Axiomatic Semantics

Computer Science and Artificial Intelligence Laboratory
MIT

October 21, 2015
Motivation

Consider the following program

```java
... 
if(x > y){
    t = x - y;
    while(t > 0){
        x = x - 1;
        y = y + 1;
        t = t - 1;
    }
}
```

I claim that for any values of x and y
- the loop will terminate
- when it does, if x > y, the values of x and y will be swapped

How could I prove this?
Motivation

The tools we have seen so far are insufficient

- **Operational semantics**
  - easy to argue that a given input will produce a given output
  - also easy to argue that all constructs in the language will 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**
  - types allow us to design custom checkers to verify specific properties
  - very good at reasoning about properties of the data pointed at by particular variables.
Axiomatic Semantics

A system for proving properties about programs

Key idea:
- we can 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
A little history

Early years: Unbridled optimism
- Heavily endorsed by the likes of Hoare and Dijkstra
- If you can prove programs correct, bugs will be a thing of the past
  • you won’t even have to test your programs

The middle ages
- 1979 paper by DeMillo, Lipton and Perllis
  • proofs in math only work because there is a social process in place to get people to argue them and internalize 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, benefits and limitations?
The basics

- **Hoare triple**
  - If the precondition holds before stmt and stmt terminates postcondition will hold afterwards

This is a partial correctness assertion
- we sometimes use the notation
  ```
  [A] stmt [B]
  ```
  to denote a total correctness assertion
  - that means you also have to prove termination
What do assertions mean?

We first need to introduce a language

For today we will be using Winskel’s IMP

e:= n | x | e_1 + e_2 | e_1 = e_2

c:= x := e | c_1 ; c_2 | if e then c_1 else c_2

| while e do c | skip

Big Step Semantics have two kinds of judgments

expressions result in values    commands change the state

\langle e, \sigma \rangle \to n        \langle c, \sigma \rangle \to \sigma'
Semantics of IMP

Commands mutate the state

\[\langle e, \sigma \rangle \rightarrow e' \quad \frac{\langle X := e, \sigma \rangle \rightarrow \sigma[X \rightarrow e']}{\langle c_1, \sigma \rangle \rightarrow \sigma''} \quad \frac{\langle c_2, \sigma'' \rangle \rightarrow \sigma'}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma'}\]

\[\langle e_1, \sigma \rangle \rightarrow \text{false} \quad \langle c_f, \sigma \rangle \rightarrow \sigma' \quad \langle e_1, \sigma \rangle \rightarrow \text{true} \quad \langle c_t, \sigma \rangle \rightarrow \sigma'\]

\[\langle \text{if } e_1 \text{ then } c_t \text{ else } c_f, \sigma \rangle \rightarrow \sigma'\]

What about loops?
Semantics of IMP

The definition for loops must be recursive

\[
\begin{array}{c}
\langle e_1, \sigma \rangle \rightarrow \text{false} \\
\hline
\langle \text{while } e_1 \text{ then } c, \sigma \rangle \rightarrow \sigma
\end{array}
\]

\[
\begin{array}{c}
\langle e_1, \sigma \rangle \rightarrow \text{true} \quad \langle c; \text{while } e_1 \text{ then } c, \sigma \rangle \rightarrow \sigma' \\
\hline
\langle \text{while } e_1 \text{ then } c, \sigma \rangle \rightarrow \sigma'
\end{array}
\]

\[
\begin{array}{c}
\langle e_1, \sigma \rangle \rightarrow \text{true} \quad \langle c, \sigma \rangle \rightarrow \sigma'' \quad \langle \text{while } e_1 \text{ then } c, \sigma'' \rangle \rightarrow \sigma' \\
\hline
\langle \text{while } e_1 \text{ then } c, \sigma \rangle \rightarrow \sigma'
\end{array}
\]
What do assertions mean?

The language of assertions
- \( A := \text{true} \mid \text{false} \mid e_1 = e_2 \mid e_1 \geq e_2 \mid A_1 \text{ and } A_2 \mid \not A \mid \forall x . A \)

Notation \( \sigma \models A \) means that the assertion holds on state \( \sigma \)
- This is defined inductively over the structure of \( A \).
- \( \text{Ex. } \sigma \models A \text{ and } B \iff \sigma \models A \text{ and } \sigma \models B \)
What do assertions mean

Complete list

- \( \sigma \models true \quad \sigma \nvdash false \)

- \( \langle e_1, \sigma \rangle \rightarrow v \quad \langle e_2, \sigma \rangle \rightarrow v \quad \sigma \models e_1 = e_2 \)

- \( \langle e_1, \sigma \rangle \rightarrow v_1 \quad \langle e_2, \sigma \rangle \rightarrow v_2 \quad v_1 \leq v_2 \quad \sigma \models e_1 \leq e_2 \)

- \( \langle e_1, \sigma \rangle \rightarrow v_1 \quad \langle e_2, \sigma \rangle \rightarrow v_2 \quad v_1 \neq v_2 \quad \sigma \nvdash e_1 = e_2 \)

- \( \langle e_1, \sigma \rangle \rightarrow v_1 \quad \langle e_2, \sigma \rangle \rightarrow v_2 \quad v_1 > v_2 \quad \sigma \nvdash e_1 \leq e_2 \)

- \( \sigma \vdash A \quad \sigma \vdash B \quad \forall v. \sigma[x \rightarrow v] \vdash A \quad \sigma \vdash A \quad \sigma \vdash B \quad \sigma \nvdash A \quad \sigma \nvdash B \quad \exists v. \sigma[x \rightarrow v] \nvdash A \)

- \( \sigma \vdash A \quad \sigma \vdash B \quad \forall v. \sigma[x \rightarrow v] \vdash A \quad \sigma \vdash A \quad \sigma \vdash B \quad \sigma \nvdash A \quad \sigma \nvdash B \quad \exists v. \sigma[x \rightarrow v] \nvdash A \)

- \( \sigma \nvdash A \quad \sigma \nvdash B \quad \forall v. \sigma[x \rightarrow v] \nvdash A \quad \sigma \nvdash A \quad \sigma \nvdash B \quad \sigma \nvdash A \quad \sigma \nvdash B \quad \exists v. \sigma[x \rightarrow v] \nvdash A \)
Partial correctness

Partial Correctness can then be defined in terms of OS

\{A\} \subseteq \{B\} \iff

\forall \sigma \forall \sigma' (\sigma \models A \land (c, \sigma) \rightarrow \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

**Derivation Rule 1:**
\[ \vdash \{A[x \rightarrow e]\} x := e \{A\} \]

**Derivation Rule 2:**
\[ \vdash \{A \land b\} c_1 \{B\} \quad \vdash \{A \land \neg b\} c_2 \{B\} \]
\[ \vdash \{A\} \text{if } b \text{ then } c_1 \text{ else } c_2 \{B\} \]

**Derivation Rule 3:**
\[ \vdash \{A \land b\} c \{A\} \]
\[ \vdash \{A\} \text{while } b \text{ do } c \{A \land \neg b\} \]

**Derivation Rule 4:**
\[ \vdash \{A\} c_1 \{C\} \quad \vdash \{C\} c_2 \{B\} \]
\[ \vdash \{A\} c_1 ; c_2 \{B\} \]

Can be combined together with the rule of consequence

**Derivation Rule 5:**
\[ \vdash A' \Rightarrow A \vdash \{A\} c \{B\} \vdash B \Rightarrow B' \]
\[ \vdash \{A'\} c \{B'\} \]
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 \( \{A\} \subset \{B\} \)

\[ \forall \sigma \forall \sigma' (\sigma \models A \land \langle c, \sigma \rangle \rightarrow \sigma') \Rightarrow \sigma' \models B \]
- 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
  - but there are no complete and consistent logics for elementary arithmetic (Gödel)
Completeness Argument

\[ \forall \sigma \forall \sigma' (\sigma \models A \land \langle c, \sigma \rangle \rightarrow \sigma') \Rightarrow \sigma' \models B \]

\[ \Rightarrow \]

\[ \vdash \{A\} c \{B\} \]

Prove by induction on the structure of the derivation of \( \langle c, \sigma \rangle \rightarrow \sigma' \)

- Look at all the different ways of proving that \( \langle c, \sigma \rangle \rightarrow \sigma' \)
- Make sure that for each of those, I can prove \( \vdash \{A\} c \{B\} \)
Completeness: Base case

\[
\begin{align*}
\langle e, \sigma \rangle & \rightarrow e' \\
\langle X := e, \sigma \rangle & \rightarrow \sigma[X \rightarrow e']
\end{align*}
\]

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

I only have one rule to prove \(\vdash \{A\}X := e \{B\}\)

\[
\vdash \{A[x \rightarrow e]\}x := e \{A\}
\]

- (well, that plus the rule of consequence).

So I need to show that

- \((\sigma \models A \land \sigma[X \rightarrow e'] \models B) \Rightarrow (A \Rightarrow B[x \rightarrow e])\)
- Equivalently \(\forall \sigma. (\sigma \models A \land \sigma[X \rightarrow e'] \models B) \Rightarrow (\sigma \models B[x \rightarrow e])\)
Completeness: An inductive case

\[ \langle c_1, \sigma \rangle \rightarrow \sigma'' \quad \langle c_2, \sigma'' \rangle \rightarrow \sigma' \]

\[ \langle c_1; c_2, \sigma \rangle \rightarrow \sigma' \]

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

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