Project report : INF551

Riade Benbaki

6 December 2020

I - Content of project

I.A - Parts 1,2,3 (file : *prover.ml*)

In the file *prover.ml* the first three parts of the TD are completed. Namely, implementing simply types lambda calculus, with all the connectives for conjunction, disjunction, truth and falsity, in addition to the base connectives of $\lambda - calculus$. Building on this, all the necessary proof strategies were added to the interactive prover, and then used to prove the formulas asked in the TD.
When using the interactive prover, I added an extra first step to specify the filename in which to save the proof before specifying the formula.

I.B - Dependent types (file : *dependent_types.ml*)

In this part, we implement dependent lambda types, starting by the constructors, then normalization and type inference. Again, with the interactive prover, we some formulas. The proofs of theses formulas can be found in a file with the same name as the formula.
We first define the predecessor ($pred\_5\_12.proof$) and addition functions ($add\_5\_12.proof$)
We then prove some properties using equality, such as the compatibility of the successor function ($Seq.proof$), and neutrality of zero for addition ($addz.proof$ and $zadd.proof$).
Proving associativity and commutativity of addition proved to be the most challenging part for me, because it revelled several bugs in the previous code (normalization and type inference for induction and equality), and proved to be a more delicate work than when it's done in Agda, with the holes system.
The proofs of part 5.14 can all be found in the file $514.proof$ (it made little sense to put every function in a file because of dependencies).
Proving associativity of addition was done using induction directly, but for commutativity, it was necessary to prove the compatibility of addition and successor : add (S n) m = S (add n m) (The way I defined addition makes the other way immediate via β-reduction), and the transitivity of equality.
The same can be done for multiplication, which was also defined in $514.proof$.
To prove it's associativity, we first prove it's distributivity over addition, using compatibility of equality with addition : $x = y \implies x + a = y + a$

II - What is working and what is not working

All the parts of the TD are working, along with the proofs in the .proof files, apart from the last distributivity proof in 514.*proof*, which I didn't have enough time to debug (but kept it because I think the idea is correct).

III - Improvements

The main improvement I would do concern the prover in the second part. I would make it more interactive and with more support to modularity (because the formulas we write tend to get quite big), making it easier to introduce connectives, similarly to how Agda refines formulas (C-c C-r). I would also add support for implicit arguments, because that should make formulas more readable and shorter.
Another thing to improve is exception handling, as it would help to make error messages more explicit, specially in type inference when types don't match (this is something I have partially done when debugging parts of my code).