Static Analysis with Demand-Driven Value Refinement

Benno Stein  
University of Colorado Boulder

Benjamin Barslev Nielsen  
Aarhus University

Bor-Yuh Evan Chang  
University of Colorado Boulder

Anders Møller  
Aarhus University

Problem

Even a minor precision loss in whole-program JavaScript static analysis can incur a huge slowdown as a result of dynamic property access.

```
// library code
var src = {
  foo: function f1(){...},
  bar: function f2(){...}
};
var lib = {};
for (var p in src) {
  lib[p] = src[p];
}
lib.foo();
```

--

Our Approach

We augment a whole-program dataflow analysis (TAJSvr) with a value refiner (VRs): a very precise and targeted analysis that non-monotonically refines the base analysis abstract state on the fly, increasing precision at crucial locations like the write to `lib[p]` above.

```
T
```

Refinement Analysis

Value refinement answers queries of the form “which values can this memory location hold at this abstract state?”, providing relational precision to a non-relational underlying analysis.

```
\begin{align*}
\text{Assume} & \quad \hat{h} = x \mapsto \hat{x} \\
\text{Consequence} & \quad \langle \hat{h} \land \hat{x} \land \pi \rangle \text{ assume } x \langle \hat{h} \land \pi \rangle
\end{align*}
```

The analysis is defined in terms of refutation-sound triples of the form `\langle \varphi \rangle s \langle \varphi' \rangle` that hold if any concrete run through `s` that ends in `\varphi'` must have started in `\varphi`, over-approximating the backwards semantics.

Evaluation

We evaluate the demand-driven value refinement technique by implementing a JavaScript type analysis TAJSvr and comparing it against two state-of-the-art JS analysis tools:

```
T
```

Abstract

Static analysis tools for JavaScript must strike a delicate balance, achieving the level of precision required by the most complex features of target programs without incurring prohibitively high analysis time. For example, reasoning about dynamic property accesses sometimes requires precise relational information connecting the object, the dynamically-computed property name, and the property value. Even a minor precision loss at such critical program locations can result in a proliferation of spurious dataflow that renders the analysis results useless.

We present a technique by which a conventional non-relational static dataflow analysis can be combined soundly with a value refinement mechanism to increase precision on demand at critical locations. Crucially, our technique is able to incorporate relational information from the value refinement mechanism into the non-relational domain of the dataflow analysis.

We demonstrate the feasibility of this approach by extending an existing JavaScript static analysis with a demand-driven value refinement mechanism that relies on backwards abstract interpretation. Our evaluation finds that precise analysis of widely used JavaScript utility libraries depends heavily on the precision at a small number of critical locations that can be identified heuristically, and that backwards abstract interpretation is an effective mechanism to provide that precision on demand.