Some background info for program synthesis
Topics

• Hoare logic
  • Reasoning about programs
  • Weakest precondition
  • See also lecture notes for CS152 in Spring 2014
    • https://www.seas.harvard.edu/courses/cs152/2014sp/

• Abstract interpretation
  • Approximating concrete execution
  • See lecture notes for CS252 in Spring 2011
    • https://www.seas.harvard.edu/courses/cs252/2011sp/
  • See also lecture notes for CS152 in Spring 2014
    • https://www.seas.harvard.edu/courses/cs152/2014sp/

• Model checking
  • See lecture notes for CS252 in Spring 2011
    • https://www.seas.harvard.edu/courses/cs252/2011sp/
Axiomatic Semantics

• Key idea: give specifications for what programs are supposed to do
  • Define meaning of programs in terms of logical formulas satisfied by program
  • Enables reasoning about programs
• Pre- and post-condition:
  \{ Pre \} c \{ Post \}
  • Partial correctness: “If Pre holds before execution of c, and c terminates, then Post holds after c.”
  • (Total correctness: “If Pre holds before execution of c then c terminates and Post holds after c.”)
Example

• Example

\{ foo = 0 \land bar = i \} \text{ baz := 0; while foo } \neq \text{ bar do (baz := baz - 2; foo := foo + 1) \{ baz = -2i \}}

• Non example

• \{ \text{ true } \} \text{ if foo } < \text{ 0 then foo := -foo else skip \{ foo } > \text{ 0 } \}
Hoare Logic Rules

- **Hoare logic is sound and relatively complete**
- No more incomplete that our language of assertions ⊨ P⇒Q
Hoare Logic Rules

**Hoare Logic and Program Correctness**

- **Soundness** requires induction on the derivations in Hoare logic.

Hoare logic doesn't allow us to derive partial correctness assertions that actually don't hold. The proof of soundness and completeness.

- **Partial Correctness**
  - A pre-condition specifies what the program expects before execution.
  - Post-conditions:
    - Hold before the first iteration.
    - Hold before and after each iteration as shown in the premise of the rule.
    - Therefore, it is both a pre-condition for the loop.

**Axiomatic Semantics**

- The idea in axiomatic semantics is to give specifications for what programs are supposed to compute.
- It turns out that there is an elegant way of deriving valid partial correctness statements.
- Without having to reason about stores, interpretations, and the execution of programs.

**Conclusion**

The question is how do these sets relate to each other? More precisely, we have to answer two questions.

- Partial correctness statements ensure that the program yields the intended result.
- Total correctness statements ensure that the program terminates and yields the intended result.

Axiomatic semantics help users to understand what the program is supposed to yield without needing to understand how it is supposed to work.

In general, a pre-condition specifies what the program expects before execution; and the post-conditions:

- Pre-condition: holds before the first iteration.
- Post-conditions: hold before and after each iteration.

In other words, if we start with a store $S$, a post-condition $Q$, and a program $P$, then $I = \{P\}; S \models Q$.

**Partial Correctness**

- Partial correctness doesn't ensure that the program terminates.

**Total Correctness**

- Total correctness statements ensure that the program terminates and yields the intended result.

**Examples**

- $\{foo = 0 \land bar = i\} \ {baz := 0}; \ {while} \ foo \neq bar \ {do} \ (baz := baz - 2; \ foo := foo + 1) \ {baz = -2i}$
Example

• Build a proof tree for the following:

\{ \text{foo} = 0 \land \text{bar} = i \} \text{ baz := 0; while foo} \neq \text{bar do (baz := baz} - 2; \text{foo := foo} + 1) \text{ {baz} = } -2i}\}
Predicate transformation

• We now have a logic to prove partial correctness triples \( \{P\} \ c \ \{Q\} \)

• Interesting question: Given \( Q \) and \( c \), what is the weakest \( P \) such that \( \{P\} \ c \ \{Q\} \)?
  
  • Weakest (liberal) pre-condition
  
  • E.g., Consider \( c \equiv \text{“}a = \text{int}[50]; i =0; \text{while} \ (i < b) \ {\ ... \ }; a[i]=0\text{”} \)
  
  • What is the weakest precondition \( P \) such that \( \{P\} \ c \ \{ i \geq 50 \} \)?
    i.e., how do we trigger an overflow?

• Dual is strongest post-condition: given \( P \) and \( c \), what is the strongest \( Q \) such that \( \{P\} \ c \ \{Q\} \)?
Weakest pre-condition

- \( \text{wp}(c, Q) = P \) where \( P \) is the weakest condition such that \( \{ P \} \ c \ \{ Q \} \)
- \( \text{wp}(\text{skip}, Q) = Q \)
- \( \text{wp}(x := e, Q) = Q\{e/x\} \)
  - e.g., \( \text{wp}(\text{foo} := \text{bar}+1, \text{foo} > 42) = (\text{bar}+1 > 42) \)
- \( \text{wp}(c1;c2, Q) = \text{wp}(c1, \text{wp}(c2, Q)) \)
- \( \text{wp}(\text{if } b \text{ then } c1 \text{ else } c2, Q) = \)
  \[ b \Rightarrow \text{wp}(c1, Q) \land \neg b \Rightarrow \text{wp}(c2, Q) \]
  - e.g.,
  \( \text{wp}(\text{if } x < 0 \text{ then } x := -x \text{ else } \text{skip}, x > 0) = ? \)
Weakest pre-condition

• $wp(\text{ while } b \text{ do } c, Q ) = ???$
• In general undecidable
• Conservative under approximation: unroll loop
  • $wp'(\text{ while } b \text{ do } c, Q ) =$
  • $wp(\text{ if } (b) \text{ then } (c;\text{if}(b) \text{ then } c), Q \land \neg b)$
  • i.e., approximate 0-2 executions of loop
• $\{P\} \text{ while } b \text{ do } c \{Q\}$ is valid if
  • $P \Rightarrow wp'(\text{ while } b \text{ do } c, Q )$
  • The converse if not necessarily true
Weakest pre-condition

• \( \text{wp}( \text{while } b \text{ do } c, Q ) = \) ???

• Conservative under approximation: loop invariant
  • A loop invariant \( I \) is true at top of each loop iteration
  • Loop invariant typically supplied by programmer, or use heuristics to guess

• \( \text{wp}'( \text{while } b \text{ do } c, Q ) = \)
  \[ I \land b \implies \text{wp}(c, I) \]

  \[ \land (\neg b \land Q \lor (I \land (I \land \neg b \implies Q))) \]

  \( I \) is a loop invariant

  loop won’t execute

  Invariant holds

  and \( Q \) holds when loop exits

• Note this is **weakest liberal precondition**: it does not require termination