Computer-Aided Reasoning for Software

Reasoning about Programs

courses.cs.washington.edu/courses/cse507/16sp/

Emina Torlak
dina@cs.washington.edu
Overview

Last lecture

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

Today

• Reasoning about (partial) correctness of programs
  • Hoare Logic
  • Verification Condition Generation
A look ahead (L9–L14)

Classic verification (L9, L10)
  • Checking that all (terminating) executions satisfy an FOL property on all inputs

Bounded verification (L11)
  • Scope-complete checking of FOL properties

Symbolic execution (L12)
  • Systematic checking of FOL properties

Model checking (L13, L14)
  • Exhaustive checking of temporal properties of abstracted programs

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
• \( Z \mid V \mid E_1 + E_2 \mid E_1 \times E_2 \)

Conditional C
• true | false | \( E_1 = E_2 \mid E_1 \leq E_2 \)

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

**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

{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

\[ Z \mid V \mid E_1 + E_2 \mid E_1 \times E_2 \]

Conditional C

\[ \text{true} \mid \text{false} \mid E_1 = E_2 \mid E_1 \leq E_2 \]

Statement S

\[ \text{skip} \quad \text{(Skip)} \]
\[ V := E \quad \text{(Assignment)} \]
\[ S_1; S_2 \quad \text{(Composition)} \]
\[ \text{if } C \text{ then } S_1 \text{ else } S_2 \quad \text{(If)} \]
\[ \text{while } C \text{ do } S \quad \text{(While)} \]

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

\[
\begin{align*}
\vdash \{P\} \text{skip} \{P\} \\
\vdash \{Q[E/x]\} \times := E \{Q\} \\
\vdash \{P_1\} S \{Q_1\} \quad P \Rightarrow P_1 \quad Q_1 \Rightarrow Q \\
\vdash \{P\} S \{Q\}
\end{align*}
\]

\[
\begin{align*}
\vdash \{P\} S_1 \{R\} & \quad \vdash \{R\} S_2 \{Q\} \\
\vdash \{P\} S_1; S_2 \{Q\} \\
\vdash \{P \land C\} S_1 \{Q\} & \quad \vdash \{P \land \neg C\} S_2 \{Q\} \\
\vdash \{P\} \text{if C then } S_1 \text{ else } S_2 \{Q\} \\
\vdash \{P\} \text{while C do } S \{P \land \neg C\}
\end{align*}
\]

loop invariant
Example: proof outline

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

\{x = X \land y = Y\}
\{y = Y \land x = X\}
\quad t := x
\quad \{y = Y \land t = X\} \quad \text{// assignment}
\quad x := y
\quad \{x = Y \land t = X\} \quad \text{// assignment}
\quad y := t
\quad \{x = Y \land y = X\} \quad \text{// 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
Automating Hoare logic with VC generation

Program annotated with pre/post conditions, loop invariants

Verification Condition Generator (VCG)

Forwards computation:
- Starting from the precondition, generate formulas to prove the postcondition.
- Based on computing strongest postconditions ($sp$).

Backwards computation:
- Starting from the postcondition, generate formulas to prove the precondition.
- Based on computing weakest liberal preconditions ($wp$).
VC generation with WP and SP

wp(S, Q)

- The weakest predicate that guarantees Q will hold after executing S from a state satisfying that predicate.

sp(S, P)

- The strongest predicate that holds after S is executed from a state satisfying P.

\{P\} S \{Q\} is valid iff

- P \rightarrow wp(S, Q) or
- sp(S, P) \Rightarrow Q
Computing \( wp(S, Q) \)

\( wp(S, Q) \):

- \( wp(\text{skip}, Q) = Q \)
- \( wp(x := E, Q) = Q[E / x] \)
- \( wp(S_1; S_2, Q) = wp(S_1, wp(S_2, Q)) \)
- \( wp(\text{if } C \text{ then } S_1 \text{ else } S_2, Q) = C \rightarrow wp(S_1, Q) \land \neg C \rightarrow wp(S_2, Q) \)
- \( wp(\text{while } C \text{ do } S, Q) = X \)

Approximate \( wp(S, Q) \) with \( awp(S, Q) \).
Computing $awp(S, Q)$

$awp(S, Q)$:

- $awp(\text{skip}, Q) = Q$
- $awp(x := E, Q) = Q[E / x]$
- $awp(S_1; S_2, Q) = awp(S_1, awp(S_2, Q))$
- $awp(\text{if } C \text{ then } S_1 \text{ else } S_2, Q) = C \rightarrow awp(S_1, Q) \land \neg C \rightarrow awp(S_2, Q)$
- $awp(\text{while } C \text{ do } \{l\} S, Q) = l$

For each statement $S$, also define $VC(S, Q)$ that encodes additional conditions that must be checked.
Computing $VC(S, Q)$

$VC(S, Q)$:

- $VC(\text{skip}, Q) = \text{true}$
- $VC(x := E, Q) = \text{true}$
- $VC(S_1; S_2, Q) = VC(S_2, Q) \land VC(S_1, awp(S_2, Q))$
- $VC(\text{if } C \text{ then } S_1 \text{ else } S_2, Q) = VC(S_1, Q) \land VC(S_2, Q)$
- $VC(\text{while } C \text{ do } \{I\} S, Q) = (I \land C \Rightarrow awp(S, I)) \land VC(S, I) \land (I \land \neg C \Rightarrow Q)$

$I$ is an invariant.

$I$ is strong enough.
Verifying a Hoare triple

Theorem: \( \{P\} S \{Q\} \) is valid if

\[ VC(S, Q) \land (P \rightarrow \text{awp}(S, Q)) \]

The other direction doesn’t hold because loop invariants may not be strong enough or they may be incorrect.

Might get false alarms.
Summary

Today

• Reasoning about partial correctness of programs
  • Hoare Logic
  • VCG, WP, SP

Next lecture

• Guest lecture by Rustan Leino!
• Verification with Dafny, Boogie, and Z3.