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 (L12)
  • Scope-complete checking of FOL properties

Symbolic execution (L13)
  • 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 (L12)
• Scope-complete checking of FOL properties

Symbolic execution (L13)
• Systematic checking of FOL properties
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** $C$

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

**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

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

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

- 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

- 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**.

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

- 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.

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

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

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

- 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

\{\text{false}\} \ S \ \{Q\}

- Valid for all S and Q.

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

- Valid for all P and Q.
Examples of Hoare triples

\{\text{false}\} \text{ S} \ {\text{Q}}

- Valid for all S and Q.

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

- Valid for all P and Q.

\{\text{true}\} \text{ S} \ {\text{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 \( E \)
- \( Z \mid V \mid E_1 + E_2 \mid E_1 \ast E_2 \)

Conditional \( C \)
- \( \text{true} \mid \text{false} \mid E_1 = E_2 \mid E_1 \leq E_2 \)

Statement \( S \)
- \( \text{skip} \) (Skip)
- \( V := E \) (Assignment)
- \( S_1; S_2 \) (Composition)
- \( \text{if } C \text{ then } S_1 \text{ else } S_2 \) (If)
- \( \text{while } C \text{ do } S \) (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 \( \{P_1\} S_1 \{Q_1\} \ldots \{P_n\} S_n \{Q_n\} \) are provable, then so is \( \{P\} S \{Q\} \).
Inference rules for Hoare logic

\[ \vdash \{P\} \text{skip} \{P\} \]
Inference rules for Hoare logic

\[ \vdash \{P\} \text{skip} \{P\} \]

\[ \vdash \{Q[E/x]\} \times := E\{Q\} \]
Inference rules for Hoare logic

\[\begin{array}{c}
\vdash \{P\} \text{skip} \{P\} \\
\vdash \{Q[E/x]\} x := E \{Q\}
\end{array}\]

\[\begin{array}{c}
\vdash \{P_1\} S \{Q_1\} \quad P \Rightarrow P_1 \quad Q_1 \Rightarrow Q \\
\vdash \{P\} S \{Q\}
\end{array}\]
Inference rules for Hoare logic

\[ \vdash \{P\} \text{skip} \{P\} \]

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

\[ \vdash \{P\} S_1 \{R\} \quad \vdash \{R\} S_2 \{Q\} \]

\[ \vdash \{P\} S_1; S_2 \{Q\} \]

\[ \vdash \{P\} S \{Q\} \]
Inference rules for Hoare logic

\[ \vdash \{P\} \text{skip} \{P\} \]

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

\[ \vdash \{P_1\} S \{Q_1\} \quad P \Rightarrow P_1 \quad Q_1 \Rightarrow Q \]

\[ \vdash \{P\} S \{Q\} \]

\[ \vdash \{P\} S_1 \{R\} \quad \vdash \{R\} S_2 \{Q\} \]

\[ \vdash \{P\} S_1; S_2 \{Q\} \]

\[ \vdash \{P \land C\} S_1 \{Q\} \quad \vdash \{P \land \neg C\} S_2 \{Q\} \]

\[ \vdash \{P\} \text{if } C \text{ then } S_1 \text{ else } S_2 \{Q\} \]
Inference rules for Hoare logic

\[ \vdash \{P\} \text{skip} \{P\} \]

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

\[ \vdash \{P\} S \{Q\} \quad \text{P} \Rightarrow \ P_1 \quad Q_1 \Rightarrow Q \]

\[ \vdash \{P\} S \{Q\} \]

\[ \vdash \{P\} \text{if } C \text{ then } S_1 \text{ else } S_2 \{Q\} \]

\[ \vdash \{P\} \text{while } C \text{ do } S \{P\wedge \neg C\} \]

\[ \vdash \{P\} S_1 \{R\} \quad \vdash \{R\} S_2 \{Q\} \]

\[ \vdash \{P\} S_1 ; S_2 \{Q\} \]

\[ \vdash \{P \land C\} S_1 \{Q\} \quad \vdash \{P \land \neg C\} S_2 \{Q\} \]

loop invariant
Example: proof outline

\{x \leq n\}
while (x < n) do
\{x \leq n \land x < n\}
\{x+1 \leq n\} \quad \text{\textbf{// consequence}}
x := x + 1
\{x \leq n\} \quad \text{\textbf{// assignment}}
\{x \leq n \land x \geq n\} \quad \text{\textbf{// while}}
\{x \geq n\} \quad \text{\textbf{// consequence}}
Example: proof outline with auxiliary variables

\{x = A \land y = B\}
\{y = B \land x = A\}
\texttt{t := x}
\{y = B \land t = A\}
\texttt{x := y}
\{x = B \land t = A\}
\texttt{y := t}
\{x = B \land y = A\}
Soundness and relative completeness

Proof rules for Hoare logic are sound

\[
\text{If } \vdash \{P\} S \{Q\} \text{ then } \models \{P\} S \{Q\}
\]

Proof rules for Hoare logic are relatively complete

\[
\text{If } \models \{P\} S \{Q\} \text{ then } \vdash \{P\} S \{Q\}, \text{ assuming an oracle for deciding implications}
\]
Summary

Today

• Reasoning about partial correctness of programs
  • Hoare Logic

Next lecture

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