How do we formalize a proof in Category Theory? What is the correct level of detail? Which entities should we introduce first?

Well, this depends on our target audience; if we are speaking to a Category Theorist then we may start by saying just, for example, “let $f$ be a morphism”, and it will be understood that we have two objects, $\mathrm{Dom}\, f$ and $\mathrm{Cod}\, f$, belonging to the same category — at this point unnamed — and that $f$ goes from $\mathrm{Dom}\, f$ to $\mathrm{Cod}\, f$; if later we say $f : A \to B$ or $A \xrightarrow{f} B$ then we will be giving better (and shorter) names for $\mathrm{Dom}\, f$ and $\mathrm{Cod}\, f$, but the category where $A$ and $B$ live may remain unnamed for a while more...

If we are talking to a proof assistant — Coq, say — instead of to a human then we are forced to decaler our entities in a certain order, and to name all of them. For example:

```
Variable CatC : Categories.
let (C_0, Hom_C, id_C, o_C, idL_C, idR_C, assoc_C) := CatC in
  Variable A B : C_0, f : Hom_C A B.
  ...
end.
```

In this note we will show how to formalize some constructions and proofs in a proof assistant in a way that:

1) lets us choose just a very few names,

2) lets us use names that are very close to a certain graphical notation,

3) lets us split our constructions and proofs in two layers, or parts: a “syntactical” part, that must necessarily come first, and a “logical” part,

4) lets us build easily dictionaries between several standard notations.

We will say that a construction (or proof) that has both its syntactical part and its logical part is happening in the “real world”; by dropping its logical part and keeping just its syntactical part we obtain a corresponding construction in the “syntactical world”. We will call this passage from the real world to the syntactical world a “projection” — as projections discard some information (intuitively coordinates, or components) and forget some distinctions. The opposite operation is a “lifting”: we may start with a syntactical construction or proof, and then try to lift that to the real world. The “projection” direction is easy, and we can always be done (sec. _; explain abelian categories, and which “always” is that); the “lifting” direction is hard, and I don’t even know how to characterize when a given lifting can be done; see the list of problems in sec. _.

The plan of this paper is as follows. In sec. _ we define “category”, “proto-category”, “isomorphism”, “proto-isomorphism”, etc, in the right way (for our purposes!). In sec. _ we explain a trick to make Coq accept our notation; in sec. _ we present an example: a syntactical proof of the Yoneda Lemma. In sec. _ we present a system of Natural Deduction for (proto-)categories, and in sec _ we sketch how it can be extended to a system of Natural Deduction for dependent types. Section _ discusses open problems and directions for future work.