Administrivia

Homework 7 is due by Thursday at 11:59PM.

I will send comments on your project proposals by the end of the week (they all look reasonable).

Case Study II: Verified Sorting

Type Theory and Mechanized Reasoning

Lecture 17

CAS CS 400

Objectives

Prove that a ***simple sorting algorithm*** actually sorts.

Demonstrate the connection between algorithms and proving things about algorithms.

Preliminary Remarks

Proofs of Correctness

Proofs of Correctness

Question. What does it mean to say that an algorithm is correct? (This is the topic of CS330)

Proofs of Correctness

Question. What does it mean to say that an algorithm is correct? (This is the topic of CS330)

There is a mathematical proof which says it does what it is supposed to.

Proofs of Correctness

Question. What does it mean to say that an algorithm is correct? (This is the topic of CS330)

There is a mathematical proof which says it does what it is supposed to.

But how do we define exactly what "*is supposed to*" *means?*

Specifications

A **specification** is a formal description of what an algorithm is supposed to do.

The specification sorting: *the output list has the same elements as the input list, and that the output has elements appears in increasing order.*

Note. Defining the *correct* specification is one of the most difficult parts of formal verification. There are usually many possible equivalent specifications, some will make your proofs more difficult.

Internal vs. External Verification

External. Write an algorithm, then prove that it is correct.

```
sort : List Nat -> List Nat

verified : (l : List Nat) -> Sorted (sort l)
```

Internal. Write an algorithm which cannot possibly be incorrect.

```
verified-sort : List Nat -> SortedList Nat

to-list : SortedList Nat -> List Nat

sort l = to-list (verified-sort l)
```

Tradeoffs

External verification is more ***natural***, but can sometimes make proving things more difficult.

Internal verification make programming more difficult, but is more ***compositional***, and can make building up programs simpler.

In reality, we want a *combination of both*.

The Goal

Define from the ground up the insertion sorting algorithm and all of the predicates we need to convince ourselves that it sorts.