Computer-Aided Reasoning for Software

# Reasoning about Programs I

courses.cs.washington.edu/courses/cse507/18sp/

#### **Emina Torlak**

emina@cs.washington.edu

#### **Overview**

#### **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 (L09-L13)

#### Classic verification (L09, L10, L11)

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

#### **Bounded verification (LI2)**

Scope-complete checking of FOL properties

## Symbolic execution (LI3)

Systematic checking of FOL properties

## A look ahead (L09-L13)

#### Classic verification (L09, L10, L11)

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

#### **Bounded verification (LI2)**

Scope-complete checking of FOL properties

## Symbolic execution (LI3)

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

•  $Z | V | E_1 + E_2 | E_1 * E_2$ 

#### **Conditional** C

• true | false |  $E_1 = E_2 \mid E_1 \le E_2$ 

#### **Statement** S

- skip (Skip)
- V := E (Assignment)
- S<sub>1</sub>; S<sub>2</sub> (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.



{**P**} **S** {**Q**}

#### **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.

#### **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)

#### **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.

#### **Total correctness**





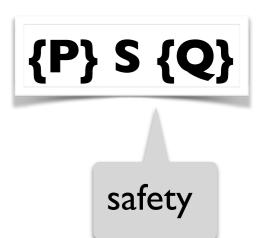
#### **Hoare triple**

- 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.

#### **Total correctness**





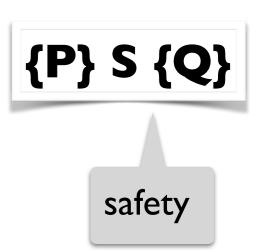
#### **Hoare triple**

- 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.

#### **Total correctness**





#### **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.

#### **Total correctness**



{false} S {Q}

## {false} S {Q}

Valid for all S and Q.

## {false} S {Q}

Valid for all S and Q.

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

## {false} S {Q}

Valid for all S and Q.

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

Valid for all P and Q.

## {false} S {Q}

Valid for all S and Q.

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

Valid for all P and Q.

## **{true} S {Q}**

## **{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.

## **{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}**

## **{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 |  $E_1 = E_2 \mid E_1 \le E_2$ 

#### **Statement** S

- skip (Skip)
- V := E (Assignment)
- S<sub>1</sub>; S<sub>2</sub> (Composition)
- if C then  $S_1$  else  $S_2$  (If)
- while C do S (While)

One inference rule for every statement in the language:

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

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

— {P} skip {P}

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

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

$$\vdash \{P_I\} S \{Q_I\} \quad P \Rightarrow P_I \quad Q_I \Rightarrow Q$$

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

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

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

$$\vdash \{P_I\} S \{Q_I\} \quad P \Rightarrow P_I \quad Q_I \Rightarrow Q$$

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

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

$$\vdash \{P \land C\} S_1 \{Q\} \vdash \{P \land \neg C\} S_2 \{Q\}$$
$$\vdash \{P\} \text{ if } C \text{ then } S_1 \text{ else } S_2 \{Q\}$$

$$\vdash \{P_I\} S \{Q_I\} \quad P \Rightarrow P_I \quad Q_I \Rightarrow Q$$

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

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

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

$$\vdash \{P \land C\} S_1 \{Q\} \vdash \{P \land \neg C\} S_2 \{Q\}$$
$$\vdash \{P\} \text{ if } C \text{ then } S_1 \text{ else } S_2 \{Q\}$$

$$\vdash \{P_I\} S \{Q_I\} \quad P \Rightarrow P_I \quad Q_I \Rightarrow Q$$

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

loop invariant

## **Example: proof outline**

## Example: proof outline with auxiliary variables

## Soundness and relative completeness

#### Proof rules for Hoare logic are sound

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

## 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)