

Dependent Types Section and Practice Problems

Apr 10–13, 2018

This week is actually a good opportunity to look back at previous section notes and material, and make sure you are comfortable with the material. This is because we don't expect you to be deeply familiar with the technical material on dependent types, nor are you required to be expert in Dafny or Coq.

1 Dependent Types

- (a) Assume that `boolvec` has kind $(x : \mathbf{nat}) \Rightarrow \mathbf{Type}$ and `init` has type $(n : \mathbf{nat}) \rightarrow \mathbf{bool} \rightarrow \mathbf{boolvec} \ n$.

Show that the expression `init 5 true` has type `boolvec 5`,

That is, prove

$$\Gamma \vdash \text{init } 5 \ \text{true} : \mathbf{boolvec} \ 5$$

where

$$\Gamma = \mathbf{boolvec} :: (x : \mathbf{nat}) \Rightarrow \mathbf{Type}, \text{init} : (n : \mathbf{nat}) \rightarrow \mathbf{bool} \rightarrow \mathbf{boolvec} \ n.$$

- (b) Show that the types `boolvec (35 + 7)` and `boolvec (($\lambda y : \mathbf{nat}. y$) 42)` are equivalent.

That is, prove that

$$\Gamma \vdash \mathbf{boolvec} \ (35 + 7) \equiv \mathbf{boolvec} \ ((\lambda y : \mathbf{nat}. y) \ 42) :: \mathbf{Type}$$

where

$$\Gamma = \mathbf{boolvec} :: (x : \mathbf{nat}) \Rightarrow \mathbf{Type}.$$

- (c) Suppose we had a function `double` that takes a `boolvec` and returns a `boolvec` that is twice the length.

Write an appropriate type for `double`. (Note that you will need make sure that the type of the `boolvec` argument is well formed! Hint: take a look at the type of `join`, mentioned in the Lecture 20 notes, for inspiration.)

2 Coq and Dafny (Optional!)

If you are interested, you can play around with Dafny online at <https://rise4fun.com/dafny>. A tutorial (on which the class demo was based) is available at <https://rise4fun.com/Dafny/tutorial/Guide>.

The Coq website is <https://coq.inria.fr/>. The easiest way to install Coq is via `opam`, OCaml's package manager. See <https://coq.inria.fr/opam/www/using.html>. In lecture, Prof. Chong was using Proof General (an extension to Emacs) to interact with Coq: <https://proofgeneral.github.io/>.

The Software Foundations series (<https://softwarefoundations.cis.upenn.edu/>) is a programming-languages oriented introduction to using Coq.