State-Based Methods: Refinement & Model-Based Testing

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems

Pavel Parízek

FACULTY OF MATHEMATICS AND PHYSICS
Charles University

Object-Z

- Main features
  - Classes & instances
  - Operations (methods)
  - Inheritance
  - History invariants
  - Dot notation
- Benefits
  - OOP: structure, modularity, reuse

Refinement

- Goal: specification → design → code
- Initial “Step 0”
  - Collecting user stories (requirements)
  - Translating into formal specs (model)
- Operation refinement
- Data refinement

Reminder: Correctness by Construction

- Incremental methodology to create programs
- Start: very abstract specification (model)
  - Hoare triple: {P} S {Q}
- Process: sequence of refinement steps
  - Each step: refining some abstract part to more concrete
    - Following only well-defined sound rules
- Result: correct and valid program (implementation)
- Verification of each step and rules: theorem prover

Operation refinement

- Abstract operation OpA
- Concrete operation OpC

- Weaker precondition
  - `pre OpA => pre OpC`

- Stronger postcondition
  - `post OpC => post OpA`

- Analogy: inheritance & method overriding
  - Object-oriented development

Data refinement

- Goal: design concrete data structures

- Abstract schemas → abstract states
- Concrete schemas → concrete states

- Abstraction schema: abstract ↔ concrete

- Correct data refinement
  - `pre OpA ∧ Abs => pre OpC`
  - `pre OpA ∧ Abs ∧ OpC => Abs' ∧ post OpA`
  - `InitC => InitA ∧ Abs`

Iterative step-wise refinement

- Target: complex systems

- Step
  - Refine some parts of the system model
  - Create procedures ➔ modular design

Example

- Bank account system
- Abstract data structures
  - Mathematical model (clarity)
- Concrete data structures
  - Computer representation (performance)

Tool support

- CorC ecosystem
  - https://github.com/KIT-TVA/CorC
  - https://helmholtz.software/software/corc
- Simple programs
- Basic data types
- Refinement rules for statements

Model-Based Testing (MBT)

- Basic usage scenario
  - Implementation derived from the model
    - Semi-automated iterative refinement
    - Writing code manually from scratch
  - Test cases also derived from the model
- Representation of models
  - Logic formulas (preconditions, postconditions)
  - State charts (finite state machines) with guards
- Test inputs: concrete solutions of logic formulas

Literature

- G. Smith. The Object-Z Specification Language
  - http://doi.org/10.1007/978-1-4615-5265-9