Introduction to Formal Specifications

Samira Sadaoui

Contents

What are Formal Specifications ?

- Programs at a more abstract level
  - abstract from certain details of implementation
  - example: the precise representation of types in memory
- A formal specification is:
  - specification language + Formal reasoning
  - formal reasoning is supported by precise mathematics and powerful analysis tools

Model/Specification/Formalism

- Model: an abstract representation
- Specification: a formal expression of a model
- Formalism: a mathematical notation for writing a specification; a specification language

What are Formal Methods ?

- Writing formal specifications
- Proving properties about the specifications
- Constructing programs by applying "correctness-preserving" transformations on the specifications
- Verifying the programs by mathematical arguments

Why study Formal Specifications?

- Increase our understanding of systems
- The only way to guarantee certain properties
- Verifiability of implementation
- Early prototyping with executable specifications
- Guidance for testing
- Higher quality software
- Transformation technology
- Automatic code generation
- More effective production
- Minimized maintenance

Formal Software Engineering

Software: computer programs and their related artifacts

Engineering: the application of scientific principles in the context of practical constraints

Formal: the use of models, techniques, and tools that are grounded in mathematics

Formal Software Engineering

- Specifications are used in all phases

  Reqts. Specification: reveal inconsistency (contractions between reqts), incompleteness (missing reqts) and ambiguity (different interpretations)

  Design: refinement demonstrating that next level of abstraction satisfies higher level

  Verification: proving a realization satisfies its specification

  Validation: specification used to produce test cases

  Documentation: unambiguous communication between the specifier, implementor, customer and user

Types of Formal Specifications

- Model-oriented specifications
- Property-oriented specifications
- Hybrid specifications: LOTOS

Model-oriented specifications

- Defines system behavior by building a model in terms of well-defined mathematical constructs (sets, functions, predicate logic, etc)
- Examples: VDM, Z, B, process algebras (Milner's CCS & Hoare's CSP)

Property-oriented specifications

- Defines system behavior in terms of properties that must be satisfied
- Algebraic specifications: a data type is an algebra and the axioms (rewriting rules) state the properties
- Examples of algebraic specifications: Act One, OBJ, Larch, LPG, PLUSS, etc

Mathematical Foundations

- set theory
- graph theory
- automata theory
- algebra theory

Logical Foundations

- predicate logic with quantifications
- temporal logic
- lambda calculus

Analysis of specifications

- Errors can be detected and corrected earlier

- the greater the investment in the testing of specifications, the lower the total development cost

- Verification: verify internal properties (consistency, completeness, deadlock, starvation)

- Validation: verify external properties (to check if the requirements are satisfied)

- Tools: theorem provers and model checkers

Model Checkers

- Build a finite model of a system (labeled transition systems for LOTOS) and check that a desired property holds in that model
- Fully automatic and fast
- The main disadvantage is the state explosion

Theorem provers

- Process of finding a proof of a property from the axioms of the specification.