Certified Analyses of JavaScript

Martin Bodin

PhD Advisors: Alan Schmitt and Thomas Jensen
Why JavaScript?

- Used everywhere, we don’t even notice it.
Usages of **JavaScript**

- Executed in unknown context (mash-up).
What Do We Want?

```javascript
function untrusted () {
    // ...
}

var secret = // ...

wrapFunction (untrusted)
```

**Objective**

Automatically prove that `wrapFunction` isolates the untrusted code:
- `secret` should not be accessed by a call to `untrusted`. 
JavaScript is Difficult to Analyse

- Very dynamic:

```javascript
var x = "global"
var o = { x : "attribute" }
var f = function () { with (o) { return x } }

f () // "attribute"

delete o.x
f () // "global"

o = { x : 42 }
f () // 42
```

- A lot of non-trivial effects.
JavaScript is very Difficult to Analyse
JavaScript is very Difficult to Analyse

' martin'
JavaScript is very Difficult to Analyse

```javascript
Array.prototype.toString = function () { return '42' }
```

→ 'undefinedftundefinedundefinedu'
1. About JavaScript
   - Usage
   - Pitfalls

2. Related Work
   - Analyses
   - Semantics of JavaScript

3. JSCert

4. From Semantics To Abstract Interpretation

5. Future Works
The Usual Approach of Analyses

- Restricting to a sublanguage.

### Issues

- No access to `this`, `Date`, `Math.random`, etc.
- No formal proof... while the language is complex.
A Lack of Formal Proof

- JSLint blocks:
  
  ```
  o.___nodes___ = []
  ```

- But accepts:
  
  ```
  o = {
      "___nodes___": []
  }
  ```

- This is enough to trick AdSafe!

- We need a full formal semantics.

---

**ADsafe**

Type-Based Verification of JavaScript Sandboxing

Joe Gibbs Politz  Spiridon Aristides Eliopoulos  Arjun Guha  Shriram Krishnamurthi

*Brown University*
JavaScript’s semantics is huge!

- How to verify we did not make mistakes?
The Problem About JavaScript’s Formal Semantics

JavaScript’s semantics is huge!

- How to verify we did not make mistakes?

Closeness to the standard

<table>
<thead>
<tr>
<th>JSSec (APLAS 2008)</th>
</tr>
</thead>
</table>

Testing

<table>
<thead>
<tr>
<th>λJS (ECOOP 2010)</th>
</tr>
</thead>
</table>
The Problem About JavaScript’s Formal Semantics

JavaScript’s semantics is huge!

- How to verify we did not make mistakes?

  Closeness to the standard
  - JSSec (APLAS 2008)

  Testing
  - λJS (ECOOP 2010)

  JSCert & JSRef (POPL 2014)
Inductive red_stat : state → execution_ctx → ext_stat → out → Prop :=

(** If statement (12.5) *)

| red_stat_if : ∀S C e1 t2 t3opt y1 o,
  red_spec S C (spec_expr_get_value_conv spec_to_boolean e1) y1 →
  red_stat S C (stat_if_1 y1 t2 t3opt) o →
  red_stat S C (stat_if e1 t2 t3opt) o

| red_stat_if_1_true : ∀S0 S C t2 t3opt o,
  red_stat S C t2 o →
  red_stat S0 C (stat_if_1 (vret S true) t2 t3opt) o

| red_stat_if_1_false : ∀S0 S C t2 t3 o,
  red_stat S C t3 o →
  red_stat S0 C (stat_if_1 (vret S false) t2 (Some t3)) o

| red_stat_if_1_false_implicit : ∀S0 S C t2,
  red_stat S0 C (stat_if_1 (vret S false) t2 None) (out_ter S resvalue_empty)

(* ... *).
How to Use JSCert?

- 793 rules.
- Pretty-big-step semantics.

How to automatically create a certified program analysis with only this?

The dream

- Abstract interpretation.
- Specify abstract domains.
- Specify abstract operations corresponding to the concrete ones.
- ... And getting an abstract semantics for free!
About Abstract Interpretation

- **Program Source**
- **Reduction**
- **Facts**
- **Unsafe States**
- **Abstract Semantics**
- **Abstract Traces**

Diagram: Flow of concepts from program source through reduction to facts and unsafe states.
About Abstract Interpretation

- Program Source
- Reduction
- Semantics
- Analysis
- Abstract Semantics
- Facts
- Unsafe States
- Property Definition
Reusing D. Schmidt’s Approach

Natural-Semantics-Based Abstract Interpretation (Preliminary Version)

David A. Schmidt
Department of Computing and Information Sciences
Kansas State University *
Natural-Semantics-Based Abstract Interpretation
(Preliminary Version)

David A. Schmidt
Department of Computing and Information Sciences
Kansas State University *

→ Making it scalable  
→ Using pretty-big-step  
→ In Coq
A Framework to Build Abstract Semantics That Scale Up

\[ \text{If1True}(s_1, s_2) \]

\[ \frac{s_1, S \Downarrow o}{\text{if}_1 s_1 s_2, (v, S) \Downarrow o} \]

\[ v = \text{true} \]

\[ \text{If1True}(s_1, s_2) \]

\[ \frac{s_1, S^\# \Downarrow o^\#}{\text{if}_1 s_1 s_2, (v^\#, S^\#) \Downarrow o^\#} \]

\[ v^\# \equiv^\# \text{true^\#} \]

How to build the abstract rule?

- **Same rule name** \text{If1True}(s_1, s_2) and terms \text{if}_1 s_1 s_2 and \( s_1 \).
- **Function** \( up : (v, S) \mapsto S \) to change.
- **Side-condition** \( cond_n : (v, S) \mapsto v = \text{true} \) to change.
A Framework to Build Abstract Semantics That Scale Up

\[
\frac{\text{If1True} (s_1, s_2)}{s_1, S \Downarrow o} \quad \frac{\text{If1True} (s_1, s_2)}{s_1, S^\# \Downarrow o^\#}
\]

\[
\frac{\text{if}_1 \ s_1 \ s_2, (v, S) \Downarrow o}{v = \text{true}} \quad \frac{\text{if}_1 \ s_1 \ s_2, (v^\#, S^\#) \Downarrow o^\#}{v^\# \equiv^\# \text{true}^\#}
\]

How to build the abstract rule?

- Same rule name \text{If1True} (s_1, s_2) and terms \text{if}_1 \ s_1 \ s_2 and s_1.
- Function \text{up} : (v, S) \mapsto S\text{ to change.}
- Side-condition \text{cond}_n : (v, S) \mapsto v = \text{true} \text{ to change.}
A Framework to Build Abstract Semantics That Scale Up

If\( \text{if\_true} (s_1, s_2) \)
\[ s_1, S \Downarrow o \]
\[ \text{if\_true} s_1 s_2, (v, S) \Downarrow o \]
\[ v = \text{true} \]

We define the new transfert functions.
We get an abstract semantics.

Theorem
If every abstract rule correctly relates with the corresponding concrete rule, then the abstract semantics is correct with respect to the concrete one.
Hypothesis Pr : ∀ n, propagates (arule n) (rule n).

Hypothesis acond_correct : ∀ n asigma sigma,
    gst asigma sigma →
    cond n sigma →
    acond n asigma.

Theorem correctness : ∀ t asigma ar,
    aeval _ _ _ t asigma ar →
    ∀ sigma r,  gst asigma sigma →
    eval _ t sigma r →
    gres ar r.
Global Scheme of the Approach

"Real World"

Huge Semantics

Lattices and Domains

Abstract Semantics

Automatically Generated

Theorem

JSCert

Analyser
1 About JavaScript
   - Usage
   - Pitfalls

2 Related Work
   - Analyses
   - Semantics of JavaScript

3 JSCert

4 From Semantics To Abstract Interpretation

5 Future Works