Formal Verification
Module 8
CSC 2218: Software Construction

Department of Computer Science

April 2023

Outline

- Basic Concepts: Formal Verification and Formal Specification
- Axioms
- Assertions and correctness proofs: pre- and post-conditions, Weakest pre-conditions
- Software inspection and walkthroughs - informal code reviews, formal inspection

Formal Verification

- The act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics.
- Formal Verification is another approach to quality assurance in addition to Technical Reviews and Software Testing
- Create a formal "model" of the system and a state machine or other mathematical abstraction that precisely captures the system's behavior
- "Check" the model by formally proving that it implements the desired behavior
- The verification of these systems is done by providing a formal proof on an abstract mathematical model of the system

Formal Methods

- The use of formal methods for software and hardware design is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical analysis can contribute to the reliability and robustness of a design
- Formal Methods refer to any activities that rely on mathematical representations of software including formal system specification, specification analysis and proof, transformational development, and program verification.
- All of these activities are dependent on a formal specification of the software.
- A formal software specification is a specification expressed in a language whose vocabulary, syntax and semantics are formally defined.
- This need for a formal definition means that the specification languages must be based on mathematical concepts whose properties are well understood.

Formal Specification

- Formal specifications are mathematically based techniques used to describe a system, to analyze its behavior, and to aid in its design by verifying key properties of interest through rigorous and effective reasoning tools
- These specifications are formal in the sense that they have a syntax, their semantics fall within one domain, and they are used to infer useful information

Axiomatic Semantics

- Axiomatic semantics, in which the meaning of a system is expressed in terms of preconditions and postconditions which are true before and after the system performs a task, respectively.
- An axiomatic semantics consists of
  - A language for stating assertions about programs
  - Rules for establishing the truth of assertions
- Some typical kinds of assertions
  - This program terminates
  - If this program terminates, the variables x and y have the same value throughout the execution of the program
  - The array accesses are within the array bounds

Axiomatic Semantics Terms

- Program State: The values of all variables in an instance of a running program (held in memory)
- Assertion: What we expect to be true about a subset of the program state during execution
- Precondition: An assertion that we expect must hold before executing a statement or procedure
- Postcondition: An assertion that we expect must hold after executing a statement or procedure
- { A } s { B }
  Where
  A is the precondition
  s is the statement(s) being run
  B is the postcondition
- Example
  { y <= x } z := x ; z := z +1 { y < z }

Weakest Preconditions

{ A } s { B }

- A is the least restrictive assertion that will guarantee B after executing s where
  A is a precondition
  s is the statement being executed
  B is a postcondition

Weakest Precondition Example

$a = b + 1 \{ a > 1 \}$

- One possible precondition: $\{ b > 10 \}$
- The weakest precondition: $\{ b > 0 \}$
- How? Variable substitution!
- $a = b + 1$ where $a > 1$
  substitute a for $b + 1$
  $b + 1 > 1$
  $b > 0$

Proof of Program Correctness

- Given:
  - A postcondition that must be upheld
  - A sequence of statements being executed
- Then:
  - Work back through the program to the first statement.
  - Substitute any/all variables with their assignment expressions
  - If the precondition on the first statement is the same as the program specification, the program is correct.

Sequence Statements

{ Q } s1; s2; s3 { Q' }

- Q is the weakest precondition for all statements
- Q' is the postcondition
- Introduce new assertions (Q1, Q2) that serve as pre/post conditions for each statement
  { Q } s1 { Q1 }
  { Q1 } s2 { Q2 }
  { Q2 } s3 { Q' }

Weakest Precondition Exercise

Find the weakest precondition for the following

- { ? } x = y*2 + 1 { x > 10 }
- { ? } x = x + y { y > x }
- { ? } y = 2 * y { y < 5 }
- { ? } sum = 2 * x + 1 { sum > 1 }
- { ? } b = a + 2; a = 4b + 2 { a > 10 }
- a = 2 * b + 1; b = a - 3 { b < 0 }
- a = 3 * (2 * b + a); b = 2 * a - 1 { b > 5 }

Weakest Precondition Exercise

Compute the weakest precondition for each of the following selection constructs and their postconditions:

- if (a == b) b = 2 * a + 1 else b = 2 * a; { b > 1 }
- if (x < y) x = x + 1 else x = 3 * x { x < 0 }
- if (x > y) y = 2 * x + 1 else y = 3 * x - 1; { y > 3 }