Mechanizations of JavaScript

ERC Workshop (25 July to 28 July, 2012)

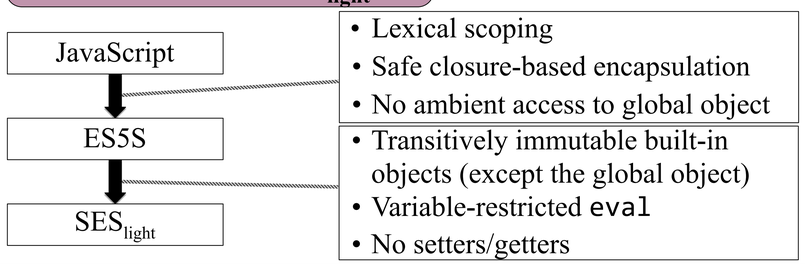
SungGyeong Bae (KAIST) Sukyoung Ryu (KAIST)

Introduction

Since the JavaScript standard is vast and informal, it is not amenable to apply conventional proof techniques. As a result, there are several studies to make formal syntax and semantics of JavaScript. In this poster, we introduce two of such studies and present our research.

SES_{light} [1]

From JS to SES_{light}

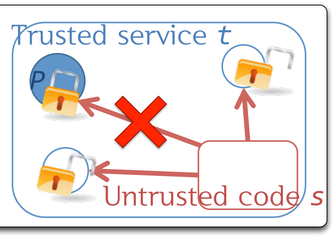


- Let `t` be the trusted code and `s` be the untrusted code. Then, the overall program that executes in the system can be expressed as :

```
t; var un; eval(s, "api", "un").
```

Confinement Property

- A trusted service `t` safely encapsulates a set of forbidden allocation-site labels *P*.

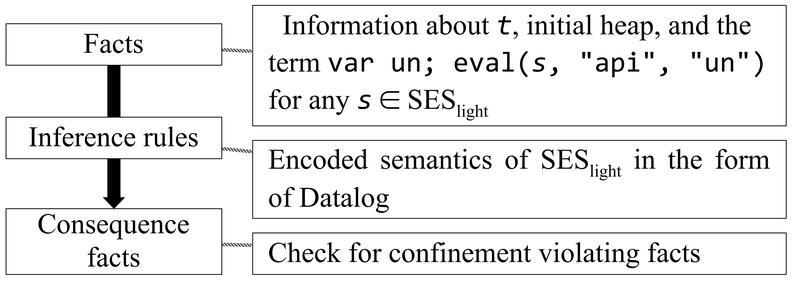


Proof Overview

Datalog

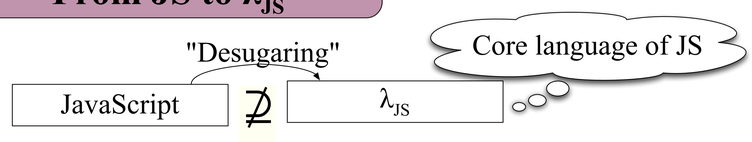
A Datalog program consists of *facts* and *inference rules*. We can get the set of all consequence facts that can be obtained by applying the inference rules to the facts, upto a fixed point.

Procedure Overview



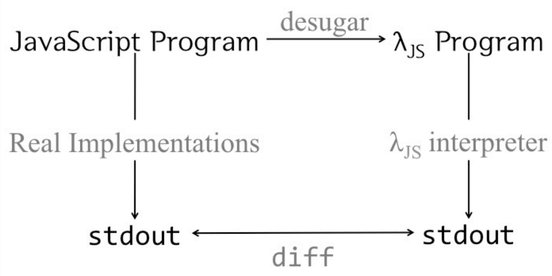
λ_{JS} [2]

From JS to λ_{JS}



- JavaScript → λ_{JS} compiler by Haskell
- λ_{JS} interpreter by PLT Scheme
- (recent) Mechanized by Coq

Testing Strategy



Bugs

- Various type-setting errors
- A semantic error

Soundness of λ_{JS} [3]

Theorem

For all λ_{JS} programs, p, either:

1. $p \rightarrow^* v$,
2. $p \rightarrow^*$ ***err***, or
3. $p \rightarrow^* p_2$, and there exists p_3 such that $p_2 \rightarrow p_3$.

Coq Code

```
Definition progress := forall sto e,
  lc e ->
  val e \/ e = exp_err \/
  (exists e', exists sto', step sto e sto' e').

Definition preservation := forall sto1 e1 sto2 e2,
  lc e1 ->
  step sto1 e1 sto2 e2 ->
  lc e2.
```

Our Research

λ_{JS} with Modules [4]

We introduce a module system to JavaScript, and we formally describe its semantics via desugaring to a slightly modified λ_{JS}.

JavaScript + module —desugar→ λ_{JS} + α

Current Work

We are working on mechanizing the soundness of the modified λ_{JS} in Coq.

Soundness?

λ_{JS} + α

References

SES_{light}

[1] A. Taly, U. Erlingson, J. C. Mitchell, M. S. Miller, J. Nagra, *"Automated Analysis of Security-Critical JavaScript APIs"*, SP 2011

λ_{JS}

[2] A. Guha, C. Saftoiu, S. Krishnamurthi, *"The Essence of JavaScript"*, ECOOP 2010

[3] http://brownplt.github.com/2012/06/04/lambdajs-coq.html

λ_{JS} with modules

[4] Seonghoon Kang and Sukyoung Ryu. *"Formal Specification of a JavaScript Module System"*, OOPSLA 2012