An Executable Formal Semantics for PHP

Daniele Filaretti & Sergio Maffeis

Department of Computing, Imperial College London

www.phpsemantics.org

Imperial College London

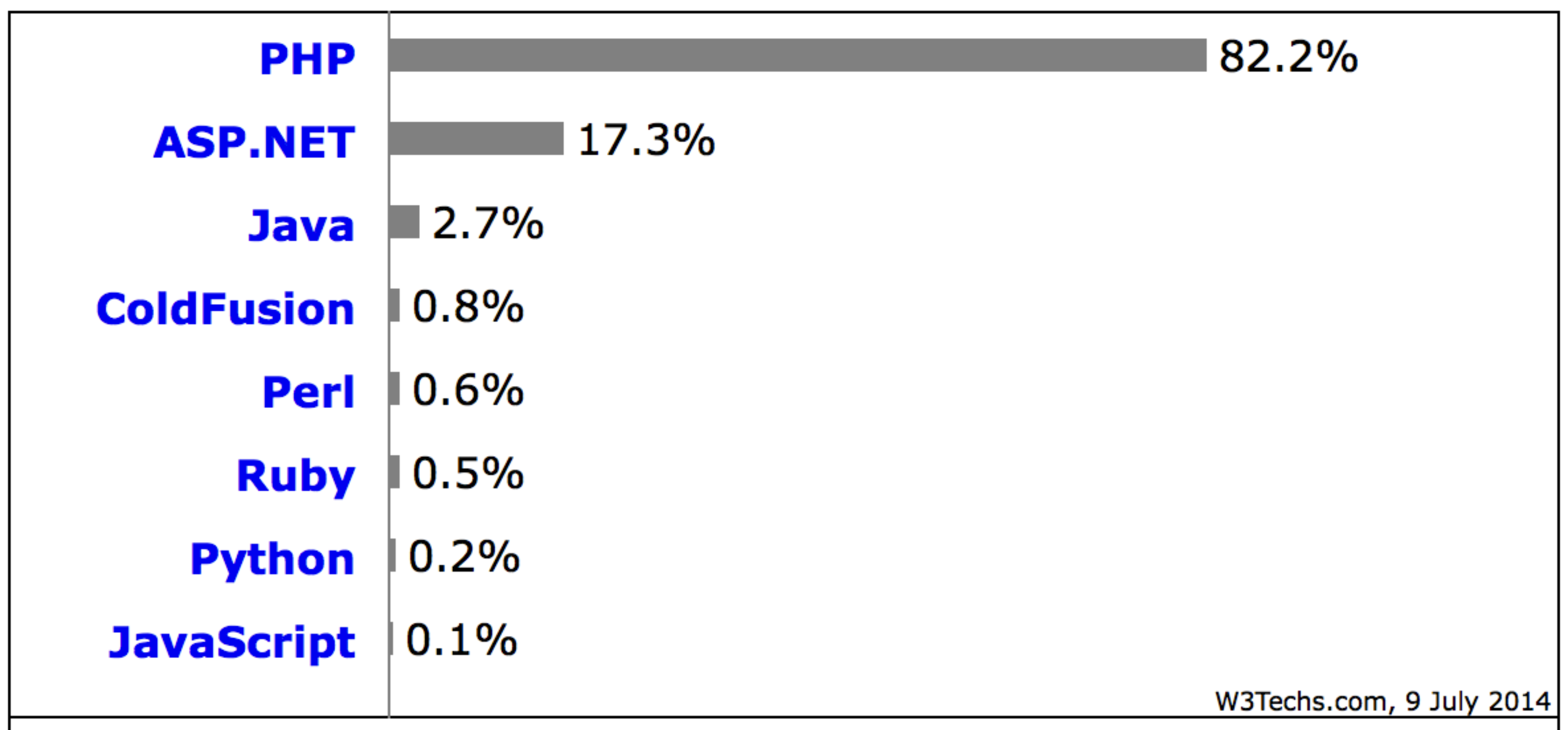
Motivation

PHP is among the most popular programming languages and it is the most common choice for server-side scripting. It powers applications such as Facebook, Yahoo, Wikipedia and Wordpress and, not surprisingly, it is the place where most vulnerabilities in web application are found. Vulnerabilities such as XSS and SQLi are an ideal target for static analysis, but PHP is notoriously **hard to analyse** (due to its dynamic features). Its semantics is poorly understood by both developers and security specialists, and the **absence of a written language specification** or even of a detailed and comprehensive documentation doesn't help. As a result, **existing bug-finding tools don't provide any formal guarantee of soundness or coverage** and miss many existing bugs. Our goal is to develop a new generation of semantics-based static analysis tools targeting web application security, based on a rigorous formal model of the PHP language [1].

A
```
$a = array("one");
$c = $a[0].($a[0] = "two");
echo $c;
```
out: ???

B
```
$a = "one";
$c = $a.($a = "two");
echo $c;
```
out: ???

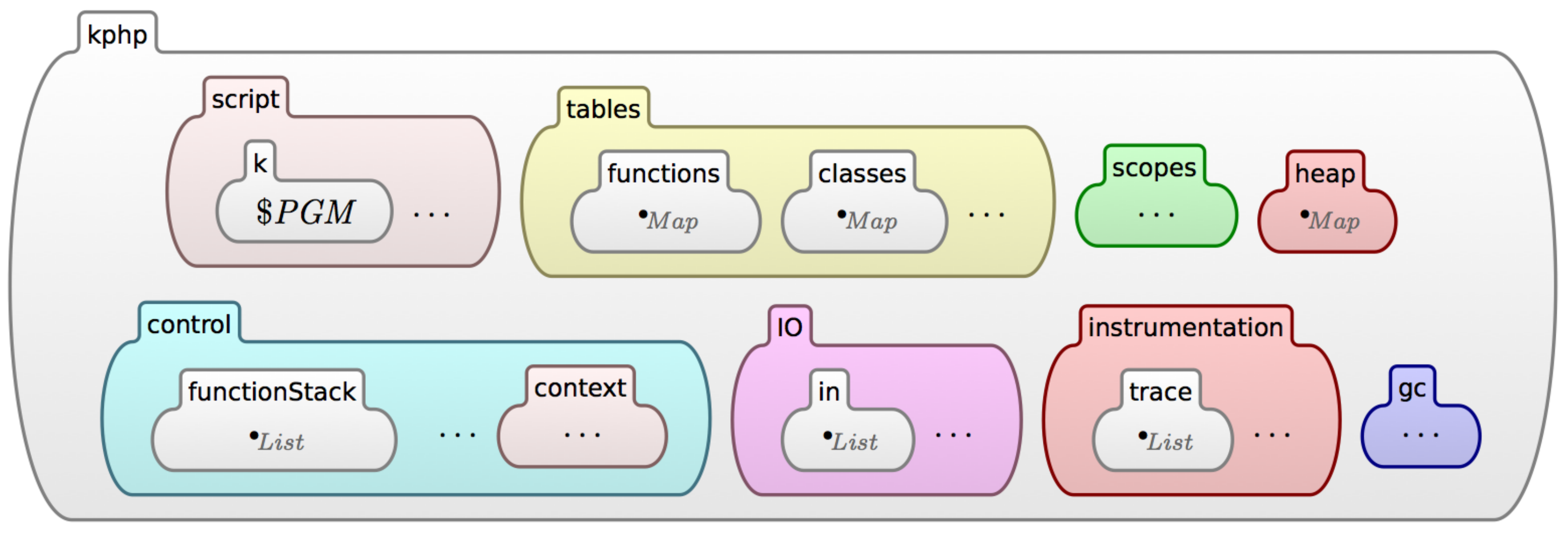
A: "onetwo"
B: "twotwo"



Percentages of websites using various server-side programming languages
Note: a website may use more than one server-side programming language

$\mathbb{K}$PHP: formalising PHP in $\mathbb{K}$

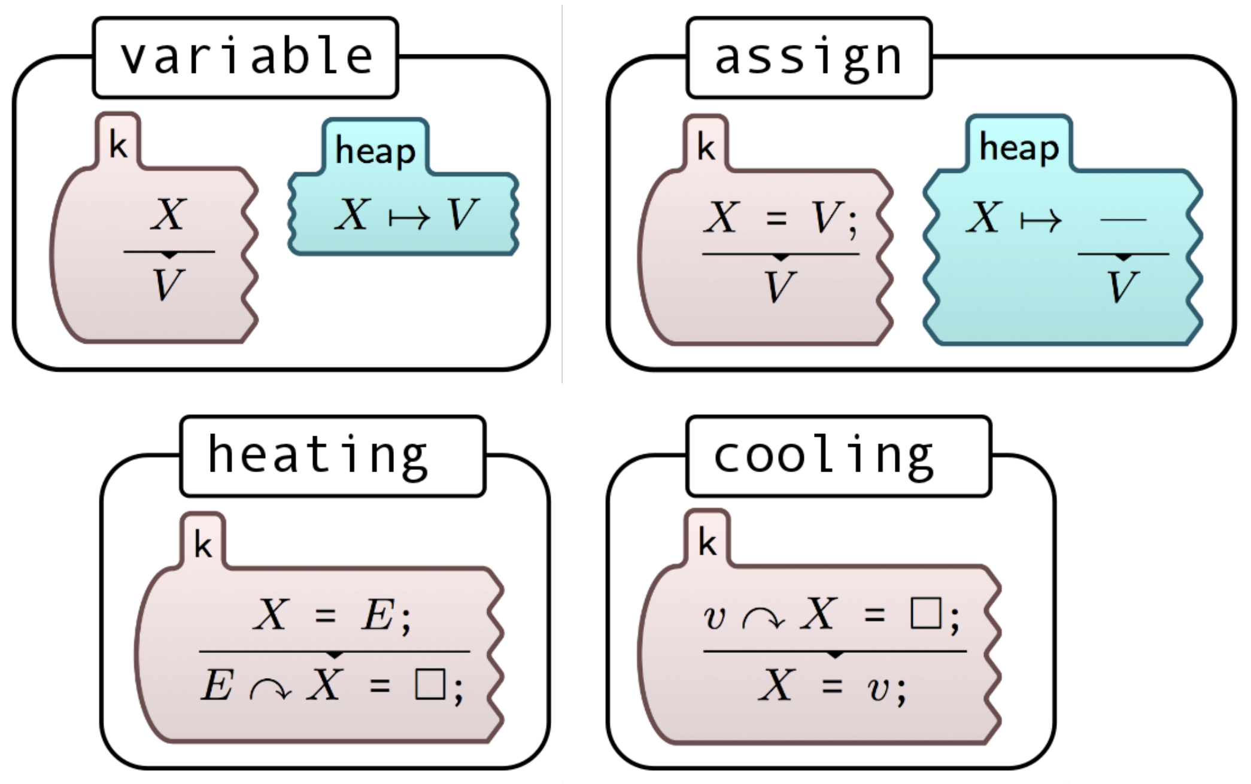
We formalise PHP in $\mathbb{K}$ [2], a semantics framework based on rewriting logic and implemented on top of the Maude system. States, or *configurations*, are represented as labeled, possibly nested *cells*, each containing part of the program state (e.g.: fragments of program, stacks, environments, function definitions etc.).



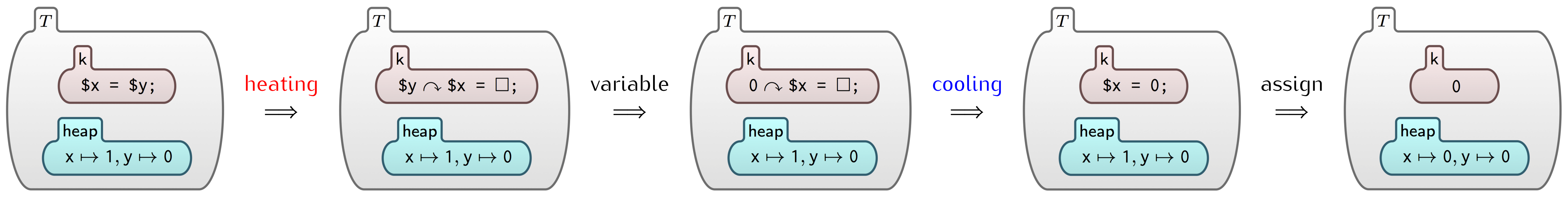
After a program is placed into the k cell, the rewrite rules can be applied, causing transitions to new configurations. A $\mathbb{K}$ semantics has a rigorous meaning as a term rewriting system, hence it is suitable for formal reasoning. It is also **directly executable**, enabling a tight design/test loop. The $\mathbb{K}$/Maude tool-chain provides support for **LTL model checking and symbolic execution**.

$\mathbb{K}$ rules

$\mathbb{K}$ rules can be *computational* or *structural*. Computational rules give meaning to atomic constructs. They apply once all subterms are in the desired form and cause a state transition.

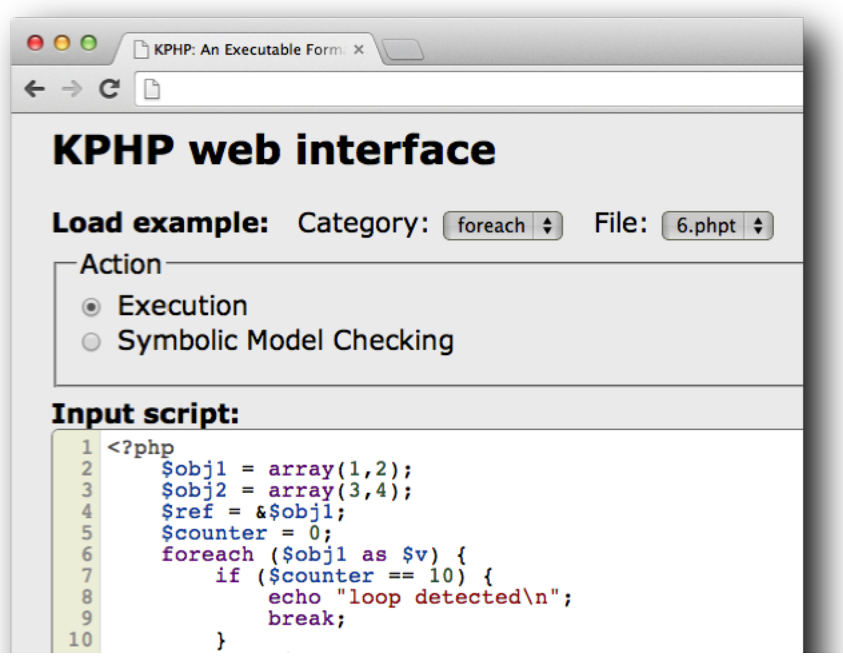


Heating and *cooling* rules are examples of *structural* rules. Their role is to define evaluation strategies and they don't count as computational steps.



$\mathbb{K}$PHP Interpreter and validation

The executable semantics doubles as a PHP interpreter. We validate our formalisation by **passing all the tests for the official PHP Zend implementation** targeting the portion of the language we cover. To improve coverage, we augment the test suite with additional cases. $\mathbb{K}$PHP, together with a web interface and examples are available online at http://phpsemantics.org.



KPHP web interface

Load example: Category: foreach File: 6.phpt

Action
[✓ Execution]
Symbolic Model Checking

Input script:
```
<?php
    $obj1 = array(1,2);
    $obj2 = array(3,4);
    $ref = &$obj1;
    $counter = 0;
    foreach ($obj1 as $v) {
        if ($counter == 10) {
            echo "loop detected\n";
            break;
        }
```

References

[1] D.Filaretti, S.Maffeis, *An Executable Formal Semantics for PHP*, ECOOP'14.
[2] G. Roşu, T.F. Şerbănuţă, $\mathbb{K}$ *Overview and SIMPLE Case Study*, ENTCS'13.

Analysis of PHP programs

LTL model checking

By **extending LTL** with predicates over configurations we are able to verify properties of PHP program.

```
function foo($x){
    global $y;
    $x = &$y;}
foo(1);
```

- $\neg\Box$`eqTo(fv(foo,var(x)),val(0))`
- $\Diamond$`alias(gv(var(y)),fv(foo,var(x)))`

Initial case studies:
- `phpMyAdmin/PMA_isValid`
- `PHP/hash_pbkdf2`

Taint Analysis

Semantics-based prototype tool to detect taint flows in PHP code. Low code coverage but precision comparable to state-of-the-art. [S. Yuwen, MSc Project, 2013]

Towards Web App Security

We are building an **abstract interpretation framework** for PHP targeted to the static analysis of security properties of web applications.

Work funded by EPSRC grant EP/I004246/1