Computer-Aided Reasoning for Software

CSE507

Reasoning about Programs I

courses.cs.washington.edu/courses/cse507/17wi/

Emina Torlak
emina@cs.washington.edu

Overview

topics

Last lecture

- Finite model finding for first-order logic with quantifiers, relations, and transitive closure

This week

- Reasoning about (partial) correctness of programs
  - Hoare Logic (today)
  - Verification Condition Generation (Friday)

A look ahead (L9–L13)

Classic verification (L9, L10, L11)

- Checking that all (terminating) executions satisfy an FOL property on all inputs

Bounded verification (L12)

- Scope-complete checking of FOL properties

Symbolic execution (L13)

- Systematic checking of FOL properties

A look ahead (L9–L13)

Classic verification (L9, L10, L11)

- Checking that all (terminating) executions satisfy an FOL property on all inputs

Bounded verification (L12)

- Scope-complete checking of FOL properties

Symbolic execution (L13)

- Systematic checking of FOL properties

Active research topic for 45 years

Classic ideas every computer scientist should know

Understanding the ideas can help you become a better programmer

A bit of history

1967: *Assigning Meaning to Programs* (Floyd)

- 1978 Turing Award

1969: *An Axiomatic Basis for Computer Programming* (Hoare)

- 1980 Turing Award

1975: *Guarded Commands, Nondeterminacy and Formal Derivation of Programs* (Dijkstra)

- 1972 Turing Award

A tiny Imperative Programming Language (IMP)

Expression E

- $Z \mid V \mid E_1 + E_2 \mid E_1 * E_2$

Conditional C

- $\text{true} \mid \text{false} \mid E_1 = E_2 \mid E_1 \leq E_2$

Statement S

- **skip** (Skip)
- V := E (Assignment)
- $S_1; S_2$ (Composition)
- **if** C **then** S_1 **else** S_2 (If)
- **while** C **do** S (While)

A minimalist programming language for demonstrating key features of Hoare logic.

Specifying correctness in Hoare logic

{P} S {Q}

Specifying correctness in Hoare logic

{P} S {Q}

Specifying correctness in Hoare logic

{P} S {Q}

Hoare triple

- S is a program statement (in IMP).
- P and Q are FOL formulas over program variables.
- P is called a ***precondition*** and Q is a ***postcondition***.

Specifying correctness in Hoare logic

{P} S {Q}

Hoare triple

- S is a program statement (in IMP).
- P and Q are FOL formulas over program variables.
- P is called a ***precondition*** and Q is a ***postcondition***.

Partial correctness (Hoare triple semantics)

- If S executes from a state satisfying P, and if its execution terminates, then the resulting state satisfies Q.

Specifying correctness in Hoare logic

Hoare triple

{P} S {Q}

- S is a program statement (in IMP).
- P and Q are FOL formulas over program variables.
- P is called a ***precondition*** and Q is a ***postcondition***.

Partial correctness (Hoare triple semantics)

- If S executes from a state satisfying P, and if its execution terminates, then the resulting state satisfies Q.

Total correctness

[P] S [Q]

- If S executes from a state satisfying P, then its execution terminates and the resulting state satisfies Q.

Specifying correctness in Hoare logic

Hoare triple

{P} S {Q}

safety

- S is a program statement (in IMP).
- P and Q are FOL formulas over program variables.
- P is called a ***precondition*** and Q is a ***postcondition***.

Partial correctness (Hoare triple semantics)

- If S executes from a state satisfying P, and if its execution terminates, then the resulting state satisfies Q.

Total correctness

[P] S [Q]

- If S executes from a state satisfying P, then its execution terminates and the resulting state satisfies Q.

Specifying correctness in Hoare logic

Hoare triple

- S is a program statement (in IMP).
- P and Q are FOL formulas over program variables.
- P is called a ***precondition*** and Q is a ***postcondition***.

{P} S {Q}

safety

Partial correctness (Hoare triple semantics)

- If S executes from a state satisfying P, and if its execution terminates, then the resulting state satisfies Q.

liveness

[P] S [Q]

Total correctness

- If S executes from a state satisfying P, then its execution terminates and the resulting state satisfies Q.

Specifying correctness in Hoare logic

Hoare triple

{P} S {Q}

safety

- S is a program statement (in IMP).
- P and Q are FOL formulas over program variables.
- P is called a ***precondition*** and Q is a ***postcondition***.

Partial correctness (Hoare triple semantics)

- If S executes from a state satisfying P, and if its execution terminates, then the resulting state satisfies Q.

liveness

Total correctness

[P] S [Q]

- If S executes from a state satisfying P, then its execution terminates and the resulting state satisfies Q.

Examples of Hoare triples

Examples of Hoare triples

{false} S {Q}

Examples of Hoare triples

{false} S {Q}

- Valid for all S and Q.

Examples of Hoare triples

{false} S {Q}

- Valid for all S and Q.

{P} while (true) do skip {Q}

Examples of Hoare triples

{false} S {Q}

- Valid for all S and Q.

{P} while (true) do skip {Q}

- Valid for all P and Q.

Examples of Hoare triples

{false} S {Q}

- Valid for all S and Q.

{P} while (true) do skip {Q}

- Valid for all P and Q.

{true} S {Q}

Examples of Hoare triples

{false} S {Q}

- Valid for all S and Q.

{P} while (true) do skip {Q}

- Valid for all P and Q.

{true} S {Q}

- If S terminates, the resulting state satisfies Q.

Examples of Hoare triples

{false} S {Q}

- Valid for all S and Q.

{P} while (true) do skip {Q}

- Valid for all P and Q.

{true} S {Q}

- If S terminates, the resulting state satisfies Q.

{P} S {true}

Examples of Hoare triples

{false} S {Q}

- Valid for all S and Q.

{P} while (true) do skip {Q}

- Valid for all P and Q.

{true} S {Q}

- If S terminates, the resulting state satisfies Q.

{P} S {true}

- Valid for all P and S.

Proving partial correctness in Hoare logic

Expression E

- Z | V | E_1 + E_2 | E_1 * E_2

Conditional C

- true | false | E_1 = E_2 | $E_1 \leq E_2$

Statement S

- **skip** (Skip)
- V := E (Assignment)
- S_1; S_2 (Composition)
- **if** C **then** S_1 **else** S_2 (If)
- **while** C **do** S (While)

One inference rule for every statement in the language:

$$\frac{\vdash \{P_1\}S_1\{Q_1\} \ \ldots \ \vdash \{P_n\}S_n\{Q_n\}}{\vdash \{P\}S\{Q\}}$$

If the Hoare triples $\{P_1\}$ $S_1\{Q_1\}$ … $\{P_n\}S_n\{Q_n\}$ are provable, then so is $\{P\}S\{Q\}$.

Inference rules for Hoare logic

$$\frac{}{\vdash \{P\}\ \textbf{skip}\ \{P\}}$$

Inference rules for Hoare logic

$$\frac{}{\vdash \{P\}\ \textbf{skip}\ \{P\}}$$

$$\frac{}{\vdash \{Q[E/x]\}\ x := E\ \{Q\}}$$

Inference rules for Hoare logic

$$\frac{}{\vdash \{P\}\ \textbf{skip}\ \{P\}}$$

$$\frac{}{\vdash \{Q[E/x]\}\ x := E\ \{Q\}}$$

$$\frac{\vdash \{P_1\}\ S\ \{Q_1\} \quad P \Rightarrow P_1 \quad Q_1 \Rightarrow Q}{\vdash \{P\}\ S\ \{Q\}}$$

Inference rules for Hoare logic

$$\frac{}{\vdash \{P\}\ \textbf{skip}\ \{P\}}$$

$$\frac{\vdash \{P\}\ S_1\ \{R\} \qquad \vdash \{R\}\ S_2\ \{Q\}}{\vdash \{P\}\ S_1; S_2\ \{Q\}}$$

$$\frac{}{\vdash \{Q[E/x]\}\ x := E\ \{Q\}}$$

$$\frac{\vdash \{P_1\}\ S\ \{Q_1\} \qquad P \Rightarrow P_1 \quad Q_1 \Rightarrow Q}{\vdash \{P\}\ S\ \{Q\}}$$

Inference rules for Hoare logic

$$\frac{}{\vdash \{P\}\ \textbf{skip}\ \{P\}}$$

$$\frac{\vdash \{P\}\ S_1\ \{R\} \qquad \vdash \{R\}\ S_2\ \{Q\}}{\vdash \{P\}\ S_1; S_2\ \{Q\}}$$

$$\frac{}{\vdash \{Q[E/x]\}\ x := E\ \{Q\}}$$

$$\frac{\vdash \{P \wedge C\}\ S_1\ \{Q\} \quad \vdash \{P \wedge \neg C\}\ S_2\ \{Q\}}{\vdash \{P\}\ \textbf{if}\ C\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \{Q\}}$$

$$\frac{\vdash \{P_1\}\ S\ \{Q_1\} \qquad P \Rightarrow P_1 \quad Q_1 \Rightarrow Q}{\vdash \{P\}\ S\ \{Q\}}$$

Inference rules for Hoare logic

$$\frac{}{\vdash \{P\}\ \textbf{skip}\ \{P\}}$$

$$\frac{\vdash \{P\}\ S_1\ \{R\} \qquad \vdash \{R\}\ S_2\ \{Q\}}{\vdash \{P\}\ S_1; S_2\ \{Q\}}$$

$$\frac{}{\vdash \{Q[E/x]\}\ x := E\ \{Q\}}$$

$$\frac{\vdash \{P \wedge C\}\ S_1\ \{Q\} \quad \vdash \{P \wedge \neg C\}\ S_2\ \{Q\}}{\vdash \{P\}\ \textbf{if}\ C\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \{Q\}}$$

$$\frac{\vdash \{P_1\}\ S\ \{Q_1\} \qquad P \Rightarrow P_1 \quad Q_1 \Rightarrow Q}{\vdash \{P\}\ S\ \{Q\}}$$

$$\frac{\vdash \{P \wedge C\}\ S\ \{P\}}{\vdash \{P\}\ \textbf{while}\ C\ \textbf{do}\ S\ \{P \wedge \neg C\}}$$

loop invariant

Example: proof outline

```
{x ≤ n}
while (x < n) do
  {x ≤ n ∧ x < n}
  {x+1 ≤ n}                // consequence
  x := x + 1
  {x ≤ n}                  // assignment
{x ≤ n ∧ x ≥ n}            // while
{x ≥ n}                    // consequence
```

Example: proof outline with auxiliary variables

$\{x = X \wedge y = Y\}$
$\{y = Y \wedge x = X\}$
t := x
$\{y = Y \wedge t = X\}$ // assignment
x := y
$\{x = Y \wedge t = X\}$ // assignment
y := t
$\{x = Y \wedge y = X\}$ // assignment

Soundness and relative completeness

Proof rules for Hoare logic are sound

If $\vdash$ {P} S {Q} then $\models$ {P} S {Q}

Proof rules for Hoare logic are relatively complete

If $\models$ {P} S {Q} then $\vdash$ {P} S {Q}, assuming an oracle for deciding implications

Summary

Today

- Reasoning about partial correctness of programs
  - Hoare Logic

Next lecture

- Verification condition generation (VCG)
- Weakest preconditions (WP)
- Strongest postconditions (SP)