Axiomatic Semantics

CS 6371: Advanced Programming Languages
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 requires us to step outside the derivation system to prove things about derivations
  – 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, some advanced type systems (like the one used by Coq) encode an entire axiomatic semantics into the type system, but let’s classify that as a merging of type theory with axiomatic semantics.
Two Kinds of “Correctness”

• Partial Correctness
  – Notation: \{A\}c\{B\} (called a Hoare Triple)
  – 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\} \text{ while } (x = 10) \text{ do } x := x + 1 \{ \ ? \ \}
Examples

- ${x \leq 10}$ while $(x \leq 10)$ do $x := x + 1$ 
  ${x = 11}$
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 [ ? ]
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]
Examples

• \{x \leq 10\} \text{ while } (x \leq 10) \text{ do } x := x + 1 \ {x=11}
• \[x \leq 10\] \text{ while } (x \leq 10) \text{ do } x := x + 1 \ [x=11]
• \[T\] \text{ while } (x \leq 10) \text{ do } x := x + 1 \ [\ ? \ ]
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]
• [T] while (x \leq 10) do x := x + 1 [x \geq 11]
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 \[x = 11\]
• \[T\] while (x \leq 10) do x := x + 1 \[x \geq 11\]
• \[x = \bar{i}\] while (x \leq 10) do x := x + 1 \[\ ? \ ]
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]
• \[T\] while \ (x \leq 10) \ do \ x := x + 1 \ [x \geq 11]
• \[x = i\] while \ (x \leq 10) \ do \ x := x + 1 \ [x = \text{max}(11, i)]
Examples

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

• \(\{x \leq 10\} \text{ while } (x \leq 10) \text{ do } x:=x+1 \{x=11\}\)
• \([x \leq 10]\) while \((x \leq 10)\) do \(x:=x+1\) \([x=11]\)
• \([T]\) while \((x \leq 10)\) do \(x:=x+1\) \([x \geq 11]\)
• \([x=i]\) while \((x \leq 10)\) do \(x:=x+1\) \([x=\max(11, i)]\)
• \(\{\text{any } A\}\) any non-terminating program \(\{\text{any } B\}\)
Examples

• $\{x \leq 10\} \text{ while } (x \leq 10) \text{ do } x := x + 1 \ {x=11}$
• $[x \leq 10] \text{ while } (x \leq 10) \text{ do } x := x + 1 \ [x=11]$
• $[T] \text{ while } (x \leq 10) \text{ do } x := x + 1 \ [x \geq 11]$
• $[x=i] \text{ while } (x \leq 10) \text{ do } x := x + 1 \ [x = \max(11, i)]$
• $\{\text{any A}\} \text{ any non-terminating program } \{\text{any B}\}$
• $\{F\} \text{ any program } \{\text{any B}\}$
Language of Assertions

• First-order logic with arithmetic:

  arithmetic exps  \( a ::= i \mid v \mid \bar{v} \mid a_1+a_2 \mid a_1-a_2 \mid a_1*a_2 \)

  assertions       \( A ::= T \mid F \mid a_1=a_2 \mid a_1\leq a_2 \mid A_1\land A_2 \mid A_1\lor A_2 \mid \neg A \mid A_1\Rightarrow A_2 \mid \forall \bar{v}.A \mid \exists \bar{v}.A \)

• 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 SIMPL consists of...
  – six axioms (rules) describing SIMPL programs
  – inference rules of first-order logic
  – axioms of arithmetic (e.g., Peano arithmetic)