

Roadmap

- Operational Semantics
 - Large-step and Small-step varieties
 - formally defines the *operation* of a machine that executes a program
- Denotational Semantics
 - defines the mathematical object (i.e., function) that a program *denotes*
- Static Semantics (Type-theory)
 - performs a *static analysis* that prevents certain runtime errors (“stuck states”)
- Today: Axiomatic Semantics

Axiomatic Semantics

- Goal: We wish to prove program correctness
 - type-theory too weak* (just proves soundness)
 - operational semantics creates a proof as large as the runtime behavior of the program!
 - denotational semantics creates a massive mathematical object that encodes all memory states (too hard to reason about)
- Solution (Axiomatic Semantics):
 - derivation system that reduces a program to a (small) set of theorems that, if proved, would collectively imply program correctness
 - prove the resulting theorems to prove correctness

*Actually type-theory can be as strong as you wish to make your type-system. Take my course next semester to find out how to develop correctness-proving type-systems.

Two Kinds of “Correctness”

- Partial Correctness
 - Notation: $\{A\}c\{B\}$
 - If “A” is true before executing c , and if c terminates, then “B” is true after executing c .
 - A is “precondition”, B is “postcondition”
- Total Correctness
 - Notation: $[A]c[B]$
 - If “A” is true before executing c , then c eventually terminates and “B” is true once it does.

Examples

- $\{x \leq 10\}$ while $(x \leq 10)$ do $x := x + 1$ $\{x = 11\}$
- $[x \leq 10]$ while $(x \leq 10)$ do $x := x + 1$ $[x = 11]$
- $[\text{true}]$ while $(x \leq 10)$ do $x := x + 1$ $[x \geq 11]$
- $[x = \bar{i}]$ while $(x \leq 10)$ do $x := x + 1$ $[x = \max(11, \bar{i})]$
- $\{T\}$ while true do skip $\{F\}$

Language of Assertions

- First-order logic with arithmetic:

IMP arithmetic exprs $a ::= i \mid v \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2$

assertion exprs $e ::= a \mid \bar{v}$

assertions $A ::= T \mid F \mid e_1 = e_2 \mid e_1 \leq e_2 \mid A_1 \wedge A_2 \mid$
 $A_1 \vee A_2 \mid \neg A \mid A_1 \Rightarrow A_2 \mid \forall \bar{v}. A \mid \exists \bar{v}. A$

- From these one can construct all functions and logical operators, so we will freely use extensions to the above.

Hoare Logic

- First published by Tony Hoare [1969]
 - First and most famous axiomatic semantics
 - “An axiomatic basis for computer programming”
 - Often cited as one of the greatest CS papers of all time (only 6 pages long!)
 - Optional: read the original paper (linked from course website)
- Adaptation to IMP consists of...
 - six axioms (rules) describing IMP programs
 - inference rules of first-order logic
 - axioms of arithmetic (e.g., Peano arithmetic)