LECTURE 6: INTRODUCTION TO FORMAL METHODS

Software Engineering

Mike Wooldridge

1 What are Formal Methods?

- *Formal methods* is that area of computer science that is concerned with the application of mathematical techniques to the design and implementation of computer hardware and (more usually) software.

  > "That part of computer science concerned with the application of mathematical methods to the production of computer software". (Jones, 1986)

- Why bother with formal methods?

  1. The correctnesss problem:
     - producing software that is "correct" is famously difficult;
     - by using rigorous mathematical techniques, it may be possible to make *provably correct* software.
  2. Programs are mathematical objects;
     - they are expressed in a formal language;

– they have a formal semantics;
– programs can be treated as *mathematical theories*.

- Diller (1988) suggests there are two main parts to formal methods:
  1. *Formal specification.*
     Using mathematics to specify the desired properties of a computer system.
  2. *Formal verification.*
     Using mathematics to *prove* that a computer system *satisfies its specification*.
- To which many would add:
  3. *Automated programming.*
     Automating the process of program generation.

ADVANTAGES

- *Formal methods can eliminate ambiguity.*

  A key problem with informal specifications is the inherent ambiguity of textual descriptions; using mathematics can eliminate such ambiguity.

- *Mathematics is concise.*

  Complex properties can be expressed succinctly.

- *Mathematics offers power.*

  There is little that *cannot* in some way be described and reasoned about using maths.

- *Maths facilitates proof.*

  The ability to prove properties of a system is potentially very valuable.

- *Formal specifications, etc., can be manipulated by computer.*
  - CASE tools;
  - automated specification checkers (e.g., CADIZ);
  - automated programming.

- *Formal methods lead to a deep understanding of systems.*

  The precision and detail required brings a deep understanding of what's going on.

OBJECTIONS/MISCONCEPTIONS

- *Formal methods eliminate the need for testing.*

  People can get get sums wrong!

- *Formal methods eliminate the need for natural language.*

  Ultimately, maths is just symbols: English is needed to relate these symbols to reality.

- *You need a PhD to use formal methods.*

  All maths looks hard until you get used to it. . .

APPROACHES TO FORMAL SPECIFICATION

- There are two schools of thought on formal specification:
  1. *Property based*;
  2. *Model based*.

Property Based Specification

- In property based specification, you describe the operations you can perform on a system, and the relationships between operations.
- A property oriented specification consists of:
  - a *signature part* which defines the syntax of operations (what parameters they take and return);
  - an *equations part*, which define the semantics of the operations via a set of equations called *axioms*.

- EXAMPLES OF PROPERTY ORIENTED SPECIFICATION TECHNIQUES.
  - algebraic specification of abstract data types (Hoare, 1969).
  - the OBJ language (Futatsugi *et al.*, 1985).

Model Based Specification

- In model based specification, you use the tools of set theory, function theory and logic to build an *abstact model* of a system.
- You can then specify the operations that may be performed on your model, either explicitly, or implicitly (in terms of pre- and post-conditions).
- The model we construct is:
  - high-level;
  - idealized;
  - free of *implementation bias* (hopefully!)

- A model based specification consists of:
  - a definition of the set of *states* a system may be in;
  - definitions for the legal operations that may be performed on your system, indicating how these change current state.
- EXAMPLES.
  - the Z specification language (Abrial, 1980; Hayes 1987; Spivey 1988);
  - the VDM (Vienna Development Method) specification language (Jones 1980, 1986).

THE Z SPECIFICATION LANGUAGE

- The Z specification language is a semi-graphical notation for writing formal specifications.
- It was developed at Oxford University programming research group in the late 1970s.
- It has been adopted by IBM as their main formal specification tool (so it's not just an academic toy!)
- It was used to specify the IBM Customer Information Control System (CICS) — a major piece of software.

SOFTWARE TOOLS FOR Z

- There are at least three software tools for developing Z specifications:
  - **FUZZ**. Developed at Oxford by Spivey *et al*, in late 1980s.
  - **CADIZ**. Developed at York University, also in late 1980s (its what we have here);
  - **ZED**. Developed at Pennsylvania state University

- Briefly, CADIZ will:
  - take a Z specification in plain-text (ASCII) form;
  - perform some simple semantic checks on it;
  - typeset ('pretty print') it, so that the schemas can be printed out;
  - allow simple interactive browsing of specifications.