

CS 6110 – Formal Methods in System Design | Spring 2015
Feb-2

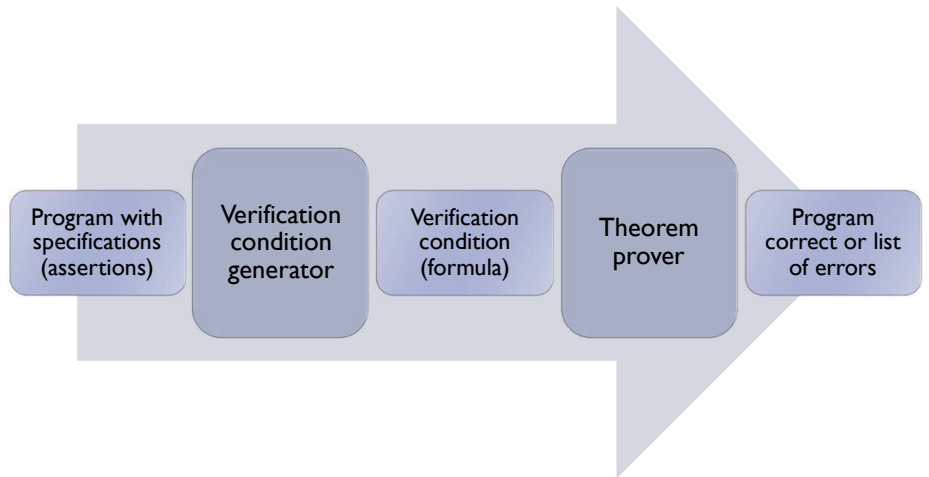
Lecture 6 Verification Conditions I

Zvonimir Rakamarić
University of Utah

This Time

- ▶ Verification condition generation
- ▶ Weakest precondition transformer

Basic Verifier Architecture



Verification Condition Generator

- ▶ Creates verification conditions (mathematical logic formulas) from program's source code
 - ▶ If VC is valid – program is correct
 - ▶ If VC is invalid – possible error in program
- ▶ Based on the theory of Hoare triples
 - ▶ Formalization of software semantics for verification
- ▶ Verification conditions computed automatically using *weakest preconditions* (wp)

Simple Command Language

$x := E$

havoc x

assert P

assume P

$S ; T$ [sequential composition]

$S \square T$ [choice statement]

Program States

▶ Program state s

- ▶ Assignment of values (of proper type) to all program variables
- ▶ Sometimes includes **program counter** variable pc
 - ▶ Holds current program location

▶ Example

$s : (x \mapsto -1, y \mapsto 1)$

$s : (pc \mapsto L, a \mapsto 0, i \mapsto 3)$

- ▶ **Reachable state** is a state that can be reached during some computation

Program States cont.

- ▶ A set of program states can be described using a FOL formula

- ▶ Example

Set of states:

$s : \{ (x \mapsto 1), (x \mapsto 2), (x \mapsto 3) \}$

FOL formulas defining s :

$x = 1 \vee x = 2 \vee x = 3$

$0 < x \wedge x < 4$ [if x is integer]

Hoare Triple

- ▶ Used for reasoning about (program) executions

$$\{ P \} S \{ Q \}$$

- ▶ S is a command
- ▶ P is a **precondition** – formula about program state before S executes
- ▶ Q is a **postcondition** – formula about program state after S executes

Hoare Triple Definition

$$\{ P \} S \{ Q \}$$

- ▶ When a state s satisfies precondition P , every terminating execution of command S starting in s
 - ▶ does not go wrong, and
 - ▶ establishes postcondition Q

Hoare Triple Examples

- ▶ $\{ a = 2 \} b := a + 3; \{ b > 0 \}$
- ▶ $\{ a = 2 \} b := a + 3; \{ b = 5 \}$
- ▶ $\{ a > 3 \} b := a + 3; \{ a > 0 \}$
- ▶ $\{ a = 2 \} b := a * a; \{ b > 0 \}$

Weakest Precondition [Dijkstra]

- ▶ The most general (i.e., weakest) P that satisfies

$$\{ P \} S \{ Q \}$$

is called the **weakest precondition** of S with respect to Q , written:

$$wp(S, Q)$$

- ▶ To check $\{ P \} S \{ Q \}$ prove $P \rightarrow wp(S, Q)$
- ▶ Example

$$\{ ?P? \} b := a + 3; \{ b > 0 \}$$

$$\{ a + 3 > 0 \} b := a + 3; \{ b > 0 \}$$

$$wp(b := a + 3, b > 0) = a + 3 > 0$$

Strongest Postcondition

- ▶ The strongest Q that satisfies

$$\{ P \} S \{ Q \}$$

is called the **strongest postcondition** of S with respect to P , written:

$$sp(S, P)$$

- ▶ To check $\{ P \} S \{ Q \}$ prove $sp(S, P) \rightarrow Q$
- ▶ Strongest postcondition is (almost) a dual of weakest precondition

Weakest Preconditions Cookbook

- ▶ $\text{wp}(x := E, Q) = Q[E / x]$
- ▶ $\text{wp}(\text{havoc } x, Q) = (\forall x. Q)$
- ▶ $\text{wp}(\text{assert } P, Q) = P \wedge Q$
- ▶ $\text{wp}(\text{assume } P, Q) = P \rightarrow Q$
- ▶ $\text{wp}(S ; T, Q) = \text{wp}(S, \text{wp}(T, Q))$
- ▶ $\text{wp}(S \square T, Q) = \text{wp}(S, Q) \wedge \text{wp}(T, Q)$

Checking Correctness with wp

```
{true}
```

```
x := 1;
```

```
y := x + 2;
```

```
assert y = 3;
```

```
{true}
```

Example II

```
{x > 1}
```

```
y := x + 2;
```

```
assert y > 3;
```

```
{true}
```

Example III

```
{true}
```

```
assume x > 1;
```

```
y := x * 2;
```

```
z := x + 2;
```

```
assert y > z;
```

```
{true}
```


Structured if Statement

- ▶ Just a “syntactic sugar”:

`if E then S else T`

gets desugared into

`(assume E ; S) □ (assume ¬E ; T)`

Absolute Value Example

```
if (x >= 0) {  
  abs_x := x;  
} else {  
  abs_x := -x;  
}  
assert abs_x >= 0;
```

Next Time

- ▶ Procedures
- ▶ Loops
- ▶ Loop invariants