Axiomatic Semantics

Computer Science and Artificial Intelligence Laboratory
MIT
Nadia Polikarpova with slides by Armando Solar-Lezama

October 26, 2015

Example

$$\frac{}{\vdash \{A[x \to e]\} x := e \ \{A\}}$$

$$\frac{\vdash \{A \wedge b\} c_1 \ \{B\} \qquad \vdash \{A \wedge not\ b\} c_2 \ \{B\}}{\vdash \{A\} if\ b\ then\ c_1 else\ c_2 \ \{B\}}$$

$$\frac{\vdash A' \Rightarrow A \vdash \{A\} c \ \{B\} \vdash B \Rightarrow B'}{\vdash \{A'\} c \ \{B'\}}$$

$$\frac{\vdash \{A \wedge b\} c \ \{A\}}{\vdash \{A\} while\ b\ do\ c\ \{A \wedge not\ b\}}$$

$$\frac{\vdash \{A\} c_1 \ \{C\} \qquad \vdash \{C\} c_2 \ \{B\}}{\vdash \{A\} c_1 ; c_2 \ \{B\}}$$

```
{ x=x0 and y=y0 }
if(x > y){
  t = x - y;
  while(t > 0){
    x = x - 1;
    y = y + 1;
    t = t - 1;
  }
}
{ x0 > y0 => y=x0 and x=y0 }
```

Example

```
{ x=x0 and y=y0 }
 if (x > y) {
  { x>y and x=x0 and y=y0 }
  { x=y0+x-y and y=x0-(x-y) and x-y>=0 }
   t = x - y;
  { x=y0+t and y=x0-t and t>=0 }
   while (t > 0) {
     { x=y0+t and y=x0-t and t>=0 and t>0 }
     { x-1=y0+t-1 and y+1=x0-(t-1) and t-1>=0 }
     x = x - 1;
     { x=y0+t-1 and y+1=x0-(t-1) and t-1>=0 }
     y = y + 1;
     { x=y0+t-1 and y=x0-(t-1) and t-1>=0 }
     t = t - 1;
     { x=y0+t and y=x0-t and t>=0 }
   }
  { x=y0+t and y=x0-t and t>=0 and !(t>0) }
  { y=x0 and x=y0 }
}
{ x0>y0 => y=x0 and x=y0 }
```

$$\frac{}{\vdash \{A[x \to e]\}x := e\,\{A\}}$$

$$\frac{\vdash \{A\}c_1\,\{C\} \qquad \vdash \{C\}c_2\,\{B\}}{\vdash \{A\}c_1; c_2\,\{B\}}$$

$$\frac{\vdash \{A \wedge b\}c_1\,\{B\} \qquad \vdash \{A \wedge not\ b\}c_2\,\{B\}}{\vdash \{A\}if\ b\ then\ c_1 else\ c_2\,\{B\}}$$

$$\frac{\vdash \{A \wedge b\}c\,\{A\}}{\vdash \{A\}while\ b\ do\ c\,\{A \wedge not\ b\}}$$

$$\frac{\vdash A' \Rightarrow A \vdash \{A\}c\,\{B\} \vdash B \Rightarrow B'}{\vdash \{A'\}c\,\{B'\}}$$

From partial to total correctness

Total correctness judgment

- $\vdash [A]\ c\ [B]$
- Just like before, but must also prove termination

$$\frac{\vdash [A \wedge b] c_1\ [B] \qquad \vdash [A \wedge not\ b] c_2\ [B]}{\vdash [A] if\ b\ then\ c_1 else\ c_2\ [B]}$$

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

$$\frac{\vdash [A] c_1\ [C] \qquad \vdash [C] c_2\ [B]}{\vdash [A] c_1; c_2\ [B]}$$

What about loops

Rank function

Function F of the state that

- a) Maps state to an integer
- b) Decreases with every iteration of the loop
- c) Is guaranteed to stay greater than zero
- Also called variant function

$$\frac{\vdash [A \wedge b \wedge F = z]c\ [A \wedge F < z] \qquad \vdash A \wedge b \Rightarrow F \geq 0}{\vdash [A]while\ b\ do\ c\ [A \wedge not\ b]}$$

Example

Can we prove this?

```
[ x=x0 and y=y0 ]
if(x > y){
  t = x - y;
  while(t > 0){
    x = x - 1;
    y = y + 1;
    t = t - 1;
  }
}
[ x0 > y0 => y=x0 and x=y0 ]
```

Example

$$\frac{\vdash [A \wedge b \wedge F = z]c\ [A \wedge F < z] \quad \vdash A \wedge b \Rightarrow F \geq 0}{\vdash [A] while\ b\ do\ c\ [A \wedge not\ b]}$$

```
{ x=x0 and y=y0 }
if (x > y) {
  { x>y and x=x0 and y=y0 }
  { x=y0+x-y and y=x0-(x-y) and x-y>=0 }
  t = x – y;
  { x=y0+t and y=x0-t and t>=0 }
  while (t > 0) {
    { x=y0+t and y=x0-t and t>=0 and t>0 and t=z }
    { x-1=y0+t-1 and y+1=x0-(t-1) and t-1>=0 and t-1<z }
    x = x – 1;
    { x=y0+t-1 and y+1=x0-(t-1) and t-1>=0 and t-1<z }
    y = y + 1;
    { x=y0+t-1 and y=x0-(t-1) and t-1>=0 and t-1<z }
    t = t – 1;
    { x=y0+t and y=x0-t and t>=0 and t<z }
  }
  [ x=y0+t and y=x0-t and t>=0 and !(t>0) ]
  [ y=x0 and x=y0 ]
}
[ x0>y0 => y=x0 and x=y0 ]
```

Weakest Preconditions

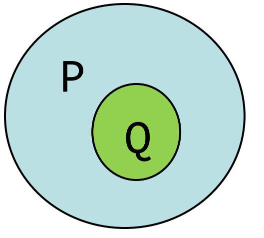
$$P = wpc(c, A)$$

Command

Predicate

Weakest predicate P such that $\vDash \{P\}\ c\ \{A\}$

- P weaker than Q iff $Q \Rightarrow P$



$$\text{wpc}(\text{skip } \{Q\}) = Q$$

$$\text{wpc}(x = e\{Q\}) = Q[e/x]$$

$$\text{wpc}(C1; C2\{Q\}) = \text{wpc}(C1\{\text{wpc}(C2\{Q\})\})$$

$$\text{wpc}(\text{if } B \text{ then } C1 \text{ else } C2\{Q\}) =$$
$$(B \text{ and wpc}(C1\{Q\})) \text{ or } (\text{not } B \text{ and wpc}(C2\{Q\}))$$

Weakest Precondition

While-loop is tricky

- Let W $= wpc(while\ e\ do\ c, B)$
- then,

$$\mathrm{W} = e \Rightarrow wpc(c, W) \ \wedge \ \neg e \Rightarrow B$$

Verification Condition

Stronger than the weakest precondition

Can be computed by using an invariant

$$VC(while_I\ e\ do\ c, B) =$$
$$I \land \ \forall x_1, \ldots x_n\ I \Rightarrow (e \Rightarrow VC(c, I) \land \neg e \Rightarrow B)$$

- Where x_i are variables modified in c.

Example

Is this program correct?

```
i = 5;
while (i > 0)
    invariant { i >= 0 }
{
    i = i - 1;
}
{ i == 0 }
```

$$VC(while_I\ e\ do\ c, B) =$$
$$I \land \ \forall x_1, \ldots x_n\ I \Rightarrow (e \Rightarrow VC(c, I) \land \neg e \Rightarrow B)$$

$vc(i \coloneqq 5; while(i > 0) i \coloneqq i\ - 1, i = 0)$

$vc(i \coloneqq 5, vc(while(i > 0) i \coloneqq i\ - 1, i = 0))$

$vc\big(i \coloneqq 5, i \geq 0 \land \forall i. i \geq 0 \Rightarrow (i > 0 \Rightarrow i - 1 \geq 0) \land (\neg(i > 0) \Rightarrow i = 0)\big)$

$5 \geq 0 \land \forall i. i \geq 0 \Rightarrow (i > 0 \Rightarrow i - 1 \geq 0) \land (\neg(i > 0) \Rightarrow i = 0)$

Assert and Assume

It is convenient to extend the language with statements that prescribe which executions are correct / feasible:

`assert` e: e must hold in every correct execution

`assume` e: e must hold in every feasible execution

```
{ x=x0 and y=y0 }
z = x;
x = y;
y = z;
{ y=x0 and x=y0 }
```

→

```
assume x == x0;
assume y == y0;
z = x;
x = y;
y = z;
assert x == x0;
assert y == y0;
```

Weakest Precondition

$wpc(assert\ e, Q) =$??

for Q to be true after, e must also be true before, because otherwise we won't get past the assert

$wpc(assume\ e, Q) =$??

if e is not true, we don't care if Q is satisfied

Example

Is this program correct?

```
y = 5;
if (x > 0) {
    assert x + y > 5;
} else {
   assume x == 0;
    y = y + x;
   assert x + y == 5;
}
```

What now? How do we decide if this formula is valid?

$$wpc(y := 5; if\ \ldots, \top)$$

$$wpc(y := 5, wpc(if\ \ldots, \top))$$

$$wpc(y := 5, \big(x > 0 \wedge wpc(assert\ x + y > 5, \top)\big) \vee$$
$$\big(x \leq 0 \wedge wpc(assume\ x = 0; y := y + x; assert\ x + y = 5, \top)\big))$$

$$wpc(y := 5, (x > 0 \wedge x + y > 5) \vee (x \leq 0 \wedge (x = 0 \Rightarrow x + y + x = 5))$$

$$(x > 0 \wedge x + 5 > 5) \vee (x \leq 0 \wedge (x = 0 \Rightarrow x + 5 + x = 5)$$

SMT-LIB

SMT-LIB is a language for specifying input to SMT solvers

Basic instructions:

`(declare-fun x () Int)`	declare an integer constant x
`(assert (> x 0))`	add x > 0 to known facts
`(check-sat)`	check if there exist an assignment that makes all known facts true
`(get-model)`	print this assignment

SMT for verification

We need to decide if $wpc(prog, true)$ is valid

- for all values of program variables on entry

How do we encode this as an SMT problem?

- ask if $\neg wpc(prog, true)$ is satisfiable
- if the answer is UNSAT, the problem is correct
- if the answer is SAT, the model gives the input values that violate correctness

Example

Is this formula valid?

$$(x > 0 \land x + 5 > 5) \lor (x \leq 0 \land (x = 0 \Rightarrow x + x + 5 = 5))$$

```
(declare-fun x () Int)

(assert (not (and (> x 0) (> (+ x 5) 5))))
(assert (not
   (and (<= x 0) (or (not (= x 0)) (= (+ x (+ x 5)) 5)))))

(check-sat)
```

6.820 Fundamentals of Program Analysis
Fall 2015