Dependent Types and Theorem Proving: Introduction to Dependent Types

Wojciech Kołowski

March 2021

General info

- The lectures will be held weekly on Fridays.
- Don't worry if you miss a lecture – the slides are pretty massive and the talks are going to be recorded.
- Each lecture ends with some exercises which will help you familiarize yourself with F* and better understand the ideas covered in the talk.
- But you don't need to do them if you don't want to.

Plan of lectures

- Lecture 1: Programming with dependent types.
- Lecture 2: Proving theorems with dependent types.
- Lecture 3: Differences between programming and proving.
- Lecture 4: Examples of bigger programs and longer proofs.
- Lecture 5: A deeper dive into F*.

Learning outcomes

- You won't be scared of all those obscure, scary and mysterious names and notations.
- You will get basic familiarity with the ideas behind dependent types.
- You will begin to see logic and mathematics in a very different light, much closer to your day job (at least if you are a programmer working in F#).
- If you do the exercises, you will gain a basic proficiency in F*.

Introducing F*

- F* (pronounced "eff star") is a general-purpose purely functional programming language.
- Member of the ML family, syntactically most similar to F#.
- Aimed at program verification.
- Dependent types.
- Refinement types.
- Effect system.
- Not a .NET language.
- Neither compiled nor interpreted – it's a proof assistant, i.e. just a typechecker.
- To run a program, it has to be extracted to some other language, like F#, OCaml, C or WASM, and then compiled.

Don't worry, be happy, ask lots of questions

I KNOW YOU DIDN'T UNDERSTAND THE PREVIOUS SLIDE, BUT BY THE END OF THESE TALKS, YOU WILL – AND THAT'S THE POINT!

Useful F* links

- You can run F* inside your browser (and have a nice tutorial guide you): `http://www.fstar-lang.org/tutorial/`
- GitHub: `https://github.com/FStarLang/FStar`
- Homepage: `http://www.fstar-lang.org/`
- Download: `http://www.fstar-lang.org/#download`
- Papers (not approachable for ordinary mortals): `http://www.fstar-lang.org/#papers`
- Talks/presentations (more approachable): `http://www.fstar-lang.org/#talks` (some of these are quite approachable if you're interested)

Prerequisites

- To understand what we will be talking about, you should have a working knowledge of F# and the basic concepts of functional programming, namely:
- Functions as first-class citizens.
- Higher-order functions.
- Algebraic data types, including sum types and product types.
- Pattern matching.
- Recursion.
- Even if you know these, you may be unfamiliar with the particular names – for example, "sum types" is a name used in academia and Haskell, but in F# they are better known as "tagged unions".
- We will now see a snippet that shows how these things look in F*.

●

●

What does “first-class” mean?

-

Computing with types

-

-

-

Example: typesafe printf

-