Specification-based Testing of Embedded Systems

Prof. Dr. Holger Schlingloff
Humboldt-Universität zu Berlin
and
Fraunhofer FIRST, Berlin

Lecture 2: Specification Formalisms
Course Outline

- L1: Introduction
  - testing vs. verification, embedded systems, …
- L2: Specification formalisms
  - labeled transition systems, UML / OCL, {CSL | JML}, …
- L3: Test generation
  - Chinese postman algorithm, transition tours, coverage, …
- L4: Test execution
  - testing algorithms in detail, e.g. ParTeG, SpecExplorer
- L5: Test evaluation and assessment
  - oracle problem, mutation analysis

Embedded Systems

- A “system” is “something composed”, here “computational system”
- “Embedded” means “fixed part of a technical system”, i.e., an information-processing unit designed, built and operated as a component in a matter- (or energy-) processing unit
- Characterizing attributes
  - integral component of a technical system, connecting the Platonic world of ideas or forms with the physical world of gravity and matter
  - interaction with environment via sensors and actuators
  - reactive system, non-terminating, “run to completion”
  - fixed purpose (not “general purpose Turing machine”)
Examples

- How many embedded systems are in this room?
- How many embedded systems did you use today?

Properties of Embedded Systems

- Most embedded systems have one or more of the following properties
  - used for steering and control tasks
  - real-time dependent, time critical
  - mass-products, commodity
  - resource bounded (memory, chip-space, energy)
  - not repairable, not maintainable, not extensible
  - high availability, dependability, criticality
  - connected, ubiquitous
Embedded Systems as a Paradigm

- Most software engineering problems can be studied with examples from embedded systems
  - since they are “small”, the examples stay feasible
- Moreover, with embedded systems there are challenges not present in “normal” computing devices
  - mechanical, physical constraints
  - extreme resource dependency
  - real-time, continuous signals processing
  - hardware-software dependencies
  - fixed interfaces, flexible software

Embedded Systems Relevance

- Embedded systems have a market share of 98.2 % of all manufactured processors.
  Only 1.8% are general-purpose processors for laptops, desktop computers and server
Embedded Systems Research Problems

- Necessity of physical building into the environment (space, heat, …)
- Modelling the physical system
- Extreme efficiency requirements
- Fixed system interfaces, no choices
- Knowledge of the environment, interfaces
- Behavior of sensors and actuators, mechanical impreciseness, feedback effects
- Interaction computer scientists / engineers
- Extreme cost pressure
- Correctness (no „patches“ possible)
- Reliability, Availability, Maintainability, Security (RAMS)
- Feature Interaction

Future Development?

- The gap will widen
  - already today we have more embedded systems than humans on this planet
  - computers as through-away-products (RFID-tags, toys, birthday greeting cards, …)
- Ubiquitous computing
  - glasses with earphones
  - jacket with MP3-player
  - GPS in mobile phone camera
- SoC, system-on-chip
- Smart dust, sensor networks
Requirements Engineering

• Assume that you are given an informal description of an embedded system to be built. How to come up with a formal specification?

• Several approaches exist in the literature
  ▪ controlled natural language
  ▪ use-cases approach
  ▪ Parnas’ four-variable model
General Model of Embedded Control

- **Open control**
  - Example: air conditioner without thermostat

- **Closed control loop**
  - Example: air conditioner with thermostat

Four Types of Variables

- **Monitored variables** whose the system supervises and which affect the behavior of the system
  - e.g. the room temperature, setting of desired temperature

- **Controlled variables** whose value can be directly changed/influenced by the system
  - e.g. the power of the compressor

- **External variables** influencing the behavior of the environment but not accessible to the system
  - e.g. open window, sunshine, …

- **Internal variables** which control the behavior of the system but are “invisible” to the outside
  - e.g. the internal state of the controller, power safe mode
  - *not* to be used in specifications!
How to Document Systems Requirements

1. Description of the functionality of the system
2. Identification of relevant environmental entities and actions
   - physical properties: voltage, pressure, temperature,…, thresholds
   - objects
   - events
3. Determining the interfaces between system and environment / user
   - buttons, switches, analog sensor inputs
   - displays, signals, actuator outputs
   - digital i/o connections (to other computational systems)
4. Representing the quantities by variables
   - a state is an assignment of values to variables
5. Defining relations between variables

Example: Air Conditioner

- “Keeps the room temperature at the required level”
- Entities: Room temperature, preset button, temperature sensor, compression motor
- Interfaces: set_preset(temp), motor_on, motor_off, get_current(temp)
- Variables: preset_temp, current_temp, motor_state
- Relation:
  \[
  \text{preset_temp} \leq \text{current_temp} \rightarrow \text{motor_state}=\text{on} \land \\
  \text{preset_temp} > \text{current_temp} \rightarrow \text{motor_state}=\text{off}
  \]
Example: Coffee Machine

Utting/Legeard: “The consumer inserts coins into the machine and then selects the desired drink via several buttons. If enough money has been entered and the selected drink is available, then the machine ejects the drink... If the amount of money inserted was more than the price of the drink, ... The consumer can also press a Return button at any time to retrieve all the money that has been inserted but not yet spent”

- Entities and actions?
- Interfaces?
- Representation?
- Relations?
Repetition: Labelled Transition Systems

Def. LTS = \langle S, L, ?, s_0 \rangle

- $S$ is a set of states
- $L$ is a set of labels
- $?$ $S \times L \times S$ is the transition relation
- $s_0$ $S$ is the initial state

Coffee Machine as LTS
Extended Labelled Transition Systems

Let $D = \langle D_1, \ldots, D_n \rangle$ be a tuple of finite domains (sets). (Without loss of generality, $D_1 = S$)
Let $V = \langle x_1, \ldots, x_n \rangle$ be variables over these domains (again, $x_1 = s$). Then $d = \langle d_1, \ldots, d_n \rangle$? $D$ is called a valuation of $x_1, \ldots, x_n$.
Def. Extended LTS = $\langle D, V, L, ?, d_0 \rangle$
- $D$ and $V$ are as defined above
- $L$ is a set of labels
- $? ? D \times L \times D$ is the transition relation
- $d_0 = \langle d_{1,0}, \ldots, d_{n,0} \rangle$? $D$ is the initial valuation
- Usually, $?$ is defined by a formula using $V$
Example: Extended LTS

\[ \text{State} \rightarrow \text{initially } s_0 \]
\[ \text{number-of-coins} \rightarrow \text{init or} \]
\[ \text{inst.\ coin/} \]
\[ s_0 \rightarrow \text{number-of-coins} \rightarrow 1 \]
\[ \text{ dispens, drink } \]
\[ \text{[number-of-coins \times \ price] / number-of-coins \rightarrow \ price } \]

\text{Domains:}
\[ \text{Coins: } 0, 0, \ldots, 0 \]
\[ \text{STATE: } \{ s_0, s_1, s_2 \} \]