Sherlock: A Profiling Tool to Find Opportunities for Early Property Initialization

Tarek Auel, David Güsewell
tarek.auel@gmail.com, d.guesewell@web.de

TECHNISCHE UNIVERSITÄT DARMSTADT

Introduction

With Sherlock, we introduce a new profiling tool to find opportunities for **early object property and array element initialization** as shown in Listing 1. In order to do this, the analysis has to keep track of the usage of the elements or properties. The **expected benefits** for such early property initializations are (i) **improved code readability**, and (ii) **improved performance** because the computation time spent in **push** and **reverse** calls is now saved.

```
1 var a = [1, 2, 3]
2 a.push(4);
3 b = a.reverse();
4 /*
5 optimization: var a = [4, 3, 2, 1]
6                b = a;
7 */
```

Listing 1: Example for early property initializationn

The main goals of Sherlock are to find these opportunities for early property initialization and keep the number of false positives as low as possible at the same time. Due this, in some use cases Sherlock prefers not to optimizing a value instead of running in a potential false positive situation.

Methods

Overall **Sherlock is an implemented plugin for Jalangi**. Jalangi is a dynamic framework for JavaScript. By transforming the code and adding hooks, it allows the user to monitor almost every operation performed bythe execution. Sherlock overrides the API exposed by Jalangi in order to perform the desired dynamic analysis. Before the code is analyzed by Sherlock it is instrumented with a simple literal using Esprima, Estraverse and Escodegen in order to track the end of conditions. This is done, because it is not possible to track the end of conditions with Jalangi.

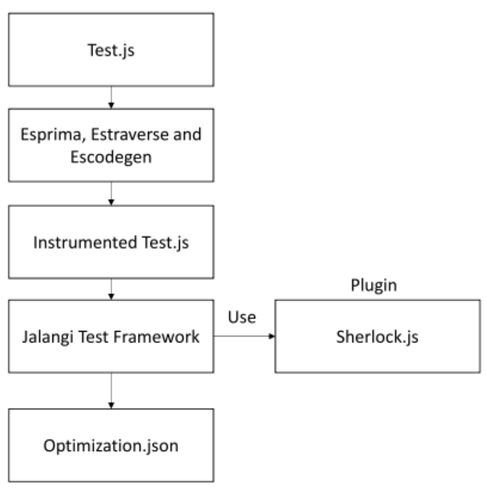

Figure 1: Architecture of Sherlock

Results

Sherlock was evalutad with a set of self written test-cases. The results show Sherlock is able to detect different early property optimizations. Due the fact that one goal of Sherlock was to minimize the number of false positives as low as possible, there are some cases which Sherlock does not support.

Supported early property optimizations
The cases which are fully supported by Sherlock are listed below in different code examples.

```
1 var a = [];
2 a[0] = 1;
3 //optimization: var a = [1];
4 var b = {};
5 b.a = 5;
6 //optimization: var b = {a: 5};
```

```
1 var a =[1,2,3,4];
2 a.reverse();
3 // optimization: var a = [4,3,2,1];
```

Listing 2: Examples for a typical potential early property and array element initialization

The left Example shows typical potential for early property and array element initilization which is covered by Sherlock. Sherlock is also able to optimize the JavaScript built in functions for arrays (right Example).
In the length example the only supported optimization would be made for *a*. *b* must not be optimized, because this would change the semantics. *c* and *d* show two corner cases that are not supported, due it is very hard to distinguish *c* and *d* using Jalangi.
Conditioned initialization should only be done, if the scope does allow it. In the conditioned example only can be optimized.

```
1  var a = [];
2  a[a.length] = 0;
3  a[1] = 1;
4  //optimization: var a =[0,1];
5  var b = [1, 2, 3];
6  console.log(b.length);
7  b[3] = 4;
8  //cannot be optimized
9  var c = [];
10 c[c.length - 1] = 0;
11 //cannot be optimized
12 var d = [];
13 d[0] = d.length - 1;
14 //cannot be optimized
```

```
1 var a = []
2 if (cond) {
3   a[0] = 1;
4   var b = []
5   b[0] = 1;
6   //optimization: var b =[1];
7 }
```

Listing 3: Array.length and conditioned example

Not supported early property optimizations
Some cases are not covered by Sherlock, due the fact that one of Sherlock's main goal is to keep the number of false positives as low as possible and to limit the scope of the tool. This includes objects that are created using a constructor or if a function is used in the initialization. The impact of optimization suggested by Sherlock can be seen in Figure ?.

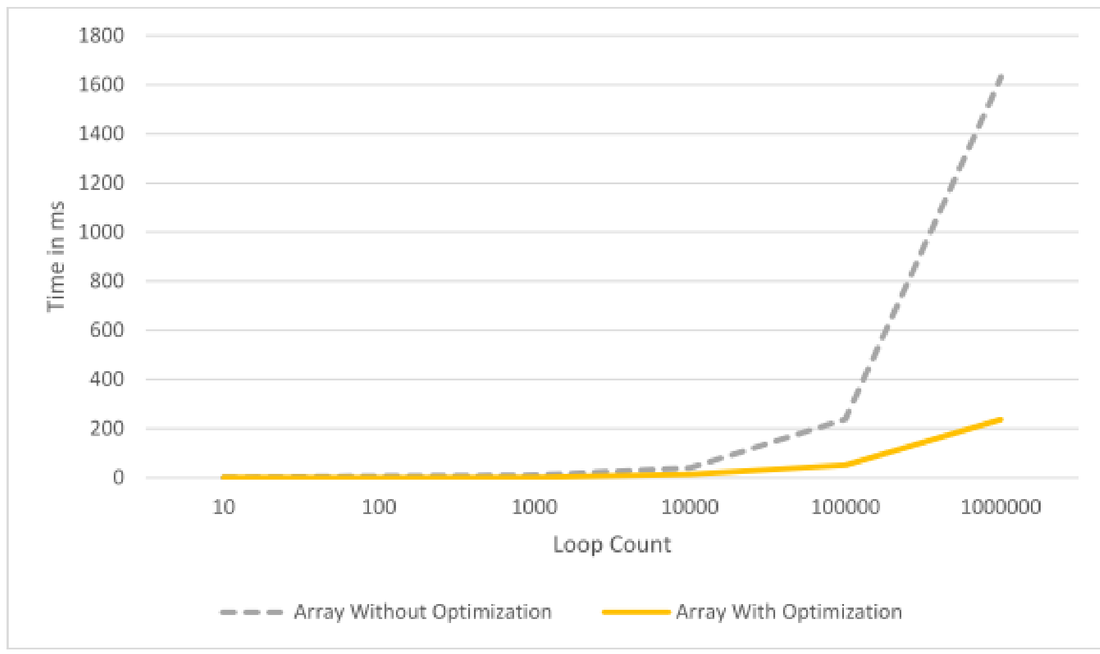

Figure 2: Benchmark for simple array initialization. Array length is 100.

References

[1] J. M. Smith and A. B. Jones. *Book Title*. Publisher, 7th edition, 2012.

[2] A. B. Jones and J. M. Smith. Article Title. *Journal title*, 13(52):123–456, March 2013.

Conclusion

We introduced a new profiling tool to find opportunities for early property initialization, named Sherlock. While there is still a long way to go to find all possible early property optimizations our results show that Sherlock is able to find a broad range of them. Furthermore Sherlock should be able to check every JavaScript file provided to it.

Acknowledgements

Fusce mattis tellus ac odio imperdiet lobortis. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Phasellus commodo blandit euismod. Ut porttitor cursus magna.