Dependent Types and Theorem Proving: Introduction to Dependent Types

Wojciech Kołowski

March 2021

1 Refinement types

The problem of richness

- In dependently typed languages there is a lot of types.
- This is a blessing, because we can express all the complicated types and properties we need in order to guarantee correctness of our programs.
- But the richness of types also causes problems: it is often the case that there are many ways to define essentially the same type.

The various faces of a `vector` 1/2

- We have already seen two incarnations of the type `vec a n` of vectors:
- Functional, recursive: by recursion on the index.
- Inductive, intrinsic: as an inductive family.
- But there are at least two more definitions:
- Functional, nonrecursive: as a function with domain `fin n`.
- Inductive, extrinsic: as a pair of a list and a proof that it's length is equal to `n`.

The various faces of a `vector` 2/2

- To better understand what the above slide tries to tell us, let's see the various definitions in code.
- See the code snippet `Lecture1/Vectors.fst`.

●

-

-

-