Where are we

Today (Thursday?) is IMP’s last day in the sun. Done:

• Abstract Syntax

• Operational Semantics (large-step and small-step)

• “Denotational” Semantics

• Semantic properties of (sets of) programs

Today:

• Program equivalence under operational semantics

• Equivalence of different semantics

Mostly proofs on the board (posted later)
Equivalence motivation

- Program equivalence: code optimizer, code maintainer
- Semantics equivalence: interpreter optimizer, language designer (prove properties for equivalent semantics with easier proof)
- Both: Great practice for strengthening inductive hypothesis (you will do this again in grad school)

Warning: Proofs are easy with the right semantics and lemmas

Note: Small-step operational often harder proofs but models more interesting things
What is equivalence

Equivalence depends on *what is observable*!

- **Partial I/O equivalence** (if terminates, same *ans*)
  - *while 1 skip* equivalent to everything
- **Total I/O** (same termination behavior, same *ans*)
- **Total heap equivalence** (at termination, all (almost all) variables have the same value)
- **Equivalence plus complexity bounds**
  - Is $O(2^n^n)$ really equivalent to $O(n)$?
- **Syntactic equivalence** (perhaps with renaming)
  - too strict to be interesting
Rest of today

Examples of:

- “Local Optimizations”
- Operator properties (sequence is associative)
- “Admissable” rules (additional rules don’t change semantics)
- Equivalence of large-step and small-step expressions

Note: With nondeterminism also have may/must distinction.

We’ll just prove “one direction” of if and only if.
Take-away messages

These arguments are can be tedious but the details expose when semantics isn’t like you think it is.

Steps:

• State theorem formally

• Use intuition to strengthen induction hypothesis

• Write out the proof

In reality, there is feedback not just in these steps, but back to the semantics (to get the properties you want). That’s a good thing.