Software Analysis 2018

Exercise 1

This is the first of a series of weekly, mandatory exercises for the course on Software Analysis (NWI-IMC035). Hand in a pdf with your answers by e-mail to `marko@cs.ru.nl`, by Tuesday February 5, 23:59 at the latest.

An important aspect of the course is clearly motivating your answers and showing insight into the problems presented. Further along the course, we will discuss cutting edge research. In that context, most of the time there is not a correct answer, only sound reasoning!

1 Software Analysis in General

At `https://en.wikipedia.org/w/index.php?title=Software_bug&oldid=736728878` you will find a list of 8 categories of the most common software bugs. For each of these categories, answer the following questions:

1. Which of the software analysis techniques discussed in the lecture (listed below) do you think is most appropriate to find these types of bugs and why?
   - Interactive Theorem Proving
   - Model Checking
   - Abstract interpretation
   - Syntactical pattern checkers
   - Resource Analysis
   - Type systems
   - Testing
   - Running the program once
2. Have you ever encountered such a bug yourself or heard about it in the news or in another course? Name a practical example.
3. Do you think finding all bugs of this type is decidable in general, i.e. by an analysis that is both sound and complete in a restricted environment? In case it is undecidable in general: would you prefer an analysis that gives many false positives or one that gives many false negatives?

See Part 2 of the exercise on the reverse side of this page.

2 Code Analysis

Now that you have an idea of the various analysis techniques, it is time to try to analyse real C code. Can you notice the kind of (un)desired behaviours the example below will exhibit? Also can you specify bounds on the execution? Please explain the techniques used to answer the exercise (show you have a clear idea how the techniques work and what kind of result they will produce, you don't have to apply the techniques in detail). Is it feasible to automate this kind of testing?.

```
int *data = new int[1024];

void pushItemOfData(int newValue) {
  unsigned int i = 1024;
  while (i >= 0) {
    data[i] = data[i-- + 1];
  }
  data[0] = newValue;
}

void test(int i) {
  if (i++<0 || i>0) {
    abort();
  }
}

int main(int argc, char **argv) {
  pushItemOfData(42);
  test(pow(10,42) / (42 >> 6))
  return 0;
}
```