

FORMAL METHODS

- Formal structural descriptions (syntax)
- Formal specifications (semantics)
- Formal verification (program proving)
 - Proofs over program statements
 - Proofs over execution traces;
concurrency
- Formal derivation of programs
- Model checking

MATHEMATICS IN FORMAL METHODS

- First Order Predicate Calculus
 - Temporal logic
- Ideal (infinite) data structures
 - Sets, bags, sequences, maps, relations

FORMAL SYNTAX

- Disciplined representation of the structural aspects of a problem or of a solution
- Graphical or textual, with support for abstraction
- Enables completeness and consistency checking
- BNF, XML

FORMAL SEMANTICS

- Describes behavior by expressing it using a mathematical notation
- Includes description of system state before and after some function is executed (pre-conditions and post-conditions)
- May be based on logical assertions (predicate calculus), formal models (denotational), or abstract machines (operational)
- Examples include Z, VDM, OCL, CSP, Alloy

OPERATIONAL VS. DECLARATIVE SPECIFICATIONS

- Formal specifications come in two major flavors: operational and declarative
- Operational specifications give an ordered set of steps for a subprogram to produce its effect
 - Writing a program for a abstract machine
- *Declarative* specifications state *what* a subprogram does but not *how* to it does it
 - Simpler to understand because you don't have to worry about order
 - Allows a variety of implementations

PROPERTIES

- Formal specification involves stating *properties* of programs
 - Boolean expressions, also called *assertions*, *constraints*, *invariants*, *pre/post conditions*
- We will be concerned with two major categories
 - *Invariants* are properties of an aggregate (system, module, component, collection of classes, class) that must hold at all times
 - *Pre* and *post conditions* state what must hold true respectively before and after execution of a method

SPECIFICATION OF A METHOD

- A method (function, subprogram) computes by taking input parameters and producing an effect
 - The effect of a method includes the output results and any changes to instance variables, global variables, var parameters and devices
- To specify a method, two descriptions must be given
 - Under what conditions is it meaningful for the method to run (*preconditions*)
 - After the method has run, what can be said about the result computed and any changes to global state (*postconditions*)

SPECIFICATION - 2

- The preconditions plus the postconditions together specify *what* it is that the program computes
 - Expressed in terms of only input and output variables
 - Avoids implementation details
 - Assumes termination

FORMAL PROGRAM VERIFICATION

- Not "proving a program correct" but proof that a program implementation matches its specification
 - Requires invention of "loop invariants"
- Limited to behavioral aspects and termination
- Automated support for proof organization and arithmetic simplifications
- Used to prove security properties of an operating system kernel and the correctness of a compiler

Program Verification - 2

- Complementary to testing and inspections
- Predicate logic assertions about program state
 - Values of variables at various positions in program
- Statement-specific verification conditions
 - Assignment; sequence; conditional; iteration

EXAMPLES OF THE USE OF FORMAL METHODS

- Honeywell secure kernel
- CICS
- TCAS (Traffic Collision Avoidance System)
- Cleanroom
- PCTE (Portable Common Tools Environment)
- Compilers
- Hardware

CICS

- Joint IBM / Oxford U. Computing Laboratory
- Use of notation for the specification of transaction processing software (CICS)
- Reduced cost of development of next release by \$5.5 M
- Reduced problems; reduced severity of problems.

ADVANTAGES OF FORMAL SPECIFICATIONS

- Alternative viewpoint supports requirements validation
 - Automated support for completeness and consistency checks
- Support for automated processing
 - Code generation
 - Interpretation
- Increased rigor and precision provide process discipline

REASONS WHY FORMAL SPECIFICATIONS ARE NOT WIDELY USED

- Difficulty in demonstrating cost benefit
- Lack of training
- Limited scope
- Limited tool support

OUTSTANDING ISSUES

- Computer arithmetic
 - Formal verification usually assumes mathematical laws
 - Algebraic simplifications
- Concurrent programs
 - Must deal with shared variables accessed by separate threads
- Non-functional requirements