Axiomatic Semantics

Computer Science and Artificial Intelligence Laboratory
MIT

October 25, 2011
Motivation

Consider the following program

```c
...
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
  \[ [A] \text{ stmt } [B] \]
  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

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

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

Big Step Semantics have two kinds of judgments

- Expressions result in values
- Commands change the state

\[ \langle e, \sigma \rangle \rightarrow n \quad \langle c, \sigma \rangle \rightarrow \sigma' \]
Semantics of IMP

Commands mutate the state

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

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

\[
\begin{align*}
\langle e_1, \sigma \rangle & \rightarrow \text{false} \\
\langle c_f, \sigma \rangle & \rightarrow \sigma' \\
\langle \text{if } e_1 \text{ then } c_t \text{ else } c_f, \sigma \rangle & \rightarrow \sigma'
\end{align*}
\]

\[
\begin{align*}
\langle e_1, \sigma \rangle & \rightarrow \text{true} \\
\langle c_t, \sigma \rangle & \rightarrow \sigma' \\
\langle \text{if } e_1 \text{ then } c_t \text{ else } c_f, \sigma \rangle & \rightarrow \sigma'
\end{align*}
\]

What about loops?
Semantics of IMP

The definition for loops must be recursive

\[
\langle e_1, \sigma \rangle \rightarrow true \quad \langle c, \sigma \rangle \rightarrow \sigma'' \\
\langle while e_1 then c, \sigma'' \rangle \rightarrow \sigma' \\
\langle while e_1 then c, \sigma \rangle \rightarrow \sigma'
\]
What do assertions mean?

The language of assertions

- $A := \text{true} \mid \text{false} \mid e_1 = e_2 \mid e_1 \geq e_2 \mid A_1 \text{ and } A_2 \mid \neg A \mid \forall x . A$

Notation $\sigma \models A$ means that the assertion holds on state $\sigma$

- This is defined inductively over the structure of $A$.
- Ex. $\sigma \models A \text{ and } B \iff \sigma \models A \text{ and } \sigma \models B$

Partial Correctness can then be defined in terms of OS

$\{A\} \subset \{B\}$ iff

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

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

\[ \vdash \{A \land b\}c_1 \{B\} \quad \vdash \{A \land \neg b\}c_2 \{B\} \]
\[ \vdash \{A\}if \ b \ then \ c_1 \ else \ c_2 \{B\} \]

\[ \vdash \{A \land b\}c \{A\} \]

\[ \vdash \{A\}while \ b \ do \ c \{A \land \neg b\} \]

\[ \vdash \{A\}c_1 \{C\} \quad \vdash \{C\}c_2 \{B\} \]

\[ \vdash \{A\}c_1 ; c_2 \{B\} \]

Can be combined together with the rule of consequence

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

\[ \vdash \{A'\}c \{B'\} \]
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 \(\{A\} \subset \{B\}\)

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