Homework 8: Refinement Types

98-317: Hype for Types

Due: 27 March 2018 at 6:30 PM

1 Introduction

In lecture we talked about refinement types, types with logical predicates attached to them. In this homework, you'll review refinements and (if you choose to) learn about some real languages that use refinement types!

This homework is divided into four parts: Required, Useful, Fun, and Completely Unnecessary but Also Fun. You will receive credit for this homework if you turn in something (not necessarily something working) for the "required" portion.

Turning in the Homework You should turn in your written solutions in class on Tuesday.

2 Required

Consider the refinements on the ML functions below. You may assume that `length` and `sum` are defined in the usual way: length is the number of elements in the list and sum is all the elements in the list summed together.

```
fun hype (x : int {x | x > 0}) : bool {r | r <=> x > 0} = ...

fun for (x : int) : bool {r | r <=> x > 0} = ...

fun types (L : int list {l | length(l) > 0}) : int {r | r = sum(L) + 1} =
     case L of
         [] => 1
       | x::xs => x + types xs
```

Required Task 1 Implement the function `hype` such that it satisfies its refinements.

Required Task 2 Implement the function `for` such that it satisfies its refinements.

Required Task 3 Does `types` satisfy its refinements? If so, briefly justify why. If not, how could we modify it so that it does?

3 Useful

One language that uses refinement types is called LiquidHaskell. LiquidHaskell is an extension of Haskell with decidable refinement types. Because it's a decidable system, it requires minimal annotations and proof on the part of the programmer. This makes it more attractive than many other refinement type systems, which place a lot of the burden of proof on the programmer.

Useful Task 1 Work through as much of this LiquidHaskell tutorial as you'd like.

`http://ucsd-progsys.github.io/lh-workshop/02-refinements.html`

Another language that uses refinement types is F*. F*, unlike LiquidHaskell, does not use decidable refinements, and so often the programmer must supply extra annotations and proofs. This makes F* much more expressive than LiquidHaskell, but also more cumbersome to program with.

Useful Task 2 Work through as much of this F* tutorial as you'd like.

`https://www.fstar-lang.org/tutorial/`

4 Fun

Fun Task 1 Jean Yang, the faculty advisor for Hype for Types, is one of the creators of F*. Last year, she did some cool interviews with a whole bunch of important people in the field of programming languages and type theory. Check it out:

`https://www.cs.cmu.edu/~popl-interviews/interviews.html`

If you don't recognize any of these names, Simon Peyton-Jones is the creator of Haskell and Xavier Leroy is the creator of OCaml.

5 Completely Unnecessary but Also Fun

Unnecessary Task 1 Ranjit Jhala, one of the creators of LiquidHaskell, makes some "high quality" music videos about grad school every year. Find some of these videos and watch them.