

# CSci 780 Advanced Software Engineering

## A Specifier's Introduction to Formal Methods

Jeanette M. Wing

10/31/2003

1

## Outline

- Syntax and Semantics
- Formal Methods
- Simple Example
- Types of Specification Languages and Examples
- Difficulties, Myths
- Soapbox
- Summary
- Discussion

10/31/2003

2

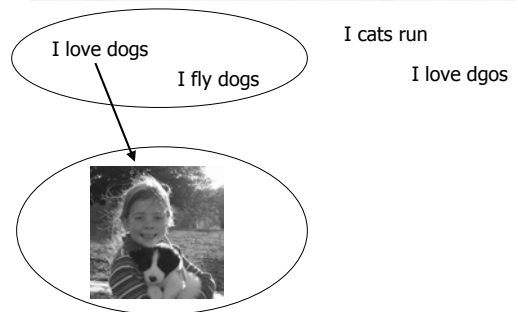
## Language Syntax and Semantics

- Syntax: The elements of the language, along with the rules for combining them
- Semantics: The meaning of each combination
  - Expressed as a mapping from the set of syntactic elements to some well-known set of elements
- Syn: the set of legal "sentences"
- Sem: the set of meanings
- Sat: a mapping of elements from Syn to Sem

10/31/2003

3

## Example



10/31/2003

4

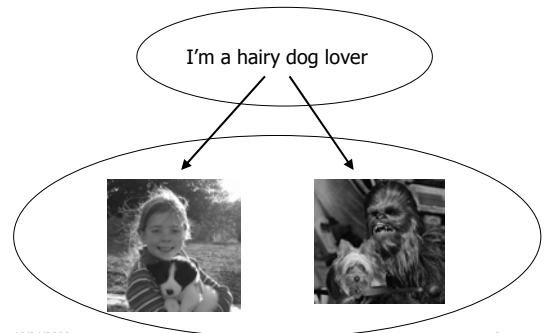
## Formal Methods

- Mathematically-based notations, tools, and techniques for the development of software
  - Wing: "Mathematically-based techniques for describing system properties"
- Use a mathematically precise language to say what we want the system to do
  - NOTE: Code is such a language, but is not declarative (i.e. says *how* more than *what*)

10/31/2003

5

## Why Use Math? Ambiguity, for One



10/31/2003

6

## Helps Assess Completeness

All title and intellectual property rights in and to the SOFTWARE PRODUCT (including but not limited to any images, photographs, animations, video, audio, music, text, and "applets" incorporated into the SOFTWARE PRODUCT), the accompanying printed materials, and any copies of the SOFTWARE PRODUCT are owned by Microsoft or its suppliers. All title and intellectual property rights in and to the content which may be accessed through use of the SOFTWARE PRODUCT is the property of the respective content owner and may be protected by applicable copyright or other intellectual property laws and treaties. This EULA grants you no rights to use such content. All rights not expressly granted are reserved by Microsoft. ["4. Copyright" from Windows 98 EULA]

10/31/2003

7

## Supports Rigorous/Automated Reasoning

The Landing Pilot is the Non-Handling Pilot until the "decision altitude" call, when the Handling Non-Landing Pilot hands the handling to the Non-Handling Landing Pilot, unless the latter calls "go-around," in which case the Handling Non-Landing Pilot continues handling and the Non-Handling Landing Pilot continues non-handling until the next call of "land" or "go-around" as appropriate. In view of recent confusions over these rules, it was deemed necessary to restate them clearly. [Hunt & Thomas]

Is this ambiguous? Consistent? Complete?

10/31/2003

8

## Library Example: Informal Statement

- A book can either be in the stacks, on reserve, or loaned out
- If a book is in the stacks or on reserve, then it can be requested
- We want to (1) formalize the concepts and the statements, (2) prove some theorems to gain confidence that the spec is correct

10/31/2003

9

## Library Example: Formalization

- First lets formalize some concepts
  - S: the book is in the stacks
  - R: the book is on reserve
  - L: the book is on loan
  - Q: the book is requested
- Formalize the constraints
  - $S \Leftrightarrow \neg(R \vee L)$ ,  $R \Leftrightarrow \neg(S \vee L)$ ,  $L \Leftrightarrow \neg(S \vee R)$
  - $Q \Rightarrow (S \vee R)$

10/31/2003

10

## Library Example: Prove of a Theorem

- Prove a theorem to help validate the spec.
  - "A book on loan can't be requested."  $L \Rightarrow \neg Q$
- See if the specification matches our understanding (and vice versa)
- Proof by contradiction
  - Make one assumption: that the statement is false
  - Derive a contradiction
  - Since only one assumption was made, it must be wrong (i.e. not false after all, but rather true)

10/31/2003

11

## Library Example: Proof By Contradiction

- $\neg(L \Rightarrow \neg Q)$  Assume the negation
- $\neg(\neg L \vee \neg Q)$  Rewrite implication
- $L \wedge Q$  Demorgan's Theorem
- L Simplification
- $\neg(S \vee R)$  Biconditional ( $L \Leftrightarrow \neg(S \vee R)$ )
- Q Simplification
- $(S \vee R)$  Modus Ponens ( $Q \Rightarrow (S \vee R)$ )
- Contradiction! So our original assumption must be incorrect, and  $L \Rightarrow \neg Q$  after all

10/31/2003

12

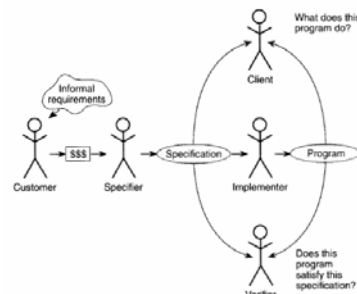
## Formal Methods Activities

- Write a specification using a formal notation
- Validate the specification
  - Inspect it with domain experts
  - Perform automated analysis to prove theorems
- Refine the specification to an implementation
  - Semantics-preserving transformations to code
- Verify that the implementation matches the spec
  - Mathematical argument

10/31/2003

13

## The Formal Specification is Key



10/31/2003

14

## Formal Methods in the Lifecycle

- Wing: "You can apply formal methods in all phases of system development"
- Requirements: clarify user requirements
- Design: modularize, then refine modules
- Verification: e.g. characterize environ.
- Validation: e.g. test case generation
- Documentation: user, maintainer, developer
- Analysis & evolution: e.g. model checking

Specifications are models of the system

10/31/2003

15

## Specification Languages

- Two major types: state-based and behavioral
- State-based (model-oriented)
  - Explicitly specify the valid state, and operations on the state
- Behavioral (property-oriented)
  - Indirectly specify the state via a set of properties (usually a set of axioms)
  - Example: Safety (nothing bad happens), liveness (something good happens)

10/31/2003

16

## Terminology for Model-Based Notations

- *State*: variables which define the important elements of the system which can change
- *Operation*: an important event which can possibly change the state
- *Invariants*: relationships between state variables which are unchanged by operations
- *Given types*: sets of values whose detailed meaning is not specified

10/31/2003

17

## State-Based Example: Z

- The database has a set of people called the "members", and it has a mapping from people to their phones. Each person has up to one phone.
- We want to be able to add a person/phone.

10/31/2003

18

## Types

[Person, Phone]

```
PhoneDB _____  
members : F Person  
phones : F Phone  
map : Person ↔ Phone
```

```
dom map ⊆ members  
ran map = phones
```

10/31/2003

19

## Initialization of the Database

Use "Init" and "prime" conventions

```
InitPhoneDB' _____  
PhoneDB [Schema inclusion, like #include]  
  
members' = {}  
phones' = {}
```

10/31/2003

20

## Changing the Database

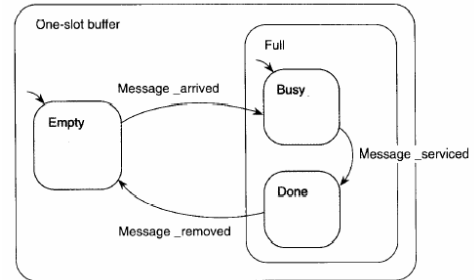
```
AddPhone _____  
ΔPhoneDB [include PhoneDB & PhoneDB']  
person? : Person  
phone? : Phone
```

```
person? ∈ members  
person? ↦ phone? ∉ map  
map' = map ∪ {person? ↦ phone? }  
phones' = phones ∪ { phone? }  
members' = members
```

10/31/2003

21

## Example: Statecharts



10/31/2003

22

## Behavioral Example: LTL

- ◇ Eventually
- □ From now on
- ○ In the next state
- ◇□P: Eventually P will be true forever
- □◇P: Henceforth P will eventually be true

10/31/2003

23

## Difficulties

- Training costs (math is weak in the US)
- Customers can't read the specification
- Incorrect specifications can still be written
- Proving theorems is hard (brain and computer)
- Automated refinement of specification to code is not possible today
- Environmental factors hard to specify
- Some stuff hard to specify: GUIs, concurrency
- No support from business or government

10/31/2003

24

## 7 Myths [Hall 1990]

---

- FM can guarantee that software is perfect
- FM are all about program proving
- FM are only useful for safety-critical systems
- FM require highly trained mathematicians
- FM increase the cost of development
- FM are unacceptable to users
- FM are not used on real, large-scale software

10/31/2003

25

## Soapbox

---

- Engineering is the application of science for the creation of objects for the good of mankind
- Discrete math is the science of CS
- We need to raise our level of science, or we're just practicing art
- Would you contract a bridge which was built without structural analysis?

10/31/2003

26

## Summary

---

- Formal methods improve our ability to build complex, reliable software
- Clarifies complexities in the problem domain
- Can help developers get the domain knowledge they need to develop software with confidence
- Enables precise communication between all parties
- Enables heavy-duty validation with tools
- Eases testing and other verification methods

10/31/2003

27

## Discussion

---

- When would you use FM?
- Really apply FM to all phases of development?
- Can you use FM in XP? Or is it waterfall only?
- What about costs?

10/31/2003

28