Static DOM Event Dependency Analysis for Testing Web Applications

(Artifact Evaluation)

Chungha Sung, Markus Kusano
Virginia Tech
{sch8906, mukusano}
@vt.edu

Nishant Sinha
IBM Research
nishant.sinha
@in.ibm.com

Chao Wang
Virginia Tech
chaowang@vt.edu

1. INTRODUCTION

The objective of this document is to help reviewers evaluate the artifacts associated with our FSE'16 paper. The artifacts include JSdep (a static analysis tool for JavaScript) and the modified Artemis [2] (a tool for automated testing of web applications). Figure 1 shows the overall structure, where *DOM Analysis* is performed by JSdep and *Modified Artemis* is used for web application testing.

- The stand-alone performance of JSdep has been evaluated by running it on all JavaScript benchmarks; the results are shown in Table 1 of our FSE paper.
- The performance of Artemis, with and without the improvement provided by JSdep, has been evaluated on all web application benchmarks; the results are shown in Tables 2 and 3 of our FSE paper.
- We provide scripts for reviewers to run all the analysis procedures provided by JSdep and Artemis, and automatically generate the experimental result tables.
- We also provide the raw data of all the experiments we have conducted to produce the result tables in our FSE paper.

The license of our artifacts is the MIT License. The URL is `https://github.com/sch8906/JSdep`

2. SCORE CARD

2.1 Insightful

We have made two main contributions. The first contribution is developing JSdep, which is the only static analysis tool at the moment that can compute DOM-event dependencies in addition to the standard control and data dependencies for JavaScript code. The second contribution is improving Artemis, where we demonstrate the effectiveness of DOM-event dependencies to improve the test coverage of Artemis.

Both JSdep and Artemis are useful in a wide range of applications, including software analysis, testing, verification, program understanding, and software maintenance.

2.2 Useful

JSdep is a static analysis tool for JavaScript. It computes not only standard control and data dependencies but also DOM-event dependencies. As such, it can be used anywhere a static analysis tool for JavaScript code is needed.

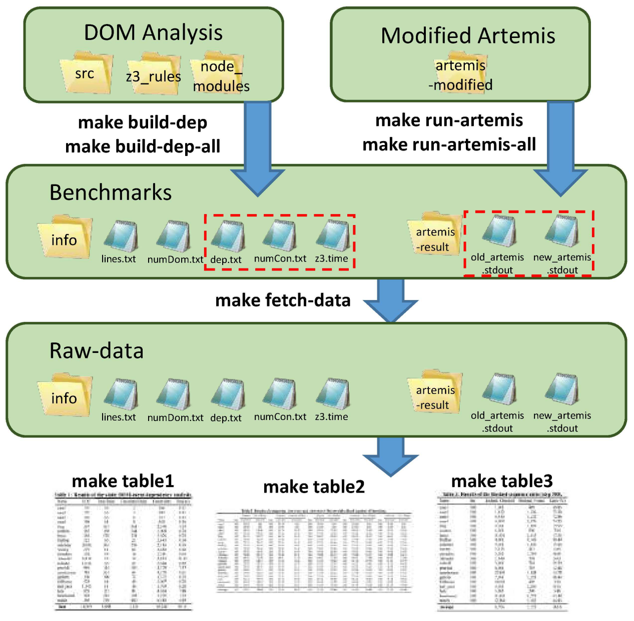

Figure 1: Structure of JSdep with scripts.

Our modified version of Artemis is also a useful platform for conducting research in automated testing of web applications. We have integrated JSdep to the Artemis framework, and developed all the scripts needed automatically to run and compare the test coverage of various Artemis configurations.

2.3 Usable

JSdep and the modified Artemis have been released on GitHub [4] for others to download. Instructions for installing and running these tools are also provided. Moreover, we have provided all the patches and required configurations of existing libraries. Thus, we expect that users can easily download, install, and run them.

3. INSTALLATION

We have tested JSdep under Ubuntu 12.04 Desktop (64-bit). To run JSdep, you need to have *nodejs* and the *Z3* SMT solver [3] installed on your machine.

- To install *nodejs*, run `sudo apt-get install nodejs`.
- To install the *Z3* SMT solver, visit the Z3 website and follow the directions there.

To install our modified version of Artemis, you need to

- install all dependencies that Artemis needs by running `make fetch-apt` under the `artemis-modified` directory, and
- follow the directions provided in the `INSTALL` file under the `artemis-modified` directory.

4. RUNNING THE TOOLS

Assume that you have properly installed both JSdep and the modified Artemis, here is how to run them.

4.1 DOM Dependency Analysis

The benchmark programs are stored in different directories. If you want to conduct dependency analysis of all benchmarks, run the following script:

```
$ make build-dep-all
```

Alternatively, you may analyze each individual benchmark. Assume the benchmark name is `prog`. You may run the following script:

```
$ make build-dep file=prog
```

The results of JSdep will be stored in the following three files under the `info` directory for each benchmark:

- `dep.txt`, which stores the dependency relations;
- `numConstraints.txt`, which stores the number of constraints, and
- `z3.time`, which stores the execution time.

4.2 Web Application Testing

If you want to apply the modified Artemis to all benchmarks with the number of iterations (Assume 500), run the following script:

```
$ make-run-artemis-all iter=500
```

In here, the number of iterations means the number of tests Artemis will run, and in our paper we use 500 iterations. Alternatively, you may run Artemis on each individual benchmark (named `prog`) with the number of iterations (500 in our paper) as follows:

```
$ make-run-artemis file=prog iter=500
```

Artemis will use the `dep.txt` file computed previously for each benchmark to improve the testing performance. The final results will be stored in the following two files under the `artemis-result` directory for each benchmark:

- `old_artemis.stdout`: output of the original Artemis, and
- `new_artemis.stdout`: output of our modified Artemis.

5. OUR RAW EXPERIMENTAL DATA

We provide this directory (`raw-data`) for reviewers who want to check the creation of result tables without going through the process of running all the experiments. Our raw experimental data have been uploaded and the structure of the directory is same as the one that you may generate yourselves by running our tools.

If you run the following three commands

```
$ make table1
$ make table2
$ make table3
```

you can get the result tables used in our FSE paper. The tables will be printed in your terminal in the LaTeX format.

To create the result tables using newly generated data, please run `make fetch-data`. It will fetch the data under the current benchmark directory to the `raw-data` directory. After that, you may run the three commands (`make table1`, `make table2` and `make table3`) to create the new result tables.

6. REFERENCES

[1] JSdep. URL: https://github.com/sch8906/JSdep.

[2] S. Artzi, J. Dolby, S. H. Jensen, A. Moller, and F. Tip. A framework for automated testing of JavaScript web applications. In *International Conference on Software Engineering*, pages 571–580, 2011.

[3] L. De Moura and N. Bjørner. Z3: An efficient SMT solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.

[4] GitHub. URL: https://github.com/.