Lecture 19
Introduction to Axiomatic Semantics
Instructor: Hossein Hojjat

October 18, 2017
Motivation

- Consider the following program:

```plaintext
var x, y, t: Int
...
if (x > y) {
  t = x - y
  while (t > 0) {
    x = x - 1
    y = y + 1
    t = t - 1
  }
}
```

- Claim: for any values of \(x\) and \(y\)
  - Loop will terminate
  - When it does, if \(x > y\), the values of \(x\) and \(y\) will be swapped
- How could we prove this?
Motivation

- Techniques we have seen so far are insufficient

**Operational Semantics**

- Easy to argue that a given input produces a given output
- Easy to argue that all constructs in language preserve some property
  - like when we proved type soundness
- Much harder to prove general properties of the behavior of a program on all inputs

**Type-based Reasoning**

- Allows to design custom checkers to verify specific properties
- Good at reasoning about properties of the data pointed at by particular variables
Axiomatic Semantics

- A system for proving properties about programs

Key idea:

- Define the semantics of a construct by describing its effect on assertions about the program state

Two components:

- A language for stating assertions
  - Can be First Order Logic (FOL) or a specialized logic such as separation logic
  - Many specialized languages developed over the years
    - Z, Larch, JML, Spec#
- Deductive rules for establishing the truth of such assertions
Little History

**Ancient years: Unbridled Optimism**

- Heavily endorsed by the scientists like Hoare, Dijkstra and Floyd
- If you can prove programs correct, bugs will be a thing of the past
  - You won’t even have to test your programs

**Medieval Skepticism**

“Social processes and proofs of theorems and programs” (1979) by DeMillo, Lipton and Perlis

- Proofs in math only work because there is a social process in place to get people to evaluate them
- Program proofs are too boring for social process to form around them
- Programs change too fast and proofs are too brittle

**The Renaissance**

- New generation of automated reasoning tools
- A handful of success stories: better appreciation of costs and benefits?
Hoare Triple

\[
\{ A \} \text{ stmt } \{ B \}
\]

- If the precondition holds before stmt and If the stmt terminates postcondition will hold afterwards
- This is a partial correctness assertion
- We sometimes use the notation \([ A ] \text{ stmt } [ B ]\) to denote a total correctness assertion
  - It means you also have to prove termination
Recap: IMP Language

- We define assertions for the simple Imperative Language (IMP)

\[
e := \ n \mid x \mid e_1 + e_2 \mid e_1 = e_2
\]

\[
c := x := e \mid c_1 ; c_2 \mid \text{if } e \text{ then } c_1 \text{ else } c_2 \mid \text{while } e \text{ do } c \mid \text{skip}
\]

- Big Step Semantics have two kinds of judgments

  expressions result in values          commands change the state

\[
\langle e, \sigma \rangle \Downarrow n \quad \quad \langle c, \sigma \rangle \Downarrow \sigma'
\]

- **State**: a function \( \sigma \) from variable names to values
• The language of assertions

\[ A := \text{true} \mid \text{false} \mid e_1 = e_2 \mid e_1 \geq e_2 \mid A_1 \land A_2 \mid \neg A \mid \forall x. A \]

• Notation \( \sigma \models A \) means that the assertion holds on state \( \sigma \)
• This is defined inductively over the structure of \( A \)
• Example. \( \sigma \models A \land B \) iff \( \sigma \models A \) and \( \sigma \models B \)
Statements

\[ \sigma \models \text{true} \quad \sigma \not\models \text{false} \]

\[
\frac{\langle e_1, \sigma \rangle \Downarrow v \quad \langle e_2, \sigma \rangle \Downarrow v}{\sigma \models e_1 = e_2}
\]

\[
\frac{\langle e_1, \sigma \rangle \Downarrow v_1 \quad \langle e_2, \sigma \rangle \Downarrow v_2 \quad v_1 \neq v_2}{\sigma \not\models e_1 = e_2}
\]

\[
\frac{\langle e_1, \sigma \rangle \Downarrow v_1 \quad \langle e_2, \sigma \rangle \Downarrow v_2 \quad v_1 \leq v_2}{\sigma \models e_1 \leq e_2}
\]

\[
\frac{\langle e_1, \sigma \rangle \Downarrow v_1 \quad \langle e_2, \sigma \rangle \Downarrow v_2 \quad v_1 > v_2}{\sigma \not\models e_1 \leq e_2}
\]

\[
\frac{\sigma \models A \quad \sigma \not\models A}{\sigma \models A \land B}
\]

\[
\frac{\sigma \not\models A}{\sigma \not\models A \land B}
\]

\[
\frac{\forall v. \sigma[x \mapsto v] \models A}{\sigma \models \forall x. A}
\]

\[
\frac{\exists v. \sigma[x \mapsto v] \not\models A}{\sigma \not\models \forall x. A}
\]

\[
\frac{\sigma \not\models A}{\sigma \models \neg A}
\]

\[
\frac{\sigma \models A}{\sigma \not\models \neg A}
\]
Partial Correctness

- Partial Correctness can then be defined in terms of Operational Semantics

\[ \{A\} \ c \ {B} \ \text{iff} \]
\[ \forall \sigma. \forall \sigma'. (\sigma \models A \land \langle c, \sigma \rangle \downarrow \sigma') \Rightarrow \sigma' \models B \]
Defining Axiomatic Semantics

- Establishing the truth of a Hoare triple in terms of the operational semantics is impractical.
- The real power of AS is the ability to establish the validity of a Hoare triple by using deduction rules.
- $\vdash \{A\} c \{B\}$ means we can deduce the triple from a set of basic axioms.
Derivation Rules

- Derivation rules for each language construct

\[
\begin{align*}
\vdash \{A \land b\} \ c_1 \ \{B\} & \quad \vdash \{A \land \neg b\} \ c_2 \ \{B\} \\
\vdash \{A[x \mapsto e]\} \ x := e \ \{A\} & \quad \vdash \{A\} \ \text{if} \ b \ \text{then} \ c_1 \ \text{else} \ c_2 \ \{B\} \\
\vdash \{A \land b\} \ c \ \{A\} & \quad \vdash \{A\} \ c_1 \ \{C\} \quad \vdash \{C\} \ c_2 \ \{B\} \\
\vdash \{A\} \ \text{while} \ b \ \text{do} \ c \ \{A \land \neg b\} & \quad \vdash \{A\} \ c_1 ; \ c_2 \ \{B\}
\end{align*}
\]

Can be combined together with the rule of **consequence**

\[
\begin{align*}
\vdash A' \Rightarrow A \quad \vdash \{A\} \ c \ \{B\} & \quad \vdash B \Rightarrow B' \\
\vdash \{A'\} \ c \ \{B'\}
\end{align*}
\]

We can weaken a Hoare triple by

- Weakening its postcondition \(B \Rightarrow B'\)
- Strengthening its precondition \(A' \Rightarrow A\)
Soundness and Completeness

What does it mean for our deduction rules to be sound?

- You will never be able to prove anything that is not true
- Truth is defined in terms of our original definition of \( \{A\} \implies \{B\} \)

\[
\forall \sigma. \forall \sigma'. (\sigma \models A \land \langle c, \sigma \rangle \downarrow \sigma') \implies \sigma' \models B
\]

- we can prove this, but it’s tricky!

What does it mean for them to be complete?

- If a statement is true, we should be able to prove it via deduction

So are they complete?

- yes and no
- They are complete relative to the logic
- There are no complete and consistent logics for elementary arithmetic (Gödel)
∀σ.∀σ'.(σ ⊨ A ∧ ⟨c, σ⟩ ↓ σ') ⇒ σ' ⊨ B
⇒
⊢ \{A\} c \{B\}

Prove by induction on the structure of the derivation of ⟨c, σ⟩ ↓ σ'

• Look at all the different ways of proving that ⟨c, σ⟩ ↓ σ'
• Make sure that for each of those, we can prove ⊢ \{A\} c \{B\}
Completeness: Base Case

\[ \langle e, \sigma \rangle \downarrow e' \]

\[ \langle x := e, \sigma \rangle \downarrow \sigma[x \mapsto e'] \]

Need to prove: \((\sigma \models A \land \sigma[x \mapsto e'] \models B) \Rightarrow \vdash \{A\} \ x := e \ \{B\}\]

There is only one rule to prove \(\{A\} \ x := e \ \{B\}\)

\[ \vdash \{\alpha[x \mapsto e]\} \ x := e \ \{\alpha\} \]

So we need to show that

\((\sigma \models A \land \sigma[x \mapsto e'] \models B) \Rightarrow (\sigma \models B[x \mapsto e])\)
Completeness: An inductive case

\[
\begin{align*}
\langle c_1, \sigma \rangle & \Downarrow \sigma'' & \langle c_2, \sigma'' \rangle & \Downarrow \sigma' \\
\hline
\langle c_1; c_2, \sigma \rangle & \Downarrow \sigma'
\end{align*}
\]

Need to prove: \((\sigma \models A \land \sigma' \models B) \Rightarrow \vdash \{A\} \ c_1; c_2 \ \{B\}\)

Assuming
\((\sigma \models A \land \sigma'' \models C) \Rightarrow \vdash \{A\} \ c_1 \ \{C\} \ \text{and} \)
\((\sigma'' \models A \land \sigma' \models B) \Rightarrow \vdash \{C\} \ c_2 \ \{B\}\)