4/3 - Proof Assistants

Guest lecture: Andrew

Programs vs. Proofs:

- Both have precise rules, structure, and are machine-checkable
    - Programming languages (checked by IDE, compiler)
    - Deduction systems (e.g. natural deduction) (checked by proof assistants)
- Program Types:
    - Guarantee that the results of a program is the given *type* (e.g. `Number`)
    - We have richer types (e.g. `Number -> Number`)
- Types sometimes aren't enough!
    - Functions might be not total - might throw error on some inputs
    - Might not terminate
- In proofs, we have a type analogy
    - We can prove propositions in proofs
    - If the proof "typechecks" then we have a valid proof

Inductive types:

- We can define inductive types by cases
    - These cases can refer to each other
- For example bools are defined as either true or false.

When proving in Coq:

- We can define inductive types
- We can write a lemma and prove it
- In Coq, on the right side: the top is our hypotheses (what we have)
    - The bottom is our goal (what we want to show)
- Working with Propositions:
    - There is only one proof of True
    - There are no proofs of False
        - What's a consequence of this?
            - If we take as a hypothesis that there exists some proof of False H, we can show that 0 = 1