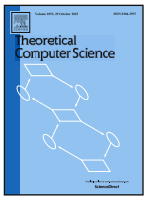
Theoretical Computer Science 1057 (2025) 115550

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs



# The Grothendieck computability model

Luis Gambarte[a,*], Iosif Petrakis[b]

[a] Mathematisches Institut der Universität München, Theresienstr. 39, D-80333 München, Deutschland
[b] Università degli studi di Verona, Strada Le Grazie 15, 37134, Verona, Italia

ARTICLE INFO

Section Editor: Pinyan Lu
Handling Editor: Katie Harris

*Keywords:*
Higher-order computability
Computability models
Category theory
Grothendieck construction

ABSTRACT

Translating notions and results from category theory to the theory of computability models of Longley and Normann, we introduce the Grothendieck computability model. We define the first-projection-simulation and prove its basic properties. With the Grothendieck computability model, the category of computability models is shown to be a type-category, in the sense of Pitts, a result that bridges the categorical interpretation of dependent types with the theory of computability models. We also show that the category of computability models is a category with 2-family arrows and a corresponding structure of Sigma-objects. Finally, we introduce the notion of a fibration and opfibration-simulation, and we prove that the first-projection-simulation is a split opfibration-simulation.

## 1. Introduction

The important role of category theory in computability theory has been emphasised by Cockett and Hofstra in [1–3], who influenced the work of Longley on computability models and simulations between them in [9–11]. The categorical notion of equivalence between computability models studied by Longley and Normann in [12] allowed a better way to "identify" seemingly different computability structures. By associating to a computability model $\mathbf{C}$ its category of assemblies $\mathcal{A}sm(\mathbf{C})$, Longley and Normann established an equivalence of Morita-type between them. We can summarize the work of Longley and Normann by the phrase "from computability models to categories".

In the previous work [13–15] of the second author, the converse direction is followed, i.e., "from categories to computability models". Given a category $\mathcal{C}$ and a copresheaf $S$ on $\mathcal{C}$, i.e., a covariant presheaf on $\mathcal{C}$, the total computability model $\mathbf{CM}^{tot}(\mathcal{C};S)$ is introduced, and if $\mathcal{C}$ is a category with pullbacks and $S$ preserves pullbacks, the partial computability model $\mathbf{CM}^{prt}(\mathcal{C};S)$ is studied. In our work in progress [5] the notion of a computability model over a category $\mathcal{C}$ with a computability base, a notion close to Rosolini's concept of dominion in [19], and a pullback-preserving copresheaf on $\mathcal{C}$, is elaborated. In this way, both constructions, that of $\mathbf{CM}^{tot}(\mathcal{C};S)$ and of $\mathbf{CM}^{prt}(\mathcal{C};S)$, are generalized. Strict computability models are very close to categories of sets and partial functions, but avoiding the equality rules for composition of partial functions (as it is mentioned by Cockett in [1], p. 16, "program equality itself is not well-understood"), they possess a more expressive power than categories. Consequently, simulations, the arrows between computability models, avoid equality, too, involving certain forcing and tracking relations instead.

Working within the direction "from categories to computability models" in this paper, too, we "translate" the categorical Grothendieck construction and the categorical notion of split (op)fibration to the partial and without equality, or relational framework of computability models. The Grothendieck computability models then become the Sigma objects, in the sense of Pitts [18], in the category of computability models. We structure this paper as follows.

* Corresponding author.
*E-mail addresses:* gambarte@math.lmu.de (L. Gambarte), iosif.petrakis@univr.it (I. Petrakis).

https://doi.org/10.1016/j.tcs.2025.115550
Received 27 February 2025; Received in revised form 29 July 2025; Accepted 8 September 2025
Available online 12 September 2025

- In Section 2, we include all basic definitions within the theory of computability models necessary to the rest of this paper. Crucial to the definition of the Grothendieck model is our introduction of the computability model **Sets**, the computability model counterpart to the category of sets and functions (Definition 2). The introduced representable simulations correspond to the representable presheaves (Example 1).
- In Section 3, we define the Grothendieck computability model $\sum_{\mathbf{C}} \boldsymbol{\gamma}$ and the first-projection-simulation $\mathsf{pr}_1 : \sum_{\mathbf{C}} \boldsymbol{\gamma} \twoheadrightarrow \mathbf{C}$ (Proposition 1). We prove basic properties of the Grothendieck computability model, such as the existence of a full, faithful, and essentially surjective functor from the category of simulations $\left[\sum_{\mathbf{C}} \boldsymbol{\gamma}, \mathsf{Sets}\right]$ to the slice category $[\mathbf{C}, \mathsf{Sets}]/\boldsymbol{\gamma}$, where $\boldsymbol{\gamma} : \mathbf{C} \twoheadrightarrow \mathsf{Sets}$ is a simulation (Proposition 3).
- In Section 4, and following [16,17], we define categories with family arrows and Sigma objects, or (fam, Σ)-categories. The type-categories of Pitts in [18] are (fam, Σ)-categories with a terminal object. Every (fam, Σ)-category has dependent arrows (Theorem 1), a new categorical formulation of Martin-Löf's notion of dependent function.
- In Section 5, we show that the category CompMod is a (fam, Σ)-category (Theorem 2), and a type-category (Corollary 1). With these results, the categorical semantics of dependent type theory are connected with the theory of computability models. For example, due to Theorem 1, the category CompMod has dependent arrows (Corollary 2).
- In Section 6, and following [4], we define categories with 2-family arrows and the corresponding Sigma-objects, or (2-fam, Σ)-categories.
- In Section 7, we show that the category CompMod is a (2-fam, Σ)-category (Proposition 4).
- In Section 8, we introduce (split) fibrations and opfibration-simulations, and show that the first-projection-simulation $\mathsf{pr}_1 : \sum_{\mathbf{C}} \boldsymbol{\gamma} \twoheadrightarrow \mathbf{C}$ is a (split) opfibration-simulation (Proposition 5 and Corollary 3).

For all notions and results from category theory that are used here without explanation or proof we refer to [21]. For various examples of computability models and simulations from higher-order computability theory we refer to [12]. This paper is an extension[1] of [6].

## 2. Basic definitions

**Definition 1.** A (strict) *computability model* $\mathbf{C}$ consists of the following data: a class $T$, whose members are called *type names*; for each $t \in T$ a set $\mathbf{C}(t)$ of *data types*; for each $s, t \in T$ a class $\mathbf{C}[s,t]$ of *computable functions*, i.e., partial functions from $\mathbf{C}(s)$ to $\mathbf{C}(t)$. Moreover, for every $r, s, t \in T$ the following hold:

1. The identity $1_{\mathbf{C}(t)}$ is in $\mathbf{C}[t,t]$.
2. For every $f \in \mathbf{C}[r,s]$ and $g \in \mathbf{C}[s,t]$ we have that $g \circ f \in \mathbf{C}[r,t]$.

Next, we describe the computability model of sets and partial functions **Sets**, as the computability model-analogue to the category of sets and functions Sets.

**Definition 2.** The computability model **Sets** has as type names the class of sets and as data types the set $U$ itself, for every type name $U$. If $U, V$ are sets, then a computable function from $U$ to $V$ is a partial function from $U$ to $V$.

A partial arrow $(i,f) : a \rightharpoonup b$ in a category $\mathcal{C}$ consists of a monomorphism $i : dom(i) \to a$ and an arrow $f : dom(i) \to b$ in $\mathcal{C}$. Given a copresheaf $S : \mathcal{C} \to \mathsf{Sets}$, we write $S(i,f)$ instead of $\big(S(i), S(f)\big)$. In [15] computability models over categories and copresheaves on them are studied.

**Definition 3.** Let $\mathcal{C}$ be a category and $S : \mathcal{C} \to \mathsf{Sets}$ a copresheaf on $\mathcal{C}$. The *total computability model* $\mathbf{CM}^{tot}(\mathcal{C}; S)$ over $\mathcal{C}$ and $S$ has as type names the class of objects $\mathcal{C}_0$ of $\mathcal{C}$ and data types the sets $S(c)$, for every $c \in \mathcal{C}_0$. If $c_1, c_2 \in \mathcal{C}_0$, a (total) function from $S(c_1)$ to $S(c_2)$ is an element of the class $\{S(f) \mid f \in \mathtt{Hom}(c_1, c_2)\}$. The *partial computability model* $\mathbf{CM}^{prt}(\mathcal{C}; S)$ over $\mathcal{C}$ and a pullback-preserving copresheaf $S$ on $\mathcal{C}$ has the same type names and data types, while a partial function from $S(c_1)$ to $S(c_2)$ is an element of the class $\{S(i,f) \mid (i,f) : c_1 \rightharpoonup c_2\}$.

The fact that $S$ preserves pullbacks is necessary to prove that $\mathbf{CM}^{prt}(\mathcal{C}; S)$ is a computability model. If $\mathsf{Sets}^{prt}$ is the category of sets and partial functions, then the computability model $\mathbf{CM}^{tot}(\mathsf{Sets}^{prt}, id_{\mathsf{Sets}^{prt}})$ is the computability model **Sets** of Definition 2. Next, we describe the arrows in the category of computability models CompMod. A notion of *contravariant simulation* can also be defined, allowing the contravariant version of the Grothendieck construction for computability models.

**Definition 4.** A *simulation* $\boldsymbol{\gamma}$ from $\mathbf{C}$ (over $T$) to $\mathbf{D}$ (over $U$) consists of a class-function $\gamma : T \to U$ and a relation $\Vdash^{\gamma}_{t} \subseteq \mathbf{D}\big(\gamma(t)\big) \times \mathbf{C}(t)$, for each $t \in T$, a so-called *forcing relation*, subject to the following conditions:

1. For each $x \in \mathbf{C}(t)$ there exists some $y \in \mathbf{D}(\gamma(t))$, such that $y \Vdash^{\gamma}_{t} x$.
2. For each $f \in \mathbf{C}[s,t]$ there exists some $f' \in \mathbf{D}\big[\gamma(s), \gamma(t)\big]$, such that

$$\forall_{x \in \mathbf{C}(s)} \forall_{y \in \mathbf{D}(\gamma(s))} \big(x \in dom(f) \wedge y \Vdash^{\gamma}_{s} x \Rightarrow y \in dom(f') \wedge f'(y) \Vdash^{\gamma}_{t} f(x)\big).$$

In this case, we say $f'$ *tracks* $f$, and we write $f' \Vdash^{\gamma}_{(s,t)} f$.

[1] Sections 4, 6, 7, the proof of Proposition 3, and Corollary 2 are added.

We also write $\boldsymbol{\gamma} : \mathbf{C} \twoheadrightarrow \mathbf{D}$ for a simulation $\boldsymbol{\gamma}$ from $\mathbf{C}$ to $\mathbf{D}$. We call a simulation $\boldsymbol{\gamma} : \mathbf{C} \twoheadrightarrow \mathbf{Sets}$ a *copresheaf-simulation*. The *identity simulation* $\mathbf{1_C} : \mathbf{C} \twoheadrightarrow \mathbf{C}$ is the pair $\big(id_T, (\Vdash_t^{\mathbf{1_C}})_{t\in T}\big)$, where $x' \Vdash_t^{\mathbf{1_C}} x :\Leftrightarrow x' = x$, for every $x', x \in \mathbf{C}(t)$. If $\boldsymbol{\delta} : \mathbf{D} \twoheadrightarrow \mathbf{E}$, the *composite simulation* $\boldsymbol{\delta}\circ\boldsymbol{\gamma} : \mathbf{C} \twoheadrightarrow \mathbf{E}$ is the pair[2] $\big(\delta\circ\gamma, (\Vdash_t^{\boldsymbol{\delta}\circ\boldsymbol{\gamma}})_{t\in T}\big)$, where the relation $\Vdash_t^{\boldsymbol{\delta}\circ\boldsymbol{\gamma}} \subseteq \mathbf{E}\big(\delta(\gamma(t))\big) \times \mathbf{C}(t)$ is defined by

$$z \Vdash_t^{\boldsymbol{\delta}\circ\boldsymbol{\gamma}} x :\Leftrightarrow \exists_{y\in\mathbf{D}(\gamma(t))}\big(z \Vdash_{\gamma(t)}^{\boldsymbol{\delta}} y \ \wedge\ y \Vdash_t^{\boldsymbol{\gamma}} x\big).$$

The following copresheaf-simulations on a computability model $\mathbf{C}$ correspond to the representable functors $\mathtt{Hom}(a, -)$ over $a$ in a category $\mathcal{C}$.

**Example 1.** Let $\mathbf{C}$ be a *locally-small* computability model over $T$, i.e., the class $\mathbf{C}[s,t]$ of computable functions from $\mathbf{C}(s)$ to $\mathbf{C}(t)$ is a set, for every $s, t \in T$. If $t_0 \in T$, the *representable-simulation* $\boldsymbol{\gamma}_{t_0} : \mathbf{C} \twoheadrightarrow \mathbf{Sets}$ consists of the class-function $\gamma_{t_0} : T \to \mathtt{Sets}$, defined by $\gamma_{t_0}(t) := \mathbf{C}[t_0, t]$, for every $t \in T$, and the forcing relations $\Vdash_t^{\gamma_{t_0}} \subseteq \mathbf{C}[t_0, t] \times \mathbf{C}(t)$, defined by

$$f \Vdash_t^{\gamma_{t_0}} x :\Leftrightarrow \exists_{y\in dom(f)}\big(f(y) = x\big).$$

To show that $\boldsymbol{\gamma}_{t_0}$ is a simulation, we suppose that $\mathbf{C}$ is *left-regular* i.e.,

$$\forall_{t\in T}\forall_{x\in\mathbf{C}(t)}\exists_{f\in\mathbf{C}[t_0,t]}\exists_{y\in dom(f)}\big(f(y) = x\big).$$

All computability models that include the constant functions are left-regular (such as Kleene's first model $K_1$ over $T = \{0\}$ with $\mathbf{C}(0) = \mathbb{N}$, and $\mathbf{C}[0,0]$ the Turing-computable partial functions from $\mathbb{N}$ to $\mathbb{N}$). If $f \in \mathbf{C}[s,t]$, it is easy to show that $f^* \Vdash_{(s,t)}^{\gamma_{t_0}} f$, where $f^*$ is the total function from $\mathbf{C}[t_0, s]$ to $\mathbf{C}[t_0, t]$, defined by $f^*(g) := f\circ g$, for every $g \in \mathbf{C}[t_0, s]$. A *right-regularity* condition on a locally-small computability model is needed, in order to define the *contravariant representable-simulations* $\boldsymbol{\delta}_{t_0} : \mathbf{C} \twoheadrightarrow \mathbf{Sets}$, where $\delta_{t_0} : \mathbf{C} \to \mathtt{Sets}$ is defined by $\delta_{t_0}(t) := \mathbf{C}[t, t_0]$, for every $t \in T$.

Next follows the notion of arrow between simulations.

**Definition 5.** If $\boldsymbol{\gamma}, \boldsymbol{\delta} : \mathbf{C} \twoheadrightarrow \mathbf{D}$, then $\boldsymbol{\gamma}$ is *transformable* to $\boldsymbol{\delta}$, in symbols $\boldsymbol{\gamma} \preceq \boldsymbol{\delta}$, if for every $t \in T$ there is $f \in \mathbf{D}[\gamma(t), \delta(t)]$ such that

$$\forall_{x\in\mathbf{C}(t)}\forall_{x'\in\mathbf{D}(\gamma(t))}\big(x' \Vdash_t^{\boldsymbol{\gamma}} x \Rightarrow x' \in dom(f) \ \wedge\ f(x') \Vdash_t^{\boldsymbol{\delta}} x\big).$$

## 3. The Grothendieck computability model

In this section, we introduce the Grothendieck computability model, the computability model counterpart to the category of elements, a special case of the general categorical Grothendieck construction. A category $\mathcal{C}$ is replaced by a computability model $\mathbf{C}$, a copresheaf $S : \mathcal{C} \to \mathtt{Sets}$ is replaced by a copresheaf-simulation $\boldsymbol{\gamma} : \mathbf{C} \twoheadrightarrow \mathbf{Sets}$, and the first-projection functor is replaced by the first-projection-simulation.

**Proposition 1.** *Let $\mathbf{C}$ be a computability model over the class $T$ and let $\boldsymbol{\gamma} : \mathbf{C} \twoheadrightarrow \mathbf{Sets}$. The structure $\sum_{\mathbf{C}}\boldsymbol{\gamma}$ with type names the class*

$$\sum_{t\in T}\boldsymbol{\gamma}(t) := \big\{(t, b) \mid t \in T \text{ and } b \in \gamma(t)\big\},$$

*with data types, for every $(t, b) \in \sum_{t\in T}\boldsymbol{\gamma}(t)$, the sets*

$$\Big(\sum_{\mathbf{C}}\boldsymbol{\gamma}\Big)(t, b) := \big\{y \in \mathbf{C}(t) \mid b \Vdash_t^{\gamma} y\big\},$$

*and computable functions from $\Big(\sum_{\mathbf{C}}\boldsymbol{\gamma}\Big)(s, a)$ to $\Big(\sum_{\mathbf{C}}\boldsymbol{\gamma}\Big)(t, b)$ the classes*

$$\Big\{f \in \mathbf{C}[s,t] \mid \forall_{x\in dom(f)}\Big(x \in \Big(\sum_{\mathbf{C}}\boldsymbol{\gamma}\Big)(s, a) \Rightarrow f(x) \in \Big(\sum_{\mathbf{C}}\boldsymbol{\gamma}\Big)(t, b)\Big)\Big\},$$

*is a computability model. The class-function $\mathsf{pr}_1 : \sum_{t\in T}\boldsymbol{\gamma}(t) \to T$, defined by the rule $(t, b) \mapsto t$, and the forcing relations, defined, for every $(t, b) \in \sum_{t\in T}\boldsymbol{\gamma}(t)$, by*

$$y' \Vdash_{(t,b)}^{\mathsf{pr}_1} y :\Leftrightarrow y' = y,$$

*determine the first-projection-simulation $\mathbf{pr}_1 : \sum_{\mathbf{C}}\boldsymbol{\gamma} \twoheadrightarrow \mathbf{C}$.*

**Proof.** We show that the computable functions include the identities and are closed under composition. Notice that the defining property of the computable functions in the Grothendieck model is equivalent to the condition $a \Vdash_s^{\gamma} x \Rightarrow b \Vdash_t^{\gamma} f(x)$, for every $x \in dom(f)$. If $(t, b) \in \sum_{t\in T}\boldsymbol{\gamma}(t)$, then the identity on $\sum_{\mathbf{C}}\boldsymbol{\gamma}(t, b)$ is the identity on $\mathbf{C}(t)$, i.e., $1_{\mathbf{C}(t)}$ is a computable function from $\Big(\sum_{\mathbf{C}}\boldsymbol{\gamma}\Big)(t, b)$ to itself: if $x \in \mathbf{C}(t)$, then the implication $b \Vdash_t^{\gamma} x \Rightarrow b \Vdash_t^{\gamma} x$ holds trivially. If $g$ is a computable function from $\sum_{\mathbf{C}}\boldsymbol{\gamma}(t, b)$ to $\sum_{\mathbf{C}}\boldsymbol{\gamma}(u, c)$ and $f$ is a computable function from $\sum_{\mathbf{C}}\boldsymbol{\gamma}(s, a)$ to $\sum_{\mathbf{C}}\boldsymbol{\gamma}(t, b)$, then $g\circ f$ is a computable function from $\sum_{\mathbf{C}}\boldsymbol{\gamma}(s, a)$ to $\sum_{\mathbf{C}}\boldsymbol{\gamma}(u, c)$. For that, let $x \in dom(f)$ and $f(x) \in dom(g)$. If $a \Vdash_s^{\gamma} x$, then $b \Vdash_t^{\gamma} f(x)$, and hence $c \Vdash_u^{\gamma} g(f(x))$. To show that $\mathbf{pr}_1$ is a simulation, let $y \in \sum_{\mathbf{C}}\boldsymbol{\gamma}(t, b)$. Then $x \Vdash_{(t,b)}^{\mathsf{pr}_1} x$, and if $f$ is a computable function from $\sum_{\mathbf{C}}\boldsymbol{\gamma}(s, a)$ to $\sum_{\mathbf{C}}\boldsymbol{\gamma}(t, b)$, then $f \Vdash_{((s,a),(t,b))}^{\mathsf{pr}_1} f$. □

[2] Notice that if $\boldsymbol{\delta}\circ\boldsymbol{\gamma} = \boldsymbol{\delta}'\circ\boldsymbol{\gamma}'$, then by the definition of equality on simulations we get $\delta\circ\gamma = \delta'\circ\gamma'$ and $\Vdash_t^{\boldsymbol{\delta}\circ\boldsymbol{\gamma}} = \Vdash_t^{\boldsymbol{\delta}'\circ\boldsymbol{\gamma}'}$, for every $t \in T$. See also [12, §3.4.1]

The following proposition expresses that the Grothendieck construction on a computability model obtained from a category with a copresheaf can be presented as the canonical partial computability model associated with the category of elements. The proof is omitted, since it is straightforward.

**Proposition 2.** *Let $C$ be a category and $S : C \to \mathsf{Sets}$ a pullback-preserving copresheaf on $C$. Let $\gamma^S : C_0 \to \mathbf{Sets}$ be defined via $\gamma^S(c) = S(c)$, let the relations $\Vdash_c^{\gamma^S}$ be the diagonals, and let $\{\mathsf{pr}_2\} : \sum_C S \to \mathsf{Sets}$ be defined by $\{\mathsf{pr}_2\}(c,x) := \{x\}$ and if $f : (c,x) \to (d,y)$ in $\sum_C S$, let $[S(f)](x) := y$. Then*

$$\sum_{\mathbf{CM}^{prt}(C;S)} \boldsymbol{\gamma}^S = \mathbf{CM}^{prt}\Big(\sum_C S; \{\mathsf{pr}_2\}\Big).$$

**Remark 1.** The functor $\mathbf{C} \mapsto \mathcal{A}sm(\mathbf{C})$, studied in [12], does not "preserve" the Grothendieck construction. E.g., if $\mathbf{1}$ is a terminal computability model with type names $\{\emptyset\}$, data type $\mathbf{1}(\emptyset) = \{\emptyset\}$, and as only computable function the identity, then one can define a presheaf $id_{\mathbf{1}} : \mathbf{1} \to \mathbf{Sets}$, and show that

$$\mathcal{A}sm\Big(\sum_{\mathbf{1}} id_{\mathbf{1}}\Big) \neq \sum_{\mathcal{A}sm(\mathbf{1})} \mathcal{A}sm(id_{\mathbf{1}}).$$

Next, we show a result analogous to [8, Proposition 1.1.7], i.e., for any copresheaf-simulation $\boldsymbol{\gamma} : \mathbf{C} \to \mathsf{Sets}$ there is an equivalence

$$\Big[\sum_{\mathbf{C}} \boldsymbol{\gamma}, \mathsf{Sets}\Big] \cong [\mathbf{C}, \mathsf{Sets}]/\boldsymbol{\gamma}.$$

For that, we define a full, faithful and essentially surjective functor

$$I : \Big[\sum_{\mathbf{C}} \boldsymbol{\gamma}, \mathsf{Sets}\Big] \to [\mathbf{C}, \mathsf{Sets}]/\boldsymbol{\gamma}.$$

Notice that for both categories the morphism-structure is thin and thus a preorder, thus we only need to define our functor on morphisms and show that it preserves this preorder. The objects of $[\mathbf{C}, \mathsf{Sets}]/\boldsymbol{\gamma}$ themselves are simulations $\boldsymbol{\delta} : \mathbf{C} \to \mathsf{Sets}$, such that $\boldsymbol{\delta} \preceq \boldsymbol{\gamma}$.

For every $\boldsymbol{\gamma} : \mathbf{C} \to \mathsf{Sets}$ we have that for every $t \in T$ the set $\gamma(t)$ is non-empty. This follows from the fact that, otherwise, $\Vdash_t^{\gamma}$ could not satisfy the first condition on simulations, as $\gamma(t)$ is empty, hence there is no $a \in \gamma(t)$ with $a \Vdash_t^{\gamma} b$ for any $b \in \mathbf{C}(t)$. (There is the possibility that $\mathbf{C}(t)$ is empty, but we shall exclude this from now on).

**Proposition 3.** *Let $\mathbf{C}$ be a computability model over $T$ and let $\boldsymbol{\gamma} : \mathbf{C} \to \mathsf{Sets}$ be a copresheaf-simulation. There is a functor $I : \big[\sum_{\mathbf{C}} \boldsymbol{\gamma}, \mathsf{Sets}\big] \to [\mathbf{C}, \mathsf{Sets}]/\boldsymbol{\gamma}$, defined as follows: if $\boldsymbol{\beta} : \sum_{\mathbf{C}} \boldsymbol{\gamma} \to \mathsf{Sets}$, let $I(\boldsymbol{\beta})$ be the underlying class function $I(\beta) : T \to \mathsf{Sets}$, where, for every $t \in T$, we have that*

$$I(\beta)(t) = \bigcup_{x \in \gamma(t)} \beta(t,x).$$

*The tracking relation $\Vdash_t^{I(\beta)} \subseteq \big(\bigcup_{x\in\gamma(t)} \beta(t,x)\big) \times \mathbf{C}(t)$ is defined as follows:*

$$a \Vdash_t^{I(\beta)} b :\Leftrightarrow \exists_{x\in\gamma(t)}\big(a \Vdash_{(t,x)}^{\beta} b\big).$$

*The functor $I$ is full, faithful, and essentially surjective.*

**Proof.** First we show that $I$ is well-defined, i.e., for every $\boldsymbol{\beta} \in \big[\sum_{\mathbf{C}} \boldsymbol{\gamma}, \mathsf{Sets}\big]$ the rule $I(\boldsymbol{\beta})$ defines a simulation and $I(\boldsymbol{\beta}) \preceq \boldsymbol{\gamma}$. We observe that $I(\beta)(t)$ is well-defined and it is a set, for every $t \in T$. To show the first property of simulations, we observe that for every $t \in T$ there exists $x \in \gamma(t)$, such that $\bigcup_{x\in\gamma(t)} \beta(t,x)$ is non-empty. If $b \in \mathbf{C}(t)$, then, as $\boldsymbol{\gamma}$ is a simulation, there is $x \in \gamma(t)$ with $x \Vdash_t^{\gamma} b$. Since $\boldsymbol{\beta}$ is a simulation, there is $a \in \beta(t,x)$, such that $a \Vdash_{(t,x)}^{\beta} b$. By the definition of $\Vdash_t^{I(\beta)}$ we get $a \Vdash_t^{I(\beta)} b$. To show the second condition on $I(\boldsymbol{\beta})$, let $f \in \mathbf{C}[t,t']$. We find $f'$, such that $f' \Vdash_{(t,t')}^{I(\beta)} f$, i.e.,

$$\forall x \in dom(f) \forall y \in I(\beta)(t) : \big(y \Vdash_t^{I(\beta)} x \Rightarrow y \in dom(f') \wedge f'(y) \Vdash_{t'}^{I(\beta)} f(x)\big).$$

Let such $x$ and $y$ be given. By definition, $y \Vdash_t^{I(\beta)} x$ if and only if there exists $r \in \gamma(t)$, such that $y \Vdash_{(t,r)}^{\beta} x$. This entails that $r \Vdash_t^{\gamma} x$, as $x$ has to be in $\big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)(t,r)$. Since $\boldsymbol{\gamma}$ is a copresheaf-simulation, there is $\tilde{f} : \gamma(t) \to \gamma(t')$, such that $\tilde{f} \Vdash_{(t,t')}^{\gamma} f$, and hence $\tilde{f}(r) \Vdash_{t'}^{\gamma} f(x)$, i.e.,

$$f \in \Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)\Big[(t,r), \big(t', \tilde{f}(r)\big)\Big],$$

as for all $z$, such that $r \Vdash_t^{\gamma} z$, we have $\tilde{f}(r) \Vdash_{t'}^{\gamma} f(z)$ by definition of $\tilde{f}$. Since $\boldsymbol{\beta}$ is a simulation, there is $\hat{f} : \beta(t,r) \to \beta\big(t', \tilde{f}(r)\big)$, such that $\hat{f} \Vdash_{((t,r),(t',\tilde{f}(r)))}^{\beta} f$. Hence, $\hat{f}(y) \Vdash_{t'}^{\beta} f(x)$. We can thus define $f'$ to be on each subset $\beta(t,r)$ of $\bigcup_{r\in\gamma(t)} \beta(t,r)$ equal to $\hat{f}$. It is then immediate to see that $f'$ fulfils the required equality. Next we show that $I(\boldsymbol{\beta}) \preceq \boldsymbol{\gamma}$. For that, we define, for every $t \in T$, a function $g_t : I(\beta)(t) \to \gamma(t)$, such that $y \Vdash_t^{I(\beta)} x$ entails $g_t(y) \Vdash_t^{\gamma} x$. We do this again by defining $g_t$ on each subset $\beta(t,r)$ of $I(\beta)(t) = \bigcup_{r\in\gamma(t)} \beta(t,r)$ separately. On $\beta(t,r)$ we define $g_t(y) = r$. Then by definition, if $y \Vdash_t^{I(\beta)} x$, i.e., $y \Vdash_{(t,r)}^{\beta} x$, then $r \Vdash_t^{\gamma} x$, as $x \in \big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)(t,r)$. Hence the functor $I$ is indeed well-defined. It remains to show the three required properties. If $\boldsymbol{\alpha} \preceq \boldsymbol{\beta}$, then we show that $I(\boldsymbol{\alpha}) \preceq I(\boldsymbol{\beta})$. From

the hypothesis we obtain, for every $(t,r) \in \sum_{\mathbf{C}} T$, a function $g_{(t,r)} : \alpha(t,r) \to \beta(t,r)$, such that, if $y \Vdash^{\alpha}_{(t,r)} x$, then $g_{(t,r)}(y) \Vdash^{\beta}_{(t,r)} x$. Hence, for $I(\beta)(t)$ we can define $g_t$ piecewise on the individual $\beta(t,r)$ by setting $g_t(y) := g_{(t,r)}(y)$, if $y \in \alpha(t,r)$. It is immediate to see that the functions $g_t$ prove that $I(\boldsymbol{\alpha}) \preceq I(\boldsymbol{\beta})$. For the converse direction $I(\boldsymbol{\alpha}) \preceq I(\boldsymbol{\beta}) \Rightarrow \boldsymbol{\alpha} \preceq \boldsymbol{\beta}$, we work as above, i.e., we define $g_{(t,r)}(y) := g_t(y)$, where $g_{(t,r)}, g_t$ have the same roles as above. To show essential surjectivity, that is, every $\boldsymbol{\beta} : \mathbf{C} \to \mathsf{Sets}$ with $\boldsymbol{\beta} \preceq \boldsymbol{\gamma}$ is equivalent to $I(\boldsymbol{\alpha})$, for some $\boldsymbol{\alpha} : \sum_{\mathbf{C}} \boldsymbol{\gamma} \to \mathsf{Sets}$. If $\boldsymbol{\beta}$ is given, then for every $(t,r) \in \sum_{\mathbf{C}} T$ let $\alpha(t,r) := \beta(t)$. Let also $x \Vdash^{\alpha}_{(t,r)} a :\Leftrightarrow x \Vdash^{\beta}_{t} a$. Next we show that $\boldsymbol{\alpha}$ is a simulation. As $\alpha : \sum_{\mathbf{C}} T \to \mathsf{Sets}$ is clearly well-defined and $\Vdash^{\alpha}_{(t,r)}$ as well, it remains to prove the two conditions on simulations. For the first, let $a \in \big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)(t,r)$. We show that there is $x \in \alpha(t,r)$, such that $x \Vdash^{\alpha}_{(t,r)} a$. This follows from the existence of $x \in \beta(t) = \alpha(t,r)$, such that $x \Vdash^{\beta}_{t} a$. The second condition is shown similarly, since $f' \Vdash^{\alpha}_{((t,r),(t',r'))} f$ if and only if $f' \Vdash^{\beta}_{(t,t')} f$. Hence, $\boldsymbol{\alpha}$ is a simulation. Clearly, $I(\boldsymbol{\alpha}) = \boldsymbol{\beta}$, since $\bigcup_{r \in \gamma(t)} \alpha(t,r) = \bigcup_{r \in \gamma(t)} \beta(t) = \beta(t)$ and

$$x \Vdash^{I(\alpha)}_{t} a \Leftrightarrow \exists_r x \Vdash^{\alpha}_{(t,r)} a \Leftrightarrow \exists_r x \Vdash^{\beta}_{t} a \Leftrightarrow x \Vdash^{\beta}_{t} a.$$

Thus, $I$ is essentially surjective, since if $I(\boldsymbol{\alpha}) = \boldsymbol{\beta}$, then $I(\boldsymbol{\alpha}) \sim \boldsymbol{\beta}$. □

## 4. Fam-categories with a Σ-structure

In this section, we present the notion of a *category with a family structure* and Σ-*objects*, introduced in [16] and directly linked to the notion of *type-category*, introduced by Pitts in [18]. The Sigma-objects generalise the Grothendieck construction in a general categorical framework.

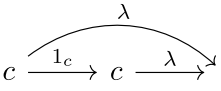
**Definition 6.** A *fam-category* is a category $C$ together with a collection of *family arrows* $\mathtt{fHom}(c)$, for every object $c$ in $C$. We use Greek letters $\lambda, \delta, \ldots$ for them, and we denote them by an arrow starting from $c$

$$c \xrightarrow{\ \lambda\ } .$$

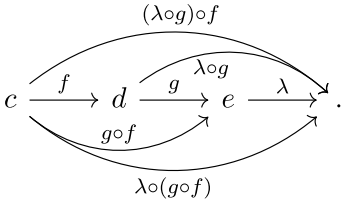
:

For every $c, d \in C$ there is a composition operation $\circ : \mathtt{fHom}(d) \times \mathtt{Hom}(c,d) \to \mathtt{fHom}(c), (\lambda, f) \mapsto \lambda \circ f$, subject to the following conditions:

($F_1$) For every $c \in C$ and $\lambda \in \mathtt{fHom}(c)$ we have $\lambda \circ 1_c = \lambda$.



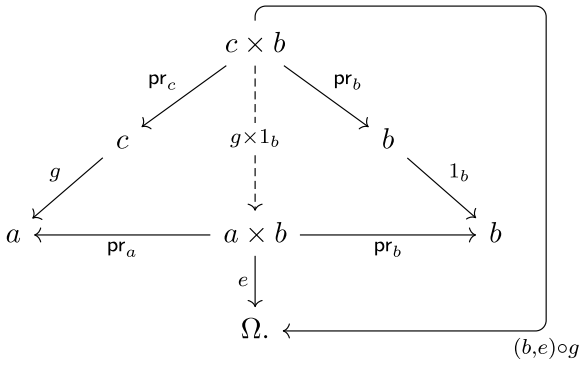

($F_2$) For every $c, d, e \in C$ and $\lambda \in \mathtt{fHom}(e), g \in \mathtt{Hom}(d,e), f \in \mathtt{Hom}(c,d)$ we have that $\lambda \circ (g \circ f) = (\lambda \circ g) \circ f$




**Example 2.** (i) *Constant families*: Every category is turned into a fam-category by setting $\mathtt{fHom}(c) = C_0$, the class of objects, for every $c \in C$. The composition is defined by $c \circ f = c$ for all $c$ and $f$ in $C$.
(ii) *Families in categories*: If **Cat** is the category of locally small categories, let $\mathtt{fHom}(C) = Fun(C^{\mathrm{op}}, \mathsf{Sets})$ with composition the composition of functors.
(iii) (Pitts) *Families in a topos* $C$ with subobject classifier $(\top, \Omega)$: If $a \in C$,

$$\mathtt{fHom}(a) := \bigcup_{b \in C} \mathtt{Hom}(a \times b, \Omega).$$

The composition is defined by the rule $\big((b,e),g\big) \mapsto \big(b, e \times (g \times 1_b)\big)$




(iv) If $A$ is a type in a universe $\mathcal{U}$ (see [23]), a family arrow on $A$ is a type-family $P : A \to \mathcal{U}$.

**Example 3.** It is immediate to show that the category CompMod can be endowed with a family arrow-structure by letting $\mathtt{fHom}(\mathbf{C})$ for each computability model be the class of simulations $\boldsymbol{\gamma} : \mathbf{C} \twoheadrightarrow \mathbf{Sets}$.

**Definition 7.** A fam-category $\mathcal{C}$ has $\Sigma$-*objects*, or is a (fam, $\Sigma$)-*category*, if for every $a, b \in \mathcal{C}$, for every $\lambda \in \mathtt{fHom}(a)$, and for every $f \in \mathtt{Hom}(b, a)$ there is a $\Sigma$-*object* $\sum_a \lambda \in \mathcal{C}$ and arrows $\mathsf{pr}_1^{a,\lambda} \in \mathtt{Hom}\big(\sum_a \lambda, a\big)$ and $\Sigma_\lambda f \in \mathtt{Hom}\big(\sum_b (\lambda \circ f), \sum_a \lambda\big)$, such that the following rectangle is a pullback,

$$\begin{array}{ccc} \sum_b(\lambda \circ f) & \xrightarrow{\sum_\lambda f} & \sum_a \lambda \\ {\scriptstyle \mathsf{pr}_1^{b,\lambda\circ f}}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{pr}_1^{a,\lambda}} \\ b & \xrightarrow[f]{} & a \end{array}$$

and the following conditions hold:

($\Sigma_1$) $\Sigma_\lambda 1_a = 1_{\sum_a \lambda}$,

($\Sigma_2$) $\Sigma_\lambda(f \circ g) = \big(\Sigma_\lambda f\big) \circ \Sigma_{(\lambda \circ f)} g$, for every $f \in \mathtt{Hom}(b, a)$ and $g \in \mathtt{Hom}(c, b)$

$$\begin{array}{ccc} \sum_a(\lambda \circ 1_a & \xrightarrow{\sum_\lambda 1_a} & \sum_a \lambda \\ {\scriptstyle \mathsf{pr}_1^{a,\lambda\circ 1_a}}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{pr}_1^{a,\lambda}} \\ a & \xrightarrow[1_a]{} & a \end{array}$$

$$\begin{array}{ccccc} & & \overset{\sum_\lambda(f\circ g)}{\frown} & & \\ \sum_c(\lambda \circ f) \circ g & \xrightarrow{\sum_{(\lambda\circ f)} g} & \sum_b(\lambda \circ f) & \xrightarrow{\sum_\lambda f} & \sum_a \lambda \\ {\scriptstyle \mathsf{pr}_1^{c,(\lambda\circ f)\circ g}}\downarrow & \lrcorner & {\scriptstyle \mathsf{pr}_1^{b,\lambda\circ f}}\downarrow \quad \lrcorner & & \downarrow{\scriptstyle \mathsf{pr}_1^{a,\lambda}} \\ c & \xrightarrow[g]{} & b & \xrightarrow[f]{} & a. \end{array}$$

A *type-category* (Pitts [18]) is a (fam, $\Sigma$)-category with a terminal object.[3]

**Example 4.** (i) If $\mathcal{C}$ is a category with constant families $\mathtt{fHom}(c) = C_0$ for every $c \in \mathcal{C}$, then let $\sum_c d := c$ and $\mathsf{pr}_1^{c,d} = 1_c$. The square

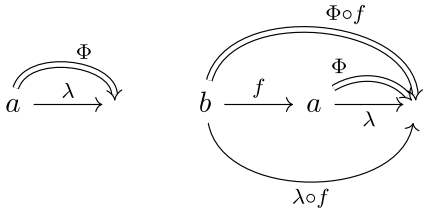
$$\begin{array}{ccc} c & \xrightarrow{f} & d \\ {\scriptstyle 1_c}\downarrow & \lrcorner & \downarrow{\scriptstyle 1_d} \\ c & \xrightarrow{f} & d \end{array}$$

[3] In the definition of a type-category in [18], pp. 110-111, Pitts does not study the family-structure of a type-category separately from its $\Sigma$-structure. In [16] examples of (fam, $\Sigma$)-categories without a terminal object are given.

is a pullback and the two conditions are trivially satisfied.
(ii) In **Cat**, if $P \in \mathtt{fHom}(C) = Fun(C^{\mathrm{op}}, \mathsf{Sets})$, the $\Sigma$-object over $P$ is the category of elements $\sum_C P$.
(iii) For the definition of the canonical $\Sigma$-objects in a topos, see [18], p. 113.
(iv) The $\Sigma$-object in the category of small types $\mathcal{U}$ over a type $A$ in $\mathcal{U}$ and a type family $P : A \to \mathcal{U}$ is the dependent pair-type $\sum_{x:A} P(x)$ (see [23]).

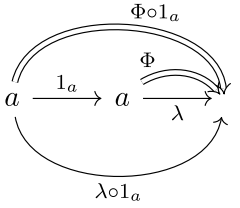
As we show in the end of this section, (fam, $\Sigma$)-categories induce in a canonical way dependent arrows, i.e., objects that are introduced in [16] and generalise categorically Martin-Löf's dependent functions (see [22,23]).

**Definition 8.** A fam-category $C$ has *dependent arrows*, or is a dep-*category*, if to every object $a$ in $C$ and $\lambda \in \mathtt{fHom}(a)$ corresponds a collection $\mathtt{dHom}(a, \lambda)$ of dependent arrows over $(a, \lambda)$. We denote the elements of $\mathtt{dHom}(a, \lambda)$ by capital Greek letters $\Phi, \Psi$, etc., and we use a double-arrow over $a$ and $\lambda$ to picture some $\Phi \in \mathtt{dHom}(a, \lambda)$.

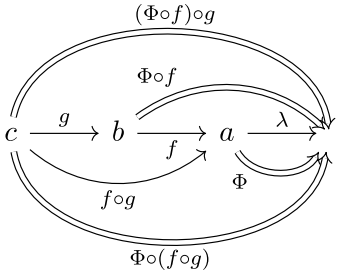



For every $a, b \in C$ and $\lambda : \mathtt{fHom}(a)$ there is a family of composition-functions $\left(\circ_f : \mathtt{dHom}(a, \lambda) \to \mathtt{dHom}(b, \lambda \circ f)\right)_{f \in \mathtt{Hom}(b,a)}$, with $\mathtt{dHom}(a, \lambda) \ni \Phi \mapsto \Phi \circ f \in \mathtt{dHom}(b, \lambda \circ f)$, where for simplicity we write $\circ$ instead of $\circ_f$, such that the following conditions hold:

$$(D_1)\ \Phi \circ 1_a = \Phi,$$




$$(D_2)\ \Phi \circ (f \circ g) = (\Phi \circ f) \circ g.$$




In the next theorem, we employ *global sections* (see [18], p. 114), or *dependent objects* (see [7], pp. 91-92).

**Theorem 1.** *If $C$ is a (fam, $\Sigma$)-category, let for every $a \in C$ and $\lambda \in \mathtt{dHom}(a)$ the class of dependent objects of $\lambda$*

$$\mathcal{D}_a\lambda := \left\{ \phi \in \mathtt{Hom}\left(a, \sum_a \lambda\right) \mid \mathsf{pr}_1^{a,\lambda} \circ \phi = 1_a \right\}$$

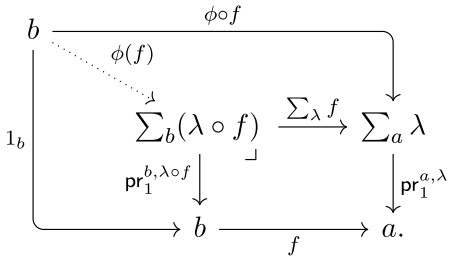
$$\mathcal{D}_a\lambda := \left\{ \phi \in \mathtt{Hom}\left(a, \sum_a \lambda\right) \mid \mathsf{pr}_1^{a,\lambda} \circ \phi = 1_a \right\}$$

$$a \xrightarrow{\phi} \textstyle\sum_a \lambda \xrightarrow{\mathsf{pr}_1^{a,\lambda}} a.$$

$$1_a$$

*If $\mathtt{dHom}(a, \lambda) := \mathcal{D}_a\lambda$, then $C$ becomes a* dep-*category.*

**Proof.** If $\phi \in \mathcal{D}_a\lambda$ and $f \in \mathtt{Hom}(b, a)$, we define a global section $\phi(f) \in \mathcal{D}_b(\lambda \circ f)$ as follows. As $\mathsf{pr}_1^{a,\lambda} \circ (\phi \circ f) = (\mathsf{pr}_1^{a,\lambda} \circ \phi) = 1_a \circ f = f = f \circ 1_b$, the following outer diagram commutes, and let $\phi(f) \in \mathtt{Hom}\left(b, \sum_b (\lambda \circ f)\right)$ be determined by the universal property of a pullback.




For the rest of the proof, see [16]. □

## 5. CompMod is a type-category

**Theorem 2.** *The category* CompMod *is a (fam, Σ)-category.*

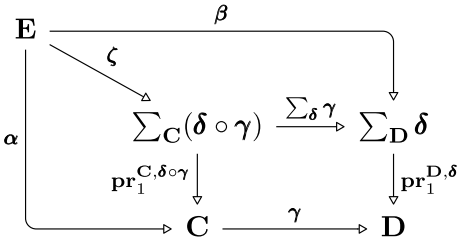
**Proof.** To each simulation $\boldsymbol{\gamma} : \mathbf{C} \to \mathbf{Sets}$ we correspond the Grothendieck model $\sum_{\mathbf{C}} \boldsymbol{\gamma}$ and the first-projection-simulation $\mathsf{pr}_1^{\mathbf{C},\boldsymbol{\gamma}}$. If $\mathbf{D} \in \mathsf{CompMod}$ and $\boldsymbol{\gamma} : \mathbf{C} \to \mathbf{D}$, we define $\sum_{\boldsymbol{\delta}} \boldsymbol{\gamma} : \sum_{\mathbf{C}} \boldsymbol{\delta} \circ \boldsymbol{\gamma} \to \sum_{\mathbf{D}} \boldsymbol{\delta}$.

$$\begin{array}{ccc} \sum_{\mathbf{C}}(\boldsymbol{\delta} \circ \boldsymbol{\gamma}) & \xrightarrow{\sum_{\boldsymbol{\delta}} \boldsymbol{\gamma}} & \sum_{\mathbf{D}} \boldsymbol{\delta} \\ \downarrow{\scriptstyle \mathbf{pr}_1^{\mathbf{C},\boldsymbol{\delta}\circ\boldsymbol{\gamma}}} & & \downarrow{\scriptstyle \mathbf{pr}_1^{\mathbf{D},\boldsymbol{\delta}}} \\ \mathbf{C} & \xrightarrow{\boldsymbol{\gamma}} & \mathbf{D} \end{array}$$

Let $\sum_{\delta} \gamma : \sum_{t \in T} \boldsymbol{\gamma}(t) \to \sum_{u \in U} \boldsymbol{\delta}(u)$ be defined by the rule $(t, b) \mapsto \big(\gamma(t), b\big)$. The forcing relations are defined by $x' \Vdash^{\sum_{\delta}\gamma}_{(t,b)} x :\Leftrightarrow x' \Vdash^{\gamma}_{t} x$. It is straightforward to show that $\sum_{\boldsymbol{\delta}} \boldsymbol{\gamma}$ is a simulation. The above rectangle commutes. For the underlying classes, this is immediate to show, since $\mathsf{pr}_1\big(\sum_{\delta} \gamma(t, b)\big) = \mathsf{pr}_1\big(\gamma(t)\big) = \gamma\big(\mathsf{pr}_1(t, b)\big)$. For the forcing relations, we observe that, if $x' \Vdash^{\mathsf{pr}_1 \circ \sum_{\delta}\gamma}_{(t,b)} x$, then $x' \Vdash^{\sum_{\delta}\gamma}_{(t,b)} x$, and thus $x' \Vdash^{\gamma}_{t} x$, which is equivalent to $x' \Vdash^{\gamma\circ\mathsf{pr}_1}_{(t,b)} x$. To show that it is a pullback, let $\mathbf{E} \in \mathsf{CompMod}$ over a class $V$ and simulations $\boldsymbol{\alpha}, \boldsymbol{\beta}$, such that the following rectangle commutes

$$\begin{array}{ccc} \mathbf{E} & \xrightarrow{\boldsymbol{\beta}} & \sum_{\mathbf{D}} \boldsymbol{\delta} \\ \downarrow{\scriptstyle \boldsymbol{\alpha}} & & \downarrow{\scriptstyle \mathbf{pr}_1^{\mathbf{D},\boldsymbol{\delta}}} \\ \mathbf{C} & \xrightarrow{\boldsymbol{\gamma}} & \mathbf{D}. \end{array} \tag{1}$$

We find unique $\boldsymbol{\zeta} : \mathbf{E} \to \sum_{\mathbf{C}}(\boldsymbol{\delta}\circ\boldsymbol{\gamma})$, such that both triangles in




(2)

commute. First, we define $\zeta$ on the level of the underlying classes. If $v \in V$, let $\zeta(v) = \big(\alpha(v), c\big)$, where $c \in \delta(\gamma(v))$ is the unique $c$ such that $\beta(v) = (u, c)$ for some $u$. Clearly, $\zeta$ is well-defined. We define the forcing relations by

$$x' \Vdash^{\zeta}_{v} x :\Leftrightarrow x' \Vdash^{\alpha}_{v} x.$$

To show that these relations are well-defined, we need to show that if $x' \Vdash^{\alpha}_{v} x$, then $x' \in \big(\sum_{\mathbf{C}}(\boldsymbol{\delta}\circ\boldsymbol{\gamma})\big)(u, c)$, i.e., we have to show that $b \Vdash^{\delta\circ\gamma}_{\alpha(v)} x'$. For this, let such $x'$ be given. As $\boldsymbol{\gamma}$ is a simulation we find $z \in \mathbf{D}\big(\gamma(\alpha(v))\big)$, such that $z \Vdash^{\gamma}_{\alpha(v)} x'$. Hence, $z \Vdash^{\gamma\circ\alpha}_{v} x$, and using the commutativity of (1) we get $z \Vdash^{\mathbf{pr}_1^{\mathbf{D},\boldsymbol{\delta}}\circ\boldsymbol{\beta}}_{v} x$. This yields a $y \in \big(\sum_{\mathbf{D}} \boldsymbol{\delta}\big)(u, c)$ (we recall that $\beta(v) = (u, c)$, as defined above), such that $z \Vdash^{\mathbf{pr}_1^{\mathbf{D},\boldsymbol{\delta}}}_{(u,c)} y \Vdash^{\beta}_{v} x$. By definition of $\mathbf{pr}_1^{\mathbf{D},\boldsymbol{\delta}}$ this yields $y = z$, and thus $z \in \big(\sum_{\mathbf{D}} \boldsymbol{\delta}\big)(u, c)$, which means that $c \Vdash^{\delta}_{u} z$. Gathering everything we see that this yields $b \Vdash^{\delta}_{u} z \Vdash^{\gamma}_{\alpha(v)} x' \Vdash^{\alpha}_{v} x$, and thus $b \Vdash^{\delta\circ\gamma}_{\alpha(v)} x'$, as required. This immediately shows that the first condition on

computability models is fulfilled, and the second follows, as we can take for any $f \in \mathbf{E}[v, v']$ the tracking function $f' \in \mathbf{C}\big[\alpha(v), \alpha(v')\big]$. To show that $f' \in \sum_{\mathbf{C}} \boldsymbol{\delta} \circ \boldsymbol{\gamma}\big[(\alpha(v), c), (\alpha(v'), c')\big]$ we first note that we obtain $f''$ that tracks this $f'$ with respect to $\boldsymbol{\gamma}$. Thus, $f''$ tracks $f$ with respect to $\boldsymbol{\gamma} \circ \boldsymbol{\alpha}$, so it also tracks it with respect to $\mathbf{pr}_1^{\mathbf{D},\boldsymbol{\delta}} \circ \boldsymbol{\beta}$. To see that $f''$ also tracks $f$ with respect to $\boldsymbol{\beta}$, let $x \in \mathbf{E}(v)$ and $z \in \mathbf{D}(u)$ be given, such that $z \Vdash_v^{\beta} x$ and $c \Vdash_u^{\delta} z$. Then $z \Vdash_{(u,c)}^{\mathbf{pr}_1^{\mathbf{D},\boldsymbol{\delta}}} z$ and thus $f''(z) \Vdash_{u'}^{\mathbf{pr}_1^{\mathbf{D},\boldsymbol{\delta}} \circ \beta} f(x)$, and by definition of $\mathbf{pr}_1^{\mathbf{D},\boldsymbol{\delta}}$ also $f''(z) \Vdash_{(u',c')}^{\beta} f(x)$. Thus, $f''$ tracks $f$ with respect to $\beta$.

Using this, let $y \in \big(\sum_{\mathbf{C}} \boldsymbol{\delta} \circ \boldsymbol{\gamma}\big)(\alpha(v), c)$ be given. Thus, we obtain $z \in \mathbf{D}(u)$, such that $z \Vdash_u^{\gamma} y$ and $c \Vdash_{\gamma(u)}^{\delta} z$. Then $f''(z) \Vdash_{u'} f'(y)$, and by the above we get $f''(z) \in \sum_{\mathbf{D}} \boldsymbol{\delta}$, thus $f'(y) \in \sum_{\mathbf{C}} \boldsymbol{\delta} \circ \boldsymbol{\gamma}$. This shows that $f' \in \sum_{\mathbf{C}} \boldsymbol{\delta} \circ \boldsymbol{\gamma}\big[(\alpha(v), c), (\alpha(v'), c')\big]$. Hence, $\boldsymbol{\zeta}$ is a simulation.

It remains to show the commutativity of the triangles in (2). Observe that the two triangles already commute on the level of the underlying class-functions, so it remains to check the forcing relations. Assume that we are given $v \in V$ and $x'' \in \mathbf{E}(v), x' \in \Big(\sum_{\mathbf{D}} \boldsymbol{\delta}\Big)\big(\beta(v)\big)$ and $x \in \mathbf{C}\big(\alpha(v)\big)$, such that

$$x' \Vdash_v^{\beta} x'' \text{ and } x \Vdash_v^{\alpha} x''.$$

By definition we have to show that there exist $y_1, y_2$ such that

$$x' \Vdash_{\zeta(v)}^{\sum_\delta \gamma} y_1 \text{ and } y_1 \Vdash_v^{\zeta} x'', \text{ and } x \Vdash_{\zeta(v)}^{\mathsf{pr}_1} y_2 \text{ and } y_2 \Vdash_v^{\zeta} x''.$$

We know that (1) commutes and $x' \Vdash_{(\sum_\delta \gamma)(\zeta(v))}^{\mathsf{pr}_1} x'$, thus from $x' \Vdash_v^{\beta} x''$ we conclude that $x' \Vdash_v^{\gamma \circ \alpha} x''$. This in turn ensures that there is $y$, such that $x' \Vdash_{\alpha(v)}^{\gamma} y$ and $y \Vdash_v^{\alpha} x''$. By the definition of $\sum_\delta \gamma$ we then have $x' \Vdash_{\alpha(v)}^{\sum_\delta \gamma} y$, and thus $y$ is our desired $y_1$. For $y_2$ we simply choose $x$ and it is easy to see that this satisfies the requirements. The above implications also work in the opposite direction. It is immediate to show that $\boldsymbol{\zeta}$ is the unique simulation that makes the triangles commutative. To show condition $(\Sigma_1)$, we observe that by its definition the simulation $\sum_{\boldsymbol{\epsilon}} \mathbf{1}_{\mathbf{E}}$ on the level of the underlying class takes a pair $(t, u)$ to $(\mathbf{1}_{\mathbf{E}}(t), u) = (t, u)$, hence on the level of the underlying class-functions the two simulations agree. For the forcing relations, we see that both simulations are the corresponding diagonals. To show $(\Sigma_2)$ on the level of underlying classes, we observe that

$$\sum_{\boldsymbol{\epsilon}} (\boldsymbol{\delta} \circ \boldsymbol{\gamma})(t, b) = \big(t, (\delta \circ \gamma)(b)\big) = \sum_{\boldsymbol{\epsilon}} \boldsymbol{\delta}\big(t, \gamma(b)\big) = \sum_{\boldsymbol{\epsilon}} \boldsymbol{\delta}\Big(\Big(\sum_{\boldsymbol{\epsilon} \circ \boldsymbol{\delta}} \boldsymbol{\gamma}\Big)(t, b)\Big).$$

For the forcing relations, we simply remark that $x \Vdash_{(t,b)}^{\sum_\epsilon \delta \circ \gamma} y$ if and only if $x \Vdash_t^{\delta \circ \gamma} y$. Similarly, we have $dash_{(t,b)}^{\sum_\epsilon \delta} y$ if and only if $x \Vdash_t^{\delta} y$, and $x \Vdash_{(t,b)}^{\sum_{\epsilon \circ \delta} \gamma} y$ if and only if $x \Vdash_t^{\gamma} y$. Hence, $x \Vdash_{(t,b)}^{\sum_\epsilon \delta \circ \sum_{\epsilon \circ \delta} \gamma} y$ if and only if $x \Vdash_t^{\delta \circ \gamma} z$, which by the above is equivalent to $x \Vdash_{(t,b)}^{\sum_\epsilon \delta \circ \gamma} z$. □

**Corollary 1.** *The category* CompMod *is a type-category.*

**Proof.** From Theorem 2 and Definition 7 it suffices to show that CompMod has a terminal object. Such an object is described in Remark 1. □

**Corollary 2.** *The category* CompMod *has dependent arrows.*

**Proof.** By Theorems 1 and 2 the canonical dependent arrows over a computability model $\mathbf{C}$ and a presheaf-simulation $\boldsymbol{\gamma} : \mathbf{C} \to \mathsf{Sets}$ are the simulations $\boldsymbol{\phi} : \mathbf{C} \to \sum_{\mathbf{C}} \boldsymbol{\gamma}$, such that $\mathbf{pr}_1^{\mathbf{C},\boldsymbol{\gamma}} \circ \boldsymbol{\phi} = \mathbf{1}_{\mathbf{C}}$. □

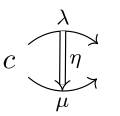
Corollary 2 bridges the theory of computability models with the categorical interpretation of dependent type theory.

## 6. 2-fam-categories with a Σ-structure

In [4] notions and results from [16] are extended to categories with 2-family arrows. The motivation for such a 2-categorical generalisation is the 2-family structure of a universe of types $\mathcal{U}$. If $P, Q : A \to \mathcal{U}$ are type-families over the type $A$ in $\mathcal{U}$, a 2-family arrow from $P$ to $Q$ is a dependent function

$$H : \prod_{x\,:\,A} \big(P(x) \to Q(x)\big).$$

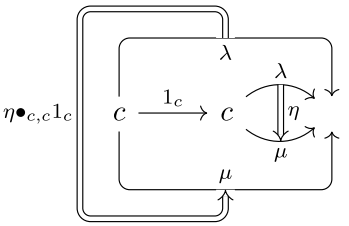
**Definition 9.** A fam-category $C$ is a 2-fam-*category*, if for each $c \in C$ the collection $\mathtt{fHom}(c)$ is a category whose morphisms are called *2-family arrows*. A 2-family arrow $\eta \in \mathtt{Hom}(\lambda, \mu)$ is pictured as follows:



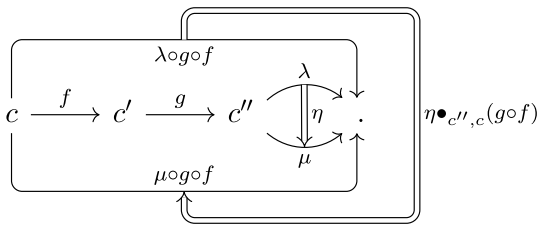

Moreover, for each $c, c' \in C, \lambda, \mu \in \mathtt{fHom}(c')$ there is an operation $\bullet_{c,c'}$ assigning to $\eta \in \mathtt{Hom}(\lambda, \mu)$ and $f : c \to c'$ the 2-family-arrow $\eta \bullet_{c,c'} f \in \mathtt{Hom}(\lambda \circ f, \mu \circ f)$, such that the following conditions hold:

*Compatibility:* For each $c, c', c'' \in C$, each $\lambda, \mu \in \mathtt{fHom}(c'')$, each $\eta \in \mathtt{Hom}(\lambda, \mu)$ and each $f \in \mathtt{Hom}(c, c'), g \in \mathtt{Hom}(c', c'')$ we have that

$$\eta \bullet_{c,c} 1_c = \eta$$

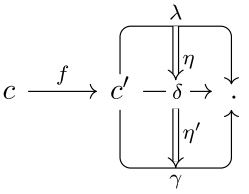



$$\eta \bullet_{c'',c} (g \circ f) = (\eta \bullet_{c'',c'} g) \bullet_{c',c} f$$




*Distributivity:* For each $c, c' \in C$, each $\lambda, \delta, \gamma \in \mathtt{fHom}(c')$ and $\eta \in \mathtt{Hom}(\delta, \gamma), \eta' \in \mathtt{Hom}(\gamma, \lambda)$ as well as $f \in \mathtt{Hom}(c, c')$ we have that

$$(\eta' \circ \eta) \bullet_{c,c'} f = (\eta' \bullet_{c,c'} f) \circ (\eta \bullet_{c,c'} f)$$




**Remark 2.** Usually, we omit the indices in $\bullet_{c,c'}$ and only write $\bullet$. The operation $\bullet$ will be called "horizontal composition" from now on.

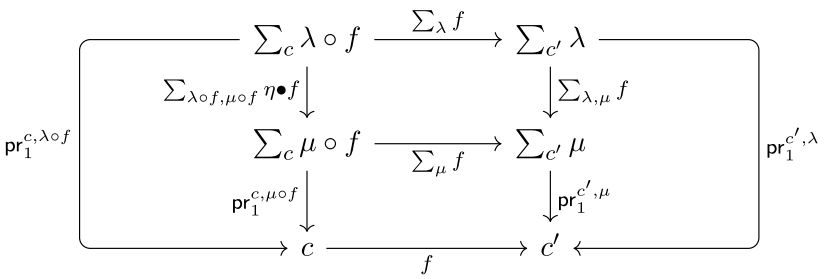
**Example 5.** (i) If $M$ is a monoid, and $C$ is a fam-category, we can define a 2-fam-structure on $C$ by letting $\mathtt{Hom}(\lambda, \mu) := M$ for $\lambda, \mu \in \mathtt{fHom}(c)$ and $c \in C_0$. The compositions $\bullet$ are defined by the rule $m \bullet f := m$. The compatibility and distributivity properties are immediate to show.
(ii) All fam-categories in Example 2 have their 2-analogue (see [4], § 3.1).

Next we define the Σ-structure that corresponds to a 2-fam-category.

**Definition 10.** A 2-fam-category $C$ is a (2-fam, Σ)-category if the underlying fam-category has a (fam,Σ)-structure and for every $c \in C$, every $\lambda, \mu \in \mathtt{fHom}(c)$ and every $\eta \in \mathtt{Hom}(\lambda, \mu)$ there is an arrow

$$\sum_{\lambda,\mu} \eta \colon \sum_c \lambda \to \sum_c \mu,$$

such that the following diagrams commute




Moreover, the following strictness conditions hold:

$$\sum_{\lambda,\lambda} 1_\lambda = 1_{\sum_c \lambda} \quad \text{and} \quad \sum_{\lambda,\mu} \eta \circ \sum_{\nu,\lambda} \eta' = \sum_{\nu,\mu} (\eta \circ \eta').$$

For various examples of (2-fam, Σ)-categories we refer to [4], § 3.2. 2-dep-categories are defined in [4], § 3.4, and the 2-analogue to Theorem 1 is also shown in [4] (Theorem 3.4.6).

## 7. CompMod is a 2-fam-category with a Σ-structure

**Proposition 4.** CompMod *is a 2-fam-category.*

**Proof.** If $\mathbf{C} \in$ CompMod over the class $U$, the category $\mathtt{fHom}(\mathbf{C})$ has objects copresheaf-simulations $\boldsymbol{\gamma} : \mathbf{C} \nrightarrow \mathbf{Sets}$ and arrows *witnesses of transformability*

$$= (\mu_t)_{t\in T} : \boldsymbol{\gamma} \Rightarrow \boldsymbol{\delta},$$

where $\mu_t : \gamma(t) \rightharpoonup \delta(t)$, such that, for every $c \in \mathbf{C}(t)$ and every $d \in \gamma(t)$, the implication $d \Vdash^{\gamma}_{t} c \Rightarrow \mu_t(d) \Vdash^{\delta}_{t} c$ holds. If $: \boldsymbol{\gamma} \Rightarrow \boldsymbol{\delta}$ and $: \boldsymbol{\delta} \Rightarrow \boldsymbol{\lambda}$, then the composition $\circ$ is defined to be $(\nu_t \circ \mu_t)_{t\in T}$. This is well defined because

$$d \Vdash^{\gamma}_{t} c \Rightarrow \mu_t(d) \Vdash^{\delta}_{t} c \Rightarrow \nu_t\big(\mu_t(d)\big) \Vdash_t c.$$

The identity for $\boldsymbol{\gamma} \in \mathtt{fHom}(\mathbf{C})$ is simply $\mathbf{1}_{\mathbf{C}(t)} \in \mathbf{C}[\gamma(t), \gamma(t)]$, which exists because $\mathbf{C}$ is a computability model. Thus $\mathtt{fHom}(\mathbf{C})$ is a category. Next, we define the horizontal composition $\bullet_{\mathbf{C},\mathbf{D}}$, where $\mathbf{C}, \mathbf{D} \in$ CompMod. If $= (\mu_u)_{u\in U}$ and $\boldsymbol{\gamma} : \mathbf{C} \nrightarrow \mathbf{D}$, let $\bullet_{\mathbf{C},\mathbf{D}}\boldsymbol{\gamma} := (\mu_{\gamma(t)})_{t\in T}$. The compatibility conditions are immediate to show. For the distributivity we have that

$$\begin{aligned}(\circ) \bullet \boldsymbol{\lambda} &= (\nu_u \circ \mu_u)_{u\in U} \bullet \boldsymbol{\lambda} = (\nu_{\lambda(t)} \circ \mu_{\lambda(t)})_{t\in T} \\ &= (\nu_{\lambda(t)})_{t\in T} \circ (\mu_{\lambda(t)})_{t\in T} = (\bullet\boldsymbol{\lambda}) \circ (\bullet\boldsymbol{\lambda}).\square\end{aligned}$$

**Definition 11.** An *equality simulation*[4] $\boldsymbol{\gamma} : \mathbf{C} \nrightarrow \mathbf{D}$ is a simulation such that $\mathbf{C}(t) = \mathbf{D}(\gamma(t))$ for all $t \in T$ and the tracking relations $\Vdash^{\gamma}_{t}$ are the equality relations on the respective set $\mathbf{C}(t)$.

**Remark 3.** In [15] it is shown that computability models together with equality simulations form a category. We will use this fact (namely the closure under composition) in the following proof. It is immediate from their definition, that the simulations $\sum_{\boldsymbol{\delta}} \boldsymbol{\gamma}$ and $\sum_{\boldsymbol{\gamma},\boldsymbol{\delta}}$ are equality simulations.

**Theorem 3.** CompMod *is a (2-fam, Σ)-category.*

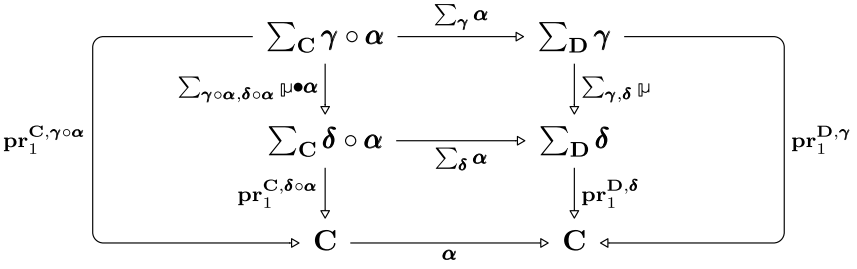
**Proof.** Given $: \boldsymbol{\gamma} \Rightarrow \boldsymbol{\delta}$, where $\boldsymbol{\gamma}, \boldsymbol{\delta} : \mathbf{C} \nrightarrow \mathbf{Sets}$, let $\sum_{\boldsymbol{\gamma},\boldsymbol{\delta}}$ be the simulation with underlying class function $(t, b) \mapsto (t, \mu_t(b))$ and tracking relations

$$\Vdash^{\sum_{\gamma,\delta}}_{(t,b)} = \left\{ (x, x) \;\middle|\; x \in \Big( \sum_{c} \boldsymbol{\lambda} \Big)(t, b) \right\}.$$

This is well-defined due to the following implications:

$$\begin{aligned} x \in \Big( \sum_{\mathbf{C}} \boldsymbol{\gamma} \Big)(t, b) &\Leftrightarrow b \Vdash^{\gamma}_{t} x \Rightarrow \mu_t(b) \Vdash^{\delta}_{t} x \\ &\Rightarrow x \in \Big( \sum_{\mathbf{C}} \boldsymbol{\delta} \Big)\big(t, \mu_t(b)\big) \Rightarrow x \in \Big( \sum_{\mathbf{C}} \boldsymbol{\delta} \Big)\big( \textstyle\sum_{\boldsymbol{\lambda},\boldsymbol{\delta}}(t, b)\big).\end{aligned}$$

The two defining conditions on simulations follow from the definition of the tracking relations. Let a simulation $\boldsymbol{\alpha} : \mathbf{C} \nrightarrow \mathbf{D}$, where $\mathbf{C}$ is over $T$ and $\mathbf{D}$ is over $U$, and simulations $\boldsymbol{\gamma}, \boldsymbol{\delta} : \mathbf{D} \nrightarrow \mathbf{Sets}$ with a witness of transformability $: \boldsymbol{\gamma} \Rightarrow \boldsymbol{\delta}$. We show the commutativity of the following diagram




by showing the following equalities:

$$\begin{aligned}\sum_{\boldsymbol{\gamma},\boldsymbol{\delta}} \circ \sum_{\boldsymbol{\gamma}} \boldsymbol{\alpha} &= \sum_{\boldsymbol{\delta}} \boldsymbol{\alpha} \circ \sum_{\boldsymbol{\gamma}\circ\boldsymbol{\alpha},\boldsymbol{\delta}\circ\boldsymbol{\alpha}} \bullet\boldsymbol{\alpha}, \\ \mathbf{pr}_1 &= \mathbf{pr}_1 \circ \sum_{\boldsymbol{\gamma},\boldsymbol{\delta}}, \\ \mathbf{pr}_1 &= \mathbf{pr}_1 \circ \sum_{\boldsymbol{\gamma}\circ\boldsymbol{\alpha},\boldsymbol{\delta}\circ\boldsymbol{\alpha}} \bullet\boldsymbol{\alpha}.\end{aligned}$$

[4] Equality simulations are elaborated in [15].

On the underlying classes we have for the first equation that

$$\Big(\sum_{\gamma,\delta} \circ \sum_{\gamma} \boldsymbol{\alpha}\Big)(t,b) = \sum_{\gamma,\delta}(\gamma(t),b) = (\alpha(t),\mu_{\alpha(t)}(b))$$
$$= \sum_{\delta} \boldsymbol{\alpha}(t,\mu_{\alpha(t)}(b)) = \Big(\sum_{\delta} \boldsymbol{\alpha} \circ \sum_{\gamma\circ\boldsymbol{\alpha},\delta\circ\boldsymbol{\alpha}} \bullet\boldsymbol{\alpha}\Big)(t,b)$$

and for the second and third equality we have that

$$\mathbf{pr}_1(t,b) = t = \mathbf{pr}_1(t,\mu_{\alpha(t)}(b)) = \mathbf{pr}_1 \circ \Big(\sum_{\gamma,\delta} \bullet\boldsymbol{\alpha}\Big)(t,b),$$

$$\mathbf{pr}_1(u,b) = u = \mathbf{pr}_1(u,\mu_u(b)) = \mathbf{pr}_1 \circ \sum_{\gamma,\delta}(u,b).$$

For the tracking relations we only note that the composition of equality simulations is an equality simulation itself. □

**Remark 4.** In analogy to Corollary 2, and using the fact that a (2-fam, Σ) category has 2-dependent arrows (Theorem 3.4.6 in [4]), one shows that the category CompMod has also 2-dependent arrows.

## 8. Fibration-simulations and opfibration-simulations

The (covariant) Grothendieck construction allows the generation of fibrations (opfibrations), since the first-projection functor $\mathsf{pr}_1 : \sum_C P \to C$ is a (split) opfibration, if $P$ is a copresheaf, and a (split) fibration, if $P$ is a presheaf. In this section we introduce fibration- and opfibration-simulations, and we show that the first-projection-simulation $\mathsf{pr}_{\mathbf{1}} : \sum_{\mathbf{C}} \boldsymbol{\gamma} \to \mathbf{C}$ is a (split) opfibration-simulation, as we work with copresheaf-simulations. The dual result is shown similarly. Throughout this section, $\mathbf{E}$ is a computability model over $T$ and $\mathbf{B}$ is a computability model over $U$. Moreover, the pair

$$\boldsymbol{\varpi} := \Big(\varpi : T \to U,\ \big(\Vdash^{\varpi}_{t}\big)_{t\in T}\Big)$$

is a simulation of type $\mathbf{E} \to \mathbf{B}$.

In contrast to what it holds for functors, for simulations $\boldsymbol{\gamma} : \mathbf{E} \to \mathbf{B}$ each computable function $f$ in $\mathbf{E}$ is tracked, in general, by a multitude of maps $f'$ in $\mathbf{B}$. Thus, for each opspan

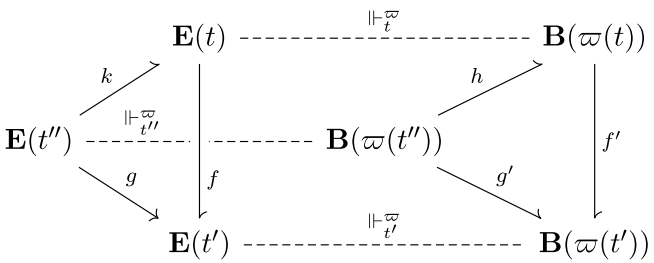
$$\mathbf{E}(t_1) \xrightarrow[g]{} \mathbf{E}(t_2) \xleftarrow[f]{} \mathbf{E}(t_3)$$

we have a whole class, in general, of opspans

$$\mathbf{B}(\gamma(t_1)) \xrightarrow[g']{} \mathbf{B}(\gamma(t_2)) \xleftarrow[f']{} \mathbf{B}(\gamma(t_3))$$

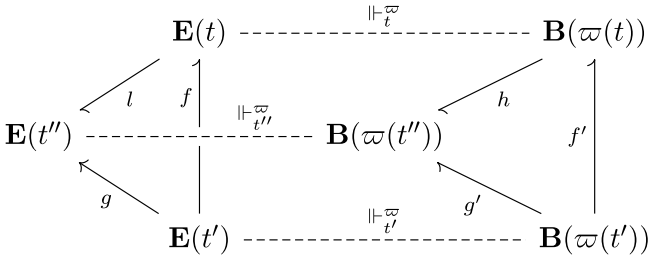
such that $f'$ tracks $f$ and $g'$ tracks $g$.

**Definition 12** (Cartesian computable function)**.** Let $f' \in \mathbf{B}[s,s']$ and $t' \in T$, such that $\varpi(t') = s'$ be given. We call a computable function $f \in \mathbf{E}[t,t']$ *cartesian* for $f'$ and $t'$, if $f' \Vdash^{\varpi}_{(t,t')} f$, and given computable functions $g \in \mathbf{E}[t'',t'], g' \in \mathbf{B}[\varpi(t''),\varpi(t')]$, and $h \in \mathbf{B}[\varpi(t''),\varpi(t)]$ as in the following diagram




that is $g'$ tracks $g$, there is some $k \in \mathbf{E}[t'',t]$ satisfying the following property: $h \Vdash^{\varpi}_{(t'',t)} k$, and for every $x \in \mathbf{E}(t''), y \in \mathbf{B}(\varpi(t''))$, such that $y \Vdash^{\varpi}_{t''} x$, $y \in dom(f'\circ h) \cap dom(g')$, and $f'(h(y)) = g'(y)$, then $x \in dom(f\circ k) \cap dom(g)$ and $g(x) = f(k(x))$.

**Definition 13** (Opcartesian computable function)**.** Let $f' \in \mathbf{B}[s',s]$ and $t' \in T$, such that $\varpi(t') = s'$ be given We call a computable function $f \in \mathbf{E}[t',t]$ *opcartesian* for $f'$ and $t'$, if $f' \Vdash^{\varpi}_{(t',t')} f$, and given computable functions $g \in \mathbf{E}[t',t''], g' \in \mathbf{B}[\varpi(t'),\varpi(t'')]$ and $h \in \mathbf{B}[\varpi(t),\varpi(t'')]$ as in the following diagram



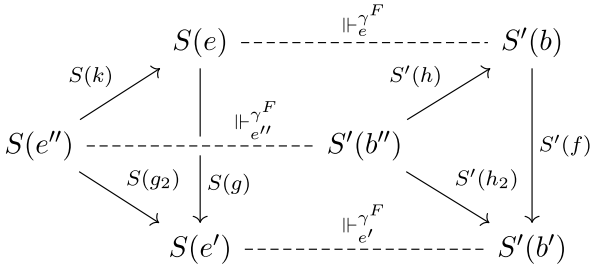

that is $g'$ tracks $g$, there is some $l \in \mathbf{E}[t, t'']$ satisfying the following property: $h$ tracks $l$, and for every $x \in \mathbf{E}(t'), y \in \mathbf{B}(\varpi(t'))$, such that $y \Vdash^{\varpi}_{t'} x$, $y \in dom(h \circ f') \cap dom(g')$, and $f'(h(y)) = g'(y)$, then $x \in dom(l \circ f) \cap dom(g)$ and $g(x) = l(f(x))$.

Note that the computable functions $k \in \mathbf{E}[t'', t]$ and $l \in \mathbf{E}[t, t'']$ in the above two definitions, respectively, are not unique.

**Definition 14** (Fibration-simulation)**.** We call $\boldsymbol{\varpi} : \mathbf{E} \twoheadrightarrow \mathbf{B}$ a *fibration-simula tion*, if for every computable function $f \in \mathbf{B}\big[u, \varpi(t)\big]$ there is $g \in \mathbf{E}[t', t]$ cartesian for $f$ and $t$. In this case, we call $g$ a *lift* of $f$.

**Definition 15** (Opfibration-simulation)**.** We call $\boldsymbol{\varpi} : \mathbf{E} \twoheadrightarrow \mathbf{B}$ an *opfibration-simulation*, if for every computable function $f \in \mathbf{B}\big[\varpi(t), u\big]$, there is $g \in \mathbf{E}[t, t']$ opcartesian for $f$ and $t$. In this case, we call $g$ a *lift* of $f$.

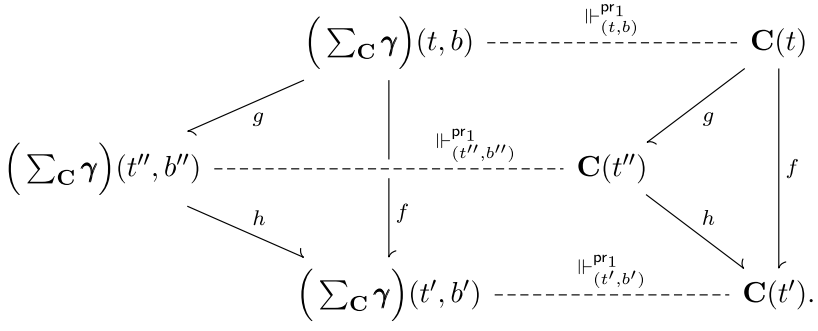
**Example 6.** Let $\mathcal{E}, \mathcal{B}$ be categories with copresheaves $S, S'$ and $F : \mathcal{E} \to \mathcal{B}$ a fibration with $S' \circ F = S$. Then, $\boldsymbol{\gamma}^F : \mathbf{CM}^{tot}(\mathcal{E}; S) \twoheadrightarrow \mathbf{CM}^{tot}(\mathcal{B}; S')$ is a fibration-simulation. To see this, assume we are given a computable function in $\mathbf{CM}^{tot}(\mathcal{B}; S')$, i.e., a function $S'(f) : S'(b) \to S'(b')$, and $e \in \mathcal{E}$ such that $F(e') = b'$. As $F$ is a fibration, we find an arrow $g : e \to e'$ cartesian over $f$ and $b'$ . We show that $S(g)$ is the desired cartesian function over $S'(f)$ and $S(b')$. For this, let functions $S(h), S(h_2), S(g_2)$ as in the following diagram,




be given, where we used that $\mathbf{CM}^{prt}(\mathcal{E}; S)(e) = S(e)$ and $\mathbf{CM}^{prt}(\mathcal{B}; S')(b) = S(b)$, for every $e$ and $b$, respectively. As $g$ is cartesian over $f$ and $b'$, we obtain an arrow $k : e'' \to e$, such that $g \circ k = g_2$ and $F(k) = h_2$. Obviously, $S(k)$ is the function needed, and hence $S(g)$ is cartesian over $S'(f)$ and $S(b')$.

**Proposition 5.** *If* $\mathbf{C} \in \mathsf{CompMod}$ *and* $\gamma : \mathbf{C} \twoheadrightarrow \mathbf{Sets}$ *a copresheaf-simulation, then* $\mathbf{pr}_1 : \sum_{\mathbf{C}} \boldsymbol{\gamma} \twoheadrightarrow \mathbf{C}$ *is an opfibration-simulation.*

**Proof.** Assume we are given a computable function $f \in \mathbf{C}[t, t']$ and $\mathsf{pr}_1(t, b) = t$. We need to find some $b \in \mathbf{C}(t')$, such that $\mathsf{pr}_1(t', b') = t'$, and a computable function $f' \in \Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)\big[(t, b), (t', b')\big]$, such that $f \Vdash^{\mathsf{pr}_1}_{((t,b),(t',b'))} f'$. By definition we know that $f \Vdash^{\mathsf{pr}_1}_{((t,b),(t',b'))} f'$ if and only if $f = f'$, so we have to find $y \in \mathbf{C}(t')$, such that $f(b) = b'$. For this, we simply take $b' := f(b)$. To show that $f$ is opcartesian for $f$ and $b$, we consider the following diagram




Since $h$ fills the triangle on the left, as $f = h \circ g$ whenever these functions are defined, hence, in particular, $f(b) = h\big(g(b)\big)$, and thus $b' = h(b'')$. Hence, $h$ is a computable function from $\Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)(t'', b'')$ to $\Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)(t', b')$. □

Next we define split fibration-simulations and split opfibration-simulations.

**Table 1**
The correspondence between category theory and theory of computablity models.

| Category theory | Theory of computablity models |
|---|---|
| category $\mathcal{C}$ | computability model $\mathbf{C}$ |
| functor $F : \mathcal{C} \to \mathcal{D}$ | simulation $\gamma : \mathbf{C} \twoheadrightarrow \mathbf{D}$ |
| category of Sets | computability model of **Sets** |
| copresheaf $P : \mathcal{C} \to$ Sets | copresheaf-simulation $\gamma : \mathbf{C} \twoheadrightarrow$ **Sets** |
| representable functor $\mathtt{Hom}(a, -)$ | representable simulation $\boldsymbol{\gamma}_{t_0}$ |
| representable functor $\mathtt{Hom}(-, a)$ | representable-simulation $\boldsymbol{\delta}_{t_0}$ |
| Grothendieck category $\sum_{\mathcal{C}} P$ | Grothendieck comp. model $\sum_{\mathbf{C}} \boldsymbol{\gamma}$ |
| first-projection functor | first-projection-simulation |
| $\mathsf{pr}_1 : \sum_{\mathcal{C}} P \to \mathcal{C}$ | $\mathsf{pr}_1 : \sum_{\mathbf{C}} \boldsymbol{\gamma} \to \mathbf{C}$ |
| (op)cartesian arrow | (op)cartesian computable function |
| (op)fibration $\pi : \mathcal{E} \to \mathcal{B}$ | (op)fibration-simulation $\boldsymbol{\varpi} : \mathbf{E} \twoheadrightarrow \mathbf{B}$ |
| split (op)fibration | split (op)fibration-simulation |

**Definition 16.** A *splitting* for a fibration-simulation $\boldsymbol{\varpi} : \mathbf{E} \twoheadrightarrow \mathbf{B}$ is a rule $\varpi^{\triangle}$ that corresponds a pair $(f, u)$, where $f \in \mathbf{B}[t_1, t_2]$ and $\varpi(u) = t_2$, to some $f' \in \mathbf{E}[u, u']$ cartesian for $f$ and $u$, such that the following conditions hold:

($S_1$) For every $f \in \mathbf{B}[t_1, t_2]$ and every $g \in \mathbf{B}[t_2, t_3]$ we have that

$$\varpi^{\triangle}(g \circ f, u_1) = \varpi^{\triangle}(g, u_1) \circ \varpi^{\triangle}(f, u_2).$$

($S_2$) For every $t \in T$ we have that $\varpi^{\triangle}(1_{\mathbf{B}(t)}, u) = (1_{\mathbf{E}(u)}, u)$.

A *splitting* for an opfibration-simulation $\boldsymbol{\varpi} : \mathbf{E} \twoheadrightarrow \mathbf{B}$ is a rule $\varpi^{\triangle}$ that corresponds a pair $(f, u)$, where $f \in \mathbf{B}[t_1, t_2]$ and $\varpi(u) = t_1$, to some $f' \in \mathbf{E}[u, u']$ opcartesian over $f$ and $u$, such that the following conditions hold:

($S_1'$) For every $f \in \mathbf{B}[t_1, t_2]$ and every $g \in \mathbf{B}[t_2, t_3]$ we have that

$$\varpi^{\triangle}(g \circ f, u_1) = \varpi^{\triangle}(g, u_2) \circ \varpi^{\triangle}(f, u_1).$$

($S_2'$) For every $t \in T$ we have that $\varpi^{\triangle}(1_{\mathbf{B}(t)}, u) = (1_{\mathbf{E}(u)}, u)$.

A (op)fibration-simulation $\boldsymbol{\varpi}$ is *split*, if it admits a splitting $\varpi^{\triangle}$.

**Corollary 3.** $\mathbf{pr}_1 : \sum_{\mathbf{C}} \boldsymbol{\gamma} \twoheadrightarrow \mathbf{C}$ *is a split opfibration-simulation.*

**Proof.** Let $\mathbf{pr}_1^{\triangle}$ be defined by the rule $\mathbf{pr}_1^{\triangle}(f, u) := f$. □

## 9. Conclusions and future work

In [12] many concepts and results from category theory are translated into the theory of computability models, where equalities between arrows are replaced by certain relations between type names and (partial) computable functions. Here we extend the work initiated in [14,15] by translating the Grothendieck construction and the notions of fibration and opfibration into the theory of computability models.

Table 1 includes the correspondences between categorical and computability model theory-notions presented here.

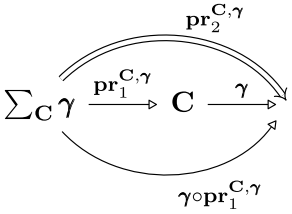
It is natural to ask whether the category of presheaves, or more generally of all functors between two categories, can be translated within computability models. As a consequence, a Yoneda-type embedding and a corresponding Yoneda lemma for computability models and appropriate presheaf-simulations can be formulated. In such a framework, the Grothendieck computability model is expected to have the same crucial role in the proof of a corresponding density theorem with that of the Grothendieck category in the proof of the categorical density theorem. For that, we need to introduce forcing and tracking-moduli in the definition of a simulation i.e., realisers for the forcing and tracking relations. We plan to develop these concepts further in the future.

Our approach to (op)fibration-simulations and (op)cartesian functions is different from the 2-categorical approach to fibrations in [20,24]. In future work, we plan to explain the exact relation between our approach and the 2-categorical one in detail. Namely, our approach to cartesian arrows yields a different notion from the 2-categorical one. Nonetheless, we can prove that by adding the weak assumption that all computability models include all constant functions, every fibration in our sense is a 2-categorical fibration.

The category CompMod is shown to be a (fam, Σ)-category with a terminal object, or a type-category, and also a (2-fam, Σ) category. These results allow the transport of concepts and facts from the theory of categories with family arrows and Sigma-objects into the theory of computability models. For example, through Corollary 2 the category CompMod has dependent arrows. The presence of dependence arrows in a category $\mathcal{C}$ allows the translation of the second-projection associated to the dependent pair-type in Martin-Löf type theory into $\mathcal{C}$ as a dependent arrow (see [16], §5). Consequently, if $\mathbf{C} \in$ CompMod and $\boldsymbol{\gamma} : \mathbf{C} \twoheadrightarrow$ Sets is a copresheaf-simulation, there is a simulation

$$\mathbf{pr}_2^{\mathbf{C},\boldsymbol{\gamma}} : \sum_{\mathbf{C}} \boldsymbol{\gamma} \twoheadrightarrow \sum_{\sum_{\mathbf{C}} \boldsymbol{\gamma}} \left(\boldsymbol{\gamma} \circ \mathbf{pr}_1^{\mathbf{C},\boldsymbol{\gamma}}\right)$$

$$\mathbf{pr}_2^{\mathbf{C},\boldsymbol{\gamma}} : \sum_{\mathbf{C}} \boldsymbol{\gamma} \to \sum_{\sum_{\mathbf{C}} \boldsymbol{\gamma}} \left(\boldsymbol{\gamma} \circ \mathbf{pr}_1^{\mathbf{C},\boldsymbol{\gamma}}\right)$$




representing the second-projection as a dependent arrow over the Grothendieck computability model $\sum_{\mathbf{C}} \boldsymbol{\gamma}$ and the copresheaf-simulation $\boldsymbol{\gamma} \circ \mathbf{pr}_1^{\mathbf{C},\boldsymbol{\gamma}}$.

In [18], Proposition 6.11, it is shown that the classifying category of a dependently typed algebraic theory $Th$ i.e., the category that contains the most general model of this theory, is a type-category. Moreover, a model of $Th$ in any type-category is defined in [18], pp. 117-118. Corollary 1 allows the seemingly unexpected connection between dependently type algebraic theories and the theory of computability models. It is a result that bridges dependent type theory with computability models, where the theory of the latter was introduced by Longley and Normann independently from type-theoretic system with dependent features.[5] In subsequent work we plan to study models of various dependently typed algebraic theories within CompMod. In [18] it is defined when a type-category has dependent products (Definition 6.23). We need to examine whether the type-category CompMod has dependent objects, in the sense of Pitts, and, if yes, to relate them to the canonical dependent arrows of a type-category.

## CRediT authorship contribution statement

**Luis Gambarte:** Writing – review & editing, Writing – original draft, Conceptualization; **Iosif Petrakis:** Writing – review & editing, Writing – original draft.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

[1] R. Cockett, Categories and Computability, manuscript, 2010. https://cs.ioc.ee/ewscs/2010/cockett/estonia-notes.pdf.
[2] R. Cockett, P. Hofstra, Introduction to Turing categories, Ann. Pure Appl. Log. 156 (2-3) (2008) 183–209.
[3] R. Cockett, P. Hofstra, Categorical simulations, J. Pure Appl. Algebra 214 (10) (2010) 1835–1853.
[4] Y. Ehrhardt, 2-dep-Categories, Bachelor's Thesis, LMU Munich, 2024. https://www.math.lmu.de/~petrakis/Ehrhardt.pdf.
[5] L. Gambarte, I. Petrakis, Categories with a Base of Computability, 2025. In preparation.
[6] L. Gambarte, I. Petrakis, The Grothendieck Computability Model, CEUR Workshop Proceedings 3811 (CEUR-WS.org 2024) (2024) 16–28. Proceedings of the 25th Italian Conference on Theoretical Computer Science.
[7] M. Hofmann, T. Streicher, The Groupoid Interpretation of Type Theory, in [22] 83–111.
[8] P. Johnstone, Sketches of an Elephant: A Topos Theory Compendium, Oxford University Press, 2002.
[9] J. Longley, Realizability Toposes and Language Semantics, PhD Thesis ECS-LFCS-95-332, University of Edinburgh, 1995.
[10] J. Longley, On the ubiquity of certain total type structures, Math. Struct. Comput. Sci. 17 (5) (2007) 841–953.
[11] J. Longley, Computability structures, simulations and realizability, Math. Struct. Comput. Sci. 24 (2) (2014) 1–49.
[12] J. Longley, D. Normann, Higher-Order Computability, Springer, (2015) 1–49.
[13] I. Petrakis, Computability Models Over Categories, 2021. https://arxiv.org/abs/2105.06933v1.
[14] I. Petrakis, Computability Models over Categories and Presheaves, Logical Foundations of Computer Science, 13137, Springer, 2022. Available on ArXiV.
[15] I. Petrakis, Strict computability models over categories and presheaves, J. Log. Comput. 077 (2022). https://doi.org/10.1093/logcom/exac077.
[16] I. Petrakis, Categories with Dependent Arrows, 2023. https://arxiv.org/abs/2303.14754v1.
[17] I. Petrakis, Y. Ehrhardt, Categories with Dependent and Codependent Arrows, 2025. Submitted, Preprint, avaliable on arxiv via http://arxiv.org/abs/2303.14754.
[18] A.M. Pitts, Categorical logic, Handbook of Logic in Computer Science Volume 5: Logic and Algebraic Methods, Clarendon Press, Oxford, 2000.
[19] G. Rosolini, Continuity and Effectiveness in Topoi, PhD Thesis, University of Oxford, 1986.
[20] E. Riehl, Two-Sided Discrete Fibrations in 2-Categories and Bicategories, manuscript, 2010. https://math.jhu.edu/~eriehl/fibrations.pdf.
[21] E. Riehl, Category Theory in Context, Dover Publications Inc, 2016.
[22] G. Sambin, J.M. Smith (Eds.), Twenty-Five Years of Constructive Type Theory, Oxford University Press, 1998.
[23] The Univalent Foundations Program, Homotopy Type Theory: Univalent Foundations of Mathematics, Institute for Advanced Study, Princeton, 2013.
[24] L.Z. Wong, The Grothendieck Construction in Enriched, Internal and ∞-Category Theory, PhD Thesis, University of Washington, 2019.

[5] As Longley and Normann remark in [12], p. 544, "there is unexplored territory here, e.g. in combining constructive type theory and set theory with classical approaches to functional algorithms".