1 Questions

1.1 What is the correct notion of parametric category?

There is a notion given by [HRR14] of reflexive graph category. I don't think this is strong enough. I would like a notion of "parametric category" that includes some axioms about having enough morphisms, so that the naturality results are as strong as possible. The morphisms are some subset of the edges and ideally we want the largest possible subset such that we can still make sense of composition and taking preimages of relations along morphisms.

One answer to this is to stipulate that the relations on Types are all of the form $f(x) = g(y)$ for some type Z functions $f : X \to Z$ and $g : Y \to Z$. This is equivalent to saying that the relation satisfies $\forall x_1 x_2 y_1 y_2, R(x_1, y_1) \to R(x_1, y_2) \to R(x_2, y_1) \to R(x_2, y_2)$. I think it is true that for any $F : Type \to Type$, if $R : X \to Y \to Prop$ satisfies the above condition then so does the corresponding relation on $F(X)$ and $F(Y)$.

Then a "parametric category" could be defined to be a category with binary pullback, such that the "edges" between objects were the pullbacks. This is better than defining an edge to be a subobject of the product both because it is simpler in a pure categorical language and because not all categories generated by parametricity have products but I cannot think of any that do not have binary pullbacks.

e.g. The category $\Sigma X : Type, X \times (X \to bool)$ has pullbacks but not products.

1.2 How to put category structure on each type?

In order to do Sigma types it is necessary to define something a little more general than a category. For example a morphism of groups $\Sigma G : Type, group(G)$ should be a morphism of types, and a "morphism" on the corresponding groups structure. i.e. if $G, H : Type$ and $g : group(G)$ and $h : group(H)$, then a morphism is a morphism between G and H and a morphism between g and h, but g and h have different types, so it does not make sense to ask what a morphism is without a more general notion of category.

1.2.1 Case 1 : Type -¿ Type

This question is hard to answer on types where $Type$ or $Prop$ appears on the left of a Pi, e.g. in the Type $Type \to Type$. One natural suggestion is to say that a morphism between F and G has type $\Pi XY, (X \to Y) \to (F(X) \to G(Y))$. The problem with this definition is that the morphisms are not composable unless F and G are functors and there is no identity on F unless F is a functor. The definition $\Pi X, F(X) \to G(X)$, seems to make more sense and these two definitions are equivalent (by parametricity) if F and G are functors.

Possibly it is morally wrong to attempt to define a category structure on the whole of $Type \to Type$ and the category should just be defined on the functors. There are no real life examples of categories with $Type$ or $Prop$ appearing on the left of a Pi anyway other than functor categories.

Potentially the parametricity translation could be changed as well when computing the "parametric category" structure of a Type in a similar way to the way arrows are computed. e.g. $Type \to Type$ could be translated as $\Pi X, F(X) \to G(X) \to Type$ when $F, G : Type \to Type$. In this case there would no longer be any content to the parametricity theorem, but you wouldn't use this translation all the time.

1.2.2 Case 2

Another difficult case is how to put the category structure on $\Sigma X : Type, ((X \to X) \to X)$. Use notation $\spadesuit$ for the operation $(X \to X) \to X)$. The natural definition is a map $f : X \to Y$ such that $\forall a : X \to X, \forall (b : Y \to Y), f \circ a = b \circ f \to f(\spadesuit a) = \spadesuit b$. This definition does not give composable morphisms. For this example, there is no obivous way to make relations into objects in the category, unlike the case of groups/rings etc.

A definition that might work would be for a morphism to be a pair of a function $f : X \to Y$ and $f_2 : (X \to X) \to (Yrightarrow Y)$ such that f_2 as a relation is the lift of f. These morphisms are probably composable but the proof is hard.

I can probably find some meaningful definition in this case, but does that mean I should and how do I demonstrate that this is the "correct" definition?

1.3 What properties does the category structure need to have?

In order for parametricity to imply naturality, given a type $\mathcal{C}$ and $X, Y : \mathcal{C}$ the type $\mathrm{Hom}(X, Y)$ must induce a map of relations from $[[\mathcal{C}]]YZ$ to $[[\mathcal{C}]]XZ$, the "preimage". There should be some partial order on the set of relations $[[\mathcal{C}]]YZ$, and the parametricity translation of $\lambda XY : \mathcal{C}, \mathrm{Hom}(X, Y)$ should be something like $\lambda(X_1X_2 : \mathcal{C})(R_X : [[\mathcal{C}]]X_1X_2)(Y_1Y_2 : \mathcal{C})(R_Y : [[\mathcal{C}]]Y_1Y_2)(f_1 : X_1 \to Y_1)(f_2 : X_2 \to Y_2), X_R \leq (f_1, f_2)^{-1}Y_R$.

1.4 What are the applications of all this?

Automatically proving naturality of things. But where does this come up, there are probably many examples of mathematical arguments that are essentially naturality even though you would never use the word naturality when talking about it.

Automatically prove functorality.

Arguments of the form, it suffices to prove this lemma on a free object. This is kind of what the ring tactic is doing.

1.5 Find the a notion of morphism of a parametric category weaker than functor, but still having some structure on the map of relations

There is a notion of a dinatural or extranatural transformation between morphisms $FG : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$. They both give a notion of naturality for something $\Pi X, F(X, X) \to G(X, X)$. However not every function $T : Type \to Type$ can be written in the $T(X) = F(X, X)$ where F is a functor $Type^{op} \times Type \to Type$ so this is not general enough. However it may be true that instead of functors $Type^{op} \times Type \to Type$ you could do include a relation between the two types, using $\Sigma AB : Type, A \to B \to Prop$, instead of $Type^{op} \times Type$. This category is completely defined in edge_category.lean. It seems that there $group : Type \to Type$ can be written as a functor $group2 : (\Sigma AB : Type, A \to B \to Prop) \to Type$, such that $groupG = group2(G, G, eq)$. It is not clear if this generalizes or if it is useful.

1.6 Generalizing representable functors

If $F : \mathcal{C} \to Type$ is a functor, then one way of defining a representation of that functor is that it is an initial object in the category $\Sigma X : \mathcal{C}, F(X)$. This definition has the advantage that F need not be a functor. How do these generalized universal properties work?

Sometimes something has a universal property in one category and this means it is "obvious" that it has some universal property in another category. For example, it is "obvious" that the initial object in the category of pointed groups is also the free group on one generator. Can parametricity be used to generate all the equivalent UMPs of an object in different categories?

1.7 What does parametricity have to do with recursion principles for inductive types?

The arity one translation of the type $\Sigma X : Type, X \times (X \to X)$ is

```
sigma (X : Type) (X_P : X -> Type) (x : X) (h0 : X_P x)
  (f : X -> X), forall H : X, X_P H -> X_P (f H)
```

References

[HRR14] Claudio Hermida, Uday S. Reddy, and Edmund P. Robinson, Logical relations and parametricity - a reynolds programme for category theory and programming languages, Electron. Notes Theor. Comput. Sci. **303** (2014), 149–180.