Software Model Checking: Theory and Practice

Lecture: *Specification Checking - Foundations*

Objectives

- To understand the goals and basic elements of every formal specification formalism
- To understand that a variety of different levels of abstraction and aspects of program behaviour that can be specified
- To understand the range of languages that can be used to express behavioural property specifications

Outline

- What is a specification?
- Why write specifications?
- What are the building blocks of specs?
- What aspects of behaviour can be captured in a formal specification?
- What are some of the different styles of specification?

What is a Specification?

- A detailed, exact statement of particulars, especially a statement prescribing materials, dimensions, and quality of work for something to be built, installed, or manufactured.

  American Heritage Dictionary 2000

- needs to be precise
- describes what is to be done

What is a Formal Specification?

- A *formal specification* is the expression, in some formal language and at some level of abstraction, of a collection of properties that some system should satisfy

  Axel van Lamsweerde, Future of Software Engineering, 2000

- formal language
  - ensures precision
- properties ... system should satisfy
  - is a prescription

Why Write Specifications?

- To drive the implementation of a system
  - Rare - usually driven from informal requirements
  - Expensive - would require a complete specification
- To provide a redundant description of intent so we can check an implementation against something
  - Generate tests
  - Perform rigorous inspections
  - Model check

A Spec for Spec Languages

Concise

- if the spec is as large and complex as the system, you've shifted the problem to the spec

Understandable

- spec needs to be right, so you must be able to read it

General

- want to be able to describe a wide range of system characteristics

Think Different

- Should force you to express properties differently than solutions

What's a Good Spec?

Consistent

- no internal contradictions

Complete

- captures all of the essential aspects of the problem that are described elsewhere

Unambiguous

- has a unique meaning

Minimal

- doesn't state irrelevant or implementation-specific properties

Essential Parts of a Specification

Components of the system that are related to the property

x

Constraints define what is demanded, desired, or restricted of the components

x>0

Order describes how, if at all, the constrained-components related to one another

if x>0 then after x++, x>0

A Familiar Example

How would you describe a phone?

- Rotary or push button
- Wired or cell
- Coin operated or billed
- Handset or speaker
- Integrated phonebook
- Color, weight, materials, …

We focus on functional behavior

Making a phone call

From a functional point of view:

What components of the phone are relevant?

What characteristics of those components do we care about?

How does the order in which components attain those characteristics influence the making of a phone call?

Variations in Specification Style

- State-based : a condition or mode of being
  - phone is off the hook
  - call is connected
- Event-based : something that happens at a given place and time
  - phone is lifted
  - number 3 is dialed

For You To Do

- Consider the property:

  Dial 532-6350 to connect to CIS
- Give a state-based specification
- Give an event-based specification
- Don’t forget to mention any implicit parts or constraints that are relevant

States and Events

- Changes in state are caused by events

$$x==5 \xrightarrow{x++} x==6$$

- Not all events cause a change in state

$$x==5 \xrightarrow{x=x+0} x==5$$

Mixed States and Events

- When the door is open and the key is not in the ignition, the alarm beeps.

  door==open

  ignitionKey!=in

  beep

- Assigning x to 7 makes x greater than 0.

  x=7

  x>0

Variations in Specification Style

- Allowable behavior : define what a correctly functioning system is able to do

  offhook, number^7, connected

- Violations : define what a correctly functioning system can never do

  onhook, ... anything but offhook ..., connected

Specification Formalisms

- Assertions
  - Describe a condition in a particular system state
- Invariants
  - Describe a condition in all system states
- State Machines
  - Describe sequences of system states
  - Finite state automata vs. Buchi automata
  - Regular expressions vs. Linear Temporal Logic
- ... lots more

Summary

- Specifications are an essential element of rigorous system analysis
- A property specification usually focuses on a specific aspect of a system's behaviour
  - Only some of the system's components are involved
  - Only concerned with a limited view of those components
- Specifications can be written in a variety of styles
  - To suite the goals of the specifier
  - To suite a particular property
  - To enable a particular form of analysis
- There are a large number of specification formalisms that one could apply to state properties of systems