Information flow Analysis for JavaScript

via a dynamically typed language with staged metaprogramming

UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE

http://mjolnir.cs.ox.ac.uk/web/slamjs/

Martin Lester   Luke Ong   Max Schäfer

Web 2.0 applications written in JavaScript handle sensitive information.
Information flow analysis is an important security problem.

Information flow...

...can be direct

if$(x)\{h\}$ **else**$\{$**false**$\} \to h$ or **false**

h directly affects the result

...can be indirect

if$(h)\{$**true**$\}$ **else**$\{$**false**$\} \to$ **true** $(= h)$ or **false** $(= h)$

h indirectly affects the result

We want to determine statically which subexpressions can affect the result.

JavaScript is hard!

- **eval** lets programs run strings as code

So use staged metaprogramming instead:

- limit manipulation to splicing well-formed code templates
- still have to handle static and dynamic scoping and loss of alpha equivalence

(**fun**$(x)\{x+1\}$)

↑ evaluates like

eval("(fun(x) {" + "x + 1" + "})")

↓ with staged metaprogramming

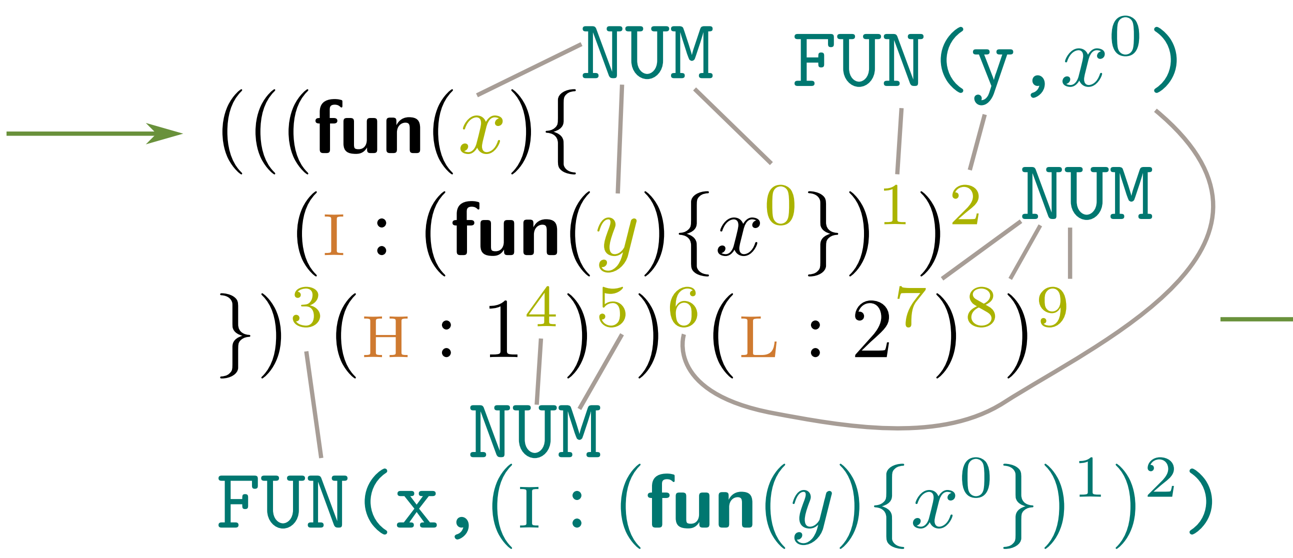
run (**box** (**fun**$(x)\{$**unbox** (**box** $(x+1)$)$\}$))

Our Analysis

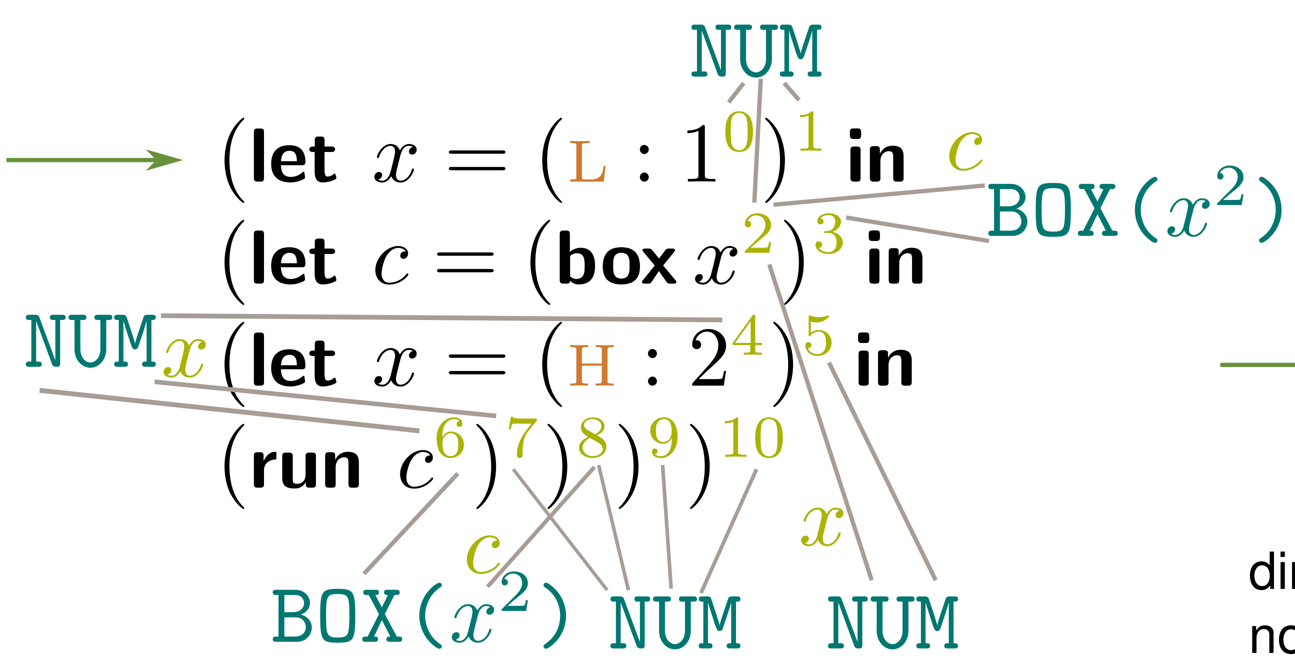
1. Mark expressions of interest ⟶ 2. CFA ⟶ 3. Generate information flow constraints

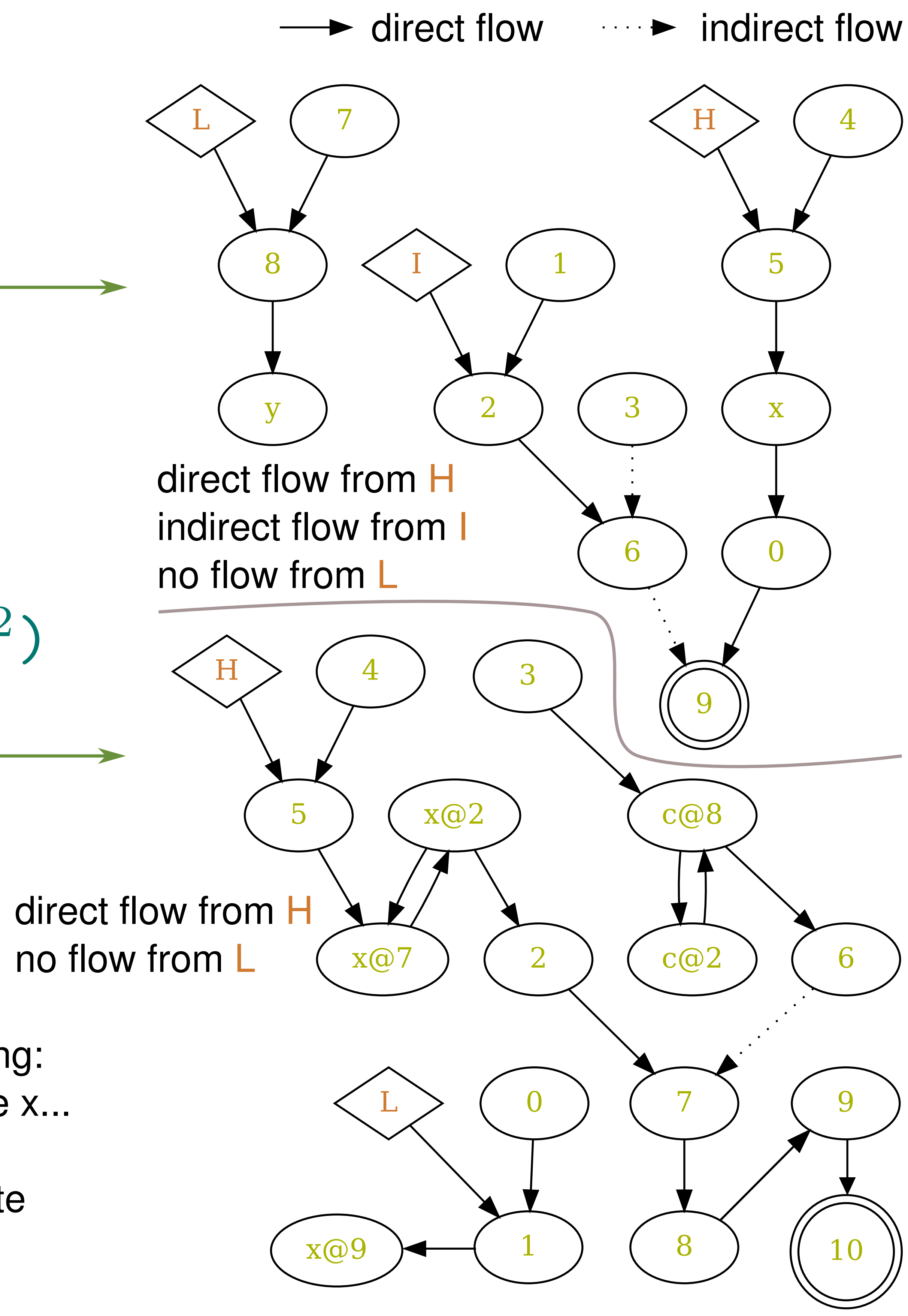
$((\textbf{fun}(x)\{$
$\quad \texttt{I} : (\textbf{fun}(y)\{x\})$
$\})(\texttt{H} : 1))(\texttt{L} : 2)$
$\to^* 1$

Use different markers for different security levels.



let $x = \texttt{L} : 1$ **in**
let $c =$ **box** x **in**
let $x = \texttt{H} : 2$ **in**
run c
$\to^* 2$





CFA: standard analysis that determines which abstract values occur at program points.

Extension to handle staged metaprogramming:
- abstract value BOX(x) models a code value x...
- ...or anything x evaluates to – this keeps the needed abstract values finite

What we have done...

- first analysis of its kind
- an implementation in OCaml
- a soundness proof in Coq

Still to do...

- automate transformation to staged metaprogramming
- improve precision of analysis of strings, numbers...
- support full JavaScript: mutable state, exceptions...