Software Integration, Validation and Verification Final Project

May 1, 2024

Abstract

This project is meant to be solved in groups of 4 students. If the number of students in the course is not evenly divisible by 4, then exactly one group can have fewer than 4 students, or if exactly one student is left alone, then one group with 5 students is allowed. The deadline for submitting this project is May 12th at 23:59. This will give you 12 whole days for completing the project. The project includes writing an Alloy model of the problem domain, writing source code and verify this code with TACO, and generate tests achieving coverage for one of the methods in the main class.

1 Project Description

The project is about a car dealership. A car dealership has vendors and cars in their stock. A dealership may carry a number of car brands. It is necessary to keep track of the sales made: each car and the vendor information must be stored in order to keep track of bonuses for the vendors (you do not need to worry about the bonuses, just make sure the information is maintained). Some rules should also be maintained:

1. A dealership may never have fewer than 5 cars available for sale.
2. A dealership may only sell cars from the brands it carries.
3. If a car is in stock, then it can not appear as having been sold.

You want to allow the dealership to perform two activities: sell cars in their stock, and acquire new cars from the makers to add to their stock.

As a solution to this project, you will have to turn in certain deliverables (to be explained in detail on further sections). In all cases we will provide templates for the deliverables so that integration across the project becomes straightforward.

2 The Deliverables

In this section we will describe each one of the expected outcomes of the project.

2.1 The Alloy Model

File `projectTemplate.als` provides a template for the Alloy model you have to write. It includes all the signatures you have to use, and interfaces for the facts and predicates modeling the problem. For example, you will find:

```
fact allCarsInSoldBrands {
//   This fact must account for the rule "A dealership may
//   only sell cars from the brands it carries".
}
```

Another example:

```
pred sell[d, d' : Dealership, c : Car, v : d.vendors]{
//    This predicate accounts for the "sell" method. Parameter d
//    stands for the initial state and parameter d' stands for
//    the final state.
}
```

You are not allowed to modify the provided signatures or predicate interfaces. **You must deliver an Alloy model project.als.**

2.2 The Java Classes

You need to complete class Dealership.java. Inside the class you will find the interface for the required methods as well as JML specifications for these methods. Your goal is to write code for the constructor and methods `sell` and `buy`, such that the analysis with TACO does not provide any counterexamples. You can use the provided auxiliary classes and only those classes. Notice that class `CarSet` does not provide a method to remove a car from a set. Since that is part of what you are expected to do in method `sell`, the idea is that instead of modifying class `CarSet` (which is not allowed), you will write the appropriate code as part of method `sell` itself. Therefore, you will have to traverse the list implementing `CarSet` within method `sell`.

2.3 How to use TACO in the project

As part of the project you will find Java sources for the auxiliary classes and for the `Dealership.java` class template. These classes must be located inside the `tests` folder in the correct package (look at the classes package information). Also, you are provided with a class `DealershipTest.java`. You have to put this class inside folder `unitest`, in the correct package (again, look at the class package information).

You must deliver the Java class Dealership.java as well as screenshots of the analysis for methods `Dealership`, `buy`, `sell` showing the UNSAT outcome for the analyses.

2.4 Generated Tests

To complete the project you must:

- Build the Control Flow Graph for method `sell`.
- Find a testset that will achieve edge coverage.

You must deliver the Control Flow Graph in a PDF file named `cfg.pdf` and a class `DealershipSellTest.java`, containing methods for each test. Each method must build a receiver object of class Dealership and values for the parameters of method `sell`.

3 If you have doubts

I will be coming to Campus every day usually from 10:30 am to 5:00 pm, so you can find me in my office (CCSB 3.1008). Also, you may contact me through Teams and email.