CCC-Semantics

This is a formalisation of the interpretation of the simply-typed lambda calculus into cartesian closed categories.

Structure

The project is split into two parts.

- `CCCSemantics.Lambda`, which contains formalisation of the syntax, i.e. simply typed lambda calculus, together with substitutions.
- `CCCSemantics.Categories`, a small library for category theory, formalising up to and including the Yoneda lemma, as well as the notion of CCCs.

`CCCSemantics.Lambda`

`CCCSemantics.Categories`

The category theory library defines categories (`Categories/Category.lean`) using a *bundled* representation, rather than following Mathlib's definition using type-classes. For better inference of operators in specific choices of categories, such as product categories, we use `unif_hint`s. This is not necessarily a better choice, but was more an experiment to see how well such a library would work.

Functors (`Categories/Functor.lean`) and natural transformations (`Categories/NaturalTransformation.lean`) are defined as standard, and by making use of the bundled representation we are able to more easily define certain categories, such as `Cat` the category of categories (`Categories/Instances/Cat.lean`).

We also define functor categories (`Categories/Instances/Func.lean`) and the category of types (`Categories/Instances/Types.lean`), so we have the category of presheaves. This then allows us to construct the Yoneda-embedding, which we make use of to show that presheaf categories are cartesian closed (`Categories/CartesianClosed/Presheaf.lean`). We prove the Yoneda lemma as a nice result, but it is not used for the rest of the formalisation.

We also define adjunctions (`Categories/Adjunction.lean`), using the unit-counit definition. There is a construction of adjunctions from the definition of naturally hom-set `Hom (F -) -` and `Hom - (G -)` as well, which is used to construct the hom-product adjunction for cartesian closed categories.

For cartesian categories (`Categories/CartesianClosed.lean`) we use the elementary definition, requireing an assignment for each pair of objects to a product, defined explicitly, rather than through a general interface for limits. This is defined as a type-class indexed by categories. We define cartesian closed category as an extended type-class on cartesian categories, additionally having an assignment of exponentials to each pair of objects. We show how both the assigment of products and of exponentials give functors, and that these are adjoint as expected. We

show that the category of types (`Categories/CartesianClosed/Types.lean`) and, as mentioned earlier, presheaf categories, are cartesian closed.