VU Qualitätssicherung in der Softwareentwicklung
LV-Nr. 716.066, SS 2021

Practical Task 3

ao. Univ.-Prof. Dr. Bernhard Aichernig
Alexander Grossmann, David Wildauer
qs@ist.tugraz.at

18.05.2021, Graz

Submission

Due to: Tuesday, 15.06.2021 23:59
Place of submission: TeachCenter

For Practical Task 3, you should submit a PDF report to TeachCenter. The report should be called **report_groupXX.pdf** where XX stands for your group number. In TeachCenter you can also find an example report using Markdown to show how a report should be structured, but of course you can use any other software to create the PDF report.

Please make sure the code can be copied/pasted from your report to PathCrawler. If we have problems reading your report, this will result in points being deducted.

You will find *implementations.zip* in the TeachCenter. It contains a file *implementations.c* which contains all functions which need to be tested.

1 Theory Questions

This assignment is all about concolic execution. Please give precise answers to the following questions. You can use the maximum example found in section 2.1 to give examples. Question 6 is a bonus question. Answers should be as long as needed, but as short as possible.

[0]In case of abnormal group participation, ie. some students had to do all the work themselves, they should send an email to study assistants and we will decide whether students can get bonus points, or if other students should have their points deducted.

1. What is an oracle? Why don't we just use the algorithm under test as an oracle?
2. What is the difference between concolic and symbolic execution? Explain it using the example.
3. What are the advantages of concolic execution over random testing?
4. Explain why loops are problematic and how they are dealt with in concolic execution?
5. Is concolic execution white-box or black-box testing and why?
6. Which bugs cannot be caught with concolic execution? *(Bonus)*

2 PathCrawler

We suggest the use of the following tool found at `https://pathcrawler.mf.at/`, instead of the default one found online, as the latter has a bug that makes testing unnecessarily harder. For a demonstration check the video found in TC.

To use PathCrawler, go to "Test your code", upload the archive *implementations.zip* we provided, use e.g. `maximum` as test function, and use `implementations.c` as file under test. Then click "Customize test parameters". Then alter the new sections for domains, preconditions and the oracle according to your needs. Then click "Run test".

You can re-run the tests with new domains, preconditions, and oracle using the "Restart" section.

2.1 Maximum

The following example is a solution to function `maximum()` provided. It gives the maximum value of a given array of length $n \geq 1$.

Implementation

```
int maximum(int array[], int n) {
    int max = array[0];
    for (int i = 1; i < n; i++) {
        if (array[i] > max) {
            max = array[i];
        }
    }
    return max;
}
```

Array domains

- $-200 \leq \texttt{array[INDEX\_0]} \leq 200$

Variable domains

- $1 \leq \texttt{n} \leq 10$
- $1 \leq \texttt{dim(array)} \leq 10$

Unquantified preconditions

- $\texttt{n} == \texttt{dim(arrray)}$

Oracle

```
void oracle_maximum(
        int *Pre_array, int *array,
        int Pre_n, int n,
        int pathcrawler__retres__maximum) {
    if (Pre_n != n) {
        pathcrawler_verdict_failure();
        return;
    }

    int found = 0;

    for (int i = 0; i < n; i++) {
        if (Pre_array[i] != array[i]) {
            pathcrawler_verdict_failure();
            return;
        }
        if (array[i] > pathcrawler__retres__maximum) {
            pathcrawler_verdict_failure();
            return;
        }
        if (array[i] == pathcrawler__retres__maximum) {
            found = 1;
        }
    }

    if (found == 0) {
```

```
        pathcrawler_verdict_failure();
    } else {
        pathcrawler_verdict_success();
    }
}
```

2.2 Task Description

In this task you should use PathCrawler to test the implementations of the functions we provided. There are some correct implementations, and some have bugs. For more examples take a look at PathCrawler website.

- `fibonacci()`
  Computes Fibonacci sequence of length n.
- `lcm()`
  Computes least common multiple of two positive integers using the Euclidean algorithm.
- `intersection()`
  Get a set of elements that are in both input sets.
- `sort()`
  Sorts the given array in-place.
- `contains()`
  Looks for a value in the given array and returns 1 if found, 0 if not. *(Bonus)*

In the report, for each function you should include:

- Array and variable domains
- Quantified and unquantified preconditions where they are needed
- Oracle *(make sure we can copy/paste it, do not include the default comment)*
- Whether the function is correct or not
- A screenshot of the "Summary" tab in PathCrawler (section "Session").
- If not correct, describe the bug and show how you used PathCrawler to find it. Explain it and also **include screenshots of how you found the bug** in PathCrawler.

The first three points can be copied directly from PathCrawler's "Context" section, see tabs "Test parameters" and "Oracle" there.

3 Grading

Following grading scheme is used:

5.0 Points	Theory questions, each 1.0 point
10.0 Points	Practical examples, each 2.5 points
3.5 Points	Bonus points

Each theory question is worth 1 point, and there are 5 (+ 1 bonus) theory questions. Each practical example is worth 2.5 points, and there are 4 (+ 1 bonus) practical examples.

You can achieve at most 3.5 bonus points, which are added to regular points. The whole assignment has 15 points, but with bonus points, you can get 18.5 points in total.

Please remind that unreadable reports and non-copyable code result in a point deduction.