Approximation as Differentiation notes

1 OVERVIEW

Some notes on the key constructions:

- Categories of families, special case of Grothendieck construction

Set will often be **Setoid** in the Agda code but I'm glossing that here.

2 DEFINITIONS

2.1 Indexed families of objects

For any set I and category C write $\mathbf{Fam}(I, C)$ for the category where:

- the objects are the I-indexed families of objects of C;
- a morphism from X to Y is a family of morphisms $f_i : X_i \to Y_i$ in C for any $i \in I$.

Equivalently, $\mathbf{Fam}(I, C)$ is the functor category $[I, C]$.
Suppose $x \in \mathbf{Fam}(I, C)$ and $y \in \mathbf{Fam}(J, C)$. Define y_f for any $f : I \to J$ to be the I-indexed family of objects of C.

2.2 Category of families

For a functor $F : C \to Set$, we have the category where:

- objects are pairs (I, x) with I an object of C and $x \in FI$
- morphisms from (I, x) to (J, y) are morphisms $f : I \to J$ in C where $(Ff)(x) = y$

This is the Grothendieck construction for a functor $F : C \to \mathbf{Cat}$, in the special case where $F : C \to \mathbf{Set}$. (Where we read a **Set**-valued functor as a **Cat**-valued functor restricted to discrete categories.)

In our code we give a slightly different construction. For a category C, define the category where:

- objects are pairs (I, x) with I a set and x an I-indexed family of objects of C
- morphisms from (I, x) to (J, y) are functions $f : I \to J$ paired with morphisms

Author's address:

Manuscript submitted to ACM