, datatypes) in an effective manner has been a very powerful idea, lying at the heart of important results in various branches of mathematical logic. More concretely, an effective encoding (also called *Gödel numbering*) of a structure A into $\mathbb{N}$ is, first of all, a function (preferably injective) $\phi : A \longrightarrow \mathbb{N}$. The structure A one is interested in is typically more than a mere set: it comes equipped with various operations $p : A^n \longrightarrow A$ on it (these may be total, or partial). One wishes (and this is the effectivity of the encoding) that these operations can be represented on the level of codes. This means that for each operation p there is a partial recursive function p_ϕ such that

$$\begin{array}{ccc} A^n & \xrightarrow{\phi^n} & \mathbb{N}^n \\ {\scriptstyle p}\downarrow & & \vdots\ {\scriptstyle p_\phi} \\ A & \xrightarrow{\phi} & \mathbb{N} \end{array}$$

commutes. However, the operation $p \mapsto p_\phi$ does not, in general, preserve composition or identity. Moreover, given p there may be several p_ϕ which successfully simulate p, and there may be no canonical choice. Note also that in case p was a partial operation, one may wish to impose conditions on where the simulating morphism p_ϕ should be defined; another question is whether the image of ϕ should be a well-behaved subset of $\mathbb{N}$ (for example, recursive or r.e.).

The example of a computable ring is almost an instance of this, with the only difference that in that case the numbering is multivalued.

Transition systems The simplest type of *transition system*, (S, τ), consists of a set of states, S, and a binary relation τ (see [12] for more on transition systems). Instead of $\tau(s_1, s_2)$ one usually writes $s_1 \longrightarrow s_2$. A *simulation* between such transitions systems $\alpha : (S, \tau) \longrightarrow (S', \tau')$, is a relation α from S to S' such that:

$$\begin{array}{ccc} S & \xrightarrow{\alpha} & S' \\ {\scriptstyle \tau}\downarrow & & \downarrow{\scriptstyle \tau'} \\ S & \xrightarrow[\alpha]{} & S' \end{array}$$

Notice that once a state, s_0 is simulated by s'_0 one must also be able to simulate any behavior from that state. In particular, one must be able to simulate each state to which s_0 can transition, although the second transition system can be more permissive. Of course, we may also consider the variant of this notion in which the simulation is strict, in which case a transition is present in the second transition system only if there is a transition in the first that it is simulating.

The situation becomes more interesting when one considers *labelled transitions*; instead of having a single transition relation τ, one can have a labelled

family $\{\tau_\alpha | \alpha \in \Sigma\}$. The notion of simulation extends readily to such systems, and the form of this system suggests another source of variation: need a transition $s_1 \xrightarrow{\alpha} s_2$ be simulated by a single transition $s_1' \xrightarrow{\alpha'} s_2'$ or could it be simulated by a sequence of transitions?

1.2 Contributions of the paper

In each of these examples, it is not that the structure is preserved but rather that the structure of the domain becomes somehow interpreted, or simulated, in the codomain. It is not immediately obvious that such morphisms compose or, indeed, what preservation properties they have. The purpose of this paper is to provide a categorical approach to simulations so that a uniform approach to their theory becomes possible. That the theory is pleasantly modular may already be suspected as the variations can be seen as corresponding to the choice of natural transformations and functors which are available in the ambient setting.

In this paper we have chosen to focus on simulations arising from partial map categories. This, admittedly, excludes examples which are deeply rooted in relations. However, it does include the motivating examples arising from realizability and computablity: these are, or can be reduced, to these more specialized simulations. It is our hope that after seeing the development in this paper it will not be hard for the reader to transfer the ideas to other settings (such as allegories). Furthermore, it should be clear that there is an even more abstract version of this development possible which based on fibrations in arbitrary bicategories. These directions need more attention than we can supply here and could usefully be developed to complete the understanding of this theory.

The central conceptual idea presented in the paper is the connection between the theory of simulations and that of fibrations; more concretely, we explain simulations between categories in terms of a Kleisli construction for the free preorder fibration monad. This allows for the application of the general theory of fibrations; moreover, by moving the theory into this abstract setting it also becomes clear how one could transfer the main ideas to other settings. The main technical result is concerned with the notion of simulation equivalence: we show that if two categories are equivalent in the 2-category of categories over a fixed base and strict simulations between them (and certain refinements as 2-cells) then the categories are already equivalent in the ordinary sense, modulo the splitting of some idempotents. In general, simulations are inherently non-functorial; therefore this result can be interpreted as saying that splitting of idempotents is the only obstruction to making simulation equivalences functorial.

Our application is a result promised in [4] concerning the construction of a Turing category from a PCA; in loc. cit. it was shown that each PCA gives rise to a cartesian category of partial computable maps - such a category may be regarded as a presentation-free representation of the PCA, in which computability may be studied. The question is then whether this construction is functorial in any useful way, and we will explain here how simulations of PCAs give rise to

categorical simulations on the level of their associated Turing categories. The result about simulation equivalences now shows that Turing categories are invariants for PCAs, i.e. that equivalent PCAs give rise to equivalent Turing categories (up to idempotent splitting). But much more is true: the 2-functor which associates to a PCA its Turing category is a local equivalence. Moreover, if we generalize PCAs to relative PCAs (which essentially amounts to picking a sub-PCA of global sections), then we obtain a 2-equivalence between relative PCAs and Turing categories under simulation.

The fibrational characterization of simulations will also tie together a number of strands from the literature concerning the nature of certain categories of (partitioned) assemblies arising in realizability. First of all, the construction of a generalized category of assemblies from certain functors appeared already in Birkedal's thesis [1]; there the question was raised what the universal property of this construction is. We shall give a precise answer to this question, and indeed discuss the 2-functoriality of the construction.

The question of what exactly happens when one constructs a category of (partitioned) assemblies from a PCA has also been addressed in Robinson and Rosolini's [14]. In their work, it was shown how the process is in fact an instance of a more general construction, dubbed the $\mathcal{F}$-construction. The input for this construction is a functor $p : \mathsf{C} \longrightarrow \mathsf{Par}$ from a partial map category into the category of sets and partial functions; the result is a category $\mathcal{F}(p)$, which is obtained by a variant of the gluing construction. This makes precise how the category of partitioned assemblies over a PCA $\mathbb{A}$ may be viewed as the result of adding $\mathbb{A}$-indexed coproducts to the category of sets. However, so far it has not been shown in which way the $\mathcal{F}$-construction is functorial.

The present paper adds this small but significant piece to the picture: we shall show that both the $\mathcal{F}$-construction and Birkedal's generalized assembly construction are best understood by considering the 2-monad whose Kleisli category gives the relevant notion of simulation. This 2-monad exists in a suitable 2-category of partial map categories; it is in this context that the functoriality and universal property are most easily obtained. The only price to pay is the fact that, since these constructions result in fibrations of partial map categories, one has to restrict to total maps in order to retrieve the original constructions.

1.3 Plan of the paper

We have chosen to present the material roughly in the same order as we discovered it: starting with the elementary definition, which is extracted directly from the motivating examples, and then moving to an explanation of how there is an elegant 2-categorical account in terms of fibrations. We hope that this order of presentation shows how the concrete examples naturally lead to the abstract formulation, which in turn is needed for the applications and to explain the connections with other work.

After a brief review of the theory of categories of partial maps, we begin (section 2) by giving a definition of a categorical simulation in elementary terms on the level of categories of partial maps and by establishing some basic prop-

erties. There are a number of variations possible; in particular, we discuss the difference between lax and strict, and between total and non-total simulations. We shall also explain how some of the examples can be viewed in this setting. The concept of simulation equivalence is an important one as it allows us to express the idea that two structures only differ in their algebraic presentation. We conclude this elementary section by providing some observations about what simulation equivalences can and cannot detect.

In section 3, we give a conceptual account of the notion of simulation in terms of fibrations. In particular, we discuss generalized assembly monads and prove that the Kleisli category for such a monad is isomorphic to a corresponding category of simulations. The assembly monad is nothing but the free preorder fibration monad (in a suitable enriched sense); in particular, it gives an answer to Birkedal's problem: the category of (partitioned) assemblies on a functor is the (total map category of the) free preorder fibration on that functor. We also prove our main technical result, which sheds light on simulation equivalences: any simulation equivalence can be straightened out (made functorial) provided one is willing to split idempotents. This means that, modulo idempotent splitting, from an equivalence between two categories in the simulation sense one may construct an actual equivalence of categories.

Section 4 concentrates on our main application of the abstract theory, namely a proof of the result promised in [4] stating that the 2-category of relative PCAs in a fixed base category C is equivalent to the 2-category of Turing categories and cartesian simulations over the base C. This shows the way in which the study of Turing categories is related to the study of relative PCAs.

R.1	$f\overline{f} = f$	
R.2	$\overline{f}\overline{g} = \overline{g}\overline{f}$	whenever $dom(f) = dom(g)$
R.3	$g\overline{f} = \overline{g}\overline{f}$	whenever $dom(f) = dom(g)$
R.4	$\overline{g}f = f\overline{gf}$	whenever $cod(f) = dom(g)$

Table 1: Axioms for a restriction combinator

2 Basic simulations

This section introduces the elementary definition of simulation between functors of categories of partial maps. For convenience, we first give a brief review of partial map categories and related matters before introducing the concept of simulation and some of its possible variants. Simulations over a fixed base form a preorder-enriched category. This means that equivalences between simulations can be considered and some elementary properties of these are developed. We also briefly revisit some of our examples to show how they can be viewed in this sense.

2.1 Review of partial map categories

We now present a concise review of restriction categories, which are a convenient algebraic formulation of abstract categories of partial maps. More detailed accounts can be found in [5, 6].

Definition 2.1 (Restriction category)**.** A *restriction category* is a category C endowed with a combinator $\overline{(-)}$, sending $f : A \longrightarrow B$ to $\overline{f} : A \longrightarrow A$, such that the axioms in Table 1 are satisfied.

The idea is that the map $\overline{f}$ measures the degree of partiality of f; we refer to $\overline{f}$ as the domain (of definition) of f. The prime example of a restriction category is Par, the category of sets and partial functions. Another example, relevant to this paper, is the category with one object $\mathbb{N}$ and with partial recursive functions as morphisms. In this example, the domains correspond to the r.e. sets.

A morphism f in a restriction category for which $\overline{f} = 1$ is called a *total map*. The total maps in a restriction category C form a subcategory denoted $\mathsf{Tot}(\mathsf{C})$.

Recall that a map f is *idempotent* if $ff = f$. It is easily verified that maps of the form $\overline{f}$ are idempotent. If a map f satisfies $f = \overline{f}$, then we say that f is a *restriction idempotent*. Given two objects A and B, we say that A is a *retract* of B if there exist maps $m : A \longrightarrow B$ and $r : B \longrightarrow A$ such that $rm = 1_A$. Note that the embedding part m, being a section, has to be monic and hence total, but that the retraction r need not be total. The composite mr is an idempotent on B, and the object A is said to be a splitting of the idempotent mr. In general, not every idempotent is a restriction idempotent; however, if mr is a restriction idempotent then the retraction r is uniquely determined by m.

Restriction categories are locally ordered: given parallel maps $f, g : A \longrightarrow B$, say that

$$f \leq g \quad \Leftrightarrow \quad f = g\overline{f}.$$

This ordering expresses the idea that f and g agree wherever f is defined, but that g may be more defined than f.

Splitting Idempotents. Given a category C and a class E of idempotents in C, one can formally split the idempotents in E; the resulting category is typically denoted by $\mathcal{K}_E(\mathsf{C})$. We indicate the construction of $\mathcal{K}_E(\mathsf{C})$. Its objects are pairs (X, e) where X is an object of C and $e \in E$ is an idempotent on X. A morphism $(X, e) \longrightarrow (Y, f)$ is a map $k : X \longrightarrow Y$ in C for which $ke = k = fk$. Provided E contains the identity arrows, there is a functor $\mathsf{C} \longrightarrow \mathcal{K}_E(\mathsf{C})$ sending X to $(X, 1_X)$; this functor preserves the restriction structure.

Note that an object (X, e) in $\mathcal{K}_E(\mathsf{C})$ becomes a retract of $(X, 1_X)$ via the morphisms $(e, e) : (X, e) \triangleleft (X, 1_X)$.

One can verify that in the case that C is actually a restriction category, then so is the splitting $\mathcal{K}_E(\mathsf{C})$. Also, the inclusion functor $\mathsf{C} \longrightarrow \mathcal{K}_E(\mathsf{C})$ then preserves the restriction structure. If two categories become equivalent after splitting idempotents we shall say that they are *Morita-equivalent*.

For the reader who is familiar with partial map categories which are presented via a stable system of monics on a category, we remark that every such partial map category is a restriction category in which all restriction idempotents split and that, conversely, every such restriction category may be presented in that way.

Restriction functors. Given two restriction categories C, D, a *restriction functor* $F : \mathsf{C} \longrightarrow \mathsf{D}$ is an ordinary functor from C to D which must satisfy the condition that $F(\overline{f}) = \overline{F(f)}$, i.e. that it preserves restriction. It follows that F preserves the enrichment as well, in the sense that $f \leq g$ implies $F(f) \leq F(g)$.

Restriction transformations. There is more than one notion of natural transformation between restriction functors, and, as we shall see, the theory of simulations directly depends on the particular choice of natural transformation. We mention here the most common types of transformation. Let $F, G : \mathsf{C} \longrightarrow \mathsf{D}$ be two restriction functors, and let $\alpha = \{\alpha_X : FX \longrightarrow GX | X \in Ob(\mathsf{C})\}$ be a family of maps. If each square

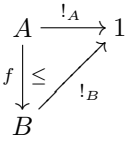
$$\begin{array}{ccc} FX & \xrightarrow{\alpha_X} & GX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FY & \xrightarrow[\alpha_Y]{} & GY \end{array}$$

commutes on the nose, then we say that α is a *strict* natural transformation. If it commutes up to inequality $\alpha_Y Ff \leq Gf\alpha_X$ then we say that α is a *lax* natural transformation. In case all components α_X are total maps, then we call α *total*. Finally, if we have equality for each $f = \overline{f}$ (but not necessarily for general f)

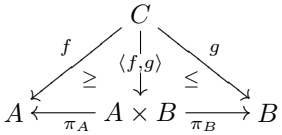
then we say that α is *tight*. There are various combinations possible; we shall be mainly interested in the following:

- $\mathfrak{Rcat}$ is the 2-category of restriction categories, restriction functors and *total, strict* natural transformations
- $\mathfrak{Rcat}_l$ the 2-category with the same 0-cells and 1-cells as $\mathfrak{Rcat}$, but with *lax total* natural transformations as arrows.
- $\mathfrak{Rcat}_p$ stands for the 2-category with again the same 0-cells and 1-cells, but with *lax, partial* transformations.
- $\mathfrak{Rcat}_t$ is the 2-category with again the same 0-cells and 1-cells, but with *tight, partial* transformations.

Cartesian structure. An object 1 in a restriction category C is said to be a *restriction terminal object* if for each object A there is a unique total map $!_A : A \longrightarrow 1$, such that $!_1 = 1$, and for each $f : A \longrightarrow B$ we have $!_B f = !_A \overline{f}$, as in the diagram below.



A *partial product* of two objects A, B is an object $A \times B$ equipped with total projections $\pi_A : A \times B \longrightarrow A$ and $\pi_B : A \times B \longrightarrow B$, such that for each C and each pair of maps $f : C \longrightarrow A, g : C \longrightarrow B$, there is a unique map $\underline{\langle f, g \rangle} : C \longrightarrow A \times B$ with the properties that $\pi_A \langle f, g \rangle \leq f, \pi_B \langle f, g \rangle \leq g$ and $\overline{\langle f, g \rangle} = \overline{f}\overline{g}$.



Thus, these finite limits are limits in the enriched sense; still, we shall simply speak about terminal objects and products, as there will be no chance of confusion. If a restriction category has a terminal object and binary (hence all finite) products, we say that the category is *cartesian*. Similarly, a restriction functor preserving all of this structure is called a *cartesian* functor. Each of the above 2-categories of restriction categories has a cartesian variant; e.g., we denote by $\mathfrak{CartRcat}_l$ the 2-category of cartesian restriction categories, cartesian functors and lax total transformations.

2.2 Simulations over a base

In what follows all categories and functors are restriction categories and functors (although everything works for ordinary categories).

Take an arbitrary but fixed base category C. We consider categories over C, that is, functors $F : \mathsf{D} \longrightarrow \mathsf{C}$; it is between such functors over the base that we wish to consider simulations.

Definition 2.2. Let $\mathsf{D} \xrightarrow{F} \mathsf{C}$ and $\mathsf{E} \xrightarrow{G} \mathsf{C}$ be two categories over C. A *simulation* from F to G consists of:

- An object map $K : \mathsf{Obj}(\mathsf{D}) \longrightarrow \mathsf{Obj}(\mathsf{E})$
- A family of maps $\alpha_D : F(D) \longrightarrow G(K(D))$ such that for every $f : D \longrightarrow D'$ in D there is an $f_\alpha : K(D) \longrightarrow K(D')$ such that

$$\begin{array}{ccc} FD & \xrightarrow{\alpha_D} & GK(D) \\ {\scriptstyle Ff}\downarrow & \leq & \vdots\,{\scriptstyle Gf_\alpha} \\ FD' & \xrightarrow[\alpha_{D'}]{} & GK(D') \end{array} \tag{1}$$

  commutes up to inequality. We shall say that f_α is a *(lax) realizer* of f under the simulation (K, α). In case the square commutes on the nose, we call f_α a *strict* realizer.

The intuition is that an assignment $(D \xrightarrow{f} D') \mapsto (KD \xrightarrow{f_\alpha} KD')$ can be chosen, but not necessarily in a functorial way. If a choice can be made which turns K into a genuine functor (and α into a (lax) natural transformation $F \Rightarrow GK$) then we call the simulation (K, α) *functorial*. It is important to note that a choice of realizers can be made, but is not part of the data of a simulation.

If (K, α) is a simulation from $\mathsf{D} \xrightarrow{F} \mathsf{C}$ to $\mathsf{E} \xrightarrow{G} \mathsf{C}$ we will often write $(K, \alpha) : (\mathsf{D} \xrightarrow{F} \mathsf{C}) \rightsquigarrow (\mathsf{E} \xrightarrow{G} \mathsf{C})$, or also $(K, \alpha) : F \rightsquigarrow G$ (or even $(K, \alpha) : \mathsf{D} \rightsquigarrow \mathsf{E}$ if no confusion is likely to arise).

Special cases. Due to the fact that we work in partial map categories, there are various special cases to consider, corresponding to the various types of natural transformation considered in the previous section.

First, for a simulation $(K, \alpha) : \mathsf{D} \rightsquigarrow \mathsf{E}$ as above, if each map f has a strict realizer then we call the simulation (K, α) a *strict simulation*. By contrast, we often refer to the general case as *lax simulations*.

Second, a simulation (K, α) which has all its components α_X total will be called a *total simulation*.

There is another relevant condition, which we call *tightness*: a simulation is called tight if every domain can be strictly simulated. The following lemma explains some of the connections between these conditions.

Lemma 2.3. *Let (K, α) be a tight simulation. Then:*

(i) *Every restriction idempotent can be strictly simulated by a restriction idempotent.*

(ii) *When α_Y is total, then every $f : X \longrightarrow Y$ can be strictly simulated.*

(iii) *If (K, α) is total, then it is also strict.*

Proof. For (i), observe that if $e = \overline{e}$ is strictly simulated by e_α, then it will also be strictly simulated by $\overline{e_\alpha}$. For (ii), suppose f is simulated by f_α; let also $(\overline{f})_\alpha$ strictly simulate $\overline{f}$. Then it is easily checked that $f_\alpha(\overline{f})_\alpha$ strictly simulates f. Finally, (iii) is immediate from (ii). □

Thus for total simulations, the notions of tightness and strictness agree, but for non-total simulations it often is tightness which is the better behaved notion (see, for example, Proposition 2.4 below).

Simulations as an enriched category. Clearly simulations may be composed: given two composable simulations $(K, \alpha) : F \rightsquigarrow G$ and $(L, \beta) : G \rightsquigarrow H$, their composite is obtained as one would compose morphisms in the lax slice, i.e. by taking the object map to be LK and the component map to be $\beta K \circ \alpha$:

$$\begin{array}{ccccc} FD & \xrightarrow{\alpha_D} & GK(D) & \xrightarrow{\beta_{KD}} & HLK(D) \\ {\scriptstyle Ff}\downarrow & \leq & \vdots\,{\scriptstyle G(f_\alpha)} & \leq & \vdots\,{\scriptstyle H((f_\alpha)_\beta)} \\ FD' & \xrightarrow[\alpha_{D'}]{} & GK(D') & \xrightarrow[\beta_{KD'}]{} & HLK(E') \end{array}$$

Moreover, it is readily seen that there exists an identity simulation on each object F. Thus, simulations over C form a category $\mathfrak{Sim}_p(\mathsf{C})$. Each of the subclasses of simulations mentioned above (total, strict, tight) gives rise to a subcategory of $\mathfrak{Sim}_p(\mathsf{C})$. We shall write $\mathfrak{Sim}(\mathsf{C})$ for the subcategory on the strict, total simulations, $\mathfrak{Sim}_l(\mathsf{C})$ for the subcategory on the lax, total simulations, and $\mathfrak{Sim}_t(\mathsf{C})$ for the subcategory on the tight simulations.

Given two parallel simulations $(K, \alpha), (L, \beta) : F \rightsquigarrow G$, we say that (K, α) is a *refinement* of (L, β), notation $(K, \alpha) \preceq (L, \beta)$, in case for each object D there is a $\lambda_D : K(D) \longrightarrow L(D)$ such that

$$\begin{array}{ccc} FD & \xrightarrow{\alpha_D} & GK(D) \\ & {\scriptstyle \beta_D}\searrow \; \leq & \vdots\,{\scriptstyle G\lambda_D} \\ & & GL(D) \end{array} \tag{2}$$

commutes up to inclusion. Clearly the relation of refinement is transitive and reflexive and, thus, the homsets of $\mathfrak{Sim}_p(\mathsf{C})$ are preorders. Furthermore, composition preserves this preorder: when $(K, \alpha) \preceq (L, \beta) : F \rightsquigarrow G$ and $(M, \gamma) : G \rightsquigarrow H$ then $(M, \gamma)(K, \alpha) \preceq (M, \gamma)(L, \beta)$ as

$$\begin{array}{ccccc} FD & \xrightarrow{\alpha_D} & GK(E) & \xrightarrow{\gamma_{KD}} & HMK(D) \\ & {\scriptstyle \beta_D}\searrow \; \leq & \downarrow{\scriptstyle G\lambda_D} & \leq & \downarrow{\scriptstyle H((\lambda_D)_\beta)} \\ & & GL(D) & \xrightarrow[\beta_{LD}]{} & HML(D) \end{array} \tag{3}$$

and similarly when we compose on the other side. Thus, $\mathfrak{Sim}_p(\mathsf{C})$ is preorder enriched, as are its subcategories $\mathfrak{Sim}_l(\mathsf{C})$, $\mathfrak{Sim}(\mathsf{C})$ and $\mathfrak{Sim}_t(\mathsf{C})$. We thus have four possible 2-categories of simulations; for convenience we list these:

- $\mathfrak{Sim}(\mathsf{C})$ is the 2-category whose objects are functors $F : \mathsf{D} \longrightarrow \mathsf{C}$, morphisms are total, strict simulations and 2-cells are refinements
- $\mathfrak{Sim}_l(\mathsf{C})$ is the 2-category whose objects are functors $F : \mathsf{D} \longrightarrow \mathsf{C}$, morphisms are total, lax simulations and 2-cells are refinements
- $\mathfrak{Sim}_p(\mathsf{C})$ is the 2-category whose objects are functors $F : \mathsf{D} \longrightarrow \mathsf{C}$, morphisms are lax, partial simulations and 2-cells are refinements
- $\mathfrak{Sim}_t(\mathsf{C})$ is the 2-category whose objects are functors $F : \mathsf{D} \longrightarrow \mathsf{C}$, morphisms are tight simulations and 2-cells are refinements

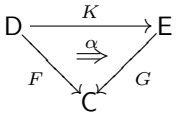
In what follows we shall sometimes wish to make a statement which holds for each of the four 2-categories; in that case, we shall use a wildcard notation $\mathfrak{Sim}_*(\mathsf{C})$.

In the cases where the component maps are partial maps there is a bit more structure: both $\mathfrak{Sim}_p(\mathsf{C})$ and $\mathfrak{Sim}_t(\mathsf{C})$ are also restriction categories:

Proposition 2.4. *Both partial simulations and tight simulations admit a pointwise restriction structure which is compatible with the preorder enrichment.*

Proof. As indicated, the restriction of a simulation (K, α) will be the simulation whose object map is the identity and whose component at X is $\overline{\alpha_X}$. The verifications are routine. □

It should be clear that there is a 2-functor from the lax slice $\mathfrak{Rcat}_* /\!\!/ \mathsf{C} \hookrightarrow \mathfrak{Sim}_*(\mathsf{C})$ whose objects are again functors into C, but where a morphism is a diagram



where K is a functor and α a natural transformation (of appropriate type). Thus, a simulation is functorial precisely when it is in the image of this inclusion. Clearly, this is a strict requirement, and we are often more interested in whether a given simulation is in the essential image of this inclusion, i.e. whether it is isomorphic to a functorial simulation. We shall sometimes speak of "straightening out" a given simulation: by this, we simply mean giving a functorial simulation which is isomorphic to the given one. The question of when this is possible will be addressed in Section 3.

2.3 Examples

We now discuss a number of examples to illustrate the definition.

Affine maps. Let M, N be two additive monoids, regarded as one-object categories. By considering M as a right M-set, we get a functor $M \longrightarrow \mathsf{Set}$, and similarly for N. If $\alpha : M \longrightarrow N$ is an affine map of the form $\alpha(x) = n + \beta(x)$ for some homomorphism β and $n \in N$, then α gives rise to (total) simulation from M to N. Indeed, all we need is that for each $m \in M$ there is an $m_\alpha \in N$ for which

$$\begin{array}{ccc} M & \xrightarrow{\alpha} & N \\ {\scriptstyle -+m}\downarrow & & \downarrow{\scriptstyle -+m_\alpha} \\ M & \xrightarrow{\alpha} & N \end{array} \tag{4}$$

commutes, i.e. such that $\alpha(x + m) = \alpha(x) + m_\alpha$ for all x. But evidently we may let $m_\alpha = \beta(m)$.

Conversely, if α is a simulation from M to N then repeated use of 4 gives that

$$\alpha(n + m) = \alpha(0) + m_\alpha + n_\alpha.$$

Now in general the mapping $m \mapsto m_\alpha$ is not unique. Moreover, it will in general not be a homomorphism, unless the monoid is cancellative (for in that case, we have $\alpha(0) + (m + n)_\alpha = \alpha(0) + m_\alpha + n_\alpha$).

Transition systems. The notion of labelled transition system discussed in the introduction was based on relations and does not, therefore, fit into the partial map setting without distortion. For sake of illustration we consider a functional variant here; however, in order to properly treat the relational version one should mimick the process in the setting of categories of relations.

Suppose that we have two *functional* labelled transition systems (meaning that the transition relations are actually partial functions) $(S, \{\sigma_a | a \in \Sigma\})$ and $(T, \{\tau_a | a \in \Theta\})$. We may form the free restriction category $\mathsf{F}(\Sigma)$ generated by the label set Σ. (For an exposition of free restriction categories, see [3].) Thus, $\mathsf{F}(\Sigma)$ has one object $*$, and the morphisms are freely generated by the elements of Σ. Note that, because this is a free category, none of the generating morphisms are total. Now to give a Σ–labelled transition system amounts to specifying a functor σ from this free category into Par, the category of sets and partial maps. Similarly, the transition system T may be viewed as a functor $\tau : \mathsf{F}(\Theta) \longrightarrow \mathsf{Par}$.

Now consider a categorical simulation $\sigma \rightsquigarrow \tau$. Such a simulation, by the property of free categories, is determined by how the basic morphisms σ_a are simulated. It should be clear that the result is a very general notion of functional simulation; also, it should be noted that in this context it may not be unreasonable to consider partial simulations; this would be appropriate if one is not interested in simulating the entire system but only in the part which can be reached from a certain $s \in S$. Finally, the difference between strict and lax simulations also has a clear interpretation here: a strict simulation would have the (very strong) property that s is in the domain of a transition σ if and only if $\alpha(s)$ is in the domain of the simulating transition σ_α.

PCAs. As already indicated in the introduction, we shall be concentrating here on functional simulations of PCAs. For concreteness, we work over the category Par of sets and partial functions, but everything readily generalizes to any cartesian base category.

Let $\mathbb{A} = (A, \bullet)$ be a PCA. First, we may consider the monoid of partial computable endofunctions of $\mathbb{A}$, i.e. those functions of the form $a \bullet -$ for some $a \in A$, and view this as a one-object category $M(\mathbb{A})$ with a faithful functor to Par. Given another PCA $\mathbb{B} = (B, \bullet)$, viewed similarly as a faithful functor $M(\mathbb{B}) \longrightarrow \mathsf{Par}$, we may now consider simulations between the two.

First, let $\alpha : \mathbb{A} \longrightarrow \mathbb{B}$ be a *strict* simulation of PCAs. Pick a witness $u \in B$ such that $u \bullet \alpha(x) \bullet \alpha(y) = \alpha(x \bullet y)$ for all x, y (this is Kleene-equality of partial terms). In particular, for fixed $a \in A$ we have that the element $a_\alpha = u \bullet \phi(a)$ satisfies

$$a_\alpha \bullet \alpha(y) = \alpha(a \bullet y)$$

for each y. Thus, α constitutes a categorical simulation between $M(\mathbb{A})$ and $M(\mathbb{B})$, which by construction is also strict.

Since every PCA has internal pairing, there are idempotents in $M(\mathbb{A})$ which encode finite products. Splitting these idempotents results in the Turing category $\mathsf{Comp}(\mathbb{A})$ whose objects are the formal finite powers of $\mathbb{A}$ and whose morphisms are the $\mathbb{A}$-computable maps (now of several variables). There is an obvious faithful functor $\delta_{\mathbb{A}} : \mathsf{Comp}(\mathbb{A}) \longrightarrow \mathsf{Par}$, and similarly we obtain $\delta_{\mathbb{B}} : \mathsf{Comp}(\mathbb{A}) \longrightarrow \mathsf{Par}$.

Now a simulation $\alpha : \mathbb{A} \longrightarrow \mathbb{B}$ also induces a categorical simulation $\delta_{\mathbb{A}} \rightsquigarrow \delta_{\mathbb{B}}$, by straightforward extension of the reasoning for the one-object case. However, note that the induced simulation on the level of Turing categories has the special feature that it preserves the finite product structure on the nose; such a simulation will later be defined properly and called *cartesian*. To sum up, we have:

Proposition 2.5. *Every simulation of PCAs $\alpha : \mathbb{A} \longrightarrow \mathbb{B}$ gives rise to a simulation $\mathsf{Comp}(\mathbb{A}) \rightsquigarrow \mathsf{Comp}(\mathbb{B})$.*

We may extend this result by defining the 2-category $\mathfrak{PCA}$ whose objects are PCAs, morphisms are strict simulations and whose 2-cells are given by $\alpha \preceq \beta$ iff $u \bullet \alpha(x) = \beta(x)$ for some $u \in B$. Then we have in fact a 2-functor

$$\mathsf{Comp}(-) : \mathfrak{PCA} \longrightarrow \mathfrak{Sim}(\mathsf{Par}).$$

Note that so far we have used strict simulations; we can also obtain a similar statement where we replace $M(\mathbb{A})$ by the monoid of *realizable* endofunctions (those majorized by a computable map), and then consider lax simulations.

It is natural to wonder about a converse: is every cartesian simulation between $\delta_{\mathbb{A}}$ and $\delta_{\mathbb{B}}$ induced by a simulation of PCAs? The problem is whether such a simulation indeed neatly sends the object A^n to B^n, so that we indeed have a component map $A \longrightarrow B$. In Section 4 we shall prove that, up to splitting of idempotents, this is indeed the case.

Gödel numbering In the introduction we hinted at the general pattern behind this phenomenon. Here, let us look at a simple instance, namely the encoding of a datatype such as the Booleans. Let $\mathbb{B} = \{T, F\}$ be the set of Booleans, and define an encoding into the natural numbers as

$$\phi(F) = 0, \qquad \phi(T) = 1.$$

Now any n-ary operation on $\mathbb{B}$ can be simulated in the natural numbers.

Consider the full cartesian subcategory $\mathcal{B}$ of **Par** on the finite powers of $\mathbb{B}$. Then there is a categorical simulation from the forgetful $\mathcal{B} \longrightarrow \mathsf{Par}$ into $\mathsf{Comp}(\mathcal{K}_1)$.

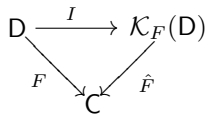
The same idea works for other datatypes, such as lists: any choice of encoding of the datatype gives rise to a simulation between the category of finite powers of that datatype and all relevant n-ary (partial) functions to the category $\mathsf{Comp}(\mathcal{K}_1)$. If it is important that the operations on the datatype are simulated not just by general recursive functions but by certain special ones (such as primitive recursive functions) then one cuts down to the relevant subcategory of $\mathsf{Comp}(\mathcal{K}_1)$.

Also, one often has the desired property that different choices of numbering result in isomorphic simulations.

2.4 Simulation equivalence

Since simulations over C form an order-enriched category, we may speak of adjunctions and equivalences in that category. If two categories over the same base are equivalent in the category $\mathfrak{Sim}_p(\mathsf{C})$, we shall say that they are *lax simulation equivalent*, and if they are equivalent in the category $\mathfrak{Sim}(\mathsf{C})$ we shall call them *strict simulation equivalent*. (It will turn out that equivalences are always total, so there is no need to consider other cases.) In this section we make a few observations about what can, and what cannot be detected by these two kinds of equivalences.

Idempotents. For a functor $\mathsf{D} \xrightarrow{F} \mathsf{C}$, consider the class of idempotents $E(F) = \{e | F(e) \text{ splits in } \mathsf{C}\}$. Now say that F is C-*split* if every idempotent e in $E(F)$ already splits in D. Formally splitting the idempotents in $E(F)$ gives rise to a category $\mathcal{K}_F(\mathsf{D})$ and a commutative diagram



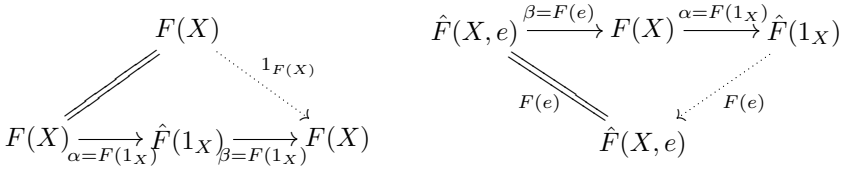
Lemma 2.6. *For any $F : \mathsf{D} \longrightarrow \mathsf{C}$ there is a strict, total simulation equivalence between F and the relative idempotent splitting $\hat{F}$.*

Proof. Define (I, α) to be the simulation arising from the inclusion of F into $\hat{F}$; thus the object map sends X to 1_X and the component $\alpha_X : FX \longrightarrow \hat{F}(1_X)$ is the identity. Then define a pseudo-inverse $(L, \beta) : \hat{F} \rightsquigarrow F$ with components

$\hat{F}(A,e) \xrightarrow{\hat{F}(e)} A$ (this map is just the monic part of the splitting of $F(e)$). Given $f : (A,e_1) \longrightarrow (B,e_2)$ the following diagram shows that f is realized by itself:

$$\begin{array}{ccc} \hat{F}(A,e_1) & \xrightarrow{\hat{F}(e_1)} & F(A) \\ {\scriptstyle \hat{F}(f)}\downarrow & & \downarrow{\scriptstyle F(f)} \\ \hat{F}(B,e_2) & \xrightarrow[\hat{F}(e_2)]{} & F(B) \end{array}$$

The unit at X and counit at (X,e) are given by:



which are clearly isomorphisms. □

Notice that every simulation equivalence is necessarily total, since any equivalence (K,α) must in particular have a right adjoint (L,β); then the unit of the adjunction $\eta : 1 \longrightarrow LK$ forces K to be total.

Faithful image. Since the defining diagrams for a simulation live in the base category C, simulations over C cannot distinguish between a functor $F : \mathsf{D} \longrightarrow \mathsf{C}$ and its faithful image. More precisely, given F, we have a congruence $f \sim g \Leftrightarrow Ff = Fg$ on D; then F factors as $\mathsf{D} \xrightarrow{Q} F/\sim \xrightarrow{F_0} \mathsf{C}$, and we find:

Lemma 2.7. *For any $F : \mathsf{D} \longrightarrow \mathsf{C}$ there is a strict simulation equivalence between F and its faithful image F_0.*

Proof. A direct proof is easy, but this will also be immediate from the considerations in the next section. □

Down-closure. Finally, we discuss an aspect which illustrates the difference between a strict and a lax simulation. First, a definition:

Definition 2.8. Let $F : \mathsf{D} \longrightarrow \mathsf{C}$ be a faithful restriction functor. We say that F is *down-closed* if $f' \leq F(g)$ implies that $f' = F(f)$ for some f in D.

Every faithful F may be factored as $\mathsf{D} \xrightarrow{J} \downarrow\mathsf{D} \xrightarrow{F_1} \mathsf{C}$, where $\downarrow\mathsf{D}$ is the down-closure; it has the same objects as D, but with homsets

$$\downarrow\mathsf{D}[A,B] = \{f' : FA \longrightarrow FB | f' \leq F(f) \text{ for some } f : A \longrightarrow B\}.$$

Lemma 2.9. *With the notation as above, the functors F and its down-closure F_1 are equivalent under lax, total simulation.*

Proof. Evident. □

This leads to the phenomenon that under lax simulations, one can have a total category equivalent to a non-total one.

We summarize the above observations by concluding that, when one considers functors up to simulation equivalence, one may assume that they are relatively split faithful functors into the base category. If one works up to lax simulation equivalence, one may in addition assume down-closedness.

3 Simulations and fibrations

The concept of a simulation as defined in the previous section, while motivated by the examples, may not look very appealing from a categorical point of view. The reason, of course, lies in the fact that the categorical dogma is to look for functoriality, while the notion at hand is essentially non-functorial. In this section, we provide a much more conceptual viewpoint which makes clear that simulations occupy a natural place in the theory of fibrations. More concretely, we shall show that simulations may be viewed as a Kleisli construction, thereby turning them into a functorial notion after all.

We first discuss preorder fibrations in 2-categories of partial map categories. The notion of fibration is of course dependent on the type of natural transformation, and we sketch for each of the four types of natural transformation discussed in Section 2 what the corresponding notion of fibration looks like. In the case of total transformations, a convenient description in terms of comma objects is available.

Preorder fibrations are algebras for a monad, which we call the assembly monad. The main point is that the Kleisli category for this monad is isomorphic to the category of simulations (all over a fixed base category). This viewpoint also allows us to view the problem of straightening out a given simulation as a lifting problem.

3.1 Preorder fibrations and assemblies

There are many possible ways to extend the notion of a fibration to the setting of partial map categories. Since fibrations are an inherently 2-categorical notion, we have to specify in which 2-category our partial map categories live. Recall our notation for various such 2 categories:

- $\mathfrak{Rcat}$ is the 2-category whose objects are restriction categories, whose morphisms are restriction functors and whose 2-cells are total, strict natural transformations.
- $\mathfrak{Rcat}_l$ is the 2-category with the same objects and arrows as $\mathfrak{Rcat}$, but whose 2-cells are *lax* natural transformations with total components.
- $\mathfrak{Rcat}_p$ is the 2-category with the same objects and arrows as $\mathfrak{Rcat}$, but whose 2-cells are lax, partial natural transformations.
- $\mathfrak{Rcat}_t$ is the 2-category with the same objects and arrows as $\mathfrak{Rcat}$, but whose 2-cells are tight, partial natural transformations.

Our main objective in this section is to show that in each of the four categories above, one may form the free preorder fibration on a functor, and that this construction is monadic in a suitable 2-categorical sense. We shall give the constructions for $\mathfrak{Rcat}_p$; each of the other three 2-categories will inherit the structure. Throughout the rest of this section, C denotes a fixed but arbitrary

restriction category. We write $\mathfrak{Rcat}_p/\mathsf{C}$ for the slice (2-)category of $\mathfrak{Rcat}_p$ over C.

Given a functor $F : \mathsf{D} \longrightarrow \mathsf{C}$, we factorize F through a new category $\mathsf{Asm}_p(F)$. This category is described as follows: it has as objects triples $(C, c : C \longrightarrow FX, X)$ where C is an object of C, X and object of D and $c : C \longrightarrow FX$ a morphism in C; a morphism from $(C, c : C \longrightarrow FX, X)$ to $(D, d : D \longrightarrow FY, Y)$ is an $f : C \longrightarrow D$ where such that there exists a $v : X \longrightarrow Y$ making the diagram

$$\begin{array}{ccc} C & \xrightarrow{c} & FX \\ {\scriptstyle f}\downarrow & \leq & \downarrow{\scriptstyle Fv} \\ D & \xrightarrow[d]{} & FY \end{array}$$

commute up to inequality. Usually we simply write $(C \xrightarrow{c} FX)$ for an object of $\mathsf{Asm}_p(F)$.

There is a projection functor $\delta_F : \mathsf{Asm}(F) \longrightarrow \mathsf{C}$ which sends $(c : C \longrightarrow FX)$ to C, and a morphism f as above to itself. Note that this functor is faithful.

Lemma 3.1. *The category* $\mathsf{Asm}(F)$ *is a restriction category in such a way that the projection* δ_F *is a restriction functor.*

Proof. Of course, for a morphism f as in the above diagram, we let the restriction of f be $\overline{f}$. It is straightforward to verify the axioms. □

We next explain the sense in which $\mathsf{Asm}_p(F)$ is the free preorder fibration on F.

First, let us say that a functor $F : \mathsf{D} \longrightarrow \mathsf{C}$ is a fibration (in $\mathfrak{Rcat}_p$) if for every $c : C \longrightarrow FX$ there is a cartesian lift in the usual sense, and when in addition there are unique lifts of 2-cells as indicated in the diagram below.

$$\begin{array}{ccccc} Y & & & & \\ & \searrow^{f} & & & \\ & {\scriptstyle k}\swarrow & \leq & & \\ & & c^*(X) & \xrightarrow[c^*]{} & X \end{array} \qquad (5)$$

$$\begin{array}{ccc} & \downarrow{\scriptstyle F} & \\ FY & \searrow^{Ff} & \\ & {\scriptstyle h}\swarrow \quad \leq & \\ & C \xrightarrow[c]{} & FX \end{array}$$

Thus, for every f, h with $ch \leq Ff$ there is a unique k with $c^*k \leq f$.

It is straightforward to verify that $\delta_F : \mathsf{Asm}_p(F) \longrightarrow \mathsf{C}$ indeed has this property. Now since this functor is also faithful, the fibres will be preorders.

Theorem 3.2. *The assignment $F \mapsto \mathsf{Asm}_p(F)$ is a 2-monad on the slice 2-category $\mathfrak{Rcat}_p/\mathsf{C}$. The pseudo-algebras for this monad are the (cloven) preorder fibrations on C.*

Proof. We indicate the constructions (which are analogous to those needed to show that the comma category monad is the free fibration monad, see [16]). First of all, $\mathsf{Asm}_p(-)$ is functorial: given a functor $H : (\mathsf{D}, F) \longrightarrow (\mathsf{E}, G)$ over C, the induced $\mathsf{Asm}_p(H) : \mathsf{Asm}_p(F) \longrightarrow \mathsf{Asm}_p(G)$ sends an object $c : C \longrightarrow FX$ to $c : C \longrightarrow GHX = FX$.

There is a unit $\eta = \eta_F : F \longrightarrow \mathsf{Asm}_p(F)$, which sends an object X of D to $FX \xrightarrow{1} FX$. This is a restriction functor over C.

The multiplication $\mu = \mu_F : \mathsf{Asm}_p(\delta_F) \longrightarrow \mathsf{Asm}_p(F)$ at F sends an object $C \xrightarrow{c} (\delta_F(D \xrightarrow{d} FX))$ to the composite $C \xrightarrow{dc} FX$. Again, this is a restriction functor over C. □

The above result specializes to the three subcategories $\mathfrak{Rcat}$, $\mathfrak{Rcat}_l$ and $\mathfrak{Rcat}_t$. In the case of $\mathfrak{Rcat}$ and of $\mathfrak{Rcat}_l$, a more conceptual description is possible: both these 2-categories admit comma objects. As is well-known from [16], the comma construction provides us with a factorization of a functor as a final functor followed by a fibration. Now the functor $\delta_F : \mathsf{Asm}(F) \longrightarrow \mathsf{C}$ may be obtained by factoring the free fibration $\mathsf{C}/F \longrightarrow \mathsf{C}$ as a quotient (bijective on objects) followed by a faithful functor. From this description it is immediate that $\mathsf{Asm}(-)$ is a monad (since it is a factorization of a monad morphism), and that its algebras are fibrations.

Remark 3.3. The reason why the category $\mathfrak{Rcat}_p$ does not admit comma objects is the following: if one tries to construct the comma category C/F in the obvious manner, then one finds that this is a well-defined category, but that it is impossible to put a restriction structure on it in such a manner that both projection functors preserve this restriction. The first projection still works, but it is the second projection here which is problematic; fortunately, the assembly construction still works, since the second projection is not required to be a restriction functor but merely a simulation!

3.2 Simulations as a Kleisli construction

We now make the connection between simulations and fibrations, by proving the following theorem.

Theorem 3.4. *There is a 2-isomorphism between the 2-categories $\mathfrak{Sim}_*(\mathsf{C})$ and $\mathsf{Kl}(\mathsf{Asm}_*(-))$.*

Since the categories $\mathfrak{Sim}_*(\mathsf{C})$ and $\mathsf{Kl}(\mathsf{Asm}_*(-))$ have the same objects, we begin by showing how a simulation (K, α) from $\mathsf{D} \xrightarrow{F} \mathsf{C}$ to $\mathsf{E} \xrightarrow{G} \mathsf{C}$ gives rise to a functor $\mathsf{D} \xrightarrow{\Phi(\alpha)} \mathsf{Asm}_*(G)$ over C.

An object D of D is sent by $\Phi(\alpha)$ to $\alpha_D : FD \longrightarrow GKD$, the component of the simulation (K, α) at D. On morphisms, we get

$$\begin{array}{ccc} D & & FD \xrightarrow{\alpha_D} GKD \\ {\scriptstyle f}\downarrow & \mapsto & {\scriptstyle Ff}\downarrow \quad \leq \quad \downarrow{\scriptstyle G(r_f)} \\ D' & & FD' \xrightarrow{\alpha_{D'}} GKD' \end{array}$$

where r_f is a realizer for f. Since F is a functor, so is $\Phi(\alpha)$.

Next, suppose we are given a functor $M : \mathsf{D} \longrightarrow \mathsf{Asm}_*(G)$ over C, then write $M_1(D) : FD \longrightarrow GM_0(D)$ for the image under M of D. That means we can let $\Psi(M)$ be the simulation which has object map $D \mapsto M_0(D)$ and components $M_1(D)$. It is clear that this is a simulation; moreover, it is straightforward to check that Ψ is functorial.

In addition, the assignments Φ, Ψ are mutually inverse, so set up a bijection between $\mathfrak{Sim}_*(\mathsf{C})[F, G]$ and $\mathfrak{Rcat}_*/\mathsf{C}[F, \mathsf{Asm}_*(G)]$.

The last thing to check is that this correspondence preserves the enrichment. So suppose that we are given simulations $(K, \alpha), (L, \beta)$ from F to G, and that $(K, \alpha) \preceq (L, \beta)$. Thus there exists, for each object D a map $\lambda_D : KD \longrightarrow LD$ such that $G(\lambda_D)\alpha_D = \beta_D$. Thus λ_D may be viewed as a vertical map in the fibre over D of $\mathsf{Asm}_*(G)$, as in

$$\begin{array}{ccc} FD & \xrightarrow{\alpha_D} & GKD \\ {\scriptstyle =}\downarrow & & \downarrow{\scriptstyle G\lambda_D} \\ FD & \xrightarrow{\beta_D} & GLD. \end{array}$$

Thus the family λ witnesses the existence of a natural transformation $\Phi(K, \alpha) \Rightarrow \Phi(L, \beta)$. It is now straightforward to see that this is locally full and faithful. This completes the proof of Theorem 3.4.

3.3 Universal simulations and lifting problems

There are a few straightforward consequences of Theorem 3.4. The first is the existence, for each $G : \mathsf{E} \longrightarrow \mathsf{C}$, of a *universal simulation*. Indeed, consider the simulation from $\mathsf{Asm}_*(G)$ to G corresponding to the identity on $\mathsf{Asm}_*(G)$. We will denote this simulation by (U_G, υ_G). Theorem 3.4 may now be restated as follows:

Corollary 3.5. *The operation of post-composing with the universal simulation (U_G, υ_G) induces a natural isomorphism of hom-categories*

$$\mathfrak{Rcat}_*/\mathsf{C}[F, \mathsf{Asm}_*(G)] \longrightarrow \mathfrak{Sim}_*[F, G].$$

Indeed, every simulation from F to G factors through (U_G, υ_U).

Note that this is really just the analogue of the universal property of the comma construction. Also for free is:

Corollary 3.6. *For a functor $G : \mathsf{E} \longrightarrow \mathsf{C}$, the following are equivalent:*

(i) *every simulation with codomain G is functorial*

(ii) *the universal simulation (U_G, υ_G) is functorial*

(iii) *the quotient functor $\mathsf{C}/G \longrightarrow \mathsf{Asm}_*(G)$ has a splitting*

Proof. The implication (i) $\Rightarrow$ (ii) is obvious. Conversely, if we are given a simulation we factor it as a functor followed by the universal simulation, and this composite is functorial if the latter is. The equivalence between (i) and (iii) is also straightforward. □

3.4 Equivalences

We conclude this section by giving a sufficient condition under which one can straighten out a simulation (up to isomorphism). By this, we understand that we are given a simulation (K, α) which is not necessarily functorial, and that we find another simulation (K', α') such that (i) $(K, \alpha) \cong (K', \alpha')$ and (ii) (K', α') is functorial. We first discuss strict simulations.

Recall that we have already shown that, if we are interested in categories over the base up to simulation equivalence, we may assume them to have split idempotents (relative to the base category). We begin with an ad-hoc definition:

Definition 3.7. Suppose $F : \mathsf{D} \longrightarrow \mathsf{C}$ and $G : \mathsf{E} \longrightarrow \mathsf{C}$ are categories over C and $(L, \beta) : F \rightsquigarrow G$ is a simulation over C. We say that M *splits pointwise* if the following hold:

(i) for each object $X \in \mathsf{D}$, the component $\beta_X : FX \longrightarrow GLX$ is a section with retraction γ_X.

(ii) The idempotent $\beta_X \gamma_X$ on GLX is in the image of G.

We can now state a technical, but useful condition for a simulation to be functorial up to isomorphism. A special case of this result appeared in [4], where it was shown that a simulation equivalence between Turing categories can be straightened out to a functorial equivalence.

Lemma 3.8. *If F, G are faithful functors, G is Morita closed then any simulation $(L, \beta) : F \rightsquigarrow B$ which splits pointwise is isomorphic to a functorial simulation $(K, \alpha) : F \rightsquigarrow G$ where α is an isomorphism.*

Proof. For each X in D we have a diagram in C:

$$GLX \xrightarrow{\gamma_X} FX \xrightarrow{\beta_X} GLX.$$

The composite $\beta_X \gamma_X$ is an idempotent, call it E_X, which is in the image of G, say $G(e_X) = E_X$. Since G is faithful, e_X must be idempotent as well, and since G is Morita closed, e_X must split, say as the composite

$$LX \xrightarrow{r_X} X' \xrightarrow{m_X} LX.$$

Now FX and GX' both split the idempotent E_X, so must be isomorphic, say via an isomorphism $\alpha_X : FX \longrightarrow GX'$. This suggests defining the functor $K : \mathsf{D} \longrightarrow \mathsf{E}$ as $KX = X'$. This is functorial: given $f : X \longrightarrow Y$, we know, by the fact that (L, β) is a simulation, that there exists a map $r_f : LX \longrightarrow LY$ making

$$\begin{array}{ccc} FX & \xrightarrow{\beta_X} & GLX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gr_f} \\ FY & \xrightarrow{\beta_Y} & GLY \end{array}$$

commute. Using the fact that Y is a retract of LY via r_Y we may form the composite $X' \xrightarrow{m_X} LX \xrightarrow{r_f} LY \xrightarrow{r_Y} Y'$. Thus we may let f' be this composite, and choose $K(f) = f'$. Now factor the above square as

$$\begin{array}{ccccc} FX & \xrightarrow{\alpha_X} & GX' & \xrightarrow{Gm_X} & GLX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf'} & & \downarrow{\scriptstyle Gr_f} \\ FY & \xrightarrow{\alpha_Y} & GY' & \xrightarrow{Gm_Y} & GLY \end{array}$$

The outer square and the right-hand square commute, so the left-hand square commutes as well, since Gm_Y is monic.

Since α is an isomorphism, this implies immediately that K is functorial. Moreover, K preserves the restriction.

Finally, as a simulation, (K, α) is isomorphic to (L, β): the entailment $(K, \alpha) \vdash (L, \beta)$ is realized by m, and the entailment $(L, \beta) \vdash (K, \alpha)$ by r. □

The conditions in this lemma are technical in nature, but there are natural situations in which they hold:

Lemma 3.9. *Let F, G be as before, and let $(L, \beta) : F \rightsquigarrow G$ be a simulation which has a left adjoint. Then (L, β) is isomorphic to a functorial simulation.*

Proof. We show that the conditions of Lemma 3.8 are satisfied.

To begin, we pass to free algebras, as to obtain a diagram

$$\begin{array}{ccc} \mathsf{Asm}_t(F) & \xrightarrow{\mathsf{Asm}_t(L,\beta)} & \mathsf{Asm}_t(G) \\ & {\scriptstyle \delta_F}\searrow \quad \swarrow{\scriptstyle \delta_G} & \\ & \mathsf{C} & \end{array}$$

where $\mathsf{Asm}_t(L, \beta)$ has a left adjoint $\mathsf{Asm}_t(K, \alpha)$. We will simply write K^+ and L^+ for these functors.

We need to prove that each of the maps $\beta_X : FX \longrightarrow GLX$ is the splitting of an idempotent E_X which is in the image of G. Since D is a full subcategory of $\mathsf{Asm}_t(G)$ (after all, we are assuming G to be faithful), it suffices to prove that E_X is in the image of δ_G.

The object X may be viewed as the assembly $(FX \xrightarrow{1} FX)$. Applying L^+, we get an object $L^+(FX \xrightarrow{1} FX) = (FX \xrightarrow{\beta_X} GY)$ for some Y. Applying K^+ to $(GY \xrightarrow{1} GY)$ we get $K^+(GY \xrightarrow{1} GY) = (GY \xrightarrow{\alpha_Y} FX')$ for some X'. Moreover, the canonical cartesian morphisms give rise to

$$K^+L^+(FX \xrightarrow{1} FX) \xrightarrow{K^+\beta_X} K^+(GY \xrightarrow{1} GY) \xrightarrow{\alpha_Y} (FX' \xrightarrow{1} FX').$$

The counit of the lifted adjunction at $(FX \xrightarrow{1} FX)$ is a vertical morphism $K^+L^+(FX \xrightarrow{1} FX) = (FX \to FX') \xrightarrow{\epsilon} (FX \to FX)$, as in

$$\begin{array}{ccc} FX & \longrightarrow & FX' \\ \downarrow & & \vdots \\ FX & \longrightarrow & FX \end{array}$$

where the dotted arrow is $F\epsilon'$.

This gives us a realizer $\epsilon' : X' \to X$, making the following diagram commute:

$$\begin{array}{ccccc} K^+L^+(FX \xrightarrow{1} FX) & \xrightarrow{K^+\beta_X} & K^+(GY \xrightarrow{1} GY) & \xrightarrow{\alpha_Y} & (FX' \xrightarrow{1} FX') \\ \downarrow{\scriptstyle \epsilon} & & & & \downarrow{\scriptstyle F\epsilon'} \\ (FX \xrightarrow{1} FX) & & \xrightarrow{\quad 1 \quad} & & (FX \xrightarrow{1} FX) \end{array}$$

To this diagram we apply the right adjoint L^+ and attach a naturality square for the unit:

$$\begin{array}{ccccc} L^+(FX \xrightarrow{1} FX) & & \xrightarrow{\beta_X} & & (GY \xrightarrow{1} GY) \\ \downarrow{\scriptstyle \eta L^+} & & & & \downarrow{\scriptstyle \eta} \\ L^+K^+L^+(FX \xrightarrow{1} FX) & \xrightarrow{L^+K^+\beta_X} & L^+K^+(GY \xrightarrow{1} GY) & \xrightarrow{L^+\alpha_Y} & L^+(FX' \xrightarrow{1} FX') \\ \downarrow{\scriptstyle L^+\epsilon} & & & & \downarrow{\scriptstyle L^+F\epsilon'} \\ L^+(FX \xrightarrow{1} FX) & & \xrightarrow{\quad 1 \quad} & & L^+(FX \xrightarrow{1} FX) \end{array}$$

The vertical composite on the left is the identity, and hence we have shown that β_X has a retraction. The resulting idempotent is on the object $(GY \xrightarrow{1} GY)$ which is in the subcategory E, as needed. □

This immediately gives the following result.

Theorem 3.10. *Any strict simulation equivalence between Morita-closed faithful functors is isomorphic to a categorical equivalence.*

Proof. In an equivalence, both maps are adjoint on either side; thus applying the previous lemma to each map, we get an equivalence consisting of functorial simulations $(K, \alpha), (L, \beta)$. Because α, β are natural isomorphisms, the fact that $(K, \alpha)(L, \beta) \cong 1$ qua simulations implies that $KL \cong 1$ qua functors. Thus, K, L constitute an equivalence of categories. □

This of course leads to the question what can be said about lax simulation equivalences. We have already observed that such equivalences do not distinguish between a functor F and its downclosure $\downarrow(F)$. Now under a mild assumption on the base category, we may prove that a lax simulation equivalence between Morita-closed, downclosed functors can be straightened out:

Theorem 3.11. *Let* C *satisfy the condition that the pullback functor on restriction idempotents* $e \mapsto \overline{fe}$ *has a left adjoint whenever* f *is monic. Then any lax simulation equivalence between Morita-closed, downclosed faithful functors is isomorphic to a categorical equivalence.*

Proof. Suppose the condition on C is satisfied, and let us agree to denote the left adjoint to the pullback functor along m by $\exists_m$. Let (K, α) be a simulation over C whose components are monic (as is the case for an equivalence). Then if we have a lax simulation diagram

$$\begin{array}{ccc} FX & \xrightarrow{\alpha_X} & GKX \\ {\scriptstyle e}\downarrow & \leq & \downarrow{\scriptstyle Gg} \\ FX & \xrightarrow{\alpha_X} & GKX \end{array}$$

where e is a restriction idempotent, we wish to find a map $h : KX \longrightarrow KX$ such that h strictly simulates e. To this end, first let $g' = G(g)\exists_{\alpha_X}(e)$. Then it follows that $g'\alpha_X = \alpha_X e$; however, since G is downclosed, g' is in the image of G, say $g' = G(h)$. Now h simulates e under α as well, and this simulation is strict. Hence we find that (K, α) is tight, and by Lemma 2.3, it is a strict simulation.

We have shown that under the condition on the base C, a lax equivalence between downclosed functors may be replaced with a strict equivalence. Now the previous theorem applies. □

The condition on the base category in the above theorem is a weak form of regularity: it always holds if the subcategory $\mathsf{Tot}(\mathsf{C})$ on the total maps is a regular category (has stable regular epi-mono factorizations). For more information on regularity in the context of partial map categories, we refer to [3].

4 Application to Turing categories

In this section we explore the precise connection between simulations on the level of (relative) PCAs and on the level of the Turing categories which they generate. This will allow us to prove the result, announced in [4], that the 2-category of Turing categories over a fixed base C is 2-equivalent to the 2-category of relative PCAs in C. (In both these 2-categories, the notion of 1-cell is the appropriate type of simulation.) In order to state the result precisely, we need a notion of cartesian simulation (since Turing categories are cartesian). Fortunately, this is relatively straightforward given the theory developed in the previous section.

We warn the reader that although some of the key ingredients will be recalled in this section, we will freely use results obtained in [4]; as such, the material in this section is not entirely self-contained.

4.1 Cartesian simulations

Let us assume we are working the the 2-category $\mathfrak{Rcat}_l$. We may consider also the 2-category on the cartesian restriction categories, cartesian functors and lax total transformations; denote this by $\mathfrak{CartRcat}_l$. It is readily checked that both the comma category monad and the assembly monad lift to $\mathfrak{CartRcat}_l$; thus we get an induced notion of *cartesian simulation*. We may describe this explicitly as follows. If D and E are cartesian restriction categories and F and G are cartesian functors then a cartesian simulation from F to G consists of a simulation (K, α) as before with the additional property that the product comparison isomorphisms can be simulated in E:

$$\begin{array}{ccc} F(D \times D') & \xrightarrow{\alpha_{D\times D'}} & GK(D \times D') \\ \downarrow{\scriptstyle m^{-1}} & \leq & \vdots\,{\scriptstyle G((m^{-1})_\alpha)} \\ FD \times FD' & \xrightarrow[(\alpha_D \times \alpha_{D'})m]{} & G(KD \times KD') \end{array} \qquad \begin{array}{ccc} FD \times FD' & \xrightarrow{(\alpha_D \times \alpha_{D'})m} & G(KD \times KD') \\ \downarrow{\scriptstyle m} & \leq & \vdots\,{\scriptstyle G((m)_\alpha)} \\ F(D \times D') & \xrightarrow[(\alpha_{D\times D'})m]{} & GK(D \times D') \end{array}$$

where $\langle F(\pi_0), F(\pi_1)\rangle = m^{-1}$, and where m is the canonical coherence isomorphism.

This gives rise to a 2-category of cartesian restriction categories over C, cartesian simulations as 1-cells and refinements as 2-cells. We denote this by $\mathfrak{CartSim}_l(\mathsf{C})$. Similarly, we obtain a strict version $\mathfrak{CartSim}(\mathsf{C})$.

Given a cartesian simulation (K, α) under certain circumstances we can normalize it so that the object map K preserves products. This is useful when the product is syntactically given: this happens when one is dealing, for example, with a type theory. A restriction category has its products *given syntactically* in case the objects of the category are elements of the a free algebra generated by 1 and $\_ \times \_$ and 1 is terminal while $A \times B$ is the partial product of A and B. Clearly every cartesian restriction category is equivalent to an inflated version of itself in which the products are given syntactically.

A simulation $(K, \alpha) : F \rightsquigarrow G$, between cartesian restriction functors, where the product is given syntactically in the domain, is *syntactically given* if

$$\alpha_{X\times Y} = m^{-1}(\alpha_X \times \alpha_Y)m : F(X \times Y) \longrightarrow G(KX \times KY).$$

Clearly we now have:

Lemma 4.1. *If* E *has products given syntactically and* $\mathsf{D} \xrightarrow{F} \mathsf{C}$ *and* $\mathsf{E} \xrightarrow{G} \mathsf{C}$ *are cartesian restriction functors then every cartesian simulation* $(K, \alpha) : F \rightsquigarrow G$ *is equivalent to a syntactically given simulation.*

This means, for example, that in searching for cartesian simulations between models of partial algebraic theories one need only determine the effect of α on the atomic objects. Furthermore, if the theory is presented using primitive operations it then suffices describe the simulation for these in order to specify the whole simulation.

4.2 Simulations of Turing categories

We now consider again the passage from a PCA $\mathbb{A}$ to the associated category $\mathsf{Comp}(\mathbb{A})$. Recall that the objects of $\mathsf{Comp}(\mathbb{A})$ are the finite formal powers of A, and the morphism are the (tuples of) $\mathbb{A}$-computable maps (see the example of PCAs in section 1). In [4], it was explained that categories of the form $\mathsf{Comp}(\mathbb{A})$ are *Turing categories*; for convenience we repeat the definition:

Definition 4.2 (Turing category)**.** A cartesian restriction category is called a *Turing category* if it possesses an object A such that for each pair of objects X, Y there exists a map $\tau_{X,Y}$ which is a universal application, in the sense that given a map $f : Z \times X \longrightarrow Y$ there exists a total map $h : Z \longrightarrow A$ for which the following diagram is commutative:

$$\begin{array}{ccc} A \times X & \xrightarrow{\tau_{X,Y}} & Y \\ {\scriptstyle h\times X}\uparrow & \nearrow_{f} & \\ Z \times X & & \end{array}.$$

It turns out (see loc. cit.) that it is sufficient to have a universal object A (an object of which each object is a retract) which possesses a universal self-application $A \times A \xrightarrow{\bullet} A$. This explains why PCAs give rise to such categories: the combinatory completeness property of a PCA guarantees that this self-application is indeed universal (see Theorem 4.7 in loc. cit.).

The notion of a PCA makes sense in any cartesian restriction category C, and the above construction of a Turing category from a PCA can be carried out in this general setting. It is important to note that the category $\mathsf{Comp}(\mathbb{A})$ is a category which comes equipped with a faithful functor into the base category C.

The fact that $\mathsf{Comp}(\mathbb{A})$ is a Turing category raises the question whether every Turing category over C is, up to suitable equivalence, of the form $\mathsf{Comp}(\mathbb{A})$ for

some PCA $\mathbb{A}$ in the base C. It turns out that one needs one extra ingredient: given a Turing category $F : \mathsf{D} \longrightarrow \mathsf{C}$ (where F is faithful), it may happen that the Turing object[1] A in D has more points (total maps $1 \longrightarrow A$) than FA. Then D cannot be of the form $\mathsf{Comp}(\mathbb{A})$, since such categories by construction satisfy $\mathsf{Tot}(\mathsf{D})(1, A) \cong \mathsf{Tot}(\mathsf{C})(1, FA)$. To overcome this, we make the following definition:

Definition 4.3. A *relative PCA* $(\mathbb{A}, \mathcal{V})$ in a cartesian restriction category C is a PCA $(A, \bullet)$ together with a specified subset $\mathcal{V} \subseteq \mathsf{Tot}(\mathsf{C})(1, A)$ of total elements of A. It is required that $\mathcal{V}$ is closed under the application and contains a choice of combinators $\mathbf{k}, \mathbf{s}$ for A.

This amounts to saying that $\mathcal{V}$ is, in a strong sense, a sub-PCA of the PCA of global sections of A. Every PCA may be seen as a relative PCA by letting $\mathcal{V}$ be the collection of all global sections. Moreover, one may still construct a Turing category $\mathsf{Comp}(\mathbb{A}, \mathcal{V})$ out of a relative PCA (by restricting the computable maps to be those witnessed by some element of $\mathcal{V}$).

In [4], Theorem 4.12, it was shown that the concept of a relative PCA is wide enough to describe Turing categories up to Morita-equivalence:

Proposition 4.4. *Let C be a cartesian category in which idempotents split.*

(i) *Every relative PCA $(\mathbb{A}, \mathcal{V})$ in C gives rise to a Turing category $\mathsf{Comp}(\mathbb{A}, \mathcal{V})$ over C.*

(ii) *If $F : \mathsf{D} \longrightarrow \mathsf{C}$ is a faithful cartesian restriction functor and D is a Turing category, then D is Morita-equivalent to a Turing category of the form $\mathsf{Comp}(\mathbb{A}, \mathcal{V})$ for some relative PCA $(\mathbb{A}, \mathcal{V})$. In fact, one may take $(\mathbb{A}, \mathcal{V})$ to be the image under F of a Turing object in D.*

We will now show how simulations of relative PCAs induce categorical simulations between the associated Turing categories (generalizing our observations in section 2.3). We first define the relevant categories (for more details on simulations of relative PCAs see again [4], section 5.1):

Definition 4.5. The 2-category $\mathfrak{Rel}\mathfrak{PCA}(\mathsf{C})$ has

- **Objects:** relative PCAs $(\mathbb{A}, \mathcal{V})$ in the base category C
- **Morphisms:** strict simulations $\alpha : (\mathbb{A}, \mathcal{V}) \longrightarrow (\mathbb{B}, \mathcal{W})$, i.e. total maps $\alpha : A \longrightarrow B$ such that there exists $u \in \mathcal{W}$ such that $u \bullet \alpha(x) \bullet \alpha(y) = \alpha(x \bullet y)$.
- **2-cells:** refinements $\alpha \preceq \beta$ of simulations; explicitly, we put $\alpha \preceq \beta$ precisely when there exists an element $w \in \mathcal{W}$ with $w \bullet \alpha(x) = \beta(x)$.

The other relevant category is the following:

[1] Strictly speaking this has to be shown to be independent of a choice of Turing object; however, in [4] it was shown that all Turing objects in a given Turing category are equivalent as PCAs, and this forces their global sections to be in bijective correspondence.

Definition 4.6. The 2-category $\mathfrak{Turing}(\mathsf{C})$ has

- **Objects:** Turing categories D equipped with a cartesian restriction functor $F : \mathsf{D} \longrightarrow \mathsf{C}$
- **Morphisms:** strict, total, cartesian simulations
- **2-cells:** refinements of simulations

Now that both categories $\mathfrak{RelPCA}(\mathsf{C})$ and $\mathfrak{Turing}(\mathsf{C})$ are defined, we may state the following functoriality result, which is a straightforward generalization of the observations concerning the absolute case made in section 2.3.

Proposition 4.7. *The assignment* $(\mathbb{A}, \mathcal{V}) \mapsto \mathsf{Comp}(\mathbb{A}, \mathcal{V})$ *is a 2-functor from the 2-category of relative PCAs in* C *to the category of Turing categories over* C.

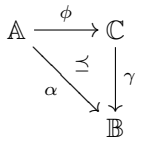
Proof. Let $\alpha : (\mathbb{A}, \mathcal{V}) \longrightarrow (\mathbb{B}, \mathcal{W})$ be a simulation of relative PCAs. We define a simulation $(K, \tilde{\alpha})$ between the associated Turing categories by letting $K(A^n) = B^n$, and by letting the component of $\tilde{\alpha}$ at A^n be α^n. If $f : A^n \longrightarrow A$ is an $(\mathbb{A}, \mathcal{V})$-computable map then since α is a simulation, there is an element $b \in \mathcal{W}$ such that $b \bullet -$ simulates f. This shows that $(K, \tilde{\alpha})$ is a simulation of Turing categories.

Finally, the extension to 2-cells is left as easy exercise. □

Before we state the main result, we need one more result, which characterizes PCAs in categories of assemblies; the idea essentially goes back to Pitts [13]; the connection with the lax slice was described in [8].

Proposition 4.8. *Let* $\mathbb{B}$ *be a (relative) PCA in* C*, and let* $\mathsf{Asm}(\mathbb{B})$ *be the assembly category associated to* $\mathsf{Comp}(\mathbb{B})$*. Then there is a 2-equivalence between the categories* $\mathfrak{RelPCA}(\mathsf{Asm}(\mathbb{B}))$ *of relative PCAs in the category* $\mathsf{Asm}(\mathbb{B})$ *and the lax slice category* $\mathfrak{RelPCA}//\mathbb{B}$.

The objects in this lax slice category are simulations $\mathbb{A} \longrightarrow \mathbb{B}$, and the morphisms are triangles of simulations



Finally, the 2-cells are refinements between the horizontal maps.

Proof. We prove this for absolute PCAs, as the relative case is a matter of some extra bookkeeping.

Suppose first that $\mathbb{A}$ is a PCA in $\mathsf{Asm}(\mathbb{B})$. Thus the underlying object of $\mathbb{A}$ is an assembly of the form $A \xrightarrow{\alpha} B$. (We may assume here that the codomain is indeed B; if not, use the fact that B^n is a retract of B.) The application $A \times A$

$\xrightarrow{\bullet} A$ is simulated by some $v : B \times B \longrightarrow B$, since this is a map of assemblies. Thus we have that v is a $\mathbb{B}$-computable map, and hence that α is a simulation from $\mathbb{A}$ to $\mathbb{B}$ in C.

Next, suppose that there is a simulation $\phi : \mathbb{A} \longrightarrow \mathbb{C}$ of PCAs in $\mathsf{Asm}(\mathbb{B})$, where we now view these PCAs as PCAs over $\mathbb{B}$, say via $\alpha : \mathbb{A} \longrightarrow \mathbb{B}, \gamma : \mathbb{C} \longrightarrow \mathbb{A}$. Now ϕ is in particular a simulation between the underlying PCAs. It does not necessarily commute with the structure maps α, γ, but because it is also a morphism of assemblies there is a witness $v \in \mathbb{B}$ such that $v \bullet \alpha(x) = \beta(\phi(x))$. This says exactly that $\alpha \preceq \beta\phi$, i.e. that we have an induced morphism in the lax slice. For two parallel maps ϕ, ψ with $\phi \preceq \psi$, we evidently get an induced refinement of simulations over $\mathbb{A}$.

It is readily seen that this assignment defines a 2-functor from the category $\mathfrak{RelPCA}(\mathsf{Asm}(\mathbb{B}))$ to $\mathfrak{RelPCA}//\mathbb{B}$. Moreover, an easy verification shows that it is fully faithful and surjective on objects. Finally, it is clear that it also reflects 2-cells, and hence is locally an isomorphism. □

We may now state and prove the main theorem:

Theorem 4.9. *Let* C *be a cartesian category in which idempotents split. The 2-functor* $(\mathbb{A}, \mathcal{V}) \mapsto \mathsf{Comp}(\mathbb{A}, \mathcal{V})$ *is a 2-equivalence of 2-categories*

$$\mathfrak{RelPCA}(\mathsf{C}) \simeq \mathfrak{Turing}(\mathsf{C}).$$

Proof. We first claim that every Turing category $F : \mathsf{D} \longrightarrow \mathsf{C}$ is, up to equivalence, of the form $\mathsf{Comp}(\mathbb{A}, \mathcal{V})$. Indeed, first replace F by its faithful image F_0; then F and F_0 are simulation equivalent. Now Proposition 4.4 tells us that F_0 is Morita-equivalent to a Turing category of the desired form. Since simulation equivalence subsumes Morita-equivalence, this proves the claim.

Next, we claim that the 2-functor Comp is essentially full and faithful. The only non-trivial part is fullness; so let us consider a cartesian simulation $(K, \alpha) : \mathsf{Comp}(\mathbb{A}) \rightsquigarrow \mathsf{Comp}(\mathbb{B})$, where for simplicity we concentrate again on the absolute case. Since products in $\mathsf{Comp}(\mathbb{A})$ are given syntactically, Lemma 4.1 applies, so that, up to isomorphism of simulations, (K, α) is given syntactically. But that means that it is completely determined by the component at the generator. However, this does not give us directly that under this simulation, A gets sent to B.

The simulation (K, α) corresponds to a Kleisli map $\kappa : \mathsf{Comp}(\mathbb{A}) \longrightarrow \mathsf{Asm}(\mathbb{B})$, which is a cartesian functor over the base C. Since cartesian functors preserve PCAs, it follows that $\kappa(\mathbb{A})$ carries a PCA structure in $\mathsf{Asm}(\mathbb{B})$. By Proposition 4.8, this PCA corresponds in turn to a simulation of PCAs $\kappa(\mathbb{A}) \longrightarrow \mathbb{B}$. Since the underlying object of $\kappa(\mathbb{A})$ is simply $\mathbb{A}$, we obtain the desired simulation $\mathbb{A} \longrightarrow \mathbb{B}$ in C.

Finally, the fact that Comp is locally full and faithful is straightforward. □

4.3 Concluding remarks

We end our discussion of simulations by pointing out a few connections to related work.

1. Relation to the $\mathcal{F}$-construction. If we regard a PCA $\mathbb{A}$ as a one-object restriction category where the morphisms are the unary computable functions, then applying the assembly monad to the inclusion functor F into Par gives rise to a category $\mathsf{Asm}_t(F)$. When we consider the subcategory on the total maps, we recover what is traditionally called the category of *partitioned* assemblies over $\mathbb{A}$. Thus the precise connection is now that $\mathcal{F}(F) \cong \mathsf{Tot}(\mathsf{Asm}_t(F))$.

2. We may, instead of the inclusion of the PCA into Par, consider the inclusion of the Turing category $\mathsf{Comp}(\mathbb{A})$ into Par. However, this would, up to equivalence, give the same result. This is essentially due to the fact that the cartesian structure of $\mathsf{Comp}(\mathbb{A})$ is already encoded in the PCA; for a fine analysis of this, see [14].

3. Nomenclature. Since we recover the traditional category of partitioned assemblies, we should explain why we have talked about the assembly monad, instead of the partitioned assembly monad. First of all, the free preorder fibration monad makes sense in many 2-categories; in the case of $\mathfrak{Rcat}$ it will give partitioned assemblies, but in the case of a suitable 2-category of categories of relations it will give (regular) assemblies. Thus it is actually a general construction which will give you exactly the type of assemblies you are looking for once you know what 2-category to start with. Secondly, one may argue that functional simulations (and correspondingly, partitioned assemblies) are a more primitive concept than relational simulations (corresponding to regular assemblies). Indeed, functional simulations may be defined over any cartesian category, while relational ones presuppose regular structure.

4. Lax vs. strict simulations. It is worth noticing that in realizability, one is typically interested in simulations of PCAs which are lax. This is of course a consequence of the fact that one is only interested in the functions which are realizable by the PCA, not which are computable. As a result, one loses control over the exact level of partiality of the PCA. Indeed, under lax simulations, one can have a total PCA equivalent to a strictly non-total one. Also, from a realizability topos one may only recover the PCA up to lax equivalence. In computability, one needs to keep track of the partiality more precisely: the restriction idempotents play the role of recursively enumerable sets. Thus strict simulations of PCAs are appropriate here. This distinction is reflected in the fact that we have defined Turing categories in terms of computable maps, and not of realizable maps.

5. Other approaches to simulations. Although technically and conceptually quite different in nature, we mention that both Hermida (in [7]) and Koslowski [10] have employed 2-categorical methods in order to shed light on relational simulations. Moreover, using dialectica morphisms, Brown, Gurr and de Paiva [2] have studied simulations between Petri nets.

References

[1] Lars Birkedal. *Developing theories of types and computability via realizability*. PhD thesis, Carnegie Mellon University, 1999.

[2] C. Brown, D. Gurr, and V. de Paiva. A categorical linear framework for petri nets. Technical Report 363, University of Aarhus, Denmark, October 1991.

[3] J. R. B. Cockett and X. Guo. Stable meet semilattice fibrations and free restriction categories. *Theory and Applications of Categories*, 16:307–341, 2006.

[4] J. R. B. Cockett and P. J. W. Hofstra. Introduction to Turing categories, 2007. Accepted for publication in Annals of Pure and Applied Logic.

[5] J.R.B Cockett and S. Lack. Restriction categories I. *Theoretical Computer Science*, 270:223–259, 2002.

[6] J.R.B Cockett and S. Lack. Restriction categories II: partial map classification. *Theoretical Computer Science*, 294:61–102, 2003.

[7] Claudio Hermida. A categorical outlook on relational modalities and simulations. Draft version, available electronically at `http://maggie.cs.queensu.ca/chermida/`.

[8] P. Hofstra. Iterated realizability as a comma construction. *Mathematical Proceedings of the Cambridge Philosophical Society*, 144:39–51, 2008.

[9] P. Hofstra and J. van Oosten. Ordered partial combinatory algebras. *Mathematical Proceedings of the Cambridge Philosophical Society*, 134:445–463, 2003.

[10] Jürgen Koslowski. Simulations as a genuinely categorical notion. available electronically at `http://www.iti.cs.tu-bs.de/TI-INFO/koslowj/RESEARCH/`, 2006.

[11] J. Longley. *Realizability toposes and language semantics*. PhD thesis, University of Edinburgh, 1994.

[12] R. Milner. *Communication and concurrency*. International series in computer science. Prentice Hall, 1989.

[13] A. M. Pitts. *The theory of triposes*. PhD thesis, University of Cambridge, 1981.

[14] E. Robinson and G. Rosolini. An abstract look at realizability. In *Computer Science Logic, 15th international workshop, CSL 2001*, Lecture Notes in Computer Science, pages 173–187. Springer Verlag, 2001.

[15] V. Stoltenberg-Hansen and J. Tucker. Computable rings and fields. In E. Griffor, editor, *Handbook of computability theory*, pages 363–447. Elsevier, 1999.

[16] R. Street. Fibrations and Yoneda's lemma in a 2-category. *Lecture Notes in Mathematics*, 420:104–133, 1974.

[17] Jaap van Oosten. *Realizability: an introduction to its categorical side*, volume 152 of *Studies in Logic*. Elsevier, 2008.