Categorical Aspects of Dependency and Computability

Dissertation zur Erlangung des Grades

Dr. rer. nat.

vorgelegt an der

Ludwig-Maximilians-Universität München
Fakultät für Mathematik, Informatik und Statistik
Mathematisches Institut

von

Luis Antonio Gambarte

Betreuer	Priv.-Doz. Dr.	Iosif Petrakis
Prüfer	Priv.-Doz. Dr.	Iosif Petrakis
	Prof. Dr.	____________________
Berichterstatter	Prof. Dr.	____________________

Contents

Acknowledgements

I would like to thank the following people for their support and help during the work on this thesis:

Dr. Iosif Petrakis: I probably will not find enough words to express my gratitude towards Iosif, who fundamentally shaped both the way I think and write about mathematics. His support and guidance through the world as an aspiring logician was invaluable in practically all stages of the present work and mentioning all the times he helped me through trouble would probably make this paragraph to long, so I will end by reiterating the enormous (positive!) impact he had on me during these times.

Prof. Jacopo Emmenegger: For his remarks during a meeting in Luminy regarding a possible equivalence between generalised categories with attributes and (2-dep,Σ)-categories, which inspired a substantial portion of section 7.3.

Prof. Helmut Schwichtenberg: He was always a great host of the Oberseminar and kindly allowed me to participate in the yearly Autumn schools “Proof and Computation” which exposed me to many great mathematicians whose talks taught my how to properly talk about mathematics myself.

Felix Weitkämper, DPhil (Oxon): I am very grateful for the two times I assisted his lecture “Logic in Computer Science” at the Institute for Computer Science and his positive spirit which uplifted me when I needed it the most.

There are no words to thank my parents for their never-ending and continuous support.

Papers related to this thesis

1. Luis Gambarte and Iosif Petrakis. “The Grothendieck computability model”. In: *[17]*. Torino, Italy, 2024
2. Luis Gambarte and Iosif Petrakis. “The Grothendieck Computability Model”. In: *Theoretical Computer Science* 1057 (Dec. 2025), p. 115550. issn: 0304-3975. doi: `10.1016/j.tcs.2025.115550`
3. Luis Gambarte and Iosif Petrakis. “Bases of Computability”. To Be Submitted

The second paper is an extension of the first and the entirety of its contents can be found in part II. The contents of the third paper are also wholly contained in this part.

Chapter 1
Introduction

Types and categories In the early 20th century Russell concerned himself with the paradox that now carries his name, which says the following: If we consider all possible sets, then the "set" $\{x \mid x \notin x\}$ can not be a set itself. As a solution he proposed in [59] that each variable should not exist in a "vacuum", but instead be assigned a type. This would make it clear that the totality $\{x \mid x \notin x\}$ could not be of the same type as the x it contains and thus the paradox is resolved, which could be seen as the birth of type theory. However his theory of types has still some shortcomings from the current point of view, that is we do not have the ability to form types in the context of a term of a given type. However such definitions lie at the heart of mathematics, as if we are for example given a field k, that is naïvely $k : \mathrm{field}$ where field is the type of fields, we want to be able to create a type $\mathrm{Vec(k)}$ of vector spaces over this field k.

A type theory emerging some time later, in the 1970s, pioneered by Martin-Löf addresses this: He proposed his intuitionistic theory of types which precisely allows one to form dependent types of the above form and further. He further proposed that this intuitionistic type theory ought to be seen as a foundation of mathematics. Even later, in the 2010s Homotopy type theory is developed [64] which features another approach to dependent types. We will get into further details about differences in these two approaches later.

Since its introduction in [22] for topological purposes category theory started to enter all areas of mathematics. One of its first impacts in mathematical logic came through Lawvere 's thesis [42] in which he subsumed many mathematical theories as what he called algebraic theories and then showed how these algebraic theories are connected to certain categories. This result led him to stipulate the "category of categories" (if such object exists) as a foundation for all mathematics. Later in [41] he continued on this path by introducing hyperdoctrines, a concept purely built from categorical notions which should serve as an abstraction from theories. He already notes in [41, p.12] that

> "Any (single-sorted) theory formalized in higher-order logic yields a hyperdoctrine [...]"

Thus the natural question arising from this observation is whether such correspondences between categorical constructs and (typed) theories can also be found in other settings. This was answered positively by Lambek and Scott in [40] where they linkey cartesian closed categories to typed λ-calculi and type theories (in their sense) and topoi. However, their type theories do not include dependent types.

For type theories with dependent types the attempts of categorisation can be associated to one of the two following sorts, depending on their underlying motivation.

- Based on Martin-Löf's approach which realises dependent types as types formed in contexts, which are finite lists of variables in their respective types, the categories realising such type theories are usually developed in the framework of fibred category theory. The most general notion in this sort of attempts are the comprehension categories of Jacobs [34, 33, 35] in which the contexts are the objects of the base categories and the types that can be formed in these contexts are realised as the object of the category above their respective contexts in the base category of the fibration. They may be seen as a generalisation of the preceding display map categories introduced by Taylor in his thesis [62].

- Similiar to the approach developed in [64] which realises dependent types as dependent function types into an universe type, types dependent on variables of another type are realised as additional structure on objects extraneous to the category. Examples of this approach are CARTMELL's categories with attributes [9],PITTS' type categories [51], and many more. The approach we will focus on mostly is that of the categories with family arrows of PETRAKIS [48].

Many attempts have been made to reconcile different approaches of these two sorts. BLANCO showed in [5] that categories with attributes due to CARTMELL are equivalent to discrete comprehension categories. A recent paper by AHRENS et al. [1] showed that all approaches are equivalent to a discrete notion of a comprehension category. However, one question remained.

Question A. *What categorical notion of the second above sort captures the non-discrete cases of comprehension categories?*

We will show that the categories with family arrows of PETRAKIS and an extension of those due to EHRHARDT [20] are the required categories.

Computability and categories As there are many different theoretical notion of computability, Turing computability, Kleene computability, the theory of partial continuous functionals to name a few, the desire arose to have a unified theory allowing one to speak about these models and compare their features in one common language. An early approach were partial combinatory algebras, whose connections to topoi were examined by LONGLEY in his thesis[44].

Independently, and in order to make categories appropriate for computability, others tried to find an appropriate primitive notion of partiality for arrows in categories. One of the earliest approaches are dominical categories due to DI PAOLA and HELLER [18], which were generalised by ROSOLINI to p-categories in his thesis [58], were further generalised to restriction categories by COCKETT and LACK in [11, 12, 13]. Surprisingly each of these categories can be embedded into the category of partial arrows of a category with a base of computability[1]

Later LONGLEY introduced another candidate for a general theory of computability, his theory of (higher) computability models [43, 45], which abstract partial combinatory algebras. He furthermore extended the notion of the category of assemblies to computability models hence a way to go from computability models to categories had been established. PETRAKIS then noticed in [50] that categories with a base of computability induce computability models.

Thus the following questions arise:

Question B. *What categorical structure does the category of computability models carry and can it be associated to a type theory as mentioned before?*

Question C. *What properties do the mappings*

$$\{\textit{Computability models}\} \mapsto \{\textit{Categories + bases of computability}\},$$
$$\{\textit{Categories + bases of computability}\} \mapsto \{\textit{Computability models}\}$$

have? Do they constitute an equivalence of categories?

We will at least determine which structure the category of computability models must not carry and give intermediate results concerning question C.

[1] These are called dominions by ROSOLINI in [58].

Organisation of this thesis

tba

Contributions

tba

Preliminaries

Chapter a
(2-)categorical prerequisites

a.1 Notation for categorical notions

We will not reiterate all basic notions of category theory here and instead refer the reader to [2] (a reprint available online is [3]) for the standard constructions of category theory, namely products, coproducts, limits, colimits and such. However we shall make the following notational conventions which may differ from the usual notation:

- We denote the identity on an arbitrary object c in a category $\mathscr{C}$ as $\mathbf{1}_c$. We also adopt this convention for sets. For a category $\mathscr{C}$ we denote the identity functor on $\mathscr{C}$ as $\mathrm{id}_{\mathscr{C}}$.
- We denote the von Neumann-ordinals using blackboard bold numbers, that is
$$\mathbb{0} = \emptyset, \mathbb{1} = \{\mathbb{0}\}, \mathbb{2} = \{\mathbb{0}, \mathbb{1}\}, \mathbb{3} = \{\mathbb{0}, \mathbb{1}, \mathbb{2}\}, \ldots .$$
- We denote the standard simplices as $[0], [1], [2], \ldots$, and regard them as linear orders (categories) where the morphisms are given by the edges and are always orientied upwards, that is
$$\begin{tikzcd} {[0] =} & 0, & {[1] =} & 0 \arrow[r] & 1, & {[2] =} & 0 \arrow[r] \arrow[dr] & 1, \arrow[d] & \ldots \\ & & & & & & & 2 & \end{tikzcd}$$
- We denote monic arrows in a category with "↪" and epic arrows with "↠".
- We denote the pullback of a morphism $g\colon c_1 \to c_2$ along a morphism $f\colon c_3 \to c_2$ with $f^{-1}g$ and the object corresponding to the pullback with $f^{-1}(c_1)$. We denote in diagrams that a square is a pullback square with a "⌟" as follows:
$$\begin{tikzcd} f^{-1}(c_1) \arrow[r, "g^{-1}f"] \arrow[d, "f^{-1}g"'] \arrow[dr, phantom, "\lrcorner", very near start] & c_1 \arrow[d, "g"] \\ c_3 \arrow[r, "f"] & c_2. \end{tikzcd}$$
For any span $c_3 \xleftarrow{f'} c \xrightarrow{g'} c_1$ making the obvious square commute we denote the unique arrow $c \to f^{-1}(c_1)$ obtained from the universal property of the pullback with $\langle f', g' \rangle$.
- Dually, we denote the pushout of a morphism $g\colon c_1 \to c_2$ along a morphism $f\colon c_1 \to c_3$ with f_*g and the object corresponding to the pushout with $f_*(c_1)$. We denote in diagrams that a square is a pushout square with a "⌟" as follows:
$$\begin{tikzcd} c_1 \arrow[r, "f"] \arrow[d, "g"'] & c_2 \arrow[d, "f_*g"] \\ c_3 \arrow[r, "g_*f"] & f_*(c_2) \arrow[ul, phantom, "\lrcorner", very near start] \end{tikzcd}$$
For any cospan $c_3 \xrightarrow{f'} c \xleftarrow{g'} c_1$ making the obvious square commute we denote the unique arrow $f_*(c_2) \to c$ obtained from the universal property of the pullback with $[f', g']$.

This notation differs from the usual $i^{-1}(c_3)$ and $c_2 +_{c_1} c_3$ used in most texts, but is used by Rosolini in [57] and as we will later concern ourselves with sets and presheaves, the notation is chosen to reflect the connection to inverse image and direct image.

a.2 Notions of fibrations

For this we first recall some terminology regarding 2-categories. For an introduction to 2-categories we refer to [36]. For us a 2-category will always be locally small, and such 2-categories are simply categories $\mathscr{C}$ enriched over **Cat**.

Notation (for 2-categories):

- We denote 2-categories using the same script as for normal categories, that is $\mathscr{C}, \mathscr{D}, \mathscr{E}, \dots$.
- The different types of n-cells in a 2-category are labelled as such:

0-cells	are denoted using uppercase latin letters like	$X, Y, Z, \dots,$
1-cells	are denoted using lowercase latin letters like	$f, g, h, \dots,$
2-cells	are denoted using lowercase greek letters like	$\alpha, \beta, \gamma, \cdots.$

- Both the composition of 1-cells and the vertical composition of 2-cells are denoted using "$\circ$".
- If $\alpha\colon e \Rightarrow e'$, where $e, e'\colon X \to Y$ and $\beta\colon h' \Rightarrow h$, where $h, h'\colon D \to X$ are 2-cells, then the result of the *horizontal composition* is denoted $\alpha \bullet \beta$. Visually:

$$\begin{tikzcd}
D \arrow[r, bend left, "h'"{name=A}] \arrow[r, bend right, "h"'{name=B}] \arrow[from=A, to=B, Rightarrow, "\beta"] & X \arrow[r, bend left, "e'"{name=C}] \arrow[r, bend right, "e"'{name=E}] \arrow[from=C, to=E, Rightarrow, "\alpha"] & Y \arrow[r, phantom, "="] & D \arrow[r, bend left, "e'\circ h'"{name=F}] \arrow[r, bend right, "e\circ h"'{name=G}] \arrow[from=F, to=G, Rightarrow, "\beta\bullet\alpha"] & Y.
\end{tikzcd}$$

In the special case that either α or β are identity transformations $\mathbf{1}_h$ (or $\mathbf{1}_e$), we use the notation $h \bullet \alpha$ (or $\beta \bullet e$ respectively) instead.

Definition a.2.1 (Cartesian 2-cells in a 2-category). Let $\mathscr{C}$ be a 2-category, $\alpha\colon e \Rightarrow e'$ (where $e, e'\colon X \to Y$) be a two-cell and $p\colon Y \to E$ be a one-cell. We say that α is *p-cartesian* if for all $f\colon D \to X$ the 2-cell $\alpha \bullet f$ is a $(p \circ =)$-cartesian arrow in the category $\mathscr{C}(D, Y)$.

The above definition of p-cartesian 2-cells can be further unpacked by analysing what it means for $\alpha \bullet f$ to be $(p \circ =)$-cartesian. It means that if we are given a 2-cell $\xi\colon e'' \Rightarrow e \circ f$ resulting in the cospan

$$\begin{tikzcd}
e'' \arrow[dr, Rightarrow, "\xi"] & \\
e' \circ f \arrow[r, Rightarrow, "\alpha\bullet f"'] & e \circ f
\end{tikzcd} \tag{1}$$

and a 2-cell γ such that

$$\begin{tikzcd}
p \circ e'' \arrow[d, Rightarrow, "\gamma"'] \arrow[dr, Rightarrow, "p\bullet\xi"] & \\
p \circ e' \circ x \arrow[r, Rightarrow, "p\bullet\alpha\bullet f"'] & p \circ e \circ x
\end{tikzcd} \tag{2}$$

(where we already used that $(p \circ =)(\alpha \bullet f) = p \bullet \alpha \bullet f$ and $(p \circ =)(\xi) = p \bullet \xi$) we get a 2-cell ζ such that

$$\begin{tikzcd}
e'' \arrow[d, Rightarrow, "\zeta"'] \arrow[dr, Rightarrow, "\xi"] & \\
e' \circ f \arrow[r, Rightarrow, "\alpha\bullet f"'] & e \circ f
\end{tikzcd}$$

commutes. Now to unravel this a bit. To give a functor ξ as in (1) amounts to giving a triangle

$$\begin{tikzcd}
D \arrow[rr, "e''"] \arrow[dr, "f"'] & {} \arrow[d, Rightarrow, "\xi"] & Y \\
& X \arrow[ur, "e'"'] &
\end{tikzcd}$$

and giving a γ as in (2) amounts to giving a 2-cell as in

$$\begin{tikzcd}
D \arrow[rr, "e''"] \arrow[d, "f"'] & & Y \arrow[d, "p"] \\
X \arrow[r, "e'"'] & Y \arrow[r, "p"'] & E.
\arrow[from=1-3, to=2-1, Rightarrow, shorten=4mm, "\gamma"]
\end{tikzcd}$$

where the equalities means that $(p \bullet \alpha \bullet f) \circ \gamma = p \bullet \xi$.

Chapter b
Type theories with dependent types

We are going to give a brief exposition of the two kinds of type theories with dependent types that underlie the different approaches to dependent function-objects in categories. This material is taken from [30] and [64], albeit sometimes adapted to our notational conventions.

b.1 Dependent types out of contexts

Assume we are given an infinite supply $x_1, x_2, x_3, \dots$ of metavariables ranging over elements of types, $\sigma_1, \sigma_2, \dots$ as metavariables for type-terms. Then a *context* Γ is a (possible empty) sequence of the form $x_1 : \sigma_1, x_2 : \sigma_2, \dots$ where each σ_i contains only x_j for $j < i$ freely for all i. Such a calculus has six elementary assertions, namely

$\vdash \Gamma\ \mathit{ctxt}$	Γ is a valid context
$\Gamma \vdash \sigma\ \mathit{type}$	σ is a type in context Γ
$\Gamma \vdash M : \sigma$	M is a term of type σ in context Γ
$\vdash \Gamma = \Delta\ \mathit{ctxt}$	Γ and Δ are definitionally equal contexts
$\Gamma \vdash \sigma = \tau\ \mathit{type}$	σ and τ are definitionally equal types in context Γ
$\Gamma \vdash M = N : \sigma$	M and N are definitionally equal terms of type σ in context Γ.

The rules to form contexts are as follows:

$$\text{C-Emp}\,\frac{}{\vdash [\,]\ \mathit{ctxt}} \qquad \frac{\Gamma \vdash \sigma\ \mathit{type}}{\vdash \Gamma, x : \sigma\ \mathit{ctxt}}\,\text{C-Ext}$$

where the variable x in the second rule has to be fresh.

$$\frac{\vdash \Gamma = \Delta\ \mathit{ctxt} \qquad \Gamma \vdash \sigma = \tau\ \mathit{type}}{\vdash \Gamma, x : \sigma = \Delta, y : \tau\ \mathit{ctxt}}\,\text{C-Ext-Eq}$$

Here the variables x, y have to be fresh.

The following rule asserts our metatheoretical assertion that contexts are build up from *only* variables.

$$\frac{\vdash \Gamma, x : \sigma, \Delta\ \mathit{ctxt}}{\Gamma, x : \sigma, \Delta \vdash x : \sigma}\,\text{Var}$$

The following groups of three rules assert that equality of contexts, types and terms is an equivalence relation.

- *For contexts:*

$$\frac{\vdash \Gamma = \Delta\ \mathit{ctxt}}{\vdash \Delta = \Gamma\ \mathit{ctxt}}\,\text{C-Eq-S} \qquad \frac{\vdash \Gamma\ \mathit{ctxt}}{\vdash \Gamma = \Gamma\ \mathit{ctxt}}\,\text{C-Eq-R}$$

  and

$$\frac{\vdash \Gamma = \Delta\ \mathit{ctxt} \qquad \vdash \Delta = \Theta\ \mathit{ctxt}}{\vdash \Gamma = \Theta\ \mathit{ctxt}}\,\text{C-Eq-T}$$

- *For types:*

$$\frac{\Gamma \vdash \sigma\ \mathit{type}}{\Gamma \vdash \sigma = \sigma\ \mathit{type}}\,\text{Ty-Eq-R} \qquad \frac{\Gamma \vdash \sigma = \tau\ \mathit{type}}{\Gamma \vdash \tau = \sigma\ \mathit{type}}\,\text{Ty-Eq-S}$$

  and

$$\frac{\Gamma \vdash \sigma = \tau\ \mathit{type} \qquad \Gamma \vdash \tau = \rho\ \mathit{type}}{\Gamma \vdash \sigma = \rho\ \mathit{type}}\,\text{Ty-Eq-T}$$

- *For terms:*

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M = M : \sigma}\text{Tm-Eq-R} \qquad \frac{\Gamma \vdash M = N : \sigma}{\Gamma \vdash N = M : \sigma}\text{Tm-Eq-S}$$

  and

$$\frac{\Gamma \vdash M = N : \sigma \qquad \Gamma \vdash N = O : \sigma}{\Gamma \vdash M = O : \sigma}\text{Tm-Eq-T}$$

The equalities between context and types interact according to the following rules:

$$\frac{\Gamma \vdash M : \sigma \qquad \vdash \Gamma = \Delta \ \textit{ctxt} \qquad \Gamma \vdash \sigma = \tau \ \textit{type}}{\Delta \vdash M : \tau}\text{Tm-Conv}$$

and

$$\frac{\vdash \Gamma = \Delta \ \textit{ctxt} \qquad \Gamma \vdash \sigma \ \textit{type}}{\Delta \vdash \sigma \ \textit{type}}\text{Ty-Conv}$$

Hofmann also demands the following two rules: (Here $\Box$ is either of the judgements $M : \sigma, M = N : \sigma, \sigma \ \textit{type}, \sigma = \tau \ \textit{type}.$

$$\frac{\Gamma, \Delta \vdash \Box \qquad \Gamma \vdash \rho \ \textit{type}}{\Gamma, x : \rho, \Delta \vdash \Box}\text{Weak}$$

$$\frac{\Gamma, x : \rho, \Delta \vdash \Box \qquad \Gamma \vdash U : \rho}{\Gamma, \Delta[U/x] \vdash \Box[U/x]}\text{Subst}$$

Here $\Box[U/x]$ and $\Delta[U/x]$ refers to the usual capture-free substitution one is familiar with from logic, that is bound variables are renamed to avoid free variables in U becoming bound.

However these additional rules are not of great importance for us, for the example of a comprehension category generated from such a type theory it is only of relevance to us that we can define our equivalence relation on context through this definitional equality and that this equivalence is well-behaved with types.

b.2 Dependent types as function types

On the other hand the type theories with dependent types as introduced in [64] are defined differently. They do not use contexts, but instead rely on *type universes.* So we are now reduced to only four judgements instead of the previous six and we no longer have contexts, thus our primitive judgements look like this:

$\vdash \sigma \ \textit{type}$	σ is a type
$\vdash M : \sigma$	M is a term of type σ
$\vdash \sigma = \tau \ \textit{type}$	σ and τ are definitionally equal types.
$\vdash t = s : \sigma$	t and s are definitionally equal terms of type σ

However we demand the existence of special types, our universes $\mathcal{U}_i$ for $i = 0, 1, \ldots$. Note that these indices i are not canonically identified with some type $\mathbb{N}$ of natural numbers. We demand that

$$\frac{}{\vdash \mathcal{U}_i \ \textit{type}} \qquad \frac{}{\vdash \mathcal{U}_i : \mathcal{U}_{i+1}} \qquad \frac{\vdash A : \mathcal{U}_i}{\vdash A \ \textit{type}}$$

for all i. In the same fashion as [64] we will write $\vdash A : \mathcal{U}$ if $\vdash A : \mathcal{U}_i$ for some i.

- *Product types*: We want to be able to form product types for all types.

$$\frac{\vdash A \ \textit{type} \qquad \vdash B \ \textit{type}}{\vdash A \times B \ \textit{type}} \qquad \frac{\vdash A = A' \ \textit{type} \qquad \vdash B = B' \ \textit{type}}{\vdash A \times B = A' \times B' \ \textit{type}}$$

The following rules ensure that these product types behave like we would the cartesian product in a category to behave.

$$\frac{\vdash t_1 : A \qquad \vdash t_2 : B}{\vdash \langle t_1, t_2 \rangle : A \times B} \qquad \frac{\vdash t : A \times B}{\vdash \mathsf{pr}_1(t) : A} \qquad \frac{\vdash t : A \times B}{\vdash \mathsf{pr}_2(t) : B}$$

$$\frac{\vdash t_1 = t_1' : A \qquad \vdash t_2 = t_2' : B}{\vdash \langle t_1, t_2 \rangle = \langle t_1', t_2' \rangle : A \times B} \qquad \frac{\vdash t = t' : A \times B}{\vdash \mathsf{pr}_1(t) = \mathsf{pr}_1(t') : A} \qquad \frac{\vdash t = t' : A \times B}{\vdash \mathsf{pr}_2(t) = \mathsf{pr}_2(t') : B}$$

- *Function types:*

$$\frac{\vdash A \ \mathit{type} \qquad \vdash B \ \mathit{type}}{\vdash A \to B \ \mathit{type}} \qquad \frac{\vdash A = A' \ \mathit{type} \qquad \vdash B = B' \ \mathit{type}}{\vdash A \to B = A' \to B' \ \mathit{type}}$$

The following rules ensure that the terms of these function type behave like functions. First we postulate that we can apply function terms to terms of the domain type:

$$\frac{\vdash A \ \mathit{type} \qquad \vdash B \ \mathit{type} \qquad \vdash f \colon A \to B \qquad \vdash a : A}{\vdash f(a) \colon B}$$

$$\frac{\vdash A \ \mathit{type} \qquad \vdash B \ \mathit{type} \qquad \vdash f = g \colon A \to B \qquad \vdash a = a' : A}{\vdash f(a) = g(a') \colon B}$$

Secondly we postulate that we can obtain functions by λ-abstraction:

$$\frac{\vdash A \ \mathit{type} \qquad \vdash B \ \mathit{type} \qquad \vdash b : B \qquad \vdash x : A}{\vdash \lambda x.b : A \to B}$$

$$\frac{\vdash A \ \mathit{type} \qquad \vdash B \ \mathit{type} \qquad \vdash b = b' : B \qquad \vdash x : A}{\vdash \lambda x.b = \lambda x.b' : A \to B}$$

where x and y are both variables. We demand that the term b contains at most the variable x freely. Then x is bound in $\lambda x.b$.

The usual rules governing this term formation are demanded, that is α-conversion,β-reduction and η-expansion.

$$\frac{\vdash A \ \mathit{type} \qquad \vdash B \ \mathit{type} \qquad \vdash t \colon B \qquad \vdash x \colon A \qquad \vdash y \colon A}{\vdash \lambda x.t = \lambda y.t : A \to B} \ \alpha\text{-conv}$$

$$\frac{\vdash A \ \mathit{type} \qquad \vdash B \ \mathit{type} \qquad \vdash t \colon B \qquad \vdash a \colon A}{\vdash \big(\lambda x.t\big)(a) = t[x/a] : B} \ \beta\text{-red}$$

$$\frac{\vdash A \ \mathit{type} \qquad \vdash B \ \mathit{type} \qquad \vdash f \colon A \to B \qquad \vdash x \colon A}{\vdash \lambda x.f(x) = f : A \to B} \ \eta\text{-exp}$$

Here $t[x/a]$ is simply the term where all occurrences of x are replaced by a.

We additionally demand all the rules turning definitional equality of types and terms into an equivalence relation as before. In this framework the dependent types are then simply modelled as types $A \to \mathcal{U}_i$ where A is a type. This theory is developed further in [64] by adding types $\prod_{a:A} B(a), \sum_{a:A} B(a)$ for types $A, B \colon A \to \mathcal{U}_i$, but we will not reproduce this here as it was merely our intent to highlight the different approaches to dependent function types that motivate categories with family arrows and comprehension categories.

Part I

Equivalences between categories with dependent arrows and comprehension categories

In this part we will compare two approaches to categories for dependent structures motivated by two different approaches to dependent types in type theory. There already exists a survey by Ahrens et al. (cf. [1]) which compares most of the established approaches, namely Comprehension categories, categories with families[1] and type categories and the categories they are organized in, where comprehension categories are the most abstract notion. Building on this we compare comprehension categories to categories with family arrows, and their generalisations introduced by Petrakis in [48] and 2-family arrows introduced by Ehrhardt in [20]. Categories with family arrows and their more abstract notions can be summarized in the cube

$$\begin{tikzcd}
 & (\mathsf{dep},\Sigma)\mathsf{C} \arrow[rr] & & (\mathsf{2\text{-}dep},\Sigma)\mathsf{C} \\
\mathsf{depC} \arrow[ur] \arrow[rr] & & \mathsf{2\text{-}depC} \arrow[ur] & \\
 & (\mathsf{fam},\Sigma)\mathsf{C} \arrow[uu] \arrow[rr] & & (\mathsf{2\text{-}fam},\Sigma)\mathsf{C} \arrow[uu] \\
\mathsf{famC} \arrow[u] \arrow[ur] \arrow[rr] & & \mathsf{2\text{-}famC} \arrow[u] \arrow[ur] &
\end{tikzcd}$$

where each category is the category of one of those notions (these will be explained later) and the arrows are inclusions.

In this cube each direction corresponds to adding an abstraction, that is

- going up the y-axis (‘↑’) corresponds to adding dependent arrows,
- going along the x-axis (‘→’) corresponds to adding arrows between family arrow (that is “2-dimensional” structure) and
- going along the z-axis (‘↗’) corresponds to adding Σ-objects. (Again, all these notions will be explained in the succeding chapters.)

On the other hand are the comprehension categories of Jacobs which build on (discrete/Grothendieck) fibrations and higher structure built using these. The notions already explored in the literature falling into this approach can be summarized as the normal nodes in

$$\begin{tikzcd}
 & \mathsf{HComprC}^{\mathrm{str}}_{\mathrm{disc}} \arrow[rr] & & \mathsf{HComprC}^{\mathrm{str}} \\
\mathsf{dFibb} \arrow[ur] \arrow[rr] & & \mathsf{sas} \arrow[ur] & \\
 & \mathsf{ComprC}_{\mathrm{disc}} \arrow[uu] \arrow[rr] & & \mathsf{ComprC}^{\mathrm{str}}_{\mathrm{spl}} \arrow[uu] \\
\mathsf{dFib} \arrow[u] \arrow[ur] \arrow[rr] & & \mathsf{Fib} \arrow[u] \arrow[ur] &
\end{tikzcd}$$

The nodes highlighted in red correspond to new notions that will be introduced in the following chapters, but first we will elaborate on the already established notions and the 2-equivalences between the corresponding nodes in the two cubes.

[1] Coraglia and Di Liberti introduced a notion of *generalised categories with families* in [15], whose biequivalence to comprehension categories was shown in [14], which were not mentioned in this survey paper.

Chapter 2
(2-)fam-categories and fibrations

In this chapter we recall the basic notions of categories with (2-)family arrows due to Petrakis and Ehrhardt as well as the notions of (Grothendieck) fibrations due to Grothendieck and Gray. The equivalence results are a trivial extension of the known equivalence of split Grothendieck fibrations and functors into the category of categories (modulo size issues) to a 2-dimensional framework.

2.1 Categories with family arrows

Definition 2.1.1 (fam-categories – [48]). A *fam-category* is a category $\mathscr{C}$ that comes equipped with a collection $\mathrm{fHom}(c)$ of *family arrows* for each $c \in \mathscr{C}$. These family arrows are usually denoted with greek letters λ, μ, et cetera. In diagrams they are denoted as $c \xrightarrow{\lambda} \cdot$.

- For each $c, c' \in \mathscr{C}$ there is a composition operation $\circ\colon \mathrm{fHom}(c) \times \mathrm{Hom}(c', c) \to \mathrm{fHom}(c')$, such that the following two conditions are fulfilled:

  (fam$_1$) for all $\lambda \in \mathrm{fHom}(c)$ we have that $\lambda \circ \mathbf{1}_c = \lambda$.

$$\begin{tikzcd} c \arrow[r, "1_c"] \arrow[rr, bend left, "\lambda"] & c \arrow[r, "\lambda"] & {} \end{tikzcd} .$$

  (fam$_2$) for all $\lambda \in \mathrm{fHom}(c)$ we have that $\lambda \circ (f \circ g) = (\lambda \circ f) \circ g$.

$$\begin{tikzcd} c \arrow[r, "f"] \arrow[rrr, bend left=40, "(\lambda\circ g)\circ f"] \arrow[rr, bend right, "g\circ f"'] \arrow[rrr, bend right=50, "\lambda\circ(g\circ f)"'] & d \arrow[r, "g"] \arrow[rr, bend left, "\lambda\circ g"] & e \arrow[r, "\lambda"] & {} \end{tikzcd} .$$

Examples 2.1.2. 1. One can establish a fam-category structure on Sets by letting $\mathrm{fHom}(S)$ for any set by the set of families $(B_s)_{s\in S}$ where the B_s are sets. The composition operation is then given by mapping $\big((B_s)_{s\in S}, f\colon T \to S\big)$ to the family $(B_{f(t)})_{t\in T}$.

2. One can turn every category into a fam-category by setting $\mathrm{fHom}(c) = \mathscr{C}_0$, the collection of objects of the category, for every $c \in \mathscr{C}$. The composition is then simply defined by mapping (d, f) to d for every object d and every morphism f of $\mathscr{C}$.

3. If $\mathscr{C}$ is locally small one can define $\mathrm{fHom}(\mathscr{C}) = [\mathscr{C}^{\mathrm{op}}, \mathsf{Sets}]$, and the composition is simply defined by $(G, F) \mapsto G \circ F$. Thus **Cat** is a fam-category.

4. (Pitts) *Families in a topos* Let $\mathscr{C}$ be a topos with subobject classifier $(\top, \Omega)$: If $a \in \mathscr{C}$ set

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

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

$$\begin{tikzcd}
& & c\times b \arrow[dl, "\mathsf{pr}_c"'] \arrow[dr, "\mathsf{pr}_b"] \arrow[dd, dashed, "g\times 1_b"] \arrow[ddd, rounded corners, to path={-- ([yshift=2ex]\tikztostart.north) -| ([xshift=8ex]\tikztotarget.east -| 0,0) -- (\tikztotarget)}, "(b,e)\circ g"] & & \\
& c \arrow[dl, "g"'] & & b \arrow[dr, "1_b"] & \\
a & & a\times b \arrow[ll, "\mathsf{pr}_a"] \arrow[rr, "\mathsf{pr}_b"'] \arrow[d, "e"'] & & b \\
& & \Omega. & &
\end{tikzcd}$$

2.1.3 Dependent function types yield fam-categories. Given a type theory T as in [64] (a quick recapitulation can be found in b.2) one can define a fam-category $\mathscr{C}_T$ as follows (see [48, 20]):

- Its objects are types $A : \mathcal{U}$. The morphisms from A to B are given by terms $t : A \to B$. The identity on A is $\lambda x.x$ for some variable $x : A$. The composition of $t\colon A \to B, s\colon B \to C$ is given by $\lambda x.s\big(t(x)\big)$.
- The family arrows of A are given as the families of types over A, that is

$$\mathrm{fHom}(A) = \bigcup_{i=0}^{\infty}\{t : A \to \mathcal{U}_i\}.$$

The composition of $t\colon A \to \mathcal{U}_i$ and $s\colon B \to A$ is given by

$$\lambda x.t\big(s(x)\big).$$

The properties of fam-categories follow immediately from the axioms of such type theories.

Definition 2.1.4 (fam-functors and fam-natural transformations). Let $\mathscr{C}$ and $\mathscr{D}$ be fam-categories. A *fam-functor* $F\colon \mathscr{C} \to \mathscr{D}$ is a functor $F\colon \mathscr{C} \to \mathscr{D}$ of the underlying categories together with an assignment rule

$$F_c\colon\ \mathrm{fHom}(c) \to \mathrm{fHom}\left(F(c)\right), \quad \lambda \mapsto F_c(\lambda)$$

such that

(fam$_3$) for all $c, d \in \mathscr{C}, f : c \to d$ and $\lambda \in \mathrm{fHom}(d)$ the following holds:

$$F_c(\lambda \circ f) = F_c(\lambda) \circ F(f).$$

A *fam-natural transformation* $\eta\colon F \Rightarrow G$ of fam-functors $F, G\colon \mathscr{C} \to \mathscr{D}$ is a natural transformation of the underlying functors F, G such that

(fam$_4$) for every $c \in \mathscr{C}$ and $\lambda \in \mathrm{fHom}(c)$

$$\begin{tikzcd}
F(c) \arrow[rr, "\eta_c"] \arrow[dr, "F_c(\lambda)"] & & G(c) \arrow[dl, "G_c(\lambda)"] \\
& {} &
\end{tikzcd}$$

commutes, that is $G_c(\lambda) \circ \eta_c = F_c(\lambda)$.

Usually we will omit the subscript in F_a, as it is evident from the family arrow it is applied to.

For general examples see [20, 48], we only give an example corresponding to the categories arising from type theory.

2.1.5 Maps between type theories yield fam-functors. If T, S are type theories with respective type universes $\mathcal{U}, \mathcal{V}$ and we are given a rule F that assigns

- to each type $A : \mathcal{U}$ a type $F(A) : \mathcal{V}$ such that each type universe $\mathcal{U}_i$ gets mapped to a type universe $\mathcal{V}_j$and
- to each term $t : A$ a term $F(t) : F(A)$

such that definitional equality is respected, we obtain a fam-functor $F\colon \mathscr{C}_T \to \mathscr{C}_S$ by

$$A \mapsto F(A), \quad \big(t : A \to B \mapsto F(t) : F(A) \to F(B)\big), \quad \big(t : A \to \mathcal{U}_i \mapsto F(t) : F(A) \to F(\mathcal{U}_i)\big).$$

Definition 2.1.6 (The category of fam-categories). Let famC be the 2-category whose

- *objects* are categories with family arrows.
- *1-maps* are fam-functors and
- *2-maps* are fam-natural transformations.

2.1.7 Some remarks about size issues. In the above definition of the categroy famC one encounters the usual question of size issues one gets when defining "categories of categories". Namely, if one does not restrict the size of the collection of objects and arrows paradoxes like the one of Russell might arise. Thus if we form categories of categories throughout this paper we restrict ourselves to the case that these are *locally small.*

However, this is not sufficient for the present definition, as we additionally need to control the size of the totality of family arrows, hence we extend the local smallness to the family-arrows, that is for each $c \in \mathscr{C}$ for a given object c of some fam-category $\mathscr{C}$ we demand that $\mathrm{fHom}(c)$ be some form of set. We do not need the full power of set theory for the equivalences in the following sections and chapter, but we need to be able to form unions indexed by objects, that is we need to be able to form the collection

$$\coprod_{c \in \mathscr{C}} \mathrm{fHom}(c)$$

given a fam-category $\mathscr{C}$. Additionally we will require in the following a form of comprehension of sets, namely given a functor $p\colon \mathscr{E} \to \mathscr{C}$ we need to be able to form collections (sets) of the form

$$\{e \in \mathscr{E} \mid p(e) = c\}$$

where c is some object of $\mathscr{C}$.

2.2 Discrete fibrations

Definition 2.2.1 (discrete (op-)fibrations). A functor $p\colon \mathscr{E} \to \mathscr{B}$ is a *discrete fibration* if for each $e \in \mathscr{E}$ and each $f\colon b \to p(e)$ in $\mathscr{B}$ there is a unique $g\colon e' \to e$ such that $p(g) = f$. It is called a *discrete opfibration* if for each $e \in \mathscr{E}$ and each $f\colon p(e) \to b$ in $\mathscr{B}$ there is a $g\colon e \to e'$ such that $p(g) = f$.

Definition 2.2.2 (Maps of discrete fibrations). Let $p\colon \mathscr{E} \to \mathscr{B}$ and $q\colon \mathscr{E}' \to \mathscr{B}'$ be two discrete fibrations. A map $p \to q$ is a pair $(\hat{F}, F)$ of a functor $F\colon \mathscr{B} \to \mathscr{B}'$ and another functor $\hat{F}\colon \mathscr{E} \to \mathscr{E}'$ which lies over F in the following sense:

- The diagram

$$\begin{tikzcd}
\mathscr{E} \arrow[r, "\hat{F}"] \arrow[d, "p"] & \mathscr{E}' \arrow[d, "q"] \\
\mathscr{B} \arrow[r, "F"] & \mathscr{B}'
\end{tikzcd}$$

  commutes and the lift of $f\colon b' \to b$ along $e \in \mathscr{E}$ with $p(e) = b$ gets mapped by $\hat{F}$ to the lift of $F(f)$ along $\hat{F}(e)$.

Furthermore a *transformation* of maps $(\hat{F}, F) \Rightarrow (\hat{G}, G)$ is a pair $(\hat{\eta}, \eta)$ of a natural transformation $\eta\colon F \Rightarrow G$ and a transformation $\hat{\eta}\colon \hat{F} \Rightarrow \hat{G}$ living over η in the following sense:

- we have that $q \bullet \hat{\eta} = \eta \bullet p$, that is

$$\begin{tikzcd}
\mathscr{E} \arrow[r, bend left, "\hat{F}", ""{name=U, below}] \arrow[r, bend right, "\hat{G}"', ""{name=D}] \arrow[d, "p"] & \mathscr{E}' \arrow[d, "q"] \\
\mathscr{B} \arrow[r, bend left, "F", ""{name=U2, below}] \arrow[r, bend right, "G"', ""{name=D2}] & \mathscr{B}'
\arrow[Rightarrow, from=U, to=D, "\hat{\eta}"]
\arrow[Rightarrow, from=U2, to=D2, "\eta"]
\end{tikzcd}$$

  commutes.

Remark 2.2.3. Actually the second condition on maps of discrete fibrations is superfluous: if we are given $f\colon b' \to b$ and a lift $\overline{f}(e)\colon f^*(e) \to e$ along e, then $\hat{F}\big(\overline{f}(e)\big)\colon F(f^*(e) \to F(e)$ is a lift of $F(f)$ along $F(e)$ and thus *the* unique lift.

Definition 2.2.4 (Category of discrete fibrations). Let dFib be the 2-category whose

- *objects* are discrete fibrations,
- *1-maps* are pseudo-maps of discrete fibrations,
- *2-maps* are transformations of pseudo-maps.

Proposition 2.2.5. *Letting* famC *be the 2-category of fam-categories, fam-functors and fam-natural transformations as well as* dFib *be the 2-category of discrete fibrations, maps of discrete fibrations and transformations of such maps we get a 2-equivalence*

$$\mathsf{famC} \simeq \mathsf{dFib}\,.$$

Proof: We begin by describing the two 2-functors.

$\boxed{p_{-}\colon \mathsf{famC} \to \mathsf{dFib}\text{:}}$ This functor is given by the following data:

- it maps a given fam-category $\mathscr{C}$ to the discrete fibration $p_{\mathscr{C}}\colon \mathscr{F} \to \mathscr{C}$ where $\mathscr{C}$ is the category whose objects are pairs (c, λ) where $c \in \mathscr{C}$ and $\lambda \in \mathrm{fHom}(c)$. The only arrows in $\mathscr{F}$ are arrows $f\colon (c, \lambda \circ f) \to (c', \lambda)$ where $f\colon c \to c'$ is an arrow in $\mathscr{C}$. The composition is then defined via the usual composition on $\mathscr{C}$. It is immediate that this composition satisfies the unity and associativity laws.
- A fam-functor $F\colon \mathscr{C} \to \mathscr{D}$ is mapped to a map $(p_F, F)\colon p_{\mathscr{C}} \to p_{\mathscr{D}}$ of fibrations as such:
  - the functor p_F takes (c, λ) in $\mathscr{F}$ to $\big(F(c), F_c(\lambda)\big)$.
  - An arrow $f\colon (c, \lambda \circ f) \to (c', \lambda)$ is taken to $F(f)\colon \big(F(c), F_c(\lambda \circ f)\big) \to \big(F(c'), F_{c'}(\lambda)\big)$. To see that this is well-defined we simply use that $F_c(\lambda \circ f) = F_{c'}(\lambda) \circ F(f)$.

  It is immediate that this constitutes a functor as the interchangeability of p_F with composition and identity follows from the functoriality of F.

- A fam-natural transformation $\eta\colon F \to G$ is taken to a transformation of maps of fibrations $(\eta, p_\eta)\colon (F, p_F) \to (G, p_G)$ as follows: the maps $\hat{\eta}_{(c,\lambda)}$ for $(c,\lambda) \in \mathscr{F}$ are given by η_c. It is immediate that $\eta_c\colon \big((F(c), F_c(\lambda)\big) \to \big(G(c), G_c(\lambda)\big)$ as by definition of fam-natural transformations we have $F_c(\lambda) = G_c(\lambda) \circ \eta_c$.

It is immediate that this p_{-} is a 2-functor from the definition, a simple calculation proves that it respects associativity and unity for both 1- and 2-morphisms.

$\boxed{\mathsf{F}_{-}\colon\ \mathsf{dFib} \to \mathsf{famC}\colon}$ In this direction we define the functor as follows:

- Given $p\colon \mathscr{E} \to \mathscr{B}$ we define our fam-category to have as underlying category $\mathscr{B}$ and the family arrows are given by the fibres, $\mathrm{fHom}(b) := p^{-1}(b)$. The precomposition of $\lambda \in \mathrm{fHom}(b)$ with $f\colon b' \to b$ is given by the domain of the unique lift of f along p and λ.

  It is immediate that this constitutes a fam-category as $\mathscr{B}$ is a category and for all $b \in \mathscr{B}$ and all $\lambda \in \mathrm{fHom}(b) = p^{-1}(b)$ we have that $\lambda \circ 1_b = \lambda$, as $1_\lambda\colon \lambda \to \lambda$ is a lift of 1_b along p and λ and as p is discrete it is the unique arrow with this property. Furthermore $\lambda \circ (g \circ f) = (\lambda \circ g) \circ f$ as the unique lift of $g \circ f$ at λ is given by the composition of the unique lift of g at λ and f at $\lambda \circ g$.

- Given a map $(\hat{F}, F)\colon p \to q$ as in

$$\begin{tikzcd}
\mathscr{E}_p \arrow[r, "\hat{F}"] \arrow[d, "p"] & \mathscr{E}_q \arrow[d, "q"] \\
\mathscr{B}_p \arrow[r, "F"] & \mathscr{B}_q
\end{tikzcd} \tag{3}$$

  we obtain our fam-functor $\mathsf{F}_{F,\hat{F}}\colon \mathscr{B}_p \to \mathscr{B}_q$ as follows: the underlying functor is $F\colon \mathscr{B}_p \to \mathscr{B}_q$ and on the family arrows we set $(\mathsf{F}_{F,\hat{F}})_b(\lambda) = \hat{F}(\lambda)$. We then simply compute that $(\mathsf{F}_{F,\hat{F}})_c(\lambda \circ f) = \hat{F}(\lambda \circ f) = \hat{F}(\lambda) \circ F(f)$ where we used the commutativity of (3) for the last equality as $\lambda \circ f$ is the domain of the unique lift of f along λ.

- Given a transformation $(\hat{\eta}, \eta)\colon (\hat{F}, F) \Rightarrow (\hat{G}, G)$ as in

$$\begin{tikzcd}
\mathscr{E}_p \arrow[r, bend left, "\hat{F}", ""{name=U, below}] \arrow[r, bend right, "\hat{G}"', ""{name=D}] \arrow[d, "p"] \arrow[Rightarrow, from=U, to=D, "\hat{\eta}"] & \mathscr{E}_q \arrow[d, "q"] \\
\mathscr{B}_p \arrow[r, bend left, "F", ""{name=U2, below}] \arrow[r, bend right, "G"', ""{name=D2}] \arrow[Rightarrow, from=U2, to=D2, "\eta"] & \mathscr{B}_q
\end{tikzcd}$$

  we define our fam-natural functor $\mathsf{F}_{(\hat{\eta},\eta)}$ via

$$(\mathsf{F}_{(\hat{\eta},\eta)})_b = \eta_b.$$

  It is then immediate that

$$\mathsf{F}_{(\hat{G},G)}(\lambda) \circ (\mathsf{F}_{(\eta,\hat{\eta})})_b = \hat{G}(\lambda) \circ \eta_b = \hat{F}(\lambda)$$

  as $\hat{G}(\lambda) \circ \eta_b$ is the lift of η_b along $\hat{G}(\lambda)$ and q, but by definition $q(\hat{\eta}_\lambda) = \eta_b$ and $\hat{F}(\lambda)$ is the codomain of $\hat{\eta}_\lambda$.

One now computes that $\mathsf{F}_{-} \circ p_{-} = \mathrm{id}_{\mathsf{famC}}$ and $p_{-} \circ \mathsf{F}_{-} = \mathrm{id}_{\mathsf{dFib}}$, hence we have an isomorphism of 2-categories. Q.E.D.

2.3 2-fam-categories

Definition 2.3.1 (2-fam-categories – [20]). A fam-category $\mathscr{C}$ is a 2-fam-*category*, if for each $c \in \mathscr{C}$ the collection $\mathrm{fHom}(c)$ is a category whose morphisms are called *2-family arrows.* A 2-family arrow $\eta \in \mathrm{Hom}(\lambda, \mu)$ is pictured as follows:

$$\begin{tikzcd} c \arrow[r, bend left=50, "\lambda"{name=U}] \arrow[r, bend right=50, "\mu"'{name=D}] & {} \arrow[Rightarrow, from=U, to=D, "\eta"] \end{tikzcd} .$$

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

(2fam$_1$) *Compatibility:* For each $c, c', c'' \in \mathscr{C}$, each $\lambda, \mu \in \mathrm{fHom}(c'')$, each $\eta \in \mathrm{Hom}(\lambda, \mu)$ and each $f \in \mathrm{Hom}(c, c'), g \in \mathrm{Hom}(c', c'')$ the equation on the left holds (which equivalently means the diagrams on the right commute.)

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

$$\begin{tikzcd} c \arrow[r, "1_c"] \arrow[rr, bend left=70, "\lambda"{name=A}] \arrow[rr, bend right=70, "\mu"'{name=B}] & c \arrow[r, bend left=50, "\lambda"{name=U}] \arrow[r, bend right=50, "\mu"'{name=D}] & {} \arrow[Rightarrow, from=U, to=D, "\eta"] \arrow[Rightarrow, from=A, to=B, bend right=90, "\eta\bullet_{c,c}1_c"'] \end{tikzcd}$$

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

$$\begin{tikzcd} c \arrow[r, "f"] \arrow[rrr, bend left=60, "\lambda\circ g\circ f"{name=A}] \arrow[rrr, bend right=60, "\mu\circ g\circ f"'{name=B}] & c' \arrow[r, "g"] & c'' \arrow[r, bend left=50, "\lambda"{name=U}] \arrow[r, bend right=50, "\mu"'{name=D}] & {} . \arrow[Rightarrow, from=U, to=D, "\eta"] \arrow[Rightarrow, from=A, to=B, bend left=90, "\eta\bullet_{c'',c}(g\circ f)"] \end{tikzcd}$$

(2fam$_2$) *Distributivity:* For each $c, c' \in \mathscr{C}$, each $\lambda, \delta, \gamma \in \mathrm{fHom}(c')$ and $\eta \in \mathrm{Hom}(\delta, \gamma), \eta' \in \mathrm{Hom}(\gamma, \lambda)$ as well as $f \in \mathrm{Hom}(c, c')$ the equation on the left holds (or equivalently the diagram on the right commutes).

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

$$\begin{tikzcd} c \arrow[r, "f"] & c' \arrow[r, "\delta"{name=M}] \arrow[r, bend left=80, "\lambda"{name=U}] \arrow[r, bend right=80, "\gamma"'{name=D}] & {} . \arrow[Rightarrow, from=U, to=M, "\eta"] \arrow[Rightarrow, from=M, to=D, "\eta'"] \end{tikzcd}$$

2.3.2 Arrows between dependent function types. Continuing the motivating example 2.1.3 Ehrhardt extends the family-arrow structure by 2-family arrows as such. Given $t, t' \in \mathrm{fHom}(A)$, that is two terms $t : A \to \mathcal{U}_i, t' : A \to \mathcal{U}_j$ we first observe that without loss of generality $j > i$ and thus $\mathcal{U}_i : \mathcal{U}_j$ which entails $t : A \to \mathcal{U}_j$. Then Ehrhardt defines

$$\mathrm{fHom}(t, t') := \prod_{x:A} \big(t(x) \to t'(x)\big).$$

The composition of $\nu \in \mathrm{fHom}(t, t')$ and $s : B \to A$ he defines as $\lambda x.\nu\big(s(x)\big)$.

Definition 2.3.3 (2-fam-functors – [20]). Given 2-fam-categories $\mathscr{C}, \mathscr{D}$ a 2-fam-functor $F \colon \mathscr{C} \to \mathscr{D}$ consists of

- a family functor $F \colon \mathscr{C} \to \mathscr{D}$ (in the sense of definition 2.1.4),
- A functor $F^{\lambda,\mu} \colon \mathrm{fHom}(\lambda, \mu) \to \mathrm{fHom}\big(F(\lambda), F(\mu)\big)$ for every $c \in \mathscr{C}, \lambda, \mu \in \mathrm{fHom}(c)$

such that the following condition holds:

(2fam$_3$) for all arrows $f \colon b \to c$ in $\mathscr{C}$ and all $\lambda, \mu \in \mathrm{fHom}(c), \eta \colon \lambda \Rightarrow \mu$ we have that

$$F^{\lambda,\mu}(\eta \circ f) = F^{\lambda,\mu}(\eta) \circ F(f).$$

Definition 2.3.4 (2-fam natural transformations). Given 2-fam categories $\mathscr{C}, \mathscr{D}$ and 2-fam functors $F, G \colon \mathscr{C} \to \mathscr{D}$, a 2-fam natural transformation is a fam-natural transformation in the sense of 2.3.3 of the underlying fam-functors F, G such that the following condition is met:

(2fam$_4$) For all $c \in \mathscr{C}, \lambda, \mu \in \mathrm{fHom}(c)$ and $\omega \in \mathrm{fHom}(\lambda, \delta)$ the equation $G^{\lambda,\mu}(\omega) \circ \eta_c = F^{\lambda,\mu}(\omega)$ holds, which is equivalent to the commutativity of

$$\begin{tikzcd}
F(c) \arrow[rr, "\eta_c"] \arrow[dr, "F(\lambda)"] \arrow[dr, bend right=50, "F(\mu)"'] & \arrow[dl, phantom, "\overset{\Leftarrow}{F(\omega)}"] \arrow[dr, phantom, "\overset{\Leftarrow}{G(\omega)}"] & G(c) \arrow[dl, "G(\mu)"'] \arrow[dl, bend left=50, "G(\lambda)"] \\
& \cdot &
\end{tikzcd}$$

Definition 2.3.5 (Category of 2-fam-categories). Let 2-famC be the 2-category whose

- *objects* are 2-fam-categories,
- *1-maps* are 2-fam functors and whose
- *2-maps* are 2-fam-natural transformations.

2.4 Grothendieck fibrations

The following notions of cartesian arrows and fibrations were initially introduced by Grothendieck in [29] and then further elaborated on in [28]. However, we use a slightly different definition which can be shown to be equivalent to the one introduced by Grothendieck.

Definition 2.4.1 (cartesian arrows, Grothendieck fibrations). Let $p \colon \mathscr{E} \to \mathscr{B}$ be a functor. Given an object $e \in \mathscr{E}$ and an arrow $f \colon b \to p(e)$ in $\mathscr{B}$, we say that an arrow $g \colon e' \to e$ is *p-cartesian for f and e* if both

- $p(g) = f$ and
- for all $h \colon e'' \to e$ and $i \colon p(e'') \to b$ such that $p(h) = f \circ i$ we get a unique $j \colon e'' \to e'$ such that $p(j) = i$. As a diagram:

$$\begin{tikzcd}
& e'' \arrow[dl, dotted, "\exists ! j"] \arrow[dd, "h"] \arrow[rr, mapsto, "p"] & & p(e'') \arrow[dl, "i"] \arrow[dd, "p(h)"] \\
e' \arrow[dr, "g"'] \arrow[rr, mapsto, "p"] & & b \arrow[dr, "f"] & \\
& e \arrow[rr, mapsto, "p"] & & p(e).
\end{tikzcd}$$

A functor $p\colon \mathscr{E} \to \mathscr{B}$ is a *Grothendieck fibration* if for each $f\colon b' \to b$ in $\mathscr{B}$ and $e \in \mathscr{E}$ such that $p(e) = b$ we obtain a unique arrow $g\colon e' \to e$ that is p-cartesian for f and e. We call this g the lift of f along p and e.

2.4.2 Historical remarks. The original definition due to Grothendieck in [29] takes a completely different approach, which by current standards would fit better to 2-category theory:

Namely he considers a category $\mathscr{F}$ fibred over a base category $\mathscr{C}$ to be an assignment $c \mapsto \mathscr{F}_c$ mapping each object of $\mathscr{C}$ to a category. Additionally Grothendieck demands that each morphism $f\colon c \to c'$ is mapped to a functor $f^*\colon \mathscr{F}_{c'} \to \mathscr{F}_c$ that maps each $\xi \in \mathscr{F}_{c'}$ to a pullback $\xi \times_{c'} c$ For composable pairs $f\colon c \to c', g\colon c' \to c''$ a natural equivalence $c_{f,g}\colon (g \circ f)^* \to f^* \circ g^*$, which turns this assignment into a pseudo-functor from $\mathscr{C}$ to the category of categories (although the term pseudo-functor was not used).

Later Gray altered the definition to more closely resemble the definition we use today, he started assembling the category $\mathscr{F}_c$ into a "supercategory" $\mathscr{E}$. Grayalso introduced the notions of pseudo-functor and of p-cartesian arrows in [27], although the latter is weakened: In the given definition the condition holds if i is the identity on b, whereas the stronger notion we use is called *strong cartesian* (It is also called *hypercartesian* in [4]).

It is straightforward to show that if p is a fibration then every arrow p-cartesian in the stronger sense is p-cartesian in the weaker sense and vice versa.

From now on a fibration will always mean Grothendieck fibration.

Examples 2.4.3. 1. Given any category $\mathscr{C}$, the domain functor $\mathrm{dom}\colon \mathscr{C}^{\to} \to \mathscr{C}$ (where $\mathscr{C}^{\to}$ is the category of arrows and commutative squares of $\mathscr{C}$) is a Grothendieck fibration. Given $g\colon b \to c \in \mathscr{C}^{\to}$ and $f\colon a \to b$ in $\mathscr{C}$ a dom-cartesian lift of f along g is given by the commutative square

$$\begin{tikzcd} a \arrow[r, "g\circ f"] \arrow[d, "f"] & c \arrow[d, "1_c"] \\ b \arrow[r, "g"] & c \end{tikzcd} \tag{4}$$

in $\mathscr{C}^{\to}$, that is $(f, 1_c)\colon g \circ f \to g$ is the desired lift. To see that it is cartesian, let another commutative square

$$\begin{tikzcd} a' \arrow[r, "i"] \arrow[d, "h"] & b' \arrow[d, "h'"] \\ b \arrow[r, "g"] & c, \end{tikzcd}$$

that is the arrow $(h, h')\colon i \to g$ be given such that $\mathrm{dom}(h, h') = h = f \circ j$ for some j in $\mathscr{C}$. Thus we obtain the commutative diagram

$$\begin{tikzcd} a' \arrow[r, "i"] \arrow[d, "j"] & b' \arrow[dd, "h'"] \\ a \arrow[d, "f"] & \\ b \arrow[r, "g"] & c. \end{tikzcd}$$

Our remaining task is to find a unique horizontal morphism in the middle splitting this diagram into two commutative squares such that the lower square is 4. This morphism is h', and is necessarily unique.

2. If the category $\mathscr{C}$ has pullbacks, then also $\mathrm{cod}\colon \mathscr{C}^{\to} \to \mathscr{C}$ is a fibration. The cartesian lift of $g\colon b \to c$ along $f\colon a \to c$ is the pullback square

$$\begin{tikzcd} g^{-1}(a) \arrow[r, "f^*g"] \arrow[d, "g^*f"'] \arrow[dr, phantom, "\lrcorner", very near start] & a \arrow[d, "f"] \\ b \arrow[r, "g"] & c. \end{tikzcd}$$

We will not elaborate this further, to confirm that one can fill the triangle uniquely one uses the universal property of the pullback.

2.4.4 Grothendieck fibrations from contexts. This is the quintessential example that motivated Jacobs to introduce his comprehension categories (which we will discuss later). For this example one needs a type theory T with contexts as in exempli grata [47, 63]). However, going forth we will use the notation and definition of [30]. A small recapitulation of this type theory can be found in b.1 Given such a type theory T Jacobs defines the fibration $p_T\colon \mathscr{E}_T \to \mathscr{B}_T$ as follows:

- The base category $\mathscr{B}_T$ has objects equivalence classes $[\Gamma]$ of well-formed contexts Γ (definitional equality suffices for this equivalence) and its morphisms are defined as such: Given $\Gamma = x_1 : \sigma_1, \dots, x_n : \sigma_n$ and $\Delta = y_1 : \tau_1, \dots, y_m : \tau_m$ and without loss of generality $m \leq n$ Jacobs defines a map $f\colon [\Gamma] \to [\Delta]$ as $\langle [M_1], \dots, [M_m] \rangle$ such that $\Gamma \vdash M_i : \tau_i[M_1/y_1, \dots, M_{i-1}/y_{i-1}]$. The equivalence on terms can again be taken to be definitional equality. Like Jacobs we abbreviate $\langle [M_1], \dots, [M_m] \rangle$ to $\langle [\mathbf{M}] \rangle$.

  The composition of $\langle [\mathbf{M}] \rangle\colon \Gamma \to \Delta$ and $\langle [\mathbf{N}] \rangle\colon \Delta \to \Theta$ is defined as (it can by found in [30])
  $$\Big\langle [N_1[\mathbf{M}/y]], \dots, [N_k[\mathbf{M}/y]] \Big\rangle$$
  where the $N_i[\mathbf{M}/y]$ are formed by replacing y_i with M_i (capture-free) for all $i = 1, \dots, m$. The identity on $[\Gamma]$ is simply $\langle [x_1], \dots, [x_n] \rangle$.

- The total category $\mathscr{E}_T$ has objects $[\Gamma, x : \sigma]$ where $\Gamma \vdash \sigma$ *type* and $[\Gamma]$ is an object of $\mathscr{B}_T$. Its morphisms are $\langle [\mathbf{M}], [N] \rangle\colon [\Gamma \vdash \sigma \textit{ type}] \to [\Delta \vdash \tau \textit{ type}]$ where $\langle [\mathbf{M}] \rangle\colon [\Gamma] \to [\Delta]$ is a morphism of $\mathscr{B}_T$ and $\Gamma, x : \sigma \vdash N : \tau[\mathbf{M}/y]$ (here the substitution with $[\mathbf{M}]$ is defined as for terms above).

  Composition and identity are then defined as expected.

- The fibration $p_T\colon \mathscr{E}_T \to \mathscr{B}_T$ maps $[\Gamma \vdash \sigma \textit{ type}]$ to $[\Gamma]$ and $\langle [\mathbf{M}], [N] \rangle$ to $\langle [\mathbf{M}] \rangle$.

  For $\langle [\mathbf{M}] \rangle\colon [\Gamma] \to [\Delta]$ in $\mathscr{B}_T$ and $[\Delta \vdash \tau \textit{ type}]$ the p_T-cartesian lift of $\langle [\mathbf{M}] \rangle$ along $[\Delta \vdash \tau \textit{ type}]$ is defined as
  $$\langle [\mathbf{M}], [x] \rangle\colon [\Gamma \vdash \tau \textit{ type}] \to [\Delta \vdash \tau \textit{ type}].$$

.

Definition 2.4.5 (Cartesian functors). Let $p\colon \mathscr{E} \to \mathscr{B}$ and $q\colon \mathscr{E}' \to \mathscr{B}'$ be Grothendieck fibrations. A functor $F\colon \mathscr{E} \to \mathscr{E}'$ is *cartesian* for $G\colon \mathscr{B} \to \mathscr{B}'$ if it maps the lift of any arrow f in $\mathscr{B}$ along p and e to the lift of $G(f)$ along q and $F(e)$. One defines a map $p \to q$ to be a pair $(\hat{F}, F)$ where $F\colon \mathscr{B}' \to \mathscr{B}$ and $\hat{F}$ is cartesian for F.

Examples 2.4.6. 1. Continuing our examples from above, given two categories $\mathscr{C}, \mathscr{D}$ and a functor $F\colon \mathscr{C} \to \mathscr{D}$, we obtain a functor $F^{\to}\colon \mathscr{C}^{\to} \to \mathscr{D}^{\to}$ cartesian for F and dom in the following way:

$$F^{\to}(f\colon a \to b) = F(f)\colon a \to b, \qquad F\left(\begin{tikzcd} a' \arrow[r, "f'"] \arrow[d, "h_1"] & b' \arrow[d, "h_2"] \\ a \arrow[r, "f"] & b \end{tikzcd}\right) = \begin{tikzcd} F(a') \arrow[r, "F(f')"] \arrow[d, "F(h_1)"] & F(b') \arrow[d, "F(h_2)"] \\ F(a) \arrow[r, "F(f)"] & F(b). \end{tikzcd}$$

One immediately sees that this is a cartesian functor as our dom-cartesian lift $g \circ f$ of f along g is mapped to $F(g \circ f) = F(g) \circ F(f)$ which is the dom-cartesian lift of $F(f)$ along $F(g)$.

2. If $F\colon \mathscr{C} \to \mathscr{D}$ also preserves pullbacks, the functor $F^{\to}$ as defined above is cartesian for F and cod.

Definition 2.4.7 (Splittings/cleavages). Let $p\colon \mathscr{E} \to \mathscr{B}$ be a Grothendieck fibration. A *cleavage)* is an assignment routine that assigns to each $e \in \mathscr{E}$ and each $f\colon b \to p(e)$ in $\mathscr{B}$ an arrow p-cartesian for f and e. We use the same notation as Jacobs and denote this p-cartesian arrow as $\bar{f}(e)\colon f^*(e) \to e$.

If $\overline{1_b} = 1_e\colon e \to e$ for every $e \in \mathscr{E}$ and $\overline{g \circ f} = \overline{g} \circ \overline{f}$ for all composable g, f in $\mathscr{B}$ the cleavage is called a *splitting*.

A Grothendieck fibration $p\colon \mathscr{E} \to \mathscr{B}$ together with a splitting is called a *split fibration*.

2.4.8 Some remarks regarding the notation. The fact that the morphisms obtained from a pullback use f^*, the same notation Grothendieck used for his functors between the categories can be seen by the adjoint relation

$$\begin{tikzcd} {[\mathscr{C}^{\mathrm{op}}, \mathbf{Cat}]_{\mathrm{ps}}} \arrow[r, bend left] \arrow[r, phantom, "\simeq"] & \mathsf{Fib}(\mathscr{C}) \arrow[l, bend left] \end{tikzcd}$$

where the category on the left is the category of contravariant pseudo-functors from $\mathscr{C}$ to **Cat**, that is the fibrations in Grothendieck's sense, and the category on the right is the category of fibrations on $\mathscr{C}$ in the modern sense. Then functors f^* of Grothendieck get mapped to the cleavage maps $\overline{f}$ and vice versa. (Technically one needs the axiom of choice to equip every fibration with a cleavage but we will not concern ourserlves with this technicality.)

Definition 2.4.9 (Maps of (split) fibrations). Given two (split) fibrations $p\colon \mathscr{E} \to \mathscr{B}$ and $q\colon \mathscr{E} \to \mathscr{B}$ a map $p \to q$ is a pair $(\hat{F}, F)$ of functors $F\colon \mathscr{B} \to \mathscr{B}', \hat{F}\colon \mathscr{E} \to \mathscr{E}'$ that preserves cartesianness, namely

- $\hat{F}$ is cartesian for F.

If the fibration is split, we demand that additionally the splitting is preserved, that is

- for all $e \in E$ and $f\colon b \to p(e)$ in $\mathscr{B}$ the following equation holds:

$$\hat{F}\big(\bar{f}(e)\big) = \overline{F(f)}\big(\hat{F}(e)\big).$$

Definition 2.4.10 (Transformations of maps of fibrations). Let $p\colon \mathscr{E} \to \mathscr{B}, q\colon \mathscr{E}' \to \mathscr{B}'$ be (split) fibrations and $(\hat{F}, F), (\hat{G}, G)\colon p \to q$ be maps between them. Then a *transformation* $(\hat{F}, F) \Rightarrow (\hat{G}, G)$ is a pair $(\hat{\eta}, \eta)$ of a natural transformation $\eta\colon F \Rightarrow G$ and natural transformation $\hat{\eta}\colon \hat{F} \Rightarrow \hat{G}$ such that

1. $q \bullet \hat{\eta} = \eta \bullet p$ and
2. for all $b \in \mathscr{B}$ and $e \in \mathscr{E}$ with $p(e) = b$ we have that $\hat{\eta}_e$ is cartesian for η_b and e.

If the map is split we instead demand
2.' $\overline{\eta_b}(\hat{G}(e)) = \hat{\eta}_e$ for all $b \in \mathscr{B}$ and $e \in \mathscr{E}$ with $p(e) = b$.

Remark 2.4.11. This definition naturally extends the definition of a natural transformation betweens maps between two fibrations over the same base category, see for example [37, p. 268] (but note that Johnstone calls morphisms cartesian with respect to p *p-prone morphisms*), and the resulting natural transformations are called p-vertical transformations. Setting F, η to be the identity functor and identity transformation the condition that $\hat{\eta}_e = 1_b$ is then recovered from our notion as then $\eta_b = 1_b$ and thus $\hat{\eta}_e = \overline{1_b}(\hat{G}(e)) = 1_{G(e)}$.

Example 2.4.12. Considering $F, F' \colon \mathscr{C} \to \mathscr{D}$ together with a natural transformation $\eta \colon F \Rightarrow F'$ such that all transformation squares are cartesian. We then obtain a transformation of maps of fibrations

$$(\hat{\eta}, \eta) \colon (F^{\to}, F) \Rightarrow (F'^{\to}, F'), \quad \eta^{\to}_{f \colon a \to b} := \begin{tikzcd} F(a) \arrow[r, "F(f)"] \arrow[d, "\eta_a"] \arrow[dr, phantom, very near start, "\lrcorner"] & F(b) \arrow[d, "\eta_b"] \\ F'(a) \arrow[r, "F'(f)"] & F'(b). \end{tikzcd}$$

To see that this is natural let $(h_1, h_2) \colon f \to g$ be given, then

$$\begin{tikzcd} F^{\to}(f) \arrow[r, "F^{\to}(h_1{,}h_2)"] \arrow[d, "\eta^{\to}_f"] & F^{\to}(g) \arrow[d, "\eta^{\to}_g"] \\ F'^{\to}(f) \arrow[r, "F'^{\to}(h_1{,}h_2)"] & F'^{\to}(g) \end{tikzcd} = \begin{tikzcd} F(a) \arrow[rr, "F(f)"] \arrow[dr, "F(h_1)"] \arrow[dd, "\eta_a"] & & F(b) \arrow[dr, "F(h_2)"] \arrow[dd, "\eta_b"] & \\ & F(c) \arrow[rr, "F(g)"] \arrow[dd, "\eta_c"] & & F(d) \arrow[dd, "\eta_d"] \\ F'(a) \arrow[rr, "F'(f)"] \arrow[dr, "F'(h_1)"] & & F'(b) \arrow[dr, "F'(h_2)"] & \\ & F'(c) \arrow[rr, "F'(g)"] & & F'(d). \end{tikzcd}$$

As the cube on the right commutes to does the square on the right and the naturality is proven.

It is immediate that dom $\bullet \eta^{\to} = \eta \bullet$ dom from the above definition of $\eta^{\to}$, and that $\eta^{\to}_f$ is cartesian for η_b and f follows from the definition of η.

In the above scenario we say that $\hat{\eta}$ *lies over* η.

Lemma 2.4.13 (Categories of Grothendieck fibrations). *Let* Fib *be the 2-category whose*

- objects: *are Grothendieck fibrations,*
- 1-maps: *are pseudo-maps of Grothendieck fibrations and whose*
- 2-maps: *are transformations of such pseudo-maps*

We additionally obtain a subcategory $\mathsf{Fib}_{\mathrm{spl}}$ *whose objects are those Grothendieck fibrations are split and whose 1-maps are split maps.*

Proof: It is immediate that Fib is a well-defined category, however, it is not immediate that the splittings of fibrations are preserved by horizontal compositions of transformations. So assume we are given split fibrations, maps of split fibrations and transformations of split maps as in

$$\begin{tikzcd}
\mathscr{E}_1 \arrow[r, bend left, "\hat{F}_1"] \arrow[r, bend right, "\hat{G}_1"'] \arrow[r, phantom, "\Downarrow\hat{\alpha}"] \arrow[d, "p_1"] & \mathscr{E}_2 \arrow[r, bend left, "\hat{F}_2"] \arrow[r, bend right, "\hat{G}_2"'] \arrow[r, phantom, "\Downarrow\hat{\beta}"] \arrow[d, "p_2"] & \mathscr{E}_3 \arrow[d, "p_3"] \\
\mathscr{B}_1 \arrow[r, bend left, "F_1"] \arrow[r, bend right, "G_1"'] \arrow[r, phantom, "\Downarrow\alpha"] & \mathscr{B}_2 \arrow[r, bend left, "F_2"] \arrow[r, bend right, "G_2"'] \arrow[r, phantom, "\Downarrow\beta"] & \mathscr{B}_3.
\end{tikzcd}$$

From the definition of the horizontal composition of natural transformations we know that

$$(\hat{\beta}\circ\hat{\alpha})_e = \hat{\beta}_{\hat{G}_1(e)}\circ\hat{F}_2(\hat{\alpha}_e) \quad\text{and}\quad (\beta\circ\alpha)_b = \beta_{G_1(b)}\circ F_2(\alpha_b).$$

Similarly we can compute for an arbitrary splitting that $\overline{g\circ f}(e) = \overline{g}(e)\circ\overline{f}\big(g^*(e)\big)$. This allows us to compute

$$\begin{aligned}
\overline{(\beta\circ\alpha)_b}\big((\hat{G}_2\circ\hat{G}_1)(e)\big) &= \overline{\beta_{G_1(b)}\circ F_2(\alpha_b)}\big((\hat{G}_2\circ\hat{G}_1)(e)\big)\\
&= \overline{\beta_{G_1(b)}}\big((\hat{G}_2\circ\hat{G}_1)(e)\big)\circ\overline{F_2(\alpha_b)}\Big(\beta^*_{G_1(b)}\big((\hat{G}_2\circ\hat{G}_1)(e)\big)\Big)\\
&= \hat{\beta}_{G_1(e)}\circ\overline{F_2(\alpha_b)}\Big(\beta^*_{G_1(b)}\big((\hat{G}_2\circ\hat{G}_1)(e)\big)\Big) && (5)\\
&= \hat{\beta}_{G_1(e)}\circ\overline{F_2(\alpha_b)}\big((\hat{F}_1\circ\hat{G}_1)(e)\big)\Big) && (6)\\
&= \hat{\beta}_{G_1(e)}\circ\hat{F}_2\big(\overline{\alpha_b}\big(\hat{G}_1(e)\big)\big)\\
&= \hat{\beta}_{G_1(e)}\circ\hat{F}_1\big(\hat{\alpha}_e\big)\\
&= (\hat{\beta}\circ\hat{\alpha})_e.
\end{aligned}$$

Here in going from (5) to (6) we used that $\beta_{G_1(b)}\colon F_2(G_1(b))\to G_2(G_1(b))$ and thus $\overline{\beta_{G_1(b)}} = \hat{\beta}_{\hat{G}_1(e)}\colon \hat{F}_2(\hat{G}_1(e))\to\hat{G}_2(\hat{G}_1(e))$ (as $(\hat{\beta},\beta)$ is a transformation of split maps), hence $\beta^*_{G_1(b)}\big((\hat{G}_2\circ\hat{G}_1)(e)\big) = (\hat{F}_2\circ\hat{G}_1)(e)$. Q.E.D.

Proposition 2.4.14. *There is a 2-equivalence*

$$\mathsf{Fib}_{\mathrm{spl}}\simeq\mathsf{2\text{-}famC}\,.$$

Proof: We begin by describing the two 2-functors.

$\boxed{p_-\colon \mathsf{2\text{-}famC}\to\mathsf{Fib}_{\mathrm{spl}}\colon}$ This functor is given by the following data:

- it maps a given 2-fam-category $\mathscr{C}$ to the fibration $p_{\mathscr{C}}\colon\mathscr{F}\to\mathscr{C}$ where $\mathscr{F}$ is the category whose objects are pairs (c,λ) where $c\in\mathscr{C}$ and $\lambda\in\mathrm{fHom}(c)$. The arrows in $\mathscr{F}$ are arrows $(f,\eta)\colon(c,\mu)\to(c',\lambda)$ where $f\colon c\to c'$ is an arrow in $\mathscr{C}$ and $\mu\colon\eta\Rightarrow\lambda\circ f$ is an arrow in $\mathrm{fHom}(c)$. The composition is then defined via $(g,\theta)\circ(f,\eta) = \big(g\circ f,(\eta\circ f)\circ\theta\big)$. It is immediate that this composition satisfies the unity and associativity laws.

  The splitting is defined via

  $$f\colon b\to c\mapsto\overline{f} = (f,1_{(c,\lambda)})\colon(b,\lambda\circ f)\to(c,\lambda).$$

- A 2-fam-functor $F\colon\mathscr{C}\to\mathscr{D}$ is mapped to a map $(p_F,F)\colon p_{\mathscr{C}}\to p_{\mathscr{D}}$ of fibrations as such:
  - the functor p_F takes (c,λ) in $\mathscr{F}$ to $\big(F(c),F_c(\lambda)\big)$.

- An arrow $(f,\eta)\colon (c,\lambda) \to (c',\mu)$ is taken to $F(f), F^{\lambda,\mu}(\eta)\colon \big(F(c), F_c(\lambda)\big) \to \big(F(c'), F_{c'}(\mu)\big)$. To see that this is well-defined we simply use that $F_{c'}(\mu \circ f) = F_c(\mu) \circ F(f)$ and thus $F^{\lambda,\mu}(\eta)\colon F_c(\lambda) \to F_c(\mu \circ f)$

It is immediate that this constitutes a functor as the interchangeability of p_F with composition and identity follows from the functoriality of F. The splittings are also preserved as

$$p_F(f, 1_\lambda \circ f) = \big(F(f), F_c(1_\lambda \circ f)\big) = (F(f), 1_{F(\lambda\circ c)})$$

.

- A 2-fam-natural transformation $\eta\colon F \to G$ is taken to a transformation of maps of fibrations $(p_\eta, \eta)\colon (F, p_F) \to (G, p_G)$ as follows: the maps $(p_\eta)_{(c,\lambda)}$ for $(c,\lambda) \in \mathscr{F}$ are given by η_c. It is immediate that $\eta_c\colon \big((F(c), F_c(\lambda)\big) \to \big(G(c), G_c(\lambda)\big)$ as by definition of 2-fam-natural transformations we have $F_c(\lambda) = G_c(\lambda) \circ \eta_c$. This last property also shows that $(p_\eta)_\lambda$ is the split of η_c at $\lambda \in \mathrm{fHom}(c)$.

It is immediate that this p_- is a 2-functor from the definition, a simple calculation proves that it respects associativity and unity for both 1- and 2-morphisms.

$\boxed{\mathsf{F}_-\colon \mathsf{Fib}_{\mathrm{spl}} \to \mathsf{2\text{-}famC}\colon}$ In this direction we define the functor as follows:

- Given $p\colon \mathscr{E} \to \mathscr{B}$ we define our fam-category to have as underlying category $\mathscr{B}$ and the family arrows are given by the fibres, $\mathrm{fHom}(b) := p^{-1}(b)$. The precomposition of $\lambda \in \mathrm{fHom}(b)$ with $f\colon b' \to b$ is given by $f^*\lambda$ and the 2-maps $\eta\colon \lambda \Rightarrow \mu$ in $\mathrm{fHom}(b)$ are given by the vertical morphisms $\eta\colon \lambda \to \mu$ in $\mathscr{E}$. To precompose a vertical morphism $\eta\colon \lambda \Rightarrow \mu$ in $\mathrm{fHom}(b)$ we note that we have the situation

$$\begin{tikzcd}
 & f^*(\lambda) \arrow[rr, mapsto, "p"] \arrow[dd, "\eta\circ\overline{f}(\lambda)"] & & b' \arrow[dl, "1_{b'}"'] \arrow[dd, "f"] \\
f^*(\mu) \arrow[rr, mapsto] \arrow[dr, "\overline{f}(\mu)"'] & & b' \arrow[dr, "f"] & \\
 & \mu \arrow[rr, mapsto, "p"] & & b
\end{tikzcd}$$

(7)

thus we get a unique $g\colon \lambda \circ f = f^*(\lambda)\colon f^*(\mu) = \mu \circ f$ making the triangle on the left commute. We set $\eta \circ f := g$.

It is immediate that this constitutes a 2-fam-category as $\mathscr{B}$ is a category and (fam_1) as well as (fam_2) follow from the conditions imposed on splittings, so $\mathscr{B}$ is a fam-category. To see that it is also a 2-fam category note that for any $\eta\colon \lambda \Rightarrow \mu$ in $\mathrm{fHom}(b)$ we have that $\eta \circ 1_b = \eta$, for this we simply observe that in (7) the map $1_{f^*(\mu)}$ makes the left triangle commute. A similar observation also yields $\eta \circ (g \circ f) = (\eta \circ g) \circ f$.

- Given a map $(\hat{F}, F)\colon p \to q$ as in

$$\begin{tikzcd}
\mathscr{E}_p \arrow[r, "\hat{F}"] \arrow[d, "p"] & \mathscr{E}_q \arrow[d, "q"] \\
\mathscr{B}_p \arrow[r, "F"] & \mathscr{B}_q
\end{tikzcd}$$

(8)

we obtain our 2-fam-functor $\mathsf{F}_{F,\hat{F}}\colon \mathscr{B}_p \to \mathscr{B}_q$ as follows: the underlying functor is $F\colon \mathscr{B}_p \to \mathscr{B}_q$ and on the family arrows we set $(\mathsf{F}_{(F,\hat{F})})_b(\lambda) = \hat{F}(\lambda)$. On the 2-family arrows we set $\mathsf{F}^{\lambda,\mu}_{(F,\hat{F})}(\eta) = \hat{F}(\eta)$. We then simply compute that $(\mathsf{F}_{F,\hat{F}})_c(\lambda \circ f) = \hat{F}(\lambda \circ f) = \hat{F}(\lambda) \circ F(f)$ where we used the commutativity of (8) for the last equality as $\lambda \circ f$ is the domain of the unique lift of f along λ. A similar computation yields $F^{\lambda,\mu}(\eta \circ f) = F^{\lambda\circ f,\mu\circ f}(\eta) \circ F(f)$.

- Given a transformation $(\hat{\eta}, \eta)\colon (\hat{F}, F) \Rightarrow (\hat{G}, G)$ as in

$$\begin{tikzcd}
\mathscr{E}_p \arrow[r, bend left, "\hat{F}"{name=U}] \arrow[r, bend right, "\hat{G}"'{name=D}] \arrow[d, "p"] & \mathscr{E}_q \arrow[d, "q"] \\
\mathscr{B}_p \arrow[r, bend left, "F"{name=U2}] \arrow[r, bend right, "G"'{name=D2}] & \mathscr{B}_q
\arrow[Rightarrow, from=U, to=D, "\hat{\eta}"]
\arrow[Rightarrow, from=U2, to=D2, "\eta"]
\end{tikzcd}$$

  we define our 2-fam-natural functor $\mathsf{F}_{(\eta,\hat{\eta})}$ via

$$(\mathsf{F}_{(\eta,\hat{\eta})})_b = \eta_b.$$

  It is then immediate that

$$\mathsf{F}_{(G,\hat{G})}(\lambda) \circ (\mathsf{F}_{(\eta,\hat{\eta})})_b = \hat{G}(\lambda) \circ \eta_b = \hat{F}(\lambda)$$

  as $\hat{G}(\lambda) \circ \eta_b$ is the lift of η_b along $\hat{G}(\lambda)$ and q, but by definition $q(\hat{\eta}_\lambda) = \eta_b$ and $\hat{F}(\lambda)$ is the codomain of $\hat{\eta}_\lambda$.

One now computes that $\mathsf{F}_{-} \circ p_{-} = \mathrm{id}_{\mathsf{2\text{-}famC}}$ and $p_{-} \circ \mathsf{F}_{-} = \mathrm{id}_{\mathsf{Fib}_{\mathrm{spl}}}$ (up to renaming of objects in $\mathscr{E}$/family arrows), hence we have a 2-equivalence of 2-categories. Q.E.D.

Chapter 3
Σ-objects and comprehension categories

Both Petrakis and Ehrhardt extend their notions of fam-categories and 2-fam-categories to include specific objects associated to a family arrow over an object c in the (2-)fam-category $\mathscr{C}$. On the side of fibred categories this additional structure corresponds to comprehension categories which were introduced by Jacobs in [33].

We give a brief exposition of these notions and then extend the equivalence result of the previous chapter.

3.1 Σ-objects in (2-)fam-categories

First we give the definition of Σ-objects.

Definition 3.1.1 ((fam,Σ)-categories - [48]). A fam-category $\mathscr{C}$ is a *(fam,Σ)-category* if it additionally comes equipped with

(FΣ$_1$) for each $c \in \mathscr{C}$ an operation assigning to each $\lambda \in \mathrm{fHom}(a)$ an object $\sum_c \lambda$ (which is called the *Sigma-object* of λ) together with first-projection arrows

$$\mathsf{pr}_1^\lambda \colon \sum_c \lambda \to c.$$

(FΣ$_2$) For each $c, c' \in \mathscr{C}$ an operation that assigns $f \in \mathrm{Hom}(c', c)$ to $\sum_c f \colon \sum_{c'} \lambda \circ f \to \sum_c \lambda$ such that

$$\begin{tikzcd}
\sum_{c'} \lambda \circ f \arrow[r, "\Sigma_\lambda f"] \arrow[d, "\mathsf{pr}_1^{f^*\lambda}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_c \lambda \arrow[d, "\mathsf{pr}_1^\lambda"] \\
c' \arrow[r, "f"] & c
\end{tikzcd}$$

is a pullback square. Here the following strictness conditions must hold:

- For all $c \in \mathscr{C}$ the diagram

$$\begin{tikzcd}
\sum_a (\lambda \circ 1_a \arrow[r, "\sum_\lambda 1_a"] \arrow[d, "\mathsf{pr}_1^{a,\lambda \circ 1_a}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_a \lambda \arrow[d, "\mathsf{pr}_1^{a,\lambda}"] \\
a \arrow[r, "1_a"'] & a
\end{tikzcd}$$

  commutes, that is $\sum_\lambda 1_c = 1_{\sum_c \lambda}$.
- For all composable f, g the diagram

$$\begin{tikzcd}
\sum_c (\lambda \circ f) \circ g \arrow[r, "\sum_{(\lambda \circ f)} g"] \arrow[rr, bend left, "\sum_\lambda (f \circ g)"] \arrow[d, "\mathsf{pr}_1^{c,(\lambda \circ f) \circ g}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_b (\lambda \circ f) \arrow[r, "\sum_\lambda f"] \arrow[d, "\mathsf{pr}_1^{b,\lambda \circ f}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_a \lambda \arrow[d, "\mathsf{pr}_1^{a,\lambda}"] \\
c \arrow[r, "g"'] & b \arrow[r, "f"'] & a.
\end{tikzcd}$$

commutes, that is

$$\sum_{\lambda}(f \circ g) = \big(\sum_{\lambda} f\big) \circ \sum_{(\lambda\circ f)} g$$

Examples 3.1.2. 1. Considering the category Sets with the fam-structure as discussed above setting

$$\begin{aligned}
\sum_{S}(B_s)_{s\in S} &= \coprod_{s\in S} B_s \text{ for all sets } S \text{ and all } (B_s)_{s\in S} \in \mathrm{fHom}(S),\\
\mathsf{pr}_1^{(B_s)_{s\in S}}(s,x) &= x \text{ for all } (s,x) \in \coprod_{s\in S} B_s\\
\sum_{T} f(t,x) &= \big(f(t),x\big) \text{ for all } f\colon T \to S.
\end{aligned}$$

turns it into a (fam,Σ)-category.

2. If we consider **Cat** with the fam-structure as above we can let $\sum_{\mathscr{C}} F$ be the *Grothendieck category* whose

   - objects are pairs (c,x) where $c \in \mathscr{C}, x \in F(c)$ and whose
   - morphisms $f\colon (c,x) \to (c',y)$ are morphism $f\colon c \to c'$ in $\mathscr{C}$ such that $F(f)(x) = y$.
   - The composition is defined as in $\mathscr{C}$ and the identities are as in $\mathscr{C}$, that is $\mathbf{1}_{(c,x)} = \mathbf{1}_c$.

   The first projection functor is then defined by $(c,x) \mapsto c, f \mapsto f$. For each functor $F\colon \mathscr{C} \to \mathscr{D}$ the functor $\sum_{\mathscr{C}} F\colon \sum_{\mathscr{C}} S \circ F \to \sum_{\mathscr{D}} S$ is defined by

$$\sum_{\mathscr{C}} F(c,x) = \big(F(c),x\big) \text{ and } \sum_{\mathscr{C}} F(f) = f.$$

3. A *type category* in the sense of Pitts [51] $\mathscr{C}$ is a category with terminal object $\mathbb{1}$ and pullbacks equipped with the following additional structure:

   a) For each object $c \in \mathscr{C}$ a collection $\mathrm{type}_{\mathscr{C}}(c)$ of c-indexed types in $\mathscr{C}$.

   b) For each object $c \in \mathscr{C}$ operations assigning to each c-indexed type λ an object $\sum_c \lambda$ called the *total object* of λ together with a morphism $\mathsf{pr}_1^{\lambda}\colon \sum_c \lambda \to c$ called the *projection-morphism* of λ.

   c) For each morphism $f\colon c \to c'$ in $\mathscr{C}$ an operation assigning to each c'-indexed type λ a c-indexed type $f^*\lambda$ called the pullback of λ along c, together with a morphism $\Sigma f\colon \sum_{c'} f^*\lambda \to \sum_c \lambda$ making the following a pullback square in $\mathscr{C}$:

$$\begin{tikzcd}
\sum_{c'} f^*\lambda \arrow[r, "\Sigma f"] \arrow[d, "\mathsf{pr}_1^{f^*\lambda}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_c \lambda \arrow[d, "\mathsf{pr}_1^{\lambda}"] \\
c \arrow[r, "f"] & c'
\end{tikzcd}$$

   The following strictness conditions are imposed on these operations:

$$\begin{aligned}
1_c^*\lambda &= \lambda & &\text{and} & \Sigma 1_c &= 1_{\sum_c \lambda},\\
g^*\big(f^*(\lambda)\big) &= (f\circ g)^*\lambda & &\text{and} & (\Sigma f)\circ(\Sigma g) &= \Sigma(f\circ g).
\end{aligned}$$

   Type categories can be translated into (fam.Σ)-categories using the following “dictionary”.

Type categories		*(fam,Σ)-categories*
c-indexed type		family arrow
Total object of λ		Sigma-object of λ
projection-morphism		first-projection arrow
pullback of λ along c	$f^*\lambda$ vs. $\lambda \circ f$	composition operation

3.1.3 Continuing the motivating example 2.3.2. If we return to our motivating example, that is the fam-category obtained by a type theory as in [48] Petrakis endows this category $\mathscr{C}_T$ with Σ-objects as follows: Given $t : A \to \mathcal{U}_i$ we set $\sum_A t := \sum_{x:A} t(x)$, the dependent sum type.

The projection is defined via $(a, b) \mapsto a$.

For given $f : A \to B$ and $t : A \to \mathcal{U}_i$ the term $\sum_t f$ is given by the definiting equation $(a, b) \mapsto (f(a), b)$. (Details on the introduction of functions between dependent sum types can be found in [64, §1.6]).

Definition 3.1.4 ((fam,Σ)-functors). Let $\mathscr{C}, \mathscr{D}$ be (fam, Σ)-categories. A (fam, Σ)-functor $\mathscr{C} \to \mathscr{D}$ consists of a fam-functor $F\colon \mathscr{C} \to \mathscr{D}$ in the sense of definition 2.1.4 subject to the following additional condition

(FΣ_3) For all $a \in \mathscr{C}$ and all $\lambda \in \mathrm{fHom}(a)$ the following equation holds:

$$F\big(\sum_a \lambda\big) = \sum_{F(a)} F_a(\lambda).$$

(FΣ_4) For all $a, b \in \mathscr{C}$ and all $\lambda \in \mathrm{fHom}(a)$, $f\colon b \to a$ the following equation holds:

$$F\big(\sum_\lambda f\big) = \sum_{F_a(\lambda)} F(f).$$

We call the functor a *weak* (fam,Σ)-functor if instead we have the following:

(wFΣ_3) For all $a \in \mathscr{C}$ and all $\lambda \in \mathrm{fHom}(a)$ we have an isomorphism

$$F^{\cong}_{\lambda}\colon F\big(\sum_a \lambda\big) \xrightarrow{\cong} \sum_{F(a)} F_a(\lambda)$$

such that $\mathsf{pr}_1^{F_a(\lambda)} \circ F^{\cong}_{\lambda} = F(\mathsf{pr}_1^{\lambda})$.

(wFΣ_4) For all $a, b \in \mathscr{C}$ and all $\lambda \in \mathrm{fHom}(a)$, $f\colon b \to a$ the diagram

$$\begin{tikzcd}
F\big(\sum_b \lambda \circ f\big) \arrow[rrr, "F(\sum_\lambda f)"] \arrow[drr, "F^{\cong}_{\lambda\circ f}"'] \arrow[ddr, "F(\mathsf{pr}_1^{\lambda\circ f})"'] & & & F\big(\sum_b \lambda\big) \arrow[drr, "F^{\cong}_{\lambda}"] \arrow[ddr, "F(\mathsf{pr}_1^{\lambda})"'] & & \\
& & \sum_{F(b)} F_b(\lambda \circ f) \arrow[rrr, "\sum_{F_a(\lambda)} F(f)"] \arrow[dl, "\mathsf{pr}_1^{F_b(\lambda\circ f)}"] & & & \sum_{F(a)} F_a(\lambda) \arrow[dl, "\mathsf{pr}_1^{F_a(\lambda)}"] \\
& F(b) \arrow[rrr, "F(f)"] & & & F(a) &
\end{tikzcd}$$

commutes.

Definition 3.1.5 ((fam, Σ)-natural transformations). Let $\mathscr{C}, \mathscr{D}$ be (fam, Σ)-categories, $F, G\colon \mathscr{C} \to \mathscr{D}$ be (fam, Σ)-functors. Then a *(fam, Σ)-natural transformation* $F \Rightarrow G$ is a fam-natural transformation of the underlying fam-functors satisfying the following conditions:

(FΣ_5) For all $c \in \mathscr{C}$ and all $\lambda \in \mathrm{fHom}(c)$ we have $\eta_{\sum_c \lambda} = \sum_{F_c(\lambda)} \eta_c$.

In case F, G are weak fam-functors we define a *weak (fam,Σ)-natural transformation* to be a fam-natural transformation $\eta\colon F \Rightarrow G$ such that:

(wFΣ_5) For all $c \in \mathscr{C}$ and all $\lambda \in \mathrm{fHom}(c)$ we are given an isomorphism $\eta_\lambda\colon \sum_{F(c)} F_c(\lambda) \to \sum_{G(c)} G(\lambda)$ such that the following diagram commutes:

$$\begin{tikzcd}
F\left(\sum_c \lambda\right) \arrow[r, "\eta_{\sum_c \lambda}"] \arrow[d, "F_\lambda^{\cong}"] & G\left(\sum_c \lambda\right) \arrow[d, "G_\lambda^{\cong}"] \\
\sum_{F(c)} F_c(\lambda) \arrow[r, "\sum_{G_c(\lambda)} \eta_c"] & \sum_{G(c)} G_c(\lambda).
\end{tikzcd}$$

Definition 3.1.6 (Category of (fam,Σ)-categories). (see 3.1.5) The 2-category $(\mathsf{fam}, \Sigma)\mathsf{C}$ has

- *objects:* (fam,Σ)-categories
- *1-maps:* (fam,Σ)-functors between those categories and
- *2-maps:* (fam,Σ)-natural transformations.

Definition 3.1.7 ((2-fam,Σ)-categories – [20]). A 2-fam-category $\mathscr{C}$ is a (2-fam,Σ)-category if the underlying category is a (fam,Σ)-category and the following data is given: for each $c \in \mathscr{C}, \lambda, \mu \in \mathrm{fHom}(c)$ and $\eta\colon \lambda \Rightarrow \mu$ an arrow $\sum_{\lambda,\mu} \eta\colon \sum_c \lambda \to \sum_c \mu$. The following conditions must hold:

(2FΣ_1) For all $c, c' \in \mathscr{C}, f\colon c' \to c$ with $\lambda, \mu \in \mathrm{fHom}(c), \eta\colon \lambda \Rightarrow \mu$ the diagram

$$\begin{tikzcd}
\sum_c \lambda \circ f \arrow[r, "\sum_\lambda f"] \arrow[d, "\sum_{\lambda\circ f, \mu\circ f} \eta \bullet f"'] \arrow[dd, bend right=60, "\mathsf{pr}_1^{c,\lambda\circ f}"'] & \sum_{c'} \lambda \arrow[d, "\sum_{\lambda,\mu} f"] \arrow[dd, bend left=60, "\mathsf{pr}_1^{c',\lambda}"] \\
\sum_c \mu \circ f \arrow[r, "\sum_\mu f"'] \arrow[d, "\mathsf{pr}_1^{c,\mu\circ f}"'] & \sum_{c'} \mu \arrow[d, "\mathsf{pr}_1^{c',\mu}"] \\
c \arrow[r, "f"'] & c'
\end{tikzcd}$$

must commute.

(2FΣ_2) For all $c \in \mathscr{C}$ and $\lambda, \mu, \nu \in \mathrm{fHom}(c)$ and $\eta\colon \lambda \Rightarrow \mu, \eta'\colon \mu \Rightarrow \nu$ the equations

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

hold.

3.1.8 Further continuation of the motivating example 3.1.3. Ehrhardt has checked that the Σ-structure on $\mathscr{C}_T$ given above extends to the 2-fam-structure we gave earlier if one sets

$$\sum_{t,s} \eta := \lambda x.\lambda y.\big(x.\eta(x)(y)\big)$$

for $t, s \in \mathrm{fHom}(A)$ and $\eta \in \mathrm{fHom}(t, s)$.

Remark 3.1.9. Usually, we omit the indices in $\bullet_{c,c'}$ and only write $\bullet$. The operation $\bullet$ will be called “horizontal composition” from now on.

Examples 3.1.10. 1. If M is a monoid, and $\mathscr{C}$ is a fam-category, we can define a 2-fam-structure on $\mathscr{C}$ by letting $\mathrm{Hom}(\lambda, \mu) := M$ for $\lambda, \mu \in \mathrm{fHom}(c)$ and $c \in \mathscr{C}_0$. The compositions $\bullet$ are defined by the rule $m \bullet f := m$. The compatibility and distributivity properties are immediate to show.

2. All fam-categories in Examples 2.1.2 have their 2-analogue (see [20, §3.1]).

Definition 3.1.11 ((2-fam,Σ)-functors). Let $\mathscr{C}, \mathscr{D}$ be (2-fam,Σ)-categories. A *(2-fam,Σ)-functor* from $\mathscr{C}$ to $\mathscr{D}$ is a 2-fam functor and the following additional conditions hold:

(2FΣ_3) F is a (fam, Σ)-functor of the underlying (fam, Σ)-categories.

(2FΣ_4) For all $c \in \mathscr{C}, \lambda, \mu \in \mathrm{fHom}(c), \eta \colon \lambda \Rightarrow \mu$ the following rectangle commutes:

$$F\big(\sum_{\lambda,\mu} \eta\big) = \sum_{F(\lambda),F(\mu)} F^{\lambda,\mu}(\eta).$$

We call F a *weak* (2-fam,Σ)-functor if

(w2FΣ_3) F is a *weak* (fam,Σ)-functor of the underlying (fam,Σ)-categories, that is we only are given an isomorphism $F^{\cong}_{\lambda} \colon \sum_{F(c)} F_c(\lambda) \to F\Big(\sum_c \lambda\Big)$ instead of an equality.

(w2FΣ_4) For all $c \in \mathscr{C}, \lambda, \mu \in \mathrm{fHom}(c), \eta \colon \lambda \Rightarrow \mu$ we have that the squares

$$\begin{tikzcd}
\sum_{F(c)} F_c(\lambda) \arrow[r, "F^{\cong}_{\lambda}"] \arrow[d, "\sum_{F_c(\lambda),F_c(\mu)} F_{\lambda,\mu}(\eta)"'] & F\Big(\sum_c \lambda\Big) \arrow[d, "F\Big(\sum_{\lambda,\mu}(\eta)\Big)"] \\
\sum_{F(c)} F_c(\mu) \arrow[r, "F^{\cong}_{\mu}"] & F\Big(\sum_c \mu\Big)
\end{tikzcd}$$

commute.

Definition 3.1.12 ((2-fam, Σ)-natural transformation). Let $\mathscr{C}, \mathscr{D}$ be two (2-fam, Σ)-categories and $F, G \colon \mathscr{C} \to \mathscr{D}$ be two (2-fam, Σ)-functors. A *(2-fam,Σ)-natural transformation* $\eta \colon F \Rightarrow G$ is simply a 2-fam natural transformation $F \Rightarrow G$ that is also a (fam,Σ)-natural transformation of the underlying (fam,Σ)-categories.

Analogously we define a weak *(2-fam,Σ)-natural transformation* $\eta \colon F \Rightarrow G$ to be a 2-fam natural transformation $F \Rightarrow G$ that is a weak (fam,Σ)-natural transformation of the underlying (fam,Σ)-categories.

Definition 3.1.13 (Categories of (2-fam,Σ)-categories). The category $(2\text{-}\mathsf{fam}, \Sigma)\mathsf{C}^{\mathrm{wk}}$ is defined through the following data:

- *Objects* are (2-fam,Σ)-categories,
- *1-maps* are weak (2-fam,Σ)-functors and whose
- *2-maps* are (2-fam,Σ)-natural transformations.

We also get the subcategory $(2\text{-}\mathsf{fam}, \Sigma)\mathsf{C}$ with the same objects and 2-maps, but with only strict (2-fam,Σ)-functors.

3.2 Comprehension categories

Definition 3.2.1 (Comprehension categories – [35]). A *comprehension category* consists of a functor $P \colon \mathscr{E} \to \mathscr{B}^{\to}$ for categories $\mathscr{E}, \mathscr{B}$ (where $\mathscr{B}^{\to}$ is the arrow category) such that

(CC$_1$) The functor $p := \mathrm{cod} \circ P \colon \mathscr{E} \to \mathscr{B}$ is a Grothendieck fibration

$$\begin{tikzcd}
\mathscr{E} \arrow[rr, "P"] \arrow[dr, "p"'] & & \mathscr{B}^{\to} \arrow[dl, "\mathrm{cod}"] \\
& \mathscr{B} &
\end{tikzcd}$$

(CC$_2$) P maps arrows cartesian with respect to $P_0 := \mathrm{dom} \circ P$ to arrows cartesian with respect to cod.

A comprehension category is *split* if the fibration p admits a splitting and *discrete* if the fibration p is discrete.

Definition 3.2.2 (Pseudo-maps of comprehension categories). Let $P \colon \mathscr{E} \to \mathscr{B}^{\to}$ and $P' \colon \mathscr{E}' \to \mathscr{B}'^{\to}$ be comprehension categories. A *pseudo-map* from P to P' is given by a map $(F, G) \colon p \to p'$ of Grothendieck fibrations together with a natural isomorphism $\phi \colon P' \circ F \Rightarrow G^{\to} \circ P$ (where $G^{\to}$ is the lift of G to the arrow categories) lying over the identity transformation of F. This data can be visualised as

$$\begin{tikzcd}
\mathscr{E} \arrow[rrr, "F"] \arrow[drr, "P"] \arrow[ddr, "p"'] & & & \mathscr{E}' \arrow[drr, "P'"] \arrow[dd, "p'"'] \arrow[dl, Rightarrow, "\sim"', "\phi"] & & \\
& & \mathscr{B}^{\to} \arrow[rrr, "G^{\to}"'] \arrow[dl, "\mathrm{cod}"] & & & \mathscr{B}'^{\to} \arrow[dll, "\mathrm{cod}"] \\
& \mathscr{B} \arrow[rrr, "G"] & & \mathscr{B}' & &
\end{tikzcd}
\quad \text{and} \quad
\begin{tikzcd}
G\big(P_0(e)\big) \arrow[rr, "\phi_e"] \arrow[dr, "P(e)"'] & & P'_0\big(F(e)\big) \arrow[dl, "F(P'(e))"] \\
& G(e). &
\end{tikzcd}$$

(Technically the transformation ϕ would have two components, but as it lies over the identity transformation on G the second component is always the identity, so we identify ϕ_c with its first component.) If the transformation η is the identity we speak of a *strict* map. If the underlying comprehension categories are split with splittings $S_p, S_{p'}$ respectively and the (strict/pseudo-)map of comprehension categories preserves the splitting then we call those (strict/pseudo-)maps *split*.

3.2.3 Some historical remarks. Comprehension categories were introduced by Jacobs in [34] in this manuscript the strict morphisms were already introduced, as is mentioned in [33, p. 55]. However in the published version [35] these morphisms are no longer mentioned. Jacobs student Blanco then compared the category of comprehension categories and strict morphisms with other categories for dependent types in [5]. In his thesis Jacobs also introduced pseudo-maps of fibrations [33, 4.1.4 Definition], so they are not due to [16] as claimed in [1].

Later Jacobs linked comprehension categories with weakening and contraction comonads in [32], a relation that is further elaborated on in [14]. In the latter paper the authors also introduce the notion of a *lax* map of comprehension categories [14, 3.4 Definition], which we will not consider here, in these the natural isomorphism η need not be invertible.

3.2.4 Continuing the categories from contexts of 2.4.4. Continuing the motivating example the functor P then takes $[\Gamma \vdash \sigma \ \mathit{type}]$ to the projection

$$[\Gamma, x : \sigma] \mapsto [\Gamma], \quad \langle [\mathbf{M}], [N] \rangle \mapsto \langle [\mathbf{M}] \rangle.$$

One can add 2-arrows between pseudomaps of comprehension categories to obtain a 2-category ComprC of comprehension categories, pseudomaps and transformations.

Definition 3.2.5 (Transformations of pseudo-maps). A transformation $(F, G, \eta) \Rightarrow (F'.G', \eta')$ consists of a 2-map of fibrations $(\tilde{\alpha}, \alpha)\colon (F, G) \to (F', G')$ in the sense of definition 2.4.10 such that for every $A \in \mathscr{E}$ we have

$$\eta'_\lambda \circ P'(\tilde{\alpha}_\lambda) = \alpha_{P(\lambda)} \circ \eta_\lambda$$

for all $\lambda \in \mathscr{E}$.

Remark 3.2.6. When they were introduced by Coraglia and Emmenegger in [14] the condition on transformations on pseudo-maps instead read as

$$\begin{tikzcd}
\mathscr{E} \arrow[rr, bend left, "F", ""{name=U, below}] \arrow[rr, bend right, "F'"', ""{name=D}] \arrow[d, "P"] & & \mathscr{E}' \arrow[d, "P'"] & = & \mathscr{E} \arrow[rr, bend left, "F", ""{name=U2, below}] \arrow[d, "P"] & & \mathscr{E}' \arrow[d, "P'"] \\
\mathscr{B}^{\to} \arrow[rr, bend right, "G'^{\to}"', ""{name=L}] & & \mathscr{B}'^{\to} & & \mathscr{B}^{\to} \arrow[rr, bend left, "G^{\to}", ""{name=M, below}] \arrow[rr, bend right, "G'^{\to}"', ""{name=N}] & & \mathscr{B}'^{\to}
\arrow[Rightarrow, from=U, to=D, "\tilde{\alpha}"]
\arrow[Rightarrow, from=D, to=L, "\eta'"]
\arrow[Rightarrow, from=U2, to=M, "\eta"]
\arrow[Rightarrow, from=M, to=N, "\alpha^{\to}"]
\end{tikzcd}$$

that is $\eta' \circ (P' \bullet \tilde{\alpha}) = (\alpha^{\to} \bullet P) \circ \eta\colon P' \circ F \Rightarrow G'^{\to} \circ P$. However as both η and η' lie over 1_G and $1_{G'}$ respectively we obtain that in the diagrams

$$\begin{tikzcd}
P_0'\big(F\lambda)\big) \arrow[r, "P'(\tilde{\alpha}_\lambda)"] \arrow[d, "P'(F\lambda)"] & P_0'\big(F'(\lambda)\big) \arrow[r, "\eta'_\lambda"] \arrow[d, "P'(F'(\lambda))"] & G'^{\to}\big(P_0(\lambda)\big) \arrow[d, "G(P(\lambda))"] \\
p(\big(F(\lambda)\big) \arrow[r, "p(\tilde{\alpha}_\lambda)"] & p\big(F'(\lambda)\big) \arrow[r, "1_{p(F'(\lambda))}"] & p\big(F'(\lambda)\big)
\end{tikzcd}$$

$$\begin{tikzcd}
P_0'\big(F(\lambda)\big) \arrow[r, "\eta_\lambda"] \arrow[d, "P'(F(\lambda))"] & G^{\to}\big(P_0(\lambda)\big) \arrow[r, "\alpha^{\to}_{P(\lambda)}"] \arrow[d, "G^{\to}(P(\lambda))"] & G'^{\to}\big(P_0(\lambda)\big) \arrow[d, "G'^{\to}(P(\lambda))"] \\
p\big(F(\lambda)\big) \arrow[r, "1_{p(F(\lambda))}"] & p\big(F(\lambda)\big) \arrow[r, "p(\alpha_\lambda)"] & p\big(F'(\lambda)\big)
\end{tikzcd}$$

the compositions in the upper and lower row must coincide. However for the lower row this is always the case so only the condition

$$\eta'_\lambda \circ P'(\tilde{\alpha}_\lambda) = \alpha_{P(\lambda)} \circ \eta_\lambda$$

for $\lambda \in \mathscr{E}$ remains.

Examples 3.2.7. 1. The most trivial example is where P is simply the identity on $\mathscr{B}^{\to}$. It is immediate that this is constitutes a comprehension category, as the arrows in $\mathscr{B}^{\to}$ cartesian with respect to cod are simply pullback squares

$$\begin{tikzcd}
f^{-1}(b) \arrow[r, "g^*f"] \arrow[d, "f^*g"'] \arrow[dr, phantom, "\lrcorner", very near start] & a \arrow[d, "f"] \\
b \arrow[r, "g"] & c
\end{tikzcd}$$

in $\mathscr{C}$.

2. Another (class of) example(s) is given by another notion introduced to model dependent types, namely *display map categories* due to Taylor (cf. [62]). These are categories $\mathscr{C}$ together with a collection D of morphisms in $\mathscr{C}$. This collection is supposed to be closed

in the following sense: given an arrow $d\colon a \to c$ in D and an arbitrary arrow $f\colon b \to c$, then the pullback

$$\begin{tikzcd} f^{-1}(a) \arrow[r, "f^*d"] \arrow[d, "d^*f"] \arrow[dr, phantom, "\lrcorner", very near start] & a \arrow[d, "d"'] \\ b \arrow[r, "f"] & c \end{tikzcd}$$

exists in $\mathscr{C}$ and d^*f is in D. If we then consider the category $\mathscr{C}^{\to}(D)$ with objects from D, then the inclusion $\mathscr{C}^{\to}(D) \hookrightarrow \mathscr{C}^{\to}$ constitutes a comprehension category.

3. A *category with attributes* (cf. [9, §3.2]) is a category $\mathscr{C}$ together with
   - a presheaf $\mathrm{Ty}\colon \mathscr{C}^{\mathrm{op}} \to \mathsf{Sets}$
   - a functor $(-,-)\colon \sum_{\mathscr{C}} \mathrm{Ty} \to \mathscr{C}$
   - a natural transformation $p\colon (-,-) \to \mathsf{pr}_1^{\mathrm{Ty}}$

   such that the naturality square of p are pullbacks,

$$\begin{tikzcd} (c', f^{-1}(a)) \arrow[r, "{(f,a)}"] \arrow[d, "p_{f^{-1}(a)}"'] \arrow[dr, phantom, "\lrcorner", very near start] & (c,a) \arrow[d, "p_a"] \\ c' \arrow[r, "f"] & c. \end{tikzcd}$$

   This can be turned into a comprehension category $P\colon \mathscr{C} \to \mathscr{B}^{\to}$ such that p is a discrete fibration in the following way: we consider

$$\begin{tikzcd} \sum_{\mathscr{C}} \mathrm{Ty} \arrow[rr, "P"] \arrow[dr, "\mathsf{pr}_1^{\mathrm{Ty}}"'] & & \mathscr{C}^{\to} \arrow[dl, "\mathrm{cod}"] \\ & \mathscr{C} & \end{tikzcd}$$

   where P sends $(c,a) \in \sum_{\mathscr{C}} \mathrm{Ty}$ to $p_{(c,a)}\colon (c,a) \to c$. By the prerequisite this P sends cartesian arrows to cartesian squares and thus is a comprehension category.

   Blanco showed in [5, Theorem 2.3] that if we consider $\mathsf{ComprC}_{\mathrm{disc}}$, the subcategory of ComprC where the p are discrete fibrations then this category is equivalent to CwA, the category of categories with attributes. This 1-categorical result immediately extends to the 2-categorical framework by chosing the appropriate 2-arrows between categories with attributes. As Cartmell does not give such 2-arrows we simply take those obtained from the translation to comprehension categories so that the 1-equivalence extends to a 2-equivalence $\mathsf{CwA} \simeq \mathsf{CompMod}_{\mathrm{disc}}$.

Definition 3.2.8 (Categories of comprehension categories). The 2-category ComprC consists of the following data:

- Its *objects* are comprehension categories.
- Its *1-maps* are pseudo-maps of comprehension categories.
- Its *2-maps* are transformations of pseudo-maps.

It has the following subcategories:

- $\mathsf{ComprC}_{\mathrm{spl}}$, this category has as object the split comprehension categories and as 1-maps the pseudo-maps that respect those splittings,
- $\mathsf{ComprC}^{\mathrm{str}}$ which has as 1-maps only the strict maps of comprehension categories,
- $\mathsf{ComprC}_{\mathrm{disc}}$ which has as objects the discrete comprehension categories.

Obviously one can combine these above modifiers to for example obtain $\mathsf{ComprC}^{\mathrm{str}}_{\mathrm{disc}}$ etcetera.

Theorem 3.2.9. *There is a 2-equivalence*

$$\mathsf{ComprC}_{\mathrm{spl}} \simeq (\mathsf{2\text{-}fam}, \Sigma)\mathsf{C}^{\mathrm{wk}}\,.$$

We divide the proof of this theorem into three parts (lemmata):

1. Firstly we show that there exists a 2-functor $\mathsf{ComprC}_{\mathrm{spl}} \to (\mathsf{2\text{-}fam}, \Sigma)\mathsf{C}^{\mathrm{wk}}$.
2. Then we show that there exists a 2-functor in the reverse direction.
3. Lastly we show that these functors constitute a 2-equivalence.

Lemma 3.2.10. *There is a 2-functor* $\mathsf{ComprC}_{\mathrm{spl}} \to (\mathsf{2\text{-}fam}, \Sigma)\mathsf{C}^{\mathrm{wk}}$.

Proof: This functor takes a comprehension category $P\colon \mathscr{E} \to \mathscr{B}^{\to}$ to the (2-fam,Σ)-category defined by the following datum:

- The 2-fam-category is given as in the proof of Theorem 2.4.14
- The Σ-*structure* is given by
$$\Sigma_b(\lambda) := P_0(\lambda)$$
and the projection arrow is given by $P(\lambda)$. Given $f\colon b \to b'$ and $\lambda \in \mathrm{fHom}(b')$, that is $p(\lambda) = b'$, the arrow $\Sigma_c(f)\colon \sum_{b'} c \circ_b f \to \sum_b c$ is given by
$$P_0\big(\overline{f}(\lambda)\big)\colon P_0\big(f^*(\lambda)\big) \to P_0(\lambda).$$
Given an arrow $g\colon \lambda \to \lambda'$ in $\mathrm{fHom}(b)$ the arrow $\sum_{\lambda,\lambda'} g$ is given as
$$P_0(g).$$

It is immediate that this constitutes a (2-fam,Σ)-category. On the functors we do the following: a pseudo-map $(F, G, \phi)\colon (P\colon \mathscr{C} \to \mathscr{B}^{\to}) \to (P\colon \mathscr{C}' \to \mathscr{B}'^{\to})$ of comprehension categories is mapped to the weak (2-fam,Σ)-functor defined as follows:

- The underlying 2-fam-functor H is given as in the proof of Theorem 2.4.14 using (F, G), that is $H(c) := G(c)$ and $H_c(\lambda) := F(\lambda)$.
- For every $\lambda \in \mathrm{fHom}(b)$ for an arbitrary $b \in \mathscr{B}$ the arrow $H^{\cong}_{\lambda}$ is defined as $\phi_\lambda\colon G\big(P_0(c)\big) \to P_0'\big(F(c)\big)$.

One can check that this fulfils the conditions of a weak (2-fam,Σ)-functor as the underlying functor is a (fam,Σ)-functor, which follows by the following argument: we have to show that for all $b \in \mathscr{B}, f\colon a \to b, \lambda, \mu \in \mathrm{fHom}(b), \eta\colon \lambda \Rightarrow \mu$ we have the commutativity of

$$\begin{tikzcd}
\sum_{H(a)} H_a(\lambda \circ f) \arrow[r, "H^{\cong}_{\lambda}"] \arrow[d, "\sum_{H_a(\lambda)} H_(f)"'] & H\Big(\sum_b \lambda\Big) \arrow[d, "H\big(\sum_\lambda f\big)"] \\
\sum_{H(b)} H_b(\lambda) \arrow[r, "H^{\cong}_{\mu}"] & H\Big(\sum_b \lambda\Big)
\end{tikzcd}
\quad \text{and} \quad
\begin{tikzcd}
\sum_{H(b)} H_b(\lambda) \arrow[r, "H^{\cong}_{\lambda}"] \arrow[d, "\sum_{H_c(\lambda), H_c(\mu)} H_{\lambda,\mu}(\eta)"] & H\big(\sum_b \lambda\big) \arrow[d, "H(\sum_{\lambda,\mu} \eta)"] \\
\sum_{H(b)} H_b(\mu) \arrow[r, "H^{\cong}_{\mu}"] & H\big(\sum_b \mu\big)\big).
\end{tikzcd}$$

The commutativity of the first square follows from ϕ being a natural iso $\phi\colon G^{\to} \circ P \to P' \circ F$ over the identity on G, as this yields the commutative diagram

$$\begin{tikzcd}
G\big(P_0(\lambda)\big) \arrow[rr, "G(P_0(f))"] \arrow[dr, "\phi_\lambda"] \arrow[ddr, "G(P(\lambda))"'] & & G\big(P_0(\lambda \circ f)\big) \arrow[dr, "\phi_{\lambda \circ f}"] \arrow[ddr, crossing over, "G(P(\lambda \circ f))"' near end] & \\
& P_0'\big(F(\lambda)\big) \arrow[rr, "P(F(\lambda))"] \arrow[d, "P(F(\lambda))"] & & P_0'(\big(F(\lambda \circ f)\big) \arrow[d, "P'(F(\lambda \circ f))"] \\
& G(b) \arrow[rr, "1_{G_b}"] & & G(b)
\end{tikzcd}$$

Using the definition of the Σ-objects we see that the top square is the desired one.

The commutativity of the second square follows in the same fashion from the commutative diagram

$$\begin{tikzcd}
G\big(P_0(\lambda)\big) \arrow[rr, "G(P_0(\eta))"] \arrow[dr, "\phi_\lambda"] \arrow[ddr, "G(P(\lambda))"'] & & G\big(P_0(\mu)\big) \arrow[drr, "\phi_\mu"] \arrow[ddr, "G(P(\mu))"'] & & \\
& P'_0\big(F(\lambda)\big) \arrow[rrr, "P(F(\lambda))"] \arrow[d, "P(F(\lambda))"] & & & P'_0\big((F(\mu)\big) \arrow[dl, "P'(F(\mu))"] \\
& G(b) \arrow[rr, "1_{G_b}"] & & G(b). &
\end{tikzcd}$$

Finally, given a transformation of pseudo-maps $(\tilde{\alpha}, \alpha)\colon (F, G, \eta) \Rightarrow (F', G', \eta')$ we get the weak (2-fam,Σ)-natural transformation $\alpha\colon H \Rightarrow H'$. As in the proof of proposition 2.4.14 it follows that this α is a 2-fam-natural transformation. It remains to show that this is also a weak (fam,Σ)-natural transformation, that is to show the commutativity of

$$\begin{tikzcd}
H\big(\sum_c \lambda\big) \arrow[r, "\alpha_{\sum_c \lambda}"] \arrow[d, "H^{\cong}_\lambda"] & H'\big(\sum_c \lambda\big) \arrow[d, "H'^{\cong}_\lambda"] \\
\sum_{H(c)} H_c(\lambda) \arrow[r, "\sum_{H_c(\lambda)} \alpha_c"] & \sum_{H'(c)} H'_c(\lambda).
\end{tikzcd}$$

Filling in the definitions of H and H' and the Σ we have to show that commutativity of

$$\begin{tikzcd}
F\big(P_0(\lambda)\big) \arrow[r, "\alpha_{P_0(c)}"] \arrow[d, "\eta_\lambda"] & F'\big(P_0(\lambda)\big) \arrow[d, "\eta'_\lambda"] \\
P'_0\big(F(\lambda)\big) \arrow[r, "P'(\alpha_c)"] & P'_0\big(F'(\lambda)\big).
\end{tikzcd}$$

But this commutes due to α being a transformation of pseudo-maps. Q.E.D.

Lemma 3.2.11. *There is a 2-functor* $(\mathsf{2\text{-}fam}, \Sigma)\mathsf{C}^{\mathrm{wk}} \to \mathsf{ComprC}_{\mathrm{spl}}$.

Before we prove this lemma, we make a few quick observations regarding the split fibration that we associated to a 2-fam-category in 2.4.14, namely the cartesian morphisms. In the definition of comprehension categories it is required that the functor P takes cartesian morphisms to cartesian squares, so we need to know what the cartesian morphisms with respect to the split fibration $p_{\mathscr{C}}$ stemming from a 2-fam-category look like.

Lemma 3.2.12. *Given a 2-fam-category $\mathscr{C}$, the arrows cartesian with respect to $p_{\mathscr{C}}$ and some morphism $\mathscr{C}$ and some λ are the morphisms $(f, \eta) \in \mathscr{F}$ such that η is an isomorphism.*

Proof: We first show that any such (f, η) is indeed cartesian with respect to $p_{\mathscr{C}}$. So assume $(f, \eta)\colon (c, \lambda) \to (c', \mu)$ is cartesian with respect to f and μ. As $(f, 1_{\mu\circ f})\colon (c, \lambda) \to (c', \mu)$ fulfils $p_{\mathscr{C}}(f, 1_{\mu\circ f}) = f = p_{\mathscr{C}}(f, \eta)$ there must be a $(h, \xi)\colon (c, \mu \circ f) \to (c, \lambda)$ such that $(f, 1_{\mu\circ f} = (f, \eta) \circ (h, \xi)$ — as (f, η) is $p_{\mathscr{C}}$-cartesian with respect to f and $c', \mu)$ and also a $(h', \xi')\colon (c, \lambda) \to (c, \mu \circ f)$ such that $(f, \eta) = (f, 1_{\mu\circ f}) \circ (h', \xi')$ as $(f, 1_{\mu\circ f})$ is $p_{\mathscr{C}}$-cartesian with respect to f and (c', μ). Thus we have the commutative diagram

$$\begin{tikzcd}
(c, \mu \circ f) \arrow[d, shift left, "(h,\xi)"] \arrow[dr, "(f,1_{\mu\circ f})"] & \\
(c, \lambda) \arrow[u, shift left, "(h',\xi')"] \arrow[r, "(f,\eta)"] & (c', \mu).
\end{tikzcd}$$

With a standard argument stemming from the uniquness we get that

$$(h', \xi') \circ (h, \xi) = 1_{(c,\mu \circ f)} \quad \text{and} \quad (h, \xi) \circ (h', \xi') = 1_{(c,\lambda)}.$$

Additionally we can compute from the commutativity of the triangles that both $f \circ h' = f = f \circ h$ and $(1_{\mu \circ f} \bullet h') \circ \xi' = \eta$ and thus $\xi' = \eta$. This helps in the computation as

$$(h, \xi) \circ (h', \xi') = \big(h \circ h', (\xi \bullet h') \circ \xi'\big)$$

and thus $(\xi \bullet h') \circ \xi' = 1_{(c,\lambda)}$ and $h \circ h' = 1_c$. Conversely we compute $(\xi' \bullet h) \circ \xi = 1_{(c,\mu \circ f)}$ and $h' \circ h = 1_{(c,\mu \circ f)}$ from the composition $(h', \xi') \circ (h, \xi)$. So we immediately get $(\xi \bullet h') \circ \eta = 1_{(c,\lambda)}$ as $\xi' = \eta$ from earlier computations and

$$\begin{aligned}
\eta \circ (\xi \bullet h) &= \xi' \circ (\xi \bullet h') \\
&= \big(\xi' \bullet (h \circ h')\big) \circ (\xi \bullet h') \\
&= \big((\xi' \bullet h) \bullet h'\big) \circ (\xi \bullet h') \\
&= \big((\xi' \bullet h) \circ \xi\big) \bullet h' \\
&= 1_{(c,\mu \circ f)} \bullet h' \\
&= 1_{(c,\lambda)}.
\end{aligned}$$

Thus η is an isomorphism with inverse $\xi \bullet h$.

If $(f, \eta)\colon (c, \lambda) \to (c', \mu)$ is given where $\eta\colon \lambda \Rightarrow \mu \circ f$ is an isomorpism we immediately see that (f, η) is $p_{\mathscr{C}}$-cartesian with respect to f and (c', μ). Namely, given $(g, \beta)\colon (d, \nu) \to (c', \mu)$ such that there exists $h\colon d \to c$ with $f \circ h = g$ we define $(h, (\eta^{-1} \bullet h) \circ \beta)\colon (d, \nu) \to (c, \lambda)$. This is well-defined as $\beta\colon \nu \to \mu \circ g$ and $\eta^{-1} \bullet h\colon \lambda\colon \mu \circ (f \circ h) \Rightarrow \lambda$ and $\mu \circ (f \circ h) = \mu \circ g$.

A simple computation then yields

$$(f, \eta) \circ \big(h, (\eta^{-1} \bullet h) \circ \beta\big) = \big(g, (\eta \bullet h) \circ (\eta^{-1} \bullet h) \circ \beta\big) = \big(g, (\eta \circ \eta^{-1}) \bullet h \circ \beta\big) = (g, \beta).$$

This $(h, (\eta^{-1} \bullet h) \circ \beta)$ is obviously unique with this property. Q.E.D.

Proof of 3.2.11: We map a (2-fam,Σ)-category $\mathscr{B}$ to the comprehension category $P\colon \mathscr{C} \to \mathscr{B}^{\to}$ given by the following data:

- The categories $\mathscr{E}, \mathscr{B}$ and the functor $p\colon \mathscr{E} \to \mathscr{B}$ are defined as in the proof of 2.4.14.
- The functor $P\colon \mathscr{C} \to \mathscr{B}^{\to}$ takes $(b, \lambda) \in \mathscr{C}$ to the arrow $\mathsf{pr}_1^\lambda\colon \sum_b \lambda \to b$ and $(f, \eta)\colon (c, \lambda) \to (d, \mu)$ to $\sum_\mu f \circ \sum_{\lambda,\mu \circ f} \eta$. This can be visualized as

$$\begin{tikzcd}
\sum_c \lambda \arrow[r, "\sum_{\lambda,\mu\circ f}\eta"] \arrow[d, "\mathsf{pr}_1^\lambda"] & \sum_c \mu \circ f \arrow[r, "\sum_\mu f"] \arrow[d, "\mathsf{pr}_1^{\lambda' \circ f}"] & \sum_d \mu \arrow[d, "\mathsf{pr}_1^\mu"] \\
c \arrow[r, "1_c"'] & c \arrow[r, "f"] & d.
\end{tikzcd}$$

It is again straightforward to check that this is indeed a comprehension category: Given $(f, \eta)\colon (c, \lambda) \to (d, \mu)$ that is $p_{\mathscr{C}}$-cartesian with respect to f and (d, μ) the resulting square

$$\begin{tikzcd}
\sum_b \lambda \arrow[r, "\sum_\lambda f \circ \sum_{\lambda,\mu\circ f}\eta"] \arrow[d, "\mathsf{pr}_1^\lambda"'] & \sum_d \mu \arrow[d, "\mathsf{pr}_1^\mu"] \\
c \arrow[r, "f"] & d
\end{tikzcd}$$

is cartesian as $\sum_{\lambda,\mu\circ f}\eta$ is an isomorphism. This follows as

$$\sum_{\lambda,\mu\circ f}\eta\circ\sum_{\mu\circ f,\lambda}\eta^{-1}=\sum_{\lambda,\lambda}(\eta\circ\eta^{-1})=\sum_{\lambda,\lambda}1_\lambda=1_{\sum_c\lambda}$$

and similarly $\sum_{\mu\circ f,\lambda}\eta^{-1}\circ\sum_{\lambda,\mu}\eta=1_{\sum_d\mu}$.

We define our splitting to take each f to $(f,1_{\lambda\circ f})$ as defined above. Applying P to $(f,1_{\lambda\circ f}$ yields the usual pullback square obtained from pulling a family arrow back along a morphism.

Now if we are given a weak (2-fam,Σ)-functor $F\colon\mathscr{B}\to\mathscr{B}'$ of (2-fam,Σ)-categories, then our split pseudomap between the associated comprehension categories is defined as $(p_F,F,\phi)\colon(p\colon\mathscr{C}\to\mathscr{B})\to(p'\colon\mathscr{C}'\to\mathscr{B}')$ where F is the given functor and p_F is defined as in the proof of Theorem 2.4.14. The natural isomorphism $\phi\colon P'\circ F\Rightarrow G^{\to}\circ P$ is defined via

$$\begin{tikzcd}
\phi_{(b,\lambda)}:=F^{\cong}_{\lambda}\colon & \sum_{F(c)}F(\lambda) \arrow[r] \arrow[d, equal] & F\Big(\sum_c\lambda\Big) \arrow[d, equal] \\
& P'\big(G(b,\lambda)\big) & G^{\to}\big(P(b,\lambda)\big)
\end{tikzcd}$$

for all $(b,\lambda)\in\mathscr{E}$. To verify that this is a split pseudomap of comprehension categories we need to check the following:

1. The square

$$\begin{tikzcd}
\mathscr{C} \arrow[r, "G"] \arrow[d, "P"] & \mathscr{C}' \arrow[d, "P'"] \\
\mathscr{B}^{\to} \arrow[r, "F^{\to}"'] & \mathscr{B}'^{\to}
\end{tikzcd}$$

commutes up to ϕ and this ϕ is natural.

2. The splittings are preserved, that is $G(f,1_{\lambda\circ f})$ is the splitting of $G(g)$.

That the given square commutes up to ϕ is immediate from the definition of ϕ from $F^{\cong}_{\lambda}$. That ϕ is natural stems from the condition (wFΣ$_4$) The second point is immediate from the definition of the splittings.

Lastly, given a weak (2-fam,Σ)-natural transformation $\eta\colon F\Rightarrow F'$ we map it to a transformation of pseudo-maps in the following way: we consider $(\hat\alpha,\eta)\colon(p_F,F,\phi)\Rightarrow(p_{F'},F',\phi')$ where $\hat\alpha$ is defined via

$$\hat\alpha_{(b,\lambda)}:=(\eta_b,1_{F_c(\lambda)})\colon G(b,\lambda)=\big(F(b),F_b(\lambda)\big)\to\big(F'(b),F_{b'}(\lambda)\big).$$

Here we used that (2fam$_4$), that is $F_b(\lambda)=F'_b(\lambda)\circ\eta_b$. It is immediate that this constitutes a natural transformation $\hat\alpha\colon G\Rightarrow G'$ and by calculation we have that $p'\bullet\hat\alpha=\eta\bullet p'$ as for all $(b,\lambda)\in\mathscr{E}$

$$(p'\bullet\hat\alpha)_{(b,\lambda)}=p'\big(\hat\alpha_{(b,\lambda)}\big)=p'\big((\eta_b,1_{F_b(\lambda)})\big)=\eta_b=\eta_{p(b,\lambda)}=(\eta\bullet p)_{(b,\lambda)}.$$

It remains to check that $(\phi_F)_{(b,\lambda)}\circ P'(\hat\alpha_{(b,\lambda)})=\eta_{P(b)}\circ\phi'_{(b,\lambda)}$. A straightforward computation yields

$$\begin{aligned}
(\phi_F)_{(b,\lambda)}\circ P'(\hat\alpha_{(b,\lambda)}) &:= F^{\cong}_{\lambda}\circ P'(\hat\alpha_{(b,\lambda)}) && \text{(definition of } \phi_F)\\
&= F^{\cong}_{\lambda}\circ P'\big((\eta_b,1_{F_b(\lambda)})\big) && \text{(definition of } \hat\alpha)\\
&= F^{\cong}_{\lambda}\circ\sum_{F'_b(\lambda)}\eta_b && \text{(definition of } P')\\
&= \eta_{\sum_b\lambda}\circ F'^{\cong}_{\lambda} && \text{(as } \eta \text{ weak nat. trafo)}\\
&= \eta_{P(b,\lambda)}\circ\phi'_{(b,\lambda)} && \text{(by the definitions.)}
\end{aligned}$$

Q.E.D.

Lemma 3.2.13. *The two above functors consitute a 2-equivalence.*

Proof: To see the equivalence first note that if we are given a (2-fam,Σ)-category $\mathscr{B}$, applying the two functors equals to applying the identity. Additionally if we are given a weak (2-fam,Σ)-functor $F\colon \mathscr{B} \to \mathscr{B}'$ applying the two functors is again the same as applying the identity, thus the composition of the two given functors equals the identity, and the same for weak 2-fam-natural transformations.

In the reverse direction, assume we are given a comprehension category $P\colon \mathscr{C} \to \mathscr{B}^{\to}$. Applying the functors yields the comprehension category given as follows:

- The underlying category $\mathscr{D}$ is the category $\mathscr{B}$.
- The category $\mathscr{E}$ has as objects pairs (b, c) where $b \in \mathscr{B}$ and $c \in p^{-1}(b)$. The arrows of this category are pairs $(f, \mu)\colon (b, c) \to (b', c')$ where $f\colon b \to b'$ is a map in $\mathscr{B}$ and $\mu\colon c \to \operatorname{dom} S_p(f, c')$ is a map in $\mathscr{C}$.
- The functor $Q\colon \mathscr{D} \to \mathscr{B}^{\to}$ takes (b, c) to $P(c)$ and $(f, \mu)\colon (b, c) \to (b', c')$ to $\Big(P_0(S_p(f, c)) \circ P_0(\mu), f\Big)$. This can be visualised as

$$\begin{tikzcd}
P_0(c) \arrow[r, "P_0(\mu)"] \arrow[d, "P(c)"'] & P_0\big(\operatorname{dom} S_p(f,c)\big) \arrow[rr, "P_0\left(S_p(f,c)\right)"] \arrow[d, "P(\operatorname{dom} S_p(f,c))"] \arrow[drr, phantom, "\lrcorner", very near start] & & P_0(c') \arrow[d, "P(c')"] \\
b \arrow[r, "1_b"] & b \arrow[rr, "f"] & & b'.
\end{tikzcd}$$

So it is immediate that cartesian arrows get taken to pullbacks as all cartesian arrows satisfy $\mu = 1_{\operatorname{dom} S_p(f,c)}$ and thus the square on the left above is trivial.

The isomorphism $(P\colon \mathscr{C} \to \mathscr{B}^{\to}) \to (Q\colon \mathscr{D} \to \mathscr{B}^{\to})$ is then defined as follows: It is the identity on $\mathscr{B}$ and takes c in $\mathscr{C}$ to $\big(p(c), c\big)$ as well as $f\colon c \to c'$ to $\big(p(f), t\big)$, where t is obtained in the following manner: first we take the cartesian lift $S_p(f, c')$ of $p(f)$ obtained by the splitting and the use the unique filler from the cartesianness of $S_p(f, c)$, as illustrated in the diagram below.

$$\begin{tikzcd}
 & \operatorname{dom}(S_p(f,c)) \arrow[rr, mapsto, "p"] \arrow[dl, dotted, "t"] \arrow[dd, "S_p(f,c)"] & & p(c) \arrow[dl, equals] \arrow[dd, "p(f)"] \\
c \arrow[rr, mapsto, "p"] \arrow[dr, "f"'] & & p(c) \arrow[dr, "p(f)"'] & \\
 & c' \arrow[rr, mapsto, "p"] & & p(c').
\end{tikzcd}$$

This is a strict, split map of comprehension categories as the diagram

$$\begin{tikzcd}
f \arrow[r, mapsto] \arrow[d, mapsto] & \big(p(f), t\big) \arrow[dr, mapsto] & \\
P(f) \arrow[r, mapsto] & P(f) \arrow[r, equals] & P_0(S_p(p(f), p(c))) \circ P_0(t)
\end{tikzcd}$$

where we used that $S_p(p(f), p(c)) = f$ and $P_0(f) \circ P_0(t)$. It is immediate that this assignment is both invertible and natural. Q.E.D.

We summarize the correspondences of the different "ingredients" in both (2-fam,Σ)-categories and comprehension categories in a table.

(2-fam,Σ)-categories	comprehension categories
objects	objects of the base category
family arrows over c	objects of the total category over c
precomposition $\lambda \circ f$	splitting of f and λ
Σ-objects and projection	comprehension functor P

By restricting to strict maps and discrete fibrations each we also get the following equivalences:

Corollary 3.2.14. *There are 2-equivalences*

$$\begin{aligned}\mathsf{ComprC}^{\mathrm{str}}_{\mathrm{spl}} &\simeq (\mathsf{2\text{-}fam}, \Sigma)\mathsf{C},\\ \mathsf{ComprC}_{\mathrm{disc}} &\simeq (\mathsf{fam}, \Sigma)\mathsf{C}^{\mathrm{wk}},\\ \mathsf{ComprC}^{\mathrm{str}}_{\mathrm{disc}} &\simeq (\mathsf{fam}, \Sigma)\mathsf{C}\,.\end{aligned}$$

It is worth noting that every discrete fibration is split as there is exactly one lift given an arrow $f\colon a \to b$ and λ over b, so the category $\mathsf{ComprC}_{\mathrm{disc,spl}}$ is the same as $\mathsf{ComprC}_{\mathrm{disc}}$.

Chapter 4
Dependent arrows in fam-categories

In this chapter we review the extension of categories with family arrows due to Petrakis and compare it to the dependent products Pitts extends his type categories by. We then introduce iterated discrete fibrations and show that these are equivalent to dependent arrows under the previously established equivalence

$$\{\text{categories with family arrows}\} \leftrightarrow \{\text{discrete fibrations}\}.$$

4.1 dep-categories

Definition 4.1.1 (dep-categories). A dep-category $\mathscr{C}$ is a fam-category $\mathscr{C}$ in the sense of definition 2.1.1 such that for each $c \in \mathscr{C}$ and each $\lambda \in \mathrm{fHom}(c)$ we are given a collection $\mathrm{dHom}(\lambda)$ of dependent arrows. A dependent arrow $\Phi \in \mathrm{dHom}(\lambda)$ for $\lambda \in \mathrm{fHom}(c)$ is depicted in a diagram as follows:

$$\begin{tikzcd} c \arrow[r, "\lambda"{name=L, near end}] \arrow[to=L, Rightarrow, bend left, "\Phi"] & {} \end{tikzcd}$$

Additionally a map that assigns to each $\Phi \in \mathrm{dHom}(\lambda)$ and each $f\colon c' \to c$ a dependent arrow $\Phi[f] \in \mathrm{dHom}(\lambda \circ f)$ is required, meeting the following conditions:

(dep$_1$) For each $c \in \mathscr{C}, \lambda \in \mathrm{fHom}(c)$ and $\Phi \in \mathrm{dHom}(\lambda)$ the equation $\Phi[1_c] = \Phi$ holds.

$$\begin{tikzcd} c \arrow[r, "1_c"] \arrow[rr, Rightarrow, bend left=60, "\Phi", to=L] & c \arrow[r, "\lambda"{name=L, near end}] \arrow[to=L, Rightarrow, bend left, "\Phi"] & {} \end{tikzcd}$$

(dep$_2$) For each $c \in \mathscr{C}, \lambda \in \mathrm{fHom}(c), f\colon c'' \to c', g\colon c' \to c$ and $\Phi \in \mathrm{dHom}(\lambda)$ we have $\Phi[g \circ f] = \big(\Phi[g]\big)[f]$.

$$\begin{tikzcd} c'' \arrow[r, "f"] \arrow[to=L, Rightarrow, bend right, "(\Phi[g])[f]"'] & c' \arrow[r, "g"] \arrow[to=L, Rightarrow, bend left, "\Phi[g]"] & c \arrow[r, "\lambda"{name=L, near end}] & {} \end{tikzcd}$$

4.1.2 The diagrammatical language. We want to make a few remarks on the diagrammatical language we chose. Both Petrakis and Ehrhardt depict their dependent arrows as living above their respective family arrows as in

$$\begin{tikzcd} c \arrow[r, "\lambda"] \arrow[r, Rightarrow, bend right, "\Phi"] & {} \end{tikzcd} .$$

We instead chose the approach of having the dependent arrow pointing at the family arrow itself while starting at the object over which the family arrow exists, that is as

$$\begin{tikzcd} c \arrow[r, "\lambda"{name=L, near end}] \arrow[to=L, Rightarrow, bend left, "\Phi"] & {} \end{tikzcd}$$

However, this means that commutative diagrams carry more information that one might expect at first glance. If we for example consider

$$\begin{tikzcd}
c' \arrow[r, "f"] \arrow[rr, bend right, "\eta"'] \arrow[r, Rightarrow, bend right=60, "\Psi"'] & c \arrow[r, "\lambda"] \arrow[r, Rightarrow, bend left=60, "\Phi"] & {}
\end{tikzcd}$$

the commutativity of this diagram not only means $\lambda \circ f = \eta$ but also $\Phi[f] = \Psi$, even though the arrows $\Phi[f], \Psi$ do not point at the same target.

Examples 4.1.3. 1. Every fam-category can be turned into a dep-category by defining

$$\mathrm{dHom}(a, \lambda) = \mathbb{1}$$

for every $a \in \mathscr{C}, \lambda \in \mathrm{fHom}(a)$. The conditions are trivially satisfied.

2. In the category Sets with the fam-category structure as before we can define the dependent arrows as such:

$$\mathrm{dHom}\left(S, (B_s)_{s\in S}\right) = \prod_{s\in S} B_s = \left\{\Phi \colon S \to \mathsf{Sets} \mid \forall_{s\in S}\Phi(s) \in B_s\right\}.$$

The applications are defined as such: for $\Phi \in \prod_{s\in S} B_s$ and $f \colon T \to S$ define $\Phi(f)$ as $\Phi \colon f$. It is immediate that this lies in $\mathrm{dHom}(T, (B_{f(t)})_{t\in T})$ as $\Phi(f(t)) \in B_{f(t)}$ for every $t \in T$ by design. The strictness conditions are immediate from the definition of the application.

Pitts also endows type categories (that is (fam,Σ)-categories with a terminal object) with dependent arrows, albeit in a different manner. Before we give Pitts definition of dependent products we need to make a few technical remarks about the pullback of projections.

1. The collection $\mathrm{type}_{\mathscr{C}}(c)$ can be made into a subcategory of $\mathscr{C}/c$ as such: Each c-indexed type λ gets mapped to pr_1^λ, and an arrow $\lambda \to \lambda'$ is simply an arrow $f \colon \sum_c \lambda \to \sum_c \lambda'$ such that $\mathsf{pr}_1^{\lambda'} \circ f = \mathsf{pr}_1^\lambda$.
2. If we have a c-indexed type λ then we can define a functor $(\mathsf{pr}_1^\lambda)^* \colon \mathscr{C}/c \to \mathscr{C}/\sum_c \lambda$ as follows:
   - Each $f \colon d \to c$ gets mapped to $\sum_d f \colon \sum_d f^*\lambda \to \sum_c \lambda$.
   - Each $g \colon f \to f'$, that is g such that

$$\begin{tikzcd}
d \arrow[r, "g"] \arrow[rd, "f"'] & d' \arrow[d, "f'"] \\
& c
\end{tikzcd}$$

commutes gets mapped to the arrow $\tilde{g} := \langle g \circ \mathsf{pr}_1^{f^*\lambda}, \sum_c f\rangle$ which arises from

$$\begin{tikzcd}
\sum_d f^*\lambda \arrow[rrd, "\sum_d f"] \arrow[rd, dotted, "\tilde{g}"] \arrow[dd, "\mathsf{pr}_1^{f^*\lambda}"] & & \\
& \sum_{d'}(f')^*\lambda \arrow[r, "\sum_{d'} f'"] \arrow[d, "\mathsf{pr}_1^{(f')^*\lambda}"] \arrow[rd, phantom, "\lrcorner", very near start] & \sum_c \lambda \arrow[d, "\mathsf{pr}_1^\lambda"] \\
d \arrow[r, "g"] & d' \arrow[r, "f"] & c.
\end{tikzcd}$$

3. If we have an arrow $f\colon d \to c$, then we obtain a functor $f^*\colon \operatorname{type}_{\mathscr{C}}(c) \to \operatorname{type}_{\mathscr{C}}(d)$ (where these are made categories as described above) in the following manner:
   - Every λ gets mapped to $f^*\lambda$, using the identification this means that pr_1^{λ} gets mapped to $\mathsf{pr}_1^{f^*\lambda}$.
   - Every $g\colon \lambda \to \lambda'$ that is $g\colon \sum_c \lambda \to \sum_c \lambda'$ such that $\mathsf{pr}_1^{f^*\lambda'} \circ g = \mathsf{pr}_1^{f^*\lambda}$ gets mapped to $\hat{g} := \langle \mathsf{pr}_1^{f^*\lambda}, g \circ \sum_d f\rangle$ which arises from

$$\begin{tikzcd}
\sum_d f^*\lambda \arrow[rr, "\Sigma_d f"] \arrow[dr, dotted, "\hat{g}"] \arrow[ddr, rounded corners, "\mathsf{pr}_1^{f^*\lambda}"'] & & \sum_c \lambda \arrow[d, "g"] \\
& \sum_{d'} f^*\lambda' \arrow[r, "\Sigma_c f'"] \arrow[d, "\mathsf{pr}_1^{f^*\lambda'}"] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_c \lambda' \arrow[d, "\mathsf{pr}_1^{\lambda'}"] \\
& d \arrow[r, "f"] & c
\end{tikzcd}$$

   where the outer square commutes due to $\mathsf{pr}_1^{f^*\lambda'} \circ g = \mathsf{pr}_1^{f^*\lambda}$.

Definition 4.1.4 (Type categories with dependent products - [51]). A type category $\mathscr{C}$ has *dependent products* if it is endowed with the following exrta structure: for each object $c \in \mathscr{C}, \lambda \in \operatorname{type}_{\mathscr{C}}(c)$ and $\lambda' \in \operatorname{type}_{\mathscr{C}}\big(\sum_c \lambda\big)$ there is a c-indexed type $\prod(\lambda, \lambda') \in \operatorname{type}_{\mathscr{C}}(c)$ and a morphism $\operatorname{ap}_{\lambda,\lambda'} : (\mathsf{pr}_1^{\lambda})^* \prod(\lambda,\lambda') \to \lambda'$ satisfying the following properties:

1. *Adjointness property:* for any $f\colon c' \to c$ in $\mathscr{C}$ and $g\colon (\mathsf{pr}_1^{\lambda})^*(f) \to \mathsf{pr}_1^{\lambda'}$ in $\mathscr{C}/\sum_c \lambda$ there is a unique morphism in $\mathscr{C}/c$ which we call $\operatorname{cur}(g)\colon f \to \mathsf{pr}_1^{\prod(\lambda,\lambda')}$ such that $\operatorname{ap}_{\lambda,\lambda'} \circ (\mathsf{pr}_1^{\lambda})^*\big(\operatorname{cur}(g)\big) = g$.
2. *Strictness property:* For any morphism $f\colon c' \to c$ in $\mathscr{C}$ we have

$$f^* \prod(\lambda,\lambda') = \prod\Big(f^*\lambda, \Big(\sum_c f\Big)^* \lambda'\Big),$$
$$\Big(\sum_c f\Big)^* (\operatorname{ap}_{\lambda,\lambda'}) = \operatorname{ap}_{f^*\lambda, (\sum_c \lambda)^*\lambda'}\,.$$

Pitts himself used these type categories with dependent products as the classifying categories of dependently typed equational logic with dependent product types. We demonstrate that these dependent products differ greatly from dependent arrows as in dep-categories by exhibiting that Sets also has dependent product types in this manner.

Proposition 4.1.5. Sets *is a type category with dependent products.*

Proof: For this we first note that in this definition the types λ are given as $\lambda\colon S \to \mathsf{Sets}$, that is each element $s \in S$ gets mapped to a set. With this definition

$$\begin{aligned}
\sum_S \lambda := \coprod_{s\in S} \lambda(s) &= \big\{(s,x) \mid x \in \lambda(s)\big\},\\
f^*\lambda &:= \lambda \circ f,\\
\Sigma_\lambda f\colon \sum_T \lambda \circ f &\to \sum_S \lambda\\
(t,x) &\mapsto \big(f(t), x\big)
\end{aligned}$$

for every $f\colon T \to S$. We then define the dependent arrow structure as follows: for $\lambda\colon S \to \mathsf{Sets}, \lambda'\colon \sum_S \lambda \to \mathsf{Sets}$ define $\prod(\lambda,\lambda')\colon S \to \mathsf{Sets}$ via $s \mapsto \prod_{x\in\lambda(s)} \lambda'(s,x)$. Define the application arrow

$$\mathrm{ap}_{\lambda,\lambda'}\colon \sum_{\sum_S \lambda} \Big(\prod(\lambda,\lambda')\circ \mathsf{pr}_1^\lambda\Big) \to \sum_{\sum_S\lambda} \lambda' \quad \text{via} \quad \big((s,x'),(a_x)_{x\in\lambda(s)}\big) \mapsto \big((s,x'),a_{x'}\big).$$

To prove that this makes Sets a type category with dependent arrows we need to show the adjunction and strictness conditions. Before we do this we need to understand the functors $(\mathsf{pr}_1^\lambda)^*\colon \mathsf{Sets}/S \to \mathsf{Sets}/\sum_S \lambda$ and $f^*\colon \mathrm{type}(S) \to \mathrm{type}(T)$ for $f\colon T\to S$ better. By definition we know that for any $g\colon U_1 \to S$ we have that $(\mathsf{pr}_1^\lambda)^*(g) = \Sigma_\lambda(g)\colon \sum_{U_1}\lambda\circ g \to \sum_S \lambda$. For $h\colon g_1 \to g_2$ we have that $(\mathsf{pr}_1^\lambda)^*(g)\colon \sum_\lambda g_1 \to \sum_\lambda g_2$ is the unique arrow arising from the pullback diagram

$$\begin{tikzcd}
\sum_{U_1} \lambda\circ g_1 \arrow[rrd, bend left, "\Sigma_\lambda g_1"] \arrow[rd, dashed, "(\mathsf{pr}_1^\lambda)^* h"] \arrow[dd, "\mathsf{pr}_1^{\lambda\circ g_1}"'] & & \\
& \sum_{U_2}\lambda\circ g_2 \arrow[r, "\Sigma_\lambda g_2"] \arrow[d, "\mathsf{pr}_1^{\lambda\circ g_2}"] & \sum_S \lambda \arrow[d, "\mathsf{pr}_1^\lambda"] \\
U_1 \arrow[r, "h"] & U_2 \arrow[r, "g_2"] & S.
\end{tikzcd}$$

Thus we can compute that

$$(\mathsf{pr}_1^\lambda)^*(h)(u,x) = \big(h(u),x\big) \quad \text{for all } (u,x) \in \sum_{U_1} \lambda\circ g_1.$$

Conversely we have for all $\lambda\colon S \to \mathsf{Sets}$ that $f^*(\lambda) = \lambda\circ f\colon T \to \mathsf{Sets}$. If $h\colon \lambda \to \lambda'$, that is $h\colon \sum_S\lambda \to \sum_S\lambda'$ and $\mathsf{pr}_1^{\lambda'}\circ h = \mathsf{pr}_1^\lambda$, then $f^*(h)$ is defined through the pullback diagram

$$\begin{tikzcd}
\sum_T \lambda\circ f \arrow[rr, "\Sigma_\lambda f"] \arrow[rd, dashed, "f^*(h)"] \arrow[rdd, bend right, "\mathsf{pr}_1^{\lambda\circ f}"'] & & \sum_S \lambda \arrow[d, "h"] \\
& \sum_T \lambda'\circ f \arrow[r, "\Sigma_{\lambda'} f"] \arrow[d, "\mathsf{pr}_1^{\lambda'\circ f}"] & \sum_S \lambda' \arrow[d, "\mathsf{pr}_1^\lambda"] \\
& T \arrow[r, "f"] & S.
\end{tikzcd}$$

Thus we compute that

$$f^*(h)(t,u) = \Big(t, h_2\big(f(t),u\big)\Big) \quad \text{for all } (t,u) \in \sum_T \lambda\circ f$$

where $h_2\big(f(t),u\big)$ arises from $h(s,u) = \big(h_1(s,u),h_2(s,u)\big)$ for all $(s,u) \in \sum_S\lambda$.

Lemma 4.1.6. *Let $\mathscr{C}$ be a type category and $X \in \mathscr{C}, \lambda,\lambda' \in \mathrm{type}_{\mathscr{C}}(X)$. Then*

$$\sum_{\sum_X \lambda} (\mathsf{pr}_1^\lambda)^*\lambda' = \sum_{\sum_X \lambda'} (\mathsf{pr}_1^{\lambda'})^*\lambda.$$

Proof: We do this by using that both

$$\begin{tikzcd}
\sum_{\sum_X\lambda}(\mathsf{pr}_1^\lambda)^*\lambda' \arrow[r, "\Sigma_\lambda \mathsf{pr}_1^\lambda"] \arrow[d, "\mathsf{pr}_1^{(\mathsf{pr}_1^\lambda)^*\lambda'}"] & \sum_X \lambda' \arrow[d, "\mathsf{pr}_1^{\lambda'}"] \\
\sum_X \lambda \arrow[r, "\mathsf{pr}_1^\lambda"] & X
\end{tikzcd}$$

and

$$\begin{tikzcd}
\sum_{\sum_X\lambda'}(\mathsf{pr}_1^{\lambda'})^*\lambda \arrow[r, "\Sigma_\lambda \mathsf{pr}_1^{\lambda'}"] \arrow[d, "\mathsf{pr}_1^{(\mathsf{pr}_1^{\lambda'})^*\lambda}"] & \sum_X \lambda \arrow[d, "\mathsf{pr}_1^{\lambda}"] \\
\sum_X \lambda' \arrow[r, "\mathsf{pr}_1^{\lambda'}"] & X
\end{tikzcd}$$

are pullback squares. So we get unique arrows f, g making the diagrams

$$\begin{tikzcd}
\sum_{\sum_X \lambda'}(\mathsf{pr}_1^{\lambda'})^*\lambda \arrow[rrd, bend left, "\sum_\lambda \mathsf{pr}_1^{\lambda'}"] \arrow[rdd, bend right, "\mathsf{pr}_1^{(\mathsf{pr}_1^{\lambda'})^*\lambda}"'] \arrow[rd, dotted, "f"] & & \\
& \sum_{\sum_X \lambda}(\mathsf{pr}_1^{\lambda})^*\lambda' \arrow[r, "\sum_\lambda \mathsf{pr}_1^{\lambda}"] \arrow[d, "\mathsf{pr}_1^{(\mathsf{pr}_1^{\lambda})^*\lambda'}"] & \sum_X \lambda' \arrow[d, "\mathsf{pr}_1^{\lambda'}"] \\
& \sum_X \lambda \arrow[r, "\mathsf{pr}_1^{\lambda}"] & X
\end{tikzcd}$$

and

$$\begin{tikzcd}
\sum_{\sum_X \lambda}(\mathsf{pr}_1^{\lambda})^*\lambda' \arrow[rrd, bend left, "\sum_\lambda \mathsf{pr}_1^{\lambda}"] \arrow[rdd, bend right, "\mathsf{pr}_1^{(\mathsf{pr}_1^{\lambda})^*\lambda'}"'] \arrow[rd, dotted, "g"] & & \\
& \sum_{\sum_X \lambda'}(\mathsf{pr}_1^{\lambda'})^*\lambda \arrow[r, "\sum_\lambda \mathsf{pr}_1^{\lambda'}"] \arrow[d, "\mathsf{pr}_1^{(\mathsf{pr}_1^{\lambda'})^*\lambda}"] & \sum_X \lambda \arrow[d, "\mathsf{pr}_1^{\lambda}"] \\
& \sum_X \lambda' \arrow[r, "\mathsf{pr}_1^{\lambda'}"] & X
\end{tikzcd}$$

commute. Thus $g \circ f = \mathbf{1}_{\sum_{\sum_X \lambda'}(\mathsf{pr}_1^{\lambda'})^*\lambda}$ and $f \circ g = \mathbf{1}_{\sum_{\sum_X \lambda}(\mathsf{pr}_1^{\lambda})^*\lambda'}$ Q.E.D.

Remark 4.1.7. In the instance where we will use this, that is $\mathscr{C} = \mathsf{Sets}$, a small notational hiccup occurs. Here for $\lambda, \lambda' \colon S \to \mathsf{Sets}$ we get that

$$\sum_{\sum_S \lambda} (\mathsf{pr}_1^{\lambda})^*\lambda' = \big\{\big((s,x),y\big) \mid s \in S \wedge x \in \lambda(s) \wedge y \in \lambda'(s)\big\},$$
$$\sum_{\sum_S \lambda'} (\mathsf{pr}_1^{\lambda'})^*\lambda = \big\{\big((s,y),x\big) \mid s \in S \wedge y \in \lambda'(s) \wedge x \in \lambda(s)\big\}.$$

We will from now on always tacitly identify the both.

Adjunction condition For this let $f \colon T \to S, \lambda \colon S \to \mathsf{Sets}, \lambda' \colon \sum_S \lambda \to \mathsf{Sets}$ be given. Additionally let $g \colon (\mathsf{pr}_1^A)^*(f) \to \mathsf{pr}_1^{\lambda'}$ be given, that is an arrow g making the triangle

$$\begin{tikzcd}
\sum_T \lambda \circ f \arrow[r, "g"] \arrow[rd, "\sum_\lambda f"'] & \sum_{\sum_S \lambda} \lambda' \arrow[d, "\mathsf{pr}_1^{\lambda'}"] \\
& \sum_S \lambda
\end{tikzcd} \tag{9}$$

commute. Then we must obtain a unique $\mathrm{cur}(g) \colon T \to \sum_S \lambda$ making

$$\begin{tikzcd}
T \arrow[r, "\mathrm{cur}(g)"] \arrow[rd, "f"'] & \sum_S \prod(\lambda, \lambda') \arrow[d, "\mathsf{pr}_1^{\prod(\lambda,\lambda')}"] \\
& S
\end{tikzcd} \tag{10}$$

commute. This $\mathrm{cur}(g)$ is defined in the following way: From the commutativity of (9) we know that $g(t,x) = \big((f(t),x), g_2(t,x)\big)$ where $g_2(t,x) \in \lambda'\big(f(t),x\big)$. We then set

$$\mathrm{cur}(g)(t) := \Big(f(t), \big(g_2(t,x)\big)_{x \in \lambda(f(t))}\Big).$$

It is immediate that this definition makes (10) commute as

$$\mathsf{pr}_1^{\prod(\lambda,\lambda')}\big(\mathrm{cur}(g)(t)\big) = \mathsf{pr}_1^{\prod(\lambda,\lambda')}\Big(f(t),\big(g_2(t,x)\big)_{x\in\lambda(f(t))}\Big) = f(t)$$

for all $t \in T$. To verify the equation $\mathrm{ap}_{\lambda,\lambda'} \circ (\mathsf{pr}_1^{\lambda})^*(\mathrm{cur}(g)) = g$ we simply compute that for all $(t,u) \in \sum_T \lambda \circ f$

$$\begin{aligned}\mathrm{ap}_{\lambda,\lambda'} \circ (\mathsf{pr}_1^{\lambda})^*(\mathrm{cur}(g))(t,u) &= \mathrm{ap}_{\lambda,\lambda'}\big(\mathrm{cur}(t),u\big)\\ &= \mathrm{ap}_{\lambda,\lambda'}\big(f(t),(g_2(t,x))_{x\in\lambda(f(t))},u\big)\\ &= \mathrm{ap}_{\lambda,\lambda'}\big(f(t),g_2(t,u),u)\big) = g(t,u).\end{aligned}$$

It is immediate that $\mathrm{cur}(g)$ is the only function with this property and thus necessarily unique.

Strictness conditions We have to check that for any $f\colon T \to S$ we have that

$$\prod(\lambda,\lambda') \circ f = \prod(\lambda \circ f, \lambda' \circ \sum_{\lambda} f),$$
$$\Big(\sum_{\lambda} f\Big)^*(\mathrm{ap}_{\lambda,\lambda'}) = \mathrm{ap}_{\lambda\circ f,\lambda'\circ(\sum_\lambda f)}\,.$$

This can by verified via

$$\begin{aligned}\Big(\prod(\lambda,\lambda')\circ f\Big)(t) = \prod_{x\in\lambda(f(t))}\lambda'\big(f(t),x\big) &= \prod_{x\in\lambda(f(t))}\Big(\lambda'\circ\sum_{\lambda}(f)\Big)(t,x)\\ &= \prod(\lambda\circ f,\lambda'\circ\sum_{\lambda}f)(t) \quad \text{(for all } t\in T)\end{aligned}$$

and

$$\begin{aligned}\Big(\sum_{\lambda} f\Big)^*(\mathrm{ap}_{\lambda,\lambda'})\big((t,x),(a_y)_{y\in\lambda(f(t))}\big) &= \Big((t,x),\mathrm{ap}^2_{\lambda,\lambda'}\big(\sum_{\lambda} f(t,x),(a_y)_{y\in\lambda(f(t))}\big)\Big)\\ &= \big((t,x),a_x\big)\\ &= \mathrm{ap}_{\lambda\circ f,\lambda'\circ(\sum_\lambda f)}\big((t,x),(a_y)_{y\in\lambda(f(t))}\big)\end{aligned}$$

for all $\big(t,x),(a_y)_{y\in\lambda(f(t))}\big) \in \sum_{\sum_T \lambda\circ f}\Big(\prod(\lambda,\lambda')\circ\mathsf{pr}_1^{\lambda\circ f}\Big)$. Q.E.D.

Definition 4.1.8 (dep-functors, dep-natural transformations). A *dep-functor* $F\colon \mathscr{C} \to \mathscr{D}$ is a fam-functor $F\colon \mathscr{C} \to \mathscr{D}$ in the sense of definition 2.1.4 together with a rule F_λ that assigns to each $\Phi \in \mathrm{dHom}(\lambda)$ a $F_\lambda(\Phi) \in \mathrm{dHom}\big(F_c(\lambda)\big)$. This is required to satisfy:

(dep$_3$) For each $f\colon c' \to c$, $\lambda \in \mathrm{fHom}(c)$ and $\Phi \in \mathrm{dHom}(\lambda)$ we have $F_{\lambda\circ f}\big(\Phi[f]\big) = F_\lambda(\Phi)\big[F(f)\big]$.

A *dep-natural transformation* $\eta\colon F \Rightarrow G$ is a fam-natural transformation $\eta\colon F \Rightarrow G$ in the sense of 2.1.4 such that additionally

(dep$_4$) for all $c \in \mathscr{C}, \lambda \in \mathrm{fHom}(c)\}$ and $\Phi \in \mathrm{dHom}(\lambda)$ we have $\big[G_\lambda(\Phi)\big](\eta_c) = F_\lambda(\Phi)$.

All of the above allows us to define the 2-category depC of dep-categories, dep-functors and dep-natural transformations.

Definition 4.1.9 (Category of dep-categories). The cateogry depC has

- as *objects* categories with dependent arrows in the sense of definition 4.1.1,
- as *1-maps* dep-functors in the sense of definition 4.1.8 and
- as *2-maps* dep-natural transformations in the sense of 4.1.8.

4.2 Iterated discrete fibrations

Definition 4.2.1. We define the 2-category dFibb via the following data:

- its objects are pairs (q,p) of composable functors $q\colon \mathscr{G} \to \mathscr{E}, p\colon \mathscr{E} \to \mathscr{B}$ such that both p and $p \circ q$ are discrete fibrations.
- A 1-map is a triple $(\tilde{F}, \hat{F}, F)\colon (q,p) \to (q',p')$ such that $(\hat{F}, F)\colon p \to p'$ and $(\tilde{F}, F)\colon p\circ q \to p' \circ q'$ are maps of discrete fibrations.
- A 2-map is a triple $(\eta, \hat{\eta}, \tilde{\eta})\colon (\tilde{F}, \hat{F}, F) \to (\tilde{G}, \hat{G}, G)$ such that $(\hat{\eta}, \eta)\colon (\hat{F}, F) \Rightarrow (\hat{G}, G)$ and $(\eta, \tilde{\eta})\colon (F, \tilde{F}) \Rightarrow (G, \tilde{G})$ are transformations of maps of discrete fibrations.

Remark 4.2.2. Similar to pullbacks discrete fibrations fulfil a pasting property, that is given two functors

$$\mathscr{G} \xrightarrow{q} \mathscr{E} \xrightarrow{p} \mathscr{B}$$

such that p is a discrete fibration, the following are equivalent:

- q is a discrete fibration.
- $p \circ q$ is a discrete fibration.

This can be motivated by the fact that if all categories involved are small, then p is a discrete fibration if and only if

$$\begin{tikzcd}
\mathscr{E}_1 \arrow[r, "p_1"] \arrow[d, "\mathrm{cod}"] & \mathscr{B}_1 \arrow[d, "\mathrm{cod}"] \\
\mathscr{E}_0 \arrow[r, "p_0"] & \mathscr{B}_0
\end{tikzcd}$$

is a pullback (cf. [56]), thus we obtain the pasting diagram

$$\begin{tikzcd}
\mathscr{G}_1 \arrow[r, "q_1"] \arrow[d, "\mathrm{cod}"] & \mathscr{E}_1 \arrow[r, "p_1"] \arrow[d, "\mathrm{cod}"] & \mathscr{B}_1 \arrow[d, "\mathrm{cod}"] \\
\mathscr{G}_0 \arrow[r, "q_0"] & \mathscr{E}_0 \arrow[r, "p_0"] & \mathscr{B}_0.
\end{tikzcd}$$

However this can also be checked for categories that are not small. Thus we could equivalently demand that $(\hat{F}, F)$ and $(\hat{G}, G)$ are maps of the respective fibrations, analogously with the transformations.

Proposition 4.2.3. *There is a 2-equivalence*

$$\mathsf{depC} \simeq \mathsf{dFibb}\,.$$

Proof. We give the two functors in each direction.

$\boxed{(q,p)_{-}\colon \mathsf{depC} \to \mathsf{dFibb}}$

Given a dep-category $\mathscr{C}$, we define two functors $p_{\mathscr{C}}\colon \mathscr{F} \to \mathscr{C}, q_{\mathscr{C}}\colon \mathscr{D} \to \mathscr{F}$ where $\mathscr{F}$ is the category of family arrows of $\mathscr{C}$, that is its objects are family arrows λ and its arrows are $f\colon \lambda \circ f \to \lambda$ where $f\colon c' \to c$ (and $\lambda \in \mathrm{fHom}(c)$). Similarly $\mathscr{D}$ is the category of dep-arrows, that is its objects are Φ and the arrows are $f\colon \Phi \circ f \to \Phi$ for $f\colon c' \to c$ (where Φ lies over $\lambda \in \mathrm{fHom}(c)$).

The two discrete fibrations are then defined as follows:

$$\begin{aligned} p(\lambda) = c \text{ for } \lambda \in \mathrm{fHom}(c), \quad p(f) = f, \\ q(\Phi) = \lambda \text{ for } \Phi \in \mathrm{dHom}(\lambda), \quad q(f) = f. \end{aligned}$$

It is immediate that these constitute functors, the associativity and unity follow from (fam$_{1\text{-}2}$) and (dep$_{1\text{-}2}$) respectively. They are also discrete fibrations, the unique lift of $f\colon c' \to c$ along p and λ is $f\colon \lambda \circ f \to \lambda$, and the unique lift of this f along $p \circ q$ and Φ is $f\colon \Phi \circ f \to \Phi$.

Given a dep-functor $F\colon \mathscr{C} \to \mathscr{D}$, this functor is mapped to $(\tilde{F}, \hat{F}, F)\colon (q,p)_{\mathscr{C}} \to (q,p)_{\mathscr{D}}$ where $\tilde{\hat{F}}, F$ are defined as follows:

$$\hat{F}(\lambda) = F_c(\lambda) \text{ for } \lambda \in \mathrm{fHom}(c), \quad \hat{F}(f) = F(f),$$
$$\tilde{F}(\lambda) = F^{\lambda}(\Phi) \text{ for } \Phi \in \mathrm{dHom}(\lambda), \quad \tilde{F}(f) = F(f).$$

The functoriality of $\tilde{\hat{F}}, F$ immediately follow from the functoriality of F. To see that $(\hat{F}, F)\colon p_{\mathscr{C}} \to p_{\mathscr{D}}$ and $(\tilde{F}, F)\colon p_{\mathscr{C}} \circ q_{\mathscr{C}} \to p_{\mathscr{D}} \circ q_{\mathscr{D}}$ are maps of discrete fibrations we remark that

$$(p_{\mathscr{D}} \circ \hat{F})(\lambda) = F(c) = F\big(p_{\mathscr{C}}(\lambda)\big) \text{ and } (p_{\mathscr{C}} \circ \hat{F})(f) = F(f) = F\big(p_{\mathscr{C}}(f)\big)$$

as well as

$$(p_{\mathscr{C}} \circ q_{\mathscr{C}} \circ \tilde{F})(\Phi) = F(\Phi) = F\big((p_{\mathscr{D}} \circ q_{\mathscr{C}})(\Phi)\big) \text{ and } (p_{\mathscr{C}} \circ q_{\mathscr{C}} \circ \tilde{F})(f) = F(f) = F\big((p_{\mathscr{D}} \circ q_{\mathscr{C}})(f)\big)$$

Similarly, for a dep-natural transformation $\eta\colon F \Rightarrow G$ we define a map

$$(\tilde{\eta}, \hat{\eta}, \eta)\colon (\tilde{F}, \hat{F}, F) \Rightarrow (\tilde{G}, \hat{G}, G), \quad \hat{\eta}_{\lambda} := \eta_c =: \tilde{\eta}_{\Phi}.$$

This choice works as $\eta_c\colon \hat{F}(\lambda) \to \hat{G}(\lambda)$ because $\hat{F}(\lambda) = F_c(\lambda) = G_c(\lambda) \circ \eta_c = \hat{G}(\lambda) \circ \eta_c$ where we used that F is a fam-functor for the second to last equality. A similar computation also yields the same for $\tilde{F}(\Phi)$ and $\tilde{G}(\Phi)$.

It is straightforward to compute that these choices for $\hat{\eta}, \tilde{\eta}$ make $(\eta, \hat{\eta}, \tilde{\eta})$ a 2-map in dFibb.

$\boxed{\mathscr{C}_{\_}\colon \mathsf{dFibb} \to \mathsf{depC}}$

In the reverse direction we map $(q\colon \mathscr{G} \to \mathscr{E}, p\colon \mathscr{E} \to \mathscr{B})$ in dFibb to the dep-category $\mathscr{B}$ defined as follows:

- The underlying category is $\mathscr{B}$.
- For each $b \in \mathscr{B}$ the family arrows are defined via $\mathrm{fHom}(b) := p^{-1}(b)$.
- For each $b \in \mathscr{B}, \lambda \in \mathrm{fHom}(b)$ the dependent arrows are defined via $\mathrm{dHom}(\lambda) := q^{-1}(\lambda)$.
- For $f\colon c \to c'$ we define $\lambda\colon f$ to be the domain of the unique lift of f along p and λ. Furthermore we define $\Phi[f]$ to be the domain of the unique lift of f along $p \circ q$ and Φ.

It is straightforward to check that this constitutes a dep-category, one only needs to use that the lifts of identities are identities and the lift of compositions is obtained by composing the lifts.

A map $(\tilde{F}, \hat{F}, F)\colon (q,p) \to (q',p')$ is then mapped to the dep-functor $F\colon \mathscr{B} \to \mathscr{B}'$ where $F_b(\lambda) := \hat{F}(\lambda), F^{\lambda}(\Phi) := \tilde{F}(\Phi)$. It is immediate that this constitutes a fam-functor as

$$F_c(\lambda \circ f) = \hat{F}(\lambda \circ f) = \hat{F}(\lambda) \circ F(f)$$

which follows from the fact that $(\hat{F}, F)\colon p \to p'$ is a map of discrete fibrations, thus it interacts well with lifts. In a diagram:

$$\begin{tikzcd}
\big(\lambda \circ f \arrow[r, "f_\ell"] \arrow[d, mapsto, "p"'] & \lambda\big) \arrow[r, mapsto, "\hat{F}"] \arrow[d, mapsto, "p"] & \big(\hat{F}(\lambda \circ f) \arrow[r, "\hat{F}(f_\ell)"] \arrow[d, mapsto, "p'"'] & \hat{F}(\lambda)\big) \arrow[d, mapsto, "p'"] \\
\big(c' \arrow[r, "f"] & c\big) \arrow[r, mapsto, "F"] & \big(F(c') \arrow[r, "F(f)"] & F(c)\big).
\end{tikzcd}$$

Here f_ℓ is the lift of f along p and λ and as $\hat{F}(f_\ell)$ is the lift of $F(f)$ along p' and $\hat{F}(\lambda)$. A similar argument proves the same statement for F^λ and $p \circ q$. Hence F is a dep-functor.

Finally, if we are given a 2-map $(\tilde{\eta}, \hat{\eta}, \eta)\colon (\tilde{F}, \hat{F}, F) \Rightarrow (\tilde{G}, \hat{G}, G)$ we obtain a dep-natural transformation $\eta\colon F \Rightarrow G$. We only need to check that $G_c(\lambda) \circ \eta_c = F_c(\lambda)$ for which we use that $G_c(\lambda) = \hat{G}(\lambda), F_c(\lambda) = \hat{F}(\lambda)$, then the statement follows from the commutativity of

$$\begin{tikzcd}
\mathscr{E} \arrow[r, bend left, "\hat{F}", ""{name=U, below}] \arrow[r, bend right, "\hat{G}"', ""{name=D}] \arrow[d, "p"'] & \mathscr{E}' \arrow[d, "p'"] \\
\mathscr{B} \arrow[r, bend left, "F", ""{name=U2, below}] \arrow[r, bend right, "G"', ""{name=D2}] & \mathscr{B}'
\arrow[Rightarrow, from=U, to=D, "\hat{\eta}"]
\arrow[Rightarrow, from=U2, to=D2, "\eta"]
\end{tikzcd}$$

which itself stems from $(\hat{\eta}, \eta)\colon p \to p'$ being a map of discrete fibrations.

At last we show that these functors constitute a 2-isomorphism. A straightforward computation shows that applying $\mathscr{C}_{-} \circ (q,p)_{-}$ to a dep-category $\mathscr{C}$ yields the same dep-category, analogously one computes the same for dep-functors and dep-natural transformations.

Conversely, applying $(q,p)_{-} \circ \mathscr{C}_{-}$ to a pair $(q\colon \mathscr{G} \to \mathscr{E}, p\colon \mathscr{E} \to \mathscr{B})$ yields (q,p) itself, the same holds for 1-maps and 2-maps in dFibb. Q.E.D.

Chapter 5
(dep,Σ)-categories and higher comprehension categories

In this chapter we recall the definition of Petrakis how to extend the dependent arrow structure to an existing Σ-object structure. We then propose a notion of higher discrete comprehension categories which are equivalent to these categories of Petrakis .

5.1 (dep,Σ)-categories

Definition 5.1.1 ((dep,Σ)-categories). A *(dep,Σ)-category* is a (fam,Σ)-category $\mathscr{C}$ that is also a dep-category together with a dependent arrow $\mathsf{pr}_2^\lambda \in \mathrm{dHom}(\lambda \circ \mathsf{pr}_1^\lambda)$ for every family arrow λ subject to the following condition:

(DΣ_1) for each $f\colon a \to b$ and each $\lambda \in \mathrm{fHom}(b)$ we have

$$\big[\,\mathsf{pr}_2^\lambda\,\big]\Big(\sum_\lambda f\Big) = \mathsf{pr}_2^{\lambda\circ f}\,.$$

This can be expressed through the commutativity of

$$\begin{tikzcd}
\sum_a \lambda\circ f \arrow[rr, "\sum_\lambda f"] \arrow[dr, "\lambda\circ f\circ\mathsf{pr}_1^{\lambda\circ f}"'{name=L}] \arrow[to=L, Rightarrow, bend right, "\mathsf{pr}_2^\lambda"'] & & \sum_b \lambda \arrow[dl, "\lambda\circ\mathsf{pr}_1^\lambda"{name=R}] \arrow[to=R, Rightarrow, bend left, "\mathsf{pr}_2^{\lambda\circ f}"] \\
& {} &
\end{tikzcd}.$$

Definition 5.1.2 ((dep,Σ)-functors). Let $\mathscr{C}, \mathscr{D}$ be (dep,Σ)-categories. We define a *(dep,Σ)-functor* from $\mathscr{C}$ to $\mathscr{D}$ to be a functor $F\colon \mathscr{C} \to \mathscr{D}$ that is both a dep-functor as well as a (fam,Σ)-functor and fulfils the following condition:

(DΣ_2) For all family arrows $\lambda \in \mathrm{fHom}(c)$ the equality $F(\mathsf{pr}_2^\lambda) = \mathsf{pr}_2^{F_c(\lambda)}$ holds.

We call F a *weak (dep,Σ)-functor* if it is a weak (fam,Σ)-functor and a dep-functor such that the following holds:

(wDΣ_2) For all $\lambda \in \mathrm{fHom}(c)$ for some c the equation $\big[\,\mathsf{pr}_2^{F_c(\lambda)}\,\big](F_\lambda^{\cong}) = F_\lambda(\mathsf{pr}_2^\lambda)$ holds.

This definition is inspired by the way Petrakis assigns to each (fam,Σ)-category a (dep,Σ)-category. The following result is due [48, Thms 4.6 & 5.4].

Theorem 5.1.3. *Every (fam,Σ)-category can be turned into a (dep,Σ)-category by setting*

$$\mathrm{dHom}(\lambda) := \left\{ \Phi\colon c \to \sum_c \lambda \;\middle|\; \begin{tikzcd} c \arrow[r, "\Phi"] \arrow[dr, "1_c"'] & \sum_c \lambda \arrow[d, "\mathsf{pr}_1^\lambda"] \\ & c \end{tikzcd} \;\textit{commutes} \right\}$$

and letting pr_2^λ *be the arrow defined through the commutative diagram*

$$\begin{tikzcd}
\sum_c \lambda \arrow[rrd, bend left, "1_{\sum_c \lambda}"] \arrow[rd, "\mathsf{pr}_2^\lambda"] \arrow[rdd, bend right, "1_{\sum_c \lambda}"'] & & \\
& \sum\limits_{\sum_c \lambda} \lambda \circ \mathsf{pr}_1^\lambda \arrow[r, "\mathsf{pr}_1^{\lambda \circ \mathsf{pr}_1^\lambda}"] \arrow[d, "\mathsf{pr}_1^{\lambda \circ \mathsf{pr}_1^\lambda}"] & \sum_c \lambda \arrow[d, "\mathsf{pr}_1^\lambda"] \\
& \sum_c \lambda \arrow[r, "\mathsf{pr}_1^\lambda"] & c.
\end{tikzcd}$$

Thus (dep,Σ)-functors should be defined in a way that this assignment routine becomes a functor between the respective categories. We observe that if we are given two (fam,Σ)-categories $\mathscr{C}, \mathscr{D}$ and a (fam,Σ)-functor $F \colon \mathscr{C} \to \mathscr{D}$ then $F(\Phi)$ is a dep-arrow in $\mathscr{D}$ for every dep-arrow Φ in $\mathscr{C}$. Furthermore using $F\big(\sum_{\sum_c \lambda} \lambda \circ \mathsf{pr}_1^\lambda\big) = \sum_{\sum_{F(c)} F_c(\lambda)} F_c(\lambda) \circ \mathsf{pr}_1^{F_c(\lambda)}$ we can observe that $F(\mathsf{pr}_2^\lambda)$ makes the diagram

$$\begin{tikzcd}
\sum_{F(c)} F_c(\lambda) \arrow[rrd, bend left, "1_{\sum_c \lambda}"] \arrow[rd, "F(\mathsf{pr}_2^\lambda)"] \arrow[rdd, bend right, "1_{\sum_{F(c)} F_c(\lambda)}"'] & & \\
& \sum\limits_{\sum_{F(c)} F_c(\lambda)} F_c(\lambda) \circ \mathsf{pr}_1^{F_c(\lambda)} \arrow[r, "\mathsf{pr}_1^{F_c(\lambda) \circ \mathsf{pr}_1^{F_c(\lambda)}}"] \arrow[d, "\mathsf{pr}_1^{F_c(\lambda) \circ \mathsf{pr}_1^{F_c(\lambda)}}"] & \sum_{F(c)} F_c(\lambda) \arrow[d, "\mathsf{pr}_1^{F_c(\lambda)}"] \\
& \sum_{F(c)} F_c(\lambda) \arrow[r, "\mathsf{pr}_1^{F_c(\lambda)}"] & F(c)
\end{tikzcd}$$

commute and thus $\mathsf{pr}_2^{F_c(\lambda)} = F(\mathsf{pr}_2^\lambda)$. This allows us to define our (dep,Σ)-functor via

$$c \mapsto F(c), \quad \lambda \mapsto F_c(\lambda), \quad \Phi \mapsto F(\Phi).$$

It is immediate from the definition that this assignment respects both identities and composition, hence we have the desired functor $(\mathsf{fam}, \Sigma)\mathsf{C} \to (\mathsf{dep}, \Sigma)\mathsf{C}$.

For the case of weak (fam,Σ)-functors we need to work a little harder, as we only are given the isomorphisms $F_\lambda^{\cong} \colon F(\sum_c \lambda) \to \sum_{F(c)} F_c(\lambda)$, thus from our $\Phi \colon c \to \sum_c \lambda$ such that $\mathsf{pr}_1^\lambda \circ \Phi = 1_c$ we do not immediately get that $F(\Phi)$ is a dep-arrow in the induced structure instead we have to consider $F_\lambda^{\cong} \circ F(\Phi)$ as the following diagram highlights:

$$\begin{tikzcd}
F(c) \arrow[r, "F(\Phi)"] \arrow[rd, "1_{F(c)}"] & F\big(\sum_c \lambda\big) \arrow[r, "F_\lambda^{\cong}"] \arrow[d, "F(\mathsf{pr}_1^\lambda)"] & \sum_{F(c)} F_c(\lambda) \arrow[ld, "\mathsf{pr}_1^{F_c(\lambda)}"] \\
& F(c). &
\end{tikzcd}$$

This also leads to further "complications" with the second-projection $\mathsf{pr}_2^{F_c(\lambda)}$. For this we need to consider the diagram

$$\begin{tikzcd}
F(\sum_c \lambda) \arrow[rr, "F_\lambda^{\cong}"] \arrow[d, "F(\mathsf{pr}_2^\lambda)"] & & \sum_{F(c)} F_c(\lambda) \arrow[d, "\mathsf{pr}_2^{F_c(\lambda)}"] \\
F\Big(\sum\limits_{\sum_c \lambda} \lambda \circ \mathsf{pr}_1^\lambda\Big) \arrow[r, "F^{\cong}_{\lambda\circ\mathsf{pr}_1^\lambda}"] \arrow[dr, "F(\mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda})"'] & \sum\limits_{F(\sum_c \lambda)} F_{\sum_c \lambda}(\lambda \circ \mathsf{pr}_1^\lambda) \arrow[r, "\sum_{F_c(\lambda)\circ\mathsf{pr}_1^{F_c(\lambda)}} F_\lambda^{\cong}"] \arrow[d, "\mathsf{pr}_1^{F_{\sum_c \lambda}(\lambda\circ\mathsf{pr}_1^\lambda)}"] \arrow[dr, phantom, "\lrcorner", very near start] & \sum\limits_{\sum_{F(c)} F_c(\lambda)} F_c(\lambda) \circ \mathsf{pr}_1^{F_c(\lambda)} \arrow[d, "\mathsf{pr}_1^{F_c(\lambda)\circ\mathsf{pr}_1^{F_c(\lambda)}}"] \\
& F\big(\sum_c \lambda\big) \arrow[r, "F_\lambda^{\cong}"] & \sum_{F(c)} F_c(\lambda)
\end{tikzcd}$$

in which the left triangle and bottom inner square (but not the top square a priori) commute. Note that we also used that $F_{\sum_c \lambda}(\lambda \circ \mathsf{pr}_1^\lambda) = F_c(\lambda) \circ \mathsf{pr}_1^{F_c(\lambda)} \circ F_\lambda^{\cong}$. To see that the outer square commutes we compute

$$\begin{aligned}
\mathsf{pr}_1^{F_c(\lambda)\circ\mathsf{pr}_1^{F_c(\lambda)}} \circ \mathsf{pr}_2^{F_c(\lambda)} \circ F_\lambda^{\cong} &= 1_{\sum_{F(c)} F_c(\lambda)} \circ F_\lambda^{\cong} && \text{(defining property of } \mathsf{pr}_2^{F_c(\lambda)}\text{)} \\
&= F_\lambda^{\cong} \circ F(1_{\sum_c \lambda}) && \text{(properties of identities)} \\
&= F_\lambda^{\cong} \circ F(\mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda} \circ \mathsf{pr}_2^\lambda) && \text{(defining property of } \mathsf{pr}_2^\lambda\text{)} \\
&= F_\lambda^{\cong} \circ F(\mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda}) \circ F(\mathsf{pr}_2^\lambda) && \text{(functoriality of } F\text{).}
\end{aligned}$$

As $\big[\mathsf{pr}_2^{F_c(\lambda)}\big](F_\lambda^{\cong})$ is defined as the unique arrow $F(\sum_c \lambda) \to \sum_{F(\sum_c \lambda)} F_{\sum_c \lambda}(\lambda \circ \mathsf{pr}_1^\lambda)$ making the required triangles in the pullback diagram commute. So by showing that $F^{\cong}_{\lambda\circ\mathsf{pr}_1^\lambda} \circ F(\mathsf{pr}_2^\lambda)$ makes them commute we obtain the desire $\big[\mathsf{pr}_2^{F_c(\lambda)}\big](F_\lambda^{\cong}) = F^{\cong}_{\lambda\circ\mathsf{pr}_1^\lambda} \circ F(\mathsf{pr}_2^\lambda)$.

The equality $\mathsf{pr}_1^{F_{\sum_c \lambda}(\lambda\circ\mathsf{pr}_1^\lambda)} \circ F^{\cong}_{\lambda\circ\mathsf{pr}_1^\lambda} \circ F(\mathsf{pr}_2^\lambda) = 1_{F(\sum_c \lambda)}$ is immediate as

$$\mathsf{pr}_1^{F_{\sum_c \lambda}(\lambda\circ\mathsf{pr}_1^\lambda)} \circ F^{\cong}_{\lambda\circ\mathsf{pr}_1^\lambda} \circ F(\mathsf{pr}_2^\lambda) = F(\mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda}) \circ F(\mathsf{pr}_2^\lambda) = F(\mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda} \circ \mathsf{pr}_2^\lambda) = F(1_{\sum_c \lambda}).$$

For the second equality we use that by showing

$$\mathsf{pr}_1^{F_c(\lambda)\circ\mathsf{pr}_1^{F_c(\lambda)}} \circ \sum_{F_c(\lambda)\circ\mathsf{pr}_1^{F_c(\lambda)}} F_\lambda^{\cong} \circ F^{\cong}_{\lambda\circ\mathsf{pr}_1^\lambda} \circ F(\mathsf{pr}_2^\lambda) = \mathsf{pr}_1^{F_c(\lambda)\circ\mathsf{pr}_1^{F_c(\lambda)}} \circ \mathsf{pr}_2^{F_c(\lambda)} \circ F_\lambda^{\cong}$$

we already get the desired equality as we can the use that $\sum_{\sum_{F(c)} F_c(\lambda)} F_c(\lambda) \circ \mathsf{pr}_1^{F_c(\lambda)}$ itself is a pullback. But the postulated equality follows immediately as

$$\begin{aligned}
\mathsf{pr}_1^{F_c(\lambda)\circ\mathsf{pr}_1^{F_c(\lambda)}} \circ \sum_{F_c(\lambda)\circ\mathsf{pr}_1^{F_c(\lambda)}} F_\lambda^{\cong} \circ F^{\cong}_{\lambda\circ\mathsf{pr}_1^\lambda} \circ F(\mathsf{pr}_2^\lambda) &= F_\lambda^{\cong} \circ \mathsf{pr}_1^{F_{\sum_c \lambda}(\lambda\circ\mathsf{pr}_1^\lambda)} \circ F^{\cong}_{\lambda\circ\mathsf{pr}_1^\lambda} \circ F(\mathsf{pr}_2^\lambda) \\
&= F_\lambda^{\cong} \circ F(\mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda}) \circ F(\mathsf{pr}_2^\lambda) \\
&= \mathsf{pr}_1^{F_c(\lambda)\circ\mathsf{pr}_1^{F_c(\lambda)}} \circ \mathsf{pr}_2^{F_c(\lambda)} \circ F_\lambda^{\cong}
\end{aligned}$$

where we used the commutativity of the lower square and triangle for the first and second equality respectively and then the commutativity of the outer square shown above.

Definition 5.1.4 ((dep,Σ)-natural transformations). Let $\mathscr{C}, \mathscr{D}$ be (dep,Σ)-categories and $F, G\colon \mathscr{C} \to \mathscr{D}$ be (dep,Σ)-functors. We define a *(dep,Σ)-natural transformation* $\eta\colon F \Rightarrow G$ to be a natural transformation fulfilling the following properties:

(DΣ_3) η is a (fam,Σ)-natural transformation of the underlying (fam,Σ)-functors.

(DΣ_4) η is a dep-natural transformation of the underlying dep-functors.

(DΣ_5) for every family arrow λ we have $\mathsf{pr}_2^{F'(\lambda)} \circ \eta_{\sum_c \lambda} = \mathsf{pr}_2^{F(\lambda)}$.

Remark 5.1.5. Actually the last condition on (dep,Σ)-natural transformations is superfluous as we already have—from η being a dep-natural transformation—that

$$F'(\mathsf{pr}_2^\lambda) \circ \eta_{\sum_c \lambda} = F(\mathsf{pr}_2^\lambda),$$

so using $F'(\mathsf{pr}_2^\lambda) = \mathsf{pr}_2^{F'(\lambda)}$ and the analogous equality for F yields the desired condition.

Definition 5.1.6 ((2-)Category of (dep,Σ)-categories). Let $(\mathsf{dep}, \Sigma)\mathsf{C}$ be the category with

- *Objects:* (dep,Σ)-categories $\mathscr{C}$,
- *1-maps:* (dep,Σ)-functors F.
- *2-maps:* (dep,Σ)-natural transformations η.

5.2 Higher discrete comprehension categories

5.2.1 Alternative viewpoints on comprehension categories. Before we proceed we note the following observation made by Jacobs in [33, p. 4.1.2]: The object part of a comprehension category $P\colon \mathscr{E} \to \mathscr{B}^{\to}$ defines a natural transformation $P\colon P_0 \Rightarrow p$. We will confuse P with this natural transformation it gives rise to, it will be obvious from the context which of the two notions we refer to.

Definition 5.2.2 (Higher discrete comprehension categories). A higher discrete comprehension category consists of the following data:

- a discrete comprehension category $P\colon \mathscr{E} \to \mathscr{B}^{\to}$.
- An iterated discrete fibration $(q\colon \mathscr{G} \to \mathscr{E}, p\colon \mathscr{E} \to \mathscr{B})$ where p stems from the comprehension category P.
- A functor $\mathsf{pr}_2\colon \mathscr{E} \to \mathscr{G}$ and a natural transformation $\chi\colon q \circ \mathsf{pr}_2 \Rightarrow \mathrm{id}_{\mathscr{E}}$.

This data will be written as (q, P, χ). It is subject to the following condition:

(HdCC$_1$) $p \bullet \chi = P$, that is for all $\lambda \in \mathscr{E}$ we have $p(\chi_\lambda) = P(\lambda)$.

Definition 5.2.3 (Pseudo-maps of higher discrete comprehension categories). Let $(q, P, \chi), (q', P', \chi')$ be two higher discrete comprehension categories. A *pseudo-map* from (q, P, χ) to $(q', P'.\chi')$ is a quadruple $(\hat{F}, F, G, \phi)$ such that

(HdCC$_2$) $(\hat{F}, F, G)\colon (q, p) \to (q', p')$ is a map of iterated discrete fibrations and

(HdCC$_3$) (F, G, ϕ) is a pseudo-map of comprehension categories.

(HdCC$_4$) $\mathsf{pr}_2' \circ F = \hat{F} \circ \mathsf{pr}_2$.

If ϕ is the identity such that (F, G, ϕ) is a strict map of comprehension categories we say that $(\hat{F}, F, G, \phi)$ is a *strict map* of higher discrete comprehension categories and drop the ϕ from the notation.

Definition 5.2.4 (Transformation of pseudo-maps). Let $(\hat{F}, F, G, \phi), (\hat{F}', F', G', \phi')\colon (q, P, \chi) \to (q', P', \chi')$ be two pseudo-maps between two higher discrete comprehension categories. We define a transformation of such pseudo-maps to be a triple $(\hat{\eta}, \tilde{\eta}, \eta)$ where

(HdCC$_5$) $(\tilde{\eta}, \eta)$ is a transformation of pseudo-maps of comprehension categories (see Definition 3.2.5).

(HdCC$_6$) $(\hat{\eta}, \eta)$ is a transformation of functors of discrete fibrations.

(HdCC$_7$) $\mathsf{pr}_2' \bullet \tilde{\eta} = \hat{\eta} \bullet \mathsf{pr}_2$.

Remark 5.2.5. Similar to (dep,Σ)-natural transformation we can see that the last condition imposed on Transformations of pseudo-maps is superfluous in the case of the pseudo-map being strict. For this we first observe that for each $b \in \mathscr{B}$ and $e \in \mathscr{E}$ with $p(e) = b$ we obtain a commutative square

$$\begin{tikzcd}
q\big(\mathsf{pr}_2^{F(e)}\big) \arrow[r, "q(\mathsf{pr}_2(\tilde{\eta}_e))"] \arrow[d, "\chi_{F(e)}"] & q\big(\mathsf{pr}_2^{F'(e)}\big) \arrow[d, "\chi_{F'(e)}"] \\
F(e) \arrow[r, "\tilde{\eta}_e"] & F'(e)
\end{tikzcd}$$

in $\mathscr{E}$. Applying p on the above square must yield

$$\begin{tikzcd}
G\big(\sum_b e\big) \arrow[r, "\eta_{\sum_b e}"] \arrow[d, "\mathsf{pr}_1^{F(e)}"'] & G'\big(\sum_b e\big) \arrow[d, "\mathsf{pr}_1^{F'(e)}"] \\
G(b) \arrow[r, "\eta_b"] & G'(b)
\end{tikzcd}$$

in $\mathscr{B}$ as p is a discrete fibration, so $q(\mathsf{pr}_2(\tilde{\eta}_e))$ has to be the unique lift of $\eta_{\sum_c e}$ with respect to $q\big(\mathsf{pr}_2^{F'(e)}\big)$. But $\hat{\eta}_{F(\mathsf{pr}_2^e)} \colon F(\mathsf{pr}_2^e) \to F'(\mathsf{pr}_2^e)$ and $F'(\mathsf{pr}_2^e) = \mathsf{pr}_2^{F'(e)}$, hence $p \circ q$ being a discrete fibration implies $\hat{\eta}_{F(\mathsf{pr}_2^e)} = \mathsf{pr}_2^{F(e)}$.

Definition 5.2.6 (2-category of Higher discrete comprehension categories). We define the 2-category $\mathsf{HComprC}_{\mathrm{disc}}$ the be the 2-category with

- *Objects:* Higher discrete comprehension categories (q, P, χ),
- *1-maps:* Pseudo-maps $(\hat{F}, F, F, \phi)$ between comprehension categories.
- *2-maps:* Transformation of pseudo-maps.

This category has a subcategory $\mathsf{HComprC}_{\mathrm{disc}}^{\mathrm{str}}$ with the same objects but the 1- and 2-maps restricted to the strict cases.

Theorem 5.2.7. *There is a 2-equivalence*

$$\mathsf{HComprC}_{\mathrm{disc}}^{\mathrm{str}} \simeq (\mathsf{dep}, \Sigma)\mathsf{C}\,.$$

5.2.8 Products in comprehension categories. Before moving on we must talk about the notion of a comprehension category with products, as in [35, §5], and how they relate to our notion of dependent arrows over (2-fam,Σ)-categories. Jacobs defines—in [35]—products in comprehension categories as a property rather than additional structure, namely a comprehension category has products, if the functor $\langle P \rangle$ has a fibred right adjoint. Here $\langle P \rangle$ is obtained by first considering the inclusion $\iota\colon \mathsf{Cart}(\mathscr{E}) \hookrightarrow \mathscr{E}$ of the wide subcategory of objects and p-cartesian morphism of $\mathscr{E}$, and then the pullbacks (under identification

$|p| := p \circ \iota, |P_0| := P_0 \circ \iota)$ as in the following diagram:

$$\begin{tikzcd}
{|p|^{-1}(\mathscr{E})} \arrow[rrr, bend left, "p^*|p|"] \arrow[dr, "\langle P\rangle"] \arrow[ddr, bend right, "|p|^*p"'] & & & \\
& {|P_0|^{-1}(\mathscr{E})} \arrow[rr, shift left, "p^*|p|"] \arrow[rr, shift right, "p^*|P_0|"'] \arrow[d, "|P_0|^*p"'] & & \mathcal{E} \arrow[d, "p"] \\
& \mathsf{Cart}(\mathscr{E}) \arrow[rr, shift left, "|p|"] \arrow[rr, shift right, "|P_0|"'] & & \mathcal{B}.
\end{tikzcd}$$

Note that $\langle P\rangle$ is *not* defined by invoking the universal property of one of the pullbacks, but instead using [19, Proposition 3]—the explicit construction will be recalled shortly. If we consider only the discrete case, every arrow becomes cartesian, and the inclusion ι the identity. In this case the functor $\langle P\rangle$ sends (e, e') with $p(e) = p(e')$ to $(P^*(e), e')$. An arrow $(f, g)\colon (e_1, e'_1) \to (e_2, e'_2)$ gets mapped to the unique arrow arising from first considering

$$\begin{tikzcd}
P_0(e'_1) \arrow[r, "P(e'_1)"] \arrow[d, "P_0(g)"'] & p(e'_1) \arrow[d, "p(g)"] \\
P_0(e'_2) \arrow[r, "P(e'_2)"'] & p(e'_2)
\end{tikzcd}$$

and observing that $p\big(\overline{P}(e_1)\big) = P_0(e'_1)$, hence

$$p(g) \circ P(e'_1) = p\big(g \circ \overline{P}(e_1)\big).$$

Invoking the universal property of $\overline{P}(e_2)$ on the situation as in the following diagram,

$$\begin{tikzcd}
& e_1 \arrow[rr, mapsto] \arrow[dl, dotted, "h"'] \arrow[dd, "g\circ\overline{P}(e_2)"] & & P_0(e'_1) \arrow[dl, "P_0(g)"'] \arrow[dd, "p(g\circ\overline{P}(e_2))"] \\
P^*(e_2) \arrow[rr, mapsto] \arrow[dr, "\overline{P}(e_2)"'] & & P_0(e'_2) \arrow[dr, "P(e'_2)"'] & \\
& e_2 \arrow[rr, mapsto] & & p(e'_2),
\end{tikzcd}$$

yields an unique arrow $h\colon e_1 \to P^*(e_2)$. Then define $\langle P\rangle(f, g) = (h, g)$.

In this discrete case a fibred adjunction $\langle P\rangle \dashv \prod$ is very restrictive, as vertical transformations can only consist of identities, as only those are mapped to identities by discrete fibrations. Thus for a product $\prod$ to exist we need that $\langle P\rangle \circ \prod = \mathrm{id}_{|P_0|^{-1}(\mathscr{E})}$ and $\prod \circ \langle P\rangle = \mathrm{id}_{|p|^{-1}(\mathscr{E})}$.

Chapter 6
2-dep-arrows and discrete ambifibrations

In this chapter we describe the relation between dependent arrows in 2-fam-categories and ambifibrations as defined by Sattler in [60]. Originally we were motivated by the known result that each Grothendieck fibration yields an orthogonal factorisation system on its total category into its vertical and cartesian morphisms. Building on this result we defined a special kind of fibration which has an oplifting property with respect to the vertical and a lifting property with respect to the cartesian morphisms. By coincidence we read in [38] that Sattler has defined such a concept in [60] – ambifibrations – which lead to a ternary factorisation system on a category even higher up. However, we do not need the full abstraction of this approach and only use what we will henceforth call *discrete ambifibrations*. In the first section we recall the notion of a 2-dep-category due to Ehrhardt , in the next section we give a brief overview of orthogonal factorisation systems and ambifibrations and then use them to define an abstraction of the iterated discrete fibrations which we will use in our 2-equivalence concerning the category of 2-dep-categories.

6.1 2-dep-categories

Definition 6.1.1 (2-dep-categories). A *2-dep-category* is a 2-fam-category $\mathscr{C}$ together with a collection $\mathrm{dHom}(\lambda)$ for every family arrow λ (over some arbitrary $c \in \mathscr{C}$)) and

- an assignment rule that assigns to every $\Phi \in \mathrm{dHom}(\lambda)$ and $f\colon c' \to c$ a dep-arrow $\Phi[f] \in \mathrm{dHom}(\lambda \circ f)$ and
- an assignment rule that assigns to every $\Phi \in \mathrm{dHom}(\lambda)$ and $\eta\colon \lambda \to \mu$ a dep-arrow $\eta \circ \Phi \in \mathrm{dHom}(\mu)$.

These rules must fulfil the following conditions:

(2dep$_1$) for every $\Phi \in \mathrm{dHom}(\lambda)$ and $f\colon c'' \to c, g\colon c' \to c$ both

$$[\Phi](g \circ f) = \big[[\Phi](g)\big](f) \quad \text{and} \quad [\Phi](1_c) = \Phi.$$

(2dep$_2$) For every $\Phi \in \mathrm{dHom}(\lambda)$ and $\eta\colon \lambda \Rightarrow \mu, \xi\colon \mu \Rightarrow \nu$ both

$$(\xi \circ \eta) \circ \Phi = \xi \circ (\eta \circ \Phi) \quad \text{and} \quad 1_\lambda \circ \Phi = \Phi.$$

(2dep$_3$) For every $\Phi \in \mathrm{dHom}(\lambda)$ and $\eta\colon \lambda \Rightarrow \mu, f\colon c' \to c$

$$[\eta \circ \Phi](f) = (\eta \circ f) \circ [\Phi](f).$$

Definition 6.1.2 (2-dep-functors). Let $\mathscr{C}, \mathscr{C}'$ be 2-dep-categories. We define a *2-dep-functor* to be a 2-fam-functor $F\colon \mathscr{C} \to \mathscr{C}'$ of the underlying 2-fam-categories together with an assignment routine F_λ for every family arrow λ that assigns $\Phi \in \mathrm{dHom}(\lambda)$ to $F_\lambda(\Phi) \in \mathrm{dHom}\left(F_c(\lambda)\right)$. The following conditions must be met:

(2dep$_4$) for every $f\colon c \to c', \lambda \in \mathrm{fHom}(c)$ and $\Phi \in \mathrm{dHom}(\lambda)$ the following holds:

$$F_\lambda(\Phi \circ f) = F_\lambda(\Phi) \circ F(f).$$

(2dep$_5$) For every $f\colon c \to c', \eta\colon \lambda \to \mu$ where $\lambda, \mu \in \mathrm{fHom}(c)$ and $\Phi \in \mathrm{dHom}(\lambda)$ the equality

$$F_\lambda(\eta \circ \Phi) = F_c(\eta) \circ F_\lambda(\Phi)$$

holds.

Definition 6.1.3 (2-dep-natural transformations). Let $\mathscr{C}, \mathscr{C}'$ be 2-dep-categories, $F, G\colon \mathscr{C} \to \mathscr{C}'$ be 2-dep-functors. We define a *2-dep-natural transformation* to be a 2-fam-natural transformation $\eta\colon F \Rightarrow G$ of the underlying 2-fam-functors (see Definition 2.3.4) that fulfils the following additional condition:

(2dep$_6$) For all $c \in \mathscr{C}, \lambda \in \mathrm{fHom}(c)$ and $\Phi \in \mathrm{dHom}(\lambda)$ we have $G_\lambda(\Phi) \circ \eta_c = F_\lambda(\Phi)$, that is

$$\begin{tikzcd}
F(c) \arrow[rrr, "\eta_c"] \arrow[d, Rightarrow, "F_\lambda(\Phi)"'] \arrow[dr, "F_c(\lambda)"] & & & G(c) \arrow[dl, "G_c(\lambda)"'] \arrow[d, Rightarrow, "G_\lambda(\Phi)"] \\
{} & {} & {} & {}
\end{tikzcd}$$

commutes.

Definition 6.1.4 (2-Category of 2-dep-categories). We define the category 2-depC to have

- objects 2-dep-categories,
- 1-cells 2-dep-functors and
- 2-cells 2-dep-natural transformations.

6.2 Ambifibrations and factorisation systems

Before we can define ambifibrations (and the required discrete variant) we need to introduce orthogonal factorisation systems. Those were introduced by Freyd and Kelly in [23] and later the weakened version was introduced by Bousfield in [8]. but we will follow the presentation of Riehl in [54] and [55].

Definition 6.2.1 (Lifting problems). A lifting problem is a commutative square

$$\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "l"] & b \arrow[d, "r"] \\
c \arrow[r, "g"] & d
\end{tikzcd} \tag{11}$$

in a category $\mathscr{C}$. Such a lifting problem has a solution if there exists an arrow h in $\mathscr{C}$ making the following diagram

$$\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "l"] & b \arrow[d, "r"] \\
c \arrow[r, "g"] \arrow[ur, "h"] & d
\end{tikzcd}$$

commutative. If l, r are fixed and every possible lifting problem as in 11 has a solution, one writes $l \boxslash r$. If all solutions to such lifting problems are unique one writes $l \perp r$.

One then defines

$$\begin{aligned}
M^{\boxslash} &:= \{c \in \mathscr{C} \mid \forall_{m \in M} m \boxslash c\}, \\
{}^{\boxslash}M &:= \{c \in \mathscr{C} \mid \forall_{m \in M} c \boxslash m\},
\end{aligned}$$

analogously for "$\perp$" instead of "$\boxslash$".

Definition 6.2.2 (Factorisation systems). A *weak factorisation system* $(\mathcal{L},\mathcal{R})$ on a category $\mathscr{C}$ consists of two classes $\mathcal{L},\mathcal{R}$ of objects of $\mathscr{C}$ such that

1. $\mathcal{L} = {}^{\boxslash}\mathcal{R}$,
2. $\mathcal{R} = \mathcal{L}^{\boxslash}$ and
3. every arrow f in $\mathscr{C}$ factors as $f = rl$ where $l \in \mathcal{L}$ and $r \in \mathcal{R}$.

A *weak factorisation system* is an *orthogonal factorisation system* if the first two properties hold with "$\perp$" instead of "$\boxslash$".

Factorisation systems were introduced (independently) in [46],[31].

Examples 6.2.3. 1. The standard orthogonal factorisation system (Epi, Mon) on Sets consists of the class Epi of surjective maps and Mon of injective maps.

2. For every category $\mathscr{C}$ one can define the orthogonal factorisation system $(\mathbf{1},\mathscr{C}_1)$ where $\mathbf{1}$ is the class of all identities in $\mathscr{C}$ and $\mathscr{C}_1$ is the class of all morphisms.

The above two examples already show that for any given category there might be multiple factorisation systems on it, so factorisation systems provide additional data not encompassed by their underlying categories.

6.2.4 Orthogonal factorisation systems from fibrations. Given a Grothendieck fibration $p\colon \mathscr{E} \to \mathscr{B}$ it is known that there is a orthogonal factorisation system $(\mathcal{V},\mathcal{C})$ on $\mathscr{E}$ where $\mathcal{V}$ consists of all morphisms vertical with respect to p, that is they are mapped to isomorphisms by p, and $\mathcal{C}$ consists of all morphisms cartesian with respect to p.

Note that the vertical morphisms here are different from the ones defined in much of the established literature in that they get mapped to isomorphisms and not identities! This change is required as one would otherwise only have $\mathcal{V} \subsetneq {}^{\boxslash}\mathcal{C}$ in case $\mathscr{B}$ has isomorphisms that are not identities.

Remark 6.2.5. A factorisation system $(\mathcal{L},\mathcal{R})$ on a category $\mathscr{C}$ can be seen as two subcategories $\mathcal{L},\mathcal{R}$ of $\mathscr{C}$ whose morphisms are those in $\mathcal{L}$ and $\mathcal{R}$ respectively. Those subcategories are full as both $\mathcal{L}$ and $\mathcal{R}$ have to contain all isomorphisms.

Definition 6.2.6 (Ambifibrations). Let $\mathscr{B}$ be a category with orthogonal factorisation system $(\mathcal{L},\mathcal{R})$. Given a functor $p\colon \mathscr{E} \to \mathscr{B}$ define $p_{\mathcal{L}}, p_{\mathcal{R}}$ through the pullbacks

$$\begin{tikzcd}
\mathscr{E}_{\mathcal{L}} \arrow[r] \arrow[d, "p_{\mathcal{L}}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathscr{E} \arrow[d, "p"] & \text{and} & \mathscr{E}_{\mathcal{R}} \arrow[r] \arrow[d, "p_{\mathcal{R}}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathscr{E} \arrow[d, "p"] \\
\mathcal{L} \arrow[r] & \mathscr{B} & & \mathcal{R} \arrow[r] & \mathscr{B}.
\end{tikzcd}$$

Then p is called

- an *ambifibration* if both $p_{\mathcal{L}}$ is a Grothendieck opfibration and $p_{\mathcal{R}}$ is a Grothendieck fibration,
- a *discrete ambifibration* if both $p_{\mathcal{L}}$ is a discrete opfibration and $p_{\mathcal{R}}$ is a discrete fibration.

We first show that the choice of pullback is not relevant for this definition, that is if we are given two pullbacks

$$\begin{tikzcd}
\mathscr{E}_{\mathcal{L}} \arrow[r] \arrow[d, "p_{\mathcal{L}}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathscr{E} \arrow[d, "p"] & \text{and} & \mathscr{E}'_{\mathcal{L}} \arrow[r] \arrow[d, "p'_{\mathcal{L}}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathscr{E} \arrow[d, "p"] \\
\mathcal{L} \arrow[r] & \mathscr{B} & & \mathcal{L} \arrow[r] & \mathscr{B}
\end{tikzcd} \tag{12}$$

(similarly for $p_{\mathcal{R}}$ in place of $p_{\mathcal{L}}$) then one being a opfibration implies the other being one and vice versa. Explicitly:

Lemma 6.2.7. *In the situation of 12 the following are equivalent:*

1. *$p_{\mathcal{L}}$ is a (split) opfibration.*
2. *$p'_{\mathcal{L}}$ is a (split) opfibration.*

Proof: Assuming 1 we use that there exists a unique isomorphism $i\colon \mathscr{E}_{\mathcal{L}} \to \mathscr{E}'_{\mathcal{L}}$ by the universal property of pullbacks, so for any arrow $l\colon b \to b'$ in $\mathcal{L}$ and any $e \in \mathscr{E}'_{\mathcal{L}}$ with $p'_{\mathcal{L}}(e) = b$ we define the opcartesian lift to be $i^{-1}(\overline{l}(e))$—where $\overline{l}(e)$ is the opcartesian lift with respect to $p_{\mathcal{L}}$. It is immediate that this is a splitting from the functoriality of i and the opcartesian property of the arrows follows from the opcartesian property of the arrow in $\mathscr{E}_{\mathcal{L}}$ in conjunction with i being a isomorphism.

The reverse direction follows from a similar argument. Q.E.D.

Alternative proof: Alternatively we can build the pasted diagram

$$\begin{tikzcd}
\mathscr{E}_{\mathcal{L}} \arrow[r, "i", "\cong"'] \arrow[d, "p_{\mathcal{L}}"'] & \mathscr{E}'_{\mathcal{L}} \arrow[r] \arrow[d, "p'_{\mathcal{L}}"] \arrow[dr, phantom, "\lrcorner", very near start] & \mathscr{E} \arrow[d, "p"] \\
\mathcal{L} \arrow[r, "\mathrm{id}_{\mathcal{L}}", "\cong"'] & \mathcal{L} \arrow[r] & \mathscr{B}
\end{tikzcd}$$

where the outher and the right square are pullbacks and thus the left is as well. As opfibrations are stable under pullbacks $p_{\mathcal{L}}$ has to be an opfibration if $p'_{\mathcal{L}}$ is. For the reverse direction we observe that both i and $\mathrm{id}_{\mathcal{L}}$ are isomorphisms, so we can exchange $\mathscr{E}_{\mathcal{L}}$ and $\mathscr{E}'_{\mathcal{L}}$ to obtain the desired result. Q.E.D.

Naturally this extends to a dual result for $p_{\mathcal{R}}$ which we will not state explicitly here. These two results allow us to always check the ambifibration property through the categories $\mathscr{E}_{\mathcal{L}}, \mathscr{E}_{\mathcal{R}}$ explicitly defined as follows:

- The both share the same objects as $\mathscr{B}$.
- The arrows in $\mathscr{E}_{\mathcal{L}}$ are those arrows f in $\mathscr{E}$ such that $p(f) \in \mathcal{L}$, similarly those in $\mathscr{E}_{\mathcal{R}}$ are those projected to an arrow in $\mathcal{R}$ under p.

How does this relate to 2-dep-categories? First note that in all earlier equivalences the precompositions of fam- and dep-arrows were solved through some lifting property: the precomposition of Φ or λ with f was defined as a cartesian lift with respect to a (discrete/split) fibration.

$$\begin{tikzcd}
c' \arrow[r, "f"'] \arrow[rr, dashed, bend left, "\lambda\circ f"] & c \arrow[r, "\lambda"'] & {}
\end{tikzcd}
\quad\rightsquigarrow\quad
\begin{tikzcd}
f^*(\lambda) \arrow[r, dashed, "\overline{f}(\lambda)"] \arrow[d, mapsto, "p"] & \lambda \arrow[d, mapsto, "p"] \\
c' \arrow[r, "f"] & c.
\end{tikzcd}$$

However, 2-dep-categories introduce a way to *post*compose dep-arrows with arrows between family arrows, thus what is required is an *op*cartesian lift with respect to this arrow between family arrows.

$$\begin{tikzcd}
& |[alias=L]| \lambda & \\
c \arrow[ru, Rightarrow, bend left, "\Phi"] \arrow[rd, Rightarrow, dashed, bend right, "\eta\circ\Phi"'] \arrow[rr, bend left=40] \arrow[rr, bend right=40] & & {} \\
& \mu \arrow[from=L, Rightarrow, "\eta"] &
\end{tikzcd}
\quad\rightsquigarrow\quad
\begin{tikzcd}
\Phi \arrow[r, dashed, "\underline{\eta}(\Phi)"] \arrow[d, mapsto, "q"] & \eta_*(\Phi) \arrow[d, mapsto, "q"] \\
(c, \lambda) \arrow[r, "{(1_c,\eta)}"] & (c, \mu).
\end{tikzcd}$$

So if we obtain that our vertical morphisms are the left class of an orthogonal factorisation system on the category obtained from translating the 2-fam-structure into a Grothendieck fibration and the cartesian morphisms are the right class of that factorisation system, the property of an ambifibration yields the desired (op-)cartesian lifts. In Lemma 3.2.12 we already saw that the cartesian morphisms with respect to the obtained split fibration are (f, η) where η is an isomorphisms. But this is not a real obstacle as the opcartesian lift of a dep-arrow Φ along this map would simply consist of first lifting in a cartesian way along f^{-1} and then in an opcartesian way along η.

For the vertical arrows with respect to the obtained split fibration we have the following characterisation, whose proof is analogous to the cartesian case and is thus ommited.

Lemma 6.2.8. *Let $\mathscr{C}$ be a 2-fam-category and $p_{\mathscr{C}} \colon \mathscr{F} \to \mathscr{C}$ be the resulting split fibration. Then the class $\mathcal{V}$ of morphisms vertical with respect to $p_{\mathscr{C}}$ can be described as such:*

$$\mathcal{V} = \{(f, \eta) \mid f \textit{ is an isomorphism}\}.$$

Similarly to before, the fact that f is an isomorphism and not the identity does not bother us as we can simply postcompose with f^{-1} before taking the oplift.

A small remark about notation: in the following definition we will indicate the (ambi-)fibration to which the (op-)lift belongs in the (sub-/super-)script, so the lift of $f \colon c \to c'$ along e with respect to p will be denoted by $\overline{f}^p(e) \colon f^{*p}(e) \to e$ (and $\underline{f}_p(e) \colon f_{*p}(e) \to e$ for the oplift).

Definition 6.2.9 (Sas-tower). We define a *sas-tower* to consist of two functors $q \colon \mathscr{G} \to \mathscr{E}, p \colon \mathscr{E} \to \mathscr{B}$ such that

(sas$_1$) p is a split Grothendieck fibration,

(sas$_2$) q is a discrete ambifibration with respect to the orthogonal factorisation system $(\mathcal{V}, \mathcal{C})$ of morphisms vertical/cartesian with respect to p and

(sas$_3$) $p \circ q$ is a split Grothendieck fibration such that for every $f \colon c \to c'$ in $\mathscr{B}$, every $e \in \mathscr{E}$ with $p(e) = c'$ and every $\Phi \in \mathscr{G}$ with $q(\Phi) = e$ we have

$$\overline{f}^{p \circ q}(\Phi) = \overline{\overline{f}^p(e)}^q(\Phi).$$

(sas$_4$) For any $f \colon c \to c'$ in $\mathscr{B}$, $g \colon e \to e'$ in $\mathscr{E}$ with $p(g) = 1_{c'}$ and any $\Phi \in \mathscr{G}$ with $q(\Phi) = e$ we have

$$f^{*(p \circ q)}\big(g_{*q}(\Phi)\big) = (g \bullet f)_{*q}(f^{*(p \circ q)}(\Phi))$$

where $g \bullet f$ is defined through the universal property of the cartesian lifts of f as in

$$\begin{tikzcd}
f^*(e) \arrow[r, "\overline{f}^p(e)"] \arrow[d, dotted, "g \bullet f"'] & e \arrow[d, "g"] \\
f^*(e') \arrow[r, "\overline{f}^p(e')"] & e.
\end{tikzcd}$$

(see the proof of Proposition 2.4.14 for more details.)

Lemma 6.2.10. *We can associate to any 2-dep-category a sas-tower in the following way:*

- *The categories $\mathscr{E}, \mathscr{B}$ and the split fibration $p \colon \mathscr{E} \to \mathscr{B}$ are defined as in the translation between 2-fam-categories and split fibrations (see 2.4.14).*
- *The category $\mathscr{G}$ has*
  - *as objects dep-arrows Φ over arbitrary λ and*
  - *a morphism $(f, \eta) \colon \Phi \to \Psi$ consists of an arrow $(f, \eta) \colon (c, \lambda) \to (c', \mu)$ in $\mathscr{E}$ such that $\Phi \in \mathrm{dHom}(\lambda), \Psi \in \mathrm{dHom}(\mu)$ and*

$$\eta \circ \Phi = [\Psi](f).$$

- *The functor* $q\colon \mathscr{G} \to \mathscr{E}$ *takes each* Φ *to the pair* (c, λ) *such that* $\Phi \in \mathrm{dHom}(\lambda)$ *and* $(f, \eta)\colon \Phi \to \Psi$ *to* (f, η).

Proof: We have already seen in Proposition 2.4.14 that $p\colon \mathscr{E} \to \mathscr{B}$ is a split fibration. To see that $p \circ q\colon \mathscr{G} \to \mathscr{B}$ is a split fibration we observe that defining $\overline{f}(\Phi) := (f, 1_\lambda)\colon \Phi \circ f \to \Phi$ yields a splitting—the proof that $(f, 1_\lambda)$ is cartesian for f and Φ is analogous to the proof that it is cartesian for f and λ—and by definition of a 2-dep-category this cleavage fulfils functoriality, thus $p \circ q$ is a split fibration. From the Lemmata 3.2.12 and 6.2.8 we know that the orthogonal factorisation system on $\mathscr{E}$ in vertical and cartesian morphisms looks as follows:

$$\mathcal{V} = \{(f, \eta) \mid f \text{ is an isomorphism}\},$$
$$\mathcal{C} = \{(f, \eta) \mid \eta \text{ is an isomorphism}\}.$$

Thus we need to exhibit the discrete lifting (oplifting) property for $q_\mathcal{C}$ (respective $q_\mathcal{V}$). Given an arrow in $\mathcal{V}$, that is $(f, \eta)\colon (c, \lambda) \to (c', \mu)$— with f invertible—and $\Phi \in \mathrm{dHom}(\lambda)$ the lift of (f, η) along Φ is defined to be $(f, \eta)\colon \Phi \to [\eta \circ \Phi](f^{-1})$. This is an arrow in $\mathscr{G}$ as

$$\big[[\eta \circ \Phi](f^{-1})\big](f) = [\eta \circ \Phi](f^{-1} \circ f) = \eta \circ \Phi.$$

It is the necessarily unique lift of (f, η) with respect to $q_\mathcal{V}$.

Similarly, given an arrow in $\mathcal{C}$, that is $(f, \eta)\colon (c, \lambda) \to (c', \mu)$—with η invertible—and $\Phi \in \mathrm{dHom}(\mu)$ the lift of (f, η) along Φ is defined to be $(f, \eta)\colon \eta^{-1} \circ \Phi[f] \to \Phi$. This is a well-defined arrow as

$$\eta \circ \big(\eta^{-1} \circ \Phi[f]\big) = (\eta \circ \eta^{-1}) \circ \Phi[f] = \Phi[f].$$

It is also the necessarily unique lift of (f, η) with respect to $q_\mathcal{C}$.

It remains to check the equalities imposed upon the splitting. We compute that

$$\overline{f}^{p \circ q}(\Phi) = \big((f, 1_\lambda)\colon \Phi \circ f \to \Phi\big) = \overline{(f, 1_\lambda)}^{q}(\Phi) = \overline{\overline{f}^{p}(e)}^{q}(\Phi).$$

Similarly for $f\colon c' \to c, \eta\colon \lambda \to \mu$ with $p(\eta) = 1_c$ and Φ with $(p \circ q)(\Phi) = c'$ we compute

$$f^{*(p \circ q)}\big((1_c, \eta)_{*q}(\Phi)\big) = f^{*(p \circ q)}\big(\eta \circ \Phi)\big) = [\eta \circ \Phi](f) \text{ and}$$
$$(1_c, \eta) \bullet f_{*q}\big(f^{*(p \circ q)}(\Phi\big) = (1_{c'}, \eta \bullet f)_{*q}(\Phi \circ f) = \eta \bullet f) \circ \Phi[f].$$

As $[\eta \circ \Phi](f) = (\eta \bullet f) \circ \Phi[f]$ in any 2-dep-category this proves the desired equality. Q.E.D.

Conversely every sas-tower gives rise to a 2-dep-category.

Lemma 6.2.11. *Let* $(q\colon \mathscr{G} \to \mathscr{E}, p\colon \mathscr{E} \to \mathscr{B})$ *be a sas-tower. Then we obtain an associated 2-dep-category* $\mathscr{C}$ *whose*

- *underlying fam-category* $\mathscr{C}$ *is the category obtained from* $p\colon \mathscr{E} \to \mathscr{B}$ *as previously.*
- *The dependent arrows over* $e \in \mathscr{E}$ *are defined as* $q^{-1}(e)$.

The compositions of $f\colon c' \to c, e \in \mathrm{fHom}(c), g\colon e' \to e \in \mathrm{fHom}(c), \Phi \in \mathrm{dHom}(e)$ *are defined as*

$$[\Phi](f) := f^*(\Phi) \qquad \textit{the lift with respect to } p \circ q.$$
$$g \circ \Phi := g_*(\Phi) \qquad \textit{the lift with respect to } q_\mathcal{V}.$$

Proof: We already know that we can obtain a 2-fam-category from the split fibration $p\colon \mathscr{E} \to \mathscr{B}$. To see that it is a 2-dep-category we first observe that the functoriality of composition $[\Phi](f)$ and $g \circ \Phi$ is guaranteed as both are lifts with respect to discrete fibrations. To check $[g \circ \Phi](f) = (g \circ f) \circ \Phi[f]$ we simply note that both are mapped to the same element of $\mathscr{E}$ by q and thus must be the same.

Q.E.D.

The notion of an arrow between sas-towers will be the expected one, but before we can state it we need a few technical Lemmata.

Lemma 6.2.12. *Let $\mathscr{C}$ be a category and let maps as in the diagram*

$$\begin{tikzcd}
a \arrow[rr, "f_1"] \arrow[dd, "f_2"'] & & \cdot \arrow[dr, "g_2"] \arrow[dd, "g_1"] & \\
& c \arrow[rr, "h_1"] \arrow[dd, "h_2"] \arrow[dr, phantom, "\lrcorner", very near start] & & \cdot \arrow[dd, "k"] \\
\cdot \arrow[rr, "l_1"] \arrow[dr, "l_2"'] & & \cdot \arrow[dr, "m"] & \\
& \cdot \arrow[rr, "n"] & & \cdot
\end{tikzcd}$$

be given such that all faces commute. Then we obtain a map $f_3\colon a \to c$ making the remaining face commutative.

Proof: This follows as

$$k \circ g_2 \circ f_1 = m \circ g_1 \circ f_1 = m \circ l_1 \circ f_2 = n \circ l_2 \circ f_2,$$

so we can use the universal property of the pullback at c. Q.E.D.

Corollary 6.2.13. *Given two ambifibrations $q\colon \mathscr{E} \to \mathscr{B}, q'\colon \mathscr{E}' \to \mathscr{B}'$ and two functors $\hat{F}\colon \mathscr{E} \to \mathscr{E}', F\colon \mathscr{B} \to \mathscr{B}'$ such that*

$$\begin{tikzcd}
\mathscr{E} \arrow[r, "\hat{F}"] \arrow[d, "q"'] & \mathscr{E}' \arrow[d, "q'"] \\
\mathscr{B} \arrow[r, "F"] & \mathscr{B}'
\end{tikzcd}$$

commutes we obtain $\hat{F}_{\mathcal{L}}\colon \mathscr{E}_{\mathcal{L}} \to \mathscr{E}_{\mathcal{L}'}$ and $\hat{F}_{\mathcal{R}}\colon \mathscr{E}_{\mathcal{R}} \to \mathscr{E}'_{\mathcal{R}'}$ such that

$$\begin{tikzcd}
\mathscr{E}_{\mathcal{L}} \arrow[r, "\hat{F}_{\mathcal{L}}"] \arrow[d, "q_{\mathcal{L}}"'] & \mathscr{E}'_{\mathcal{L}'} \arrow[d, "q'_{\mathcal{L}'}"] & \textit{and} & \mathscr{E}_{\mathcal{R}} \arrow[r, "\hat{F}_{\mathcal{R}}"] \arrow[d, "q_{\mathcal{R}}"'] & \mathscr{E}'_{\mathcal{R}'} \arrow[d, "q'_{\mathcal{R}'}"] \\
\mathcal{L} \arrow[r, "F_{\mathcal{L}}"] & \mathcal{L}' & & \mathcal{R} \arrow[r, "F_{\mathcal{R}}"] & \mathcal{R}'
\end{tikzcd}$$

commute (where $F_{\mathcal{L}}, F_{\mathcal{R}}$ are obtained through pullbacks like $q_{\mathcal{L}}, q_{\mathcal{R}}$).

Proof: We obtain diagrams

$$\begin{tikzcd}
\mathscr{E}_{\mathcal{L}} \arrow[rr] \arrow[dd, "q_{\mathcal{L}}"] \arrow[dr, phantom, "\lrcorner", very near start] & & \mathscr{E} \arrow[dr, "\hat{F}"] \arrow[dd, "q" near start] & \\
& \mathscr{E}'_{\mathcal{L}'} \arrow[rr] \arrow[dd, "q'_{\mathcal{L}'}" near start] \arrow[dr, phantom, "\lrcorner", very near start] & & \mathscr{E}' \arrow[dd, "q'"] \\
\mathcal{L} \arrow[rr] \arrow[dr, "F_{\mathcal{L}}"'] & & \mathscr{B} \arrow[dr, "F"] & \\
& \mathcal{L}' \arrow[rr] & & \mathscr{B}'
\end{tikzcd}$$

and

$$\begin{tikzcd}
\mathscr{E}_{\mathcal{R}} \arrow[rr] \arrow[dd, "q_{\mathcal{R}}"] \arrow[dr, phantom, "\lrcorner", very near start] & & \mathscr{E} \arrow[dr, "\hat{F}"] \arrow[dd, "q" near start] & \\
& \mathscr{E}'_{\mathcal{R}'} \arrow[rr] \arrow[dd, "q'_{\mathcal{R}'}" near start] \arrow[dr, phantom, "\lrcorner", very near start] & & \mathscr{E}' \arrow[dd, "q'"] \\
\mathcal{R} \arrow[rr] \arrow[dr, "F_{\mathcal{R}}"'] & & \mathscr{B} \arrow[dr, "F"] & \\
& \mathcal{R}' \arrow[rr] & & \mathscr{B}'
\end{tikzcd}$$

so by the preceding Lemma we get the desired functors $\hat{F}_{\mathcal{L}}, \hat{F}_{\mathcal{R}}$. Q.E.D.

Definition 6.2.14 (Maps between sas-towers). Let $(q\colon \mathscr{G} \to \mathscr{E}, p\colon \mathscr{E} \to \mathscr{B}), (q'\colon \mathscr{G}' \to \mathscr{E}', p'\colon \mathscr{E}' \to \mathscr{B}')$ be two sas-towers. A *map* $(q,p) \to (q',p')$ is a triple $(\hat{F}, \tilde{F}, F)$ of functors

$$\hat{F}\colon \mathscr{G} \to \mathscr{G}', \tilde{F}\colon \mathscr{E}' \to \mathscr{E}, F\colon \mathscr{B} \to \mathscr{B}'$$

such that

(sas$_5$) $(\hat{F}, F)$ is a map of split fibrations (see Definition 2.4.9),

(sas$_6$) $(\tilde{F}, F)$ is a map of split fibrations and

(sas$_7$) Both $(\hat{F}_{\mathcal{V}}, F), (\hat{F}_{\mathcal{C}}, F)$ are maps of discrete fibrations, where $\hat{F}_{\mathcal{V}}, \hat{F}_{\mathcal{C}}$ are obtained as in the preceding corollary.

Lemma 6.2.15. *Let $\mathscr{C}, \mathscr{C}'$ be two 2-dep-categories and $(q_{\mathscr{C}}, p_{\mathscr{C}}), (q_{\mathscr{C}'}, p_{\mathscr{C}'})$ the corresponding sas-towers obtained as in Lemma 6.2.10. Then we obtain for every 2-dep-functor $F\colon \mathscr{C} \to \mathscr{C}'$ a map $(\hat{F}, \tilde{F}, F)\colon (p_{\mathscr{C}}, q_{\mathscr{C}}) \to (p_{\mathscr{C}'}, q_{\mathscr{C}'})$ of sas-towers.*

Proof: The map is defined as follows: The functors $(\tilde{F}, F)$ are defined as the map of split fibrations obtained from the underlying functor of 2-fam-categories. The functor $\hat{F}\colon \mathscr{G} \to \mathscr{G}'$ is defined by

$$\hat{F}(\Phi) := F_\lambda(\Phi) \text{ and } \hat{F}(f,\eta) = (f,\eta).$$

It follows from Corollary 6.2.13 that both $\hat{F}_{\mathcal{V}}, \hat{F}_{\mathcal{C}}$ are maps of discrete fibrations—that is they commute with q restricted to either $\mathcal{C}$ or $\mathcal{V}$—and $(\hat{F}, F)$ is also a map of split fibrations, as

$$(p \circ q)\big(\hat{F}(\Phi)\big) = F(c) = F\big((p \circ q)(\Phi)\big), \quad (p \circ q)\big(\hat{F}(f,\eta)\big) = F(f) = F\big((p \circ q)(f,\eta)\big).$$

and for $f\colon c' \to c, \lambda \in \mathrm{fHom}(c)$ and $\Phi \in \mathrm{dHom}(\lambda)$ we have

$$\overline{F(f)}\big(\tilde{F}(\Phi)\big) = \big(F(f), 1_{F_c(\lambda)}\big) = \tilde{F}(f, 1_\lambda) = F\big(\overline{f}(\Phi)\big).$$

Hence $(\hat{F}, \tilde{F}, F)$ is a map of sas-towers. Q.E.D.

Conversely every map of sas-towers yields a 2-dep-functor between the corresponding 2-dep-categories.

Lemma 6.2.16. *Let $(q'\colon \mathscr{G}' \to \mathscr{E}', p'\colon \mathscr{E}' \to \mathscr{B}')$ and $(q\colon \mathscr{G} \to \mathscr{E}, p\colon \mathscr{E} \to \mathscr{B})$ be sas-towers and $\mathscr{C}', \mathscr{C}$ be the associated 2-dep-categories from Lemma 6.2.11. Then any map $(\hat{F}, \tilde{F}, F)\colon (q',p') \to (q,p)$ of sas-towers translates to a 2-dep functor $F\colon \mathscr{C}' \to \mathscr{C}$ where*

$$F_c(e) := \tilde{F}(e) \text{ and } F_e(\Phi) := \hat{F}(\Phi).$$

Proof: As in the case for 2-fam-categories the 2-fam-functor F is already well-defined. To see that F_e are well-defined we observe that for all $e \in \mathrm{fHom}(c)$, that is $p(e) = c$ and all $\Phi \in \mathrm{dHom}(e)$, that is $q(\Phi) = e$ we have

$$q\big(F_e(\Phi)\big) = q\big(\hat{F}(e)\big) = \tilde{F}(q'(\Phi)) = F_{F(c)}\big(q'(\Phi)\big)$$

as desired. To see the equalities (2dep$_4$) and (2dep$_5$) we compute

$$\begin{aligned} F_e(g \circ \Phi) &= \hat{F}(g \circ \Phi) = \hat{F}\big(\underline{g}(\Phi)\big) = \underline{\tilde{F}(g)}\big(\hat{F}(\Phi)\big) \text{ for } (g\colon e' \to e) \in \mathcal{V}, \\ F_e(\Phi \circ f) &= \hat{F}(\Phi \circ f) = \hat{F}\big(\overline{f}\Phi)\big) = \overline{F(g)}\big(\hat{F}(\Phi)\big) \text{ for } (f\colon c' \to c) \in \mathscr{B}. \end{aligned}$$

Thus F is a 2-dep-functor. Q.E.D.

It is immediate from the above definition that this assignment of sas-maps to 2-dep-functors fulfils functoriality.

Definition 6.2.17 (Transformations of maps of sas-towers). Let $(q,p),(q',p')$ be sas-towers and $(\hat{F},\tilde{F},F),(\hat{G},\tilde{G},G)\colon (q,p) \to (q',p')$ be maps of sas-towers. We define a *transformation of sas-maps* $(\hat{F},\tilde{F},F) \to (\hat{G},\tilde{G},G)$ to be a triple $(\hat{\eta},\tilde{\eta},\eta)$ where

$$\hat{\eta}\colon \hat{F} \Rightarrow \hat{G}, \tilde{\eta}\colon \tilde{F} \Rightarrow \tilde{G}, \eta\colon F \Rightarrow G$$

are natural transformations such that the following conditions are met:

(sas$_8$) Both $q' \bullet \hat{\eta} = \tilde{\eta} \bullet q$ and $p' \bullet \tilde{\eta} = \eta \bullet p$, that is the diagram

$$\begin{tikzcd} \mathscr{G} \arrow[r, "q"'] \arrow[d, bend right=50, "\hat{G}"'] \arrow[d, bend left=50, "\hat{F}"] \arrow[d, phantom, "\underset{\hat{\eta}}{\Leftarrow}"] & \mathscr{E} \arrow[r, "p"'] \arrow[d, bend right=50, "\tilde{G}"'] \arrow[d, bend left=50, "\tilde{F}"] \arrow[d, phantom, "\underset{\tilde{\eta}}{\Leftarrow}"] & \mathscr{B} \arrow[d, bend right=50, "G"'] \arrow[d, bend left=50, "F"] \arrow[d, phantom, "\underset{\eta}{\Leftarrow}"] \\ \mathscr{G}' \arrow[r, "q'"] & \mathscr{E}' \arrow[r, "p'"] & \mathscr{B}' \end{tikzcd}$$

commutes.

(sas$_9$) Both $(\hat{\eta},\eta)\colon (\hat{F},F) \to (\hat{G},G)$ and $(\tilde{\eta},\eta)\colon (\tilde{F},F) \to (\tilde{G},G)$ fulfil

$$\begin{aligned} &\overline{\eta_b}^{p\circ q}(\Phi) = \hat{\eta}_\Phi \quad \text{for } b \in \mathscr{B}, \Phi \in \mathscr{G} \text{ such that } (p \circ q)(e) = b, \\ &\overline{\eta_b}^{p}(e) = \tilde{\eta}_e \quad \text{for } b \in \mathscr{B}, e \in \mathscr{E} \text{ such that } p(e) = b. \end{aligned}$$

Lemma 6.2.18. *Let $\mathscr{C},\mathscr{C}'$ be 2-dep-categories, $F,G\colon \mathscr{C} \to \mathscr{C}'$ be 2-dep-functors and $\eta\colon F \Rightarrow G$ be a 2-dep-natural transformation. Then we obtain a transformation of the associated sas-maps, $(\hat{\eta},\tilde{\eta},\eta)\colon (\hat{F},\tilde{F},F) \to (\hat{G},\tilde{G},G)$, where $(\hat{F},\tilde{F},F),(\hat{G},\tilde{G},G)$ are obtained as in Lemma 6.2.15.*

Proof: We first note that we obtain the 2-fam-natural transformation $(\tilde{\eta},\eta)\colon (\tilde{F},F) \Rightarrow (\tilde{G},G)$ as in Proposition 2.4.14, thus in

$$\begin{tikzcd} \mathscr{G} \arrow[r, "q"'] \arrow[d, bend right=50, "\hat{G}"'] \arrow[d, bend left=50, "\hat{F}"] \arrow[d, phantom, "\underset{\hat{\eta}}{\Leftarrow}"] & \mathscr{E} \arrow[r, "p"'] \arrow[d, bend right=50, "\tilde{G}"'] \arrow[d, bend left=50, "\tilde{F}"] \arrow[d, phantom, "\underset{\tilde{\eta}}{\Leftarrow}"] & \mathscr{B} \arrow[d, bend right=50, "G"'] \arrow[d, bend left=50, "F"] \arrow[d, phantom, "\underset{\eta}{\Leftarrow}"] \\ \mathscr{G}' \arrow[r, "q'"] & \mathscr{E}' \arrow[r, "p'"] & \mathscr{B}' \end{tikzcd}$$

we already know that the right square commutes. To see that the left square commutes we first have to define $\hat{\eta}$. We set $\hat{\eta}_\Phi = (\eta_c, 1_{F_c(\lambda)})\colon \hat{F}(\Phi) \to \hat{G}(\Phi)$—which is possible as $F(\Phi) = G(\Phi) \circ f$. This allows us to compute

$$(q' \bullet \hat{\eta})_\Phi = q'(\eta_c, 1_{F_c(\lambda)}) = (\eta_c, 1_{F_c(\lambda)}) = \tilde{\eta}_\lambda = \tilde{\eta}_{q(\Phi)} = (\tilde{\eta} \bullet q)_\Phi.$$

Additionally we compute that

$$\overline{\eta_c}^{p\circ q}(\Phi) = \big((\eta_c, 1_{F_c(\lambda)})\colon G(\Phi) \circ \eta_c \to G(\Phi)\big) = \hat{\eta}_\Phi. \qquad \text{Q.E.D.}$$

Obviously we have the reverse direction as well.

Lemma 6.2.19. *Let $(q,p),(q',p')$ be sas-towers, $(\hat{F},\tilde{F},F),(\hat{G},\tilde{G},G)\colon (q,p)\to(q',p')$ be maps of sas-towers. Then any transformation $(\hat{\eta},\tilde{\eta},\eta)\colon(\hat{F},\tilde{F},F)\to(\hat{G},\tilde{G},G)$ gives rise to a 2-dep-natural transformation $\eta\colon F\Rightarrow G$ between the associated 2-dep-functors from Lemma 6.2.16.*

Proof: We already know from 2.4.14 that the underlying map of split fibrations $(\tilde{\eta},\eta)$ gives rise to a 2-fam-natural transformation. So it remains to confirm the remaining property, that is

$$G_\lambda(\Phi)\circ\eta_c = F_\lambda(\Phi).$$

But $G_\lambda(\Phi)\circ\eta_c := (\eta_c)^{(p'\circ q')*}\big(G_\lambda(\Phi)\big)$ and thus we compute

$$\begin{aligned}(\eta_c)^{(p'\circ q')*}\big(G_\lambda(\Phi)\big) &= \hat{F}(\Phi) && \text{as } \hat{\eta}_\Phi\colon \hat{F}(\Phi)\to\hat{G}(\Phi)\\ &= F_\lambda(\Phi) && \text{by definition.}\end{aligned}$$

Q.E.D.

The functoriality of the assignment "transformation of sas-maps" ↦ "2-dep-natural transformation" is again immediate from the definition (similarly for the reverse direction). Thus these assignment routines determine two 2-functors between categories we now define.

Definition 6.2.20 (Category of sas-towers). We define the 2-category sas of sas-tower as follows:

- Its objects (0-cells) are sas-towers as in Definition 6.2.9,
- its 1-maps are maps of sas-towers as in Definition 6.2.14.
- its 2-maps are transformations of sas-maps as in Definition 6.2.17.

The composition of 1-cells and the vertical and horizontal composition of 2-maps is defined component-wise.

Lemma 6.2.21. *We obtain two 2-functors*

$$\begin{array}{rrl}(q_{\text{-}},p_{\text{-}})\colon \mathsf{2\text{-}depC}\to\mathsf{sas}, & \mathscr{C}\mapsto(q,p) & \textit{as in Lemma 6.2.10}\\ & F\mapsto(\hat{F},\tilde{F},F) & \textit{as in Lemma 6.2.15}\\ & \eta\mapsto(\hat{\eta},\tilde{\eta},\eta) & \textit{as in Lemma 6.2.18}\\ \mathscr{C}_{\text{-}}\colon \mathsf{sas}\to\mathsf{2\text{-}depC}, & (q,p)\mapsto\mathscr{C} & \textit{as in Lemma 6.2.11}\\ & (\hat{F},\tilde{F},F)\mapsto F & \textit{as in Lemma 6.2.16}\\ & (\hat{\eta},\tilde{\eta},\eta)\mapsto\eta & \textit{as in Lemma 6.2.19}\end{array}$$

Proof: As we remarked the functoriality with respect to 1-maps and 2-maps is immediate in both cases. Q.E.D.

Theorem 6.2.22. *The two 2-functors of the previous lemma establish a 2-equivalence*

$$\mathsf{sas}\simeq\mathsf{2\text{-}depC}\,.$$

Proof: It is immediate that mapping a 2-dep-category to the associated sas-tower and that tower to its associated 2-dep-category yields the same 2-dep-category (up to renaming). An analogous result holds for the 2-dep-functors and 2-dep-natural transformations.

Conversely, if we are given a sas-tower applying $\mathscr{C}_{\text{-}}$ first and then $(q_{\text{-}},p_{\text{-}})$ yields the same sas-tower (up to renaming), similar for sas-maps and transformations of such maps. Q.E.D.

Chapter 7
(2-dep,Σ)-categories and higher comprehension categories

In this chapter we will examine the most abstract notion explored by Ehrhardt in [20], (2-dep,Σ)-categories. To this end we will combine the notion of a higher discrete comprehension category from chapter 5 and the notion of a sas-tower from the preceding chapter to arrive at a notion of higher discrete comprehension category, which will turn out to be (2-)equivalent to those (2-dep,Σ)-categories.

As only (2-dep,Σ)-categories defined in [20], but not the corresponding notion of maps between those categories and transformation between these maps, we also have to define these notions, inspired by the core result [20, Theorem 3.5.8], which allows associate to each (2-fam,Σ)-category a (2-dep,Σ)-category in a canonical way.

7.1 (2-dep,Σ)-categories

Definition 7.1.1 ((2-dep,Σ)-categories). 0 A (2-dep,Σ)-category is a 2-dep-category (in the sense of Definition 6.1.1) together with

- a morphism $\mathsf{pr}_1^\lambda\colon \sum_c \lambda \to c$ for each family arrow λ such that $\mathscr{C}$ becomes a (2-fam,Σ) category through this data and
- a family arrow $\mathsf{pr}_2^\lambda \in \mathrm{dHom}(\lambda \circ \mathsf{pr}_1^\lambda)$ for each family arrow λ

such that the following conditions hold:

(2dΣ$_1$) for every family arrow $\lambda \in \mathrm{fHom}(c)$ for arbitrary c and $f\colon b \to c$

$$[\mathsf{pr}_2^\lambda]\Big(\sum_\lambda f\Big) = \mathsf{pr}_2^{\lambda\circ f}\,.$$

(2dΣ$_2$) For every $c \in \mathscr{C}, \lambda, \mu \in \mathrm{fHom}(c), \eta\colon \lambda \to \mu$

$$(\eta \bullet \mathsf{pr}_1^\lambda) \circ \mathsf{pr}_2^\lambda = [\mathsf{pr}_2^\mu]\Big(\sum_{\lambda,\mu} \eta\Big).$$

This can be expressed through the commutativity of the following diagram:

$$\begin{tikzcd}
\sum_c \lambda \arrow[rr, Rightarrow, bend left, "\mathsf{pr}_2^\lambda"] \arrow[dr, "\mathsf{pr}_1^\lambda"] \arrow[drr, bend left, "\lambda\circ\mathsf{pr}_1^\lambda"] \arrow[dd, "\sum_{\lambda,\mu}\eta"'] & & \\
 & a \arrow[r, bend left, "\lambda"{name=L}] \arrow[r, bend right, "\mu"'{name=M}] \arrow[from=L, to=M, Rightarrow, "\eta"] & . \\
\sum_c \mu \arrow[ur, "\mathsf{pr}_1^\mu"'] \arrow[ur, phantom] \arrow[urr, bend right, "\mu\circ\mathsf{pr}_1^\mu"'] \arrow[rr, Rightarrow, bend right, "\mathsf{pr}_2^\mu"'] & &
\end{tikzcd}$$

Definition 7.1.2 ((2-dep,Σ)-functors). Let $\mathscr{C}, \mathscr{D}$ be (2-dep,Σ)-categories. We define a *(2-dep,Σ)-functor* $F\colon \mathscr{C} \to \mathscr{D}$ to be a (2-fam,Σ)-functor F (see Definition 3.1.11) such that

(2dΣ_3) F is also a 2-dep-functor (see Definition 6.1.2),

(2dΣ_4) F is also a (dep,Σ)-functor (see Definition 5.1.2),

Definition 7.1.3 ((2-dep,Σ)-natural transformations). Let $\mathscr{C}, \mathscr{D}$ be (2-dep,Σ)-categories and $F, G\colon \mathscr{C} \to \mathscr{D}$ (2-dep,Σ)-functors. We define a *(2-dep,Σ)-natural transformation*$\eta\colon F \Rightarrow G$ to be a (2-fam,Σ)-natural transformation (see Definition 3.1.12) such that

(2dΣ_5) η is a 2-dep-natural transformation (see Definition 6.1.3) and

(2dΣ_6) η is a (dep,Σ)-natural transformation (see Definition 5.1.4).

7.2 Higher comprehension categories

Before we define higher comprehension categories we will define a property of natural transformations related by fibrations, and prove some small results regarding this property, whose importance will become apparent after the definition of higher comprehension categories.

Definition 7.2.1. Let $\mathscr{C}, \mathscr{E}, \mathscr{B}$ be categories, $p\colon \mathscr{E} \to \mathscr{B}$ a split fibration and $F, G\colon \mathscr{C} \to \mathscr{B}$, $F', G'\colon \mathscr{C} \to \mathscr{E}$ be functors connected by natural transformations χ, η as in

$$\begin{tikzcd}
& \mathscr{E} \arrow[dd, "p"] \\
\mathscr{C} \arrow[ur, bend left=50, "F'"{name=Fp}] \arrow[ur, "G'"'{name=Gp}] \arrow[dr, "F"{name=F}] \arrow[dr, bend right=50, "G"'{name=G}] & \\
& \mathscr{B}
\arrow[from=Fp, to=Gp, Rightarrow, "\chi"]
\arrow[from=F, to=G, Rightarrow, "\eta"]
\end{tikzcd} \tag{13}$$

We then say that χ is the *p-lift of η* if the above diagram commutes on the level of functors, and for all objects c of $\mathscr{C}$ we have that

$$\overline{\eta_c}(G(c) = \chi_c.$$

In this case we write $\overline{\eta} = \chi$ or $\overline{\eta}^p = \chi$.

First we show that the notation $\overline{\eta} = \chi$ is actually justified, that is, if both $\overline{\eta} = \chi$ and $\overline{\eta} = \xi$, then $\chi = \xi$.

Lemma 7.2.2. *In the situation of the definition above, if $\overline{\eta} = \chi$ and $\overline{\eta} = \xi$, then $\chi = \xi$.*

Proof: This is immediate, as for all c in $\mathscr{C}$ we can compute

$$\xi_e = \overline{\eta_c}G(c) = \chi_c.$$

Q.E.D.

Lemma 7.2.3. *In the situation of the preceding definition, if $\overline{\eta} = \chi$, then*

$$p \bullet \chi = \eta,$$

that is the entirety of diagram (13) *commutes.*

Proof: The equation is immediate, because if χ_c is a lift of η_c along $G(c)$, the equality $p(\chi_c) = \eta_c$ follows by definition. As this holds for all c the equality follows. Q.E.D.

Lemma 7.2.4. *If in the situation of the preceding definition the split fibration is discrete, then the following are equivalent:*

1. $\chi = \overline{\eta}^p$.

2. $p \bullet \chi = \eta$ *and* $p \circ G' = G$.

Proof: The direction 1 ⇒ 2 is immediate, as any discrete fibration is a split fibration, so we can simply apply Lemma 7.2.3. For the direction 2 ⇒ 1 we remark that any lift of η_c along $G(c)$ must be unique, so as $p(\chi_c) = \eta_c$ we already know that it is that unique lift. That it is p-cartesian is then immediate. Furthermore we can infer that $(p \circ F')(f = F(f)$ for all arrows $f \colon c \to d$, as both $F'(f)$ and the unique lift of $F(f)$ make

$$\begin{tikzcd}
F'(c) \arrow[r, "\chi_c"] \arrow[d, dotted] & G'(c) \arrow[d, "G'(g)"] \\
F'(c) \arrow[r, "\chi_d"'] & G'(d)
\end{tikzcd}$$

commute, thus $F'(f)$ must be that unique lift and $p\big(F'(f)\big) = F(f)$. Q.E.D.

Before we give the notion of higher comprehension categories, we recall some notation (and introduce some new notation) that will make the definition more amenable.

7.2.5 Some notation for specific lifts. When working with fibrations $p \colon \mathscr{E} \to \mathscr{B}$ we already saw (in 2.4.14) that the precomposition of a vertical arrow $\eta \colon e' \to e$ in $\mathscr{E}$ with an arrow f of the base category involves invoking the universal property as such:

$$\begin{tikzcd}
f^*(e') \arrow[r, "\overline{f}(e')"] \arrow[d, dotted, "h"] & e' \arrow[d, "\eta"] \\
f^*(e) \arrow[r, "\overline{f}(e)"'] & e,
\end{tikzcd}$$

where h is then $\eta \bullet f$. From now on we will always use $\eta \bullet f$ when we mean the arrow obtained in this manner. It is immediate that when we consider a fibration stemming from a 2-fam-category this $\eta \bullet f$ coincides with the $\eta \bullet f$ of the 2-fam-category.

Another issue we have is regarding arrows vertical with respect to a fibration. As discussed earlier section 6.2 the factorisation system obtained from a fibration distinguishes between arrow which are vertical, in the sense that they map to an isomorphism under p, and those that are cartesian, that is they are isomorphic to a chosen p-cartesian lift. However, when discussing fibrations stemming from 2-fam-categories this distinction is to coarse for us, it merely distinguishes morphisms (f, η) where either f is an isomorphism or η is. What we want, however is that either f is an identity or η is.

- For η, this can obtained easily, just consider $\overline{p(f, \eta))}^p$, which will have η "removed". Similarly, this works for any morphism $f \colon e' \to e$ in the total category of a (split) fibration $p \colon \mathscr{E} \to \mathscr{B}$, taking $\overline{p(f)}(e)$. We will call this the *p-prone part* of f—a slight abuse of the terminology coined by Johnstone and Taylor in [37, 265ff] and [61, Definition 9.2.6] respectively—and denote it as $\mathtt{prn}(f)$.

- For f we need to work more: considering the diagram

$$\begin{tikzcd}
\lambda \arrow[d, dotted, "h"'] \arrow[dr, "{(f,\eta)}"] & \\
f^*(\mu) \arrow[r, "\overline{f}(\mu)"'] & \mu
\end{tikzcd}$$

we can see that h must be η. Inspired from this, for an arbitrary fibration $p\colon \mathscr{E} \to \mathscr{B}$ and a vertical morphism $f\colon e' \to e$, we define through the same universal property

$$\begin{tikzcd}
e' \arrow[d, "\mathtt{tv}(f)"'] \arrow[dr, "f"] & \\
\big(p(f)\big)^*(e) \arrow[r, "\overline{p(f)}(e)"'] & e,
\end{tikzcd}$$

the *true vertical* $\mathtt{tv}(f)$ of f. It is immediate from this definition that $p(\mathtt{tv}(f)) = 1_{p(e')}$, as

$$p(f) = p\big(\overline{p(f)}(e) \circ \mathtt{tv}(f)\big) = p\big(\overline{p(f)}(e)\big) \circ p\big(\mathtt{tv}(f)\big) = p(f) \circ p\big(\mathtt{tv}(f)\big),$$

so as $p(f)$ is invertible the claim follows.

Note however, that while any morphism in the total category of a fibration can be decomposed into $\mathtt{prn}(f) \circ \mathtt{tv}(f)$, this does not always yield the desired outcome, as for example $\mathtt{tv}(f)$ may not lie over the identity, as the proof of this fact rested on $p(f)$ being invertible!

The following lemma will be needed for the definition of higher comprehension categories to be well-defined.

Lemma 7.2.6. *If $P\colon \mathscr{E} \to \mathscr{B}^{[1]}$ is a comprehension category, then $f \bullet P_0(f)$ is vertical if f is.*

Proof: First we use that any pullback of an isomorphism is an isomorphism, so $P_0(f)$ is an isomorphism. Then $f \bullet P_0(f)$ is be vertical because $p\big(f \bullet P_0(f)\big) = P_0(f)$. Q.E.D.

Definition 7.2.7 (Higher comprehension categories). We define a *higher comprehension category* to consist of

(HCC$_1$) a sas-tower (Definition 6.2.9) $(q\colon \mathscr{G} \to \mathscr{E}, p\colon \mathscr{E} \to \mathscr{B})$,

(HCC$_2$) a comprehension category $P\colon \mathscr{E} \to \mathscr{B}^{[1]}$ such that $\mathrm{cod} \circ P = p$ and

(HCC$_3$) a functor $\mathsf{pr}_2\colon \mathscr{E} \to \mathscr{G}$ together with a natural transformation $\chi\colon q \circ \mathsf{pr}_2 \Rightarrow \mathrm{id}_{\mathscr{E}}$ such that

a) for all $e \in \mathscr{E}$ and all $f\colon c' \to p(e)$ we have

$$\mathsf{pr}_2\left(\overline{f}^p(e)\right) = \overline{P_0\big(\overline{f}^p(e)\big)}^{p\circ q}(\mathsf{pr}_2(e)).$$

b) for all $e, e' \in \mathscr{E}$ and all $f\colon e' \to e$ vertical with respect to p we have

$$\underline{\mathtt{tv}(f) \bullet P(e')}_q\, \mathsf{pr}_2(e') = \overline{P_0(f)}^{p\circ q}\big(\,\mathsf{pr}_2(e)\big).$$

c) $\overline{P}^p = \chi$.

We will denote this data as (q, P, χ).

Remark 7.2.8. From Lemma 7.2.3 it is now immediate that condition (HCC$_3$) is an abstraction of condition (HdCC$_1$) in the definition of higher discrete comprehension categories to the setting of split fibrations. We could have defined higher discrete comprehension categories simply as higher comprehension categories where $p, p \circ q$ are discrete, but this would have resulted in a lot of unnecessary data attached to then, like the discrete ambifibration, which could be replaced by a discrete fibration in this instance.

So we choose the more "streamlined" version for this definition, which can be recovered from the general one after accounting for all simplifications—like Lemma 7.2.4—that the discreteness of the involved fibrations allows for.

Lemma 7.2.9. *We can map any (2-dep,Σ)-category to a higher comprehension category by letting* $(q\colon \mathscr{G} \to \mathscr{E}, p\colon \mathscr{E} \to \mathscr{B}$ *be the sas-tower obtained from Lemma 6.2.10,* $P\colon \mathscr{E} \to \mathscr{B}^{[1]}$ *be the comprehension category obtained from Lemma 3.2.11 and* $\mathsf{pr}_2\colon \mathscr{E} \to \mathscr{G}$ *be the functor defined through*

$$\mathscr{E} \ni \lambda \mapsto \mathsf{pr}_2^\lambda \textit{ and } \big((f,\eta)\colon \lambda \to \mu\big) \mapsto \Bigg(\Big(\sum_\mu f \circ \sum_{\lambda,\mu} \eta, \eta \bullet \mathsf{pr}_1^\lambda\Big)\colon\ \mathsf{pr}_2^\lambda \to \mathsf{pr}_2^\mu\Bigg).$$

The natural transformation is defined through $\chi_\lambda := (\mathsf{pr}_1^\lambda, 1_{\lambda\circ\mathsf{pr}_1^\lambda})$.

Proof: It only remains to check the conditions on pr_2, all other conditions are checked in their respective lemmata. To check that pr_2 is well-defined we observe that obviously $\mathsf{pr}_2^\lambda \in \mathscr{G}$ for all $\lambda \in \mathscr{E}$. For $(f,\eta)\colon \lambda \to \mu$ we recall that this corresponds to $\eta\colon \lambda \Rightarrow \mu \circ f$, where $f\colon p(\lambda) \to p(\mu)$. This implies

$$(\eta \bullet \mathsf{pr}_1^\lambda) \circ \mathsf{pr}_2^\lambda = [\mathsf{pr}_2^{\mu\circ f}]\Big(\sum_{\lambda,\mu} \eta\Big) = \Big[[\mathsf{pr}_2^\mu]\Big(\sum_\mu f\Big)\Big]\Big(\sum_{\lambda,\mu} \eta\Big) = [\mathsf{pr}_2^\mu]\Big(\sum_\mu f \circ \sum_{\lambda,\mu} \eta\Big),$$

hence $\big(\sum_\lambda f \circ \sum_{\lambda,\mu} \eta, \eta \bullet \mathsf{pr}_1^\lambda\big)\colon\ \mathsf{pr}_2^\lambda \to \mathsf{pr}_2^\mu$.

For the functoriality we compute for $\lambda \in \mathrm{fHom}(c)$

$$\begin{aligned}\Big(\sum_\lambda 1_c \circ \sum_{\lambda,\lambda} 1_\lambda, 1_\lambda \bullet \mathsf{pr}_1^\lambda\Big) &= \Big(1_{\sum_c \lambda} \circ \sum_{\lambda,\lambda} 1_\lambda, 1_\lambda \bullet \mathsf{pr}_1^\lambda\Big) = \Big(1_{\sum_c \lambda} \circ 1_{\sum_c \lambda}, 1_\lambda \bullet \mathsf{pr}_1^\lambda\Big)\\ &= \Big(1_{\sum_c \lambda}, 1_\lambda \bullet \mathsf{pr}_1^\lambda\Big) = \Big(1_{\sum_c \lambda}, 1_{\lambda\bullet\mathsf{pr}_1^\lambda}\Big)\\ &= 1_{\lambda\circ\mathsf{pr}_1^\lambda} \text{ in } \mathscr{E}.\end{aligned}$$

Analogously one checks the compatibility with composition, using the fact that given $(g,\theta) \circ (f,\eta)$ we have the commutativity of

$$\begin{tikzcd}
\sum_{c''} \lambda \arrow[d, "\sum_{\lambda,\mu\circ f} \eta"'] & & & \\
\sum_{c''} \mu \circ f \arrow[r, "\sum_\mu f"] \arrow[d, "\sum_{\mu\circ f,\nu\circ(g\circ f)} \theta\bullet f"'] & \sum_{c'} \mu \arrow[d, "\sum_{\mu,\nu\circ g} \theta"] & & \\
\sum_{c''} \nu \circ (g \circ f) \arrow[r, "\sum_{\nu\circ g} f"] \arrow[d, "\mathsf{pr}_1^{\nu\circ(g\circ f)}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_{c'} \nu \circ g \arrow[r, "\sum_\nu g"] \arrow[d, "\mathsf{pr}_1^{\nu\circ g}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_c \nu \arrow[d, "\mathsf{pr}_1^\nu"] & \\
c'' \arrow[r, "f"'] & c' \arrow[r, "g"'] & c \arrow[r, "\nu"] & {}
\end{tikzcd},$$

which allows us to compute

$$\begin{aligned}\sum_\nu g \circ f \circ \sum_{\lambda,\nu} \theta \circ \eta &= \sum_\nu g \circ \sum_{\nu\circ g} f \circ \sum_{\substack{\mu\circ f,\\ \eta\circ(g\circ f)}} \theta \bullet f \circ \sum_{\lambda,\mu\circ f} \eta\\ &= \sum_\nu g \circ \sum_{\mu,\nu\circ g} \theta \circ \sum_\mu f \circ \sum_{\lambda,\mu\circ f} \eta.\end{aligned}$$

The naturality of χ can be checked via computation as well. Let $(f,\eta)\colon \lambda \to \mu$ be given, then we have to show that

$$\begin{tikzcd}
\lambda \circ \mathsf{pr}_1^\lambda \arrow[r, "(\mathsf{pr}_1^\lambda{,}1_{\lambda\circ\mathsf{pr}_1^\lambda})"] \arrow[d, "(\sum_\mu f\circ\sum_{\lambda,\mu}\eta{,}\eta\bullet\mathsf{pr}_1^\lambda)"'] & \lambda \arrow[d, "(f{,}\eta)"] \\
\mu \circ \mathsf{pr}_1^\mu \arrow[r, "(\mathsf{pr}_1^\mu{,}1_{\mu\circ\mathsf{pr}_1^\mu})"'] & \mu
\end{tikzcd}$$

commutes, that is to show

$$f \circ \mathsf{pr}_1^\lambda = \mathsf{pr}_1^\mu \circ \sum_\mu f \circ \sum_{\lambda,\mu} \eta \quad \text{and} \quad \big(1_{\mu\circ\mathsf{pr}_1^\mu} \bullet \sum_\mu f \circ \sum_{\lambda,\mu} \eta\big) \circ (\eta \bullet \mathsf{pr}_1^\lambda) = (\eta \bullet \mathsf{pr}_1^\lambda) \circ 1_{\lambda\circ\mathsf{pr}_1^\lambda}.$$

The second equality is immediate. For the first equality note that from the definition of (2-fam,Σ)-categories we know that

$$\begin{tikzcd}
\sum_{c'} \lambda \arrow[d, "\sum_{\lambda,\mu} f"'] \arrow[dd, bend right=80, "\mathsf{pr}_1^\lambda"'] & \\
\sum_{c'} \mu \circ f \arrow[r, "\sum_\mu f"] \arrow[d, "\mathsf{pr}_1^{\mu\circ f}"'] & \sum_c \mu \arrow[d, "\mathsf{pr}_1^\mu"] \\
c' \arrow[r, "f"'] & c
\end{tikzcd}$$

commutes, which yields the desired equality.

From the definition of the postcomposition with an arrow $f\colon c \to p(\lambda)$ we can compute

$$\begin{aligned}
\mathsf{pr}_2\left(\overline{f}^p(\lambda)\right) = \mathsf{pr}_2(f, 1_{\lambda\circ f}) &= \Big(\sum_\lambda f \circ \sum_{\lambda\circ f,\lambda} 1_{\lambda\circ f}, 1_{\lambda\circ f} \bullet \mathsf{pr}_1^{\lambda\circ f}\Big) \\
&= \Big(\sum_\lambda f, 1_{\lambda\circ f\circ\mathsf{pr}_1^{\lambda\circ f}}\Big) = \Big(P_0(\overline{f}^p(\lambda)), 1_{\lambda\circ f\circ\mathsf{pr}_1^{\lambda\circ f}}\Big) \\
&= \overline{P_0\big(\overline{f}^p(\lambda)\big)}^{p\circ q}(\mathsf{pr}_2^\lambda),
\end{aligned}$$

For the precomposition with a vertical arrow $(f,\eta)\colon \lambda \to \mu$ we compute

$$\begin{aligned}
\underline{\mathtt{tv}(f,\eta) \bullet P(\lambda)}_q\, \mathsf{pr}_2(\lambda) &= \underline{(1_{p(\lambda)}, \eta) \bullet \mathsf{pr}_1^\lambda}_q(\mathsf{pr}_2(\lambda)) = \underline{1_{p(\lambda\circ\mathsf{pr}_1^\lambda)}, \eta \bullet \mathsf{pr}_1^\lambda}_q(\mathsf{pr}_2(\lambda)) \\
&= \underline{\big(1_{P(\lambda)}, \eta \bullet \mathsf{pr}_1^\lambda\big)}_q\big(\mathsf{pr}_2(\lambda)\big) = (\eta \bullet \mathsf{pr}_1^\lambda) \circ \mathsf{pr}_2^\lambda \\
&= [\mathsf{pr}_2^{\mu\circ f}]\Big(\sum_{\lambda,\mu} \eta\Big) = [\mathsf{pr}_2^\mu]\Big(\sum_\mu f \circ \sum_{\lambda,\mu} \eta\Big) \\
&= \overline{\sum_\mu f \circ \sum_{\lambda,\mu} \eta}^{p\circ q}(\mathsf{pr}_2(\mu)) = \overline{P_0(f)}^{p\circ q}(\mathsf{pr}_2(\mu))
\end{aligned}$$

Lastly, it is immediate from the definitions that $p(\chi_\lambda) = \mathsf{pr}_1^\lambda$ for all λ and thus $\overline{P}^p = \chi$. Q.E.D.

Lemma 7.2.10. *We can map any higher comprehension category (q,p,P,χ) to a (2-dep,Σ)-category by letting the underlying 2-dep-category be the category $\mathscr{C}$ obtained from the sas-tower (q,p) by Lemma 6.2.11 and the (2-fam,Σ)-structure is obtained from p and P using Lemma 3.2.10.*

For each $e \in \mathscr{E}$ we define $\mathsf{pr}_2^e := \mathsf{pr}_2(e)$.

Proof: As the two lemmata already yield the underlying 2-dep- and (2-fam,Σ)-structure, it remains to check conditions ($2\mathrm{d}\Sigma_1$) and ($2\mathrm{d}\Sigma_2$). For the first condition we first compute

$$\begin{aligned}[\mathsf{pr}_2^e]\big(\sum_e f\big) &= \overline{P_0(\overline{f}^p(e))}^{*(p\circ q)}(\mathsf{pr}_2(e)) = \mathsf{pr}_2\left(\overline{f}^p(e)\right)\\ &= \mathsf{pr}_2^{e\circ f}\,.\end{aligned}$$

For the second condition we compute (where $f\colon e' \to e$ is such that $p(f) = 1_{p(e)}$

$$\begin{aligned}(f \bullet \mathsf{pr}_1^{e'}) \circ \mathsf{pr}_2^{e'} &= \big(\mathtt{tv}(f) \bullet P_0(e')\big) \circ \mathsf{pr}_2(e') = \underline{\mathtt{tv}(f) \bullet P_0(e')}_q\big(\,\mathsf{pr}_2(e')\big)\\ &= \overline{P_0(f)}^{p\circ q}\big(\,\mathsf{pr}_2(e)\big) = \overline{\sum_{p(e)} f}^{p\circ q}\big(\,\mathsf{pr}_2(e)\big)\\ &= [\mathsf{pr}_2^e]\Big(\sum_{p(e)} f\Big).\end{aligned}$$

Q.E.D.

Definition 7.2.11 (Maps of higher comprehension categories). Let (q, P, χ), (q', P', χ') be higher comprehension categories. A *map* is a triple $(\hat{F}, \tilde{F}, F)\colon (q, P, \chi) \to (q', P', \chi')$ such that

(hCC_4) $(\hat{F}, \tilde{F}, F)$ is a map of sas-towers (see Definition 6.2.14).

(hCC_5) $(\tilde{F}, F)$ is a strict map of comprehension categories (see Definition 3.2.2).

(hCC_6) $\mathsf{pr}_2' \circ \tilde{F} = \hat{F} \circ \mathsf{pr}_2$ and $\chi' \bullet \tilde{F} = \tilde{F} \bullet \chi$.

Lemma 7.2.12. *Any (2-dep,Σ)-functor $F\colon \mathscr{C} \to \mathscr{C}'$ can be translated into a map between the associated higher comprehension categories (from Lemma 7.2.9).*

Proof: From Lemma 6.2.15 we know that the underlying 2-dep-functor F yields a map $(\hat{F}, \tilde{F}, F)\colon (q, p) \to (q', p')$ of sas-towers. Analogously we know from Proposition 3.2.11 that the underlying (2-fam,Σ)-functor yields a strict map $(\dot{F}, F)$ of comprehension categories. It is immediate from the definition of those maps that $\dot{F} = \tilde{F}$, so the maps agree on the 2-fam-structure. To see that $\mathsf{pr}_2' \circ \tilde{F} = \hat{F} \circ \mathsf{pr}_2$ we simply compute for $(f, \eta)\colon \lambda \to \mu$ where $\lambda \in \mathrm{fHom}(c), \mu \in \mathrm{fHom}(c')$

$$\begin{aligned}(\mathsf{pr}_2' \circ \tilde{F})(\lambda) &= \mathsf{pr}_2^{F_c(\lambda)} = F_{\lambda\circ\mathsf{pr}_1^\lambda}(\mathsf{pr}_2^\lambda) = (\hat{F} \circ \mathsf{pr}_2)(\lambda)\\ \text{and } (\mathsf{pr}_2' \circ \tilde{F})(f, \eta) &= \mathsf{pr}_2'\left((F(f), F_c(\eta))\right)\\ &= \Big(\sum_{F_{c'}(\mu)} F(f) \circ \sum_{F_c(\lambda), F_{c'}(\mu)} F_c(\eta), F_c(\eta) \bullet F_{\sum_c \lambda}(\mathsf{pr}_1^\lambda)\Big)\\ &= \Big(F\Big(\sum_\mu f \circ \sum_{\lambda,\mu}\Big), F_c(\eta \bullet \mathsf{pr}_1^\lambda\Big)\\ &= \hat{F}\big(\,\mathsf{pr}_2(f, \eta)\big)\end{aligned}$$

This shows the last remaining property, so $(\hat{F}, \tilde{F}, F)$ is a map of higher comprehension categories. Q.E.D.

Lemma 7.2.13. *If $((\hat{F}, \tilde{F}, F))\colon (q, P, \chi) \to (q', P', \chi')$ is a map of higher comprehension categories, then F can be made a (2-dep,Σ)-functor of the associated (2-dep,Σ)-categories from Lemma 7.2.10 by setting*

$$F_c(\lambda) := \tilde{F}(\lambda) \qquad \textit{and} \qquad F_\lambda(\Phi) := \hat{F}(\Phi).$$

Proof: It is immediate that the F obtained in this manner is both a (2-fam,Σ)-functor of the underlying (2-fam,Σ)-categories from Lemma 3.2.10 and a 2-dep-functor of the underlying 2-dep-categories from Lemma 6.2.16. Thus by showing

$$F(\mathsf{pr}_2^\lambda) = \hat{F}\big(\,\mathsf{pr}_2(\lambda)\big) = \mathsf{pr}'_2\left(\tilde{F}(\lambda)\right) = \mathsf{pr}_2^{F(\lambda)}$$

we have that F is a (2-dep,Σ)-functor. Q.E.D.

Definition 7.2.14 (Transformation of maps of higher comprehension categories). Let $(\hat{F},\tilde{F},F),(\hat{G},\tilde{G},G)\colon(q,P,\chi)\to(q',P',\chi')$ are two maps of higher comprehension categories, we define a *transformation* to be a transformation between maps of sas-towers (see Definition 6.2.17) that is also a transformation of maps of the underlying comprehension categories (see Definition 3.2.5).

Lemma 7.2.15. *Any (2-dep,Σ)-natural transformation between (2-dep,Σ)-functors $F,G\colon\mathscr{C}\to\mathscr{D}$ induces a transformation between the associated maps of higher comprehension categories from Lemma 7.2.12.*

Proof: Immediate from the Lemmata 6.2.18 and 3.2.11. Q.E.D.

The reverse direction holds trivially as well.

Lemma 7.2.16. *Any transformation of maps $(\hat{F},\tilde{F},F)\Rightarrow(\hat{G},\tilde{G},G)$ induces a (2-dep,Σ)-natural transformation between the associated (2-dep,Σ)-functors of Lemma 7.2.13.*

Proof: Immediate from Lemmata 6.2.19 and 3.2.10. Q.E.D.

7.3 A comparison with generalised categories with families

Coraglia and Di Liberti introduced a notion very similar looking to higher discrete comprehension categories—generalised categories with families—in [15]. Coraglia and Emmenegger then proceed to show in [14] that generalised categories with families are biequivalent to comprehension categories.

We start by recalling the definition.

Definition 7.3.1 (generalised categories with families). A *generalised category with families* consists of a map of fibrations $\Sigma\colon\big(\dot{u}\colon\dot{\mathcal{U}}\to\mathscr{B}\big)\to\big(u\colon\mathcal{U}\to\mathscr{B}\big)$ together with an adjunction

$$\begin{tikzcd}
\dot{\mathcal{U}} \arrow[r, bend right, "\Sigma"'] \arrow[r, phantom, "\top"] \arrow[rd, bend right, "u"'] & \mathcal{U} \arrow[l, bend right, "\Delta"'] \arrow[d, "\dot{u}"] \\
& \mathscr{B}
\end{tikzcd}$$

such that all components of the unit and counit are cartesian with respect to $\dot{u}$ and u respectively.

Looking at the diagrams governing the two notions they appear eerily similar:

generalised categories with families	higher comprehension categories
$\begin{tikzcd} \dot{\mathcal{U}} \arrow[r, bend right, "\Sigma"'] \arrow[r, phantom, "\top"] \arrow[dr, bend right, "\dot{u}"'] & \mathcal{U} \arrow[l, bend right, "\Delta"'] \arrow[d, "u"] \\ & \mathscr{B} \end{tikzcd}$	$\begin{tikzcd} \mathscr{G} \arrow[r, bend right, "q"'] \arrow[r, phantom, "\overset{\chi}{\Rightarrow}"] \arrow[dr, bend right, "q"'] & \mathscr{E} \arrow[l, bend right, "\mathsf{pr}_2"'] \arrow[r, "P"] \arrow[d, "p"] & \mathscr{B}^{[1]} \arrow[dl, bend left, "\mathrm{cod}"] \\ & \mathscr{B} & \end{tikzcd}$

However, there are some differences, which we will emphasise in the following paragraphs, which will make it easier when we examine how these generalised categories relate to categories with dependent arrows, and how the biequivalence between comprehension categories and generalised categories with families fits into our picture.

7.3.2 Adjunction versus counit. The immediately visible difference (discarding the triangle on the right for higher comprehension categories) is that where generalised categories with families demand an adjunction, higher discrete comprehension categories demand only a natural transformation. Unpacking the definition of an adjunction one sees that an adjunction is a stricter version, as it unpacks to two natural transformations,

$$\text{the unit } \mathrm{id}_{\mathcal{U}} \Rightarrow \Delta \circ \Sigma \text{ and the counit } \Sigma \circ \Delta \Rightarrow \mathrm{id}_{\dot{\mathcal{U}}},$$

so our natural transformation χ plays the role of a counit.

7.3.3 Split versus unsplit. One difference between generalised categories with families is that they are concerned with normal fibration, whereas we demand that all fibrations come equipped with a normalised cleavage, i.e. are split. This is the reason we demand that all splittings are preserved by the functors involved, more precisely:

- in the definition of sas-towers in conditions (sas$_2$) and (sas$_4$), the equivalent part to (sas$_{??}$) in the definition of generalised categories with families is that Σ is a functor between fibrations and hence preserves cartesian morphisms. That it reflects cartesian morphisms can be deduced from the definition (see [14, Lemma 3.20.3]). The part of oplifts with respect to vertical arrows (with respect to u) is obviously missing, but we will explain later why this does not matter for the translation to comprehension categories.
- In condition (hCC$_3$) in the definition of higher comprehension categories the part c) corresponds to the counit consisting of (chosen) lifts of arrows in $\mathscr{E}$. That the counit for the adjunction in a generalised category with families is at least cartesian (as no specific chosen lifts are given) is guaranteed by the definition.
- That pr_2 preserves our choices of lifts for p corresponds to Δ respecting the cartesianness of morphisms, this is guaranteed by [14, Lemma 3.20.2].

7.3.4 The (bi-/2-)equivalences. The biequivalences described in [14] are between comprehension categories and weakening comonads, and between weakening comonads and generalised categories with families. Extending this picture with the notions of categories with families/dependent arrows and the corresponding equivalence proved earlier, that is as in Lemma 3.2.11, we get

$$\begin{tikzcd} (\mathsf{2\text{-}dep}, \Sigma)\mathsf{C} \arrow[r, shift right, "\text{2-}\simeq"'] & \mathsf{HComprC} \arrow[l, shift right] \arrow[rr, dashed, bend left, dash, "?"] & & \\ (\mathsf{2\text{-}fam}, \Sigma)\mathsf{C} \arrow[u] \arrow[r, shift left] & \mathsf{ComprC} \arrow[u] \arrow[l, shift left, "\text{2-}\simeq"] \arrow[r, shift right, "\text{bi-}\simeq"'] & \mathsf{WCmd} \arrow[l, shift right] \arrow[r, shift right, "\text{bi-}\simeq"'] & \mathsf{gCwF}, \arrow[l, shift right] \arrow[ull, dashed, dash, bend right] \end{tikzcd} \tag{14}$$

where the relation indicated with "?" is the one we want to explore in the following paragraphs. For this reason we compute the obtained generalised category with families from a (2-fam,Σ)-category $\mathscr{C}$ by applying all the functors in the above diagram. This generalised category with families has as $\dot{\mathcal{U}}$ the category $\mathscr{E}$ of family arrows and 2-family arrows, and $\mathcal{U}$ is the category of K-coalgebras, for the weakening comonad (K, ϵ, ν) defined via

1. $K(\lambda) = \lambda \circ \mathsf{pr}_1^\lambda$ for $\lambda \in \mathrm{fHom}(c)$, and for $(f, \eta)\colon \lambda \to \mu$
$$K(f, \eta) = \Big(\sum_\mu f \circ \sum_{\lambda,\mu} \eta, \eta \bullet \mathsf{pr}_1^\lambda \Big). \tag{15}$$
The well-definedness can be seen from the commutativity of
$$\begin{tikzcd}
\sum_c \lambda \arrow[r, "\sum_{\lambda,\mu\circ f} \eta"] \arrow[d, "\mathsf{pr}_1^\lambda"'] & \sum_c \mu \circ f \arrow[r, "\sum_\mu f"] \arrow[d, "\mathsf{pr}_1^{\mu\circ f}"] & \sum_{c'} \mu \arrow[d, "\mathsf{pr}_1^\mu"] \\
c \arrow[r, "1_c"'] & c \arrow[r, "f"'] & c',
\end{tikzcd}$$
which yields $\mu \circ \mathsf{pr}_1^\mu \circ \sum_\mu f \circ \sum_{\lambda,\mu\circ f} \eta = \mu \circ \mathsf{pr}_1^\lambda$.
2. $\epsilon_\lambda = (\mathsf{pr}_1^\lambda, 1_{\lambda\circ\mathsf{pr}_1^\lambda})\colon \lambda \circ \mathsf{pr}_1^\lambda \to \lambda$
3. $\nu_\lambda = (Q, 1_{\lambda\circ\mathsf{pr}_1^\lambda \circ \mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda}})\colon \lambda\circ\mathsf{pr}_1^\lambda \to \lambda\circ\mathsf{pr}_1^\lambda \circ \mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda}$ where Q stems from the pullback diagram

$$\begin{tikzcd}
\sum_c \lambda \arrow[rrd, dotted, "Q"] \arrow[rrrd, bend left, "\mathrm{id}_{\sum_c \lambda}"] \arrow[rrdd, bend right, "\mathrm{id}_{\sum_c \lambda}"'] & & & \\
& & \sum_{\sum_c \lambda} \lambda \circ \mathsf{pr}_1^\lambda \arrow[r, "\sum_\lambda \mathsf{pr}_1^\lambda"] \arrow[d, "\mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda}"'] & \sum_c \lambda \arrow[d, "\mathsf{pr}_1^\lambda"] \\
& & \sum_c \lambda \arrow[r, "\mathsf{pr}_1^\lambda"'] & c.
\end{tikzcd} \tag{16}$$

This category $\mathsf{CoAlg}(K)$ can be spelled out explicitly. It has as

- objects arrows $(\Phi, \eta)\colon \lambda \to \lambda \circ \mathsf{pr}_1^\lambda$ such that
$$\begin{tikzcd}
\lambda \arrow[r, "{(\Phi,\eta)}"] \arrow[rd, "{(1_c,1_\lambda)}"'] & \lambda \circ \mathsf{pr}_1^\lambda \arrow[d, "{(\mathsf{pr}_1^\lambda,1_\lambda)}"] \\
& \lambda
\end{tikzcd}
\quad \text{and} \quad
\begin{tikzcd}
\lambda \arrow[r, "{(\Phi,\eta)}"] \arrow[d, "{(\Phi,\eta)}"'] & \lambda \circ \mathsf{pr}_1^\lambda \arrow[d, "\nu_\lambda"] \\
\lambda \circ \mathsf{pr}_1^\lambda \arrow[r, "{K(\Phi,\eta)}"'] & \lambda \circ \mathsf{pr}_1^\lambda \circ \mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda}
\end{tikzcd}$$
commute. The commutativity of the first diagram simply spells out that both $\mathsf{pr}_1^\lambda \circ \Phi = 1_c$ and $(1_\lambda \bullet \mathsf{pr}_1^\lambda) \circ \eta = 1_\lambda$, which simplifies to $\eta = 1_\lambda$. This in turn can be used for the commutativity of the second diagram, which entails both
$$Q \circ \Phi = \sum_{\lambda\circ\mathsf{pr}_1^\lambda} \Phi \circ \sum_{\lambda,\lambda\circ\mathsf{pr}_1^\lambda} \eta \circ \Phi \quad \text{and} \quad \eta \bullet (\mathsf{pr}_1^\lambda \circ \Phi) \circ \eta = 1_{\lambda\circ\mathsf{pr}_1^\lambda \circ \mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda}} \bullet \Phi \circ \eta.$$
We immediately see that the left hand side of the second equality simplifies to $\eta \circ \eta = 1_\lambda$, and the right hand side simplifies to 1_λ as well, so this equality always holds. For the

first equality we observe that as $\eta = 1_\lambda$ we have $\sum_{\lambda,\lambda\circ\mathsf{pr}_1^\lambda} \eta = 1_{\sum_c \lambda}$, and we can extend the pullback diagram (16) to

$$\begin{tikzcd}
c \arrow[r, "\Phi"] \arrow[dr, dashed] \arrow[ddr, bend right, "1_c"'] & \sum_c \lambda \arrow[d, "1_{\sum_c \lambda}"] \arrow[dr, dotted, "Q"] \arrow[rrd, bend left, "\Phi"] & & \\
& \sum_c \lambda \arrow[r, "\sum_{\lambda\circ\mathsf{pr}_1^\lambda} \Phi"'] \arrow[d, "\mathsf{pr}_1^\lambda"'] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_{\sum_c \lambda} \lambda \circ \mathsf{pr}_1^\lambda \arrow[r, "\sum_\lambda \mathsf{pr}_1^\lambda"] \arrow[d, "\mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \sum_c \lambda \arrow[d, "\mathsf{pr}_1^\lambda"] \\
& c \arrow[r, "\Phi"'] & \sum_c \lambda \arrow[r, "\mathsf{pr}_1^\lambda"'] & c.
\end{tikzcd}$$

It is immediate that the dashed arrow has to be Φ, which yields the desired equality. Thus (Φ, η) can be reduced to Φ such that

$$\begin{tikzcd}
c \arrow[r, "\Phi"] \arrow[dr, "1_c"'] & \sum_c \lambda \arrow[d, "\mathsf{pr}_1^\lambda"] \\
& c
\end{tikzcd}$$

commutes.

- An arrow $(f, \eta)\colon (\Phi\colon c \to \sum_c \lambda) \to (\Psi\colon d \to \sum_d \mu)$ is an arrow $f\colon c \to d$ together with a 2-fam-arrow $\eta\colon \lambda \Rightarrow \mu \circ f$ such that

$$\begin{tikzcd}
\lambda \circ \mathsf{pr}_1^\lambda \arrow[d, "(\sum_{\mu f} \circ \sum_{\lambda,\mu} \eta{,}\eta\bullet\mathsf{pr}_1^\lambda)"'] & \lambda \arrow[l, "(\Phi{,}1_\lambda)"] \arrow[d, "(f{,}\eta)"] \\
\mu \circ \mathsf{pr}_1^\mu & \mu \arrow[l, "(\Psi{,}1_\mu)"]
\end{tikzcd}$$

commutes. This can be reduced to the commutativity of both

$$\sum_\mu f \circ \sum_{\lambda,\mu} \eta \circ \Phi = \Psi \circ f \quad \text{and} \quad \big(\eta \bullet (\mathsf{pr}_1^\lambda \circ \Phi)\big) \circ 1_\lambda = (1_\mu \bullet f) \circ \eta.$$

Again the second equality is immediate (both sides equate to η), so only the first equality has to be checked. Thus the condition on (f, η) reduces to the commutativity of

$$\begin{tikzcd}
c \arrow[rr, "1_c"] \arrow[dr, "\Phi"] \arrow[dd, "1_c"'] & & c \arrow[rr, "f"] \arrow[dr, "\eta\circ\Phi"] \arrow[dd, "1_c"] & & d \arrow[dr, "\Psi"] \arrow[dd, "1_d"] & \\
& \sum_c \lambda \arrow[rr, "\sum_{\lambda,\mu\circ f} \eta"] \arrow[dl, "\mathsf{pr}_1^\lambda"] & & \sum_c \mu \circ f \arrow[rr, "\sum_\mu f"] \arrow[dl, "\mathsf{pr}_1^{\mu\circ f}"] & & \sum_d \mu \arrow[dl, "\mathsf{pr}_1^\mu"] \\
c \arrow[rr, "1_c"'] & & c \arrow[rr, "f"'] & & d. &
\end{tikzcd}$$

The only nontrivial condition of this is $\sum_\mu f \circ \eta \circ \Phi = \Psi \circ f$. However, this condition is equivalent to $\eta \circ \Phi = [\Psi](f)$.

It is thus immediate that the objects of $\mathsf{CoAlg}(K)$ are the canonical dependent arrows of a (2-fam,Σ)-category introduced in [20] and the arrows are obtained as combinations of lifts of $(f, \eta)\colon \lambda \to \mu$ such that $\eta \circ \Phi = [\Psi](f)$.

Conversely, if we first compute the canonical dependent arrow structure for a (2-fam,Σ)-category and then the higher comprehension category obtained from the 2-equivalence consists of the following data:

- The comprehension category structure $P\colon \mathscr{E} \to \mathscr{B}^{[1]}$ consists of (2-)family arrow and $\mathscr{C}$
- The category $\mathscr{G}$ has as objects coalgebras $c \to \sum_c \lambda$ for the weakening comonad K described earlier. Morphisms $\Phi \to \Psi$—where $q(\Phi) = c$ and $p(\Psi) = c'$—are given by $(f, \eta)\colon \lambda \to \mu$ such that $\eta \circ \Phi = [\Psi](f)$.

From the discussion above we thus infer that $\mathscr{G} = \mathsf{CoAlg}(K)$.

Comparing the functors, we compute that Σ takes $\Phi\colon c \to \sum_c \lambda$ to λ and (f,η) to (f,η), hence $q = \Sigma$. The functor Δ takes λ to Q from $\nu_\lambda = (Q, 1_{\lambda\circ\mathsf{pr}_1^\lambda \circ \mathsf{pr}_1^{\lambda\circ\mathsf{pr}_1^\lambda}})$ and (f,η) to $K(f,\eta)$ defined in (15). Examining how the second projection arrow is defined in the canonical way in [48] we see that $\mathsf{pr}_2^\lambda = Q$. A similar computation shows that also $\mathsf{pr}_2(f,\eta) = K(f,\eta)$. Thus $\mathsf{pr}_2(\lambda) = Q$.

The adjunction

$$\begin{tikzcd}
\mathsf{CoAlg}(K) \arrow[r, shift right=1.5ex, "\Sigma"'] \arrow[r, phantom, "\top"] & \mathscr{E}. \arrow[l, shift right=1.5ex, "\Delta"']
\end{tikzcd}$$

is not part of the higher comprehension category, but the natural transformation $\chi\colon \lambda\circ\mathsf{pr}_1^\lambda \to \lambda$ can be extended to obtain this adjunction, as χ and the counit coincide.

Thus we obtain that the diagram (14) can be extended to

$$\begin{tikzcd}
(2\text{-}\mathsf{dep}, \Sigma)\mathsf{C} \arrow[r, shift right] \arrow[r, phantom, "2\text{-}\simeq"] & \mathsf{HComprC} \arrow[l, shift right] & & \\
(2\text{-}\mathsf{fam}, \Sigma)\mathsf{C} \arrow[u] \arrow[r, shift left] \arrow[r, phantom, "2\text{-}\simeq"] & \mathsf{ComprC} \arrow[u] \arrow[l, shift left] \arrow[r, shift right] \arrow[r, phantom, "\text{bi-}\simeq"] & \mathsf{WCmd} \arrow[l, shift right] \arrow[r, shift right] \arrow[r, phantom, "\text{bi-}\simeq"] & \mathsf{gCwF}\,. \arrow[l, shift right] \arrow[llu, bend right]
\end{tikzcd}$$

The biequivalence $\mathsf{ComprC} \simeq \mathsf{gCwF}$ proved in [14] actually yields more than [20, Theorem 3.5.8]: not only does it give the canonical (dep,Σ)-structure given a (2-fam,Σ)-category, but also says that if we are given a (2-dep,Σ)-category, such that in the higher comprehension category, formed as in Lemma 7.2.9, the natural transformation $\chi\colon q \circ \mathsf{pr}_2 \Rightarrow \mathrm{id}_{\mathscr{E}}$ can be extended to an adjunction $q \dashv \mathsf{pr}_2$, then we obtain a pseudo-natural isomorphism $\mathscr{G} \to \mathsf{CoAlg}(K)$, where K is the comonad defined by bullet points 1–3 on page 75.

Part II

Computability models induced by categories

Chapter 8
Computability models

8.1 Computability models

The original definition due to Longley only considers the case that T is a proper set, but we do not require this restriction.

Definition 8.1.1 (Computability models). A *computability model* $\mathbf{C}$ over a class T is a pair

$$\mathbf{C} = \Big(\big(\mathbf{C}(\tau)\big)_{\tau \in T}, \big(\mathbf{C}[\sigma, \tau]\big)_{\sigma, \tau \in T} \Big),$$

of families of sets such that all $\mathbf{C}(\tau)$ are sets and all $\mathbf{C}[\tau, \sigma]$ are sets of partial functions between $\mathbf{C}(\sigma)$ and $\mathbf{C}(\tau)$ such that:

1. For all $\tau \in T$ we have $\mathbf{1}_{\mathbf{C}(t)} \in \mathbf{C}[\tau, \tau]$.
2. For all $\sigma, \tau, \rho \in T$ and all $f \in \mathbf{C}[\sigma, \tau], g \in \mathbf{C}[\tau, \rho]$ there exists $g \circ f \in \mathbf{C}[\sigma, \rho]$.

Many examples of computability models can be found in [45, subsection 3.1.2, pp. 54]. We will repeat a few of the most important ones here.

Examples 8.1.2 (see [45]).

1. If one takes as underlying set the set $\mathbb{1}$ and only data type the set $\mathbb{N}$ and lets the computable functions be the Turing computable ones, then one obtains *Kleene's first model* K_1, perhaps the most fundamental of all computability models.
2. The terms of the untyped λ-calculus are generated from a set V of variable symbols x by the grammar

   $$M ::= x \mid MM \mid \lambda x.M.$$

   Let Λ be the set of terms obtained in this way modulo α-equivalence. Write $M[x \mapsto N]$ for the term that is obtained by replacing all free occurrences of x in M by N. If $\sim$ is a equivalence relation on Λ satisfying

   $$(\lambda x.M)N \sim M[x \mapsto N], \quad M \sim N \Rightarrow PM \sim PN$$

   for all terms P, then we can define a computability model on $\Lambda/\sim$ in the following manner:

   The underlying set is again $\mathbb{1}$ and the only datatype is $\Lambda/\sim$ the computable functions are the mappings $\Lambda/\sim \to \Lambda/\sim, [M] \mapsto [PM]$ induced by the terms P.
3. This example is actually a class of examples. For this let $B = (B_i)_{i \in I}$ be a given family of sets and let $F = (F_{(i,j)})_{(i,j) \in I^2}$ be a family of sets where for each $(i, j) \in I^2$ we have that $F_{(i,j)} \subseteq B_j^{B_i}$. Denote by $\langle B \rangle$ the family of sets by adding $\mathbb{1}$ to B and closing under products and function spaces. Define a computability model $K(B, F)$ by letting the underlying set be I and setting $K(B, F)(i) = B_i$ for all $i \in I$ and the computable functions are defined as follows:
   - For each $i, j \in I$, we have that $F_{(i,j)} \subset K(B, F)[i, j]$.
   - For each i we have that $\{B_i \to \mathbb{1}\} \subset K(B, F)(i, \mathbb{1})$.

- For each $i, j \in I$ we have that
$$\mathsf{pr}_1 \in K(B,F)[i \times j, i] \quad \text{and} \quad \mathsf{pr}_2 \in K(B,F)[i \times j, j].$$
- If $f \in K(B,F)[i,j]$ and $g \in K(B,F)[j,k]$, then $g \circ f \in K(B,F)[i,k]$.
- If $f \in K(B,F)[i,j]$ and $g \in K(B,F)[i,k]$, then $\langle f, g\rangle \in K(F,B)(i, j \times k)$.
- If $f \in K(B,F)[i \times j, k]$, then $\hat{f} \in K(B,F)[i, k^j]$.

Remark 8.1.3. In another work Longley defines the notion of computability model slightly different: In [43] he writes that the datatype sets need to be inhabited and may not be empty. Although a small difference at first we will later discuss why this distinction actually matters. Furthermore he allows the computable function to be relations rather than partial functions. We will discuss this approach in subsection 8.2.7.

As we want to discuss the connection between computability models and category theory, we highlight the method to associate to a category a computability model introduced by Petrakis [49]. A small notational convention beforehand:

Notation: Let $\mathscr{C}, \mathscr{D}$ be categories with pullbacks and $S\colon \mathscr{C} \to \mathscr{D}$ be a pullback-preserving functor. We denote a span of the kind

$$\begin{tikzcd}
c \arrow[r, hook, "i"] \arrow[dr, "f"] & a \\
& b
\end{tikzcd}$$

where i is monic as $(i,f)\colon a \rightharpoonup b$. As S is pullback-preserving $S(i)$ is monic as well and we denote $\big(S(i), S(f)\big)\colon S(a) \rightharpoonup S(b)$ as $S(i,f)$.

Definition 8.1.4 (Canonical (total/partial) computability models). Let $\mathscr{C}$ be a category and $S\colon \mathscr{C} \to \mathsf{Sets}$ be a presheaf.

- The *canonical total computability model* $\mathbf{CM}^{\mathrm{tot}}(\mathscr{C}; S)$ associated to $\mathscr{C}$ is defined as
$$\Big(\big(S(a)\big)_{a \in \mathscr{C}}, \big(\{S(f) \mid f\colon a \to b\}\big)_{a,b \in \mathscr{C}}\Big).$$
From this is apparent that the underlying class of $\mathbf{CM}^{\mathrm{tot}}(\mathscr{C}; S)$ is $\mathscr{C}$.
- If $\mathscr{C}$ has pullbacks and S preserves those pullbacks, then the *canonical partial computability model* $\mathbf{CM}^{\mathrm{prt}}(\mathscr{C}; S)$ associated to $\mathscr{C}$ is defined as
$$\Big(\big(S(a)\big)_{a \in \mathscr{C}}, \big(\{S(i,f) \mid (i,f)\colon a \rightharpoonup b\}\big)_{a,b \in \mathscr{C}}\Big).$$

One immediately notices the similarities between these computability models associated to categories and the computability models constructed on a given family of sets we constructed earlier. Namely giving a category $\mathscr{C}$ and a presheaf S amounts to giving exactly such families as above where $I = \mathscr{C}_0, B_i = S(i)$ and $F_{(i,j)} = \{S(f) \mid f\colon i \to j\}$.

8.2 Simulations

Definition 8.2.1 (Simulations). Let $\mathbf{C}$ be a computability model over the class T and $\mathbf{D}$ be a computability model over the class U. A *simulation* $\boldsymbol{\gamma}$ between $\mathbf{C}, \mathbf{D}$, written $\boldsymbol{\gamma}\colon \mathbf{C} \mathrel{\rightarrow\!\!\!\triangleright} \mathbf{D}$ consists of the following data: A class function $\gamma\colon T \to U$ and for each $\tau \in T$ a subset $\Vdash^{\gamma}_{\tau} \subseteq \mathbf{D}(\gamma(\tau)) \times \mathbf{C}(\tau)$, such that the following two conditions are satsified:

($\mathtt{Siml}_1$) For every $\tau \in T$ and each $x \in \mathbf{C}(\tau)$ exists a $y \in \mathbf{D}(\gamma(\tau))$ such that $(y, x) \in \Vdash^{\gamma}_{\tau}$. We also write $y \Vdash^{\gamma}_{\tau} x$ instead of $(y, x) \in \Vdash^{\tau}_{\gamma}$.

($\mathtt{Siml}_2$) For every $\sigma, \tau \in T$ and each $f \in \mathbf{C}[\sigma, \tau]$ exists a $f' \in \mathbf{D}[\gamma(\sigma), \gamma(\tau)]$ such that $f' \Vdash^{\gamma}_{(\sigma,\tau)} f$, where $f' \Vdash^{\gamma}_{(\sigma,\tau)} f$ stands for the following:

$$\forall x \in \mathbf{C}(\sigma)\, \forall y \in \mathbf{D}(\gamma(\tau)) :$$
$$\big(x \in \mathrm{dom}(f) \& y \Vdash^{\gamma}_{\sigma} x \Rightarrow y \in \mathrm{dom}(f') \,\&\, f'(y) \Vdash^{\gamma}_{\tau} f(x)\big).$$

Example 8.2.2. If we two categories $\mathscr{C}, \mathscr{D}$ with presheaves S, S' respectively, then any functor $F \colon \mathscr{C} \to \mathscr{D}$ making the diagram

$$\begin{tikzcd} \mathscr{C} \arrow[rr, "F"] \arrow[dr, "S"'] & & \mathscr{D} \arrow[dl, "S'"] \\ & \mathsf{Sets} & \end{tikzcd} \tag{17}$$

commute corresponds to a simulation $\boldsymbol{\gamma}^F \colon \mathbf{CM}^{\mathrm{tot}}(\mathscr{C}; S) \rightarrowtriangle \mathbf{CM}^{\mathrm{tot}}(\mathscr{C}; S')$ defined as follows:

- The underlying class function is given via $\gamma^F(c) = F(c)$ for all $c \in \mathscr{C}$.
- The tracking relations $\Vdash^{\gamma^F}_{c} \subseteq S'\big(F(c)\big) \times S(c)$ are given by

$$s_1 \Vdash^{\gamma^F}_{c} s_2 :\Leftrightarrow s_1 = s_2.$$

This is possible as by (17) $S'\big(F(c)\big) = S(c)$.

The above is an example of an important class of simulations, the so-called equality simulations.

Definition 8.2.3 (Equality simulations). If $\mathbf{C}, \mathbf{D}$ are simulations over T, U respectively, then a simulation $\boldsymbol{\gamma} \colon \mathbf{C} \rightarrowtriangle \mathbf{D}$ is an *equality simulation*, if for all $t \in T$ the following three properties are satisfied:

(EqSim_1) For all $t \in T$ we have that $\mathbf{C}(t) = \mathbf{D}\big(\gamma(t)\big)$.

(EqSim_2) For all $t \in T$ we have that
$$d \Vdash^{\gamma}_{t} c \Leftrightarrow d = c.$$

(EqSim_3) For all $t, t' \in T$ we have that $\mathbf{C}[t, t'] \subseteq \mathbf{D}[t, t']$.

Another (class of) example(s) of simulations can be obtained as follows:

Example 8.2.4. If $\mathscr{C}$ is a category and S, S' presheaves on that category, then any natural transformation $\eta \colon S \to S'$ an be translated into a simulation $\boldsymbol{\eta} \colon \mathbf{CM}^{\mathrm{tot}}(\mathscr{C}; S) \rightarrowtriangle \mathbf{CM}^{\mathrm{tot}}(\mathscr{C}; S')$ as follows:

- The underlying class function is given by the identity on $\mathscr{C}$.
- The tracking relations $\Vdash^{\eta}_{c}$ are defined via

$$s' \Vdash^{\eta}_{c} s :\Leftrightarrow \eta_c(s) = s'.$$

The simulations in example 8.2.4 are instances of so-called natural simulations.

Definition 8.2.5 (Natural simulations). A simulation $\boldsymbol{\gamma} \colon \mathbf{C} \rightarrowtriangle \mathbf{C}'$ where the models live over the same class T is a *natural simulation* if the following conditions are fulfilled:

(NatSim_1) The underlying class function is the identity on T.

(NatSim$_2$) For all $t \in T$ there exists a $\gamma_t^*\colon \mathbf{C}(t) \to \mathbf{C}'(t)$ such that for all $x \in \mathbf{C}(t)$ and $y \in \mathbf{C}'(t)$ we have that $y \Vdash_t^\gamma x$ if and only if $y = \gamma_t^*(x)$.

(NatSim$_3$) For all $t, t' \in T$ and all $f \in \mathbf{C}[t,t']$ there exists a $f' \in \mathbf{C}[t,t']$ such that for all $x \in (t)$ we have that if $x \in \operatorname{dom}(f)$ then $\gamma_t^*(x) \in \operatorname{dom}(f')$ and

$$f'\big(\gamma_t^*(x)\big) \Vdash_{t'}^\gamma \gamma_t^*\big(f(x)\big).$$

This definition might suggest that all natural simulations between computability models obtained from categories in the previously given manner arise from natural transformations between the categories. The following proposition shows that this is not the case.

Proposition 8.2.6. *Consider the category given by*

$$\begin{tikzcd}
a \arrow[r, bend left=50, "f_1"] \arrow[r, bend left=15, "f_2"] & b, \arrow[l, bend left=15, "g_2"] \arrow[l, bend left=50, "g_1"] \arrow[loop right, "q"]
\end{tikzcd}$$

where $f_1 \neq f_2$ as well as $g_1 \neq g_2$ and

$$\begin{aligned}
f_1 \circ g_1 = \mathbf{1}_b = f_1 \circ g_2 = f_2 \circ g_2,\\
f_2 \circ g_1 = q,\\
q^2 = \mathbf{1}_b, g_2 \circ q = g_1, g_1 \circ q = g_2, q \circ f_1 = f_2, q \circ f_2 = f_1,\\
g_1 \circ f_1 = g_1 \circ f_2 = g_2 \circ f_1 = g_2 \circ f_2 = \mathbf{1}_a.
\end{aligned}$$

Let $\boldsymbol{\gamma}\colon \mathbf{CM}^{\mathrm{tot}}(\mathscr{C}; \operatorname{Hom}(a, \text{-})) \multimap \mathbf{CM}^{\mathrm{tot}}(\mathscr{C}; \operatorname{Hom}(b, \text{-}))$ be the natural simulation given by

$$\begin{aligned}
&\gamma_a^*\colon \operatorname{Hom}(a,a) \to \operatorname{Hom}(b,a), \mathbf{1}_a \mapsto g_1,\\
&\gamma_b^*\colon \operatorname{Hom}(a,b) \to \operatorname{Hom}(b,b), f_1 \mapsto f_2 \circ g_1, f_2 \mapsto f_1 \circ g_1.
\end{aligned}$$

Then $(\gamma_a^*)_{a \in \mathscr{C}_0}$ is not a natural transformation $\operatorname{Hom}(a, \text{-}) \Rightarrow \operatorname{Hom}(b, \text{-})$.

Proof: We first have to prove that the simulation $\boldsymbol{\gamma}$ given through the γ_a^* is actually natural. For this, we check the axioms of a natural simulation.

(`NatSim`$_1$) By definition we have $\gamma = \mathrm{id}_{\mathscr{C}_0}$.

(`NatSim`$_2$) By definition we have $y \Vdash_a^\gamma x \Leftrightarrow y = \gamma_a^*(x)$ for all x, y, a.

(`NatSim`$_3$) We have to consider all possible cases.

We start of with the case $\operatorname{Hom}(a, \text{-})^{\mathrm{tot}}[a,a]$. As there is only one arrow $\mathbf{1}_a\colon a \to a$, it suffices to remark that the diagram

$$\begin{tikzcd}
\operatorname{Hom}(a,a) \arrow[r, "\mathbf{1}_a \circ \text{-}"] \arrow[d, "\gamma_a^*"'] & \operatorname{Hom}(a,a) \arrow[d, "\gamma_a^*"] \\
\operatorname{Hom}(b,a) \arrow[r, "\mathbf{1}_a \circ \text{-}"] & \operatorname{Hom}(b,a)
\end{tikzcd}$$

commutes, as

$$\gamma_a^*(\mathbf{1}_a \circ \mathbf{1}_a) = \gamma_a^*(\mathbf{1}_a) = g_1 = \mathbf{1}_a \circ g_1 = \mathbf{1}_a \circ \gamma_a^*(\mathbf{1}_a).$$

Next we consider the case $\operatorname{Hom}(a, \text{-})^{\mathrm{tot}}[a,b]$. Here we have two possible arrows $f_1, f_2\colon a \to b$. We start of with f_1. One can see that the diagram

$$\begin{tikzcd}
\operatorname{Hom}(a,a) \arrow[r, "f_1 \circ \text{-}"] \arrow[d, "\gamma_a^*"'] & \operatorname{Hom}(a,b) \arrow[d, "\gamma_b^*"] \\
\operatorname{Hom}(b,a) \arrow[r, "f_2 \circ \text{-}"'] & \operatorname{Hom}(b,b)
\end{tikzcd}$$

commutes, as there is only $\mathbf{1}_a \in \mathrm{Hom}(a,a)$ and

$$\gamma_b^*(f_1 \circ \mathbf{1}_1) = \gamma_b^*(f_1) = f_2 \circ g_1 = f_2 \circ \gamma_a^*(\mathbf{1}_a).$$

If f_2 is considered we can see that the diagram

$$\begin{tikzcd}
\mathrm{Hom}(a,a) \arrow[r, "f_2 \circ \text{-}"] \arrow[d, "\gamma_a^*"'] & \mathrm{Hom}(a,b) \arrow[d, "\eta_b^*"] \\
\mathrm{Hom}(b,a) \arrow[r, "f_1 \circ \text{-}"] & \mathrm{Hom}(b,b)
\end{tikzcd}$$

commutes, as

$$\gamma_b^*(f_2 \circ \mathbf{1}_a) = \gamma_b^*(f_2) = f_1 \circ g_1 = f_1 \circ \gamma_a^*(\mathbf{1}_a).$$

Next we consider the case $\mathrm{Hom}(a, \text{-}\,)^{\mathrm{tot}}[b,a]$. Here we have two possible arrows, $g_1, g_2 \colon b \to a$. We start with the case g_1. One can see that the diagram

$$\begin{tikzcd}
\mathrm{Hom}(a,b) \arrow[r, "g_1 \circ \text{-}"] \arrow[d, "\gamma_b^*"] & \mathrm{Hom}(a,a) \arrow[d, "\gamma_a^*"] \\
\mathrm{Hom}(b,b) \arrow[r, "g_1 \circ \text{-}"'] & \mathrm{Hom}(b,a)
\end{tikzcd}$$

commutes, as the only arrows in $\mathrm{Hom}(a,b)$ are f_1, f_2 and we can compute

$$\begin{aligned}
\gamma_a^*(g_2 \circ f_1) &= \gamma_a^*(\mathbf{1}_a) = g_1 = \mathbf{1}_a \circ g_1 \\
&= g_1 \circ f_2 \circ g_1 = g_1 \circ \gamma_b^*(f_1), \\
\gamma_a^*(g_2 \circ f_2) &= \gamma_a^*(\mathbf{1}_a) = g_1 = \mathbf{1}_a \circ g_1 \\
&= g_1 \circ f_1 \circ g_1 = g_1 \circ \gamma_b^*(f_2).
\end{aligned}$$

Similarly, we have that the diagram

$$\begin{tikzcd}
\mathrm{Hom}(a,b) \arrow[r, "g_2 \circ \text{-}"] \arrow[d, "\gamma_a^*"] & \mathrm{Hom}(a,a) \arrow[d, "\gamma_a^*"] \\
\mathrm{Hom}(b,b) \arrow[r, "g_1 \circ \text{-}"] & \mathrm{Hom}(b,a)
\end{tikzcd}$$

commutes, as we can compute

$$\begin{aligned}
\gamma_a^*(g_2 \circ f_1) &= \gamma_a^*(\mathbf{1}_a) = g_1 = g_1 \circ \mathbf{1}_a \\
&= g_1 \circ f_2 \circ g_1 = g_1 \circ \gamma_b^*(f_1), \\
\gamma_a^*(g_2 \circ f_2) &= \gamma_a^*(\mathbf{1}_a) = g_1 = g_1 \circ \mathbf{1}_a \\
&= g_1 \circ f_1 \circ g_1 = g_1 \circ \gamma_b^*(f_2).
\end{aligned}$$

At last we consider the case $\mathrm{Hom}(a, \text{-}\,)^{\mathrm{tot}}[b,b]$. Here we have to consider two morphisms, $\mathbf{1}_b, q \colon b \to b$. We start with the case $\mathbf{1}_b$. One can see that the diagram

$$\begin{tikzcd}
\mathrm{Hom}(a,b) \arrow[r, "\mathbf{1}_b \circ \text{-}"] \arrow[d, "\gamma_b^*"] & \mathrm{Hom}(a,b) \arrow[d, "\gamma_b^*"] \\
\mathrm{Hom}(b,b) \arrow[r, "\mathbf{1}_b \circ \text{-}"] & \mathrm{Hom}(b,b)
\end{tikzcd}$$

commutes trivially, so we can redirect our attention to the case q. In this case we obtain the commutative diagram

$$\begin{tikzcd}
\mathrm{Hom}(a,b) \arrow[r, "q\circ\text{-}"] \arrow[d, "\gamma_b^*"] & \mathrm{Hom}(a,b) \arrow[d, "\gamma_b^*"] \\
\mathrm{Hom}(b,b) \arrow[r, "q\circ\text{-}"] & \mathrm{Hom}(b,b)
\end{tikzcd}$$

which is commutative because the only arrows in $\mathrm{Hom}(a,b)$ are f_1, f_2 and we can compute

$$\begin{aligned}
\gamma_b^*(q\circ f_1) &= \gamma_b^*(f_2) = f_1\circ g_1 = q\circ f_2\circ g_1 \\
&= q\circ\gamma_b^*(f_1), \\
\gamma_b^*(q\circ f_2) &= \gamma_b^*(f_1) = f_2\circ g_1 = q\circ f_1\circ g_1 \\
&= q\circ\gamma_b^*(f_2).
\end{aligned}$$

This finishes the proof that $\boldsymbol{\gamma}$ is indeed a natural simulation.

To prove that $(\gamma_a^*)_{a\in\mathscr{C}_0}$ does not constitue a natural transformation, we show that the diagram

$$\begin{tikzcd}
\mathrm{Hom}(a,a) \arrow[r, "f_1\circ\text{-}"] \arrow[d, "\gamma_a^*"'] & \mathrm{Hom}(a,b) \arrow[d, "\gamma_b^*"] \\
\mathrm{Hom}(b,a) \arrow[r, "f_1\circ\text{-}"] & \mathrm{Hom}(b,b)
\end{tikzcd}$$

does not commute. One the one hand one can calculate

$$\gamma_b^*(f_1\circ\mathbf{1}_a) = \gamma_b^*(f_1) = f_2\circ g_1,$$

but we also obtain

$$f_1\circ\gamma_a^*(\mathbf{1}_a) = f_1\circ g_1,$$

and by design we have $f_2\circ g_1 \neq f_1\circ g_1$. Q.E.D.

8.2.7 Simulations in other contexts. In [50] poses the question whether one can define computability models where the data types do not live in the category Sets, but another category. Such an approach is taken by Cockett and Hofstra in [10], which bases on [11, 12, 13]. In this work one first considers restriction categories, which are categories where the domains of a morphism $f\colon c\to c'$ is realised not as an object but as an endomorphism $\overline{f}\colon c\to c$. This definition reads as follows:

Definition 8.2.8 (Restriction categories). A category $\mathscr{C}$ is a *restriction category* if it comes equipped with a map $\overline{(\text{-})}$ that maps an arbitrary morphism $f\colon c\to c'$ to a morphism $\overline{f}\colon c\to c$. This map $\overline{(\text{-})}$ must be subject to the following conditions:

1. For all f we have that $f\circ\overline{f} = f$.
2. For all $f\colon c\to c', g\colon c\to c''$ we have $\overline{f}\circ\overline{g} = \overline{g}\circ\overline{f}$.
3. For all $f\colon c\to c', g\colon c\to c''$ we have $\overline{g\circ\overline{f}} = \overline{g}\circ\overline{f}$.
4. For all $f\colon c\to c', g\colon c'\to c''$ we have $\overline{g}\circ f = f\circ\overline{g\circ f}$.

The map $\overline{(\text{-})}$ is called the *restriction map.*

It is immediate that the presence of such a restriction map allows us to define a partial order on each of the Hom-sets, by

$$f\preceq g :\Leftrightarrow g\circ\overline{f} = f.$$

Example 8.2.9. The easiest example of a restriction category is the category of sets and *partial* functions. The domain of a partial function $f\colon S \rightharpoonup S'$ is then simply the partial function

$$\overline{f}\colon S \rightharpoonup S, \quad \overline{f}(x) = \begin{cases} x & \text{if } f(x) \text{ is defined,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

A computability model in Cockett's sense is then simply a functor $F\colon \mathscr{C} \to \mathscr{R}$ where $\mathscr{R}$ is a restriction category. He defines a simulation as follows:

Definition 8.2.10 (Simulations - due to Cockett). Let $F\colon \mathscr{C} \to \mathscr{R}, G\colon \mathscr{D} \to \mathscr{R}$ be computability models. A *simulation* consists of

- a map K that maps objects of $\mathscr{C}$ to objects of $\mathscr{D}$ and
- for each object $c \in \mathscr{C}$ a morphism $\alpha\colon F(c) \to G\big(K(c)\big)$ such that for each $f\colon c \to c'$ in $\mathscr{C}$ there exists a $f_\alpha\colon K(c) \to K(c')$ such that

$$\begin{tikzcd}
F(c) \arrow[r, "\alpha_c"] \arrow[d, "F(f)"'] \arrow[dr, phantom, "\succeq"] & G\big(K(c)\big) \arrow[d, "G(f_\alpha)"] \\
F(c') \arrow[r, "\alpha_{c'}"] & G\big(K(c')\big).
\end{tikzcd}$$

We write $(K, \alpha)\colon F \rightsquigarrow G$ if (K, α) constitutes a simulation.

Remarks: If one takes the restriction category $\mathscr{R}$ to be the category of sets and partial functions, then the definition of a computability model is identical to the computability models we obtained from categories. Also the notion of simulation seems very similar to the notion defined by Longley and Norman. However there are a few key differences:

- Firstly, in the definition due to Cockett each element in a data type can only be tracked by another element, whereas in the definition of Longley an element in a data type can be tracked by many different elements.
- Secondly, which is really a consequence of the first observation, this fact can not be remedied by changing the underlying category: In the computability models due to Cockett the computable functions are partial functions, whereas the tracking relations are relations. Thus passing to the category of sets and relations as the underlying category (if one defines an appropriate restriction map on it) does not solve this problem, as it would alter the definition of computability models.

As a final remark we note that if one takes the restriction category to be the category of sets and relations on sets, then the notion of computability model that arises is very similar to the slightly different version of a computability model defined by Longley in [43], however in this restriction category approach the data types need not be inhabited.

8.3 Transformability

Definition 8.3.1. Given two simulations $\boldsymbol{\gamma}, \boldsymbol{\delta}\colon \mathbf{C} \twoheadrightarrow \mathbf{D}$, we say that $\boldsymbol{\gamma}$ is *transformable* into $\mathbf{D}$, in symbols $\boldsymbol{\gamma} \preceq \boldsymbol{\delta}$, if for each $\sigma \in T$ there exists a $t_\sigma \in \mathbf{D}[\gamma(\sigma), \delta(\sigma)]$, such that the following condition is satisfied:

(`Trafo`$_1$) For each $a \in \mathbf{C}(\sigma)$ and each $a' \in \mathbf{D}(\gamma(\tau))$ such that $a' \Vdash^{\gamma}_{\sigma} a$, we have that $t(a') \Vdash^{\delta}_{\sigma} a$.

If $\boldsymbol{\gamma} \preceq \boldsymbol{\delta}$ and $\boldsymbol{\delta} \preceq \boldsymbol{\gamma}$ then we write $\boldsymbol{\gamma} \sim \boldsymbol{\delta}$.

Definition 8.3.2. Let $\mathsf{CompMod}$ be the category whose

- *objects* are compuatbility models $\mathbf{C}$ over arbitrary classes and whose
- *morphisms* $\boldsymbol{\gamma}\colon \mathbf{C} \rightarrowtriangle \mathbf{D}$ in $\mathsf{CompMod}$ is a simulation.
- For each computability model $\mathbf{C}$ over the class T the identity simulation $\mathrm{id}_{\mathbf{C}}$ is given as $\mathrm{id}_{\mathbf{C}} = (\mathbf{1}_T, \mathrm{diag}^{\mathbf{C}}_{\tau})$, where $\mathrm{diag}^{\mathbf{C}}_{\tau} \subseteq \mathbf{C}(\tau) \times \mathbf{C}(\tau)$ is defined through
$$(y, x) \in \mathrm{diag}^{\mathbf{C}}_{\tau} :\Leftrightarrow y = x.$$
- The composition of two simulations $\boldsymbol{\gamma}\colon \mathbf{C} \rightarrowtriangle \mathbf{D}, \boldsymbol{\epsilon}\colon \mathbf{D} \rightarrowtriangle \mathbf{E}$ is defined as $\boldsymbol{\epsilon} \circ \boldsymbol{\gamma} = (\epsilon \circ \gamma, \Vdash^{\epsilon\circ\gamma}_{\tau})$, where $\Vdash^{\epsilon\circ\gamma}_{\tau} \subseteq \mathbf{E}\big(\epsilon(\gamma(\tau))\big) \times \mathbf{C}(\tau)$ is defined through the following clause:
$$z \Vdash^{\epsilon\circ\gamma}_{\tau} x :\Leftrightarrow \exists y \in \mathbf{D}(\gamma(\tau) : z \Vdash^{\epsilon}_{\gamma(\tau)} y \,\&\, y \Vdash^{\gamma}_{\tau} x.$$

Definition 8.3.3 (Equivalence of computability models). If $\mathbf{C}, \mathbf{D}$ are computability models we say that they are *equivalent* if there exist simulations $\boldsymbol{\gamma}\colon \mathbf{C} \rightarrowtriangle \mathbf{D}$ and $\boldsymbol{\delta}\colon \mathbf{D} \rightarrowtriangle \mathbf{C}$ such that
$$\boldsymbol{\gamma} \circ \boldsymbol{\delta} \sim \mathrm{id}_{\mathbf{D}} \quad \text{and} \quad \boldsymbol{\delta} \circ \boldsymbol{\gamma} \sim \mathrm{id}_{\mathbf{C}}\,.$$
We write $\mathbf{C} \simeq \mathbf{D}$ if $\mathbf{C}$ and $\mathbf{D}$ are equivalent.

One should note that this notion of equivalence is neither weaker nor stronger than the notion of isomorphism of the underlying set, as the following example highlights.

Example 8.3.4. Consider the computability models $\mathbf{B}, \mathbf{C}$ defined over $\mathbb{1}, \mathbb{2}$ respectively where $\mathbf{B}(\mathbb{0}) = \mathbf{C}(\mathbb{0}) = \mathbf{C}(\mathbb{1}) = \emptyset$. The computable functions are simply the nowhere defined partial functions. Then $\mathbf{B}, \mathbf{C}$ are equivalent via the simulations
$$\begin{aligned}
&\boldsymbol{\gamma}\colon \mathbf{B} \rightarrowtriangle \mathbf{C}, && \boldsymbol{\delta}\colon \mathbf{C} \rightarrowtriangle \mathbf{B},\\
&\gamma(\mathbb{0}) = \mathbb{0}, && \delta(\mathbb{0}) = \delta(\mathbb{1}) = \mathbb{0},\\
&\Vdash^{\gamma}_{\mathbb{0}} = \emptyset, && \Vdash^{\delta}_{\mathbb{0}} = \Vdash^{\delta}_{\mathbb{1}} = \emptyset.
\end{aligned}$$

It is immediate that these are well-defined simulations as the two conditions are trivially satisfied (all data types are empty). They also constitute an equivalence: Obviously on underlying class functions we have $\delta \circ \gamma = \mathbf{1}_{\mathbb{1}}$ and $\Vdash^{\delta\circ\gamma}_{\mathbb{0}} = \emptyset$. Thus the nowhere defined partial function in $\mathbf{B}[\mathbb{0}, \mathbb{0}]$ witnesses the transformability $\delta \circ \gamma \preceq \mathrm{id}_{\mathbf{B}}$ as no $a \in \mathbf{B}(\mathbb{0})$ and $a' \in \mathbf{B}(\mathbb{0})$ such that $a' \Vdash^{\delta\circ\gamma}_{\mathbb{0}} a$ exist. For the same reason the nowhere defined partial function witnesses the transformability $\mathrm{id}_{\mathbf{B}} \preceq \delta \circ \gamma$ as well.

We shall now return to the discussion hinted at earlier, whether the requirement that the data types need to be inhabited actually matters. For this we consider the following example.

Example 8.3.5. Let $\mathbf{C}$ be the computability model over $\mathbb{1}$ given by $\mathbf{C}(\mathbb{0}) = \mathbb{1}$ where the only computable function is the identity and $\mathbf{D}$ be the computability model over $\mathbb{2}$ where $\mathbf{D}(\mathbb{0}) = \mathbb{1}, \mathbf{D}(\mathbb{1}) = \mathbb{0}$. Again the only computable functions shall be the identities. We seek to show that $\mathbf{C} \not\simeq \mathbf{D}$. If $\mathbf{C} \simeq \mathbf{D}$, there would be simulations $\boldsymbol{\gamma}\colon \mathbf{C} \rightarrowtriangle \mathbf{D}, \boldsymbol{\delta}\colon \mathbf{D} \rightarrowtriangle \mathbf{C}$ such that $\boldsymbol{\gamma} \circ \boldsymbol{\delta} \sim \mathbf{1}_{\mathbf{D}}, \boldsymbol{\delta} \circ \boldsymbol{\gamma} \sim \mathbf{1}_{\mathbf{C}}$. We can conclude that δ maps both $\mathbb{0}, \mathbb{1}$ to $\mathbb{0}$ and
$$\Vdash^{\delta}_{\mathbb{0}} = \mathbb{1}^2, \quad \Vdash^{\delta}_{\mathbb{1}} = \emptyset.$$

For γ there are a priori two possibilities: either $\gamma(\mathbb{0}) = \mathbb{1}$ or $\gamma(\mathbb{0}) = \mathbb{0}$. However the first case can actually not occur as it would violate the first condition on simulations, as there can be no $x \in \mathbf{D}(\mathbb{1}) = \emptyset$ such that $x \Vdash^{\gamma}_{\mathbb{0}} \mathbb{0}$. Thus only the second case remains. But in this case we can show that $\boldsymbol{\gamma} \circ \boldsymbol{\delta} \not\preceq \mathbf{1}_{\mathbf{D}}$ as this would mean there exists $f \in \mathbf{D}[\gamma(\delta(\mathbb{1})), 1_{\mathbf{C}}(\mathbb{1})] = \mathbf{D}[\mathbb{0}, \mathbb{1}]$ which is by definition not the case. Thus $\mathbf{C} \not\simeq \mathbf{D}$.

This example shows that if one allows data types to be empty, then the notion of equivalence does not allow one to eliminate empty data types that are not connected to other data types via computable functions.

8.3.6 Transformability in other contexts. Coming back to the computability models and simulations due to Cockett, he also has a notion similar to transformability, namely refinements.

Definition 8.3.7 (Refinements). Let $F\colon \mathscr{C} \to \mathscr{R}, G\colon \mathscr{D} \to \mathscr{R}$ be computability models and $(K,\alpha),(L,\beta)\colon F \rightsquigarrow G$ be simulations. Then (L,β) is a *refinement* of (K,α) (written $(K,\alpha) \preceq (L,\beta)$) if for each $c \in \mathscr{C}$ there is a $\lambda_c\colon K(c) \to L(c)$ such that

$$\begin{tikzcd}
F(c) \arrow[r, "\alpha_c"] \arrow[dr, "\beta_c"'] \arrow[dr, phantom, "\preceq", shift left=3] & G\big(K(c)\big) \arrow[d, "G(\lambda_c)"] \\
& G\big(L(c)\big).
\end{tikzcd}$$

8.4 The category of assemblies

Definition 8.4.1 (Category of assemblies - [45]). Let $\mathbf{C}$ be a computability model over the class T. The category $\mathcal{A}\mathit{sm}(\mathbf{C})$ of *assemblies* is defined as follows:

- Its objects are triples $(X, t_X, \Vdash_X)$ where X is a set, $t_X \in T$ and $\Vdash_X \subseteq \mathbf{C}(t_X) \times X$ such that for all $x \in X$ there exists $c \in \mathbf{C}(t_X)$ with $c \Vdash_x x$.
- Its morphisms $f\colon (X, t_X, \Vdash_X) \to (Y, t_Y, \Vdash_Y)$ are maps $f\colon X \to Y$ such that there exists $f' \in \mathbf{C}[t_X, t_Y]$ with the following property:
$$\forall_{x\in X}\forall_{c\in\mathbf{C}(t_X)}\big(x \in \operatorname{dom}(f) \wedge c \Vdash_X x \Rightarrow c \in \operatorname{dom}(f') \wedge f'(c) \Vdash_Y f(x)\big).$$
- Composition and identities are defined as for maps of sets.

This category of assemblies has strong 2-categorical properties discovered by Longley, which we will briefly state. Proofs can be found in [43, Section 4].

Definition 8.4.2 (Subobjects, quotients and copies - [43]). Let $\mathscr{C}$ be a category and $P\colon \mathscr{C} \to \mathsf{Sets}$ be a faithful presheaf. Then $(\mathscr{C}, P)$ has

- *subobjects* if for any $c \in \mathscr{C}$ and any monomorphism $s\colon s \to P(c)$ in Sets there exists a morphism $\tilde{s}\colon c' \to c$ such that $P(\tilde{s}) = s$ and for any morphism $f\colon d \to c$ such that $P(f)$ factors through s there is a unique $g\colon d \to c'$ such that $f = \tilde{s} \circ g$.
- *quotients* if for any object $c \in \mathscr{C}$ and any epi $q\colon P(c) \to q$ in Sets there exists a morphism $\tilde{q}\colon c \to c'$ in $\mathscr{C}$ such that $P(\tilde{q}) = q$ and for any morphism $f\colon c \to d$ such that $P(f)$ factors through q there is a unique $g\colon c' \to d$ such that $f = g \circ \tilde{q}$.
- *copies* if for any $c \in \mathscr{C}$ and $S \in \mathsf{Sets}$ there is an object $c \propto S$ in $\mathscr{C}$ equipped with morphisms $\pi\colon c \propto S \to c, \theta\colon P(c \propto S) \to S$ such that for any $f\colon d \to C$ and $\phi\colon P(d) \to S$ there is a unique $g\colon d \to c \propto S$ such that $f = \pi \circ g$ and $\phi = \theta \circ P(g)$.

We say that $(\mathscr{C}, P)$ is a *quasi-regular category over* Sets if it has subobjects, quotients and copies.

Longley then shows that each category of assemblies $\mathcal{A}\mathit{sm}(\mathbf{C})$ for an arbitrary computability model $\mathbf{C}$ comes equipped with a forgetful functor $\mathtt{Frg}^{\mathbf{C}}\colon \mathcal{A}\mathit{sm}(\mathbf{C}) \to \mathsf{Sets}$ which maps each triple $(X, t_X, \Vdash_X)$ to X and each f to itself, and the pair $(\mathcal{A}\mathit{sm}(\mathbf{C}), \mathtt{Frg}^{\mathbf{C}})$ constitutes a quasi-regular category over Sets.

Further, he defines the 2-category QReg of quasi-regular categories as such:

Definition 8.4.3. The 2-category QReg of quasi-regular categories has as

- as *objects (0-cells)* quasi-regular categories $(\mathscr{C}, P)$ over Sets and
- as *morphisms (1-cells)* $(\mathscr{C}, P_{\mathscr{C}}) \to (\mathscr{D}, P_{\mathscr{D}})$ functors $Q\colon \mathscr{C} \to \mathscr{D}$ together with a natural isomorphism $\iota\colon P_{\mathscr{C}} \to P_{\mathscr{D}} \circ Q$ such that
  - Q preserves subobjects modulo ι, that is if $\tilde{s}\colon c' \to c$ is a subobject lifting of $s\colon S \to P_{\mathscr{C}}(c)$, then $F(\tilde{s})\colon F(c') \to F(c)$ is a subobject lifting of $\iota_c \circ s \circ \iota_{c'}^{-1}$.
  - Q preserves quotients modulo ι, this is defined similar as the above.
  - Q preserves copies modulo ι, that is if $(c \propto S, \pi, \theta)$ is an S-fold copy of x in $\mathscr{C}$, then $\big(F(c \propto S), F(\pi), \theta \circ \iota_{c\propto S}^{-1}\big)$ is an S-fold copy of $F(x)$ in $\mathscr{D}$.
- as *2-cells* $(Q, \iota) \Rightarrow (Q', \iota')$ natural transformations $\alpha\colon Q \Rightarrow Q'$.

He then shows that one obtains a 2-functor $\mathcal{A}sm\colon \mathsf{CompMod} \to \mathsf{QReg}$ which maps

- computability models $\mathbf{C}$ to $\mathcal{A}sm(\mathbf{C})$,
- simulations $\boldsymbol{\gamma}\colon \mathbf{C} \to \mathbf{D}$ to functors $\mathcal{A}sm(\boldsymbol{\gamma})\colon (\mathcal{A}sm(\mathbf{C}), \mathtt{Frg}^{\mathbf{C}}) \to \mathcal{A}sm(\mathbf{D}), \mathtt{Frg}^{\mathbf{D}})$ and
- transformability $\boldsymbol{\gamma} \preceq \boldsymbol{\delta}$ to a natural transformation $\mathcal{A}sm(\boldsymbol{\gamma} \preceq \boldsymbol{\delta})\colon \mathcal{A}sm(\boldsymbol{\gamma}) \to \mathcal{A}sm(\boldsymbol{\delta})$.

The theorem is then the following:

Theorem 8.4.4. *The 2-functor $\mathcal{A}sm\colon$ $\mathsf{CompMod} \to \mathsf{QReg}$ is locally an equivalence, that is $\mathbf{C}, \mathbf{D}$ are equivalent in $\mathsf{CompMod}$ if and only if $\mathcal{A}sm(\mathbf{C}), \mathcal{A}sm(\mathbf{D})$ are equivalent in QReg.*

Chapter 9
The category of computability models and simulations

We now direct our study to the category of computability models itself. This (2-)category is defined as follows:

Definition 9.0.1 (Category of computability models). The category $\mathsf{CompMod}$ has

- *objects* computability models over arbitrary classes as defined in 8.1.1,
- *1-maps* simulations between computability models as defined in 8.2.1,
- *2-maps* transformatbility of simulations as defined in 8.3.1.

The sub-2-categories $\mathsf{CompMod}_{\mathrm{nat}}$ and $\mathsf{CompMod}_{\mathrm{eq}}$ are those where the 1-maps are restricted to natural and equality simulations respectively.

9.1 $\mathsf{CompMod}$ is finitely complete

To show its completeness we use the standard result (e.g. in [6, Theorem 2.8.1]) that the existence of set-indexed products and equalisers suffices to obtain that a category is complete.

Lemma 9.1.1. *Let $\mathbf{C}_i$ be a family of computability models indexed by a set I. Then the product $\prod_{i\in I}\mathbf{C}_i$ is the computability model defined by the following data:*

- *The underlying class is $\prod_{i\in I} T_i$, where the T_i are the underlying classes of the $\mathbf{C}_i$.*
- *The datatypes are given by*

$$\Big(\prod_{i\in I}\mathbf{C}_i\Big)(\tau_i)_{i\in I} = \prod_{i\in I}\big(\mathbf{C}_i(\tau_i)\big).$$

- *The computable functions are given by*

$$\Big(\prod_{i\in I}\mathbf{C}_i\Big)[\tau_i,\sigma_i]_{i\in I} = \prod_{i\in I}\big(\mathbf{C}_i[\tau_i,\sigma_i]\big).$$

The projections $\mathbf{pr}_i\colon \prod_{i\in I}\mathbf{C}_i \mathrel{-\!\!\triangleright} \mathbf{C}_i$ are given by

- *the underlying class function $\mathsf{pr}_i\colon \prod_{i\in I} T_i \to T_i$ defined by $(\tau_i)_{i\in I}\mapsto \tau_i$,*
- *the tracking relations $\Vdash^{\mathsf{pr}_i}_{(\tau_i)_{i\in I}}$, defined via the equivalence*

$$(x_j)_{j\in I} \Vdash^{\mathsf{pr}_i}_{(\tau_i)_{i\in I}} y \textit{ if and only if } x_i = y.$$

Proof: It is immediate that the above definition constitutes a computabiity model and simulations. To see that is is a product in $\mathsf{CompMod}$ we simply observe that if we are given simulations $\boldsymbol{\gamma}_i\colon \mathbf{D} \mathrel{-\!\!\triangleright} \mathbf{C}_i$ for a computability model $\mathbf{D}$ (over some class U) then we can define the simulation $\langle\boldsymbol{\gamma}_i\rangle_{i\in I}\colon \mathbf{D} \mathrel{-\!\!\triangleright} \prod_{i\in I}\mathbf{C}_i$ in the following manner:

- The underlying class function $\langle\gamma_i\rangle_{i\in I}\colon U \to \prod_{i\in I} T_i$ is defined via $u \mapsto \big(\gamma_i(u)\big)$.
- The tracking relations $\Vdash^{\langle\gamma_i\rangle_{i\in I}}_u$ are given through the equivalence

$$y \Vdash^{\langle\gamma_i\rangle_{i\in I}}_u (x_i)_{i\in I} \text{ if and only if for all } i \in I y \Vdash^{\gamma_i}_u x_i.$$

It is straightforward to check that this consitutes a simulation and that $\mathbf{pr}_i \circ \langle \boldsymbol{\gamma}_i \rangle_{i \in I} = \boldsymbol{\gamma}_i$ for all $i \in I$ and that this is the only simulation with this property. Q.E.D.

Proposition 9.1.2. *Given two simulations $\boldsymbol{\gamma}, \boldsymbol{\delta}\colon \mathbf{C} \rightarrowtriangle \mathbf{D}$ $\mathbf{C}, \mathbf{D}$ liver over T, U respectively), the computability model $\mathrm{Eq}(\boldsymbol{\gamma}, \boldsymbol{\delta})$ (defined in the next paragraph) together with the simulation $e(\boldsymbol{\gamma}, \boldsymbol{\delta})\colon \mathrm{Eq}(\boldsymbol{\gamma}, \boldsymbol{\delta}) \rightarrowtriangle \mathbf{D}$ (also defined in the next paragraph) constitute an equalizer.*

Here $\mathrm{Eq}(\boldsymbol{\gamma}, \boldsymbol{\delta})$ is defined through the following data:

- *The underlying class is the equalizer of*

$$\begin{tikzcd} T \arrow[r, shift left, "\gamma"] \arrow[r, shift right, "\delta"'] & U \end{tikzcd}$$

in the category Sets*, that is the subset of T consisting of only those $t \in T$ such that $\gamma(t) = \delta(t)$;*

- *The datatypes are given by*

$$\mathrm{Eq}(\boldsymbol{\gamma}, \boldsymbol{\delta})(t) = \mathbf{C}(t).$$

- *The computable functions are given by*

$$\mathrm{Eq}(\boldsymbol{\gamma}, \boldsymbol{\delta})[t, s] = \mathbf{C}[t, s].$$

The simulation $e(\boldsymbol{\gamma}, \boldsymbol{\delta})$ is defined as follows:

- *The underlying class function maps t to t.*
- *The tracking relations $\Vdash_t^{e(\gamma,\delta)}$ are defined by the equivalence*

$$x \Vdash_t^{e(\gamma,\delta)} y \textit{ if and only if } x = y.$$

Proof: It is straightforward to check that $\mathrm{Eq}(\boldsymbol{\gamma}, \boldsymbol{\delta})$ is a computability model and that $e(\boldsymbol{\gamma}, \boldsymbol{\delta})$ is a simulation. To see that it is also an equaliser assume we are given another simulation $\boldsymbol{\alpha}\colon \mathbf{A} \rightarrowtriangle \mathbf{C}$ (where $\mathbf{A}$ lives over S) such that $\boldsymbol{\gamma} \circ \boldsymbol{\alpha} = \boldsymbol{\delta} \circ \boldsymbol{\alpha}$. This data allows us to define the simulation $\overline{\alpha}\colon \mathbf{A} \rightarrowtriangle \mathrm{Eq}(\boldsymbol{\gamma}, \boldsymbol{\delta})$ via

- $s \mapsto \alpha(s)$ for all $s \in S$,
- $\Vdash_s^{\overline{\alpha}} = \Vdash_s^{\alpha}$ for all $s \in S$.

A simple computation proves that $e(\boldsymbol{\gamma}, \boldsymbol{\delta}) \circ \overline{\alpha} = \alpha$ and this $\overline{\alpha}$ is the only simulation with this property, hence $\mathrm{Eq}(\boldsymbol{\gamma}, \boldsymbol{\delta})$ together with $e(\boldsymbol{\gamma}, \boldsymbol{\delta})$ is an equaliser. Q.E.D.

9.2 CompMod has pushouts

We show that the category CompMod has pushouts.

Proposition 9.2.1. *Let $\mathbf{B}, \mathbf{C}, \mathbf{D}$ be computability models over T, U, V respectively and $\boldsymbol{\gamma}\colon \mathbf{C} \to \mathbf{B}, \boldsymbol{\delta}\colon \mathbf{C} \to \mathbf{D}$ be simulations. Then a pushout $\boldsymbol{\gamma}_*(\mathbf{D})$ is given by the underlying class*

$$\gamma_*(V) = (V \amalg T)/\sim$$

where $v \sim t$ for $v \in V, t \in T$ if there exists a $u \in U$ such that $t = \gamma(u), v = \delta(u)$. For simplicity we write $T' := T \setminus \gamma(U), V' := V \setminus \delta(U)$. Then $\gamma_*(V) = T' \amalg V' \amalg \gamma(U)$. The data types are given by

$$\boldsymbol{\gamma}_*(\mathbf{D})(x) = \begin{cases} \mathbf{B}(x) & \textit{if } x \in T' \\ \mathbf{D}(x) & \textit{if } x \in V', \\ \mathbf{B}(\gamma(u)) \amalg \mathbf{D}(\delta(u)) & \textit{if } x = \gamma(u) \textit{ for } u \in U. \end{cases}$$

The computable functions are defined via

$$\big(\boldsymbol{\gamma}_*(\mathbf{D})\big)[x,y] = \begin{cases} \emptyset & \textit{if } x \in T', y \in V' \textit{ or } y \in T', x \in V', \\ \mathbf{B}[x,y] & \textit{if } x \in T, y \in T' \textit{ or } y \in T, x \in T', \\ \mathbf{D}[x,y] & \textit{if } x \in V, y \in V' \textit{ or } y \in V, x \in V', \\ \mathbf{B}[\gamma(u),\gamma(u')] \amalg \mathbf{D}[\delta(u),\delta(u')] & \textit{if } x = \gamma(u), x' = \gamma(u'). \end{cases} \tag{18}$$

The simulations $\boldsymbol{\gamma}_*\boldsymbol{\delta}, \boldsymbol{\delta}_*\boldsymbol{\gamma}$ are given by

$$\begin{aligned} \boldsymbol{\delta}_*\boldsymbol{\gamma} \colon \mathbf{D} \to \boldsymbol{\gamma}_*(\mathbf{C}),\ & \gamma_*\delta(v) = v \textit{ for all } v \in V, \\ & a \Vdash_v^{\gamma_*\delta} d :\Leftrightarrow a = d, \\ \boldsymbol{\gamma}_*\boldsymbol{\delta} \colon \mathbf{B} \to \boldsymbol{\gamma}_*(\mathbf{C}),\ & \delta_*\gamma(t) = t \textit{ for all } t \in T, \\ & a \Vdash_t^{\delta_*\gamma} d :\Leftrightarrow a = d. \end{aligned}$$

Proof: It is immediate to prove that the above definition constitutes a computability model. For the simulations it is immediate that the rules given are indeed well-defined and it remains to check the two properties of simulations. We only check these properties for $\boldsymbol{\delta}_*\boldsymbol{\gamma}$, the other case is analogous.

1. Let $v \in V$ and $d \in \mathbf{D}(v)$ be given. Then $v \in U' \amalg V'$, and in both cases $d \in \boldsymbol{\gamma}_*(\mathbf{D})(v)$ as $\mathbf{D}(v) \subseteq \boldsymbol{\gamma}_*(\mathbf{D})(v)$ in both cases. So $d \Vdash_v^{\delta_*\gamma} d$.
2. Let $v_1, v_2 \in V$ and $f \in \mathbf{D}[v_1, v_2]$ be given. Thus we know one of the last two cases of (18) arises, and in both cases $\mathbf{D}[x,y] \subseteq \big(\boldsymbol{\gamma}_*(\mathbf{D})\big)[x,y]$. We show that $f \Vdash_{(v_1,v_2)}^{\delta_*\gamma} f$. If $d \in \mathrm{dom}(f)$ and $d' \Vdash_{v_1}^{\delta_*\gamma} d$, then $d' = d$, so $d' \in \mathrm{dom}(f)$ and thus $f(d') = f(d) \Vdash_{v_2}^{\delta_*\gamma} f(d)$.

It remains to check the universal property. For this let another computability model $\mathbf{E}$ over W with simulations $\boldsymbol{\epsilon} \colon \mathbf{B} \to \mathbf{E}, \boldsymbol{\iota} \colon \mathbf{D} \to \mathbf{E}$ be given such that the obvious square commutes. Then the unique $[\boldsymbol{\epsilon}, \boldsymbol{\iota}] \colon \boldsymbol{\gamma}_*(\mathbf{D}) \to \mathbf{E}$ making the usual diagram commute is given by the underlying class function $\gamma_*(V) \to W$ that embeds $\gamma_*(V)$ as the union $\epsilon(V) \cup \iota(T)$. The tracking relations $\Vdash_x^{[\epsilon,\iota]}$ are given by

$$\Vdash_x^{[\epsilon,\iota]} := \begin{cases} \Vdash_x^{\epsilon} & \text{if } x \in T', \\ \Vdash_x^{\iota} & \text{if } x \in V', \\ \Vdash_{\gamma(u)}^{\epsilon} \amalg \Vdash_{\delta(u)}^{\iota} & \text{if } u \in U. \end{cases}$$

It is immediate to prove that this is well-defined, and the two conditions of a simulation hold because

1. if $x \in \gamma_*(V)$ we can distinguish the following three cases:
   1.1 if $x \in T'$ then $\boldsymbol{\gamma}_*(\mathbf{D})(x) = \mathbf{D}(x)$ and for every $d \in \boldsymbol{\gamma}_*(\mathbf{D})(x)$ we find $e \in \mathbf{E}(\epsilon(x))$ such that $e \Vdash_x^{\epsilon} d$ and thus by definition $e \Vdash_x^{[\epsilon,\iota]} d$.

   1.2 If $x \in V'$ then a similar argument yields the desired $e \in \mathbf{E}(\iota(x))$ such that $e \Vdash_x^{[\epsilon,\iota]} d$.

   1.3 If $x = \gamma(u)$ then every $d \in \boldsymbol{\gamma}_*(\mathbf{D})(x)$ is either in $\mathbf{B}(\gamma(u))$ or $\mathbf{D}(\delta(u))$. In both cases we can proceed as above to obtain the desired $e \in \mathbf{E}(\iota(x))$.
2. If $f \in (\boldsymbol{\gamma}_*(\mathbf{D}))[x,y]$ then we need to distinguish four cases:
   2.1 if $x \in T', y \in V'$ or $y \in T', x \in V'$, then there are no computable functions to consider.

   2.2 If $x \in T, y \in T'$ or $y \in T, x \in T'$, then the computable functions are those in $\mathbf{B}[x,y]$. It is immediate from the definition of the tracking relations that $g \Vdash_{(x,y)}^{[\epsilon,\iota]} f$ for the $g \in \mathbf{E}[\epsilon(x), \epsilon(y)]$ such that $g \Vdash_{(x,y)}^{\epsilon} f$.

2.3 If $x \in V, y \in V'$ or $y \in V, x \in V'$, then the computable functions are those in $\mathbf{D}[x,y]$. It is immediate from the definition of the tracking relations that $g \Vdash^{[\epsilon,\iota]}_{(x,y)} f$ for the $g \in \mathbf{E}[\iota(x),\iota(y)]$ such that $g \Vdash^{\iota}_{(x,y)} f$.

2.4 In the last case we can similar to above reduce to case 2.2 or 2.3.

At last we show that $[\boldsymbol{\epsilon},\boldsymbol{\iota}]$ is unique. But this is immediate from the fact that $[\boldsymbol{\epsilon},\boldsymbol{\iota}]$ makes the diagram

$$\begin{tikzcd}
\mathbf{C} \arrow[r, "\boldsymbol{\delta}"] \arrow[d, "\boldsymbol{\gamma}"] & \mathbf{D} \arrow[d, "\boldsymbol{\delta}_*\boldsymbol{\gamma}"] \arrow[rdd, "\boldsymbol{\iota}"] & \\
\mathbf{B} \arrow[r, "\boldsymbol{\gamma}_*\boldsymbol{\delta}"] \arrow[rrd, "\boldsymbol{\epsilon}"'] & \boldsymbol{\gamma}_*(\mathbf{D}) \arrow[rd, dotted, "{[\boldsymbol{\epsilon},\boldsymbol{\iota}]}"] & \\
 & & \mathbf{E}
\end{tikzcd}$$

commute. This ensure that $[\boldsymbol{\epsilon},\boldsymbol{\iota}]$ is unique on the level of underlying class functions, and for the tracking relations one notes that $\Vdash^{\epsilon}_{t} = \Vdash^{[\epsilon,\iota]\circ\gamma_*\delta}_{t}$ and as $\Vdash^{[\epsilon,\iota]}_{t}$ is nothing but equality this yields the desired uniqueness for tracking relations (the case $\Vdash^{\iota}_{t}$ is analogous). Q.E.D.

We will later need the explicit description of the coequalisers in CompMod, so we will state how they are obtained from a parallel pair. The universal property can be checked as in the case of pushouts and is straightforward, so we omit the proof.

Proposition 9.2.2. *Given a parallel pair* $\mathbf{C} \overset{\boldsymbol{\gamma}_1}{\underset{\boldsymbol{\gamma}_2}{\rightrightarrows}} \mathbf{D}$, *where* $\mathbf{C}$ *lives over the class* T *and* $\mathbf{D}$ *lives over* U*, a coequaliser is given be the computability model* $\mathrm{Coeq}(\mathbf{C},\mathbf{D})$ *defined via the following data:*

- *The underlying class is* $T/\sim$ *defined through the coequaliser diagram*

$$T \overset{\gamma_1}{\underset{\gamma_2}{\rightrightarrows}} U \twoheadrightarrow T/\sim \tag{19}$$

  in Sets*.*
- *The datatypes are given by* $\mathrm{Coeq}(\mathbf{C},\mathbf{D})(t) = \coprod_{t'\sim t} \mathbf{C}(t')$*.*
- *The partial functions are given as coproducts of partial functions, that is*

$$\mathrm{Coeq}(\mathbf{C},\mathbf{D})[t_1,t_2] = \coprod_{\substack{t'_1\sim t_1\\ t'_2\sim t_2}} \mathbf{D}[t'_1,t'_2].$$

The simulation $\mathrm{Coeq}(\boldsymbol{\gamma}_1,\boldsymbol{\gamma}_2)\colon \mathbf{D} \to \mathrm{Coeq}(\mathbf{C},\mathbf{D})$ *is defined by the following data:*

- *The underlying class function is simply the projection arising from the coequaliser diagram (19).*
- *The tracking relations* $\Vdash^{\mathrm{Coeq}(\gamma_1,\gamma_2)}_{t}$ *are defined via*

$$c' \Vdash^{\mathrm{Coeq}(\gamma_1,\gamma_2)}_{t} c \Leftrightarrow c = c'.$$

9.3 CompMod is not regular

To show that CompMod is a regular category we have to show three things:

- CompMod is complete.

- In CompMod if we consider pullback diagrams of the form

$$\begin{tikzcd}
\boldsymbol{\gamma}^{-1}(\mathbf{C}) \arrow[r, "\boldsymbol{\rho}_1"] \arrow[d, "\boldsymbol{\rho}_2"] & \mathbf{C} \arrow[d, "\boldsymbol{\gamma}"] \\
\mathbf{C} \arrow[r, "\boldsymbol{\gamma}"] & \mathbf{D}
\end{tikzcd}$$

  then we obtain a coequaliser $\boldsymbol{\gamma}^{-1}(\mathbf{C}) \rightrightarrows \mathbf{C} \longrightarrow \mathrm{Coeq}(\boldsymbol{\rho}_1, \boldsymbol{\rho}_2)$.

- In CompMod the pullback of a regular epimorphism is again a regular epimorphism.

The first two are immediate because as we have shown CompMod is also cocomplete. For the last property we first examine the (regular) epimorphisms in CompMod.

Lemma 9.3.1. *If $\boldsymbol{\gamma}\colon \mathbf{C} \to \mathbf{D}$ is a simulation, then the following are equivalent:*

1. *The underlying class function $\boldsymbol{\gamma}$ is surjective and all tracking relations $\Vdash^{\gamma}_{t}$ fulfil the following:*
   - *For every $y \in \mathbf{D}\big(\gamma(t)\big)$ there exists a $x \in \mathbf{C}(t)$ such that $y \Vdash^{\gamma}_{t} x$ and if $y' \Vdash^{\gamma}_{t} x$, then $y = y'$.*
2. *The simulation $\boldsymbol{\gamma}$ is an epimorphism in* CompMod*.*

Proof: We first show that if the condition hold, then $\boldsymbol{\gamma}$ is an epimorphism. For this let two simulations $\boldsymbol{\delta}_1, \boldsymbol{\delta}_2\colon \mathbf{D} \to \mathbf{E}$ such that $\boldsymbol{\delta}_1 \circ \boldsymbol{\gamma} = \boldsymbol{\delta}_2 \circ \boldsymbol{\gamma}$. We have to show that $\boldsymbol{\delta}_1 = \boldsymbol{\delta}_2$. On the level of underlying class functions this is immediate. For the tracking relations suppose that we are given $u \in U$ and $x \in \mathbf{D}(t), y \in \mathbf{E}\big(\delta_1(u)\big)$ such that $y \Vdash^{\delta_1}_{u} x$. By assumption we know that there exists $t \in T$ such that $\gamma(t) = u$, additionally we find $z \in \mathbf{C}(t)$ such that $x \Vdash^{\gamma}_{t} z$ and if $x' \Vdash^{\gamma}_{t} z$, then $x = x'$. This yields that $y \Vdash^{\delta_1 \circ \gamma}_{t} z$, and thus $y \Vdash^{\delta_2 \circ \gamma}_{t} z$ by assumption. So we have a $x' \in \mathbf{D}(u)$ such that $y \Vdash^{\delta_2}_{u} x'$ and $x' \Vdash^{\gamma}_{t} z$. But then by assumption $x = x'$ and thus $y \Vdash^{\delta_2}_{u} x$ as desired.

The same argument with δ_2 and δ_1 swapped shows the reverse direction. Thus $\Vdash^{\delta_1}_{u} = \Vdash^{\delta_2}_{u}$ for all $u \in U$ and by extension $\boldsymbol{\delta}_2 = \boldsymbol{\delta}_1$.

Conversely assume that $\boldsymbol{\gamma}$ is an epimorphism. Then γ must be clearly surjective. To show the condition for $\Vdash^{\gamma}_{t}$, assume that there was a $y \in \mathbf{D}\big(\gamma(t)\big)$ such that if $y \Vdash^{\gamma}_{t} x$, then there exists $y' \neq y$ such that $y' \Vdash^{\gamma}_{t} x$. We can then define simulations $\boldsymbol{\delta}_1, \boldsymbol{\delta}_2\colon \mathbf{D} \to \mathbf{E}$, where $\mathbf{E}$ lives over the set $\mathbb{1}$ and $\mathbf{E}(\emptyset) = \mathbb{2}$, as follows: the underlying class function sends every $u \in U$ to $\emptyset$, and the tracking relations are defined as such:

$$\Vdash^{\delta_1}_{u} := \begin{cases} \mathbf{E}(\emptyset) \times \mathbf{D}(u) & \text{if } u \neq \gamma(t), \\ \big(\mathbf{E}(\emptyset) \times \mathbf{D}(\gamma(t)) \setminus \{y\}\big) \cup \big(\{\mathbb{1}\} \times \{y\}\big) & \text{else,} \end{cases}$$

$$\Vdash^{\delta_2}_{u} := \begin{cases} \mathbf{E}(\emptyset) \times \mathbf{D}(u) & \text{if } u \neq \gamma(t), \\ \mathbf{E}(\emptyset) \times \mathbf{D}(\gamma(t)) & \text{else.} \end{cases}$$

Then one can compute that $\Vdash^{\delta_1 \circ \gamma}_{t} = \Vdash^{\delta_2 \circ \gamma}_{t} = \mathbf{E}(\emptyset) \times \mathbf{C}(t)$ for all t where we use that if $y \Vdash^{\gamma}_{t} x$, then also $y' \Vdash^{\gamma}_{t} x$ for some $y \neq y'$ and thus $\emptyset \Vdash^{\delta_1}_{u} y'$ and by extension $\emptyset \Vdash^{\delta_1 \circ \gamma}_{u} x$. But clearly $\boldsymbol{\delta}_1 \neq \boldsymbol{\delta}_2$, in violation of our assumption that $\boldsymbol{\gamma}$ is epi. Q.E.D.

Theorem 9.3.2. *The category* CompMod *is not regular.*

Proof: To prove this we adapt the counterexample in the category of partial orders. So let $\mathbf{A}, \mathbf{B}, \mathbf{C}$ be the computability models given as such:

- $\mathbf{A}$ lives over $\mathbb{4}$ where all sets of data types are simply $\mathbb{1}$, and the computable functions are given by
$$\mathbf{A}[\mathbb{0}, \mathbb{1}] = \{\mathbf{1}_{\mathbb{1}}\} = \mathbf{A}[\mathbb{2}, \mathbb{3}]$$
together with the identities.
- $\mathbf{B}$ lives over $\mathbb{3}$ where again all sets of data types are simply $\mathbb{1}$, the computable functions are given by
$$\mathbf{B}[\mathbb{0}, \mathbb{1}] = \mathbf{B}[\mathbb{0}, \mathbb{2}] = \mathbf{B}[\mathbb{1}, \mathbb{2}] = \{\mathbf{1}_{\mathbb{1}}\}$$
and the identities.
- $\mathbf{C}$ lives over $\mathbb{2}$, all sets of data types are $\mathbb{1}$ and the computable functions are given by $\mathbf{B}[\mathbb{0}, \mathbb{1}] = \{\mathbf{1}_{\mathbb{1}}\}$ and the identities.

We then define the simulations $\boldsymbol{\gamma} \colon \mathbf{A} \to \mathbf{B}, \boldsymbol{\delta} \colon \mathbf{C} \to \mathbf{B}$ through

$$\begin{aligned}
&\gamma(\mathbb{0}) = \mathbb{0}, \gamma(\mathbb{1}) = \gamma(\mathbb{2}) = \mathbb{1}, \gamma(\mathbb{3}) = \mathbb{2}, \\
&\delta(\mathbb{0}) = \mathbb{0}, \delta(\mathbb{1}) = \mathbb{1}.
\end{aligned}$$

The tracking relations are always $\mathbb{1} \times \mathbb{1}$. It is immediate that these are well-defined simulations and from the preceding lemma we know that $\boldsymbol{\gamma}$ is an epimorphism.

If we calculate the pullback we obtain the model $\boldsymbol{\gamma}^{-1}(\mathbf{C})$ which lives over the set $\{(\mathbb{0}, \mathbb{0}), (\mathbb{3}, \mathbb{1})\}$, which we will tacitly identity with $\mathbb{2}$ and whose sets of data types are $\mathbb{1}$. The only computable functions are the identities. The pulled back simulations are $\boldsymbol{\gamma}^{-1}\boldsymbol{\delta}, \boldsymbol{\delta}^{-1}\boldsymbol{\gamma}$, here the first one maps $\mathbb{0}$ to $\mathbb{0}$ and $\mathbb{1}$ to $\mathbb{3}$, the second maps $\mathbb{0}$ to $\mathbb{0}$ and $\mathbb{1}$ to $\mathbb{1}$. The tracking relations are again $\mathbb{1} \times \mathbb{1}$.

We show that $\boldsymbol{\delta}^{-1}\boldsymbol{\gamma}$ is not the coequaliser of its kernel pair which is equivalent to it arising from a coequaliser [44, p. 41]. We show that

$$\begin{tikzcd}
\boldsymbol{\gamma}^{-1}(\mathbf{C}) \arrow[r, "\mathrm{id}"] \arrow[d, "\mathrm{id}"] & \boldsymbol{\gamma}^{-1}(\mathbf{C}) \arrow[d, "\boldsymbol{\delta}^{-1}\boldsymbol{\gamma}"] \\
\boldsymbol{\gamma}^{-1}(\mathbf{C}) \arrow[r, "\boldsymbol{\delta}^{-1}\boldsymbol{\gamma}"] & \mathbf{C}
\end{tikzcd}$$

is a pullback square. It is obviously commutative and thus it remains to check the universal property. For this let $\mathbf{E}$ over V with $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \colon \mathbf{E} \to \boldsymbol{\gamma}^{-1}(\mathbf{C})$ with $\boldsymbol{\delta}^{-1}\boldsymbol{\gamma} \circ \boldsymbol{\alpha}_1 = \boldsymbol{\delta}^{-1}\boldsymbol{\gamma} \circ \boldsymbol{\alpha}_2$. Thus on the underlying sets we already have that $\alpha_1 = \alpha_2$ as $\delta^{-1}\gamma$ is a monomorphism. The tracking relations have to be $\mathbb{1} \times \mathbf{E}(w)$ for all $w \in W$ due to the first condition on simulations, so we obtain $\boldsymbol{\alpha}_1 = \boldsymbol{\alpha}_2$ and thus $\boldsymbol{\alpha}_1$ is the desired unique simulation making the usual triangles commute. But obviously $\boldsymbol{\delta}^{-1}\boldsymbol{\gamma}$ is not the equaliser of the identities, for this would entail that if we consider the simulation $\boldsymbol{\sigma} \colon \boldsymbol{\gamma}^{-1}(\mathbf{C}) \to \boldsymbol{\gamma}^{-1}(\mathbf{C})$ that swaps $\mathbb{0}$ and $\mathbb{1}$ there exists a simulation $\boldsymbol{\iota}\mathbf{C} \to \boldsymbol{\gamma}^{-1}(\mathbf{C})$ making

$$\begin{tikzcd}
\boldsymbol{\gamma}^{-1}(\mathbf{C}) \arrow[r, "\boldsymbol{\delta}^{-1}\boldsymbol{\gamma}"] \arrow[dr, "\boldsymbol{\sigma}"'] & \mathbf{C} \arrow[d, "\boldsymbol{\iota}"] \\
 & \mathbf{C}
\end{tikzcd}$$

commute. But then we have $\iota(\mathbb{1}) = \mathbb{0}, \iota(\mathbb{0}) = \mathbb{1}$, but this can not be as no function in $\mathbf{C}[\mathbb{1}, \mathbb{0}]$ exists which could track $\mathbf{1}_{\mathbb{1}} \in \mathbf{C}[\mathbb{0}, \mathbb{1}]$. Q.E.D.

Corollary 9.3.3. CompMod *is not a topos.*

Proof: As every topos is regular [37, p. 92] and CompMod is not regular it can not be a topos. Q.E.D.

9.4 The Grothendieck computability model

The Grothendieck computability model is the computabiltiy model counterpart to the category of elements, a special case of the general categorical Grothendieck construction. A category $\mathscr{C}$ is replaced by a computability model $\mathbf{C}$, and a (covariant) presheaf $S \colon \mathscr{C} \to \mathsf{Sets}$ by a (covariant) simulation $\boldsymbol{\gamma} \colon \mathbf{C} \rightharpoonup \mathbf{Sets}$. Moreover, the first-projection functor is replaced by the first-projection-simulation.

Proposition 9.4.1. *Let* $\mathbf{C}$ *be a computability model over the class* T *together with a simulation* $\boldsymbol{\gamma} \colon \mathbf{C} \rightharpoonup \mathbf{Sets}$*. 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 $\left(\sum_{\mathbf{C}} \boldsymbol{\gamma}\right)(s, a)$ *to* $\left(\sum_{\mathbf{C}} \boldsymbol{\gamma}\right)(t, b)$ *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*

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

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

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 $\left(\sum_{\mathbf{C}} \boldsymbol{\gamma}\right)(t, b)$ 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 $x \in \sum_{\mathbf{C}} \boldsymbol{\gamma}(t, b)$, then only $x \Vdash_{(t,b)}^{\mathsf{pr}_1} x$, and if f is a computable function from $\sum_{\mathbf{C}} \boldsymbol{\gamma}(s, a)$ to $\sum_{\mathbf{C}} \boldsymbol{\gamma}(t, b)$, then $f \Vdash_{((s,a),(t,b))}^{\mathsf{pr}_1} f$. Q.E.D.

The following fact is straightforward to prove.

Proposition 9.4.2. *Let* $\mathscr{C}$ *be a category and* $S \colon \mathscr{C} \to \mathsf{Sets}$ *a pullback-preserving presheaf on* $\mathscr{C}$*. Let* $\gamma^S \colon \mathcal{C}_0 \rightharpoonup \mathbf{Sets}$ *be defined via* $\gamma^S(c) = S(c)$ *and the relations* $\Vdash_c^{\gamma^S}$ *are simply the diagonal, and let* $\{\mathsf{pr}_2\} \colon \sum_{\mathscr{C}} S \to \mathsf{Sets}$ *be defined by* $\{\mathsf{pr}_2\}(c, x) := \{x\}$ *and if* $f \colon (c, x) \to (d, y)$ *in* $\sum_{\mathscr{C}} S$*, let* $[S(f)](x) := y$*. Then*

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

Proof: *Canonical model → Grothendieck model:* We want to compute the Grothendieck construction for the canonical computability model. For this we first need a simulation

$$\mathbf{CM}^{\mathrm{prt}}(\mathcal{C}; S) \multimap \mathbf{Sets}\,.$$

Here we simply take the simulation obtained from the presheaf S (called $\boldsymbol{\gamma}^S$) that is $\gamma^S\colon \mathcal{C}_0 \to \mathsf{Sets}$ is defined via $\gamma^S(c) = S(c)$ and the relations $\Vdash^{\gamma^S}_c$ are simply the diagonal. Thus we can compute the Grothendieck computability model[1]

$$\sum_{\mathbf{CM}^{\mathrm{prt}}(\mathcal{C};S)} \boldsymbol{\gamma}^S$$

to live over the class $\{(c,u) \mid c \in \mathcal{C}_0, u \in \gamma^S(c) = S(c)\}$. In order to see that $\Sigma(c,u)$ looks like we can immediately compute that

$$\Sigma(c,u) = \{x \in S(u) \mid u \Vdash x\}.$$

But by definition we know that $u \Vdash x$ if and only if $x = u$. Thus we have that $\Sigma(c,u)$ has as only element u itself. We compute the functions $\Sigma\big[(c,u),(d,v)\big]$. By definition functions in this set are partial functions $S(i,f)\colon S(c) \to S(d)$ such that $S(i,f)(u) = v$. Here $S(i,f)$ is identified with the partial arrow $S(c) \to S(d)$ that has its domain restricted to $\mathrm{dom}(i)$.
Grothendieck category → canonical model: On the converse, if we fist compute the Grothendieck category associated to the presheaf and then the canonical model, we obtain the following: The functor $\Sigma(\mathcal{C}, S) \to \mathsf{Sets}$ we use is not the composition of the functor S with the projection to $\mathcal{C}$, but the functor $S \circ \mathsf{pr}_1$ is not of use as we would then have on objects that

$$\mathbf{CM}^{\mathrm{prt}}\Big(\sum_{\mathcal{C}} S; S \circ \mathsf{pr}_1\Big)(c,u) = S(c) \neq \{u\}$$

as desired. So we have to show that the second projection $\{\mathsf{pr}_2\}$ that sends each (c,u) to $\{u\}$ is a functor. As well-definedness on objects is immediate it remains to show that it is well-defined on arrows, that is each arrow $f\colon (c,u) \to (c',u')$ maps to an arrow $g\colon \{u\} \to \{u'\}$. But this is again immediate as for each such arrow we have that $S(f)(u) = u'$, so we can choose g to be $S(f)$. The functoriality is now immediate. So we seek to show that $\mathbf{CM}^{\mathrm{prt}}(\sum_{\mathcal{C}} S; \{\mathsf{pr}_2\})$ is the same computability model as $\sum_{\mathbf{CM}^{\mathrm{prt}}(\mathcal{C},S)} \boldsymbol{\gamma}^S$. But this is now immediate from the definition. Q.E.D.

Remark 9.4.3. The functor $\mathbf{C} \mapsto \mathcal{A}sm(\mathbf{C})$, studied in [45], does not "preserve" the Grothendieck construction. Namely, if $\mathbf{1}$ is the 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 $\boldsymbol{\iota}_{\mathbf{1}}\colon \mathbf{1} \multimap \mathbf{Sets}$, and show that

$$\mathcal{A}sm\Big(\sum_{\mathbf{1}} \boldsymbol{\iota}_{\mathbf{1}}\Big) \neq \sum_{\mathcal{A}sm\,(\mathbf{1})} \mathcal{A}sm(\boldsymbol{\iota}_{\mathbf{1}}).$$

Next we show that the category of computability models $\mathsf{CompMod}$ is a type-category. First, we lift a simulation $\boldsymbol{\gamma}\colon \mathbf{C} \multimap \mathbf{D}$ to a simulation between the Grothendieck computability models $\sum_{\mathbf{C}}(\boldsymbol{\delta} \circ \boldsymbol{\gamma})$ and $\sum_{\mathbf{D}} \boldsymbol{\delta}$.

[1] We will write $\Sigma(c,u)$ instead of $\big(\sum_{\mathbf{CM}^{\mathrm{prt}}(\mathcal{C};S)} \boldsymbol{\gamma}^S)\big)(c,u)$ to make the notation a little more readable.

Lemma 9.4.4. *Let* $\mathbf{C}, \mathbf{D}$ *be computability models over the classes* T, U *respectively, and* $\boldsymbol{\gamma}\colon \mathbf{C} \rightarrow \mathbf{D}, \boldsymbol{\delta}\colon \mathbf{D} \rightarrow \mathbf{Sets}$ *simulations. There is a simulation* $\sum_{\boldsymbol{\delta}} \boldsymbol{\gamma}\colon \sum_{\mathbf{C}}(\boldsymbol{\delta} \circ \boldsymbol{\gamma}) \rightarrow \sum_{\mathbf{D}} \boldsymbol{\delta}$*, such that the following is a pullback square*

$$\begin{tikzcd}
\sum_{\mathbf{C}}(\boldsymbol{\delta} \circ \boldsymbol{\gamma}) \arrow[r, "\sum_{\delta} \gamma"] \arrow[d, "\mathbf{pr}_1"] & \sum_{\mathbf{D}} \boldsymbol{\delta} \arrow[d, "\mathbf{pr}_1"] \\
\mathbf{C} \arrow[r, "\gamma"] & \mathbf{D}.
\end{tikzcd}$$

Proof: To define $\sum_{\boldsymbol{\delta}} \boldsymbol{\gamma}$, let the underlying class-function $\sum_{\delta} \gamma\colon \sum_{t \in T} \boldsymbol{\gamma}(t) \rightarrow \sum_{u \in U} \boldsymbol{\delta}(u)$ be defined by the rule $(t, b) \mapsto \big(\gamma(t), b\big)$. The corresponding forcing relations are defined by $x' \Vdash^{\sum_{\delta} \gamma}_{(t,b)} x :\Leftrightarrow x' \Vdash^{\gamma}_{t} x$. It is straightforward to show that $\sum_{\boldsymbol{\delta}} \boldsymbol{\gamma}$ is a simulation. 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

$$\begin{tikzcd}
\mathbf{E} \arrow[r, "\beta"] \arrow[d, "\alpha"] & \sum_{\mathbf{D}} \boldsymbol{\delta} \arrow[d, "\mathbf{pr}_1"] \\
\mathbf{C} \arrow[r, "\gamma"] & \mathbf{D}.
\end{tikzcd} \tag{20}$$

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

$$\begin{tikzcd}
\mathbf{E} \arrow[rrd, bend left, "\beta"] \arrow[rd, "\zeta"] \arrow[rdd, bend right, "\alpha"'] & & \\
& \sum_{\mathbf{C}}(\boldsymbol{\delta} \circ \boldsymbol{\gamma}) \arrow[r, "\sum_{\delta} \gamma"] \arrow[d, "\mathbf{pr}_1"] & \sum_{\mathbf{D}} \boldsymbol{\delta} \arrow[d, "\mathbf{pr}_1"] \\
& \mathbf{C} \arrow[r, "\gamma"] & \mathbf{D}
\end{tikzcd}$$

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 (20) 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 $\boldsymbol{\beta}, \boldsymbol{\alpha}$. Q.E.D.

Lemma 9.4.5. *If* $\mathbf{C}, \mathbf{D}, \mathbf{E} \in \mathsf{CompMod}$, $\boldsymbol{\gamma} \colon \mathbf{C} \rightarrowtriangle \mathbf{D}, \boldsymbol{\delta} \colon \mathbf{D} \rightarrowtriangle \mathbf{E}$, *and* $\boldsymbol{\epsilon} \colon \mathbf{E} \rightarrowtriangle \mathbf{Sets}$ *is a presheaf-simulation, then the following strictness conditions hold:*

1. $\sum_{\boldsymbol{\epsilon}} \mathbf{1}_{\mathbf{E}} = \mathbf{1}_{\sum_{\mathbf{E}} \boldsymbol{\epsilon}}$.

2. $\sum_{\boldsymbol{\epsilon}} (\boldsymbol{\delta} \circ \boldsymbol{\gamma}) = \sum_{\boldsymbol{\epsilon}} \boldsymbol{\delta} \circ \sum_{(\boldsymbol{\epsilon}\circ\boldsymbol{\delta})} \boldsymbol{\gamma}$.

Proof:

1. 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.

2. 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 rem that $x \Vdash^{\sum_\epsilon \delta\circ\gamma}_{(t,b)} y$ if and only if $x \Vdash^{\delta\circ\gamma}_{t} y$. Similarly, we have that $x \Vdash^{\sum_\epsilon \delta}_{(t,b)} y$ if and only if $x \Vdash^{\delta}_{t} y$, and $x \Vdash^{\sum_{\epsilon\circ\delta} \gamma}_{(t,b)} y$ if and only if $x \Vdash^{\gamma}_{t} y$. Hence, $x \Vdash^{\sum_\epsilon \delta \circ \sum_{\epsilon\circ\delta} \gamma}_{(t,b)} y$ if and only if $x \Vdash^{\delta\circ\gamma}_{t} z$, which by the above is equivalent to $x \Vdash^{\sum_\epsilon \delta\circ\gamma}_{(t,b)} z$. Q.E.D.

Theorem 9.4.6. *The category* $\mathsf{CompMod}$ *is a type-category.*

Proof: This follows immediately from Lemma 9.4.4, Lemma 9.4.5, and the fact that $\mathsf{CompMod}$ has a terminal object, as explained in Remark 9.4.3. Q.E.D.

9.4.7 A generalised Grothendieck construction.

We can generalise this notion of the Grothendieck construction to arbitrary simulations $\mathbf{C} \rightarrowtriangle \mathbf{D}$ by observing the fact that every computability model allows a realising simulation into the model **Sets** defined as follows:

Proposition 9.4.8. *Let* $\mathbf{C}$ *be a computability model over the class* T*. We then obtain a simulation* $r^{\mathbf{C}} \colon \mathbf{C} \rightarrowtriangle \mathbf{Sets}$ *defined by the following data:*

- *the underlying class function is defined by* $r^{\mathbf{C}}(t) = \mathbf{C}(t)$ *for all* $t \in T$,
- *the tracking relations are given as*

$$y \Vdash^{r^{\mathbf{C}}}_{t} x \textit{ if and only if } y = x.$$

Proof: Immediate as the first condition is satisfied as each x is tracked by itself and the second condition is immediate as each function f is tracked by itself. Q.E.D.

This allows us to make the following definition (with a small abuse of notation):

Definition 9.4.9. If $\boldsymbol{\gamma}\colon \mathbf{C} \rightarrowtriangle \mathbf{D}$ is a simulation define

$$\sum_{\mathbf{C}} \boldsymbol{\gamma} := \sum_{\mathbf{C}} r^{\mathbf{D}} \circ \boldsymbol{\gamma}.$$

It is immediate that this extends the notion we defined in Proposition 9.4.1 as it is immediate that $r^{\mathbf{Sets}} = \mathrm{id}_{\mathbf{Sets}}$.

9.4.10 The Grothendieck model and presheaf simulations.

We show an analogous result to [37, Proposition 1.1.7]. Our goal is to show for any presheaf simulation $\boldsymbol{\gamma}\colon \mathbf{C} \to \mathsf{Sets}$ that we obtain an equivalence

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

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

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

Before we do this we make a few following observations:

Remarks:

1. For both categories the morphism structure is thin and thus a preorder, thus we only need to define our functor on morphisms and show this functor preserves this preorder.
2. More explicitly the objects of $[\mathbf{C}, \mathsf{Sets}]/\boldsymbol{\gamma}$ themselves are simply simulations $\boldsymbol{\delta}\colon \mathbf{C} \to \mathsf{Sets}$ such that $\boldsymbol{\delta} \preceq \boldsymbol{\gamma}$.
3. For all simulations $\boldsymbol{\gamma}\colon \mathbf{C} \to \mathsf{Sets}$ 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).

So we define our functor and then prove the desired properties:

Proposition 9.4.11. *Let* $\mathbf{C}$ *over* T *be a computability model and* $\boldsymbol{\gamma}\colon \mathbf{C} \to \mathsf{Sets}$ *be a simulation. We have a functor* $I\colon \big[\sum_{\mathbf{C}} \boldsymbol{\gamma}, \mathsf{Sets}\big] \to [\mathbf{C}, \mathsf{Sets}]/\boldsymbol{\gamma}$ *defined through the following rule: for* $\boldsymbol{\beta}\colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \rightarrowtriangle \mathsf{Sets}$ *we define* $I(\boldsymbol{\beta})$ *through the underlying class function* $I(\beta)\colon T \to \mathsf{Sets}$ *defined as*

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

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

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

This functor fulfils the following three properties:

1. *I is full.*
2. *I is faithful.*
3. *I is essentially surjective.*

Proof: We begin by showing that I is well-defined, that is that for every $\boldsymbol{\beta} \in \left[\sum_{\mathbf{C}} \boldsymbol{\gamma}, \mathsf{Sets}\right]$ the rule $I(\boldsymbol{\beta})$ actually constitutes a simulation and $I(\boldsymbol{\beta}) \preceq \boldsymbol{\gamma}$. First we observe that $I(\beta)(t)$ is well-defined and a set for all $t \in T$. To check the first property of simulations we observe that for all $t \in T$ we have that at least one $x \in \gamma(t)$ exists, so $\bigcup_{x \in \gamma(t)} \beta(t, x)$ is non-empty. Furthermore if we are given $b \in \mathbf{C}(t)$ we know as $\boldsymbol{\gamma}$ is a simulation we find $x \in \gamma(t)$ such that $x \Vdash_t^{\gamma} b$. Now as $\boldsymbol{\beta}$ is a simulation we find $a \in \beta(t, x)$ such that $a \Vdash_{(t,x)}^{\beta} b$. By definition of $\Vdash_t^{I(\beta)}$ we have that $a \Vdash_t^{I(\beta)} b$.

To check the second condition on $I(\boldsymbol{\beta})$ let a function $f \in \mathbf{C}[t, t']$ be given. We have to find f' such that $f' \Vdash_{(t,t')}^{I(\beta)} f$. We recall that this latter condition means the following

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

So 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 also entails that $r \Vdash_t^{\gamma} x$ as x has to be in $\left(\sum_{\mathbf{C}} \boldsymbol{\gamma}\right)(t, r)$. We use that $\boldsymbol{\gamma}$ is a simulation from $\mathbf{C}$ to Sets, so we obtain a $\tilde{f} \colon \gamma(t) \to \gamma(t')$ such that $\tilde{f} \Vdash_{(t,t')}^{\gamma} f$. We thus have $\tilde{f}(r) \Vdash_{t'}^{\gamma} f(x)$. This in turn shows that

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

as for all z such that $r \Vdash_t^{\gamma} z$ we have $\tilde{f}(r) \Vdash_{t'}^{\gamma} f(z)$ by definition of $\tilde{f}$. So as β is a simulation we obtain a $\hat{f} \colon \beta(t, r) \to \beta\big(t', \tilde{f}(r)\big)$ such that $\hat{f} \Vdash_{((t,r),(t',\tilde{f}(r)))}^{\beta} f$. Then by design $\hat{f}(y) \Vdash_{t'}^{\beta} f(x)$. We can thus define our final f' to be on each part $\beta(t, r)$ of $\bigcup_{r \in \gamma(t)} \beta(t, r)$ defined as $\hat{f}$ obtained in the above manner. It is immediate that this f' fulfils the desired equality.

So we have shown that $I(\boldsymbol{\beta})$ is a simulation and it remains to show that $I(\boldsymbol{\beta}) \preceq \boldsymbol{\gamma}$. For this we have to exhibit for each $t \in T$ a $g_t \colon I(\beta)(t) \to \gamma(t)$ such that $y \Vdash_t^{I(\beta)} x$ entails $g_t(y) \Vdash_t^{\gamma} x$. We again do this by defining g_t on each part $\beta(t, r)$ of $I(\beta)(t) = \bigcup_{r \in \gamma(t)} \beta(t, r)$ separately. On $\beta(t, r)$ define $g_t(y) = r$. Then by definition if $y \Vdash_t^{I(\beta)} x$, that is $y \Vdash_{(t,r)}^{\beta} x$ then $r \Vdash_t^{\gamma} x$ as $x \in \left(\sum_{\mathbf{C}} \boldsymbol{\gamma}\right)(t, r)$.

This shows that the functor I is indeed well-defined. It remains to show the three properties.

If $\boldsymbol{\alpha} \preceq \boldsymbol{\beta}$ then we have to show that $I(\boldsymbol{\alpha}) \preceq I(\boldsymbol{\beta})$. From the assumption we obtain for each $(t, r) \in \sum_{\mathbf{C}} T$ a function $g_{(t,r)} \colon \alpha(t, r) \to \beta(t, r)$ such that if $y \Vdash_{(t,r)}^{\alpha} x$ then $g_{(t,r)}(y) \Vdash_{(t,r)}^{\beta} x$. So 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 that these g_t show that $I(\boldsymbol{\alpha}) \preceq I(\boldsymbol{\beta})$.

For the reverse direction we need to show that $I(\boldsymbol{\alpha}) \preceq I(\boldsymbol{\beta})$ entails $\boldsymbol{\alpha} \preceq \boldsymbol{\beta}$. But this is immediate as we do the above "in reverse", that is we define $g_{(t,r)}(y) := g_t(y)$, where $g_{(t,r)}, g_t$ take the same roles as in the preceding paragraph.

At last we have to show essential surjectivity, that is every $\boldsymbol{\beta} \colon \mathbf{C} \to \mathsf{Sets}$ with $\boldsymbol{\beta} \preceq \boldsymbol{\gamma}$ is equivalent to a $I(\boldsymbol{\alpha})$ for some $\boldsymbol{\alpha} \colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \rightarrowtriangle \mathsf{Sets}$. So let $\boldsymbol{\beta}$ be given. We define $\boldsymbol{\alpha}$ in the following way: For $(t, r) \in \sum_{\mathbf{C}} T$ we let $\alpha(t, r) := \beta(t)$ and We further define the tracking relations $\Vdash_{(t,r)}^{\alpha}$ via

$$x \Vdash_{(t,r)}^{\alpha} a :\Leftrightarrow x \Vdash_t^{\beta} a.$$

We have to show that this $\boldsymbol{\alpha}$ indeed constitutes a simulation. As $\alpha \colon \sum_{\mathbf{C}} T \to \mathsf{Sets}$ is clearly well-defined and $\Vdash_{(t,r)}^{\alpha}$ as well it remains to show the two conditions imposed on simulations.

For the first condition let $a \in \left(\sum_{\mathbf{C}} \boldsymbol{\gamma}\right)(t, r)$ be given. We have to show that there exists $x \in \alpha(t, r)$ such that $x \Vdash_{(t,r)}^{\alpha} a$. This is given as there exists $x \in \beta(t) = \alpha(t, r)$ such that

$x \Vdash^{\beta}_{t} a$. The second condition is shown analogously, as one observes that $f' \Vdash^{\alpha}_{((t,r),(t',r'))} f$ if and only if $f' \Vdash^{\beta}_{(t,t')} f$.

So this $\boldsymbol{\alpha}$ constitutes a simulation. It is immediate that $I(\boldsymbol{\alpha}) = \boldsymbol{\beta}$ as $\bigcup_{r\in\gamma(t)} \alpha(t,r) = \bigcup_{r\in\gamma(t)} \beta(t) = \beta(t)$ and

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

This shows that I is essentially surjective, as if $I(\boldsymbol{\alpha}) = \boldsymbol{\beta}$, then obviously $I(\boldsymbol{\alpha}) \sim \boldsymbol{\beta}$. Q.E.D.

9.4.12 Presheaf-simulations out of Grothendieck models.

One of the key differences that arises when one considers computability models and presheaf simulations instead of categories and presheaves is that a datatype $b \in \mathbf{C}(t)$ can be tracked by multiple $r \in \gamma(t)$. Thus in passing to the Grothendieck construction $\sum_{\mathbf{C}} \boldsymbol{\gamma}$ one has that b is present in $\big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)(t,r)$ for each of the r that track b. The question then is whether in case we take a presheaf-simulation $\boldsymbol{\delta}\colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \mathrel{-\!\!\triangleright} \mathbf{Sets}$ the sets $\delta(t,r)$ intersect for all the r above.

We attack this question with a small example.

Example 9.4.13. Let $\mathbf{C}$ be the computability model over the set $\{0\}$ such that $\mathbf{C}(0) = \{0\}$ with only the identity. We define the simulation $\boldsymbol{\gamma}\colon \mathbf{C} \mathrel{-\!\!\triangleright} \mathbf{Sets}$ in the following way: $\gamma(0) = \{0,1\}$ and $0 \Vdash^{\gamma}_{0} 0, 1 \Vdash^{\gamma}_{0} 0$. This is a simulation as the identity on $\mathbf{C}(0) = \{0\}$ is tracked by the identity on $\{0,1\}$.

The Grothendieck model has thus as underlying set the set $\{(0,0),(0,1)\}$ which we will tacitly identify with $\{0,1\}$. Furthermore $\big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)(0) = \big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)(1) = \{0\}$. Now consider the simulation $\boldsymbol{\delta}\colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \mathrel{-\!\!\triangleright} \mathbf{Sets}$ defined by $\delta(0) = \{0,2\}, \delta(1) = \{1,3\}$ and where $0 \in \big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)(0)$ is tracked by 0 and $0 \in \big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)(1)$ is tracked by 1. This is indeed a simulation as the first condition imposed on simulations is satisfied, and for the second we observe that the only computable functions present are

$$\mathrm{id}_{\{0\}} \in \Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)[0,0] = \Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)[0,1] = \Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)[1,0] = \Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)[1,1]$$

and these are tracked by

- $\mathrm{id}_{\{0,2\}}$ in case $\mathrm{id}_{\{0\}} \in \big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)[0,0]$,
- $\{(0,1),(2,3)\}$ in case $\mathrm{id}_{\{0\}} \in \big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)[0,1]$
- $\{(1,0),(3,2)\}$ in case $\mathrm{id}_{\{0\}} \in \big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)[1,0]$,
- $\mathrm{id}_{\{1,3\}}$ in case $\mathrm{id}_{\{0\}} \in \big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)[1,1]$.

This example shows that unfortunately we do not have that $\boldsymbol{\delta}(0)$ and $\boldsymbol{\delta}(1)$ intersect, but in our case we at least obtained a computable function between the different sets of data types. However this need not always be the case as the following example shows:

Example 9.4.14. Let $\mathbf{C}$ be the computability model over $\{0\}$ given by $\mathbf{C}(0) = \{0,1,2\}$ and let $\boldsymbol{\gamma}$ be given in the following different way: the underlying class function γ is given by $0 \mapsto \{0,1\}$ and

$$0 \Vdash^{\gamma}_{0} 0, \quad 1 \Vdash^{\gamma}_{0} 0, \quad 1 \Vdash^{\gamma}_{0} 1, \quad 0 \Vdash^{\gamma}_{0} 2.$$

Then the Grothendieck model is given again as the model over the set $\{(0,0),(0,1)\}$, which we again replace by $\{0,1\}$ and the data types are given as $\big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)(0) = \{0,2\}$, $\big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)(1) = \{0,1\}$. However $\big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)[0,1] = \big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\big)[1,0]$ as the identity can not be present in the former because $1 \nVdash^{\gamma}_{0} \mathrm{id}_{\{0,1,2\}}(2) = 2$ and not in the latter because $0 \nVdash \mathrm{id}_{\{0,1,2\}}(1) = 1$.

9.5 Fibration-simulations and opfibration-simulations

The (covariant) Grothendieck construction allows the generation of fibrations (opfibrations), as the first-projection functor $\mathsf{pr}_1 \colon \sum_{\mathscr{C}} P \to \mathscr{C}$ 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 $\mathsf{pr}_{\mathbf{1}} \colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \rightharpoonup \mathbf{C}$ 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 $\mathbf{E} \rightharpoonup \mathbf{B}$. In contrast to what it holds for functors, for simulations $\boldsymbol{\gamma} \colon \mathbf{E} \rightharpoonup \mathbf{B}$ 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 9.5.1 (Cartesian computable functions). 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 $f' \Vdash_{(t,t')}^{\varpi} f$, 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

$$\begin{tikzcd}
 & \mathbf{E}(t) \arrow[rr, dashed, no head, "\Vdash_t^{\varpi}"] \arrow[dd, "f"] & & \mathbf{B}(\varpi(t)) \arrow[dd, "f'"] \\
\mathbf{E}(t'') \arrow[ur, "k"] \arrow[dr, "g"'] \arrow[rr, dashed, no head, "\Vdash_{t''}^{\varpi}"] & & \mathbf{B}(\varpi(t'')) \arrow[ur, "h"] \arrow[dr, "g'"] & \\
 & \mathbf{E}(t') \arrow[rr, dashed, no head, "\Vdash_{t'}^{\varpi}"] & & \mathbf{B}(\varpi(t'))
\end{tikzcd}$$

that is g' tracks g, there is some $k \in \mathbf{E}[t'', t]$ satisfying the following property: $h \Vdash_{(t'',t)}^{\varpi} k$, and for every $x \in \mathbf{E}(t''), y \in \mathbf{B}(\varpi(t''))$, such that $y \Vdash_{t''}^{\varpi} x$, $y \in \mathrm{dom}(f' \circ h) \cap \mathrm{dom}(g')$, 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 9.5.2 (Opcartesian computable functions). 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 $f' \Vdash_{(t',t')}^{\varpi} f$, 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

$$\begin{tikzcd}
 & \mathbf{E}(t) \arrow[rr, dashed, no head, "\Vdash_t^{\varpi}"] \arrow[dl, "l"'] & & \mathbf{B}(\varpi(t)) \arrow[dl, "h"] \\
\mathbf{E}(t'') \arrow[rr, dashed, no head, "\Vdash_{t''}^{\varpi}"] & & \mathbf{B}(\varpi(t'')) & \\
 & \mathbf{E}(t') \arrow[uu, "f"] \arrow[ul, "g"] \arrow[rr, dashed, no head, "\Vdash_{t'}^{\varpi}"] & & \mathbf{B}(\varpi(t')) \arrow[uu, "f'"'] \arrow[ul, "g'"']
\end{tikzcd}$$

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 9.5.3 (Fibration-simulations). We call $\boldsymbol{\varpi}\colon \mathbf{E} \mathrel{-\!\!\triangleright} \mathbf{B}$ 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 9.5.4 (Opfibration-simulations). 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 9.5.5. 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, $\boldsymbol{\gamma}^F\colon\ \mathbf{CM}^{\mathrm{tot}}(\mathcal{E}; S) \mathrel{-\!\!\triangleright} \mathbf{CM}^{\mathrm{tot}}(\mathcal{B}; S')$ 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,

$$\begin{tikzcd}
& S(e) \arrow[rr, dash, dashed, "\Vdash^{\gamma^F}_{e}"] \arrow[dd, "S(g)"] & & S'(b) \arrow[dd, "S'(f)"] \\
S(e'') \arrow[ur, "S(k)"] \arrow[dr, "S(g_2)"'] \arrow[rr, dash, dashed, "\Vdash^{\gamma^F}_{e''}"] & & S'(b'') \arrow[ur, "S'(h)"] \arrow[dr, "S'(h_2)"] & \\
& S(e') \arrow[rr, dash, dashed, "\Vdash^{\gamma^F}_{e'}"] & & S'(b')
\end{tikzcd}$$

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 9.5.6. *If* $\mathbf{C}$ *is a computability model and* $\gamma\colon \mathbf{C} \to \mathbf{Sets}$ *a simulation, then the first-projection-simulation* $\mathbf{pr}_1\colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \mathrel{-\!\!\triangleright} \mathbf{C}$ *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

$$\begin{tikzcd}
& \Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)(t, b) \arrow[dl, "g"'] \arrow[dd, "f"] \arrow[rr, dash, dashed, "\Vdash^{\mathsf{pr}_1}_{(t,b)}"] & & \mathbf{C}(t) \arrow[dl, "g"'] \arrow[dd, "f"] \\
\Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)(t'', b'') \arrow[dr, "h"] \arrow[rr, dash, dashed, "\Vdash^{\mathsf{pr}_1}_{(t'',b'')}"] & & \mathbf{C}(t'') \arrow[dr, "h"] & \\
& \Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)(t', b') \arrow[rr, dash, dashed, "\Vdash^{\mathsf{pr}_1}_{(t',b')}"] & & \mathbf{C}(t')
\end{tikzcd}$$

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 $\Big(\sum_{\mathbf{C}} \boldsymbol{\gamma}\Big)(t', b')$. Q.E.D.

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

Definition 9.5.7 (Splittings). A *splitting* for a fibration-simulation $\boldsymbol{\varpi}\colon \mathbf{E} \mathrel{-\!\!\vartriangleright} \mathbf{B}$ is a rule $\varpi^{\triangle}$ that takes 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 $\boldsymbol{\varpi}\colon \mathbf{E} \mathrel{-\!\!\vartriangleright} \mathbf{B}$ 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 9.5.8. *The simulation* $\mathbf{pr}_1\colon \sum_{\mathbf{C}} \boldsymbol{\gamma} \mathrel{-\!\!\vartriangleright} \mathbf{C}$ *is a split opfibration-simulation.*

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

9.5.9 A 2-categorical approach.

Our approach to fibrations and cartesian arrows looks a bit different than what the 2-categorically inclined mind envisioned. Indeed, in the works of Riehl [56] and Wong [65] one sees a 2-categorical approach to cartesian arrows and fibrations that should also be translatable to our framework, as the category CompMod of computability models constitutes a 2-category.

We want to examine the differences between the 2-categorical approach to fibrations and our bottom-up approach. It remains to interpret the meaning of the the abstract notion of a fibration in a 2-category in the category CompMod.

A 2-cell $\boldsymbol{\gamma} \Rightarrow \boldsymbol{\delta}$ expresses nothing more than that $\boldsymbol{\gamma}$ is transformable into $\boldsymbol{\delta}$. So what does it mean for this transformability to be cartesian with respect to another simulation $\boldsymbol{\varpi}$? So assume we are given this exact situation, that is

$$\begin{tikzcd}
\mathbf{C} \arrow[r, bend left, "\boldsymbol{\gamma}"] \arrow[r, bend right, "\boldsymbol{\delta}"'] \arrow[r, phantom, "\rotatebox{90}{$\preceq$}"] & \mathbf{D} \arrow[r, "\boldsymbol{\varpi}"] & \mathbf{E}
\end{tikzcd}$$

Then if we are given a simulations $\boldsymbol{\beta}\colon \mathbf{B} \to \mathbf{C}, \boldsymbol{\epsilon}\colon \mathbf{B} \to \mathbf{D}$ such that $\boldsymbol{\epsilon} \preceq \boldsymbol{\delta} \circ \boldsymbol{\beta}$ and $\boldsymbol{\varpi} \circ \boldsymbol{\epsilon} \preceq \boldsymbol{\varpi} \circ \boldsymbol{\gamma} \circ \boldsymbol{\beta}$ subject to

$$\begin{tikzcd}
\mathbf{B} \arrow[r, bend left, "\boldsymbol{\varpi}\circ\boldsymbol{\epsilon}"] \arrow[r, bend right, "\boldsymbol{\varpi}\circ\boldsymbol{\delta}\circ\boldsymbol{\beta}"'] \arrow[r, phantom, "\rotatebox{90}{$\succeq$}"] & \mathbf{E} & = & \mathbf{B} \arrow[r, "\boldsymbol{\beta}"'] \arrow[rrr, bend left=50, "\boldsymbol{\varpi}\circ\boldsymbol{\epsilon}"] & \mathbf{C} \arrow[r, bend left, "\boldsymbol{\gamma}"] \arrow[r, bend right, "\boldsymbol{\delta}"'] \arrow[r, phantom, "\rotatebox{90}{$\preceq$}"] & \mathbf{D} \arrow[r, "\boldsymbol{\varpi}"] & \mathbf{E}
\end{tikzcd},$$

we obtain that $\boldsymbol{\epsilon} \preceq \boldsymbol{\gamma} \circ \boldsymbol{\beta}$. Visually, this can be encoded as

$$\begin{tikzcd}
\mathbf{B} \arrow[r, "\boldsymbol{\beta}"] \arrow[rrr, bend left=50, "\boldsymbol{\varpi}\circ\boldsymbol{\epsilon}"] \arrow[rr, bend right=50, "\boldsymbol{\epsilon}"'] & \mathbf{C} \arrow[r, bend left, "\boldsymbol{\gamma}"] \arrow[r, bend right, "\boldsymbol{\delta}"'] & \mathbf{D} \arrow[r, "\boldsymbol{\varpi}"] & \mathbf{E} & \rightsquigarrow & \mathbf{B} \arrow[r, "\boldsymbol{\beta}"] \arrow[rr, bend left=50, "\boldsymbol{\epsilon}"] & \mathbf{C} \arrow[r, bend left, "\boldsymbol{\gamma}"] \arrow[r, bend right, "\boldsymbol{\delta}"'] & \mathbf{D} \arrow[r, "\boldsymbol{\varpi}"] & \mathbf{E}.
\end{tikzcd}$$

Having seen what being cartesian with respect to a simulation means, we can turn to examining fibration simulations. The definition is easy to translate. A simulation $\boldsymbol{\gamma}\colon \mathbf{C} \to \mathbf{D}$ is a fibration if for all simulations $\boldsymbol{\alpha}\colon \mathbf{B} \to \mathbf{C}, \boldsymbol{\beta}\colon \mathbf{B} \to \mathbf{D}$ such that $\boldsymbol{\beta} \preceq \boldsymbol{\gamma} \circ \boldsymbol{\alpha}$ we find a simulation $\boldsymbol{\alpha}'\colon \mathbf{B} \to \mathbf{C}$ cartesian with respect to $\boldsymbol{\gamma}$ such that $\boldsymbol{\beta} \preceq \boldsymbol{\gamma} \circ \boldsymbol{\alpha}'$. At last we need to concern ourselves with how to transform the notion "$p \bullet \alpha = \beta$" into the language of computability models. This simply means the following: If $f_t \in \mathbf{D}[\beta(t), \gamma(\alpha((t))]$ is our witness of transformability and $x \in \mathrm{dom}(f_t), x \Vdash_t^{\beta} y$, then there exists a q such that $q \Vdash_t^{\alpha'} x$ and a computable arrow f'_t such that $f_t \Vdash^{\alpha'}_{(\alpha'(t),\alpha(t)} f'_t$ and $q \in \mathrm{dom}(f'_t)$.
In pictures:

$$\begin{tikzcd}
 & & & x \arrow[r, "f_t"] \arrow[d, dash, dashed] & f_t(x) \arrow[d, dash, dashed] \\
x \arrow[r, "f_t"] \arrow[dr, dash, dashed] & f_t(x) & = & q \arrow[r, "f'_t"] \arrow[dr, dash, dashed] & f_t(x) \\
 & y \arrow[u, dash, dashed] & & & y \arrow[u, dash, dashed]
\end{tikzcd}$$

Thus we can restate the definition of a $\boldsymbol{\varpi}$-cartesian transformability

Definition 9.5.10 ($\boldsymbol{\varpi}$-cartesian transformability). Let $\mathbf{B}, \mathbf{C}, \mathbf{D}$ be computability models over S, T, U respectively with simulations $\boldsymbol{\gamma}, \boldsymbol{\delta}\colon \mathbf{B} \rightarrowtriangle \mathbf{C}, \boldsymbol{\varpi}\colon \mathbf{C} \rightarrowtriangle \mathbf{D}$. We say that a transformability $\boldsymbol{\gamma} \preceq \boldsymbol{\delta}$ is *cartesian* (in the 2-categorical sense) with respect to $\boldsymbol{\varpi}$ if:

- given a simulation $\boldsymbol{\alpha}\colon \mathbf{A} \to \mathbf{B}$ where $\mathbf{A}$ lives over V and a simulation $\boldsymbol{\epsilon}\colon \mathbf{A} \to \mathbf{C}$ such that $\boldsymbol{\epsilon} \preceq \boldsymbol{\delta} \circ \boldsymbol{\alpha}$ and $\boldsymbol{\varpi} \circ \boldsymbol{\epsilon} \preceq \boldsymbol{\varpi} \circ \boldsymbol{\gamma} \circ \boldsymbol{\alpha}$, that is for all $v \in V$ we obtain
  - $f_v \in \mathbf{C}[\epsilon(v), \delta(\alpha(v))]$ witnessing $\boldsymbol{\epsilon} \preceq \boldsymbol{\delta} \circ \boldsymbol{\alpha}$,
  - $f'_v \in \mathbf{D}[\varpi(\epsilon(v)), \varpi(\delta(\alpha(v)))]$ such that $f'_v \Vdash^{\varpi}_{(\epsilon(v),\delta(\alpha(v)))} f_v$,
  - $g_{\alpha(v)} \in \mathbf{B}[\gamma(\alpha(v)), \delta(\alpha(v))]$ arising from $\boldsymbol{\gamma} \preceq \boldsymbol{\delta}$,
  - $g'_{\alpha(v)} \in \mathbf{E}[\varpi(\alpha(\gamma(v))), \varpi(\alpha(\delta(v)))]$ such that $g'_{\alpha(v)} \Vdash^{\varpi}_{(\alpha(\gamma(v)),\alpha(\delta(v)))} g_{\alpha(v)}$,
  - $h_v \in \mathbf{E}[\varpi(\epsilon(v)), \varpi(\gamma(\alpha(v)))]$ arising from $\boldsymbol{\varpi} \circ \boldsymbol{\epsilon} \preceq \boldsymbol{\varpi} \circ \boldsymbol{\gamma} \circ \boldsymbol{\alpha}$

  and these computable functions fulfil the computability condition that if $x \in \mathbf{A}(v)$ and $y \Vdash^{\varpi\circ\epsilon}_v x$ then

  $$y \in \mathrm{dom}(f'_v) \cap \mathrm{dom}(g'_{\alpha(v)} \circ h_v) \quad \text{and} \quad f'_v(y) = g'_{\alpha(v)}(h_v(y)),$$

- then we obtain $\boldsymbol{\epsilon} \preceq \boldsymbol{\gamma} \circ \boldsymbol{\alpha}$ in the following compatible manner: for all $v \in V$ we obtain
  - $f_v \in \mathbf{C}[\epsilon(v), \delta(\alpha(v))]$ witnessing $\boldsymbol{\epsilon} \preceq \boldsymbol{\delta} \circ \boldsymbol{\alpha}$,
  - $i_v \in \mathbf{C}[\epsilon(v), \gamma(\alpha(v))]$ witnessing $\boldsymbol{\epsilon} \preceq \boldsymbol{\gamma} \circ \boldsymbol{\alpha}$
  - $g_{\alpha(v)} \in \mathbf{B}[\gamma(\alpha(v)), \delta(\alpha(v))]$ arising from $\boldsymbol{\gamma} \preceq \boldsymbol{\delta}$

  such that if $x \in \mathbf{A}(v)$ and $y \Vdash^{\epsilon}_v x$, then

  $$y \in \mathrm{dom}(g_{\alpha(v)} \circ i_v) \cap \mathrm{dom}(f_v) \quad \text{and} \quad g_{\alpha(v)}(i_v(y)) = f_v(y).$$

The definition of a fibration 2-cell is the following:

Definition 9.5.11 (Fibration-2-cells in a 2-category). Let $\mathscr{C}$ be a small 2-category. A 1-cell $p\colon B \to C$ is a *fibration* if and only if every 2-cell β as in

$$\begin{tikzcd}
X \arrow[r, "b"] \arrow[dr, "c"'] & B \arrow[d, "p"] \\
& C
\arrow[from=2-2, to=1-2, phantom, ""]
\end{tikzcd}$$

$$\begin{tikzcd}
X \arrow[r, "b"{name=B}] \arrow[dr, "c"'{name=C}] & B \arrow[d, "p"] \\
& C
\arrow[from=C, to=B, Rightarrow, "\beta"]
\end{tikzcd}$$

has a p-cartesian lift $\alpha\colon b' \Rightarrow b$ such that $p \bullet \alpha = \beta$.

We want to show that this definition of a fibration is entailed by the bottom-up definition we gave earlier. However this is (as of our knowledge) only possible if our computability models $\mathbf{C}, \mathbf{D}$ have constants, which is the following:

Definition 9.5.12. We say that a computability model $\mathbf{C}$ *has all constants* if for any $t \in T$ and any $a \in \mathbf{C}(T)$ and any $t' \in T$ the total function that sends every $x \in \mathbf{C}(t')$ to a is in $\mathbf{C}[t', t]$. We write $c_a^{t'}$ for this function.

Theorem 9.5.13. *Let* $\mathbf{E}, \mathbf{B}$ *be computability models over* T, U *respectively with all constants. If* $\boldsymbol{\varpi}$ *is a fibration in the bottom-up sense, then* $\boldsymbol{\varpi}$ *is a fibration in the 2-categorical sense.*

Proof: We first show that if it is a fibration in the bottom-up sense it is in the 2-categorical sense. For this let a computability model $\mathbf{C}$ (over V) with simulations $\boldsymbol{\chi}\colon \mathbf{C} \to \mathbf{E}, \boldsymbol{\xi}\colon \mathbf{C} \to \mathbf{B}$ be given such that $\boldsymbol{\xi} \preceq \boldsymbol{\varpi} \circ \boldsymbol{\chi}$. We need to find a simulation $\boldsymbol{\chi}'\colon \mathbf{C} \to \mathbf{E}$ such that $\boldsymbol{\chi}' \preceq \boldsymbol{\chi}$ is $\boldsymbol{\varpi}$-cartesian. For this consider an arbitrary $v \in V$ and $f_v \in \mathbf{B}[\xi(v), \varpi(\chi(v))]$ that tracks the transformability. As $\boldsymbol{\varpi}$ is a fibration in the bottom-up sense we obtain a lift $\hat{f}_v \in \mathbf{E}[w, \chi(x)]$ cartesian for f_v and $\chi(x)$. For each v we make such a choice of w and arrive at a map $\chi'\colon V \to T, v \mapsto w$. This will be the underlying class function of our desired simulation $\boldsymbol{\chi}'$.

For the tracking relations $\Vdash_v^{\chi'}$ we take $x \in \mathbf{C}(v)$. By definition there exists a $y \in \mathbf{B}(\xi(v))$ such that $y \Vdash_v^{\xi} x$, and thus $f_v(y) \Vdash_v^{\varpi \circ \chi} x$. Thus by definition we obtain a $z \in \mathbf{E}(\chi(v))$ such that $f_v(y) \Vdash_{\chi(v)}^{\varpi} z$ and $z \Vdash_v^{\chi} x$. We consider the functions

$$\begin{aligned}
c_z^{\chi(v)} &\in \mathbf{E}[\chi(v), \chi(v)], & c_y^{\varpi(\chi(v))} &\in \mathbf{B}[\varpi(\chi(v)), \xi(v)], \\
c_{f_v(y)}^{\varpi(\chi(v))} &\in \mathbf{B}[\varpi(\chi(v)), \varpi(\chi(v))], & \hat{f}_v &\in \mathbf{E}[w, \chi(v)].
\end{aligned}$$

We then obtain the situation as in the following diagram:

$$\begin{tikzcd}
& \mathbf{E}(w) \arrow[rr, dash, dashed, "\Vdash_w^{\varpi}"] \arrow[dd, "\hat{f}_v"] & & \mathbf{B}(\xi(v)) \arrow[dd, "f_v"] \\
\mathbf{E}(\chi(v)) \arrow[rr, dash, dashed, "\Vdash_{\chi(v)}^{\varpi}"] \arrow[dr, "c_z^{\chi(v)}"] & & \mathbf{B}(\varpi(\chi(v))) \arrow[ur, "c_y^{\varpi(\chi(v))}"] \arrow[dr, "c_{f_v(y)}^{\varpi(\chi(v))}"] & \\
& \mathbf{E}(\chi(v)) \arrow[rr, dash, dashed, "\Vdash_{\chi(v)}^{\varpi}"] & & \mathbf{B}(\varpi(\chi(v)))
\end{tikzcd}$$

Here the triangle on the right commutes, so as $\hat{f}_v$ is cartesian for f_v and $\chi(v)$ we obtain a function $h \in \mathbf{E}[\chi(v), w]$ making the triangle on the right commute. Furthermore $z \in \operatorname{dom}(h)$ so we can define $b := h(z)$. We now say that $b \Vdash_v^{\chi'} x$ if b arises in the fashion described above from some y, z. Now obviously the first condition of a simulation is fulfilled.

For the second condition we assume we are given $h \in \mathbf{C}[v, v']$. Then we can find $h' \in \mathbf{B}[\xi(v), \xi(v')], h'' \in \mathbf{E}[\chi(v), \chi(v')]$ and $h''' \in \mathbf{B}[\rho(\chi(v)), \rho(\chi(v'))]$ such that

$$h' \Vdash^{\xi}_{(v,v')} h, \quad h'' \Vdash^{\chi}_{(v,v')} h, \quad h''' \Vdash_{(\chi(v),\chi(v'))} h''.$$

In a diagram this looks like this:

$$\begin{tikzcd}
 & & \mathbf{C}(v) \arrow[dl, harpoon, "h"] \arrow[ddl, dash, dashed] \arrow[ddr, dash, dashed] & \\
 & \mathbf{C}(v') \arrow[ddl, dash, dashed] \arrow[ddr, dash, dashed] & & \\
 & \mathbf{E}(w) \arrow[rr, harpoon, "\hat{f}_v"] \arrow[dd, dash, dashed] & & \mathbf{E}(\chi(v)) \arrow[dl, harpoon, "h''"] \arrow[dd, dash, dashed] \\
\mathbf{E}(w') \arrow[rr, harpoon, "\hat{f}_{v'}"] \arrow[dd, dash, dashed] & & \mathbf{E}(\chi(v')) \arrow[dd, dash, dashed] & \\
 & \mathbf{B}(\xi(v)) \arrow[rr, harpoon, "f_v"] \arrow[dl, harpoon, "h'"] & & \mathbf{B}(\varpi(\chi(v))) \arrow[dl, harpoon, "h'''"] \\
\mathbf{B}(\xi(v')) \arrow[rr, harpoon, "f_{v'}"] & & \mathbf{B}(\varpi(\chi(v'))) &
\end{tikzcd}$$

Thus as $\hat{f}_v$ is cartesian for $\chi(v)$ and f_v we obtain a $\tilde{h} \in \mathbf{E}[w, w']$ such that if $x \in E(w')$ and $y \Vdash^{\varpi}_{w'} x$ for some $y \in \mathbf{E}(\xi(v))$ such that $y \in \mathrm{dom}(f_v \circ h') \cap \mathrm{dom}(h''' \circ f_{v'})$ and $h'''\big(f_v(y)\big) = f_{v'}\big(h'(y)\big)$ then $x \in \mathrm{dom}(\tilde{h}) \cap \mathrm{dom}(h'' \circ \hat{f}_v)$ and $h''\big(\hat{f}_v(x)\big) = \hat{f}_{v'}\big(h(x)\big)$.

We now show that $\tilde{h} \Vdash^{\chi}_{(v,v')} h$. For this let $a \in \mathbf{C}(v)$ with $a \in \mathrm{dom}(h)$ and $b \in \mathbf{E}(w)$ with $b \Vdash^{\chi'}_{v} a$ be given. By definition this means that there exists $c \in \xi(v)$ and $d \in \mathbf{E}(\chi(v))$ such that

$$y \Vdash^{\xi}_{v} a, \quad d \Vdash^{\chi}_{v} a, \quad f_v(c) \Vdash^{\varpi}_{\chi(v)} d.$$

and $b = i(z)$ where i arises from lifting c_z through f_v as above. Thus $c \Vdash^{\varpi}_{w} b$ and

- $c \in \mathrm{dom}(h')$ as h' tracks h and $a \in \mathrm{dom}(h)$,
- $c \in \mathrm{dom}(f_v)$ as f_v witnesses the transformability of $\boldsymbol{\xi}$ to $\boldsymbol{\varpi} \circ \boldsymbol{\chi}$
- $h'(c) \in \mathrm{dom}(f_{v'})$ as $h'(c) \Vdash h(a)$ and $f_{v'}$ witnesses the transformability of $\boldsymbol{\xi}$ to $\boldsymbol{\varpi} \circ \boldsymbol{\chi}$.
- $f_v(c) \in \mathrm{dom}(h''')$ as $f_v(c) \Vdash^{\varpi \circ \chi}_{v} a$ and h''' tracks h.

Then $f_{v'}\big(h'(c)\big) = h'''\big(f_v(c)\big)$ and thus as desired $b \in \mathrm{dom}(\tilde{h})$. Thus $\tilde{h}$ tracks h.

This finishes the proof that $\boldsymbol{\chi}'$ is a simulation and it remains to show that $\boldsymbol{\chi}' \preceq \boldsymbol{\chi}$. The functions that witness the transformability are the $\hat{f}_v \in \mathbf{E}[w, \chi(v)]$ for all $v \in V$. If $a \in \mathbf{C}(v)$ and $b \Vdash^{\chi'}_{v} a$ then $b \in \mathrm{dom}(\hat{f}_v)$ by the above discussion and obviously $\hat{f}_v(b) \Vdash^{\chi}_{v} a$.

It remains to show that this transformability is $\boldsymbol{\varpi}$-cartesian. For this let another computability model $\mathbf{A}$ over S together with simulations $\boldsymbol{\alpha}\colon \mathbf{A} \multimap \mathbf{C}$ and $\boldsymbol{\beta}\colon \mathbf{A} \to \mathbf{E}$ be given such that

$$\boldsymbol{\beta} \preceq \boldsymbol{\chi} \circ \boldsymbol{\alpha} \text{ and } \boldsymbol{\varpi} \circ \boldsymbol{\beta} \preceq \boldsymbol{\varpi} \circ \boldsymbol{\chi}' \circ \boldsymbol{\alpha}.$$

and We have to show that $\boldsymbol{\beta} \preceq \boldsymbol{\chi}' \circ \boldsymbol{\alpha}$. Let $s \in S$ be given. We know that there exist $h_s \in \mathbf{E}[\beta(s), \chi(\alpha(s))]$ and $g_s \in \mathbf{B}[\varpi(\beta(s)), \varpi(\chi'(\alpha(s)))]$ witnessing these transformabilities. and $h'_s \in \mathbf{B}[\varpi(\beta(s)), \varpi(\chi(\alpha(s)))]$ tracking h_s through $\boldsymbol{\varpi}$ compatible. The desired $j_s \in$

$[\beta(s), \chi'(\alpha(s))]$ witnessing the transformability is constructed as such: As $\boldsymbol{\varpi} \circ \boldsymbol{\chi}' = \boldsymbol{\xi}$ our given computable functions are arranged as follows:

$$\begin{tikzcd}
& \mathbf{E}(\chi'(\alpha(s))) \arrow[rr, dash, dashed, "\Vdash^{\varpi}_{\chi'(\alpha(s))}"] \arrow[dd, "\hat{f}_{\alpha(s)}"] & & \mathbf{B}(\xi(\alpha(s))) \arrow[dd, "f_{\alpha(s)}"] \\
\mathbf{E}(\beta(s)) \arrow[rr, dash, dashed, "\Vdash^{\varpi}_{\beta(s)}"] \arrow[dr, "h_s"] & & \mathbf{B}(\varpi(\beta(s))) \arrow[ur, "g_s"] \arrow[dr, "h'_s"] & \\
& \mathbf{E}(\chi(\alpha(s))) \arrow[rr, dash, dashed, "\Vdash^{\varpi}_{\chi(\alpha(s))}"] & & \mathbf{B}[\varpi(\chi(\alpha(s)))
\end{tikzcd}$$

Now as $\hat{f}_{\alpha(s)}$ is cartesian for $\chi'(\alpha(s))$ and $f_{\alpha(s)}$ we obtain a $k \in \mathbf{E}[\beta(s)), \chi'(\alpha(s))]$ such that if $x \in \mathrm{dom}(h_s)$ and $y \Vdash^{\varpi}_{\beta(s)}$ and $y \in \mathrm{dom}(f_{\alpha(s)} \circ g_s) \cap \mathrm{dom}(h'_s)$ then $x \in \mathrm{dom}(f_{\alpha(s)} \circ k)$ and $f_{\alpha(s)}(k(x)) = h_s(x)$.

It remains to show that this k witnesses the transformability. For this let $s \in S$ be given with $y \in \mathbf{E}(\beta(s))$ such that $y \Vdash^{\beta}_{s} x$. Then $y \in \mathrm{dom}(h_s)$, and as $\boldsymbol{\varpi}$ is a simulation we find $z \in \mathbf{B}(\varpi(\beta(s)))$ such that $z \Vdash^{\varpi}_{\beta(s)} y$. Then $z \in \mathrm{dom}(h'_s)$, as $y \in \mathrm{dom}(h_s)$ and h' tracks h, as well as $z \in \mathrm{dom}(g_s)$ as $z \Vdash^{\varpi \circ \beta}_{s} x$ and g_s witnesses the transformability. Furthermore $g_s(y) \in \mathrm{dom}(f_{\alpha(s)})$ as $f_{\alpha(s)}$ witnesses the transformability and $g_s(y) \Vdash^{\beta}_{\alpha(s)} q$ for some $q \in \mathbf{C}(\alpha(s))$ as we know that $g_s(y) \Vdash^{\varpi \circ \chi' \circ \alpha}_{s} x$ and as $\boldsymbol{\varpi} \circ \boldsymbol{\chi}' = \boldsymbol{\xi}$ we get that $g_s(y) \Vdash^{\xi \circ \alpha}_{s} x$, that is there exists q such that $g_s(y) \Vdash^{\xi}_{\alpha(s)} q$ and $q \Vdash^{\alpha}_{s} x$.

Furthermore $f_{\alpha(s)}(g_s(y)) = h'_s(y)$, as these computable functions are compatible. Thus $x \in \mathrm{dom}(k)$ as desired and $\hat{f}_{\alpha(s)}(k(x)) = h_s(x)$. Q.E.D.

Chapter 10
The category of categories with a base of computability

Having established our working notion of a computability model, we turn to the bases of computability. The notion we use was introduced in [49].

Let $\mathscr{C}$ be a category. A *base of computability* B on $\mathscr{C}$ is a family $(B(a))_{a\in\mathscr{C}_0}$ of subclasses $B(a) \subseteq \mathrm{Mon}(\text{-}\,, a)$, such that the following conditions are satisfied:

($\texttt{Base}_1$) For each $a \in \mathscr{C}$ we have $\mathbf{1}_a \in B(a)$.

($\texttt{Base}_2$) Given $a, b \in \mathscr{C}$ and $i\colon s \to a, j\colon t \to b$ such that $i \in B(a), j \in B(b)$ and $f\colon s \to b$, we have the following: The pullback $f^{-1}(t)$ exists, and the diagram

$$\begin{tikzcd}
 & f^{-1}(t) \arrow[r, "j^{-1}f"] \arrow[d, "f^{-1}j"] \arrow[dl, "i\circ f^{-1}j"'] \arrow[dr, phantom, "\lrcorner", very near start] & t \arrow[d, "j"] \\
a & s \arrow[l, "i"] \arrow[r, "f"'] & b
\end{tikzcd}$$

commutes and $i \circ f^{-1}j \in B(a)$.

Examples 10.0.1.

1. For any category $\mathscr{C}$ we can consider the total base tot consisting of only identities, that is $\mathrm{tot}(c) = \{\mathbf{1}_c\}$ for any $c \in \mathscr{C}$.
2. On the other extreme if $\mathscr{C}$ has pullbacks we can consider the base prt consisting of all monomorphisms, that is $\mathrm{prt}(c) = \{f\colon b \to c \text{ mono }\}$.
3. We can also for any category $\mathscr{C}$ consider the base I of isomorphisms, that is $I(c)$ is the set of all isomorphisms with codomain c. This works as for all isomorphisms $i\colon b \to c$ and arbitrary $f\colon a \to c$ we have that

$$\begin{tikzcd}
a \arrow[r, "i^{-1}\circ f"] \arrow[d, "\mathbf{1}_a"'] \arrow[dr, phantom, "\lrcorner", very near start] & b \arrow[d, "i"] \\
a \arrow[r, "f"] & c
\end{tikzcd}$$

is a pullback square.

10.1 The category CatBaseComp

Definition 10.1.1 (Computability transformations). Suppose $\mathscr{C}$ and $\mathscr{D}$ are categories, with bases of computability $B = (B(a))_{a\in\mathscr{C}}$ and $B' = (B'(d))_{d\in\mathscr{D}}$ respectively. A *computability transformation* $F\colon (\mathscr{C}, B) \to (\mathscr{D}, B')$ is a functor $F\colon \mathscr{C} \to \mathscr{D}$ such that we have:

($\texttt{CTraf}_1$) The functor F is pullback-preserving.

($\texttt{CTraf}_2$) For each $a \in \mathscr{C}$ we have $F\big(B(a)\big) = \big\{F(i) \mid i \in B(a)\big\} \subseteq B\big(F(a)\big)$.

It is quite interesting, that a very similar notion to our notion of a base of computability is introduced by Rosolini , which he calls *dominions* in [58, p. 28].

“Suppose A is a given category with binary products, and $\mathcal{M}$ a family of monos of A closed under identity and composition, with the property that any pullback of a monic in $\mathcal{M}$ exists in A and a representative of it belongs to $\mathcal{M}$. We shall call such a family a *dominion.*”

The only difference between our bases of computability and dominions is that we do not require the underlying category to have binary products. The property that our structure must be closed can be immediately derived from property ($\mathtt{Base}_2$): Let B be our base of computability on the category $\mathscr{C}$ and $i\colon s \hookrightarrow a$ be in $B(a)$ and $j\colon t \hookrightarrow s$ be in $B(s)$. Then we can consider the pullback diagram

$$\begin{tikzcd}
\mathbf{1}_s^{-1}(t) \arrow[r, "j^{-1}\mathbf{1}_s"] \arrow[d, hook, "\mathbf{1}_s^{-1}j"'] \arrow[dr, phantom, "\lrcorner", very near start] & t \arrow[d, hook, "j"] \\
s \arrow[r, "\mathbf{1}_s"] & s.
\end{tikzcd}$$

This pullback $\mathbf{1}_s^{-1}(t)$ is given by t and $j^{-1}\mathbf{1}_s = j$ as well as $\mathbf{1}_s^{-1}j = \mathbf{1}_s$, so we obtain the diagram

$$\begin{tikzcd}
& t \arrow[r, "\mathbf{1}_s"] \arrow[d, hook, "j"'] \arrow[dl, "i\circ j"'] \arrow[dr, phantom, "\lrcorner", very near start] & t \arrow[d, hook, "j"] \\
a & s \arrow[l, hook', "i"'] \arrow[r, "\mathbf{1}_s"] & s
\end{tikzcd}$$

where $i \circ j \in B(a)$, as desired. Furthermore, an arrow that repects this structure is described as follows in [58, p. 28]:

“If B is another category with finite products and a dominion $\mathcal{N}$, a functor $F\colon \mathsf{A} \to \mathsf{B}$ which takes monos in $\mathcal{M}$ to monos in $\mathcal{N}$ is said to *preserve dominions.*”

Clearly this notion agrees with our definition of a computability transfer, albeit our categories need not have binary products.

Definition 10.1.2 (The category $\mathsf{CatBaseComp}$). Let $\mathsf{CatBaseComp}$ be the category whose

- *objects* are pairs $(\mathscr{C}, B)$, where $\mathscr{C}$ is a category and B is a base of computability on $\mathscr{C}$ and whose
- *morphisms* $F\colon (\mathscr{C}, B) \to (\mathscr{D}, B')$ are computability transformations.

Evidently we have a forgetful functor $\mathtt{Frg}\colon \mathsf{CatBaseComp} \to \mathbf{Cat}$, that simply “forgets” the computability base on the category, where $\mathbf{Cat}$ is the large category of locally small categories.

10.2 $\mathsf{CatBaseComp}$ has all PIE limits

In this section we show that $\mathsf{CatBaseComp}$ has all set-indexed products, as well as all equifiers and inserters, hence it has all PIE-limits as described in [53]. Pie limits were defined in [53] as a special kind of weighted limit for 2-categories.

Definition 10.2.1. Let $\mathscr{C}, \mathscr{W}$ be 2-categories and $F\colon \mathscr{W} \to \mathbf{Cat}, G\colon \mathscr{W} \to \mathscr{C}$ be 2-functors. An *F-weighted limit* of G—if it exists—is a representing object $\lim^F G$ of

$$[\mathscr{W}, \mathbf{Cat}]\Big(F, \mathscr{C}\big(\,\text{-}\,, G(\text{-})\big)\Big),$$

that is for every $C \in \mathscr{C}$ we have an isomorphism of categories

$$\mathscr{C}[C, \lim^F G] \cong [\mathscr{W}, \mathbf{Cat}]\Big(F, \mathscr{C}\big(C, G(\text{-})\big)\Big)$$

natural in C.

A textbook account of weighted limits can be found in [7, §6.1]. In this book weighted limits are introduced for arbitrary categories enriched over a symmetric monoidal closed category $\mathscr{V}$. We only require this notion in the case that $\mathscr{V} = \mathbf{Cat}$.

Examples 10.2.2.

1. *Products* are weighted limits where $\mathscr{W}$ is a discrete 2-category on a set I. The weight F maps all objects i to the terminal category $\mathbf{1}$.
2. *Inserters* are weighted limits where $\mathscr{W}$ is the 2-category given by

$$\begin{tikzcd} 0 \arrow[r, shift left, "f_0"] \arrow[r, shift right, "f_1"'] & 1. \end{tikzcd}$$

   The weight F takes 0 to the terminal 2-category $\mathbf{1}$ and 1 to the 2-category $[1]$ with two objects and only the non-trivial arrow $0 \to 1$. All 2-cells in $[1]$ are trivial. The arrow f_0 is mapped to the functor taking 0 to 0 and the identity of 0 to the identity of 0, and the arrow f_1 is mapped to the functor taking 0 to 1 and the identity to the identity.
3. *Equifiers* are weighted limits where $\mathscr{W}$ is the 2-category given by

$$\begin{tikzcd}[column sep=large] 0 \arrow[r, bend left, "f_0"{name=U}] \arrow[r, bend right, "f_1"'{name=D}] & 1. \arrow[Rightarrow, from=U, to=D, shift right=2, "\gamma_0"'] \arrow[Rightarrow, from=U, to=D, shift left=2, "\gamma_1"] \end{tikzcd}$$

   The weight F takes 0 to the terminal 2-category $\mathbf{1}$ and 1 to the 2-category $[1]$ as above.

Power and Robinson give an explicit description of these weighted limits in elementary terms in [53].

1. Products are simply products in the categorical sense with their universal property together with the following 2-dimensional universal property:

   Given two 1-cells (morphisms) $f, g\colon A \to \prod_{i \in I} A_i$, the 2-cells $\eta\colon f \Rightarrow g$ are in one-to-one-correspondence with families of 2-cells $\eta_i\colon \mathsf{pr}_i \circ f \Rightarrow \mathsf{pr}_i \circ g$.
2. An inserter of a diagram

$$\begin{tikzcd} A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B \end{tikzcd}$$

   in a 2-category $\mathscr{C}$ consists of an object I of $\mathscr{C}$ together with a morphism $i\colon I \to A$ and a 2-cell $\alpha\colon f \circ i \Rightarrow g \circ i$ that is universal with this property, namely the following properties are fulfilled:

   - *1-dimensional part:* For every other J with $j\colon J \to A, \beta\colon f \circ j \Rightarrow g \circ j$ we obtain a unique $h\colon J \to I$ making the obvious triangle commute.
   - *2-dimensional part:* For every other J, J' with j, β, j', β' as above and a 2-cell $\nu\colon j \to j'$ such that $\beta' \circ (f \bullet \nu) = (g \bullet \nu) \circ \beta$, that is

$$\begin{tikzcd} f \circ j \arrow[r, Rightarrow, "\beta"] \arrow[d, Rightarrow, "f \bullet \nu"'] & g \circ j \arrow[d, Rightarrow, "g \bullet \nu"] \\ f \circ j' \arrow[r, Rightarrow, "\beta'"'] & g \circ j' \end{tikzcd}$$

     commutes, we obtain a 2-cell $\mu\colon h \to h'$ such that $i \bullet \mu = \nu$

3. An equifier of a diagram

$$\begin{tikzcd} A \arrow[r, shift left=3, "f"] \arrow[r, shift right=3, "g"'] \arrow[r, phantom, "\gamma_0\Downarrow\,\Downarrow\gamma_1"] & B \end{tikzcd}$$

   in a 2-category $\mathscr{C}$ consists of an object I of $\mathscr{C}$ together with a morphism $i\colon I \to A$ such that $\gamma_1 \bullet i = \gamma_0 \bullet i$ that is universal with this property.

   - *1-dimensional part:* For every other J with $j\colon J \to A$ such that $\gamma_0 \bullet j = \gamma_1 \bullet j$ we obtain $h\colon J \to I$ such that $j = i \circ h$.
   - *2-dimensional part:* For every other J, J' with j, j' as above and $\nu\colon j \Rightarrow j'$ we obtain $\mu\colon h \to h'$ such that $i \bullet \mu = \nu$.

Examples 10.2.3. We will exhibit inserters and equifiers in the category of categories.

1. Given two functors $F, G\colon \mathscr{C} \to \mathscr{D}$, the inserter of F and G is given by its *category of dialgebras*—introduced by Lambek in [39] as the *subequalizing category* of F and G—$\mathsf{DiAlg}(F,G)$.

   - The objects of this category are pairs (x, γ) of arrows $\gamma\colon F(x) \to G(x)$.
   - The arrows $(x,\gamma) \to (y,\delta)$ are given by maps $f\colon x \to y$ such that

   $$\begin{tikzcd} F(x) \arrow[r, "\gamma"] \arrow[d, "F(f)"'] & G(x) \arrow[d, "G(f)"] \\ F(y) \arrow[r, "\delta"'] & G(y) \end{tikzcd}$$

     commutes.

   The functor $i\colon \mathsf{DiAlg}(F,G) \to \mathscr{C}$ maps (x,γ) to x and f to f.

2. Given two functors $F, G\colon \mathscr{C} \to \mathscr{D}$ and two natural transformations $\alpha, \beta\colon F \Rightarrow G$ the equifier $\mathrm{Eq}(\alpha,\beta)$ of α and β is given as the following category:

   - The objects are objects c of $\mathscr{C}$ such that $\alpha_c = \beta_c$.
   - The arrows are all maps $f\colon c \to c'$ such that c, c' are as above.

   The functor $i\colon \mathrm{Eq}(\alpha,\beta)$ maps c to c and f to f.

Lemma 10.2.4. *If $(\mathscr{C}_i, C_i)$ is a family of categories with bases of computability indexed by some set I, then the product $\prod_{i\in I}(\mathscr{C}_i, C_i)$ is given by*

$$\prod_{i\in I}(\mathscr{C}_i, C_i) = \Big(\prod_{i\in I}\mathscr{C}_i, \prod_{i\in I} C_i\Big) \text{ where } \Big(\prod_{i\in I} C_i\Big)(c_i)_{i\in I} = \prod_{i\in I} C_i(c_i).$$

Proof: It is immediate that the base contains all identities, as $1_{c_i} \in C_i(c_i)$ for all c_i. The closure under composition follows similarly, and for the pullback we simply remark that the pullback in $\prod_{i\in I}\mathscr{C}_i$ is simply product of the pullbacks in the respective $\mathscr{C}_i$. That this constitutes a product in the 2-limit sense is immediate. Q.E.D.

Lemma 10.2.5. *Let $(\mathscr{C}, C), (\mathscr{D}, D)$ be two categories with bases of computability and $F, G\colon (\mathscr{C}, C) \to (\mathscr{D}, D)$ be two computability transfers. Then the inserter of F, G is given by $\big(\mathsf{DiAlg}(F,G), \mathsf{DiAlg}(F,G;C)\big)$ where $\mathsf{DiAlg}(F,G)$ is the category of dialgebras on F, G and $\mathsf{DiAlg}(F,G;C)$ is defined via*

$$\mathsf{DiAlg}(F,G;C)\big(\alpha\colon F(c) \to G(c)\big) := \bigcup_{\beta\in\mathsf{DiAlg}(F,G)} \{f \in \mathsf{DiAlg}(F,G)(\beta,\alpha) | f \in C(c)\}.$$

The computability transfer $X\colon \big(\mathsf{DiAlg}(F,G), \mathsf{DiAlg}(F,G;C)\big) \to (\mathscr{C}, C)$ *takes* $\alpha\colon F(c) \to G(c)$ *to* c *and* $F(f)$ *to* f. *The natural transformation* $F \Rightarrow G$ *is the same as for the inserter in the normal categorical setting.*

Proof: It is immediate from the definition of $\mathsf{DiAlg}(F,G;C)$ that this is closed under composition and identities. Hence it remains to check the pullback condition. For this let $\big(F(f), G(f)\big)\colon \beta \to \alpha$ and $F(g), G(g)\colon \gamma \to \alpha$ be given. To compute the pullback we first note that we obtain a pullback square

$$\begin{tikzcd}
f^*(c'') \arrow[r, "f^*g"] \arrow[d, "g^*f"'] \arrow[dr, phantom, "\lrcorner", very near start] & c \arrow[d, "f"] \\
c'' \arrow[r, "g"'] & c'
\end{tikzcd}$$

in $\mathscr{C}$. As F, G are computability transfers they preserve pullbacks of arrows in C and we thus obtain the cube

$$\begin{tikzcd}
F\big(f^*(c'')\big) \arrow[rrr, "F(f^*g)"] \arrow[dr, dotted, "\xi"] \arrow[dd, "F(g^*f)"'] & & & F(c) \arrow[dr, "\beta"] \arrow[dd, "F(f)" near start] & \\
& G\big((f^*(c'')\big) \arrow[rrr, "G(f^*g)" near start] \arrow[dd, "G(g^*f)" near end] \arrow[ddrrr, phantom, "\lrcorner", very near start] & & & G(c) \arrow[dd, "G(f)"] \\
F(c'') \arrow[rrr, "F(g)" near start] \arrow[dr, "\gamma"'] & & & F(c') \arrow[dr, "\alpha"] & \\
& G(c'') \arrow[rrr, "G(g)"] & & & G(c').
\end{tikzcd}$$

The arrow ξ is obtained by the universal property of $G\big((f^*(c'')\big)$ applied to the arrows $G(f) \circ \beta \circ F(f^*g)$ and $G(g) \circ \gamma \circ F(g^*f)$, which are equal, as a straightforward computation shows. The pullback of $\big(F(f), G(f)\big)$ along $\big(F(g), G(g)\big)$ is given by the vertical arrows on the left face of the above cube, which are in $\mathsf{DiAlg}(F,G;C)(\gamma)$, as $g^*f \in C(c'')$.

It is immediate from the definition that X constitutes a computability transfer. To check the universal property let another category with base of computability $(\mathscr{E}, E)$ together with a computability transfer $H\colon (\mathscr{E}, E) \to (\mathscr{C}, C)$ and a natural transformation $\beta\colon F \circ H \Rightarrow G \circ H$ be given. Then the functor $J\colon \mathscr{E} \to \mathsf{DiAlg}(F,G)$ takes $e \in E$ to $\beta_e\colon F\big(H(e)\big) \to G\big(H(e)\big)$ and $h\colon e \to e'$ to

$$\begin{tikzcd}
F\big(H(e)\big) \arrow[r, "\beta_e"] \arrow[d, "F(H(h))"'] & G\big(H(e)\big) \arrow[d, "G(H(h))"] \\
F\big(H(e')\big) \arrow[r, "\beta_{e'}"'] & G\big(H(e')\big).
\end{tikzcd}$$

It is immediate that this is the only functor such that $X \circ J = H$.

The 2-dimensional aspect can be seen immediately as well. Suppose we are given two computability transfers $H, H'\colon (\mathscr{E}, E) \to (\mathscr{C}, C)$ and natural transformations $\beta\colon F \circ H \Rightarrow G \circ H, \beta'\colon F \circ H' \to G \circ H'$, as well as a natural transformation $\xi\colon H \to H'$ such that

$$\begin{tikzcd}
F \circ H \arrow[r, Rightarrow, "\beta"] \arrow[d, Rightarrow, "F \bullet \xi"'] & G \circ H \arrow[d, Rightarrow, "G \bullet \xi"] \\
F \circ H' \arrow[r, Rightarrow, "\beta'"'] & G \circ H'
\end{tikzcd}$$

commutes. Let J, J' be the two functors as in the 1-dimensional part of the universal property. Then we can define $\gamma\colon J \to J'$ through $\gamma_e := \xi_e$. That this is constitutes a natural transformation follows from the commutativity of

$$\begin{tikzcd}
F\big(H(e)\big) \arrow[r, "\beta_e"] \arrow[d, "F(\xi_e)"'] & G\big(H(e)\big) \arrow[d, "G(\xi_e)"] \\
F\big(H'(e)\big) \arrow[r, "\beta'_e"'] & G\big(H'(e)\big),
\end{tikzcd}$$

which itself stems from the commutativity of the preceding square. A straightforward computation shows $I \bullet \gamma = \xi$, and it is immediate that γ is the only natural transformation with this property. Q.E.D.

Lemma 10.2.6. *Let $F, G\colon (\mathscr{C}, C) \to (\mathscr{D}, D)$ be two computability transfers as in the preceding lemma and $\alpha, \beta\colon F \Rightarrow G$ be two natural transformations. Then the equifier $\mathrm{Eq}(\alpha, \beta)$ of α and β is given by the category $\mathrm{Eq}(\alpha, \beta)$ whose objects are those $c \in \mathscr{C}$ such that $\alpha_c = \beta_c$ and whose morphisms are all morphisms in $\mathscr{C}$ between those c.*

The computability base on $\mathrm{Eq}(\alpha, \beta)$ consists of all arrows in C that are in $\mathrm{Eq}(\alpha, \beta)$, and the computability transfer $I\colon \mathrm{Eq}(\alpha, \beta) \to (\mathscr{C}, C)$ is simply the inclusion.

Proof: It is immediate that $\alpha \bullet I = \beta \bullet I$ from the definition of $\mathrm{Eq}(\alpha, \beta)$. The fact that the base is closed under identity and composition is again immediate. To see that it is closed under arbitrary pullbacks, let arrows $f\colon c \to c'$ and $g\colon c'' \to c'$ in $\mathrm{Eq}(\alpha, \beta)$—such that $f \in C(c')$—be given. If we compute the pullback, we have to show that $g^*f\colon f^*(c'') \to c''$ lies in the restricted base—that is we have to show that $f^*(c'') \in \mathrm{Eq}(\alpha, \beta)$. For this consider the cube

$$\begin{tikzcd}
F\big(F^*(c'')\big) \arrow[rr, "F(f^*g)"] \arrow[dd, "F(g^*f)"'] \arrow[dr, shift left, "\beta_{G(f^*(c''))}"] \arrow[dr, shift right, "\alpha_{F(f^*(c''))}"'] & & F(c) \arrow[dr, "\alpha_c"] \arrow[dd, "F(f)" near end] & \\
& G\big(f^*(c'')\big) \arrow[rr, crossing over, "G(f^*g)" near start] \arrow[dd, "G(g^*f)" near end] \arrow[dr, phantom, "\lrcorner", very near start] & & G(c) \arrow[dd, "G(f)"] \\
F(c'') \arrow[rr, "F(g)" near start] \arrow[dr, "\alpha_{c''}"'] & & F(c') \arrow[dr, "\alpha_{c'}"] & \\
& G(c'') \arrow[rr, "G(g)"] & & G(c').
\end{tikzcd}$$

As both $\alpha_{G(f^*(c''))}$ and $\beta_{G(f^*(c''))}$ make the entirety of the cube commute, we get from the pullback property of $G\big(f^*(c'')\big)$ that they are equal.

Furthermore I is a computability transfer, as is immediate from the definition of the base on $\mathrm{Eq}(\alpha, \beta)$. To verify the universal property of the equifier, let another $(\mathscr{E}, E)$ together with $H\colon (\mathscr{E}, E) \to (\mathscr{C}, C)$, such that $\alpha \bullet H = \beta \bullet H$, be given. Then we can define $J\colon \mathscr{E} \to \mathrm{Eq}(\alpha, \beta)$ via $J(e) := H(e)$, $J(f) = H(f)$. This is possible as $\alpha_{H(e)} = \beta_{H(e)}$ by assumption. It is immediate that this J is a computability transfer, as H is. Furthermore H is unique, which follows from the definition and the condition $I \circ J = H$.

For the 2-dimensional aspect suppose we are given $H, H'\colon (\mathscr{E}, E) \to (\mathscr{C}, C)$ as above and $\gamma\colon H \Rightarrow H'$ be given. We then define $\omega\colon J \Rightarrow J'$—where J and J' are defined from H and H' as above, respectively—by $\omega_e := \gamma_e$. It is immediate that $I \bullet \omega = \gamma$, and that ω is the only natural transformation with this property. Q.E.D.

Theorem 10.2.7. $\mathsf{CatBaseComp}$ *has all limits of pie weight.*

Proof: This follows immediately from [53, Theorem 2.2], as CatBaseComp has all products, equifiers and inserters by the preceding lemmata. Q.E.D.

Remark 10.2.8. The above theorem entails that CatBaseComp has 2-limits of many shapes, including pseudo-pullbacks and powers. If we want to have regular equalisers, however, a problem arises. Considering the equaliser of two computability transfers, we know that if a computability base on the equaliser of F, G as functors exists, it must be closed under pullbacks. Taking

$$\begin{tikzcd}
f^*(b) \arrow[r, "i^*f"] \arrow[d, hook, "f^*i"] \arrow[dr, phantom, "\lrcorner", very near start] & b \arrow[d, hook, "i"] \\
a \arrow[r, "f"] & c
\end{tikzcd} \tag{21}$$

in $\mathscr{C}$, mapped into $\mathscr{D}$ with F, G respectively, we only obtain the diagram

$$\begin{tikzcd}
G\big(f^*(b)\big) \arrow[drr, "G(i^*f)"] \arrow[ddr, "G(f^*i)"'] \arrow[dr, phantom, "\lrcorner", very near start] & & \\
& F\big(f^*(b)\big) \arrow[r, "F(i^*f)"] \arrow[d, hook, "F(f^*i)"'] \arrow[dr, phantom, "\lrcorner", very near start] & F(b) \arrow[d, hook, "F(i)"] \\
& F(a) \arrow[r, "f"] & c
\end{tikzcd}$$

in $\mathscr{D}$ where both the inner and outer square are pullback squares. Thus we only obtain an isomorphism, not an identity between $G\big(f^*(b)\big)$ and $F\big(f^*(b)\big)$, hence this pullback does not lie in the equaliser of F and G and thus the base would not be closed under pullbacks

To remedy this fact we would have to endow the base of computability with choices of pullbacks, that is to require the stronger condition

$(\mathtt{Base}_2{}^{\mathsf{s}})$ Given $a, b \in \mathscr{C}_0$ and $i\colon s \to a, j\colon t \to b$ such that $i \in C(a), j \in C(b)$ and $f\colon s \to b$, we are given a *specific choice* of a pullback $s \times_b t$ and the square in

$$\begin{tikzcd}
& s \times_b t \arrow[r, "j^*f"] \arrow[d, hook, "f^*j"] \arrow[dl, "i\circ f^*j"'] \arrow[dr, phantom, "\lrcorner", very near start] & t \arrow[d, hook, "j"] \\
a & s \arrow[l, hook', "i"] \arrow[r, "f"] & b
\end{tikzcd}$$

is a pullback square.

instead of just $(\mathtt{Base}_2)$. In the following we will make the above remark precise in the following way: not only can the equaliser in the category of locally small categories not be endowed with a base of computability making it an equaliser in CatBaseComp, but in general no equaliser exists.

Lemma 10.2.9. *Let $F, G\colon (\mathscr{C}, C) \to (\mathscr{D}, D)$ be computability transfers. If $I\colon (\mathscr{B}, B) \to (\mathscr{C}, C)$ is an equaliser in* CatBaseComp*, then for any cospan $m \in C(c), f\colon c'' \to c$, that is*

$$\begin{tikzcd}
c'' \arrow[r, "f"] & c & c' \arrow[l, "m"']
\end{tikzcd}$$

for an arbitrary object c of $\mathscr{C}$, such that $F(m) = G(m), F(f) = G(f)$, there exists a unique cospan k, l in $\mathscr{B}$ with $I(k) = m, I(l) = f$ such that k lies in the base of computability B.

Proof: Let S be the category with objects $0, 1, 2$ and non-identity arrows $\leq_0 \colon 0 \to 2, \leq_1 \colon 1 \to 2$. We define a base of computability Q on S by setting $Q(0) = \{1_0\}, Q(2) = \{\leq_0, 1_2\}, Q(1) = \{1_1\}$. Considering a cospan $m \colon c' \to c, f \colon c'' \to c$ in $\mathscr{C}$ with $F(m) = G(m), F(f) = G(f)$, we define the computability transfer $H \colon (S, Q) \to (\mathscr{C}, C)$ that sends $\leq_0$ to m and $\leq_1$ to f.

Obviously $F \circ H = G \circ H$, so we obtain a unique $J_H \colon (S, Q) \to (\mathscr{B}, B)$ such that $I \circ J_H = H$. It is then immediate that $J_H(\leq_0)$ is the unique arrow k in $\mathscr{B}$ with $I(k) = m$ and $J_H(\leq_1)$ is the unique arrow l with $I(l) = f$. It is also immediate that k lies $B(J_H(2))$, as $\leq_0 \in Q(2)$ and J_H is a computability transfer. Q.E.D.

Proposition 10.2.10. *Considering the categories $\mathscr{C}, \mathscr{D}$, generated by the graphs*

$$\begin{tikzcd}
0 \arrow[r, "f"] \arrow[d, "g"] & 1 \arrow[d, "h"] \\
2 \arrow[r, "i"] & 3
\end{tikzcd} \quad \textit{and} \quad \begin{tikzcd}
0' \arrow[rrd, "k"] \arrow[ddr, "l"'] \arrow[dr, shift left, "t"] & & \\
 & 0 \arrow[ul, shift left, "s"] \arrow[r, "f"] \arrow[d, "g"] & 1 \arrow[d, "h"] \\
 & 2 \arrow[r, "i"] & 3
\end{tikzcd}$$

respectively, we can endow $\mathscr{C}$ with a base of computability C by setting

$$C(0) = \{1_0\}, \quad C(1) = \{1_1\}, \quad C(2) = \{1_2, g\}, \quad C(3) = \{1_3, h\}$$

and $\mathscr{D}$ with a base of computability D by setting

$$C(0') = \{1_{0'}\}, \quad C(0) = \{1_0\}, \quad C(1) = \{1_1\}, \quad C(2) = \{1_2, g, l\}, \quad C(3) = \{1_3, h\}.$$

Then the computability transfers F, G mapping $1, 2, 3$ in $\mathscr{C}$—and the corresponding morphisms between them—to their counterparts in $\mathscr{D}$, but

$$F(0) = 0, F(f) = f, F(f) = g, \quad G(0) = 0', G(f) = k, G(g) = l,$$

have no equaliser in $\mathsf{CatBaseComp}$.

Proof: We begin by confirming that C, D constitute bases of computability. To this end we first show that the only square in $\mathscr{C}$ is a pullback square and that both squares in $\mathscr{D}$ consisting of $0, 1, 2, 3$ and $0', 1, 2, 3$ are pullback squares. The first claim is immediate, as the square commutes and the only object with arrows to both 1 and 2 is 0, so the universal property is trivially true. For $\mathscr{D}$ we observe that the square $0, 1, 2, 3$ commutes, and the only object besides 0 with arrows to 1 and 2 is $0'$. The square $0', 1, 2, 3$ commutes and the unique mediating arrow is t, thus the universal property is fulfilled. An analogue argument shows that $0', 1, 2, 3$ is a pullback square.

Thus to confirm that C is a base of computability on $\mathscr{C}$ it suffices to observe that $g \in B(2)$, as g is the pullback of h along i. Similarly we observe that both l and g are pullbacks of h along i, and thus in $D(2)$.

It is then straightforward to check that F, G are computability transfers. If there was an equaliser $I \colon (\mathscr{E}, E) \to (\mathscr{C}, C)$ of F, G in $\mathsf{CatBaseComp}$, by the preceding Lemma we obtain a cospan $e'' \xrightarrow{k} e \xleftarrow{k} e'$ in $\mathscr{K}$ that maps to the cospan $2 \xrightarrow{i} 3 \xleftarrow{h} 1$ and fulfils $k \in E(e)$. Then k is in the base E, so the pullback

$$\begin{tikzcd}
e'^*(e'') \arrow[r] \arrow[d, "l^*k"'] \arrow[dr, phantom, "\lrcorner", very near start] & e' \arrow[d, "k"] \\
e'' \arrow[r, "l"'] & e
\end{tikzcd}$$

must exist. As I must be pullback-preserving we know $I\big(l^*k\big) = g$, but $G(g) \neq F(g)$, a contradiction to $(\mathscr{E}, E)$ being an equaliser. Q.E.D.

10.3 CatBaseComp and fibrations

We will show that Grothendieck fibrations can be used both to lift bases of computability along them, and, that in restricting ourselves to fibrations, we regain some pullbacks, which do not in general exist (as the previous section shows).

Proposition 10.3.1. *Let $F\colon \mathscr{E} \to \mathscr{B}$ be a Grothendieck fibration where $\mathscr{E}$ has all pullbacks and let B be a base of computability on $\mathscr{B}$. Then we can define a base of computability B_F on $\mathscr{E}$ by setting*

$$B_F(e) = \{i \in \mathrm{Mon}(-, e) \mid i \textit{ cartesian lift of } j \in B\big(F(e)\big)\}.$$

for every $e \in \mathscr{E}_0$.

Proof. Assume we are given $\mathscr{E}, \mathscr{B}, F$ as in the proposition. We show that B_F contains identities, is closed under composition and pullbacks.

To see that it contains all identities let $e \in \mathscr{E}_0$. We have to show that 1_e is a cartesian lift of $1_{F(e)}$. If we are given $g\colon d \to e$ in $\mathscr{E}$ and we have $h\colon F(d) \to F(e)$ such that

$$\begin{tikzcd}
F(d) \arrow[d, "h"] \arrow[dr, "F(g)"] & \\
F(e) \arrow[r, "1_{F(e)}"'] & F(e)
\end{tikzcd}$$

commutes, we can immediately deduce that $h = F(g)$ and the only possible arrow making the triangle in $\mathscr{E}$ commute is g itself. $F(g) = h$ by the above. This shows the desired claim as obviously $F(1_e) = 1_{F(e)}$.

To show that B_F is closed under composition we simply remark that the composition of cartesian morphisms is cartesian.

To see that it is closed under pullback, let $i\colon e' \to e$ in $B_F(e)$ and $f\colon e'' \to e$ be given. We can then form the pullback

$$\begin{tikzcd}
f^*(e') \arrow[r, "i^*f"] \arrow[d, hook, "f^*i"'] \arrow[dr, phantom, "\lrcorner", very near start] & e' \arrow[d, hook, "i"] \\
e'' \arrow[r, "f"'] & e
\end{tikzcd}$$

in $\mathscr{E}$. It remains to show that f^*i is a cartesian lift of $F(f^*i)$. So let $g\colon d \to e''$ and $h\colon F(d) \to F\big(f^*(e')\big)$ be given such that

$$\begin{tikzcd}
F(d) \arrow[d, "h"'] \arrow[dr, "F(g)"] & \\
F\big(f^*(e')\big) \arrow[r, "F(f^*i)"'] & F(e'')
\end{tikzcd}$$

commutes. We can then fit this commutative triangle to our previous square (after having applied F) to obtain the commutative diagram

$$\begin{tikzcd}
F(d) \arrow[r, "h"] \arrow[dr, rounded corners, "F(g)"'] & F\big(f^*(e')\big) \arrow[r, "F(i^*f)"] \arrow[d, hook, "F(f^*i)"'] & F(e') \arrow[d, hook, "F(i)"] \\
& F(e'') \arrow[r, "F(f)"'] & F(e).
\end{tikzcd}$$

Thus by considering $F(i^*f) \circ h$ and $F(f) \circ F(g)$ we can use that i is a cartesian lift of $F(i)$ to obtain a morphism $p\colon d \to e'$ such that the outer square in

$$\begin{tikzcd}
d \arrow[rrd, rounded corners, to path={-| (\tikztotarget)}, "p"] \arrow[rdd, rounded corners, to path={|- (\tikztotarget)}, "g"'] & & \\
& f^*(e') \arrow[r, "i^*f"] \arrow[d, hook, "f^*i"'] \arrow[dr, phantom, "\lrcorner", very near start] & e' \arrow[d, hook, "i"] \\
& e'' \arrow[r, "f"'] & e
\end{tikzcd}$$

commutes. But as the inner square is a pullback square we obtain a unique $q\colon d \to f^*(e')$ such that

$$\begin{tikzcd}
d \arrow[rrd, rounded corners, to path={-| (\tikztotarget)}, "p"] \arrow[rdd, rounded corners, to path={|- (\tikztotarget)}, "g"'] \arrow[rd, dotted, "q"] & & \\
& f^*(e') \arrow[r, "i^*f"] \arrow[d, hook, "f^*i"'] \arrow[dr, phantom, "\lrcorner", very near start] & e' \arrow[d, hook, "i"] \\
& e'' \arrow[r, "f"'] & e
\end{tikzcd}$$

commutes. This q is the desired map making the triangle in $\mathscr{E}$ commute. So f^*i is a cartesian lift of $F(f^*i) \in B\big(F(e'')\big)$ and is thus in $B_F(e'')$ as desired. Q.E.D.

Actually the fact that $i \in \mathrm{Mon}(-, e)$ does not have to be checked, as it is automatic.

Proposition 10.3.2. *Let $p\colon \mathscr{E} \to \mathscr{B}$ be a fibration. If $f\colon b' \to b$ is a monomorphism, and $g\colon e' \to e$ is p-cartesian for f and e, then g is a monomorphism.*

Proof. Assume we are given $k, l\colon e'' \to e'$ such that $g \circ k = g \circ l$. Then we obtain the commutative triangles

$$\begin{tikzcd}
e'' \arrow[d, "?"] \arrow[dr, "g\circ k"] & \\
e' \arrow[r, "g"'] & e
\end{tikzcd}
\quad \text{and} \quad
\begin{tikzcd}
b'' \arrow[d, "p(k)"'] \arrow[dr, "f\circ p(k)"] & \\
b' \arrow[r, "f"] & b.
\end{tikzcd}$$

As both k, l for ? make the left triangle commute, we get by uniqueness—from g being p-cartesian—that $k = l$. Q.E.D.

Corollary 10.3.3. *If we have a pullback preserving fibration $F\colon \mathscr{E} \to \mathscr{B}$ and a base of computability B on $\mathscr{B}$ as above, then the fibrationis a computability transfer with respect to the lifted base B_F.*

Proof. From the definition of B_F it is immediate that it maps monomorphisms in B_F to monomorphisms in B. It also preserves pullbacks by assumption, so it is a computability transfer. Q.E.D.

Theorem 10.3.4. *Let $(\mathscr{B}, B), (\mathscr{C}, C)$ be categories with bases of computability, $\mathscr{E}$ be a category such that $p\colon (\mathscr{E}, E) \to (\mathscr{B}, B)$ is a fibrations, and $F\colon (\mathscr{C}, C) \to (\mathscr{B}, B)$ be computability transfers. Then if we are given a pullback square*

$$\begin{tikzcd}
p^{-1}(\mathscr{C}) \arrow[r, "p^*F"] \arrow[d, "F^*p"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathscr{E} \arrow[d, "p"] \\
\mathscr{C} \arrow[r, "F"] & \mathscr{B}
\end{tikzcd}$$

of categories, we obtain that

$$\begin{tikzcd}
\big(p^{-1}(\mathscr{C}), C_{F^*p}\big) \arrow[r, "p^*F"] \arrow[d, "F^*p"'] \arrow[dr, phantom, "\lrcorner", very near start] & \big(\mathscr{E}, B_p\big) \arrow[d] \\
(\mathscr{C}, C) \arrow[r, "F"] & (\mathscr{B}, B)
\end{tikzcd}$$

is a pullback square in CatBaseComp*.*

Proof. We show the universal property of the pullback. If we are given another commutative square

$$\begin{tikzcd}
(\mathscr{G}, G) \arrow[r, "H"] \arrow[d, "G"'] & (\mathscr{E}, E) \arrow[d, "p"] \\
(\mathscr{C}, C) \arrow[r, "F"] & (\mathscr{B}, B)
\end{tikzcd}$$

we obtain—in **Cat**—a unique functor $\langle G, H\rangle \colon \mathscr{G} \to p^{-1}(\mathscr{C})$ making

$$\begin{tikzcd}
 & \mathscr{G} \arrow[dl, "G"'] \arrow[dr, "H"] \arrow[d, dashed, "\langle G,H\rangle"] & \\
\mathscr{C} & p^{-1}(\mathscr{C}) \arrow[l] \arrow[r] & \mathscr{E}
\end{tikzcd}$$

commute. We show that $\langle G, H\rangle$ is also a computability transfer. For this let $i \in G(x)$ be given. It then suffices to show that $\langle G, H\rangle(i)$ is F^*p-cartesian for $G(i)$ and $\langle G, H\rangle(x)$. If we are given another ℓ with codomain $\langle G, H\rangle(x)$ and k such that $G(i) \circ k = p^*F(\ell)$, we can apply F to obtain the same triangle in $\mathscr{B}$, that is we have

$$\begin{tikzcd}
- \arrow[dr, "\ell"] & \\
- \arrow[r, "\langle G,H\rangle i"'] & \langle G, H\rangle(x)
\end{tikzcd}
\quad \text{and} \quad
\begin{tikzcd}
- \arrow[d, "k"'] \arrow[dr, "F^*p(\ell)"] & \\
- \arrow[r, "G(i)"'] & G(x)
\end{tikzcd}
\quad \text{and} \quad
\begin{tikzcd}
- \arrow[d, "F(k)"'] \arrow[dr, "(G\circ F^*p)(\ell)"] & \\
- \arrow[r, "(F\circ G)(i)"'] & F\big(G(x)\big)
\end{tikzcd}$$

in $p^{-1}(\mathscr{C}), \mathscr{C}$ and $\mathscr{B}$ respectively. As p is a fibration we can lift the last triangle to a triangle

$$\begin{tikzcd}
- \arrow[d, "f"'] \arrow[dr, "p^*F(\ell)"] & \\
- \arrow[r, "H(i)"'] & H(x)
\end{tikzcd}$$

as H is a computability transfer and thus $H(i)$ is cartesian for $F(G(i)) = p(H(i))$.

Next we observe that $\langle G, H\rangle i$ is the only arrow in $p^{-1}(\mathscr{C})$ that is mapped to $H(i)$ by p^*F and to $G(i)$ by F^*p, as otherwise we would not have a unique fill for

$$\begin{tikzcd}
 & {} \arrow[dl, "\mapsto G(i)"'] \arrow[dr, "\mapsto H(i)"] \arrow[d, dotted, "?"] & \\
\mathscr{C} & p^{-1}(\mathscr{C}) \arrow[l, "F^*p"] \arrow[r, "p^*F"'] & \mathscr{E}.
\end{tikzcd}$$

A similar computation shows that ℓ is also uniquely determined by its images under p^*F, F^*p. So by considering $[2]$ and the maps

$$\begin{tikzcd}
- \arrow[d, "k"'] \arrow[dr, "F^*p(\ell)"] & \\
- \arrow[r, "G(i)"'] & G(x)
\end{tikzcd}
\quad \longleftarrow\!\shortmid \quad
\begin{tikzcd}
0 \arrow[d] \arrow[dr] & \\
1 \arrow[r] & 2
\end{tikzcd}
\quad \longmapsto \quad
\begin{tikzcd}
- \arrow[d, "f"'] \arrow[dr, "p^*F(\ell)"] & \\
- \arrow[r, "H(i)"'] & H(x)
\end{tikzcd}$$

from the pullback property we get the desired commutative triangle

$$\begin{tikzcd}
- \arrow[d, "g"] \arrow[dr, "\ell"] & \\
- \arrow[r, "\langle G,H\rangle(i)"'] & \langle G,H\rangle(x)
\end{tikzcd}$$

(where g is unique from the uniqueness of the pullback) showing that $\langle G,H\rangle i$ is indeed $F^{*}p$-cartesian for $G(i)$. Q.E.D.

10.4 Relation to comprehension categories

It is immediate that any category $\mathscr{C}$ with a base of computability C can be mapped to a comprehension category $P\colon \mathscr{E} \to \mathscr{C}^{[1]}$, where $\mathscr{E}$ has as objects all maps in C, and an arrow from $f \to g$ in $\mathscr{E}$ is a cartesian square

$$\begin{tikzcd}
- \arrow[r] \arrow[d, "f"'] \arrow[dr, phantom, "\lrcorner", very near start] & - \arrow[d, "g"] \\
- \arrow[r] & -
\end{tikzcd}$$

in $\mathscr{C}$. The functor P is then simply the inclusion of $\mathscr{E}$ as a subcategory of $\mathscr{C}^{[1]}$. From this definition it is obvious that p sends an arrow $f \in \mathscr{E}$ to its codomain.

An immediately visible specialisation from general comprehension categories is that in this case all arrows $P(e)$ are monic, which is not always the case in an arbitrary comprehension category, just observe that $\mathrm{id}_{\mathscr{C}^{[1]}}\colon \mathscr{C}^{[1]} \to \mathscr{C}^{[1]}$ is a comprehension category if $\mathscr{C}$ has all pullbacks. Thus it is sensible to restrict oneself to the sub-2-category $\mathsf{ComprC}_{\mathrm{Mon}}$ of all comprehension categories where the $P(e)$ are monic, with the same 1-maps and 2-maps. The assignment routine described above can be extended to a 2-functor $\mathsf{CatBaseComp} \to \mathsf{ComprC}_{\mathrm{Mon}}$ in the following way:

Proposition 10.4.1. *The rule $(\mathscr{C}, C) \mapsto P\colon \mathscr{E} \to \mathscr{C}^{[1]}$ from above can be extended to a 2-functor $P_{\text{-}}$ via*

$$F\colon (\mathscr{C}, C) \to (\mathscr{D}, D) \mapsto \begin{tikzcd}
\mathscr{E}_C \arrow[r, "F^{[1]}"] \arrow[d, "P_C"] & \mathscr{E}_D \arrow[d, "P_D"] \\
\mathscr{C} \arrow[r, "F"'] & \mathscr{D}
\end{tikzcd}$$

and $\eta\colon F \Rightarrow G \mapsto \eta^{[1]}$.

Proof: Immediate, as obviously $(G \circ F)^{[1]} = G^{[1]} \circ F^{[1]}$ from the definition of $\text{-}^{[1]}$, analogously for $\eta^{[1]}$. Q.E.D.

For the reverse direction $\mathsf{ComprC}_{\mathrm{Mon}} \to \mathsf{CatBaseComp}$ we need a non-trivial construction, as another difference between these comprehension categories arising from (categories with) a base of computability and ordinary comprehension categories arise: In the former we can always represent composition through an element of $\mathscr{E}$, that is the following condition holds:

$$\begin{aligned}
&\text{if } P(e) \text{ can be precomposed with } P(e'),\\
&\text{there exists an object } e'' \text{ of } \mathscr{E} \text{ such that } P(e'') = P(e) \circ P(e').
\end{aligned} \tag{22}$$

This might seem surprising at first, as if we are given

$$P_0(e) \xrightarrow{P(e)} p(e) \xrightarrow{P(e')} p(e')$$

it might seem that we can use some p-cartesian lift to obtain e'', but such e'' would need to fulfil both $P_0(e'') = P_0(e)$ and $p(e'') = p(e')$, so if it stemmed from a lift, that lift would have to be taken using an automorphism on $p(e')$,

In fact, it is possible to give an example of a comprehension category where condition (22) fails:

Example 10.4.2. Consider the groupoid $\mathscr{C}$ obtained from the graph

$$\begin{tikzcd} c \arrow[r, bend left, "f"] & c' \arrow[l, bend left, "f^{-1}"] \arrow[r, bend left, "g"] & c'' \arrow[l, bend left, "g^{-1}"] \end{tikzcd}$$

and let $\mathscr{E}$ be another copy of the same groupoid. For clarity the objects of this second copy will be called $0, 1, 2$ and the arrows will simply be denoted as $n \mapsto m$ for $n, m \in \{0, 1, 2\}$. Let P be the functor sending 0 to 1_c, 1 to f and 2 to g. Its action on arrows generated by

$$(0 \to 1) \mapsto \begin{tikzcd} c \arrow[r, "1_c"] \arrow[d, "1_c"'] & c \arrow[d, "f"] \\ c \arrow[r, "f"] & c' \end{tikzcd} \quad \text{and} \quad (1 \to 2) \mapsto \begin{tikzcd} c \arrow[r, "f"] \arrow[d, "f"'] & c' \arrow[d, "g"] \\ c' \arrow[r, "g"] & c''. \end{tikzcd}$$

$$\text{as well as } (1 \to 0) \mapsto \begin{tikzcd} c' \arrow[r, "f"] \arrow[d, "f"'] & c \arrow[d, "1_c"] \\ c \arrow[r, "1_c"] & c \end{tikzcd} \quad \text{and} \quad (2 \to 1) \mapsto \begin{tikzcd} c' \arrow[r, "f^{-1}"] \arrow[d, "g"'] & c \arrow[d, "f"] \\ c'' \arrow[r, "g^{-1}"] & c', \end{tikzcd}$$

composites are simply obtained from pasting the squares. Observe that all these squares are pullback squares, so for this to be a comprehension category only the second condition of p being a fibration has to be checked. However, the lifts can be explicitly computed:

$$\begin{aligned} \overline{f^{-1}}(0) &= 1_0, & \overline{f^{-1} \circ g^{-1}}(0) &= (2 \to 0), \\ \overline{f}(1) &= (0 \to 1), & \overline{g^{-1}}(1) &= (1 \to 0), \\ \overline{g}(2) &= (1 \to 2), & \overline{g^{-1} \circ f^{-1}}(2) &= (0 \to 2) \end{aligned}$$

and all identities lift to identities. To see that this comprehension category violates (22) just observe that no object e in $\mathscr{E}$ exists such that $P(e) = g \circ f = P(2) \circ P(1)$.

Thus to map an arbitrary comprehension category in $\mathsf{ComprC}_{\mathrm{Mon}}$ to $\mathsf{CatBaseComp}$ we first need to manually add objects such that (22) is fulfilled.

10.4.3 Some notation regarding lists. In the following we will

1. denote by $P(\mathscr{E})$—for $P\colon \mathscr{E}$—the collection of all arrows $P(e)$ where e is an object of $\mathscr{E}$,
2. denote by M^+ the set of all finite lists of length ≥ 1 for an arbitrary set M,
3. denote by $\ell \frown \ell'$ the concatenation of ℓ, ℓ', that is
$$\ell \frown \ell' = \ell_1 \cdots \ell_{|\ell|} \ell'_1 \cdots \ell'_{|\ell'|},$$
4. call a list ℓ of arrows in a category $\mathscr{E}$ *composable* if
$$\forall_{0 \leq i < |\ell|} \operatorname{cod}(\ell_i) = \operatorname{dom}(\ell_{i+1}) \text{ and}$$
5. denote by ℓ° the arrow $\ell_{|\ell|} \circ \cdots \circ \ell_1$ for a list ℓ of composable arrows.

Definition 10.4.4. For a given comprehension category $P\colon \mathscr{E} \to \mathscr{B}^{[1]}$ in $\mathsf{ComprC}_{\mathrm{Mon}}$ we define a base of computability B_P on $\mathscr{B}$ where

$$B_P(b) = \big\{\ell^\circ \in P(\mathscr{E})^+ \mid \ell \text{ composable} \wedge \operatorname{cod}(\ell_{|\ell|}) = b\big\} \cup \{1_b\}.$$

Proposition 10.4.5. *The above definition yields a well-formed base of computability for such a* $P \in \mathsf{ComprC}_{\mathrm{Mon}}$.

Proof: By definition B_P contains all identities. To see that it is closed under composition consider a two list $\ell \in B_P(b)$ with $\operatorname{dom}(\ell_1) = a$, and $\ell' \in B_P(a)$. It is then immediate that $(\ell' \frown \ell)^\circ \in B_P(b)$. For the pullback property, assume we are given $\ell^\circ \in B_P(b)$ and $f\colon b' \to b$. To see that the pullback

$$\begin{tikzcd}
\big((\ell)^\circ\big)^{-1}(b') \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & b' \arrow[d, "f"] \\
\operatorname{dom}(\ell_1) \arrow[r, "\ell^\circ"'] & b
\end{tikzcd}$$

exists, first observe that we can find objects $e_1, \dots, e_{|\ell|}$ of $\mathscr{E}$ such that $\ell_i = P(e_i)$ for all i. Thus we start by considering the pullback

$$\begin{tikzcd}
P_0\big(f^*(e_{|\ell|})\big) \arrow[r, "P(f^*(e_{|\ell|}))"] \arrow[d, "P(\overline{f}(e_{|\ell|}))"'] \arrow[dr, phantom, "\lrcorner", very near start] & b' \arrow[d, "f"] \\
P_0(e_{|\ell|}) \arrow[r, "P(e_{|\ell|})"'] & b.
\end{tikzcd}$$

Then by setting $f_1 := P\big(\overline{f}(e_{|\ell|})\big)$ and recursively $f_{i+1} := P\big(\overline{f_i}(e_{|\ell|-i})\big)$ we obtain the pullback diagrams

$$\begin{tikzcd}
P_0(f_i^*(e_{|\ell|-i}) \arrow[r, "P\left(f_i^*(e_{|\ell|-i})\right)"] \arrow[d, "f_{i+1}"'] \arrow[dr, phantom, "\lrcorner", very near start] & p(f_i^*(e_{|\ell|-i}) \arrow[d, "f_i"] \\
P_0(e_{|\ell|-i}) \arrow[r, "P(e_{|\ell|-i})"'] & p(e_{|\ell|-i})
\end{tikzcd}$$

Thus by pasting we obtain the diagram

$$\begin{tikzcd}
P_0\big(f_{|\ell|-1}^*(e_1)\big) \arrow[r, "P(f_{|\ell|-1}^*(e_1))"] \arrow[d, "f_{|\ell|-1}"'] \arrow[dr, phantom, "\lrcorner", very near start] & p\big(f_{|\ell|-1}^*(e_1)\big) \arrow[r] \arrow[d, "f_{|\ell|-2}"] \arrow[dr, phantom, "\lrcorner", very near start] & \cdots \arrow[r] & P_0\big(f^*(e_{|\ell|})\big) \arrow[r, "P(f^*(e_{|\ell|}))"] \arrow[d, "f_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & b' \arrow[d, "f"] \\
P_0(e_1) \arrow[r, "P(e_1)"'] & p(e_1) \arrow[r] & \cdots \arrow[r] & P_0(e_{|\ell|}) \arrow[r, "P(e_{|\ell|})"'] & b
\end{tikzcd}$$

where the outer rectangle is a pullback by pasting. Hence we can take our pullback to be

$$\begin{tikzcd}
P_0\big(f_{|\ell|-1}^*(e_1)\big) \arrow[r, "P(f^*(e_{|\ell|}))\circ\cdots\circ P(f_{|\ell|-1}^*(e_1))"] \arrow[d, "f_{|\ell|-1}"'] \arrow[dr, phantom, "\lrcorner", very near start] & b' \arrow[d, "f"] \\
P_0(e_1) \arrow[r, "\ell^\circ"'] & b,
\end{tikzcd}$$

which yields the desired arrow in $B_P(b')$. Q.E.D.

Proposition 10.4.6. *The assignment routing of Definition 10.4.4 can be made into a functor by mapping each strict map $(G,F)\colon P \to P'$ of comprehension categories to the computability transfer $F\colon (\mathscr{B}, B_P) \to (\mathscr{B}', B_{P'})$ and a transformation $(\tilde{\alpha}, \alpha)\colon (G,F) \to (G,F)$ to α.*

Proof: It suffices to show that F is a computability transfer, the functoriality is then immediate. So we have to show that if $\ell^\circ \in B_P(b)$, then $F(\ell^\circ) \in B_{P'}\big(F(b)\big)$. By definition we obtain $e_1, \dots, e_{|\ell|}$ such that $\ell^\circ = P(e_{|\ell|}) \circ \cdots \circ P(e_1)$. We compute

$$\begin{aligned} F(\ell^\circ) &= F\big(P(e_{|\ell|}) \circ \cdots \circ P(e_1)\big) = F\big(P(e_{|\ell|}\big) \circ \cdots \circ F\big(P(e_1)\big) \\ &= F^{[1]}\big(P(e_{|\ell|}\big) \circ \cdots \circ F^{[1]}\big(P(e_1)\big) \\ &= P'\big(G(e_{|\ell|})\big) \circ \cdots \circ P'\big(G(e_1)\big) \in B_{P'}\big(F(b)\big) \end{aligned}$$

where we used $(p' \circ G)(e_{|\ell|}) = (F \circ p)(e_{|\ell|} = F(b)$, as $b = p(e_{|\ell|})$.

For transformations of strict maps it is immediate that the resulting transformation of functors is natural, and the functoriality of this assignment is immediate as well. Q.E.D.

Theorem 10.4.7.

10.5 CatBaseComp is a (2-fam,Σ)-category

Definition 10.5.1. CatBaseComp is a fam-category where for each $(\mathscr{C}, C)$ we define

$$\mathrm{fHom}((\mathscr{C}, C)) := \{F\colon \mathscr{C}^{\mathrm{op}} \to \mathsf{Sets}\}.$$

The composition is simply the composition of the underlying functors.

Lemma 10.5.2. *For any $P \in \mathrm{fHom}((\mathscr{C}, C))$, the projection pr_1^P is a computability transfer*

$$\mathsf{pr}_1^P\colon \Big(\sum_{\mathscr{C}} P, C_{\mathsf{pr}_1^P}\Big) \to (\mathscr{C}, C).$$

Proof. We only need to show that $\mathsf{pr}_1^{\mathscr{C},P}$ preserves pullbacks to apply corollary 10.3.3. So suppose we are given a pullback square

$$\begin{tikzcd} f^*(c_1, x_1) \arrow[r, "g^*f"] \arrow[d, "f^*g"'] \arrow[dr, phantom, "\lrcorner", very near start] & (c_1, x_1) \arrow[d, "g"] \\ (c_2, x_2) \arrow[r, "f"] & (c_3, x_3). \end{tikzcd}$$

To see that this remains a pullback under $\mathsf{pr}_1^{\mathscr{C},P}$ assume we are given a commuting square

$$\begin{tikzcd} d \arrow[r, "h_1"] \arrow[d, "h_2"] & c_1 \arrow[d, "g"] \\ c_2 \arrow[r, "f"] & c_3 \end{tikzcd}$$

in $\mathscr{C}$. As $\mathsf{pr}_1^{\mathscr{C},P}$ is a fibration we obtain cartesian lifts $\hat{h}_1, \hat{h}_2$ of h_1, h_2 respectively. We see that $\hat{h}_1, \hat{h}_2$ have the same domain, as P maps this commuting square to the square

$$\begin{tikzcd} P(d) & P(c_1) \arrow[l, "P(h_1)"'] \\ P(c_2) \arrow[u, "P(h_2)"] & P(c_3) \arrow[l, "P(f)"] \arrow[u, "P(g)"'] \end{tikzcd}$$

of sets. We know that $x_1 = P(g)(x_3), x_2 = P(f)(x_3)$ and thus a simple computation shows that

$$\begin{aligned} P(h_1)(x_1) &= P(h_1)\big(P(g)(x_2)\big) = P(g \circ h_1) = P(f \circ h_2) \\ &= P(h_2)\big(P(f)(x_3)\big) = P(h_2)(x_2). \end{aligned}$$

Setting $y := P(h_2)(x_3)$ we get that

$$\begin{tikzcd} (d, y) \arrow[r, "\hat{h}_1"] \arrow[d, "\hat{h}_2"] & (c_1, x_1) \arrow[d, "g"] \\ (c_2, x_2) \arrow[r, "f"] & (c_3, x_3). \end{tikzcd}$$

commutes and thus we obtain a unique $q\colon (d, y) \to f^*(c_1, x_1)$ rendering

$$\begin{tikzcd} (d, y) \arrow[rrd, bend left, "\hat{h}_1"] \arrow[rdd, bend right, "\hat{h}_2"'] \arrow[rd, dotted, "q"] & & \\ & f^*(c_1, x_1) \arrow[r, "g^* f"] \arrow[d, "f^* g"'] \arrow[rd, phantom, "\lrcorner", very near start] & (c_1, x_1) \arrow[d, "g"] \\ & (c_2, x_2) \arrow[r, "f"] & (c_3, x_3). \end{tikzcd}$$

commutative. Thus we also know that

$$\begin{tikzcd} d \arrow[rrd, bend left, "\hat{h}_1"] \arrow[rdd, bend right, "\hat{h}_2"'] \arrow[rd, dotted, "q"] & & \\ & f^*(c_1) \arrow[r, "g^* f"] \arrow[d, "f^* g"'] \arrow[rd, phantom, "\lrcorner", very near start] & c_1 \arrow[d, "g"] \\ & c_2 \arrow[r, "f"] & c_3. \end{tikzcd}$$

is a commutative square in $\mathscr{B}$. The uniqueness stems from the fact that $\mathsf{pr}_1^{\mathscr{C},P}$ is also a discrete fibration so every other q' making the diagram commutative would have a unique lift $\tilde{q}$ making the diagram in $\sum_{\mathscr{C}} P$ commute and thus also be q. But this entails $P(q) = q = q'$. Q.E.D.

Theorem 10.5.3. $\mathsf{CatBaseComp}$ *is a (fam,Σ)-category.*

Proof. To see that

$$\begin{tikzcd} \left(\sum_{\mathscr{D}} P \circ F, D_{\mathsf{pr}_1^{P \circ F}}\right) \arrow[r, "\sum_P F"] \arrow[d, "\mathsf{pr}_1^{P \circ F}"'] & \left(\sum_{\mathscr{C}} P, C_{\mathsf{pr}_1^P}\right) \arrow[d, "\mathsf{pr}_1^P"] \\ (\mathscr{D}, D) \arrow[r, "F"'] & (\mathscr{C}, C) \end{tikzcd}$$

is a pullback square, we simply remark that

$$\begin{tikzcd} \sum_{\mathscr{D}} P \circ F \arrow[r, "\sum_P F"] \arrow[d, "\mathsf{pr}_1^{P \circ F}"'] & \sum_{\mathscr{C}} P \arrow[d, "\mathsf{pr}_1^P"] \\ \mathscr{D} \arrow[r, "F"] & \mathscr{C} \end{tikzcd}$$

is a pullback square in **Cat**, so we can apply Theorem 10.3.4. The strictness conditions are then immediately inherited from the strictness condition in **Cat**. Q.E.D.

10.6 Transporting bases of computability

Proposition 10.6.1. *Let $\mathscr{E}$ be a category with base of computability B. If we have a pullback-preserving functor $F\colon \mathscr{E} \to \mathscr{B}$ we obtain a base B^F on $\mathscr{B}$ defnned through*

$$B^F(d) = \{F(i) \mid i \in B(c) \wedge F(c) = d\} \cup \{\mathbf{1}_d\}.$$

Proof. The first property of a base is immediate as we have by design $\mathbf{1}_d \in B^F(a)$ for all $d \in \mathscr{B}$. So we check the second one. For this let the morphisms i, j, f in $\mathscr{B}$ as in

$$\begin{tikzcd} & & d_4 \arrow[d, hook, "j"] \\ d_1 & d_2 \arrow[l, hook', "i"'] \arrow[r, "f"'] & d_3 \end{tikzcd}$$

be given where $i \in B^F(d_1)$ and $j \in B^F(d_3)$. Again we distinguish two cases:

- If $j = \mathbf{1}_{d_3}$ we immediately see that $f^*\mathbf{1}_{d_3} = \mathbf{1}_{d_3}$ and thus

$$i \circ f^*\mathbf{1}_{d_3} = i \circ \mathbf{1}_{d_3} = i \in B^F(d_1)$$

  by defnnition of i.

- If $j = F(k)$ for some $k\colon c_4 \to c_3$ where $k \in B(c_4)$, we obtain a F-cartesian lift $f'\colon c_2 \to c_3$ of f, as $F(c_3) = d_3$. We can thus compute the pullback

$$\begin{tikzcd} c_3 \times_{c_4} c_2 \arrow[r, "k^*f'"] \arrow[d, "f'^*k"] & c_3 \arrow[d, "k"] \\ c_2 \arrow[r, "f'"] & c_4 \end{tikzcd}$$

  and as F is pullback-preserving we obtain that $F(c_3 \times_{c_4} c_2) \cong d_3 \times_{d_4} d_2$. Furthermore $F(c_2) = d_2$ we obtain that either i is always given as $h\colon c_2 \to c_1$ such that $F(h) = i$, as in the case that i is the identity we simply have that $h = \mathbf{1}_{c_2}$. So we can infer that $h \circ f'^*k \in B(c_1)$ and thus $F(h \circ f'^*k) = F(h) \circ F(f'^*k) \in B^F(d_1)$ as needed.

Q.E.D.

Lemma 10.6.2. *If we are given $\mathscr{C}, \mathscr{D}, \mathscr{E}$ and pullback-preserving fibrations $F\colon \mathscr{C} \to \mathscr{D}, G\colon \mathscr{D} \to \mathscr{E}$ then for a computability model B on $\mathscr{C}$ we have that*

- $B^{\mathbb{1}_\mathscr{C}} = B$,
- $B^{G \circ F} = (B^F)^G$.

Proof. The first claim is immediate from the defnnition. For the second claim we simply observe that if $i \in B^{G\circ F}(e)$ for some $e \in \mathscr{E}$, then either $i = \mathbf{1}_a$ and thus $i \in (B^G)^F(e)$ or $i = G(F(k))$ for some $k \in B(c)$ such that $G\big(F(c)\big) = e$. Then $F(k) \in B^F\big(F(c)\big)$ and thus $G\big(F(k)\big) = i \in (B^F)^G(e)$. The reverse direction is computed analogously. Q.E.D.

Definition 10.6.3. If $\mathscr{A}, \mathscr{B}, \mathscr{C}$ are categories and $F\colon \mathscr{A} \to \mathscr{B}, G\colon \mathscr{C} \to \mathscr{B}$ are functors, then the *comma-category* $F \downarrow G$ has

- as object triples (a, h, c) of objects $a \in \mathscr{A}, c \in \mathscr{C}$ and an arrow $h\colon F(a) \to G(c)$.

- Morphisms are pairs $(f,g)\colon (a,h,c) \to (a',h',c')$ of morphisms $f\colon F(a) \to F(a')$ and $g\colon G(c) \to G(c')$ such that
$$\begin{tikzcd}
F(a) \arrow[r, "h"] \arrow[d, "f"] & F(a') \arrow[d, "h'"] \\
G(c) \arrow[r, "g"] & G(c')
\end{tikzcd}$$
commutes.
- The composition $(i,j) \circ (f,g)$ is defnned as $(i \circ f, j \circ g)$.

Lemma 10.6.4. *Let $\mathscr{A}, \mathscr{B}, \mathscr{C}$ be categories and $F\colon \mathscr{A} \to \mathscr{B}, G\colon \mathscr{C} \to \mathscr{B}$ be pullback-preserving functors. Then the category $F \downarrow G$ has pullbacks.*

Proof. We want to show that $F \downarrow G$ has pullbacks, for this let a cospan

$$(a_1, h_1, c_1) \xrightarrow{(f,g)} (a_2, h_2, c_3) \xleftarrow{(i,j)} (a_3, h_3, c_3) \tag{23}$$

in $F \downarrow G$ be given. By defnnition this span stems from to spans in $\mathscr{A}, \mathscr{C}$ which are transported into $\mathscr{B}$ in the following manner:

$$\begin{tikzcd}
a_1 \arrow[d, "f"] & F(a_1) \arrow[r, "h_1"] \arrow[d, "F(f)"] & G(c_1) \arrow[d, "G(i)"] & c_1 \arrow[d, "i"] \\
a_2 & F(a_2) \arrow[r, "h_2"] & G(c_2) & c_2 \\
a_3 \arrow[u, "g"] & F(a_3) \arrow[r, "h_3"] \arrow[u, "F(g)"] & G(c_3) \arrow[u, "G(j)"] & c_3 \arrow[u, "j"] \\
\mathscr{A} & \mathscr{B} & & \mathscr{C}
\end{tikzcd}$$

We can take the pullbacks of both the span in $\mathscr{A}$ and in $\mathscr{C}$, these become the pullbacks of the two spans in $\mathscr{B}$ as both F and G are pullback-preserving. Our remaining goal is to show that $(a_1 \times_{a_2} a_3, \tilde{h}, c_1 \times_{c_2} c_3)$ is the desired pullback of (23). As F, G are pullback preserving we can use them to map the pullbacks into $\mathscr{B}$ which yields

$$\begin{tikzcd}
 & F(a_1) \arrow[r, "h_1"] \arrow[d, "F(f)"] & G(c_1) \arrow[d, "G(i)"] & \\
F(a_1 \times_{a_2} a_3) \arrow[ur, "F(g^* f)"] \arrow[dr, "F(f^* g)"'] & F(a_2) \arrow[r, "h_2"] & G(c_2) & G(c_1 \times_{c_2} c_3) \arrow[ul, "G(i^* j)"'] \arrow[dl, "G(j^* i)"] \\
 & F(a_3) \arrow[r, "h_3"] \arrow[u, "F(g)"] & G(c_3) \arrow[u, "G(j)"] &
\end{tikzcd} \tag{24}$$

where all squares commute. This in turn entails

$$\begin{aligned}
G(i) \circ h_1 \circ F(f^* g) = h_2 \circ F(f) \circ F(F^* g) &= h_2 \circ F(g) \circ F(g^* f) \\
&= G(j) \circ h_3 \circ F(g^* f).
\end{aligned}$$

Thus from the universal property of pullbacks we obtain a $\tilde{h}\colon F(a_1 \times_{a_2} a_3) \to G(c_1 \times_{c_2} c_3)$ such that

$$G(j^* i) \circ \tilde{h} = h_3 \circ F(f^* g) \quad \text{and} \quad G(i^* j) \circ \tilde{h} = h_1 \circ F(g^* f). \tag{25}$$

The above also shows that

$$\begin{aligned} (f^*g, j^*i)\colon (a_1 \times_{a_2} a_3, \tilde{h}, c_1 \times_{c_2} c_3) &\to (a_3, h_3, c_3) \\ \text{and } (g^*f, i^*j)\colon (a_1, h_1, c_1) &\to (a_1 \times_{a_2} a_3, \tilde{h}, c_1 \times_{c_2} c_3) \end{aligned}$$

are morphisms in $F \downarrow G$. It remains to show that $(a_1 \times_{a_2} a_3, \tilde{h}, c_1 \times_{c_2} c_3)$ fulfills the universal property of the pullback. So assume we are given $(a', h', c') \in (F \downarrow G)$ together with morphisms $(n_1, m_1)\colon (a', h', c') \to (a_1, h_1, c_1)$ and $(n_3, m_3)\colon (a', h', c') \to (a_3, h_3, c_3)$ such that

$$\begin{tikzcd}
(a', h', c') \arrow[r, "{(n_2,m_2)}"] \arrow[d, "{(n_3,m_3)}"] & (a_3, h_3, c_3) \arrow[d, "{(i,j)}"] \\
(a_1, h_1, c_1) \arrow[r, "{(f,g)}"] & (a_2, h_2, c_2)
\end{tikzcd}$$

commutes. This in turn decomposes to the respective commuting squares in $\mathscr{A}, \mathscr{C}$, so we can use the pullback properties of $a_1 \times_{a_2} a_3$ and $c_1 \times_{c_2} c_3$ to obtain arrows n', m' making the diagrams

$$\begin{tikzcd}
& a' \arrow[dl, "n_1"'] \arrow[d, "n'"] \arrow[dr, "n_3"] & & \text{and} & & c' \arrow[dl, "m_1"'] \arrow[d, "m'"] \arrow[dr, "m_3"] & \\
a_1 & a_1 \times_{a_2} a_3 \arrow[l, "f^*g"] \arrow[r, "g^*f"'] & a_3 & & c_1 & c_1 \times_{c_2} c_3 \arrow[l, "i^*j"] \arrow[r, "j^*i"'] & c_3
\end{tikzcd}$$

commute. Our remaining goal is to show that

$$\begin{tikzcd}
F(a') \arrow[r, "h'"] \arrow[d, "F(n')"] & G(c') \arrow[d, "G(m')"] \\
F(a_1 \times_{a_2} a_3) \arrow[r, "\tilde{h}"] & G(c_1 \times_{c_2} c_3)
\end{tikzcd}$$

commutes. We pack all morphisms considered so far into one big diagram:

$$\begin{tikzcd}
F(a') \arrow[dr, "h'"] \arrow[ddr, "F(n')"'] \arrow[ddrrr, bend left, "F(n_3)"] \arrow[ddddr, bend right, "F(n_1)"'] & & & & \\
& G(c') \arrow[ddr, "G(m')"] \arrow[ddrrr, bend left, "G(m_3)"] \arrow[ddddr, "G(m_1)"'] & & & \\
& F(a_1 \times_{a_2} a_3) \arrow[rr, "F(g^*f)"] \arrow[dr, "\tilde{h}"] \arrow[dd, "F(f^*g)"'] & & F(a_3) \arrow[dr, "h_3"] \arrow[dd, "F(g)"] & \\
& & G(c_1 \times_{c_2} c_3) \arrow[rr, "G(g^*f)"] \arrow[dd, "G(f^*g)"] & & G(c_3) \arrow[dd, "G(g)"] \\
& F(a_1) \arrow[rr, "F(f)"] \arrow[dr, "h_1"'] & & F(a_2) \arrow[dr, "h_2"] & \\
& & G(c_1) \arrow[rr, "G(f)"] & & G(c_2).
\end{tikzcd}$$

In this diagram all squares commute (except the one we want to prove) as well as the triangles of the pullbacks. This allows us to compute that

$$G(f) \circ G(m_1) \circ h' = G(f) \circ G(f^*g) \circ G(m') \circ h' = G(g) \circ G(g^*f) \circ G(m') \circ h'$$

$$= G(g) \circ G(m_3) \circ h'.$$

Thus by the pullback property we obtain an arrow $\check{h}\colon F(a') \to G(c_1 \times_{c_2} c_3)$ such that $G(g^*f) \circ \check{h} = G(m_3) \circ h$ and $G(f^*g) \circ \check{h} = G(m_1) \circ h'$. One can check that both $\check{h} = G(m') \circ h'$ and $\check{h} = \tilde{h} \circ F(n')$ fulfil these conditions, for $G(m') \circ h'$ this is immediate and for $\tilde{h} \circ F(n')$ we compute

$$G(g^*f) \circ \tilde{h} \circ F(n') = h_3 \circ F(g^*f) \circ F(n') = h_3 \circ F(n_3) = G(m_3) \circ h'$$

and

$$G(f^*g) \circ \tilde{h} \circ F(n') = h_1 \circ F(f^*g) \circ F(n') = h_1 \circ F(n_1) = G(m_1) \circ h'.$$

By uniqueness we have the desired equality. Q.E.D.

Proposition 10.6.5. *Let $\mathscr{A}, \mathscr{B}, \mathscr{C}$ be categories and $F\colon \mathscr{A} \to \mathscr{B}, G\colon \mathscr{C} \to \mathscr{B}$ be pullback-preserving functors. Let B_A, B_B, B_C be bases of computability on $\mathscr{A}, \mathscr{B}, \mathscr{C}$ respectively. Then we obtain a base of computability $B_\downarrow$ on $F \downarrow G$ defnned by*

$$B_\downarrow(a, h, c) = \big\{(i, j) \mid i \in B_A(a) \,\&\, F(i) \in B_B(F(a)) \,\&\, j \in B_C(c) \,\&\, G(j) \in B_B(F(c))\big\}.$$

Proof. The first condition on bases of computability is immediate, as the identity on (a, h, c) is $(\mathbf{1}_a, \mathbf{1}_c)$ and we can immediately see that $\mathbf{1}_a \in B_A(a), S(\mathbf{1}_a) = 1_{S(a)} \in B\big(S(a)\big)$, analogously for c. For the second condition we assume we are given objects and morphisms as in the diagram

$$\begin{tikzcd}
 & & (a_1, h_1, c_1) \arrow[d, "(i_1, j_1)"] \\
(a_2, h_2, c_2) & (a_3, h_3, c_3) \arrow[l, "(i_2, j_2)"] \arrow[r, "(f, g)"] & (a_4, h_4, c_4)
\end{tikzcd}$$

where we assume that $(i_1, j_1), (i_2, j_2)$ are in the computability base $B_\downarrow$ and (f, g) is arbitrary. As the comma category has pullbacks in our case by the preceding lemma we obtain the pullback

$$\begin{tikzcd}
 & (a_1 \times_{a_4} a_3, \tilde{h}, c_1 \times_{c_4} c_3) \arrow[r, "(i_1^* f, j_1^* g)"] \arrow[d, "(f^* i_1, g^* j_1)"] & (a_1, h_1, c_1) \arrow[d, "(i_1, j_1)"] \\
(a_2, h_2, c_2) & (a_3, h_3, c_3) \arrow[l, "(i_2, j_2)"] \arrow[r, "(f, g)"] & (a_4, h_4, c_4).
\end{tikzcd}$$

As $(i_2, j_2) \circ (f^*i_1, g^*j_1) = (i_2 \circ f^*i_1, j_2 \circ g^*j_1)$ and both of these morphisms are in B_A, B_C respectively it remains to check that $F(i_2 \circ f^*i_1) \in B_B\big(F(a_2)\big)$ and $T(j_2 \circ g^*j_1) \in B_B\big(T(c)\big)$. By defnnition of $B_\downarrow$ we know that $F(i), F(f) \in B_B\big(F(a_4)\big)$, $F(i_2) \in B_B\big(F(a_2)\big)$. This allows us to infer that $F(f^*i_1) = F(f)^*F(i_1) \in B_B\big(F(a_3)\big)$ as S is pullback preserving. Thus $F(f^*i_1) \circ F(i_2) = F(f^*i_1 \circ i_2) \in B_B\big(F(a_2)\big)$. The same argument work for G and the second components of the morphisms. Q.E.D.

Examples 10.6.6.

1. Given a category $\mathscr{C}$ with a base of computability B and if c is an object of $\mathscr{C}$ we can endow the slice category $\mathscr{C}_{/c}$ with a base of compitability $B_/$ defnned by

$$B_/(f\colon b \to c) = \{i\colon a \to b \mid i \in B(b)\}.$$

   This stems from the fact that the slice category is the comma category of the span

$$\mathscr{C} \xrightarrow{\mathbf{1}_{\mathscr{C}}} \mathscr{C} \xleftarrow{\iota_c} \mathbb{1}$$

   where ι_c is the functor that sends the only object $\emptyset$ of $\mathbb{1}$ to c and its identity to the identity of c.

2. In the same way we can endow the coslice category ${}_{c/}\mathscr{C}$ with the computability base ${}_{/}B$
$$ {}_{/}B(f\colon c\to b) = \{i\colon a\to b \mid i\in B(b)\}.$$
This stems from the fact that the coslice category stems from the span
$$\mathbb{1} \xrightarrow{\iota_c} \mathscr{C} \xleftarrow{\mathbf{1}_{\mathscr{C}}} \mathscr{C}.$$

3. If B is a base of computability on a category $\mathscr{C}$ we can equip the arrow category $\mathscr{C}^{\to}$ with the base of computability $B^{\to}$ defnned by
$$B^{\to}(f\colon c\to d) = \{(i,j)\colon (c',d')\to(c,d) \mid i\in B(c)\wedge j\in B(d)\}.$$
This stems from the fact that the arrow category stems from the span
$$\mathscr{C} \xrightarrow{\mathbf{1}_{\mathscr{C}}} \mathscr{C} \xleftarrow{\mathbf{1}_{\mathscr{C}}} \mathscr{C}.$$

Proposition 10.6.7. *If $\mathscr{D}$ is a category with pullbacks and B is a base of computability on $\mathscr{D}$, then for any category $\mathscr{C}$ the functor category $[\mathscr{C},\mathscr{D}]$ can by equipped with a base of computability $[\mathscr{C},B]$ defnned by*
$$[\mathscr{C},B](F) = \big\{\eta\colon F'\Rightarrow F \mid \forall_{c\in\mathscr{C}}\eta_c\in B\big(F(c)\big)\big\}.$$

Proof. We show that $[\mathscr{C},B]$ defnned in this manner constitutes a base of computability. The first property of bases is immediate, as if F is a functor, then the identity $\mathbf{1}_F$ is the natural transformation consisting of the identities $\mathbf{1}_{F(c)}\colon F(c)\to F(c)$ for all c. But $\mathbf{1}_{F(c)}$ is in B for all $c\in\mathscr{C}$ as B is a base of computability, so $\mathbf{1}_F$ is in $[\mathscr{C},B]$. The second condition can be computed explicitly in the same way, if we are given a functors and natural transformations as in the diagram
$$\begin{tikzcd}
 & & F_4 \arrow[d, "\eta"] \\
F_1 & F_2 \arrow[l, "\chi"'] \arrow[r, "\mu"'] & F_3,
\end{tikzcd}$$
where χ,η as in $[\mathscr{C},B]$ and μ is arbitrary, then for each $c\in\mathscr{C}$ we obtain the situation as in
$$\begin{tikzcd}
 & & F_4(c) \arrow[d, "\eta_c"] \\
F_1(c) & F_2(c) \arrow[l, "\chi_c"'] \arrow[r, "\mu_c"'] & F_3(c),
\end{tikzcd}$$
which allows us to take the pullback $F_2(c)\times_{F_3(c)}F_4(c)$ and as B is a base of computability we obtain that $\chi_c\circ\mu_c^*\eta_c$ is in B, and thus $\chi\circ\mu^*\eta$ is in $[\mathscr{C},B]$. Q.E.D.

Proposition 10.6.8. *If we are given two categories $(\mathscr{C},C),(\mathscr{D},D)$ with respective bases, then the category $[\mathscr{C},\mathscr{D}]$ together with the base $[\mathscr{C},D]$ described above is the exponential of $(\mathscr{C},C),(\mathscr{D},D)$ in* CatBaseComp.

Proof. We first show that if we consider the commutative triangle
$$\begin{tikzcd}
{[\mathscr{C},\mathscr{E}]\times\mathscr{D}} \arrow[r, "\mathtt{eval}"] & \mathscr{E} \\
\mathscr{C}\times\mathscr{D} \arrow[u, "\hat{F}\times 1_{\mathscr{D}}"] \arrow[ur, "F"] &
\end{tikzcd}$$

of categories and functors, where $F\colon (\mathscr{C}, C) \times (\mathscr{D}, D) \to (\mathscr{E}, E)$ is a computability transformation, then `eval` and $\hat{F}$ are already computability transfers

$$\texttt{eval}\colon \big([\mathscr{C}, \mathscr{E}], [\mathscr{C}, C]\big) \to (\mathscr{E}, E), \quad \hat{F}\colon (\mathscr{C}, C) \to \big([\mathscr{C}, \mathscr{E}], [\mathscr{C}, B]\big).$$

First recall that

$$\big([\mathscr{C}, \mathscr{E}], [\mathscr{C}, C]\big) \times (\mathscr{D}, D) = \big([\mathscr{C}, \mathscr{E}] \times \mathscr{D}, [\mathscr{C}, C] \times D\big)$$

as defnned earlier, that is an element of $[\mathscr{C}, C] \times D$ is a pair (η, f). So assume we are given $(\eta, f) \in \big(\mathscr{C}, C] \times D\big)(G, d)$, that is

$$\eta\colon H \Rightarrow G, \quad f\colon d' \to d.$$

By defnnition `eval`(η, f) is defnned as either composite in the commutative square

$$\begin{tikzcd}
H(d') \arrow[r, "H(f)"] \arrow[d, "\eta_{d'}"'] & H(d) \arrow[d, "\eta_d"] \\
G(d') \arrow[r, "G(f)"] & G(d).
\end{tikzcd}$$

By defnnition of $[\mathscr{C}, C]$ we have that $\eta_d \in E\big(G(d)\big)$ and as H is a computability transfer we obtain that $\eta_d \circ H(f) \in E\big(G(d)\big)$. Thus `eval` is a computability transfer. To see that $\hat{F}$ is a computability transfer we simply compute that if $g\colon c \to c'$ lies in $C(c')$, then

$$\hat{F}(g) = F(g, -)\colon F(c, -) \Rightarrow F(c', -), \quad \hat{F}(g)_d = F(g, d)\colon F(c, d) \to F(c', d).$$

Q.E.D.

10.7 Bases of computability and computability models

So far we have not seen any reason why we call this structure "bases of computability" instead of "dominions" or "stable systems of monics". Hence it is time to start discussing the relations between these two concepts. However for this relation to be established we need to demand that every category is "concrete" in the way that its objects are sets and its morphisms are maps (with additional structure). Thus from now on we will require our categories $\mathscr{C}$ to come equipped with a presheaf $S\colon \mathscr{C} \to \mathsf{Sets}$. A bit of notation beforehand

Notation: Let $\mathscr{C}$ be a category with pullbacks, $S\colon \mathscr{C} \to \mathsf{Sets}$ be a pullback preserving presheaf and B be a base of computability on $\mathscr{C}$. We denote by $S^B[c, c']$ the set

$$S^B[c, c'] = \{S(i, f) \mid (i, f)\colon c \rightharpoonup c' \wedge i \in B(c)\}$$

for all $c, c' \in \mathscr{C}$.

Definition 10.7.1 (Canonical computability models - [50]). Let $\mathscr{C}$ be a category with a presheaf $S\colon \mathscr{C} \to \mathsf{Sets}$ and B be a base of computability on $\mathscr{C}$. Then we define the *canonical computability model* associated to $\mathscr{C}, S, B$ to be the model

$$\mathbf{CM}^B(\mathscr{C}; S) = \big((S(c))_{c \in \mathscr{C}}, (S^B[c, c'])_{c, c' \in \mathscr{C}}\big).$$

Example 10.7.2. If we take tot to be the total base on a category $\mathscr{C}$, that is the case consisting of just the identities, then $\mathbf{CM}^{\text{tot}}(\mathscr{C}; S)$ agrees with the definition we gave earlier, similarly $\mathbf{CM}^{\text{prt}}(\mathscr{C}; S)$ agrees with this earlier definition.

The above definition shows that we can associate to each category with base of computability a canonical computability model. This assignment routine can be expanded into a functor.

Proposition 10.7.3. *Let $(\mathscr{C}, B)$ and $(\mathscr{D}, Y)$ be two categories with bases of computability, $S\colon \mathscr{C} \to \mathsf{Sets}$ and $R\colon \mathscr{D} \to \mathsf{Sets}$ be two pullback-preserving presheaves and $F\colon (\mathscr{C}, B) \to (\mathscr{D}, Y)$ be a computability transfer, such that the triangle*

$$\begin{tikzcd}
\mathscr{C} \arrow[rr, "F"] \arrow[dr, "S"'] & & \mathscr{D} \arrow[dl, "R"] \\
& \mathsf{Sets} &
\end{tikzcd}$$

commutes. Then we obtain a simulation $\boldsymbol{\gamma}\colon \mathbf{CM}^B(\mathscr{C}; S) \mathrel{-\!\!\triangleright} \mathbf{CM}^Y(\mathscr{D}; R)$ defined by the following clauses:

- *The class function $\gamma\colon \mathscr{C}_0 \to \mathscr{D}_0$ is defined by $\gamma := F_0$.*
- *For all $c \in \mathscr{C}_0$ we define $\Vdash^{\gamma}_{c} \subseteq R(F(c)) \times S(c)$ in the following way:*

$$y \Vdash^{\gamma}_{c} x :\Leftrightarrow y = x.$$

Hence we obtain a simulation $\boldsymbol{\gamma}\colon \mathbf{CM}^B(\mathscr{C}; S) \mathrel{-\!\!\triangleright} \mathbf{CM}^Y(\mathscr{D}; R)$ from a computability transfer $F\colon (\mathscr{C}, B) \to (\mathscr{D}, Y)$ such that $S = R \circ F$ for pullback-preserving presheaves $S\colon \mathscr{C} \to \mathsf{Sets}, R\colon \mathscr{D} \to \mathsf{Sets}$. Using the definitions of [**PetStrict**] we can see that this simulation $\boldsymbol{\gamma}$ is an equality simulation, albeit not always a full one. So schematically we have

$$\begin{tikzcd}
& \mathscr{C} \arrow[dl, "S"'] \arrow[dd, "F"'{name=F}] & & \mathbf{CM}^B(\mathscr{C}; S) \arrow[dd, "\boldsymbol{\gamma}"'{name=G}, "\shortparallel"] \\
\mathsf{Sets} \arrow[r, phantom, "\circlearrowright"] & {} & & \\
& \mathscr{D} \arrow[ul, "R"] & & \mathbf{CM}^Y(\mathscr{D}; R).
\arrow[from=F, to=G, mapsto]
\end{tikzcd}$$

Definition 10.7.4 (The category $\mathsf{CatBasePshv}^{\mathsf{id}}$). The category $\mathsf{CatBasePshv}^{\mathsf{id}}$ is defined through the following data:

- The objects of $\mathsf{CatBasePshv}^{\mathsf{id}}$ are triples $(\mathscr{C}, B, S)$, where $(\mathscr{C}, B)$ is a category with a base of computability and $S\colon \mathscr{C} \to \mathsf{Sets}$ is a pullback-preserving presheaf.
- The arrows $F\colon (\mathscr{C}, B, S) \to (\mathscr{D}, Y, R)$ are computability transfers $F\colon \mathscr{C} \to \mathscr{D}$, such that the triangle

$$\begin{tikzcd}
\mathscr{C} \arrow[rr, "F"] \arrow[dr, "S"'] & & \mathscr{D} \arrow[dl, "R"] \\
& \mathsf{Sets} &
\end{tikzcd}$$

commutes.

The "id" in the superscript at the end stems from the fact that this category is a special case of a more general case which we define in 10.7.7.

Lemma 10.7.5. *We have a functor $\mathrm{CM}^{\text{-}}(\text{-}; \text{-})\colon \mathsf{CatBasePshv}^{\mathsf{id}} \to \mathsf{CompModEq}$, defined through the following clauses: for each $(\mathscr{C}, B, S) \in \mathsf{CatBasePshv}^{\mathsf{id}}$ we set*

$$\mathrm{CM}^{\text{-}}(\text{-}; \text{-})(\mathscr{C}, B, S) = \mathbf{CM}^B(\mathscr{C}; S)$$

and for $F\colon (\mathscr{C}, B, S) \to (\mathscr{D}, Y), R)$ *we define* $\mathrm{CM}^{\text{-}}(\text{-}; \text{-})(F) = \boldsymbol{\gamma}$*, where* $\boldsymbol{\gamma}$ *is defined as in lemma 10.7.3. We will write* $\mathrm{CM}^{\text{-}}(F; \text{-})$ *instead of* $\mathrm{CM}^{\text{-}}(\text{-}; \text{-})(F)$ *for a morphism F from* $(\mathscr{C}, B, S)$ *to* $(\mathscr{D}, Y), R)$

Proof. The preservation of identities is immediate, whereas preservation of composition is a bit more tedious to prove. So let two morphisms $F\colon (\mathscr{C}, B, S) \to (\mathscr{D}, Y, R)$ and $G\colon (\mathscr{D}, Y, R) \to (\mathscr{E}, Z, U)$ in $\mathsf{CatBasePshv}^{\mathrm{id}}$ be given, so we obtain the commutative diagram

$$\begin{tikzcd}
\mathscr{C} \arrow[r, "F"] \arrow[dr, "S"'] & \mathscr{D} \arrow[r, "G"] \arrow[d, "R"] & \mathscr{E} \arrow[dl, "U"] \\
& \mathsf{Sets}\,. &
\end{tikzcd}$$

Let $\boldsymbol{\gamma}$ be the simulation assigned to F, $\boldsymbol{\delta}$ be the simulation assigned to G and $\boldsymbol{\epsilon}$ be the simulation assigned to $G \circ F$. Our objective is to prove $\boldsymbol{\epsilon} = \boldsymbol{\delta} \circ \boldsymbol{\gamma}$. On the level of the class functions $\mathscr{C}_0 \to \mathscr{E}_0$ this is immediate. The class function $\delta \circ \gamma = G_0 \circ F_0$ by definition of the composite simulation and $\epsilon = G_0 \circ F_0$ by definition of the assigned simulation. It remains to examine the relations $\Vdash_c^{\epsilon}$ and $\Vdash_c^{\delta\circ\gamma}$ for $c \in \mathscr{C}_0$. We know that for $x \in S(c)$ and $y \in U(G(F(c)))$ the equivalence

$$y \Vdash_c^{\epsilon} x \Leftrightarrow y = x$$

holds by definition. On the other hand we have the equivalence

$$y \Vdash_c^{\delta\circ\gamma} \Leftrightarrow \exists z : y \Vdash_{F(c)}^{\delta} z \,\&\, z \Vdash_c^{\gamma} x.$$

But if such a $z \in R(F(c))$ exists, we immediately have $y = z = x$, so the two relations $\Vdash_c^{\epsilon}$ and $\Vdash_c^{\delta\circ\gamma}$ coincide. Q.E.D.

Next we generalise $\mathsf{CatBasePshv}^{\mathrm{id}}$ by allowing more morphisms. The morphisms of this category were computability transfers $(\mathscr{C}, B) \to (\mathscr{D}, Y)$ that respect presheaves in the sense that

$$\begin{tikzcd}
\mathscr{C} \arrow[rr, "F"] \arrow[dr, "S"'] & & \mathscr{D} \arrow[dl, "R"] \\
& \mathsf{Sets} &
\end{tikzcd}$$

commutes. We now allow this diagram to commute only “up to” an endofunctor on Sets, that is up to a $\text{Ł}\colon \mathsf{Sets} \to \mathsf{Sets}$ such that the diagram

$$\begin{tikzcd}
\mathscr{C} \arrow[r, "F"] \arrow[d, "S"'] & \mathscr{D} \arrow[d, "R"] \\
\mathsf{Sets} \arrow[r, dashed, "\text{Ł}"'] & \mathsf{Sets}
\end{tikzcd}$$

commutes, together with a natural transformation $\eta\colon \text{Ł} \Rightarrow \mathrm{id}_{\mathsf{Sets}}$. We assign to such a functor F a simulation as in the following lemma.

Proposition 10.7.6. *Let* $(\mathscr{C}, B), (\mathscr{D}, Y)$ *be categories with bases of computability, and let*

$$S\colon \mathscr{C} \to \mathsf{Sets}, R\colon \mathscr{D} \to \mathsf{Sets}, F\colon \mathscr{C} \to \mathscr{D}$$

be given where R, S *are pullback-preserving functors and* F *is a computability transfer. Let* $\text{Ł}\colon \mathsf{Sets} \to \mathsf{Sets}$ *be an endofunctor (which we shall call the* mirroring morphism*) such that the diagram*

$$\begin{tikzcd}
\mathscr{C} \arrow[r, "F"] \arrow[d, "S"'] & \mathscr{D} \arrow[d, "R"] \\
\mathsf{Sets} \arrow[r, dashed, "\text{Ł}"'] & \mathsf{Sets}
\end{tikzcd}$$

commutes. Furthermore we demand the existence of a natural transformation $\eta\colon \mathrm{id}_{\mathsf{Sets}} \to \Finv$. *Then we obtain a simulation* $\boldsymbol{\gamma}^F\colon \mathbf{CM}^B(\mathscr{C}; S) \to \mathbf{CM}^Y(\mathscr{C}; R)$ *defined by the following clauses:*

- *The class function* $\gamma^F\colon \mathscr{C}_0 \to \mathscr{D}_0$ *is given by* $\gamma^F := F_0$.
- *For all* $c \in \mathscr{C}_0$ *we define* $\Vdash_c^{\gamma^F} \subseteq R(F(c)) \times S(c)$ *via*

$$y \Vdash_c^{\gamma^F} x :\Leftrightarrow y = \eta_{S(a)}(x).$$

Similar to the less general case we this rule assigning to a computability transfer a simulation gives rise to a functor between appropriate categories. For this we define the more general version of $\mathsf{CatBasePshv}^{\mathrm{id}}$.

Definition 10.7.7 (The category $\mathsf{CatBasePshv}$). Let $\mathsf{CatBasePshv}$ be the category defined by the following data:

- The objects of $\mathsf{CatBasePshv}$ are the same objects as the ones of $\mathsf{CatBasePshv}^{\mathrm{id}}$, that is pairs $(\mathscr{C}, B, S)$, where $(\mathscr{C}, B)$ is a category with a base of computability and $S\colon \mathscr{C} \to \mathsf{Sets}$ is a pullback-preserving presheaf.
- The arrows between two objects $(\mathscr{C}, B, S) \to (\mathscr{D}, Y, R)$ are given as triplets $(F, \Finv, \eta)$, where
  - $F\colon (\mathscr{C}, B) \to (\mathscr{D}, Y)$ is a computability transfer,
  - $\Finv\colon \mathsf{Sets} \to \mathsf{Sets}$ is an endofunctor making the diagram

$$\begin{tikzcd}
\mathscr{C} \arrow[r, "F"] \arrow[d, "S"'] & \mathscr{D} \arrow[d, "R"] \\
\mathsf{Sets} \arrow[r, dashed, "\Finv"'] & \mathsf{Sets}
\end{tikzcd}$$

    commute and
  - $\eta\colon \mathrm{id}_{\mathsf{Sets}} \to \Finv$ is a natural transformation.

Proposition 10.7.8. *We have a functor* $\mathrm{CM}^{\text{-}}(\text{-}; \text{-})\colon \mathsf{CatBasePshv} \to \mathsf{CompMod}$*, which takes each* $(\mathscr{C}, B, S) \in \mathsf{CatBasePshv}$ *to its computability model* $\mathbf{CM}^B(\mathscr{C}; S)$ *and*

$$\Big((F, \Finv, \eta)\colon (\mathscr{C}, B, S) \to (\mathscr{D}, Y, R)\Big) \mapsto \gamma^F\colon \mathbf{CM}^B(\mathscr{C}; S) \to \mathbf{CM}^Y(\mathscr{D}; R)$$

as defined in the lemma 10.7.6.

Proof. One can immediately see that `Assign` preserves identities, as the identity of $(\mathscr{C}, B, S)$ is simply $(\mathrm{id}_{\mathscr{C}}, \mathrm{id}_{\mathsf{Sets}}, \mathrm{id})$, and by definition we have that $\gamma^{\mathrm{id}_{\mathscr{C}}} = (\mathrm{id}_{\mathscr{C}_0}, (\Vdash_c^{\gamma^{\mathrm{id}_{\mathscr{C}}}})_{c \in \mathscr{C}_0})$ where $x \Vdash_c^{\gamma^{\mathrm{id}_{\mathscr{C}}}} y \Leftrightarrow \mathrm{id}_{S(c)}(x) = x = y$, so $\Vdash_c^{\gamma^{\mathrm{id}_{\mathscr{C}}}}$ is identical to the equality relation.

Suppose we have two morphisms,

$$(F, \Finv, \eta)\colon (\mathscr{C}, B, S) \to (\mathscr{D}, Y), R), \quad (G, \Game, \rho)\colon (\mathscr{D}, Y, R) \to (\mathscr{E}, Z, U)$$

in the category $\mathsf{CatBasePshv}$. So we have the commutative diagram

$$\begin{tikzcd}
\mathscr{C} \arrow[r, "F"] \arrow[d, "S"'] & \mathscr{D} \arrow[r, "G"] \arrow[d, "R"] & \mathscr{E} \arrow[d, "U"] \\
\mathsf{Sets} \arrow[r, dashed, "\Finv"'] & \mathsf{Sets} \arrow[r, dashed, "\Game"'] & \mathsf{Sets}.
\end{tikzcd}$$

Our goal is now to show that $\boldsymbol{\gamma}^{G\circ F} = \boldsymbol{\gamma}^{G} \circ \boldsymbol{\gamma}^{F}$. On the level of the class function this is immediate, we have that in both cases the class function is given by $G_0 \circ F_0$. So it remains to examine the tracking relations for $c \in \mathscr{C}_0$.

Let $c \in \mathscr{C}_0$ be given. We seek to show that for all $y \in U(G(F(c)))$ and $x \in S(c)$ we have the equivalence

$$y \Vdash_c^{\gamma^{G\circ F}} x \Leftrightarrow y \Vdash_c^{\gamma^{G}\circ\gamma^{F}} x.$$

So we first assume that $y \Vdash_c^{\gamma^{G\circ F}} x$. This means that if we define ξ to be the natural transformation $\xi \colon \mathrm{id}_{\mathsf{Sets}} \Rightarrow G \circ F$ by setting $\xi_c := \rho_{F_0(c)} \circ \eta_c$, we obtain $y = \xi_{S(c)}(x)$. So to see that $y \Vdash_c^{\gamma^{G}\circ\gamma^{F}} x$ we only need to prove the existence of a z such that

$$y \Vdash_{F_0(c)}^{\gamma^{G}} z \,\&\, z \Vdash_c^{\gamma^{F}} x.$$

But one can see that $z := \eta_c^x$ does the trick.

On the other hand, let z be given such that

$$y \Vdash_{F_0(c)}^{\gamma^{G}} z \,\&\, z \Vdash_c^{\gamma^{F}} x.$$

Then $z = \eta_c(x)$ and $y = \rho_{F_0(c)}(z) = \rho_{F_0(c)}(\eta_c(x))$, so $y \Vdash_c^{\gamma^{G}\circ\gamma^{F}} x$. So the desired equivalence has been proven and $\mathrm{CM}^{\text{-}}(\text{-}\,;\text{-}) \colon \mathsf{CatBasePshv} \to \mathsf{CompMod}$ is a functor. Q.E.D.

So we can now view the case $\mathrm{CM}^{\text{-}}(\text{-}\,;\text{-}) \colon \mathsf{CatBasePshv}^{\mathrm{id}}$ as a special case of this more general functor in the following way: It is immediate that we have an inclusion $\iota \colon \mathsf{CatBasePshv}^{\mathrm{id}} \to \mathsf{CatBasePshv}$, so we can summarize the situation in the following commutative diagram:

$$\begin{tikzcd}
\mathsf{CatBasePshv}^{\mathrm{id}} \arrow[r, "\mathrm{CM}^{\text{-}}(\text{-}\,;\text{-})"] \arrow[d, hook, "\iota"] & \mathsf{CompModEq} \arrow[d, hook, "\imath ncl"] \\
\mathsf{CatBasePshv} \arrow[r, "\mathrm{CM}^{\text{-}}(\text{-}\,;\text{-})"] & \mathsf{CompMod},
\end{tikzcd}$$

where $\imath ncl \colon \mathsf{CompModEq} \to \mathsf{CompMod}$ is the obvious inclusion.

Having considered the more general case, we re our attention to the case that the category with base of computability is the same, and only the base of computability and the presheaf $S \colon \mathscr{C} \to \mathsf{Sets}$ vary.

Lemma 10.7.9. *Let $(\mathscr{C}, B)$ be a category with a base of computability and $S, S' \colon \mathscr{C} \to \mathsf{Sets}$ be pullback-preserving presheaves. If we are also given a natural transformation $\mu \colon S \to S'$ we obtain a simulation $\boldsymbol{\gamma}^{\eta} = (\mathrm{id}_{\mathscr{C}_0}, (\Vdash_c^{\gamma^{\mu}})_{c\in\mathscr{C}_0}) \colon \mathbf{CM}^B(\mathscr{C}; S) \twoheadrightarrow \mathbf{CM}^B(\mathscr{C}; S')$, where $\Vdash_c^{\gamma^{\mu}} \subseteq S'(c) \times S(c)$ is defined through the equivalence*

$$\big(y \Vdash_c^{\gamma^{\mu}} x\big) :\Leftrightarrow \big(y = \mu_{S(c)}(x)\big).$$

Proof. One can immediately see that for all $c \in \mathscr{C}_0$ the relation $\Vdash_c^{\gamma^{\mu}} \subseteq S'(c) \times S(c)$ is well-defined. So it remains to prove the axioms of a simulation.

($\mathtt{Siml}_1$) Let $c \in \mathscr{C}_0$ be given and $x \in S(c)$. Then by definition $\mu_{S(c)}(x) \in S'(c)$ and $\mu_{S(c)}(x) \Vdash_c^{\gamma^{\mu}} x$.

($\mathtt{Siml}_2$) Let $S(i,f) \in S[a,b]$ be given. We seek to show that $S'(i,f) \Vdash_{(a,b)}^{\gamma^{\mu}} S(i,f)$. First we observe that $S'(i,f) \in S'[a,b]$, as $i \in B(a)$ and $f \colon \mathrm{dom}(i) \to b$, otherwise we could not have $S(i,f) \in S[a,b]$. Now to see that $S'(i,f) \Vdash_c^{\gamma^{\mu}} S(i,f)$. Let

$x \in S(a)$ be given and $y \in S'(a)$ such that $y \Vdash_c^{\gamma^\mu} x$, that means $y = \mu_{S(a)}(x)$. Then $y \in S'(\mathrm{dom}(i))$ and the diagram

$$\begin{tikzcd}
S\big(\mathrm{dom}(i)\big) \arrow[r, "S(f)"] \arrow[d, "\mu_{S(a)}"'] & S(b) \arrow[d, "\mu_{S(b)}"] \\
S'\big(\mathrm{dom}(i)\big) \arrow[r, "S'(f)"'] & S'(b)
\end{tikzcd}$$

commutes. This already yields that $S(f)(y) \Vdash_b^{\gamma^\mu} S(f)(x)$. Q.E.D.

Schematically one can visualize this assignment as follows:

$$\begin{tikzcd}
\mathscr{C} \arrow[d, bend right=60, "S"'] \arrow[d, bend left=60, "S'"] \arrow[d, phantom, "\overset{\mu}{\Longrightarrow}"] & & \mathbf{CM}^B(\mathscr{C};S) \arrow[d, "*"] \arrow[d, phantom, "\longmapsto\ \boldsymbol{\gamma}^\mu"', pos=0.5] \\
\mathsf{Sets} & & \mathbf{CM}^Y(\mathscr{D};R).
\end{tikzcd}$$

Here, the "$*$" denotes that the simulation $\boldsymbol{\gamma}^\mu$ is of a special kind, a so called "natural simulation". This notion has been introduced in [**PetStrict**]. We briefly recall this definition, but do not go further into details.

Definition 10.7.10 (Natural simulations). Let $\mathbf{C}$ and $\mathbf{C}'$ be computability models over the class T . A *natural simulation* $\boldsymbol{\gamma}\colon \mathbf{C} \mathrel{-\!\!\triangleright} \mathbf{C}\prime$ is a simulation $\boldsymbol{\gamma} = (\mathrm{id}_T, (\Vdash_\tau^\gamma)_{\tau \in T})$, such that the following properties are met.

(`NatSim`$_1$) For each $\tau \in T$ there exists $\gamma_\tau^*\colon \mathbf{C}(\tau) \to \mathbf{C}'(\tau)$ such that for all $y \in \mathbf{C}'(\tau)$ and all $x \in \mathbf{C}(\tau)$ the equivalence
$$y \Vdash_\tau^\gamma x \Leftrightarrow y = \gamma_\tau^*(x)$$
holds.

(`NatSim`$_2$) For all $\sigma, \tau \in T$ and all $f \in \mathbf{C}[\sigma,\tau]$ there exists $f' \in \mathbf{C}'[\sigma,\tau]$ such that forall $x \in \mathbf{C}(\sigma)$, such that $x \in \mathrm{dom}(f)$ holds that $\gamma_\sigma^*(x) \in \mathrm{dom}(f')$ and $f'(\gamma_\sigma^*(x)) = \gamma_\tau^*(f(x))$.

If additionally the following property is fulfilled, $\boldsymbol{\gamma}$ is called a *full natural simulation.*

(`NatSim`$_3$) For each $\tau \in T$ the map γ_τ^* is a surjection.

It may appear as if a natural simulations and natural transformations are "essentially the same", as in that each natural simulation between canonical computability models $\mathbf{CM}^B(\mathscr{C};S)$ and $\mathbf{CM}^{B'}(\mathscr{C}';S')$ must come from a natural transformation (and the reverse direction is lemma 10.7.9), but this is not true. To see this, we consider the following example.

Remark 10.7.11. Consider the category given by the following diagram of objects and arrows:

$$\begin{tikzcd}
a \arrow[r, bend left=60, "f_1"] \arrow[r, bend left=20, "f_2"] & b \arrow[l, bend left=20, "g_2"] \arrow[l, bend left=60, "g_1"] \arrow[loop right, "q"]
\end{tikzcd}$$

where $f_1 \neq f_2$ as well as $g_1 \neq g_2$ and

$$f_1 \circ g_1 = \mathbf{1}_b = f_1 \circ g_2 = f_2 \circ g_2,$$

$$\begin{aligned} f_2 \circ g_1 &= q, \\ q^2 = \mathbf{1}_b, g_2 \circ q = g_1, g_1 \circ q = g_2, q \circ f_1 &= f_2, q \circ f_2 = f_1, \\ g_1 \circ f_1 = g_1 \circ f_2 = g_2 \circ f_1 &= g_2 \circ f_2 = \mathbf{1}_a. \end{aligned}$$

Let the natural transformation $\boldsymbol{\gamma}\colon \mathbf{CM}^{\mathrm{tot}}(\mathscr{C}; \mathrm{Hom}(a, \text{-})) \mathrel{-\!\!\triangleright} \mathbf{CM}^{\mathrm{tot}}(\mathscr{C}; \mathrm{Hom}(b, \text{-}))$ by given by

$$\begin{aligned} \gamma_a^*\colon \mathrm{Hom}(a, a) &\to \mathrm{Hom}(b, a), \mathbf{1}_a \mapsto g_1, \\ \gamma_b^*\colon \mathrm{Hom}(a, b) &\to \mathrm{Hom}(b, b), f_1 \mapsto f_2 \circ g_1, f_2 \mapsto f_1 \circ g_1. \end{aligned}$$

Then $(\gamma_a^*)_{a \in \mathscr{C}_0}$ is not a natural transformation $\mathrm{Hom}(a, \text{-}) \Rightarrow \mathrm{Hom}(b, \text{-})$.

In the following $[\mathscr{C}, \mathsf{Sets}]^{\mathtt{pull}}$ is the category of pullback-preserving presheaves with codomain $\mathscr{C}$. The following result follows with a straightforward use of 10.7.9:

Proposition 10.7.12. *Let $(\mathscr{C}, B)$ be a category with a base of computability. We have a functor $\mathrm{CM}^B(\mathscr{C}; \text{-})\colon [\mathscr{C}, \mathsf{Sets}]^{\mathtt{pull}} \to \mathsf{CompModNat}$, defined through the following clauses:*

- *On the level of objects we define $\mathrm{CM}^B(\mathscr{C}; \text{-})(S) = \mathbf{CM}^B(\mathscr{C}; S)$.*
- *On the level of morphisms we define $\mathrm{CM}^B(\mathscr{C}; \text{-})(\mu\colon S \Rightarrow S') = \boldsymbol{\gamma}^\mu$, where $\boldsymbol{\gamma}^\mu$ is defined as in lemma 10.7.9.*

We write $\mathrm{CM}^B(\mathscr{C}; \mu)$ instead of $\mathrm{CM}^B(\mathscr{C}; \text{-})(\mu)$.

The preceding assignment can be easily generalised. For this, we consider the case where the category $\mathscr{C}$ is fixed, but the computability bases on $\mathscr{C}$ can vary, to be precise we consider $(\mathscr{C}, B)$ and $(\mathscr{C}, B')$ where B, B' are bases of computability on $\mathscr{C}$. Additionally we demand the existence of a computability transfer $F\colon (\mathscr{C}, B) \to (\mathscr{C}, B')$. Then we have the situation

$$\begin{tikzcd} \mathscr{C} \arrow[rr, "F"] \arrow[dr, "S"'] & & \mathscr{C} \arrow[dl, "S'"] \\ & \mathsf{Sets}, & \end{tikzcd}$$

where this diagram *does not* commute. At last, we demand the existence of a natural transformation $\eta\colon S \Rightarrow S' \circ F$. Then we want to find a simulation $\mathbf{CM}^B(\mathscr{C}; S) \mathrel{-\!\!\triangleright} \mathbf{CM}^{B'}(\mathscr{C}; S')$. This can be done in the following way:

Proposition 10.7.13. *Let $\mathscr{C}$ be a category with bases of computability B, B' and presheaves $S, S'\colon \mathscr{C} \to \mathsf{Sets}$. For any computability transfer $F\colon (\mathscr{C}, B) \to (\mathscr{C}, B')$ we obtain a simulation $\boldsymbol{\gamma}^{F,\eta}\colon \mathbf{CM}^B(\mathscr{C}; S) \mathrel{-\!\!\triangleright} \mathbf{CM}^{B'}(\mathscr{C}; S')$ defined through the following clauses:*

- *On the level of the class function we have $\gamma := F_0$.*
- *On the level of relations we have for each $c \in \mathscr{C}_0$ the relation $\Vdash_c^{\gamma^{F,\eta}} \subseteq S'(F(c)) \times S(c)$ via the following equivalence:*
$$(y \Vdash_c^{\gamma^{F,\eta}} x) :\Leftrightarrow (y = \eta_c(x)).$$

This assignment rule can again be packaged into a functor from a suitable category, which is the following

Definition 10.7.14 (The category $\lfloor \mathscr{C}, \mathsf{Sets} \rfloor^{\mathtt{pull}}$). The category $\lfloor \mathscr{C}, \mathsf{Sets} \rfloor^{\mathtt{pull}}$ is defined through the following data:

- The *objects* of $\lfloor \mathscr{C}, \mathsf{Sets} \rfloor^{\mathtt{pull}}$ are pairs (B, S); where B is a base of computability on $\mathscr{C}$ and $S\colon \mathscr{C} \to \mathsf{Sets}$ is a pullback-preserving functor.

- A *morphism* $(F, \eta)\colon (B, S) \to (B', S')$ consists of a computability transfer $F\colon (\mathscr{C}, B) \to (\mathscr{C}, B')$ and a natural transformation $\eta\colon S \Rightarrow S' \circ F$.
- The composition is defined as $(G, \nu) \circ (F, \mu) = (G \circ F, (\mathbf{1}_F \star \nu) \circ \mu)$, where $\star$ signifies the horizontal composition.

Then the asignment rule of lemma 10.7.13 is the following a functor.

Proposition 10.7.15. $\mathrm{CM}^{\text{-}}(\mathscr{C};\, \text{-})\colon \lfloor \mathscr{C}, \mathsf{Sets} \rfloor^{\mathtt{pull}} \to \mathsf{CompMod}$ *defined via the following clauses:*

- *On the level of objects we define* $\mathrm{CM}^{\text{-}}(\mathscr{C};\, \text{-})(B, S) = \mathbf{CM}^B(\mathscr{C}; S)$.
- *On the level of morphisms we define* $\mathrm{CM}^{\text{-}}(\mathscr{C};\, \text{-})(F, \eta) = \boldsymbol{\gamma}^{F,\eta}$ *as defined in proposition 10.7.13.*

is a functor.

We can finally sum up our functors in the following diagram with the obvious inclusions:

$$\begin{tikzcd}
{[\mathscr{C}, \mathsf{Sets}]^{\mathtt{pull}}} \arrow[r, "\mathrm{CM}^B(\mathscr{C};\text{-})"] \arrow[d, hook] & \mathsf{CompModNat} \arrow[d, hook] \\
\lfloor \mathscr{C}, \mathsf{Sets} \rfloor^{\mathtt{pull}} \arrow[r, "\mathrm{CM}^{\text{-}}(\mathscr{C};\text{-})"'] & \mathsf{CompMod}
\end{tikzcd}$$

Together with the already established functors we obtain

$$\begin{tikzcd}
\mathsf{CatBasePshv}^{\mathrm{id}} \arrow[r, "\mathrm{CM}^{\text{-}}(\text{-};\text{-})"] \arrow[d, hook, "\iota"] & \mathsf{CompModEq} \arrow[d, hook, "\imath ncl"] & \\
\mathsf{CatBasePshv} \arrow[r, "\mathrm{CM}^{\text{-}}(\text{-};\text{-})"] & \mathsf{CompMod} & \lfloor \mathscr{C}, \mathsf{Sets} \rfloor^{\mathtt{pull}} \arrow[l, "\mathrm{CM}^{\text{-}}(\mathscr{C};\text{-})"] \\
& \mathsf{CompModNat} \arrow[u, hook] & {[\mathscr{C}, \mathsf{Sets}]^{\mathtt{pull}}} \arrow[l, "\mathrm{CM}^B(\mathscr{C};\text{-})"] \arrow[u, hook]
\end{tikzcd}$$

Backmatter

Glossary …

… of (2-)categories

2-depC	*objects:* 2-dep-categories (see 6.1.1) *1-maps:* 2-dep-functors (see 6.1.2) *2-maps:* 2-dep-natural transformations (see 6.1.3)	57
2-famC	*objects:* 2-fam-categories (see 2.3.1) *1-maps:* 2-fam functors (see 2.3.3) *2-maps* 2-fam-natural transformations (see 2.3.4)	19
$(\mathsf{2\text{-}fam},\Sigma)\mathsf{C}$	*objects:* (2-fam,Σ)-categories (see 3.1.7) *1-maps:* (2-fam,Σ)-functors (see 3.1.11) *2-maps:* (2-fam,Σ)-natural transformations (see 3.1.12)	31
$(\mathsf{2\text{-}fam},\Sigma)\mathsf{C}^{\mathrm{wk}}$	*objects:* (2-fam,Σ)-categories (see 3.1.7) *1-maps:* *weak* (2-fam,Σ)-functors (see 3.1.11) *2-maps:* (2-fam,Σ)-natural transformations (see 3.1.12)	31
ComprC	*objects:* comprehension categories (see 3.2.1) *1-maps:* pseudo-maps of comprehension categories (see 3.2.2) *2-maps:* transformations of pseudo-maps (see 3.2.5	34
$\mathsf{ComprC}_{\mathrm{disc}}$	*objects:* *discrete* comprehension categories (see 3.2.1) *1-maps:* pseudo-maps of comprehension categories (see 3.2.2) *2-maps:* transformations of pseudo-maps (see 3.2.5)	34
$\mathsf{ComprC}_{\mathrm{spl}}$	*objects:* *split* comprehension categories (see 3.2.1) *1-maps:* *split* pseudo-maps of comprehension categories (see 3.2.2) *2-maps:* transformations of pseudo-maps (see 3.2.5)	34
$\mathsf{ComprC}^{\mathrm{str}}$	*objects:* comprehension categories (see 3.2.1) *1-maps:* *strict* maps of comprehension categories (see 3.2.2) *2-maps:* transformations of strict maps (see 3.2.5)	34
$\mathsf{CompMod}_{\mathrm{eq}}$	*objects:* computability models (see 8.1.1) *1-maps:* *equality* simulations (see 8.2.3) *2-maps:* transformatbility of simulations (see 8.3.1)	89
CompMod	*objects:* computability models (see 8.1.1) *1-maps:* simulations (see 8.2.1) *2-maps:* transformatbility of simulations (see 8.3.1)	89
$\mathsf{CompMod}_{\mathrm{nat}}$	*objects:* computability models (see 8.1.1) *1-maps:* *natural* simulations (see 8.2.5) *2-maps:* transformatbility of simulations (see 8.3.1)	89

$\mathsf{ComprC}_{\mathrm{Mon}}$	*objects:* Comprehension categories with all $P(e)$ monic *1-maps:* same as in ComprC *2-maps:* same as in ComprC	120
depC	*objects:* dep-categores (see 4.1.1) *1-maps:* dep-functors (see 4.1.8 *2-maps:* dep-natural transformations (see 4.1.8)	46
$(\mathsf{dep}, \Sigma)\mathsf{C}$	*objects:* (dep,Σ)-categories (see 5.1.1 *1-maps:* (dep,Σ)-functors (see 5.1.2) *2-maps:* (dep,Σ)-natural transformations (see 5.1.4)	53
dFib	*objects:* discrete fibrations (see 2.2.1) *1-maps:* pseudo-maps of discrete fibrations (see 2.2.2) *2-maps:* transformations of pseudo-maps (see 2.2.2)	16
famC	*objects:* categories with family arrows (see 2.1.1) *1-maps:* family functors (see 2.1.4) *2-maps:* fam-natural transformations (see 2.1.4)	15
$(\mathsf{fam}, \Sigma)\mathsf{C}$	*objects:* (fam,Σ)-categories (see 3.1.1) *1-maps:* (fam,Σ)-functors (see 3.1.4) *2-maps:* (fam,Σ)-natural transformations	30
Fib	*objects:* Grothendieck fibrations (see 2.4.1) *1-maps:* maps of Grothendieck fibrations (see 2.4.5) *2-maps:* transformations of pseudo-maps (see 2.4.10)	23
$\mathsf{Fib}_{\mathrm{spl}}$	*objects:* *split* fibrations (see 2.4.7) *1-maps:* maps of *split* fibrations (see 2.4.9) *2-maps:* transformations of (split) maps (see 2.4.10)	23
$\mathsf{HComprC}_{\mathrm{disc}}$	*objects:* Higher discrete comprehension categories (see 5.2.2) *1-maps:* Pseudo-maps (see 5.2.3) *2-maps:* Transformations of pseudo-maps (see 5.2.3)	54
$\mathsf{HComprC}_{\mathrm{disc}}^{\mathrm{str}}$	*objects:* Higher discrete comprehension categories (see 5.2.2) *1-maps:* *strict* maps (see 5.2.3) *2-maps:* Transformations of strict maps (see 5.2.3)	54
sas	*objects:* sas-towers (see 6.2.9) *1-maps:* sas-maps (see 6.2.14) *2-maps:* transformations of sas-maps (see 6.2.17)	65

. . . of binary symbols

$(\#1) \frown (\#2)$ If both $(\#1), (\#2)$ are lists $\ell^1, \ell^2 \in M^+$, the concatenated list 121

$$\ell^1 \frown \ell^2 = \ell^1_1 \cdots \ell^1_{|\ell^1|}\ell^2_1 \cdots \ell^2_{|\ell^2|}.$$

. . . of unary symbols

$(\texttt{\#1})^\circ$	if $(\texttt{\#1})$ is a list ℓ of composable arrows, the composition $\ell_{\lvert\ell\rvert} \circ \cdots \circ \ell_1$	121

Bibliography

[1] Benedikt AHRENS, Peter LEFANU LUMSDAINE, and Paige Randall NORTH. "Comparing Semantic Frameworks for Dependently-Sorted Algebraic Theories". In: vol. 15194. 2025, pp. 3–22. DOI: 10.1007/978-981-97-8943-6_1. arXiv: 2412.19946 [math] (cit. on pp. 2, 12, 32).

[2] Michael BARR and Charles WELLS. *Category Theory for Computing Science*. 2. [print.] Prentice-Hall International Series in Computer Science. London: Prentice-Hall, 1991. ISBN: 978-0-13-120486-7 (cit. on p. 5).

[3] Michael BARR and Charles WELLS. *Category Theory for Computing Science*. Reprints in Theory and Applications of Categories 22. 2020 (cit. on p. 5).

[4] Jean BÉNABOU. "Fibered Categories and the Foundations of Naive Category Theory". In: *The Journal of Symbolic Logic* 50.1 (1985), pp. 10–37. ISSN: 0022-4812. DOI: 10.2307/2273784. JSTOR: 2273784 (cit. on p. 20).

[5] Javier BLANCO. "Relating Categorical Approaches to Type Dependency". MA thesis. Nijmegen: Catholieke Universiteit Nijmegen, 1991 (cit. on pp. 2, 32, 34).

[6] Francis BORCEUX. *Basic Category Theory*. 1st ed. Vol. 1. Handbook of Categorical Algebra. Cambridge University Press, Aug. 1994. ISBN: 978-0-521-44178-0 978-0-521-06119-3 978-0-511-52585-8. DOI: 10.1017/CBO9780511525858 (cit. on p. 89).

[7] Francis BORCEUX. *Handbook of Categorical Algebra, Vol.2 Categories and Structures*. Encyclopedia of Mathematics and Its Applications 51. Cambridge: Cambridge university press, 1994. ISBN: 978-0-521-44179-7 (cit. on p. 111).

[8] A. K. BOUSFIELD. "Constructions of Factorization Systems in Categories". In: *Journal of Pure and Applied Algebra* 9.2 (Jan. 1977), pp. 207–220. ISSN: 0022-4049. DOI: 10.1016/0022-4049(77)90067-6 (cit. on p. 57).

[9] John CARTMELL. "Generalised Algebraic Theories and Contextual Categories". PhD thesis. Jan. 1978 (cit. on pp. 2, 34).

[10] J. R. B. COCKETT and Pieter J. W. HOFSTRA. "Categorical Simulations". In: *Journal of Pure and Applied Algebra* 214.10 (Oct. 2010), pp. 1835–1853. ISSN: 0022-4049. DOI: 10.1016/j.jpaa.2009.12.028 (cit. on p. 84).

[11] J. R. B. COCKETT and Stephen LACK. "Restriction Categories I: Categories of Partial Maps". In: *Theoretical Computer Science* 270.1 (Jan. 2002), pp. 223–259. ISSN: 0304-3975. DOI: 10.1016/S0304-3975(00)00382-0 (cit. on pp. 2, 84).

[12] J. R. B. COCKETT and Stephen LACK. "Restriction Categories II: Partial Map Classification". In: *Theoretical Computer Science*. Category Theory and Computer Science 294.1 (Feb. 2003), pp. 61–102. ISSN: 0304-3975. DOI: 10.1016/S0304-3975(01)00245-6 (cit. on pp. 2, 84).

[13] J. R. B. COCKETT and Stephen LACK. "Restriction Categories III: Colimits, Partial Limits, and Extensivity". In: *Theoretical Computer Science* 270.1-2 (Jan. 2002), pp. 223–259. ISSN: 03043975. DOI: 10.1016/S0304-3975(00)00382-0. arXiv: math/0610500 (cit. on pp. 2, 84).

[14] Greta CORAGLIA and Jacopo EMMENEGGER. "A 2-Categorical Analysis of Context Comprehension". In: *Theory and Applications of Categories* 41 (2024), Paper No. 42,1476–1512. ISSN: 1201-561X (cit. on pp. 12, 32, 33, 73, 74, 77).

[15] Greta CORAGLIA and Ivan Di LIBERTI. *Context, Judgement, Deduction*. Nov. 2024. DOI: 10.48550/arXiv.2111.09438. arXiv: 2111.09438 [math] (cit. on pp. 12, 73).

[16] Pierre-Louis Curien, Richard Garner, and Martin Hofmann. "Revisiting the Categorical Interpretation of Dependent Type Theory". In: *Theoretical Computer Science* 546 (Aug. 2014), pp. 99–119. issn: 03043975. doi: 10.1016/j.tcs.2014.03.003 (cit. on p. 32).

[17] Ugo de'Liguoro, Luca Roversi, and Matteo Palazzo. *Proceedings of the 25th Italian Conference on Theoretical Computer Science.* Vol. 3811. CEUR Workshop Proceedings. Torino, Italy, Sept. 2024 (cit. on pp. vi, 143).

[18] Robert A. Di Paola and Alex Heller. "Dominical Categories: Recursion Theory without Elements". In: *The Journal of Symbolic Logic* 52.3 (Sept. 1987), pp. 594–635. issn: 0022-4812, 1943-5886. doi: 10.2307/2274352 (cit. on p. 2).

[19] T. Ehrhard. "A Categorical Semantics of Constructions". In: *[1988] Proceedings. Third Annual Symposium on Logic in Computer Science.* July 1988, pp. 264–273. doi: 10.1109/LICS.1988.5125 (cit. on p. 55).

[20] Yannick Ehrhardt. "2-Dep-Categories". Bachelor Thesis. München: Ludwig-Maximilians-Universität München, 2024. url: https://www.math.lmu.de/~petrakis/Ehrhardt.pdf (cit. on pp. 2, 12, 14, 15, 18, 19, 30, 31, 66, 76, 77).

[21] S. Eilenberg et al. *Proceedings of the Conference on Categorical Algebra.* 1st ed. La Jolla: Springer, 1966. isbn: 978-3-642-99902-4 (cit. on p. 143).

[22] Samuel Eilenberg and Saunders MacLane. "General Theory of Natural Equivalences". In: *Transactions of the American Mathematical Society* 58.0 (1945), pp. 231–294. issn: 0002-9947, 1088-6850. doi: 10.1090/S0002-9947-1945-0013131-6 (cit. on p. 1).

[23] P. J. Freyd and G. M. Kelly. "Categories of Continuous Functors, I". In: *Journal of Pure and Applied Algebra* 2.3 (Sept. 1972), pp. 169–191. issn: 0022-4049. doi: 10.1016/0022-4049(72)90001-1 (cit. on p. 57).

[24] Luis Gambarte and Iosif Petrakis. "Bases of Computability". To Be Submitted (cit. on p. vi).

[25] Luis Gambarte and Iosif Petrakis. "The Grothendieck computability model". In: *[17].* Torino, Italy, 2024 (cit. on p. vi).

[26] Luis Gambarte and Iosif Petrakis. "The Grothendieck Computability Model". In: *Theoretical Computer Science* 1057 (Dec. 2025), p. 115550. issn: 0304-3975. doi: 10.1016/j.tcs.2025.115550 (cit. on p. vi).

[27] John W. Gray. "Fibred and Cofibred Categories". In: *[21].* 1966, pp. 21–83. doi: 10.1007/978-3-642-99902-4_2 (cit. on p. 20).

[28] Alexander Grothendieck. "Categories Fibrees et Descente". In: *Revêtements Etales et Groupe Fondamental.* Ed. by Alexander Grothendieck and Michèle Raynaud. Berlin, Heidelberg: Springer, 1971, pp. 145–194. isbn: 978-3-540-36910-3. doi: 10.1007/BFb0058662 (cit. on p. 19).

[29] Alexander Grothendieck. "Technique de descente et théorèmes d'existence en géométrie algébrique. I. Généralités. Descente par morphismes fidèlement plats". In: *Séminaire Bourbaki : années 1958/59 - 1959/60, exposés 169-204.* Séminaire bourbaki 5. Société mathématique de France, 1960, pp. 299–327. url: https://www.numdam.org/item/SB_1958-1960__5__299_0/ (cit. on pp. 19, 20).

[30] Martin Hofmann. "Syntax and Semantics of Dependent Types". In: *[52].* Jan. 1997, pp. 79–130. doi: 10.1017/CBO9780511526619.004 (cit. on pp. 8, 21).

[31] J. R. Isbell. "Some Remarks Concerning Categories and Subspaces". In: *Canadian Journal of Mathematics* 9 (1957), pp. 563–577. issn: 0008-414X, 1496-4279. doi: 10.4153/CJM-1957-064-6 (cit. on p. 58).

[32] Bart JACOBS. *Categorical Logic and Type Theory.* 1st ed. Studies in Logic and the Foundations of Mathematics v. 141. Amsterdam ; New York: Elsevier Science, 1999. ISBN: 978-0-444-50170-7 (cit. on p. 32).

[33] Bart JACOBS. "Categorical Type Theory". PhD thesis. Nijmegen: Katholieke Universiteit Nijmegen, Sept. 1991 (cit. on pp. 1, 27, 32, 53).

[34] Bart JACOBS. "Comprehension Categories and the Semantics of Type Dependency". 1990. URL: `https://linkinghub.elsevier.com/retrieve/pii/030439759390169T` (cit. on pp. 1, 32).

[35] Bart JACOBS. "Comprehension Categories and the Semantics of Type Dependency". In: *Theoretical Computer Science* 107.2 (Jan. 1993), pp. 169–207. ISSN: 03043975. DOI: `10.1016/0304-3975(93)90169-T` (cit. on pp. 1, 31, 32, 54).

[36] Niles JOHNSON and Donald YAU. *2-Dimensional Categories.* June 2020. arXiv: `2002.06055 [math]`. URL: `http://arxiv.org/abs/2002.06055` (cit. on p. 6).

[37] Peter T. JOHNSTONE. *Sketches of an Elephant: A Topos Theory Compendium.* Oxford Logic Guides 43-<44 >. Oxford ; New York: Oxford University Press, 2002. ISBN: 978-0-19-852496-0 978-0-19-853425-9 978-0-19-851598-2 (cit. on pp. 23, 68, 94, 99).

[38] Tom de JONG et al. *A Study of Kock's Fat Delta.* Mar. 2025. DOI: `10.48550/arXiv.2503.10963`. arXiv: `2503.10963 [math]` (cit. on p. 56).

[39] Joachim LAMBEK. "Subequalizers". In: *Canadian Mathematical Bulletin* 13.3 (Sept. 1970), pp. 337–349. ISSN: 0008-4395, 1496-4287. DOI: `10.4153/CMB-1970-065-6` (cit. on p. 112).

[40] Joachim LAMBEK and Philip J. SCOTT. *Introduction to Higher Order Categorical Logic.* Paperback ed. (with corr.), reprinted. Cambridge Studies in Advanced Mathematics 7. Cambridge: Cambridge Univ. Press, 1986. ISBN: 978-0-521-35653-4 (cit. on p. 1).

[41] F. William LAWVERE. "Adjointness in Foundations". In: *dialectica* 23.3-4 (Dec. 1969), pp. 281–296. ISSN: 0012-2017, 1746-8361. DOI: `10.1111/j.1746-8361.1969.tb01194.x` (cit. on p. 1).

[42] F. William LAWVERE. "Functorial Semantics of Algebraic Theories". In: *Proceedings of the National Academy of Sciences of the United States of America* 50.5 (1963), pp. 869–872. ISSN: 00278424, 10916490. JSTOR: `71935`. URL: `http://www.jstor.org/stable/71935` (cit. on p. 1).

[43] John LONGLEY. "Computability Structures, Simulations and Realizability". In: *Mathematical Structures in Computer Science* 24.2 (Apr. 2014), e240201. ISSN: 0960-1295, 1469-8072. DOI: `10.1017/S0960129513000182` (cit. on pp. 2, 80, 85, 87).

[44] John LONGLEY. "Realizability Toposes and Language Semantics". PhD thesis. Edinburgh: University of Edinburgh, 1994 (cit. on pp. 2, 94).

[45] John LONGLEY and Dag NORMANN. *Higher-Order Computability.* Heidelberg: Springer, 2015. ISBN: 978-3-662-47991-9 (cit. on pp. 2, 79, 87, 96).

[46] Saunders MACLANE. "Duality for Groups". In: *Bulletin of the American Mathematical Society* 56.6 (1950), pp. 485–516. ISSN: 0002-9904, 1936-881X. DOI: `10.1090/S0002-9904-1950-09427-0` (cit. on p. 58).

[47] Per MARTIN-LÖF. *Intuitionistic Type Theory.* Studies in Proof Theory Lecture Notes 1. Napoli: Bibliopolis, 1984. ISBN: 978-88-7088-105-9 (cit. on p. 21).

[48] Iosif PETRAKIS. *Categories with Dependent Arrows.* Mar. 2023. arXiv: `2303.14754 [math]`. URL: `http://arxiv.org/abs/2303.14754` (cit. on pp. 2, 12, 13, 14, 15, 27, 29, 50, 77).

[49] Iosif PETRAKIS. "Computability Models over Categories and Presheaves". In: *Logical Foundations of Computer Science.* Ed. by Sergei ARTEMOV and Anil NERODE. Cham: Springer International Publishing, 2022, pp. 253–265. ISBN: 978-3-030-93100-1. DOI: `10.1007/978-3-030-93100-1_16` (cit. on pp. 80, 109).

[50] Iosif Petrakis. “Strict Computability Models over Categories and Presheaves”. In: *Journal of Logic and Computation* 32.8 (Dec. 2022), pp. 1815–1838. issn: 0955-792X. doi: 10.1093/logcom/exac077 (cit. on pp. 2, 84, 130).

[51] Andrew M. Pitts. “Categorical Logic”. In: *Handbook of Logic in Computer Science: Volume 5: Logic and Algebraic Methods.* USA: Oxford University Press, Inc., Apr. 2001, pp. 39–123. isbn: 978-0-19-853781-6 (cit. on pp. 2, 28, 43).

[52] Andrew M. Pitts and P. Dybjer. *Semantics and Logics of Computation.* 1st ed. Cambridge University Press, Jan. 1997. isbn: 978-0-521-58057-1 978-0-521-11846-0 978-0-511-52661-9. doi: 10.1017/CBO9780511526619.004 (cit. on p. 143).

[53] John Power and Edmund Robinson. “A Characterization of Pie Limits”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 110.1 (July 1991), pp. 33–47. issn: 1469-8064, 0305-0041. doi: 10.1017/S0305004100070092 (cit. on pp. 110, 111, 115).

[54] Emily Riehl. “Algebraic Model Structures”. PhD thesis. Chicago: University of Chicago, June 2011 (cit. on p. 57).

[55] Emily Riehl. “FACTORIZATION SYSTEMS”. In: () (cit. on p. 57).

[56] Emily Riehl. *Two-Sided Discrete Fibrations in 2-Categories and Bicategories.* Dec. 2010 (cit. on pp. 47, 104).

[57] E. Robinson and G. Rosolini. “Categories of Partial Maps”. In: *Information and Computation* 79.2 (Nov. 1988), pp. 95–130. issn: 0890-5401. doi: 10.1016/0890-5401(88)90034-X (cit. on p. 5).

[58] Giuseppe Rosolini. “Continuity and Effectiveness in Topoi”. PhD thesis. University of Oxford, Mar. 1986 (cit. on pp. 2, 109, 110).

[59] Bertrand Russell. “Mathematical Logic as Based on the Theory of Types”. In: *American Journal of Mathematics* 30.3 (1908), pp. 222–262. doi: 10.2307/2272708 (cit. on p. 1).

[60] Christian Sattler. “Kock’s fat Δ is a direct replacement of Δ”. 2017. url: https://www.cse.chalmers.se/~sattler/docs/fat-delta.pdf (cit. on p. 56).

[61] Paul Taylor. *Practical Foundations of Mathematics.* Cambridge Studies in Advanced Mathematics 59. Cambridge: Cambridge university press, 1999. isbn: 978-0-521-63107-5 (cit. on p. 68).

[62] Paul Taylor. “Recursive Domains, Indexed Category Theory and Polymorphism”. PhD thesis. Trinity College, Cambridge: University of Cambridge, July 1983 (cit. on pp. 1, 33).

[63] A. S. Troelstra. “On the Syntax of Martin-Löf’s Type Theories”. In: *Theoretical Computer Science* 51.1 (Jan. 1987), pp. 1–26. issn: 0304-3975. doi: 10.1016/0304-3975(87)90047-8 (cit. on p. 21).

[64] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics.* Institute for Advanced Study: https://homotopytypetheory.org/book, 2013 (cit. on pp. 1, 2, 8, 9, 10, 14, 29).

[65] Liang Ze Wong. “The Grothendieck Construction in Enriched, Internal and ∞-Category Theory”. PhD thesis. University of Washington, 2019 (cit. on p. 104).

Index

The page of the definition of a concept is set in *italic* numbers. People are sorted by family names, which are set in SMALL CAPITALS.

Preliminary draft

Typeset using the memoir class
↳`https://www.ctan.org/pkg/memoir`
Illustrations created using Ti*k*Z & PGF
↳`https://www.ctan.org/pkg/pgf`
Diagrams created using Ti*k*Z-cd
↳`https://www.ctan.org/pkg/tikz-cd`

compiled with pdfLaTeX
03.01.2026, 16:20

Eidesstattliche Versicherung

nach der Promotionsordnung vom 12.07.2011, §8, Absatz 2, Punkt 5

Hiermit versichere ich an Eides statt, dass die vorliegende Arbeit von mir selbstständig ohne unerlaubte Beihilfe angefertigt wurde.

München, den.....................

Gambarte, Luis Antonio

....

⟨Name, Vorname Doktorand⟩ ⟨Unterschrift Doktorand⟩