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 that incorporates the advances of 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

- $\{x \leq 10\}$ while ($x<=10$) do $x:=x+1$ $\{\ ?\ \}$

Examples

- $\{x \leq 10\}$ while $(x<=10)$ do $x:=x+1$ $\{x=11\}$

Examples

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

Examples

- {x≤10} while (x<=10) do x:=x+1 {x=11}
- [x≤10] while (x<=10) do x:=x+1 [x=11]

Examples

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

Examples

- $\{x \leq 10\}$ while (x<=10) do x:=x+1 $\{x=11\}$
- $[x \leq 10]$ while (x<=10) do x:=x+1 $[x=11]$
- $[T]$ while (x<=10) do x:=x+1 $[x \geq 11]$

Examples

- $\{x \le 10\}$ while (x<=10) do x:=x+1 $\{x=11\}$
- $[x \le 10]$ while (x<=10) do x:=x+1 $[x=11]$
- $[T]$ while (x<=10) do x:=x+1 $[x \ge 11]$
- $[x=\bar{i}]$ while (x<=10) do x:=x+1 [?]

Examples

- $\{x \leq 10\}$ while (x<=10) do x:=x+1 $\{x=11\}$
- $[x \leq 10]$ while (x<=10) do x:=x+1 $[x=11]$
- $[T]$ while (x<=10) do x:=x+1 $[x \geq 11]$
- $[x=\bar{\imath}]$ while (x<=10) do x:=x+1 $[x=\max(11,\bar{\imath})]$

Examples

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

Examples

- $\{x \le 10\}$ while (x<=10) do x:=x+1 $\{x=11\}$
- $[x \le 10]$ while (x<=10) do x:=x+1 $[x=11]$
- $[T]$ while (x<=10) do x:=x+1 $[x \ge 11]$
- $[x=\bar{\imath}]$ while (x<=10) do x:=x+1 $[x=\max(11,\bar{\imath})]$
- {any A} any non-terminating program {any B}

Examples

- $\{x \leq 10\}$ while (x<=10) do x:=x+1 $\{x=11\}$
- $[x \leq 10]$ while (x<=10) do x:=x+1 $[x=11]$
- $[T]$ while (x<=10) do x:=x+1 $[x \geq 11]$
- $[x=\bar{\imath}]$ while (x<=10) do x:=x+1 $[x=\max(11,\bar{\imath})]$
- {any A} any non-terminating program {any B}
- {F} any program {any B}

Language of Assertions

- First-order logic with arithmetic:

SIMPL arithmetic exps	$a ::= i \mid v \mid a_1{+}a_2 \mid a_1{-}a_2 \mid a_1{*}a_2$
assertion exps	$e ::= a \mid \bar{v}$
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 SIMPL consists of…
  - six axioms (rules) describing SIMPL programs
  - inference rules of first-order logic
  - axioms of arithmetic (e.g., Peano arithmetic)