### JSConTest Contract-Driven Testing of JavaScript Code

#### Phillip Heidegger, Peter Thiemann

Albert-Ludwigs-Universität Freiburg

01.07.2010



andom Testing

uided Random Testi

Monitoring

Case Study

Related Wor

Conclusion o

#### Introduction

#### JavaScript is the language of the Web

ndom Testing

Buided Random Test

Monitoring

Case Study

Related Wor

Conclusion o

#### Introduction

#### JavaScript is the language of the Web

# 99.3%

of all websites use JavaScript (http://w3techs.com)

ndom Testing

Buided Random Test

Monitoring

Case Study

Related Work

Conclusior o

#### Introduction

#### JavaScript is the language of the Web

# 99.3%

of all websites use JavaScript (http://w3techs.com)

How do we ensure that they work correctly and reliably?

Introduction	Random Testing	Guided Random Testing	Monitoring	Case Study	Related Work	Conclusion
0000	0000	000	00	00	0	0

# **Unit Testing**

0000

#### What do we want to test?

- Does the program crash?
- Does the program behave as intended by the programmer? (expressed by contracts)

andom Testing

uided Random Testi

Monitoring

Case Stud

Related Worl

Conclusion o

#### Contributions

- Contract language for JavaScript
- Random testing based on contracts
- Guided random testing to improve coverage
- Contract monitoring
- Implemented in the JSConTest tool

```
roduction Random Testing
```

Guided Random Tes

Monitoring

Case Study

Related Work

Conclusion o

# Simple Type Contracts

Monitoring

Case Study

Related Work

Conclusion o

# Random Testing with Contracts

- Type signature-like contracts
- Type contract in argument position:
   ⇒ random generator
- Type contract in result position:
  - $\Rightarrow$  contract checker

Random Testing

Guided Random Testing

Monitoring 00 Case Stud

Related Worl

Conclusion o

#### **Contract Demo**

Demo - ex1.html

Random Testing

ided Random Testing

Monitoring

Case Study

Related Work

Conclusion o

### **Distribution of Test Values**

Monitoring

Case Study

Related Work

Conclusion o

# **Distribution of Test Values**

• random generator for int uniformly distributed

$$\Rightarrow P(x=10) pprox 2^{-32}$$

⇒ uniformly distributed generators are not always a good choice

```
Introduction Random Testir
```

# **Guided Contract**

- - annotate the int contract with @numbers.

```
ntroduction Random Testin
```

Case Study

Related Work

Conclusion o

# **Guided Contract**

- $_{1}$  /\*\* (int@numbers,int)  $\rightarrow$  bool \*/
- <sup>2</sup> function p(x,y) {
- 3 **if (x != y)** {
- if (f(x) = x + 10) return "false"; // contract violation

```
5 };
```

```
<sup>6</sup> ر return true;
```

```
7 };
```

- annotate the int contract with @numbers.
- $\Rightarrow$  Changes the probability distribution
- ⇒ Generates random expressions with numbers from the source program
- $\Rightarrow$  Usually locates the violation in less than 10 test runs

Random Testing

Guided Random Testing

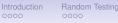
Monitoring 00 Case Study

Related Work

Conclusion o

#### Guided Contract – Demo

Demo - ex1a.html





Case Study

Related Work

Conclusion o

# **Guided Contract**

- $_{1}$  /\*\* (int@numbers,int@numbers,int@numbers)  $\rightarrow$  bool \*/
- <sup>2</sup> function fut\_1(x,y,z) {
- if ((x\*3+5==y\*5+4) && (x\*2-1==z\*9-1))
- 4 return "false"; // contract violation
- s return true;
- 6 };



Case Study

Related Work

Conclusion o

# **Guided Contract**

- $_{1}$  /\*\* (int@numbers,int@numbers,int@numbers)  $\rightarrow$  bool \*/
- <sup>2</sup> function fut\_1(x,y,z) {
- <sup>3</sup> if ((x\*3+5==y\*5+4) && (x\*2-1==z\*9-1))
- <sup>4</sup> return "false"; // contract violation
- s return true;
- 6 };
- complex conditional  $\rightarrow$  difficult to archive high coverage
- Our approach detects the violation in less than 5 sec

```
duction Random Tes
```

Case Study

Related Work

Conclusion o

# **Guided Contract for Objects**

- $_{1}$  /\*\* (object) ightarrow bool \*/
- <sup>2</sup> function h(x) {
- 3 if (x && x.p && x.quest)
- <sup>4</sup> return "false"; // contract violation
- s return true;
- 6 };
  - Blindly generating random objects does not lead to high coverage
  - How to guide the random generator for objects?

duction Random T

g Guided Random Testing

ng Monito

Case Study

Related Work

Conclusion O

# **Guided Contract for Objects**

- $_{1}$  /\*\* (object@labels) ightarrow bool \*/
- <sup>2</sup> function h(x) {
- 3 if (x && x.p && x.quest)
- 4 return "false"; // contract violation

```
5 return true;
```

```
6 };
```

- Annotation @labels
- Generator prefers to use the labels inside of the function body
- ⇒ Raises probability to generate a property with names p or quest

```
troduction Rando
```

Guided Random Testing

Monitoring •• Case Study

Related Work

Conclusior o

# **Contract Monitoring**

```
1 /** int \rightarrow int */

2 function f(x) { return 2 * x; };

3

4 /** (int,int) \rightarrow bool */

5 function g(x,y) {
```

```
<sup>6</sup> return (f(x * "3O") == 60);
```

7 }

• Where is the bug?

```
troduction Random
```

Monitoring • O Case Study

Related Work

Conclusion o

# **Contract Monitoring**

```
_1 /** int \rightarrow int */
```

```
2 function f(x) { return 2 * x; };
```

```
\begin{array}{l} 3 \\ 4 \end{array} /** (int,int) \rightarrow bool */ \\ 5 \\ 5 \\ 6 \\ return (f(x * "30") == 60); \\ 7 \end{array}
```

- Where is the bug?
- $\rightarrow~$  Programmer wrote O instead of zero.
  - Does not violate the contract of g.
  - But violates the contract of f.



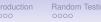
Case Study

Related Work

Conclusion o

# **Contract Monitoring**

- JSConTest generates assertions for checking argument and result contracts (pre- and postcondition)
- If during a run of g, the contract of f is violated, the assertions report this violation



Guided Random Testing

Monitoring

Case Study

Related Work

Conclusion o

# Huffman Encoding

- Take textbook algorithm
- Specify the behavior of the code
- Custom contract for Huffman Trees (13 loc)
- Annotations to functions (3 loc)

ndom Testing

duided Random Testing

Monitoring

Case Study

Related Work

Conclusion o

# Huffman Encoding

- After contract specification we found
  - a typing error in our code
  - a bug inside of the contract specification
- To check the effectivity of contract checking: We applied mutators to the Huffman Code
  - 88% of the mutated programs were rejected
  - 12% pass
    - manual inspection of the 12% shows that they behave correct with respect to the (type) specification
  - $\Rightarrow$  JSConTest detects type errors reliably



#### **Related Work**

- K. Claessen, J. Hughes, QuickCheck, ICFP 2000
- C. Csallner, Y. Smaragdakis, JCrasher, SPE 2004
- Guha, Matthews, Findler, Krishnamurthi, Relationally-Parametric Polymorphic Contracts, DLS 2007

Introduction Random Testing Guided Random Testing	Monitoring	Case Study	Related Work	Conclusion
	00	oo	o	●

#### Conclusion

- Contract language for JavaScript
- Random testing and contract monitoring
- Guided random testing to improve coverage
- Implemented in the JSConTest tool

Introduction	Random Testing	Guided Random Testing	Monitoring	Case Study oo	Related Work	Conclusion •
--------------	----------------	-----------------------	------------	------------------	--------------	-----------------

#### Conclusion

- Contract language for JavaScript
- Random testing and contract monitoring
- Guided random testing to improve coverage
- Implemented in the JSConTest tool

#### Future Work

- Minimization of counterexamples
- Transactions for JavaScript (Side Effects)
- Extension of contract language to describe side effects