Lecture 4: Formal SE

Kenneth M. Anderson

Foundations of Software Engineering

CSCI 5828 - Spring Semester, 2000

Today's Lecture

- Introduction to Formal Software Engineering
  – Discuss Models
  – Discuss Formal Notations

Formal Software Engineering

- Software

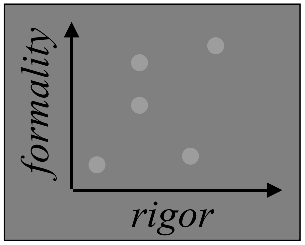
  Computer programs and their related artifacts
- Engineering

  The application of scientific principles in the context of practical constraints
- Formal

  The use of models, techniques, and tools that are grounded in mathematics

Some Important Points

- *Formal* does not mean *Hard*
- *Formal* does not mean *Good*
- *Informal* does not mean *Bad*
  – unless it means *ad hoc*



What Are "Formal Methods"?

❶ *Writing* a formal specification

❷ *Proving* properties about the specification

❸ *Constructing* a program by mathematically manipulating the specification

❹ *Verifying* the program by mathematical argument

Formal SE is Broader

Not just specification and verification of programs…

- Architecture
- Analysis/Testing
- Reliability and Performance Engineering
- Configuration Management
- Process Management, etc.

Model/Specification/Formalism

- Model
  - An abstract representation
- Specification
  - A formal expression of a model or of a property of a model
- Formalism
  - A mathematical notation for writing specifications; a specification language

Specification and the Life Cycle

- Requirements
- Design
  - High level and Low level
- Implementation
- Test

Specification is used in All Activities

Specification/Modeling Styles

- Operational
- Declarative
  - Axiomatic
  - Algebraic
- Structural/Relational

Choice of style dictated by focus of concerns

Specification/Modeling Styles

- Operational (or Imperative)
  - Described according to desired *actions*
  - Usually given in terms of an *execution model*
- Descriptive (or Declarative)
  - Described according to desired *properties*
  - Usually given in terms of *axioms* or *algebras*
- Structural (or Relational)
  - Described according to desired *relationships*
  - Usually given in terms of *multi/hyper graphs*

Logical Foundations

- Predicate/Propositional Logic
- Temporal Logic Systems
- Lambda calculus, etc.

Propositional Logic

- A proposition is a statement that is either true or false, but not both
- Propositional Logic is the language of propositions
  - It consists of well-formed formulas constructed from atomic formulas and logical connectives
  - The meaning of a proposition is determined by the truth values assigned to its assertions

Example

- P = "program does not terminate"
- Q = "alarm rings forever"
- $P \Rightarrow Q$ (If the program does not terminate then alarm rings forever)

P	Q	$P \Rightarrow Q$
T	T	T
T	F	F
F	T/F	T

Library Example

- S: a book is on the stacks
- R: a book is on reserve
- L: a book is on loan
- Q: a book is requested
- Constraints
  - A book can be in only one of three states S, R, and L
  - If a book is on the stacks or on reserve then it can be requested

Library Example, continued

- Constraints specified as propositions

  A: $S \Leftrightarrow \neg(R \vee L)$

  B: $R \Leftrightarrow \neg(S \vee L)$

  C: $L \Leftrightarrow \neg(S \vee R)$

  D: $Q \Rightarrow (S \vee R)$
- Prove "if a book is on loan then it is not requested" is a logical consequence

(One Possible) Solution

- Proof by Contradiction
  - ❶ $L \Rightarrow \neg Q$
  - ❷ $\neg(\neg L \vee \neg Q)$
  - ❸ $L \wedge Q$
  - ❹ L
  - ❺ $\neg(S \vee R)$
  - ❻ Q
  - ❼ $(S \vee R)$
  - ❽ Contradiction
- Steps
  - This is our goal
  - We negate it…
  - …and get this
  - Conjunction Elim, 3
  - Biconditional Elim, 4, C
  - Conjunction Elim, 3
  - Modus Ponens, 6, D
  - 5 and 7 contradict

Another Solution

- Direct Proof
  - $Q \rightarrow (S \vee R)$
  - $\neg Q \vee (S \vee R)$
  - $L \Leftrightarrow \neg(S \vee R)$
  - $\neg L \Leftrightarrow (S \vee R)$
  - $\neg Q \vee \neg L$
  - $\neg L \vee \neg Q$
  - $L \rightarrow \neg Q$
- Steps
  - D
  - Logical Equivalence of 1
  - C
  - Double Negation of 3
  - Substitution, 2, 4
  - $\vee$ is communicative
  - Logical Equivalence of 6

Predicate Logic

- Propositional Logic cannot specify the relationships between objects
  - It can only assert that particular properties hold or do not hold within a set of propositions
- Predicate Logic has the power to do so
  - consists of
    - constants, predicates, variables, and functions

Example use of predicate logic

- Consider lines and points on a plane
  - (1) two lines meet at a unique point
  - (2) there is a unique line through any two points
  - line(x) = x is a line
  - point(x) = x is a point
  - lies_on(x, y) = point x is contained in line y

Example, continued

- domain distinction
  - (a) $\forall$ x • (point(x) $\vee$ line(x));
  - (b) $\forall$ x • ($\neg$(point(x) $\wedge$ line(x)));
- incidence
  - $\forall$ x, y • (lies_on(x, y) $\Rightarrow$ (point(x) $\wedge$ line(y)));

Mathematical Foundations

- Set theory
- Graph theory
- Automata theory
- Abstract algebra
- Probability and statistics

Analysis of Specifications

- Static Analysis

  Examines specification text to reveal properties

- Dynamic Analysis

  Executes specification text to reveal properties

Choice of analysis dictated by focus of concerns and choice of specification style