The Grothendieck Computability Model

Luis Gambarte[1], Iosif Petrakis[2]

[1]Ludwig-Maximilians-Universität München
[2]Università di Verona

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 corresponding first-projection-simulation, and we prove some of 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 introduce the notion of a fibration and opfibration-simulation, and we show that the first-projection-simulation is a split opfibration-simulation.

Keywords

Computability models, Grothendieck construction, Fibrations

1. Introduction

The important role of category theory in computability theory has been emphasised by Cockett and Hofstra in [1, 2, 3], who influenced the work of Longley on computability models and simulations between them in [7, 8, 9]. The categorical notion of equivalence between computability models that is studied by Longley and Normann in [10] 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 summarise the work of Longley and Normann by the phrase "from computability models to categories".

In the previous work [11, 12, 13] of the second author the converse direction i.e., "from categories to computability models", is followed. Given a category $\mathcal{C}$ and a presheaf S on $\mathcal{C}$, the total computability model $\mathbf{CM}^{\mathrm{tot}}(\mathcal{C};S)$ was introduced, and if $\mathcal{C}$ is a category with pullbacks and S preserves pullbacks, the partial computability model $\mathbf{CM}^{\mathrm{prt}}(\mathcal{C};S)$ was studied. In our joint work in progress [5] the notion of a computability model over a category $\mathcal{C}$ with a base of computability, a notion close to Rosolini's concept of dominion in [16], and a pullback-preserving presheaf on $\mathcal{C}$, is elaborated. In this way, both constructions, that of $\mathbf{CM}^{\mathrm{tot}}(\mathcal{C};S)$ and of $\mathbf{CM}^{\mathrm{prt}}(\mathcal{C};S)$, are generalised. 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

ICTCS'24: Italian Conference on Theoretical Computer Science, September 11–13, 2024, Torino, Italy
gambarte@math.lmu.de (L. Gambarte); iosif.petrakis@univr.it (I. Petrakis)
0000-0002-4121-7455 (I. Petrakis)

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 become then the Sigma-objects, in the sense of Pitts [15], in the category of computability models. We structure this paper as follows:

- 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 $\mathbf{Sets}$, the computability model-counterpart to the category of sets and functions (Definition 2.2). The introduced representable-simulations correspond to the representable presheaves (Example 2.5).
- In section 3 we define the Grothendieck computability model $\sum_{\mathbf{C}} \boldsymbol{\gamma}$ and the corresponding first-projection-simulation $\mathsf{pr}_{\mathbf{1}} \colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \twoheadrightarrow \mathbf{C}$ (Proposition 3.1). 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 $\boldsymbol{\gamma} \colon \mathbf{C} \to \mathsf{Sets}$ is a simulation (Proposition 3.4).
- In section 4 we show that the category of computability models $\mathsf{CompMod}$ is a type-category, in the sense of Pitts [15] (Theorem 4.3). With this result the categorical semantics of dependent type theory are connected with the theory of computability models.
- In section 5 we introduce the notion of a (split) fibration and opfibration-simulation and we show that the first-projection-simulation $\mathsf{pr}_{\mathbf{1}} \colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \twoheadrightarrow \mathbf{C}$ is a (split) opfibration-simulation (Proposition 5.6 and Corollary 5.8).
- In section 6 we include some questions and topics for future work.

For all notions and results from category theory that are used here without explanation or proof we refer to [18]. For various examples of computability models and simulations from higher-order computability theory we refer to [10].

2. Basic definitions

Definition 2.1. 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 $\mathbf{1}_{\mathbf{C}(t)}$ 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 $\mathbf{Sets}$, as the computability model-analogue to the category of sets and functions Sets.

Definition 2.2. The computability model $\mathbf{Sets}$ 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, the computable functions from U to V is the class of partial functions from U to V.

A partial arrow $(i, f)\colon a \rightharpoonup b$ in a category $\mathcal{C}$ consists of a monomorphism $i\colon \mathrm{dom}(i) \to a$ and an arrow $f\colon \mathrm{dom}(i) \to b$ in $\mathcal{C}$. Given a (covariant) presheaf $S\colon \mathcal{C} \to \mathsf{Sets}$, we write $S(i, f)$ instead of $\big(S(i), S(f)\big)$. In [13] computability models over categories and presheaves on them were defined in a canonical way.

Definition 2.3. Let $\mathcal{C}$ be a category and $S\colon \mathcal{C} \to \mathsf{Sets}$ a presheaf on $\mathcal{C}$. The total canonical computability model $\mathbf{CM}^{\mathrm{tot}}(\mathcal{C}; S)$ 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 $c_1, c_2 \in \mathcal{C}_0$, the (total) functions from $S(c_1)$ to $S(c_2)$ is the class $\{S(f) \mid f \in \mathrm{Hom}(c_1, c_2)\}$. The partial canonical computability model $\mathbf{CM}^{\mathrm{prt}}(\mathcal{C}; S)$ over $\mathcal{C}$ and a pullback-preserving presheaf S has the same type names and data types, while the partial functions from $S(c_1)$ to $S(c_2)$ is the class $\{S(i, f) \mid (i, f)\colon c_1 \rightharpoonup c_2\}$.

The pullback-preserving property on S is necessary to prove that $\mathbf{CM}^{\mathrm{prt}}(\mathcal{C}; S)$ is a computability model. We can also use the category $\mathsf{Sets}^{\mathrm{prt}}$ of sets and partial functions, and the computability model $\mathbf{CM}^{\mathrm{tot}}(\mathsf{Sets}^{\mathrm{prt}}, \mathrm{id}_{\mathsf{Sets}^{\mathrm{prt}}})$ is the computability model $\mathbf{Sets}$ of Definition 2.2. Next, we describe the arrows in the category of computability models $\mathsf{CompMod}$. A notion of contravariant simulation can also be defined, allowing the contravariant version of the Grothendieck construction for computability models.

Definition 2.4. A simulation $\boldsymbol{\gamma}$ from $\mathbf{C}$ (over T) to $\mathbf{D}$ (over U) consists of a class-function $\gamma\colon T \to U$ 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

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

In this case we say that f' tracks f, and we write $f' \Vdash^{\gamma}_{(s,t)} f$. We also write $\boldsymbol{\gamma}\colon \mathbf{C} \twoheadrightarrow \mathbf{D}$ for a simulation $\boldsymbol{\gamma}$ from $\mathbf{C}$ to $\mathbf{D}$. We call a simulation $\boldsymbol{\gamma}\colon \mathbf{C} \twoheadrightarrow \mathbf{Sets}$ a (covariant) presheaf-simulation. The identity simulation $\mathbf{1}_{\mathbf{C}}\colon \mathbf{C} \twoheadrightarrow \mathbf{C}$ is the pair $\big(\mathrm{id}_T, (\Vdash^{\iota_{\mathbf{C}}}_{t})_{t \in T}\big)$, where $x' \Vdash^{\iota_{\mathbf{C}}}_{t} x :\Leftrightarrow x' = x$, for every $x', x \in \mathbf{C}(t)$. If $\boldsymbol{\delta}\colon \mathbf{D} \twoheadrightarrow \mathbf{E}$, the composite simulation $\boldsymbol{\delta} \circ \boldsymbol{\gamma}\colon \mathbf{C} \twoheadrightarrow \mathbf{E}$ is the pair $\big(\delta \circ \gamma, (\Vdash^{\delta \circ \gamma}_{t})_{t \in T}\big)$, where the relation $\Vdash^{\delta \circ \gamma}_{t} \subseteq \mathbf{E}\big(\delta(\gamma(t))\big) \times \mathbf{C}(t)$ is defined by

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

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

Example 2.5. 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 $\boldsymbol{\gamma}_{t_0}\colon \mathbf{C} \twoheadrightarrow \mathbf{Sets}$ consists of the class-function $\gamma_{t_0}\colon T \to \mathsf{Sets}$, defined by $\gamma_{t_0}(t) := \mathbf{C}[t_0, t]$, for every $t \in T$, and the forcing relations $\Vdash^{\gamma_{t_0}}_{t} \subseteq \mathbf{C}[t_0, t] \times \mathbf{C}(t)$, defined by

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

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

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

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 $f^*\Vdash^{\gamma_{t_0}}_{(s,t)} f$, 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, to define the contravariant representable-simulations $\delta_{t_0}\colon\mathbf{C}\twoheadrightarrow\mathbf{Sets}$, where $\delta_{t_0}\colon\mathbf{C}\to\mathsf{Sets}$ is defined by $\delta_{t_0}(t):=\mathbf{C}[t,t_0]$, for every $t\in T$.

Next follows the notion of an arrow between simulations.

Definition 2.6. If $\gamma,\delta\colon\mathbf{C}\twoheadrightarrow\mathbf{D}$, then γ is *transformable* to δ, in symbols $\gamma\preceq\delta$, if for every $t\in T$ there is $f\in\mathbf{D}[\gamma(t),\delta(t)]$ such that

$$\forall_{x\in\mathbf{C}(t)}\forall_{x'\in\mathbf{D}(\gamma(t))}\big(x'\Vdash^{\gamma}_{t}x\Rightarrow x'\in\mathrm{dom}(f)\ \&\ f(x')\Vdash^{\delta}_{t}x\big).$$

3. The Grothendieck computability model

The Grothendieck computability model is the categorical 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 $\mathbf{C}$, and a (covariant) presheaf $S\colon\mathcal{C}\to\mathsf{Sets}$ by a (covariant) simulation $\boldsymbol{\gamma}\colon\mathbf{C}\twoheadrightarrow\mathbf{Sets}$. Moreover, the first-projection functor is replaced by the first-projection-simulation.

Proposition 3.1. *Let $\mathbf{C}$ be a computability model over the class T together with a simulation $\boldsymbol{\gamma}\colon\mathbf{C}\twoheadrightarrow\mathbf{Sets}$. 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

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

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

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

is a computability model. The class-function $\mathsf{pr}_1\colon\sum_{t\in T}\boldsymbol{\gamma}(t)\to T$, 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

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

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

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 $x \in \mathrm{dom}(f)$. 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., $\mathbf{1}_{\mathbf{C}(t)}$ 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 $x \in \mathrm{dom}(f)$ and $f(x) \in \mathrm{dom}(g)$. If $a \Vdash_s^\gamma x$, then $b \Vdash_t^\gamma f(x)$, and hence $c \Vdash_u^\gamma g(f(x))$. Next, we show that $\mathbf{pr}_1$ is a simulation. If $y \in \sum_{\mathbf{C}} \boldsymbol{\gamma}(t,b)$, then $x \Vdash^{\mathsf{pr}_1}_{(t,b)} x$, and if f is a computable function from $\sum_{\mathbf{C}} \boldsymbol{\gamma}(s,a)$ to $\sum_{\mathbf{C}} \boldsymbol{\gamma}(t,b)$, then $f \Vdash^{\mathsf{pr}_1}_{((s,a),(t,b))} f$. □

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

Proposition 3.2. *Let $\mathcal{C}$ be a category and $S\colon \mathcal{C} \to \mathsf{Sets}$ a pullback-preserving presheaf on $\mathcal{C}$. Let $\gamma^S\colon \mathcal{C}_0 \to\!\!\!\!\triangleright \mathbf{Sets}$ be defined via $\gamma^S(c) = S(c)$ and the relations $\Vdash_c^{\gamma^S}$ are simply the diagonal, and let $\{\mathsf{pr}_2\}\colon \sum_{\mathcal{C}} S \to \mathbf{Sets}$ be defined by $\{\mathsf{pr}_2\}(c,x) := \{x\}$ and if $f\colon (c,x) \to (d,y)$ in $\sum_{\mathcal{C}} S$, let $[S(f)](x) := y$. Then*

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

Remark 3.3. The functor $\mathbf{C} \mapsto \mathcal{A}sm(\mathbf{C})$, studied in [10], does not "preserve" the Grothendieck construction. Namely, 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 $\mathrm{id}_{\mathbf{1}}\colon \mathbf{1} \to\!\!\!\!\triangleright \mathbf{Sets}$, and show that

$$\mathcal{A}sm\left(\sum_{\mathbf{1}} \mathrm{id}_{\mathbf{1}}\right) \neq \sum_{\mathcal{A}sm\,(\mathbf{1})} \mathcal{A}sm(\mathrm{id}_{\mathbf{1}}).$$

Next we show an analogous result to [6, Proposition 1.1.7]. Our goal is to show that for any presheaf simulation $\boldsymbol{\gamma}\colon \mathbf{C} \to\!\!\!\!\triangleright \mathbf{Sets}$ we obtain an equivalence

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

We do this by exhibiting a full, faithful and essentially surjective functor

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

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. More

explicitly the objects of $[\mathbf{C}, \mathbf{Sets}]/\boldsymbol{\gamma}$ themselves are simply simulations $\boldsymbol{\delta} \colon \mathbf{C} \nrightarrow \mathbf{Sets}$ such that $\boldsymbol{\delta} \preceq \boldsymbol{\gamma}$. For all simulations $\boldsymbol{\gamma} \colon \mathbf{C} \nrightarrow \mathbf{Sets}$ we have that for all $t \in T$ the set $\gamma(t)$ is nonempty. This stems from the fact that otherwise $\Vdash_t^\gamma$ could not fulfil the first condition on simulations as $\gamma(t)$ is empty so 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). Due to space restrictions the proof of the following proposition is omitted.

Proposition 3.4. *Let $\mathbf{C}$ over T be a computability model and $\boldsymbol{\gamma} \colon \mathbf{C} \nrightarrow \mathbf{Sets}$ be a simulation. There is a functor $I \colon \left[\sum_{\mathbf{C}} \boldsymbol{\gamma}, \mathbf{Sets}\right] \to [\mathbf{C}, \mathbf{Sets}]/\boldsymbol{\gamma}$ defined as follows: if $\boldsymbol{\beta} \colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \nrightarrow \mathbf{Sets}$, let $I(\boldsymbol{\beta})$ be the underlying class function $I(\beta) \colon T \to \mathsf{Sets}$, where, for every $t \in T$,*

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

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.

4. Computability models form a type-category

In this section we show that the category of computability models $\mathsf{CompMod}$ is a type-category, in the sense of Pitts [15], pp. 110-111, a reformulation of Cartmell's categories with attributes in [4]. Type-categories are what we call (fam, Σ)-categories with a terminal object in [14], and serve as categorical models of dependent type systems. First, we lift a simulation $\boldsymbol{\gamma} \colon \mathbf{C} \nrightarrow \mathbf{D}$ to a simulation between the Grothendieck computability models $\sum_{\mathbf{C}}(\boldsymbol{\delta} \circ \boldsymbol{\gamma})$ and $\sum_{\mathbf{D}} \boldsymbol{\delta}$.

Lemma 4.1. *Let $\mathbf{C}, \mathbf{D}$ be computability models over the classes T, U respectively, and*

$$\boldsymbol{\gamma} \colon \mathbf{C} \nrightarrow \mathbf{D}, \boldsymbol{\delta} \colon \mathbf{D} \nrightarrow \mathbf{Sets}$$

simulations. There is a simulation $\sum_{\boldsymbol{\delta}} \boldsymbol{\gamma} \colon \sum_{\mathbf{C}}(\boldsymbol{\delta} \circ \boldsymbol{\gamma}) \nrightarrow \sum_{\mathbf{D}} \boldsymbol{\delta}$, such that the following is a pullback square

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

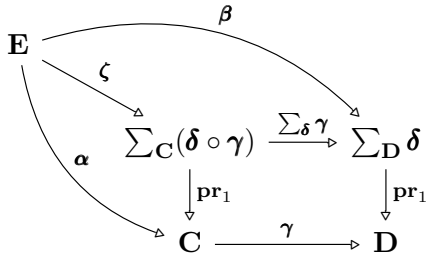
Proof. To define $\sum_{\boldsymbol{\delta}} \boldsymbol{\gamma}$, let the underlying class-function $\sum_{\delta} \gamma \colon \sum_{t \in T} \boldsymbol{\gamma}(t) \to \sum_{u \in U} \boldsymbol{\delta}(u)$ be defined by the rule $(t, b) \mapsto \big(\gamma(t), b\big)$. The corresponding forcing relations are defined by $x' \Vdash_{(t,b)}^{\sum_\delta \gamma} x :\Leftrightarrow x' \Vdash_t^\gamma x$. It is straightforward to show that $\sum_{\boldsymbol{\delta}} \boldsymbol{\gamma}$ is a simulation. Next we show that the above square commutes. On the underlying classes this is immediate as

$$\mathsf{pr}_1\big(\sum_{\delta} \gamma(t, b)\big) = \mathsf{pr}_1\big(\gamma(t)\big) = \gamma\big(\mathsf{pr}_1(t, b)\big).$$

On 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 also equivalent to $x' \Vdash^{\gamma \circ \mathsf{pr}_1}_{(t,b)} x$. Finally, we show the pullback property. Let a computability model $\mathbf{E}$ over a class V with simulations $\boldsymbol{\alpha}, \boldsymbol{\beta}$ be given, such that the following rectangle commutes

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

We find a unique simulation $\boldsymbol{\zeta} \colon \mathbf{E} \twoheadrightarrow \sum_{\mathbf{C}} (\boldsymbol{\delta} \circ \boldsymbol{\gamma})$ such that both following triangles



commute. First we define ζ 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, ζ is well-defined. Next we define the forcing relations. Let

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

This relations are well-defined and in conjunction with the aforementioned class-function they constitute a simulation. Observe that the two triangles already commute on the level of the underlying class-functions, so it remains to check the forcing relations. Assume 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

$$x' \Vdash^{\beta}_{v} x'' \text{ and } x \Vdash^{\alpha}_{v} x''.$$

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

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

We know that the square (1) commutes and $x' \Vdash^{\mathsf{pr}_1}_{(\sum_\delta \gamma)(\zeta(v))} x'$, thus from $x' \Vdash^{\beta}_{v} x''$ we conclude that $x' \Vdash^{\gamma \circ \alpha}_{v} x''$. This in turn ensures that there is y such that $x' \Vdash^{\gamma}_{\alpha(v)} y$ and $y \Vdash^{\alpha}_{v} x''$. By definition of $\sum_\delta \gamma$ we then have that $x' \Vdash^{\sum_\delta \gamma}_{\alpha(v)} y$ and thus y is our desired y_1. For y_2 we simply choose x and it is easy to see that this fulfills the requirements. The above implications also work in the reverse direction. It is immediate to show that $\boldsymbol{\zeta}$ is the unique simulation making the triangles commutative, as it is determined by the definition of $\boldsymbol{\beta}, \boldsymbol{\alpha}$. □

Lemma 4.2. *If* $\mathbf{C}, \mathbf{D}, \mathbf{E} \in \mathsf{CompMod}$, $\boldsymbol{\gamma}\colon \mathbf{C} \to\!\!\!\!\bullet\, \mathbf{D}, \boldsymbol{\delta}\colon \mathbf{D} \to\!\!\!\!\bullet\, \mathbf{E}$, *and* $\boldsymbol{\epsilon}\colon \mathbf{E} \to\!\!\!\!\bullet\, \mathbf{Sets}$ *is a presheaf-simulation, then the following strictness conditions hold:*
(i) $\sum_{\boldsymbol{\epsilon}} \mathbf{1}_{\mathbf{E}} = \mathbf{1}_{\sum_{\mathbf{E}} \boldsymbol{\epsilon}}$.
(ii) $\sum_{\boldsymbol{\epsilon}} (\boldsymbol{\delta} \circ \boldsymbol{\gamma}) = \sum_{\boldsymbol{\epsilon}} \boldsymbol{\delta} \circ \sum_{(\boldsymbol{\epsilon} \circ \boldsymbol{\delta})} \boldsymbol{\gamma}$.

Proof. (i) It suffices to 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 $(\mathbf{1}_{\mathbf{E}}(t), u) = (t, u)$, so on the level of the underlying class-functions the two simulations agree. For the forcing relations we see that both simulations are the corresponding diagonal.
(ii) To verify this equation on the level of underlying classes we have 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 that $x \Vdash_{(t,b)}^{\sum_{\epsilon} \delta} y$ 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 $x \Vdash_{(t,b)}^{\sum_{\epsilon} \delta \circ \gamma} z$. □

Theorem 4.3. *The category* CompMod *is a type-category.*

Proof. This follows immediately from Lemma 4.1, Lemma 4.2, and the fact that CompMod has a terminal object, as explained in Remark 3.3. □

This result bridges the categorical interpretation of dependent types with the theory of computability models. In section 6 we discuss its importance and its role in our future work.

5. Fibration-simulations and opfibration-simulations

The (covariant) Grothendieck construction allows the generation of fibrations (opfibrations), as the first-projection functor $\mathsf{pr}_1\colon \sum_{\mathcal{C}} P \to \mathcal{C}$ is a (split) opfibration, if P is a covariant presheaf, or a (split) fibration, if P is a contravariant presheaf. In this section we introduce the notion of a fibration and opfibration-simulation and we show that the first-projection-simulation $\mathsf{pr}_{\mathbf{1}}\colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \to\!\!\!\!\bullet\, \mathbf{C}$ is a (split) opfibration-simulation, as we work with covariant presheaf-simulations. The dual result is shown similarly.

In this section, $\mathbf{E}$ is a computability model over T and $\mathbf{B}$ a computability model over U. Moreover, the pair

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

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

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

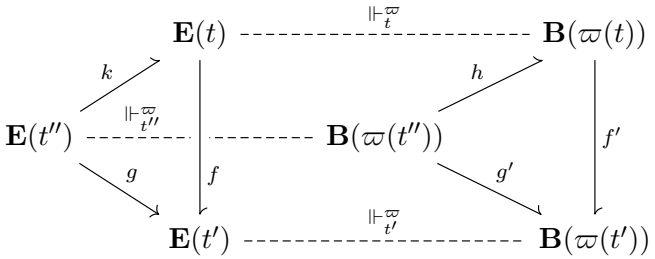
$$\mathbf{E}(t_1) \xrightarrow[g]{} \mathbf{E}(t_2) \xleftarrow[f]{} \mathbf{E}(t_3)$$

we have a whole class, in general, of opspans

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

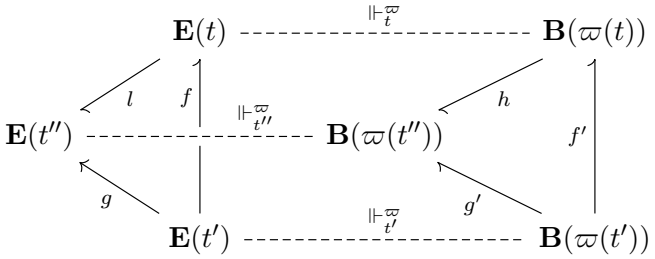
such that f' tracks f and g' tracks g.

Definition 5.1 (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 $y \Vdash^{\varpi}_{t''} x$, $y \in \mathrm{dom}(f' \circ h) \cap \mathrm{dom}(g')$, and $f'(h(y)) = g'(y)$, then $x \in \mathrm{dom}(f \circ k) \cap \mathrm{dom}(g)$ and $g(x) = f(k(x))$.

Definition 5.2 (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



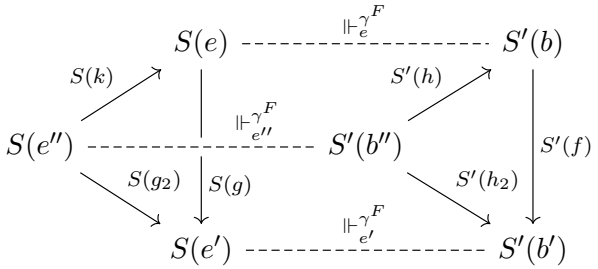
that is g' tracks g, there is some $l \in \mathbf{E}[t, t'']$ satisfying the following property: h tracks l, and for every $x \in \mathbf{E}(t'), y \in \mathbf{B}(\varpi(t'))$, such that $y \Vdash^{\varpi}_{t'} x$, $y \in \mathrm{dom}(h \circ f') \cap \mathrm{dom}(g')$, and $f'(h(y)) = g'(y)$, then $x \in \mathrm{dom}(l \circ f) \cap \mathrm{dom}(g)$ 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 5.3 (Fibration-simulation)**.** We call $\boldsymbol{\varpi} \colon \mathbf{E} \twoheadrightarrow \mathbf{B}$ a fibration-simulation, 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 5.4 (Opfibration-simulation)**.** We call $\boldsymbol{\varpi}\colon \mathbf{E} \to \mathbf{B}$ 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.

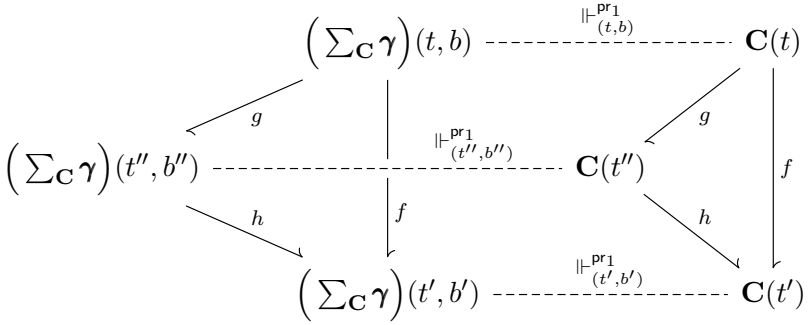
Example 5.5. Let $\mathcal{E}, \mathcal{B}$ be categories with presheaves S, S' and let $F\colon \mathcal{E} \to \mathcal{B}$ be a fibration, such that $S' \circ F = S$. Then, $\boldsymbol{\gamma}^F\colon \mathbf{CM}^{\mathrm{tot}}(\mathcal{E}; S) \twoheadrightarrow \mathbf{CM}^{\mathrm{tot}}(\mathcal{B}; S')$ is a fibration-simulation. To see this, assume we are given a computable function in $\mathbf{CM}^{\mathrm{tot}}(\mathcal{B}; S')$, that is a function $S'(f)\colon S'(b) \to S'(b')$, and $e \in \mathcal{E}$ such that $F(e') = b'$. As F is a fibration, we find an arrow $g\colon e \to e'$ 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 $\mathbf{CM}^{\mathrm{prt}}(\mathcal{E}; S)(e) = S(e)$ and $\mathbf{CM}^{\mathrm{prt}}(\mathcal{B}; S')(b) = S(b)$, for every e and b, respectively. As g is cartesian over f and b', we obtain an arrow $k\colon e'' \to e$, 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 5.6. *If* $\mathbf{C}$ *is a computability model and* $\gamma\colon \mathbf{C} \to \mathbf{Sets}$ *a simulation, then the first-projection-simulation* $\mathbf{pr}_1\colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \twoheadrightarrow \mathbf{C}$ *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



and we observe that h fills also the triangle on the left, as we have that $f = h \circ g$ whenever they are defined, so 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.

Definition 5.7. A splitting for a fibration-simulation $\boldsymbol{\varpi}\colon \mathbf{E} \twoheadrightarrow \mathbf{B}$ 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 a function $f' \in \mathbf{E}[u, u']$ cartesian for f and u. This rule $\varpi^{\triangle}$ is subject to the following conditions:

- For every $f \in \mathbf{B}[t_1, t_2]$ and every $g \in \mathbf{B}[t_2, t_3]$ we have that
$$\varpi^{\triangle}(g \circ f, u_1) = \varpi^{\triangle}(g, u_1) \circ \varpi^{\triangle}(f, u_2).$$
- For every $t \in T$ we have that $\varpi^{\triangle}(\mathbf{1}_{\mathbf{B}(t)}, u) = (\mathbf{1}_{\mathbf{E}(u)}, u)$.

A splitting for an opfibration-simulation $\boldsymbol{\varpi}\colon \mathbf{E} \twoheadrightarrow \mathbf{B}$ 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 a function $f' \in \mathbf{E}[u, u']$ opcartesian over f and u. This rule $\varpi^{\triangle}$ is subject to the following conditions:

- For every $f \in \mathbf{B}[t_1, t_2]$ and every $g \in \mathbf{B}[t_2, t_3]$ we have that
$$\varpi^{\triangle}(g \circ f, u_1) = \varpi^{\triangle}(g, u_2) \circ \varpi^{\triangle}(f, u_1).$$
- For every $t \in T$ we have that $\varpi^{\triangle}(\mathbf{1}_{\mathbf{B}(t)}, u) = (\mathbf{1}_{\mathbf{E}(u)}, u)$.

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

Corollary 5.8. *The simulation* $\mathbf{pr}_1\colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \twoheadrightarrow \mathbf{C}$ *is a split opfibration-simulation.*

Proof. We can simply take $\mathbf{pr}_1^{\triangle}$ to be defined by the rule $\mathbf{pr}_1^{\triangle}(f, u) := (f, u)$. □

6. Conclusions and future work

In [10] many concepts and results from category theory were translated to the theory of computability models, where equalities between arrows are replaced by certain relations between type names and (partial) computable functions. In this paper we extended the work initiated in [12, 13] by translating the Grothendieck construction and the notions of fibration and opfibration within computability models. The category CompMod was shown to be a type-category, a fact that allows the transport of concepts and facts from the theory of type-categories to 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 to the proof of a corresponding density theorem with that of the Grothendieck category to the proof of the categorical density theorem. For that, we need to introduce forcing and tracking-moduli in

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 $F\colon \mathcal{C} \to \mathcal{D}$	simulation $\gamma\colon \mathbf{C} \to\!\!\!\!\triangleright\, \mathbf{D}$
category of Sets	computability model of $\mathbf{Sets}$
presheaf $P\colon \mathcal{C} \to$ Sets	presheaf-simulation $\gamma\colon \mathbf{C} \to\!\!\!\!\triangleright\, \mathbf{Sets}$
representable functor $\mathrm{Hom}(a, -)$	representable simulation γ_{t_0}
representable functor $\mathrm{Hom}(-, a)$	representable-simulation δ_{t_0}
Grothendieck category $\sum_{\mathcal{C}} P$	Grothendieck computability model $\sum_{\mathbf{C}} \boldsymbol{\gamma}$
first-projection functor	first-projection-simulation
$\mathsf{pr}_1\colon \sum_{\mathcal{C}} P \to \mathcal{C}$	$\mathsf{pr}_{\mathbf{1}}\colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \to\!\!\!\!\triangleright\, \mathbf{C}$
(op)cartesian arrow	(op)cartesian computable function
(op)fibration $\pi\colon \mathcal{E} \to \mathcal{B}$	(op)fibration-simulation $\boldsymbol{\varpi}\colon \mathbf{E} \to\!\!\!\!\triangleright\, \mathbf{B}$
split (op)fibration	split (op)fibration-simulation

the definition of a simulation i.e., realisers for the forcing and tracking relations. We hope to elaborate these concepts in subsequent work.

In [15], 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 [15], pp. 117-118. Theorem 4.3 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[1]. In subsequent work we plan to study models of various dependently typed algebraic theories within CompMod. In [15] 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 that every type-category has (see Theorem 4.6 in [14]). Namely, the simulations

$$\boldsymbol{\phi}\colon \mathbf{C} \to\!\!\!\!\triangleright\, \sum_{\mathbf{C}} \boldsymbol{\gamma}, \quad \text{with} \quad \mathsf{pr}_{\mathbf{1}} \circ \boldsymbol{\phi} = \mathbf{1}_{\mathbf{C}}$$

are the canonical *dependent functions* over $\mathbf{C}$ and $\boldsymbol{\gamma}$.

Our approach to (op)fibration-simulations and (op)cartesian functions is different from the 2-categorical approach to fibrations in [17, 19]. In a subsequent work we will explain the exact relation between our approach and the 2-categorical one in detail. Namely, we will show that our approach to cartesian arrows yields a different notion from the 2-categorical one. Nonetheless, we 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.

[1] As Longley and Normann remark in [10], p. 544, "there is unexplored territory here, e.g. in combining constructive type theory and set theory with classical approaches to functional algorithms".

References

[1] R. Cockett: *Categories and Computability*, Lecture Notes, 2014.

[2] R. Cockett, P. Hofstra: Introduction to Turing categories, Annals of Pure and Applied Logic 156(2-3), 2008, 183–209.

[3] R. Cockett, P. Hofstra: Categorical simulations, Journal of Pure and Applied Algebra 214(10), 2010, 1835–1853.

[4] J. Cartmell: Generalised algebraic theories and contextual categories, Annals of Pure and Applied Logic, 32, 1986, 209–243.

[5] L. Gambarte, I. Petrakis: Categories with a base of computability, in preparation, 2024.

[6] P. Johnstone: *Sketches of an elephant: A topos theory compendium*, Oxford University Press, 2002

[7] J. Longley: *Realizability Toposes and Language Semantics*, PhD Thesis ECS-LFCS-95-332, University of Edinbourgh, 1995.

[8] J. Longley: On the ubiquity of certain total type structures, Mathematical Structures in Computer Science 17(5), 2007, 841–953.

[9] J. Longley: Computability structures, simulations and realizability, Mathematical Structures in Computer Science 24(2), 2014, E240201.

[10] J. Longley, D. Normann: *Higher-Order Computability*, Springer, 2015.

[11] I. Petrakis: Computability models over categories, arXiv:2105.06933v1, 2021.

[12] I. Petrakis: Computability models over categories and presheaves, Logical Foundations of Computer Science 2022, S. Artemov, A. Nerode (Eds.), Springer, LNCS 13137, 2022, 253-265.

[13] I. Petrakis: Strict computability models over categories and presheaves, Journal of Logic and Computation, exac077, 2022, https://doi.org/10.1093/logcom/exac077.

[14] I. Petrakis: Categories with dependent arrows, arXiv:2303.14754v1, 2023.

[15] A. M. Pitts: Categorical logic, in S. Abramsky, D. M. Gabbay, T. S. E. Maibaum (Eds.) *Handbook of Logic in Computer Science*, Vol. 5, Clarendon Press, Oxford, 2000, 39–128.

[16] G. Rosolini: *Continuity and effectiveness in topoi*, PhD Thesis, University of Oxford, 1986.

[17] E. Riehl: *Two-sided discrete fibrations in 2-categories and bicategories*, 2010, available at `https://math.jhu.edu/~eriehl/fibrations.pdf`.

[18] E. Riehl: *Category Theory in Context*, Dover Publications Inc., 2016.

[19] L. Z. Wong: *The Grothendieck Construction in Enriched, Internal and ∞-Category Theory*, PhD Thesis, University of Washington, 2019.