EXAM
System Validation
(192140122)
08:45 - 11:45
08-11-2021

- This exam consists of 5 exercises.
- The exercises are worth a total of 100 points.
- The final grade for the course is $\frac{(a_1+a_2+a_3)/3+exam/10}{2}$, provided that you obtain at least 50 points for the exam (otherwise, the final grade for the course is at most 4).
- The exam is open book: all *paper* copies of slides, and papers are allowed. Anything else is not allowed. So in particular: copies of old exams are *not* allowed, notes are *not* allowed, and the course handout is *not* allowed.

1 Formal Tools and Techniques (18pts)

(a) In the guest lecture you saw that in some cases the model checker couldn't check the property because it took too much time. What is the cause for this?

(b) For finding bugs in a program, what is the main disadvantage of using a model checker, like NuSMV, compared to using a program verification tool, like Frama-C?

(c) Discuss for the following techniques whether they would be able to check the following property: "Some trees may still have leaves next week.". Assume that there is a model/program that simulates the number of leaves the trees have, depending on the weather conditions.

  (a) NuSMV
  (b) Larva
  (c) Static verification in Frama-C

(d) Suppose you have a program that calculates the average number of fallen leaves per day, and want to check that this average is calculated correctly. You can do this by using CIVL, or static verification in Frama-C. Are there any differences in the type of guarantees that those tools offer? Explain you answer.

2 Specifications (12pts)

Write formal specifications for the following informal requirements in an appropriate specification formalism of your choice. You may assume that appropriate atomic propositions, program functions, and classes exist.

(a) If I provide the right password, then I get logged in on my laptop.

(b) I have to inflate my bycicle tire now and then.

(c) I never miss the train of 7AM.

3 Modelling and Temporal Logic (20pts)

In this exercise we will model a roller coaster in an attraction park. The `main` module looks as follows:

```
MODULE main
VAR
  p : park;
  r : rollercoaster(q.inQueue, 20, p.state);
  q : queue(r.inRollercoaster, p.state);
```

(a) Define the module `rollercoaster(inQueue, nrOfSeats, state)`. It keeps track of a number `inRollercoaster` of the visitors in the rollercoaster, such that:

- At most `nrOfSeats` visitors can be seated in the rollercoaster.
- There are `inQueue` visitors waiting to be seated in the rollercoaster.
- The rollercoaster is filled with visitors as much as possible.
- Each step the rollercoaster performs a ride, so each step the rollercoaster is emptied and new visitors are seated in the rollercoaster.
- Variable `state` models the state of the park: it can be either `Open` or `Closed`. If `Closed`, no visitors can be seated in the rollercoaster.

(b) Specify the following properties for module `rollercoaster`:

i No visitors are in the rollercoaster when the park is closed.

ii If the park is open, then at some point in time, the maximum number of visitors may be seated in the rollercoaster.

(c) State for each of the two properties of the question (b) whether it is a safety or a liveness property.

(d) Explain for each of the two properties of question (b) whether it holds in the model or not. If you need to make assumptions about the complete definition of the model, then write those down.

(e) The module `queue(inRollercoaster, state)` models the number of visitors waiting in the queue of the rollercoaster. Suppose that the module has been implemented, and that it stores the number of waiting visitors in variable `inQueue`. Specify the following properties for the module `queue`:

i The queue regularly has waiting visitors.

ii The queue only has visitors waiting when the park is open.

4 Runtime and Static Verification (25pts)

In this exercise we verify programs about a rollercoaster system.

(a) The engineers have installed a sensor at the entrance of the rollercoaster, that measures the height of each of the rollercoaster passengers. They want to analyze the data that they collected using a function `analyzeRide(int passengerHeights[], int nrPassengers)`. They collected data of two rides and call the function in the following way:

```
int main() {
    int p1[3] = {200,150,180};
    int p2[3] = {140, 120,190};
    analyzeRide(p1,3);
    analyzeRide(p2,3);
}
```

In their analysis the engineers assume that all passengers have a height of 130cm or bigger. Write an assertion that can be used for checking at run-time whether this is the case. Also point out where this assertion needs to be placed.

(b) Next, the engineers want to use run-time verification to keep track of the tallest passenger that entered the rollercoaster, and check that this tallest person has a height of less than 200cm. How would you adapt the program to do this?

(c) What will be the verification results of the previous two questions? Explain your answers.

(d) The organization of the park wants to notify visitors when the rollercoaster is busy and when not. Provide all ACSL specifications to fully verify the function `isBusy` below. Use `behaviors`.

```
bool isBusy(int train1, int train2, int nrSeats) {
 bool busy = false;
 if(train1 == nrSeats || train2 == nrSeats) {
  busy = true;
 } else if(train1 + train2 > nrSeats) {
   busy = true;
 }
 return busy;
}
```

(e) The engineers of the rollercoaster updated the rollercoaster such that it can have more than 2 trains. Each train then has the same number of seats. They register the current number of seats of the trains with the following function:

```
 void updateNrSeats(int *trains, int newNrSeats, int nrTrains) {
/*@ loop invariant 0 <= i <= nrTrains;

    loop assigns trains[0 .. nrTrains-1],i;
@*/
  for(int i = 0; i < nrTrains; i++) {
     trains[i] = newNrSeats;
  }
}
```

Provide the missing loop invariant that states that `newNrSeats` is assigned to each element of the array `trains`.

(f) The engineers installed sensors that test whether a belt of a passenger in the rollercoaster has been attached. They use the following function to check that all belts have been attached:

```
bool checkBelts(bool *belts, int nrSeats) {
  for(int i = 0; i < nrSeats; i++) {
    if(!belts[i]) {
      return false;
    }
  }
  return true;
}
```

Provide all specifications to fully verify the function using static verification.

5 Software analysis (25pts)

In this exercise we analyze programs about a rollercoaster system.

(a) The rollercoaster engineers increased the maximum speed of all the trains of the rollercoaster with a factor of 2. They want to define the following function and check it with CIVL:

`void updateSpeed(int factor, int oldTrainSpeeds[], newTrainSpeeds[], int nrTrains)`

Give a definition for the function `updateSpeed` such that the new speeds are stored in `newTrainSpeeds`. Add a CIVL assertion that the elements of `newTrainSpeeds` have been assigned the expected values.

(b) Draw the symbolic state space for function `updateSpeed`. Assume that `nrTrains == 2`.

(c) The engineers would like to monitor that the actual speed of a train is between 30 and 60 km/h during the whole ride on the rollercoaster track (excluding the part in the station). The current speed of a train on the track is registered via function `register(int speed)`. Write a a Larva automaton for monitoring whether the registered speeds are in between 30 and 60 km/h.

(d) The engineers would also like to monitor that the trains return to the station within 180 seconds. Write a Larva automaton for monitoring this. Assume that there are functions to register the trains' departures from, and arrivals at the station.

(e) The engineers installed some breaks and accelerators on the rollercoaster track to regulate the speeds of the trains. They say the speed of a train now changes according to the following program:

```
int speed;

int main() {
    if(speed > 50) {
        speed = 40;
    }
    if(speed <= 35) {
        speed = 45;
    }
    if(speed >= 40) {
        speed = 30;
    }
}
```

Write an abstract program, such that the engineers can model check it. Use predicate `p = speed >= 40`.