Lecture 6

Verification Condition Generator

Zvonimir Rakamarić
University of Utah

This Time

- Homework 2 questions?
- Verification condition generation
- Weakest precondition transformer
Basic Verifier Architecture

- Program with specifications (assertions)
- Verification condition generator
- Verification condition (formula)
- Theorem prover
- Program correct or list of errors

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

- \( x := E \)
- \( \text{havoc } x \)
- \( \text{assert } P \)
- \( \text{assume } P \)
- \( S ; T \) [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 \( s \):
    - \( x = 1 \lor x = 2 \lor x = 3 \)
    - \( 0 < x \land x < 4 \) [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\} $b := a \times a$; \{b > 0\}
Weakest Precondition

- 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**

\[
\begin{align*}
\{ \text{?P?} \} & \ b := a + 3; \ {b > 0} \\
\{ a + 3 > 0 \} & \ b := a + 3; \ {b > 0} \\
wp(b := a + 3, b > 0) & = a + 3 > 0
\end{align*}
\]

Weakest Preconditions Cookbook

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

{true}

\[
x := 1;
\]

\[
y := x + 2;
\]

assert \( y = 3; \)

{true}

Checking Correctness with \( wp \) cont.

{true}

\[
wp(x := 1, x + 2 = 3) = 1 + 2 = 3 \land \text{true}
\]

\[
x := 1;
\]

\[
wp(y := x + 2, y = 3) = x + 2 = 3 \land \text{true}
\]

\[
y := x + 2;
\]

\[
wp(\text{assert } y = 3, \text{true}) = y = 3 \land \text{true}
\]

assert \( y = 3; \)

{true}

Check: true \( \rightarrow \) 1 + 2 = 3 \( \land \) true
Example II
\{x > 1\}

\[y := x + 2;\]

assert \(y > 3;\)
\{true\}

Example II cont.
\{x > 1\}

wp(\(y := x + 2, y > 3\)) = \(x + 2 > 3\)

\[y := x + 2;\]

wp(assert \(y > 3, true\)) = \(y > 3 \land true = y > 3\)

assert \(y > 3;\)
\{true\}

Check: \(x > 1 \rightarrow (x + 2 > 3)\)
Example III

{true}

\[
\text{assume } x > 1; \\
\text{y := x \times 2;} \\
\text{z := x + 2;} \\
\text{assert } y > z; \\
{\text{true}}
\]

Example III cont.

{true}

\[
\text{wp(assume } x > 1, x \times 2 > x + 2) = x > 1 \rightarrow x \times 2 > x + 2 \\
\text{assume } x > 1; \\
\text{wp(y := x \times 2, y > x + 2) = } x \times 2 > x + 2 \\
\text{y := x \times 2;} \\
\text{wp(z := x + 2, y > z) = } y > x + 2 \\
\text{z := x + 2;} \\
\text{wp(assert } y > z, \text{ true) = } y > z \land \text{ true} = y > z \\
\text{assert } y > z; \\
{\text{true}}
\]
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