JSAVER: JavaScript Static Analyzer via ECMAScript Representations

Jihyeok Park
Oracle Labs
Brisbane, Australia
jihyeok.park@oracle.com

Seungmin An
KAIST
Daejeon, South Korea
h2oche@kaist.ac.kr

Sukyoung Ryu
KAIST
Daejeon, South Korea
sryu.cs@kaist.ac.kr

ABSTRACT

This document describes the artifact package accompanying the ESEC/FSE 2022 paper "Automatically Deriving JavaScript Static Analyzers from Specifications using Meta-Level Static Analysis." The artifact includes the source code of JSAVER, the accepted paper, a companion report, ECMA-262 (JavaScript language specification) open-source repository as a git submodule, and scripts to replicate the evaluation results presented in the paper. JSAVER stands for a JavaScript Static Analyzer via ECMAScript Representation. It is the first tool that automatically derives JavaScript static analyzers from language specifications using an *interpreter*-based approach called meta-level static analysis instead of traditional a *compiler*-based approach. It extends another tool JISET to extract JavaScript definitional interpreters written in IR_{ES}, an intermediate representation for ECMAScript, from diverse versions of ECMA-262.

KEYWORDS

JavaScript, definitional interpreter, meta-level static analysis

1 GETTING STARTED GUIDE

The source code of JSAVER and the dataset of our study are publicly available at https://doi.org/10.5281/zenodo.6668789, and the latest version is maintained as a GitHub repository:

```
$ git clone --recurse-submodules \
    https://github.com/kaist-plrg/jsaver.git
```

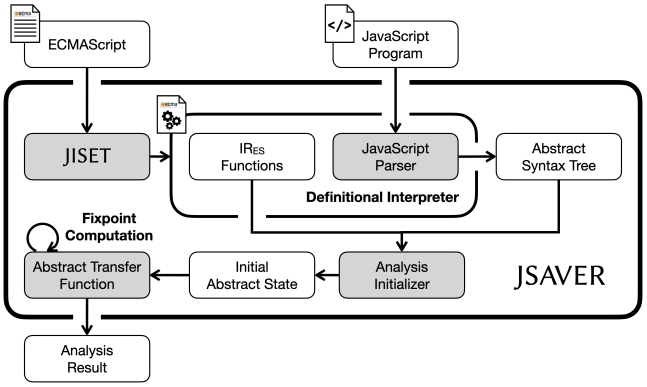
Please see `INSTALL.md` for the detailed guide on installation and how to use this artifact.

Additionally, we packaged the artifact in a docker container. If you want to skip the environment setting, we recommend you to use it. You can install the docker by following the instruction in https://docs.docker.com/get-started/ and downlaod our docker image with the following command:

```
$ docker pull jhnaldo/icse-21-jest
$ docker run -it -m=16g --rm \
    jhnaldo/fse22-jsaver
# user: guest, password: guest
```

Note that the docker image is 3GB large thus be patient when you download it and please assign more than 16GB memory for the docker engine.

2 OVERALL STRUCTURE



JSAVER consists of two phases: 1) definitional interpreter extraction and 2) meta-level static analysis.

2.1 Definitional Interpreter Extraction

We utilize another tool JISET, a JavaScript IR-based Semantics Extraction Toolchain,[1] to extract JavaScript definitional interpreters from given ECMA-262. In this artifact, we extracted the definitional interpreter from ES2021 (ES12), the latest version of ECMA-262, and manually filled out essential steps of its not-yet-compiled parts. It consists of two different main parts for semantics and syntax of JavaScript. For semantics, it compiles abstract algorithms in ECMA-262 to corresponding IR_{ES} functions. For syntax, it generates a JavaScript parser in Scala.

2.2 Meta-Level Static Analysis

JSAVER performs a *meta-level static analysis* with JavaScript as its *defined*-language and IR_{ES} as its *defining*-language. Thus, it indirectly analyzes a JavaScript program by analyzing IR_{ES} functions with the AST of the program as an argument. Using the generated parser, it first parses a given JavaScript program to produce an Abstract Syntax Tree (AST). Then, Analysis Initializer constructs an initial abstract state with the extracted IR_{ES} functions and the produced AST. Finally, JSAVER computes the fixpoint of Abstract Transfer Function with the initial abstract state.

It utilizes a worklist algorithm to update the abstract state per control point, a pair of the following two components:

- A node in control-flow graph of the extracted definitional interpreter
- A view that represents an analysis sensitivity

[1] https://github.com/kaist-plrg/jiset