Harvard School of Engineering and Applied Sciences — CS 152: Programming Languages

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 $(x\!:\!\textbf{nat}) \Rightarrow \textbf{Type}$ and init has type $(n : \textbf{nat}) \to \textbf{bool} \to \textbf{boolvec}\ n)$.
Show that the expression init 5 true has type **boolvec** 5,
That is, prove

$$\Gamma \vdash \mathsf{init}\ 5\ \mathsf{true} : \textbf{boolvec}\ 5$$

where

$$\Gamma = \mathsf{boolvec} :: (x\!:\!\textbf{nat}) \Rightarrow \textbf{Type}, \mathsf{init} : (n : \textbf{nat}) \to \textbf{bool} \to \textbf{boolvec}\ n.$$

Answer:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \mathit{init} : (n : \textbf{nat}) \to \textbf{bool} \to \textbf{boolvec}\ n} \qquad \dfrac{}{\Gamma \vdash 5 : \textbf{nat}}}{\Gamma \vdash \mathit{init}\ 5 : \textbf{bool} \to \textbf{boolvec}\ 5} \qquad \dfrac{}{\Gamma \vdash \mathit{true} : \textbf{bool}}}{\Gamma \vdash \mathit{init}\ 5\ \mathit{true} : \textbf{boolvec}\ 5}$$

(b) Show that the types **boolvec** $(35 + 7)$ and **boolvec** $((\lambda y\!:\!\textbf{nat}.\ y)\ 42)$ are equivalent.
That is, prove that

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

where

$$\Gamma = \mathsf{boolvec} :: (x\!:\!\textbf{nat}) \Rightarrow \textbf{Type}.$$

Answer: *Let T_1 be defined as*

$$\dfrac{\dfrac{}{\Gamma \vdash \textbf{boolvec} \equiv \textbf{boolvec} :: (x\!:\!\textbf{nat}) \Rightarrow \textbf{Type}} \qquad \dfrac{}{\Gamma \vdash 35 + 7 \equiv 42 :: \textbf{nat}}}{\Gamma \vdash \textbf{boolvec}\ (35 + 7) \equiv \textbf{boolvec}\ 42 :: \textbf{Type}}$$

and let T_2 be defined as

$$\dfrac{\dfrac{}{\Gamma \vdash \textbf{boolvec} \equiv \textbf{boolvec} :: (x\!:\!\textbf{nat}) \Rightarrow \textbf{Type}} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, y\!:\!\textbf{nat} \vdash y : \textbf{nat}} \qquad \dfrac{}{\Gamma \vdash 42 : \textbf{nat}}}{\Gamma \vdash (\lambda y\!:\!\textbf{nat}.\ y)\ 42 \equiv 42 :: \textbf{nat}}}{\Gamma \vdash 42 \equiv (\lambda y\!:\!\textbf{nat}.\ y)\ 42 :: \textbf{nat}}}{\Gamma \vdash \textbf{boolvec}\ 42 \equiv \textbf{boolvec}\ ((\lambda y\!:\!\textbf{nat}.\ y)\ 42) :: \textbf{Type}}$$

in

$$\dfrac{\dfrac{T_1}{\Gamma \vdash \textbf{\textit{boolvec}}\,(35+7) \equiv \textbf{\textit{boolvec}}\,42 :: \textbf{\textit{Type}}} \qquad \dfrac{T_2}{\Gamma \vdash \textbf{\textit{boolvec}}\,42 \equiv \textbf{\textit{boolvec}}\,((\lambda y\!:\!\textbf{\textit{nat}}.\,y)\ 42) :: \textbf{\textit{Type}}}}{\Gamma \vdash \textbf{\textit{boolvec}}\,(35+7) \equiv \textbf{\textit{boolvec}}\,((\lambda y\!:\!\textbf{\textit{nat}}.\,y)\ 42) :: \textbf{\textit{Type}}}$$

where here T_1 is similar to T_2 and left as an exercise to the reader.

(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.)

Answer:

$$(n\!:\!\textbf{\textit{nat}}) \to \textbf{\textit{boolvec}}\ n \to \textbf{\textit{boolvec}}\ (n+n)$$

*Note that we need to take a natural number n as an argument, in order for us to specify the type of the second argument (i.e., a boolean vector of length n, **boolvec** n).*

*If we wrote **boolvec** $n \to$ **boolvec** $(n+n)$, then n is free and the type isn't well formed. Note that **boolvec** $\to$ **boolvec** is not well-kinded.*

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.