Semantic conflict detection via dynamic analysis

Amanda Moraes
CIn – UFPE

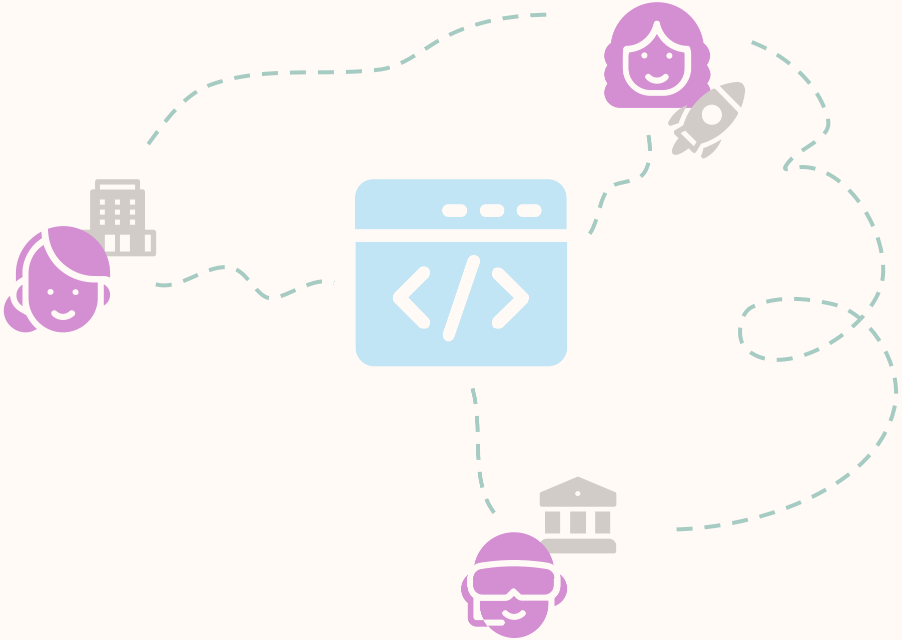
Paulo Borba
CIn – UFPE

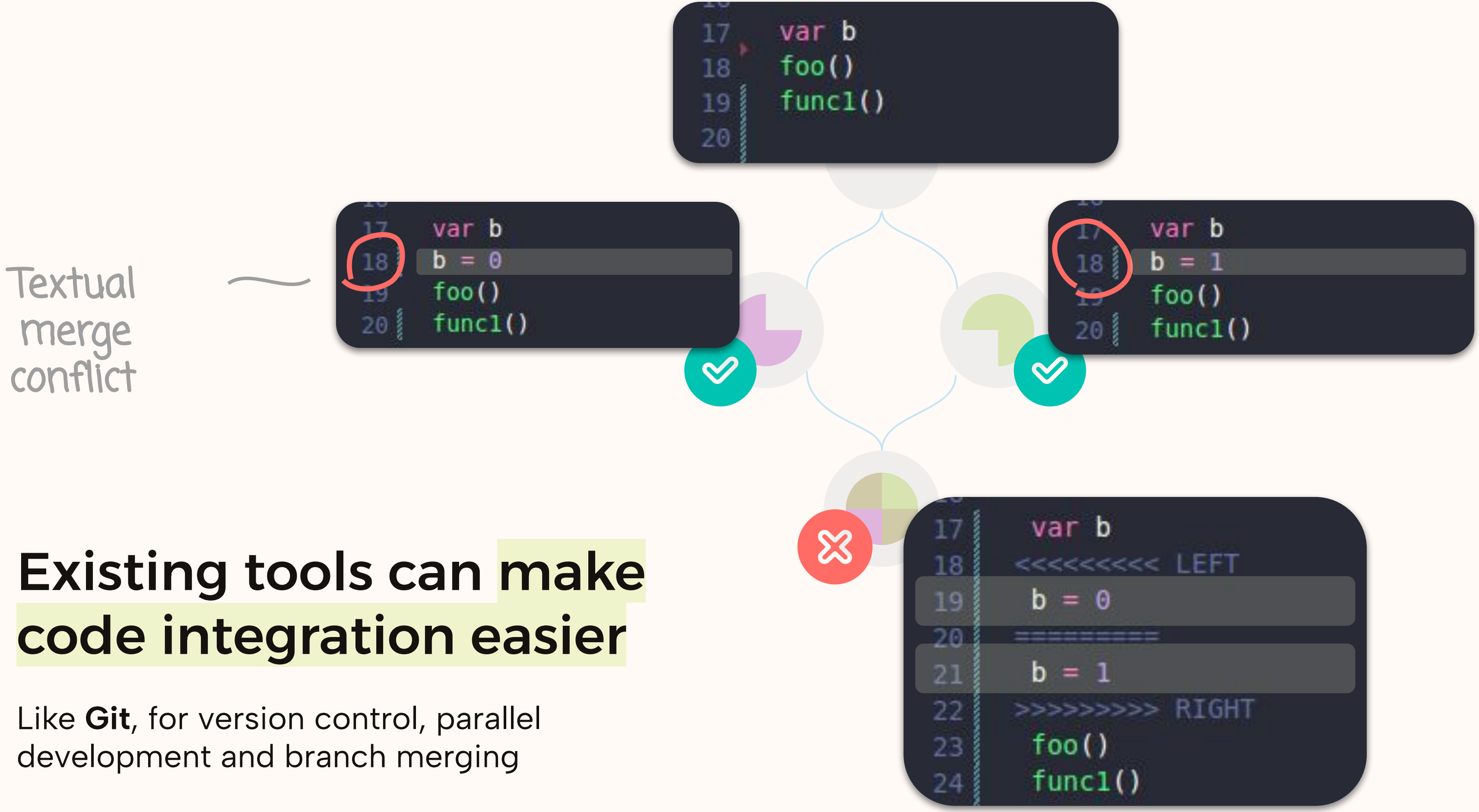
Léuson Da Silva
Polytechnique Montreal Canada

National Institute
of Science and Technology
in Software Engineering

Centro de
Informática
UFPE

The software development process is strongly collaborative

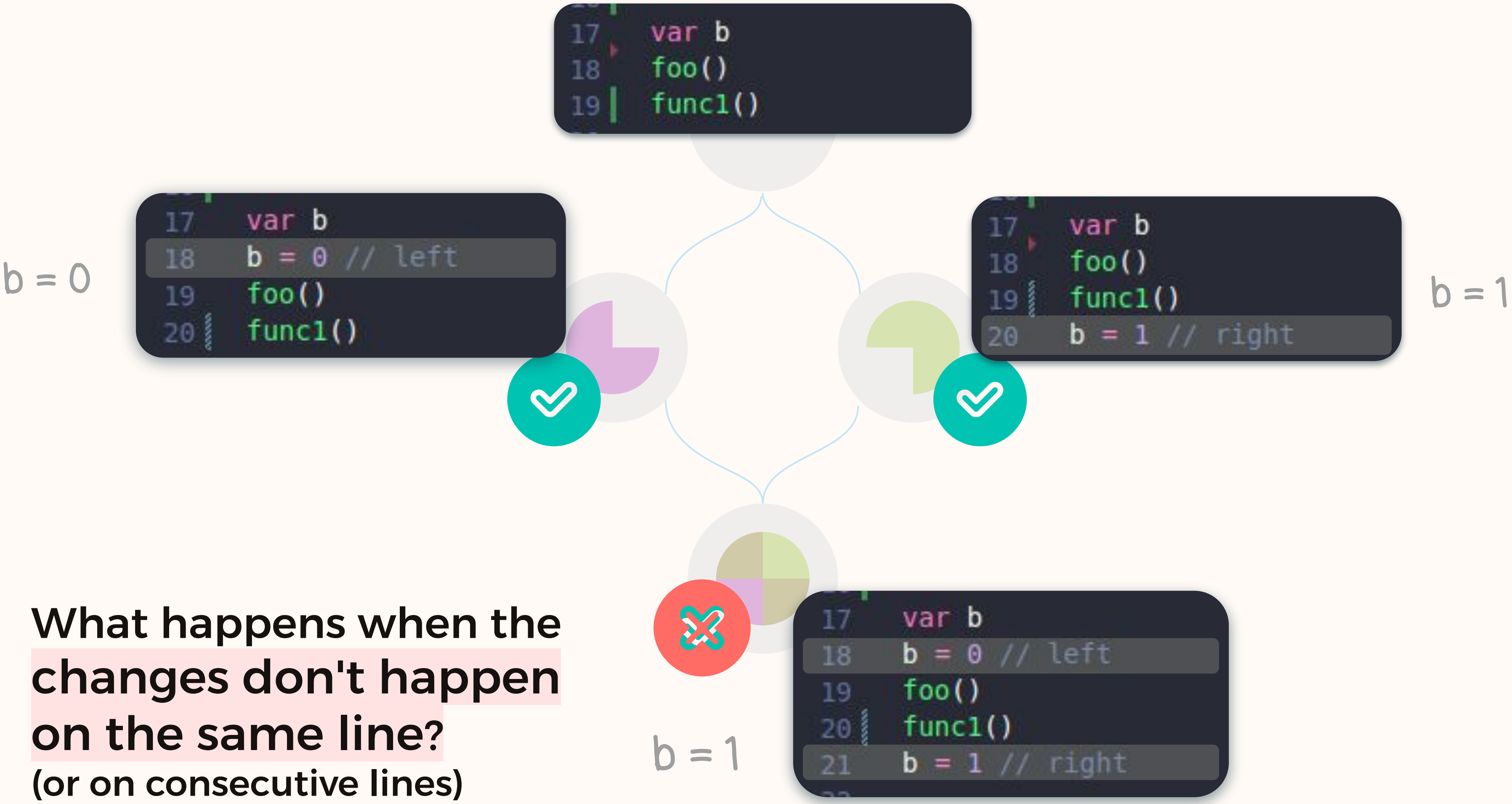




Existing tools can make code integration easier

Like **Git**, for version control, parallel development and branch merging

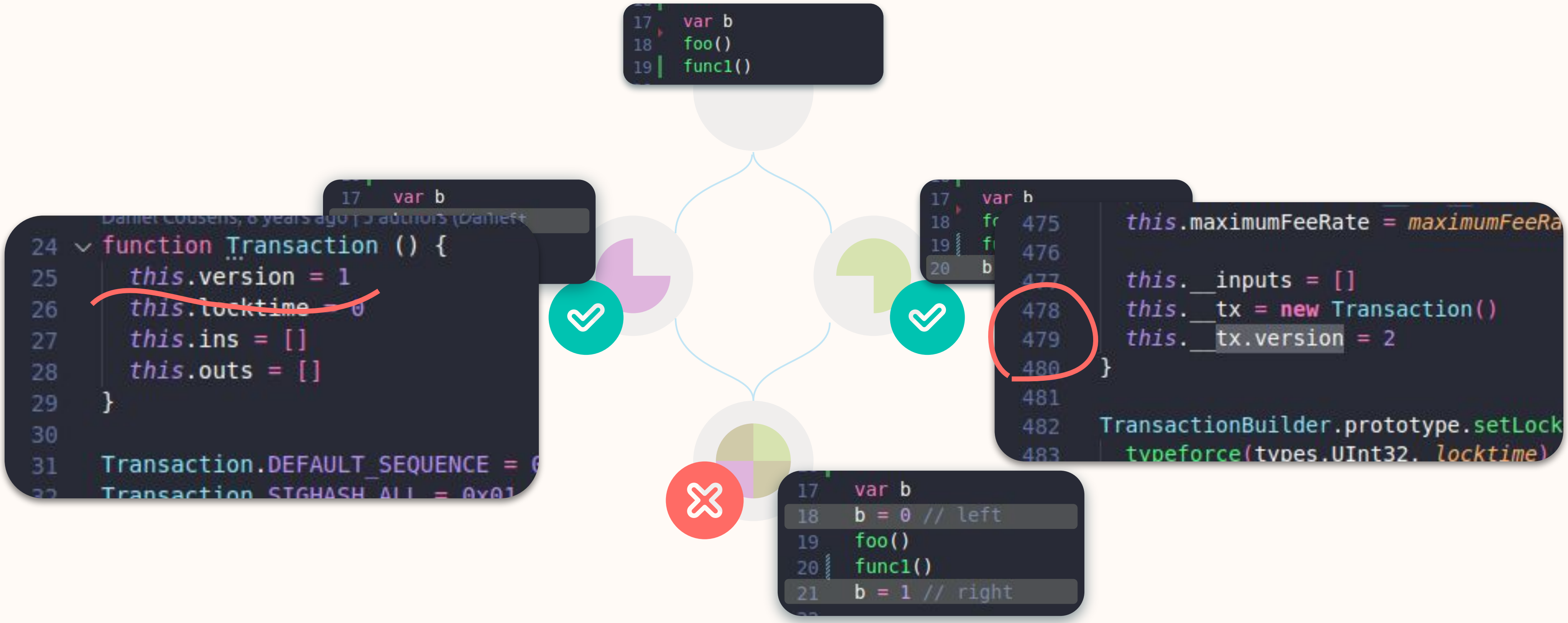
Centro de Informática UFPE



What happens when the changes don't happen on the same line?

(or on consecutive lines)

Centro de Informática UFPE



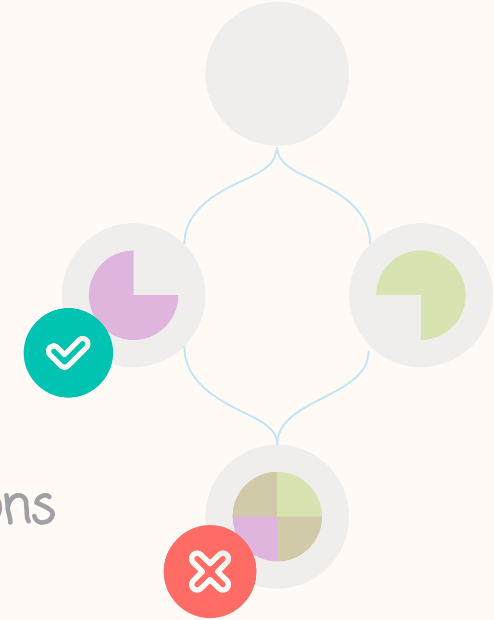
Centro de Informática
UFPE

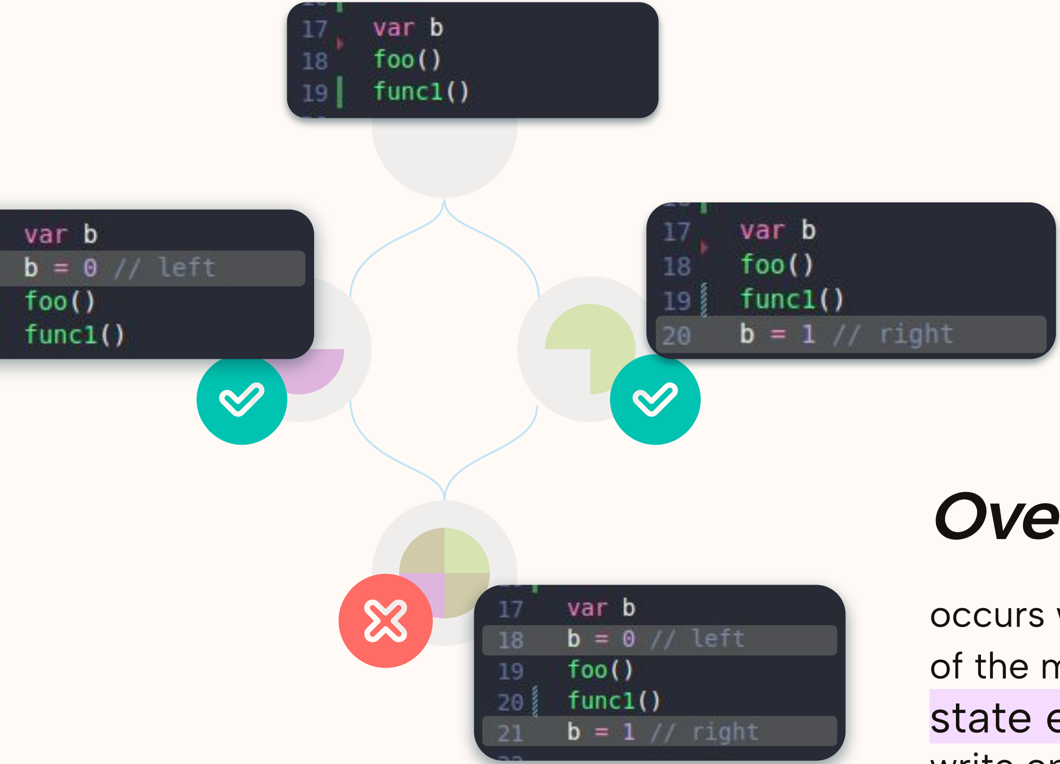
Semantic conflict (behavioral)

When the merge of contributions from 2 developers results in unplanned behavior when compared to the behavior of the versions that originated it

Behavior ~ specifications

Unplanned behavior ~ Interference

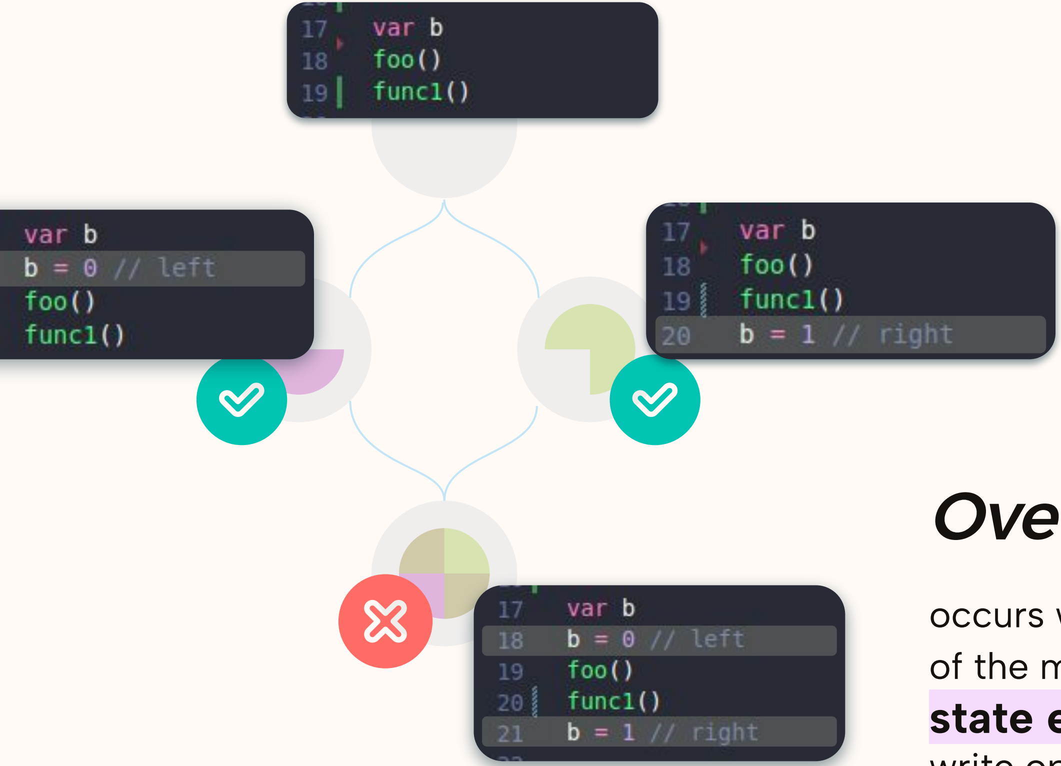




Overriding Assignment

occurs when additions or modifications to one of the merge branches involve writing to a state element that is also associated with a write operation due to the contribution of the other branch

Centro de Informática
UFPE



Overriding Assignment

occurs when additions or modifications to one of the merge branches **involve** writing to a **state element** that is also associated with a write operation due to the contribution of the other branch

[...] **involve** writing to a state element

```
class Text {
    constructor() {
        this.text = "";
        this.fixes = 0;
        this.comments = 0;
    }
    generateReport() {
        countDuplicatedWhitespaces(); // Left
        countComments();
        countDuplicatedWords(); // Right
    } ...
```

Centro de Informática UFPE

[...] involve writing to a **state element**

```
var arr = [0, 0, 0, 0, 0]
arr = [1, 1, 1, 1, 1, 1, 1]
console.log('hello')
arr[1] = 1
console.log('world!')
arr[2] = 2
console.log(arr)
```

Centro de Informática UFPE

Due to overriding assignments

Semantic conflict detection via dynamic analysis

Amanda Moraes
Centro de Informática
Universidade Federal de Pernambuco
Brazil
ascm@cin.ufpe.br

Paulo Borba
Centro de Informática
Universidade Federal de Pernambuco
Brazil
phmb@cin.ufpe.br

Léuson Da Silva
Polytechnique Montreal
Canadá
leuson-mario-pedro.da-silva@polymtl.ca

Abstract

During collaborative software development, a semantic conflict may occur when the individual behavior expected by different developers is no longer preserved after merging their branches. While potential semantic conflicts are not captured via textual merge tools, different approaches have already been proposed based on static analysis or automated test generation to verify behavioral changes given a merge scenario. However, these approaches share some limitations regarding scalability and reporting false positives and negatives. Trying to address these limitations, in this work, we assess the detection of conflicts by focusing on overriding assignments in JavaScript code through dynamic analysis

textual conflicts, behavioral semantic conflicts represent the behavioral differences between a program's pre- and post-merge versions, impacting how the software works when executed (either during test suites or even in production environments). The motivation to detect this kind of conflict is due to the divergence of behavior that occurs even when the integration is textually and syntactically valid [7], resulting in a scenario that incurs costs for software development when not noticed before execution.

The expected behavior intended by a software contributor is hard to assess and, in this context, can be approximated into program specifications about how it should work, while its unexpected change (potential semantic conflict) can be

Centro de Informática UFPE

Due to overriding assignments

Why and how ?

Semantic conflict detection via dynamic analysis

Amanda Moraes
Centro de Informática
Universidade Federal de Pernambuco
Brazil
ascm@cin.ufpe.br

Paulo Borba
Centro de Informática
Universidade Federal de Pernambuco
Brazil
phmb@cin.ufpe.br

Léuson Da Silva
Polytechnique Montreal
Canadá
leuson-mario-pedro.da-silva@polymtl.ca

Abstract

During collaborative software development, a semantic conflict may occur when the individual behavior expected by different developers is no longer preserved after merging their branches. While potential semantic conflicts are not captured via textual merge tools, different approaches have already been proposed based on static analysis or automated test generation to verify behavioral changes given a merge scenario. However, these approaches share some limitations regarding scalability and reporting false positives and negatives. Trying to address these limitations, in this work, we assess the detection of conflicts by focusing on overriding assignments in JavaScript code through dynamic analysis

textual conflicts, behavioral semantic conflicts represent the behavioral differences between a program's pre- and post-merge versions, impacting how the software works when executed (either during test suites or even in production environments). The motivation to detect this kind of conflict is due to the divergence of behavior that occurs even when the integration is textually and syntactically valid [7], resulting in a scenario that incurs costs for software development when not noticed before execution.

The expected behavior intended by a software contributor is hard to assess and, in this context, can be approximated into program specifications about how it should work, while its unexpected change (potential semantic conflict) can be

Centro de Informática
UFPE

Related work

Static Analysis

Conservative; Potential drop in accuracy

Unit Test Generation

Run on each version of merge; potential increase of false negatives

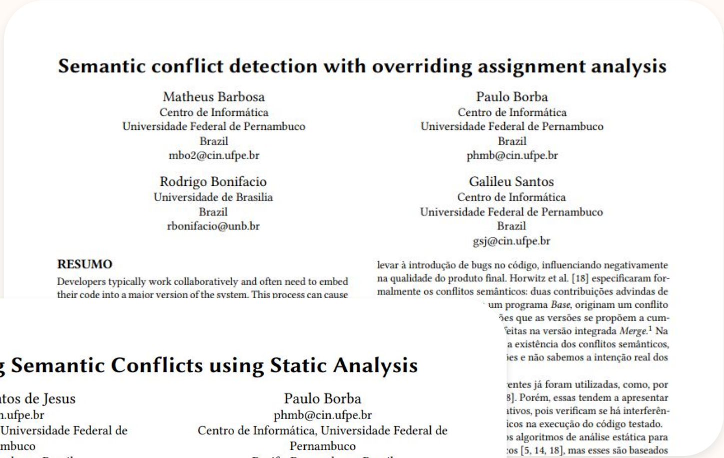
Related work

Static Analysis

Conservative; Potential drop in accuracy

Unit Test Generation

Run on each version of merge; potential increase of false negatives



Semantic conflict detection with overriding assignment analysis

Matheus Barbosa
Centro de Informática
Universidade Federal de Pernambuco
Brazil
mbo2@cin.ufpe.br

Paulo Borba
Centro de Informática
Universidade Federal de Pernambuco
Brazil
phmb@cin.ufpe.br

Rodrigo Bonifacio
Universidade de Brasilia
Brazil
rbonifacio@unb.br

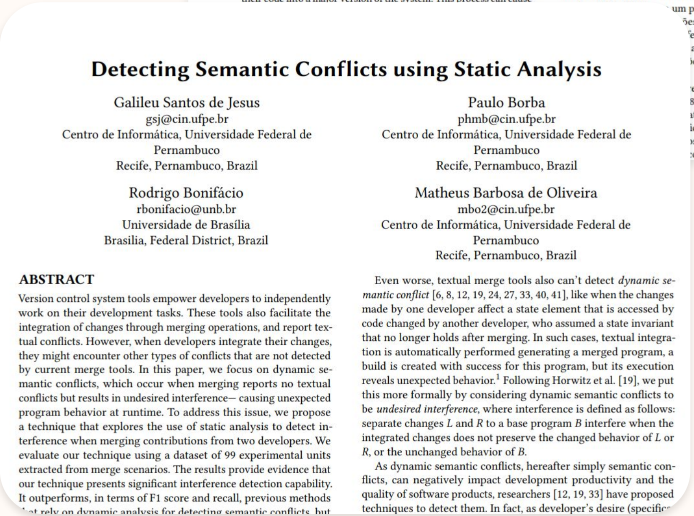
Galileu Santos
Centro de Informática
Universidade Federal de Pernambuco
Brazil
gsj@cin.ufpe.br

RESUMO

Developers typically work collaboratively and often need to embed their code into a major version of the system. This process can cause

levar à introdução de bugs no código, influenciando negativamente na qualidade do produto final. Horwitz et al. [18] especificaram formalmente os conflitos semânticos: duas contribuições advindas de … um programa *Base*, originam um conflito … ões que as versões se propõem a cum- … feitas na versão integrada *Merge*.[1] Na … a existência dos conflitos semânticos, … ões e não sabemos a intenção real dos

… entes já foram utilizadas, como, por … 8]. Porém, essas tendem a apresentar … tivos, pois verificam se há interferên- … icos na execução do código testado. … s algoritmos de análise estática para … os [5, 14, 18], mas esses são baseados



Detecting Semantic Conflicts using Static Analysis

Galileu Santos de Jesus
gsj@cin.ufpe.br
Centro de Informática, Universidade Federal de Pernambuco
Recife, Pernambuco, Brazil

Paulo Borba
phmb@cin.ufpe.br
Centro de Informática, Universidade Federal de Pernambuco
Recife, Pernambuco, Brazil

Rodrigo Bonifácio
rbonifacio@unb.br
Universidade de Brasília
Brasilia, Federal District, Brazil

Matheus Barbosa de Oliveira
mbo2@cin.ufpe.br
Centro de Informática, Universidade Federal de Pernambuco
Recife, Pernambuco, Brazil

ABSTRACT

Version control system tools empower developers to independently work on their development tasks. These tools also facilitate the integration of changes through merging operations, and report textual conflicts. However, when developers integrate their changes, they might encounter other types of conflicts that are not detected by current merge tools. In this paper, we focus on dynamic semantic conflicts, which occur when merging reports no textual conflicts but results in undesired interference— causing unexpected program behavior at runtime. To address this issue, we propose a technique that explores the use of static analysis to detect interference when merging contributions from two developers. We evaluate our technique using a dataset of 99 experimental units extracted from merge scenarios. The results provide evidence that our technique presents significant interference detection capability. It outperforms, in terms of F1 score and recall, previous methods

Even worse, textual merge tools also can't detect *dynamic semantic conflict* [6, 8, 12, 19, 24, 27, 33, 40, 41], like when the changes made by one developer affect a state element that is accessed by code changed by another developer, who assumed a state invariant that no longer holds after merging. In such cases, textual integration is automatically performed generating a merged program, a build is created with success for this program, but its execution reveals unexpected behavior.[1] Following Horwitz et al. [19], we put this more formally by considering dynamic semantic conflicts to be *undesired interference*, where interference is defined as follows: separate changes L and R to a base program B interfere when the integrated changes does not preserve the changed behavior of L or R, or the unchanged behavior of B.

As dynamic semantic conflicts, hereafter simply semantic conflicts, can negatively impact development productivity and the quality of software products, researchers [12, 19, 33] have proposed techniques to detect them. In fact, as developer's desire (specific

Centro de Informática
UFPE

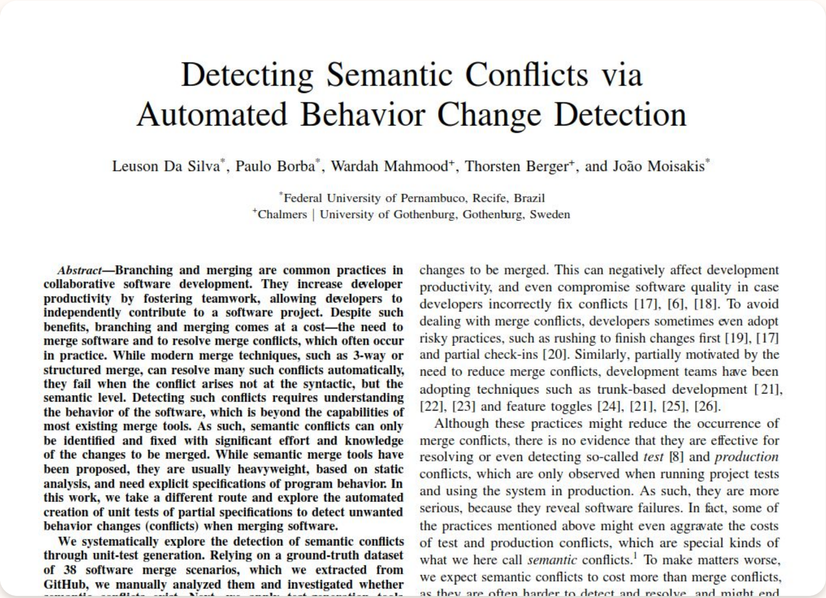
Related work

Static Analysis

Conservative; Potential drop in accuracy

Unit Test Generation

Run on each version of merge; potential increase of false negatives



Detecting Semantic Conflicts via Automated Behavior Change Detection

Leuson Da Silva*, Paulo Borba*, Wardah Mahmood+, Thorsten Berger+, and João Moisakis*

*Federal University of Pernambuco, Recife, Brazil
+Chalmers | University of Gothenburg, Gothenburg, Sweden

Abstract—Branching and merging are common practices in collaborative software development. They increase developer productivity by fostering teamwork, allowing developers to independently contribute to a software project. Despite such benefits, branching and merging comes at a cost—the need to merge software and to resolve merge conflicts, which often occur in practice. While modern merge techniques, such as 3-way or structured merge, can resolve many such conflicts automatically, they fail when the conflict arises not at the syntactic, but the semantic level. Detecting such conflicts requires understanding the behavior of the software, which is beyond the capabilities of most existing merge tools. As such, semantic conflicts can only be identified and fixed with significant effort and knowledge of the changes to be merged. While semantic merge tools have been proposed, they are usually heavyweight, based on static analysis, and need explicit specifications of program behavior. In this work, we take a different route and explore the automated creation of unit tests of partial specifications to detect unwanted behavior changes (conflicts) when merging software.

We systematically explore the detection of semantic conflicts through unit-test generation. Relying on a ground-truth dataset of 38 software merge scenarios, which we extracted from GitHub, we manually analyzed them and investigated whether

changes to be merged. This can negatively affect development productivity, and even compromise software quality in case developers incorrectly fix conflicts [17], [6], [18]. To avoid dealing with merge conflicts, developers sometimes even adopt risky practices, such as rushing to finish changes first [19], [17] and partial check-ins [20]. Similarly, partially motivated by the need to reduce merge conflicts, development teams have been adopting techniques such as trunk-based development [21], [22], [23] and feature toggles [24], [21], [25], [26].

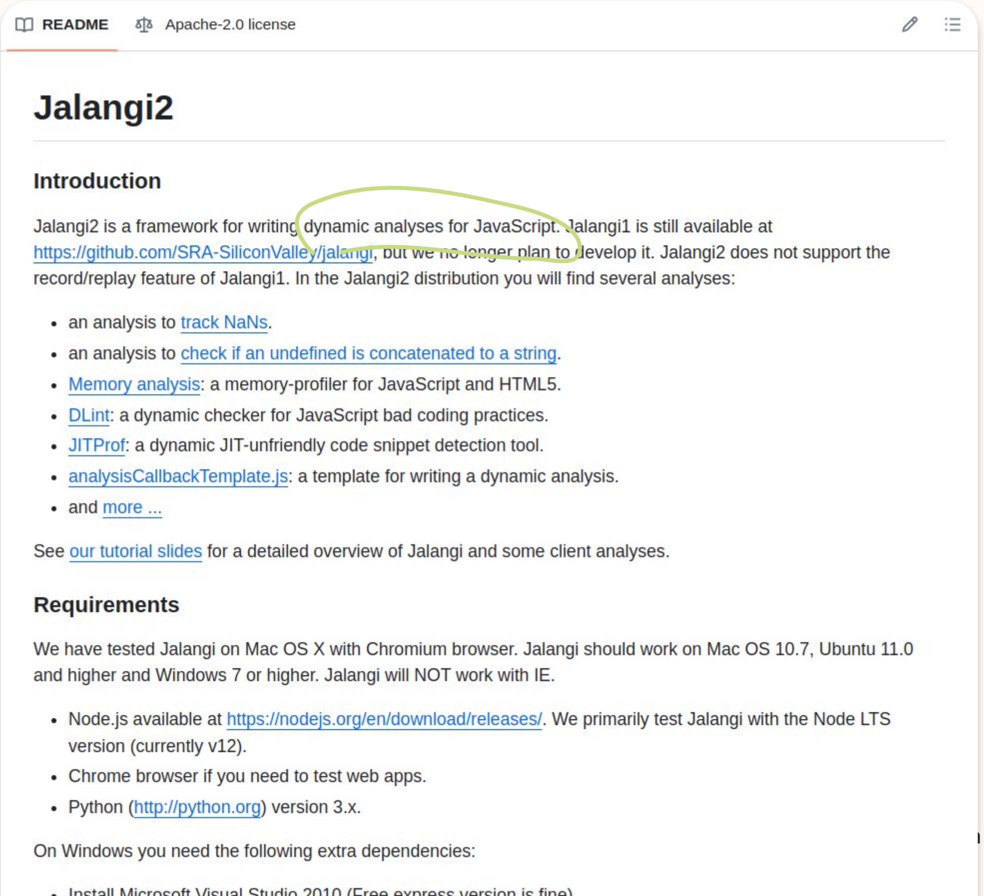
Although these practices might reduce the occurrence of merge conflicts, there is no evidence that they are effective for resolving or even detecting so-called *test* [8] and *production* conflicts, which are only observed when running project tests and using the system in production. As such, they are more serious, because they reveal software failures. In fact, some of the practices mentioned above might even aggravate the costs of test and production conflicts, which are special kinds of what we here call *semantic* conflicts.[1] To make matters worse, we expect semantic conflicts to cost more than merge conflicts,

Centro de Informática UFPE

Available resources

Dynamic Analysis

- Occurs at runtime
- Does not demand assertions about the program
- It is only executed on the post-merge code



README   Apache-2.0 license

Jalangi2

Introduction

Jalangi2 is a framework for writing dynamic analyses for JavaScript. Jalangi1 is still available at https://github.com/SRA-SiliconValley/jalangi, but we no longer plan to develop it. Jalangi2 does not support the record/replay feature of Jalangi1. In the Jalangi2 distribution you will find several analyses:

- an analysis to track NaNs.
- an analysis to check if an undefined is concatenated to a string.
- Memory analysis: a memory-profiler for JavaScript and HTML5.
- DLint: a dynamic checker for JavaScript bad coding practices.
- JITProf: a dynamic JIT-unfriendly code snippet detection tool.
- analysisCallbackTemplate.js: a template for writing a dynamic analysis.
- and more ...

See our tutorial slides for a detailed overview of Jalangi and some client analyses.

Requirements

We have tested Jalangi on Mac OS X with Chromium browser. Jalangi should work on Mac OS 10.7, Ubuntu 11.0 and higher and Windows 7 or higher. Jalangi will NOT work with IE.

- Node.js available at https://nodejs.org/en/download/releases/. We primarily test Jalangi with the Node LTS version (currently v12).
- Chrome browser if you need to test web apps.
- Python (http://python.org) version 3.x.

On Windows you need the following extra dependencies:

- Install Microsoft Visual Studio 2010 (Free express version is fine).

Jalangi2 Analysis

Overriding Assignment Analysis

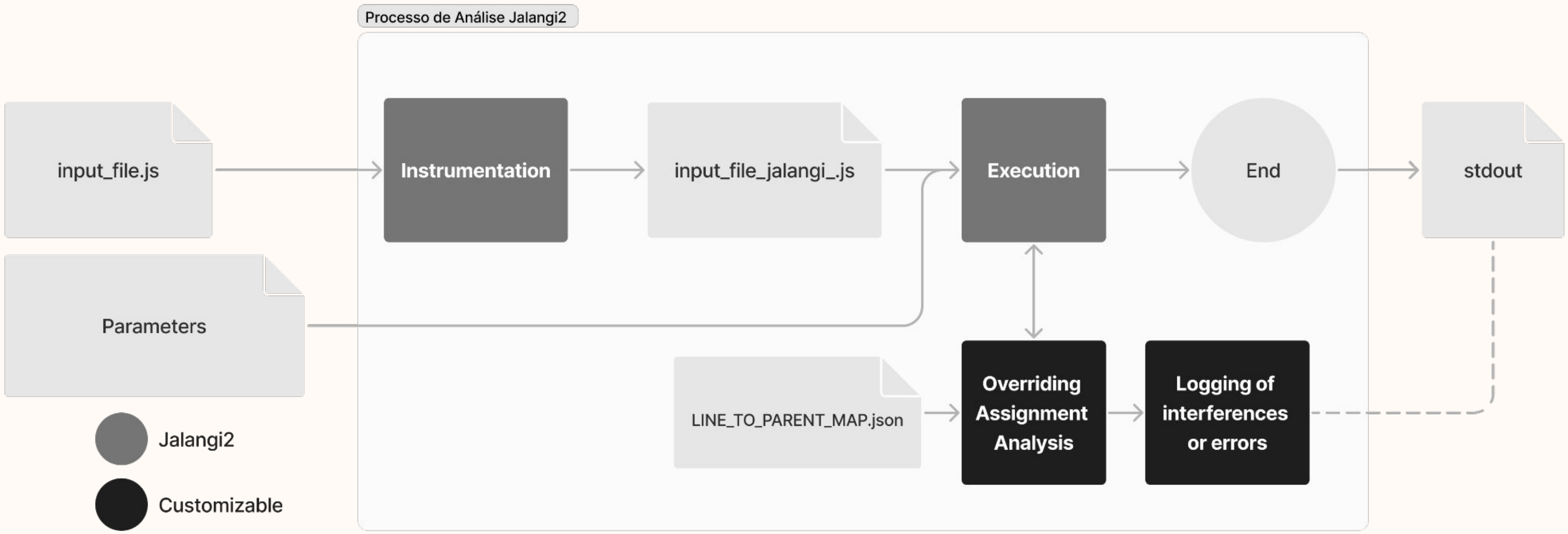
LINE_TO_PARENT_MAP.json

var x = 23

```
(function (sandbox) {
    function OverridingAssignmentAnalysis() {

            this.write = function (iid, name, val, lhs, isGlobal, isScriptLoc
                const frameId = sandbox.smemory.getIDFromShadowObjectOrFrame(
                const location = J$.iidToLocation(J$.sid, iid)
                const branch = LocationToBranchService.getInstance().mapLocat

                overridingAssignmentService.assignmentHandler(new Assignment(
                assignmentService.assignmentHandler(new Assignment(frameId, n

                return {result: val}
            };

```

Custom OA Analysis

Centro de Informática
UFPE



Centro de Informática UFPE

```javascript
function OverrideAssignmentAnalysis() {

    const overrideAssignmentController = new
    const assignmentController = new Assignme

    this.invokeFunPre = function (iid, f, bas
    };

    this.invokeFun = function (iid, f, base,
    };

    this.putFieldPre = function (iid, base, o
    };

    this.write = function (iid, name, val, lhs, isGlobal, isScriptLocal) {⋯
    };

    this.endExecution = function () {⋯
    };
}
```

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨ ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨ ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

Right assignments

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática
UFPE

```
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

Right assignments

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script *s*
Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

Right assignments

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s
Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática
UFPE

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

Right assignments

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

Right assignments

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

for $event \in s$ **do**
  if $isWrite(event) \lor isPutFieldPre(event)$ **then**
    if $isFromSomeBranch(event) \lor$
     $\neg isFunctionCallStackEmpty()$ **then**
      $addAssignmentToCurrentBranch(event)$
      if *hasAssignmentFromOtherBranch(event)* **then**
        $updateInterferences(event)$
        $removeAssignmentFromOtherBranch(event)$
    else
      $removeAssignmentFromAllBranches(event)$
  if *isInvokeFunPre(event)* **then**
    if $isFromSomeBranch(event) \lor$
     $\neg isFunctionCallStackEmpty()$ **then**
      if *isArrayInplaceMethod(event)* **then**
        $handleAssignedIndices(event)$
      else
        $functionCallStack.push(event)$
  if *isInvokeFun(event)* **then**
    $functionCallStack.pop()$
  if *isEndExecution(event)* **then**
    $logResults()$

Centro de Informática UFPE

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

Right assignments

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s
Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

b

Right assignments

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

for $event \in s$ **do**
  if $isWrite(event) \lor isPutFieldPre(event)$ **then**
    if $isFromSomeBranch(event) \lor \neg isFunctionCallStackEmpty()$ **then**
      $addAssignmentToCurrentBranch(event)$
      if $hasAssignmentFromOtherBranch(event)$ **then**
        $updateInterferences(event)$
        $removeAssignmentFromOtherBranch(event)$
    else
      $removeAssignmentFromAllBranches(event)$
  if $isInvokeFunPre(event)$ **then**
    if $isFromSomeBranch(event) \lor \neg isFunctionCallStackEmpty()$ **then**
      if $isArrayInplaceMethod(event)$ **then**
        $handleAssignedIndices(event)$
      else
        $functionCallStack.push(event)$
  if $isInvokeFun(event)$ **then**
    $functionCallStack.pop()$
  if $isEndExecution(event)$ **then**
    $logResults()$

Centro de Informática
UFPE

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

b

Right assignments

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática
UFPE

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

b

Right assignments

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática
UFPE

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

b

Right assignments

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática
UFPE

```
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

b

Right assignments

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

for $event \in s$ **do**
    if $isWrite(event) \lor isPutFieldPre(event)$ **then**
        if $isFromSomeBranch(event) \lor \neg isFunctionCallStackEmpty()$ **then**
            $addAssignmentToCurrentBranch(event)$
            if $hasAssignmentFromOtherBranch(event)$ **then**
                $updateInterferences(event)$
                $removeAssignmentFromOtherBranch(event)$
        else
            $removeAssignmentFromAllBranches(event)$
    if $isInvokeFunPre(event)$ **then**
        if $isFromSomeBranch(event) \lor \neg isFunctionCallStackEmpty()$ **then**
            if $isArrayInplaceMethod(event)$ **then**
                $handleAssignedIndices(event)$
            else
                $functionCallStack.push(event)$
    if $isInvokeFun(event)$ **then**
        $functionCallStack.pop()$
    if $isEndExecution(event)$ **then**
        $logResults()$

Centro de Informática
UFPE

```
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

b

Right assignments

Function call stack

func1

Algorithm 1: Overriding Assignment Detection

```
Data: A script s
Result: A list of overriding assignment interferences
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática UFPE

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

b

Right assignments

Function call stack

func1

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

for $event \in s$ **do**
    if $isWrite(event) \lor isPutFieldPre(event)$ **then**
        if $isFromSomeBranch(event) \lor$ $\neg isFunctionCallStackEmpty()$ **then**
            $addAssignmentToCurrentBranch(event)$
            if $hasAssignmentFromOtherBranch(event)$ **then**
                $updateInterferences(event)$
                $removeAssignmentFromOtherBranch(event)$
        else
            $removeAssignmentFromAllBranches(event)$
    if $isInvokeFunPre(event)$ **then**
        if $isFromSomeBranch(event) \lor$ $\neg isFunctionCallStackEmpty()$ **then**
            if $isArrayInplaceMethod(event)$ **then**
                $handleAssignedIndices(event)$
            else
                $functionCallStack.push(event)$
    if $isInvokeFun(event)$ **then**
        $functionCallStack.pop()$
    if $isEndExecution(event)$ **then**
        $logResults()$

```
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

b

Right assignments

Function call stack

func1

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática
UFPE

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

b

Right assignments

b

Function call stack

func1

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática
UFPE

```javascript
function func1() {
    b = 1
}

var b
b = 0 // LEFT
foo()
func1() // RIGHT
```

Interferences

Left assignments

b

Right assignments

b

Function call stack

func1

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática UFPE

```
49  function func1() {
50      b = 1
51  }
52
53  var b
54  b = 0 // LEFT
55  foo()
56  func1() // RIGHT
```

Interferences

b (L:54; R:56,50)

Left assignments

b

Right assignments

b

Function call stack

func1

Algorithm 1: Overriding Assignment Detection

```
Data: A script s
Result: A list of overriding assignment interferences
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

```
49  function func1() {
50      b = 1
51  }
52
53  var b
54  b = 0 // LEFT
55  foo()
56  func1() // RIGHT
```

Interferences

b (L:54; R:56,50)

Left assignments

Right assignments

b

Function call stack

func1

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática
UFPE

```
49  function func1() {
50      b = 1
51  }
52
53  var b
54  b = 0 // LEFT
55  foo()
56  func1() // RIGHT
```

Interferences

b (L:54; R:56,50)

Left assignments

Right assignments

b

Function call stack

func1

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática
UFPE

```
49  function func1() {
50      b = 1
51  }
52
53  var b
54  b = 0 // LEFT
55  foo()
56  func1() // RIGHT
```

Interferences

b (L:54; R:56,50)

Left assignments

Right assignments

b

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s
Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

```
49  function func1() {
50      b = 1
51  }
52
53  var b
54  b = 0 // LEFT
55  foo()
56  func1() // RIGHT
```

Interferences

b (L:54; R:56,50)

Left assignments

Right assignments

b

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s
Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática
UFPE

```
49  function func1() {
50      b = 1
51  }
52
53  var b
54  b = 0 // LEFT
55  foo()
56  func1() // RIGHT
```

Interferences

b (L:54; R:56,50)

Left assignments

Right assignments

b

Function call stack

Algorithm 1: Overriding Assignment Detection

Data: A script s

Result: A list of overriding assignment interferences

```
for event ∈ s do
    if isWrite(event) ∨ isPutFieldPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            addAssignmentToCurrentBranch(event)
            if hasAssignmentFromOtherBranch(event) then
                updateInterferences(event)
                removeAssignmentFromOtherBranch(event)
        else
            removeAssignmentFromAllBranches(event)
    if isInvokeFunPre(event) then
        if isFromSomeBranch(event) ∨
          ¬isFunctionCallStackEmpty() then
            if isArrayInplaceMethod(event) then
                handleAssignedIndices(event)
            else
                functionCallStack.push(event)
    if isInvokeFun(event) then
        functionCallStack.pop()
    if isEndExecution(event) then
        logResults()
```

Centro de Informática UFPE

```
{
  "type": "OVERRIDING_ASSIGNMENT_CONFLICT",
  "label": "Overriding Assignment Conflict",
  "body": {
    "description": "Interference detected on 3_b:\nBranch L at line 54 (/index.js:54:5:54:6)\nBranch R at line 50 (/index.js:50:9:50:10) with interprocedural stack: [{\"id\":1241,\"name\":\"func1\",\"location\":\"(/index.js:56:1:56:8)\",\"branch\":\"R\",\"beforeInvoke\":true}]",
    "interference": {
      "previousAssignment": {
        "id": 3,
        "name": "b",
        "location": "(/index.js:54:5:54:6)",
        "branch": "L",
        "isObject": false
      },
      "currentAssignment": {
        "id": 3,
        "name": "b",
        "location": "(/index.js:50:9:50:10)",
        "branch": "R",
        "isObject": false,
        "functionCallStack": [
          {
            "id": 1241,
            "name": "func1",
            "location": "(/index.js:56:1:56:8)",
            "branch": "R",
            "beforeInvoke": true
          }
```


```
49  function func1() {
50      b = 1
51  }
52
53  var b
54  b = 0 // LEFT
55  foo()
56  func1() // RIGHT
```

Evaluation & Results

Correctness
With test cases

Capacity
With third party merges

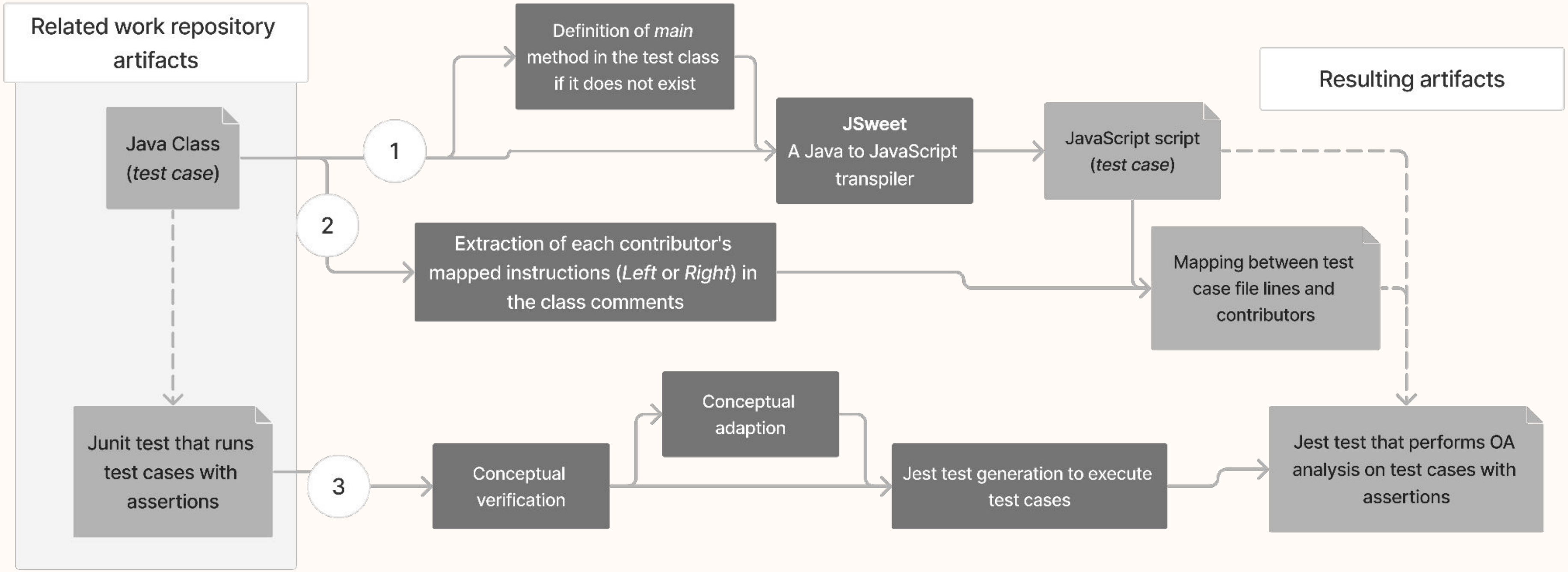
Overhead
On execution time

Evaluation & Results

Correctness
With test cases

Capacity
With third party merges

Overhead
On execution time



Centro de Informática UFPE

IfBranchConflictSample

```java
package br.unb.cic.analysis.samples.ioa;

// Conflict: [left, main():10] --> [right, foo():16]
public class IfBranchConflictSample {
    private int x;

    public static void main() {
        IfBranchConflictSample m = new IfBranchConflictSample();

        m.x = 0; // LEFT
        m.foo(); // RIGHT
    }

    private void foo() {
        if (x >= 0) {
            x = 1;
        } else {
            int a = 0;
            // System.out.println(x)
        }
    }
}
```

IfBranchConflictSample

```javascript
/* Generated from Java with JSweet 3.0.0 - http://www.jsweet.org */
var br;
(function (br) {
    var unb;
    (function (unb) {
        var cic;
        (function (cic) {
            var analysis;
            (function (analysis) {
                var samples;
                (function (samples) {
                    var ioa;
                    (function (ioa) {
                        var IfBranchConflictSample = /** @class */ (function () {
                            function IfBranchConflictSample() {
                                if (this.x === undefined) {
                                    this.x = 0;
                                }
                            }
                            IfBranchConflictSample.main = function () {
                                var m = new IfBranchConflictSample();
                                m.x = 0;
                                m.foo();
                            };
                            /*private*/ IfBranchConflictSample.prototype.foo = function () {
                                if (this.x >= 0) {
                                    this.x = 1;
                                }
                                else {
                                    var a = 0;
                                }
                            };
                            return IfBranchConflictSample;
                        }());
                        ioa.IfBranchConflictSample = IfBranchConflictSample;
                        IfBranchConflictSample["__class"] =
"br.unb.cic.analysis.samples.ioa.IfBranchConflictSample";
                    })(ioa = samples.ioa || (samples.ioa = {}));
                })(samples = analysis.samples || (analysis.samples = {}));
            })(analysis = cic.analysis || (cic.analysis = {}));
        })(cic = unb.cic || (unb.cic = {}));
    })(unb = br.unb || (br.unb = {}));
})(br || (br = {}));
br.unb.cic.analysis.samples.ioa.IfBranchConflictSample.main();
```

Centro de Informática
UFPE

IfBranchConflictSample

```java
package br.unb.cic.analysis.samples.ioa;

// Conflict: [left, main():10] --> [right, foo():16]
public class IfBranchConflictSample {
    private int x;

    public static void main() {
        IfBranchConflictSample m = new IfBranchConflictSample();

        m.x = 0; // LEFT
        m.foo(); // RIGHT
    }

    private void foo() {
        if (x >= 0) {
            x = 1;
        } else {
            int a = 0;
            // System.out.println(x)
        }
    }
}
```

```javascript
}
IfBranchConflictSample.main = function () {
    var m = new IfBranchConflictSample();
    m.x = 0;
    m.foo();
};
/*private*/ IfBranchConflictSample.prototype.foo = function () {
    if (this.x >= 0) {
        this.x = 1;
    }
    else {
        var a = 0;
    }
};
```

```javascript
      })(cic = unb.cic || (unb.cic = {}));
    })(unb = br.unb || (br.unb = {}));
})(br || (br = {}));
br.unb.cic.analysis.samples.ioa.IfBranchConflictSample.main();
```

Centro de Informática
UFPE

Test Results

In translated validation scenarios

Transpilation Result	Amount of tests	Success Rate
TRANSPILATION_DIVERGENCE	2	–
CONCEPTUALLY_ADAPTED	14	100%
SUCCESS	55	100%

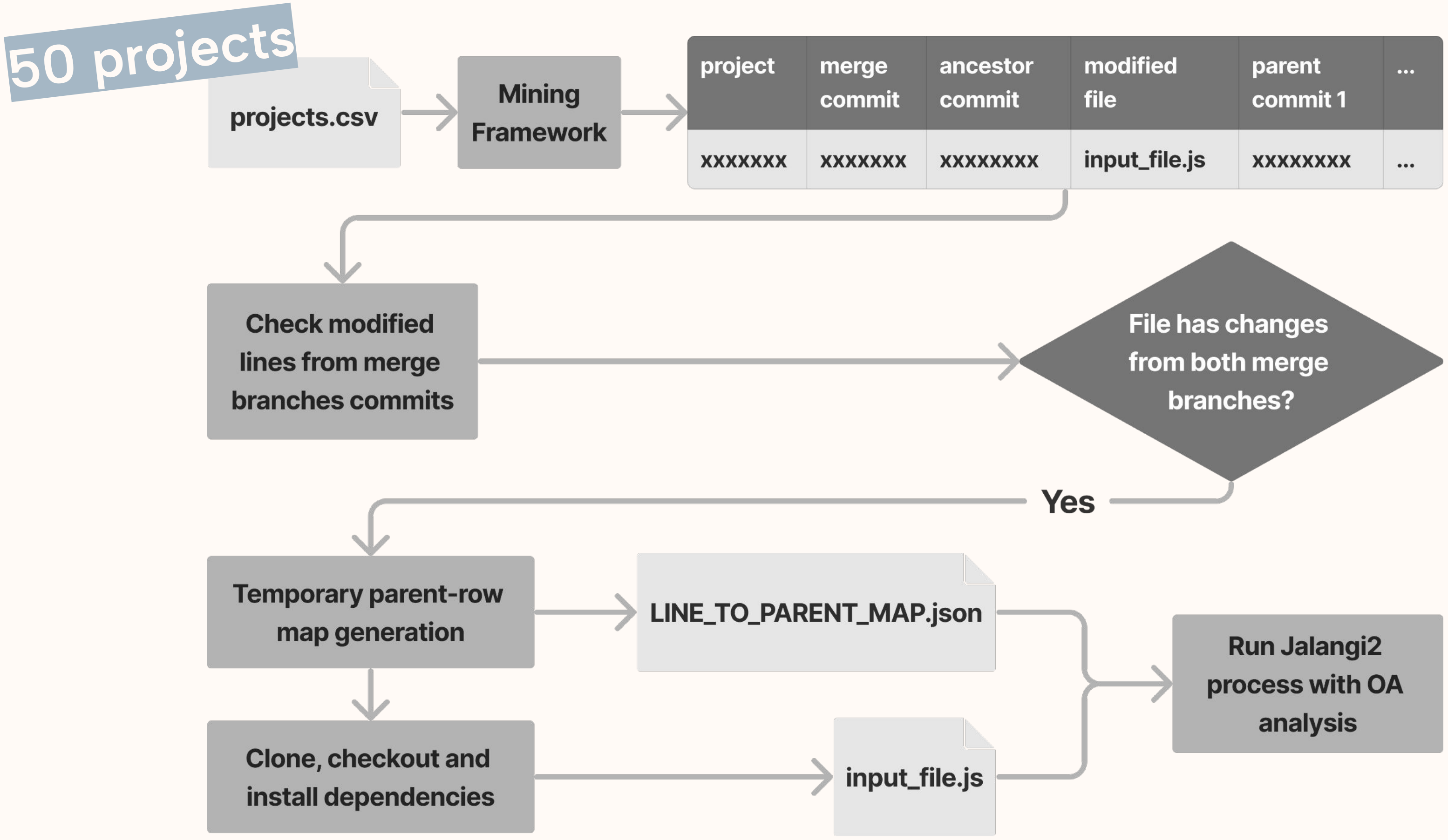
Evaluation & Results

Correctness
With test cases

Capacity
With third party merges

Overhead
On execution time

Centro de Informática
UFPE



Centro de Informática
UFPE

Results

In open source merge scenarios

452 executable scenarios

159 ~35% successful executions

1 overriding assignment

158 with no events

293 ~65% execution errors

Centro de Informática
UFPE

159

successful executions

1 overriding assignment

~1%

158 with no events

~99%

Centro de Informática UFPE

159

successful executions

1 overriding assignment

~1%

158 with no events

~99%

Centro de Informática
UFPE

```
function aumentarContador() {
    ...
}

function iniciarProcesso() {
    ...
}

module.exports = {
    aumentarContador, iniciarProcesso
}
```

Why? ~99%

With no OA

Centro de Informática
UFPE

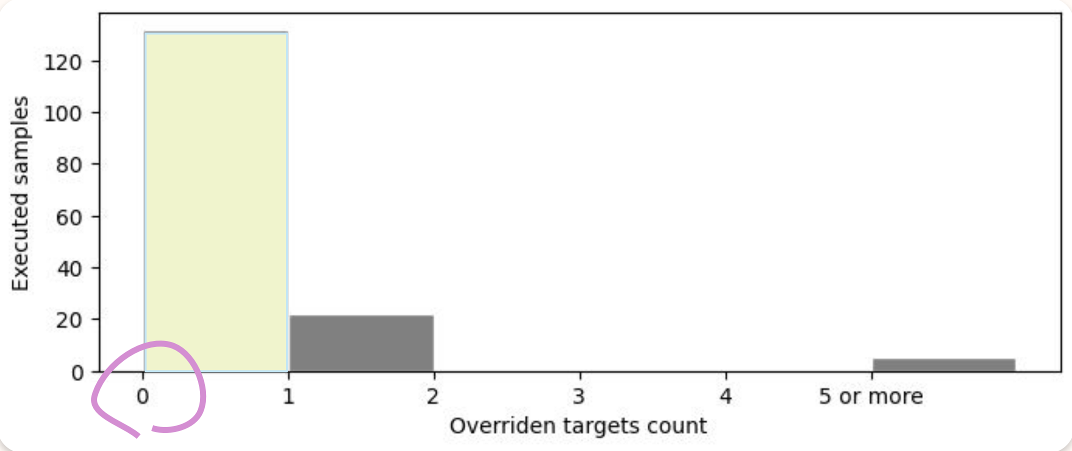
Why? ~99%

With no OA

```
function aumentarContador() {
    ...
}

function iniciarProcesso() {
    ...
}

module.exports = {
    aumentarContador, iniciarProcesso
}
```

```
iniciarProcesso()

aumentarContador()
```

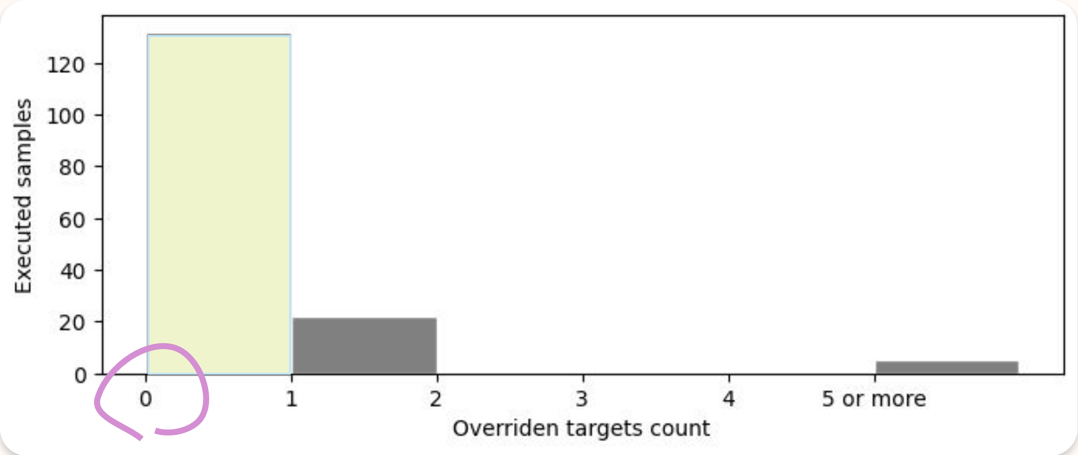
Centro de Informática
UFPE



Why? ~99%
With no OA

With an additional overriding assignment analysis, this time disregarding branches or whether it was modified,

83% of **159** scenarios showed no override of any state element



Why? ~99%

With no OA

With an additional overriding assignment analysis, this time disregarding branches or whether it was modified,

83% of **159** scenarios showed no override of any state element

83% true negatives

Centro de Informática
UFPE

159

successful executions

1 overriding assignment

~1%

158 with no events

~99%

Centro de Informática
UFPE

```
Interference detected on 5_inject:
Branch L at (.../Nightmare/lib/actions.js:100:1:120:2)
Branch R at (.../Nightmare/lib/actions.js:226:1:240:2)
```

100% de positivos verdadeiros

Apenas 1 positivo, que é verdadeiro

```
100 exports.inject = function(type, file, done){
101   debug('.inject()-ing a file');
102   var startTag, endTag;
103   if ( type !== "js" && type !== "css" ){
104     debug('unsupported file type in .inject()');
105     done();
106   }
107   if (type === "js"){
108     startTag = "<script>";
109     endTag = "</script>";
110   } else if (type === "css"){
111     startTag = "<style>";
112     endTag = "</style>";
113   }
114   var self = this;
115   this.page.getContent( function(pageContent){
116     var injectedContents = fs.readFileSync(file);
117     self.page.setContent(pageContent + startTag + inject
118   });
119
120 }       John Titus, 9 years ago • Added injectJs action,
```

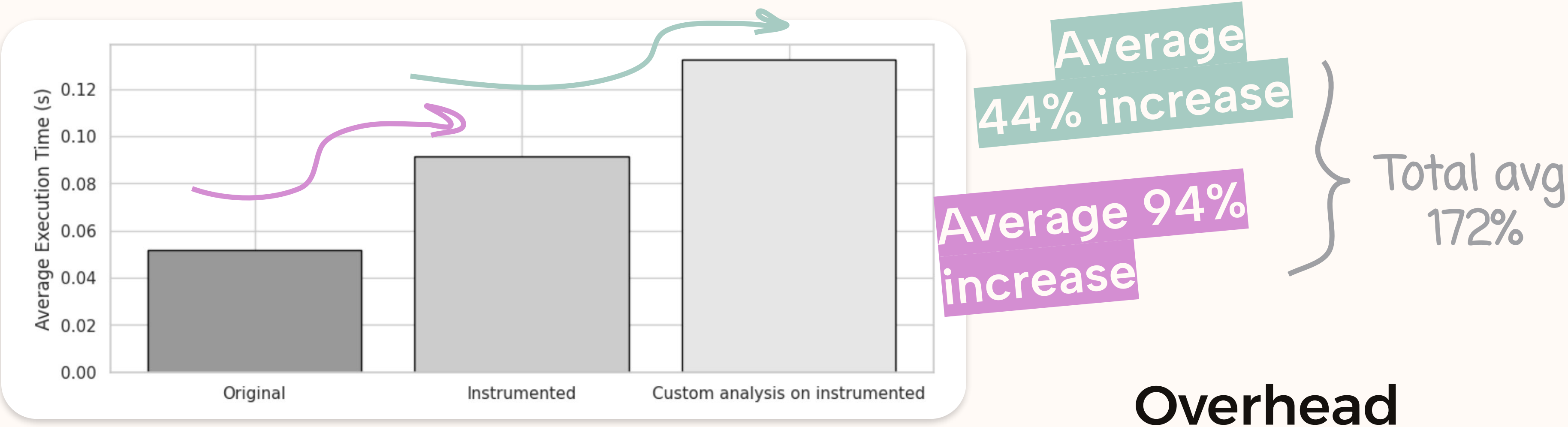
```
220  * Inject a JavaScript or CSS file onto the page
221  *
222  * @param {String} type
223  * @param {String} file
224  * @param {Function} done
225  */
226 exports.inject = function(type, file, done){
227   var startTag, endTag;
228   if (type === "js"){
229     startTag = "<script>";
230     endTag = "</script>";
231   } else {
232     startTag = "<style>";
233     endTag = "</style>";
234   }
235   var self = this;
236   this.page.getContent( function(pageContent){
237     var injectedContents = fs.readFileSync(file);
238     self.page.setContent(pageContent + startTag + injecte
239   });
240 }
```

Evaluation & Results

Correctness
With test cases

Capacity
With third party merges

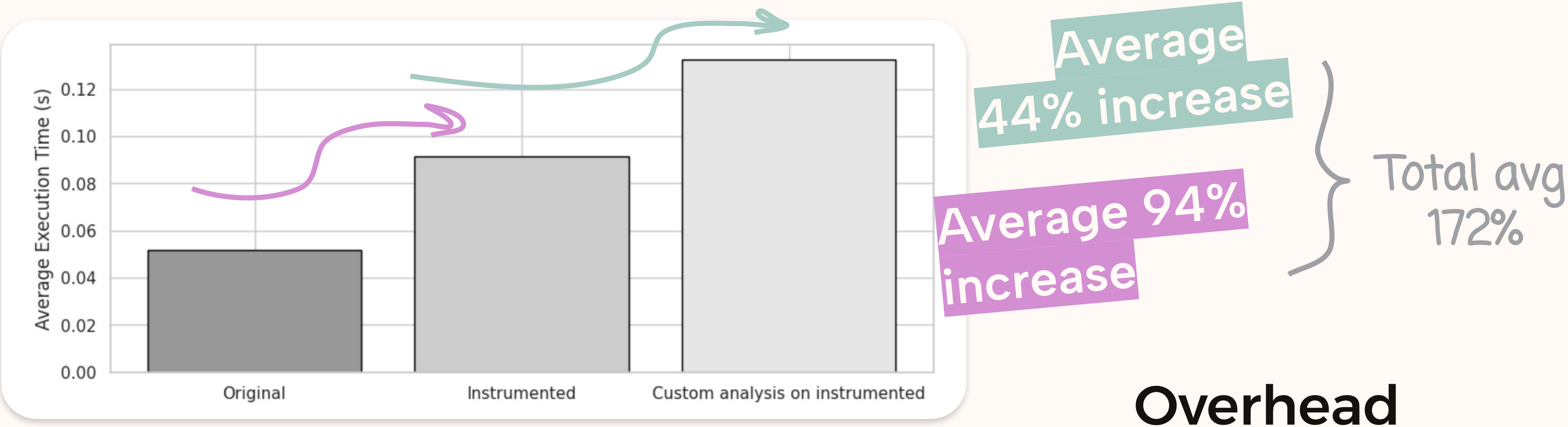
Overhead
On execution time

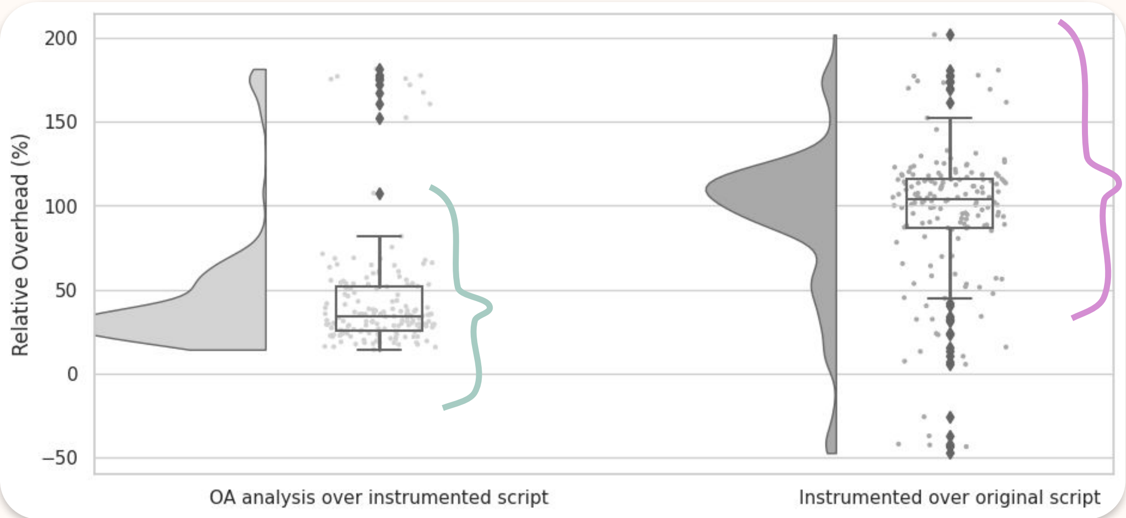


Overhead analysis

The native Jalangi2 instrumentation and analyses represent the major overhead

Centro de Informática UFPE





Overhead analysis

The native Jalangi2 instrumentation and analyses represent the major overhead

Centro de Informática UFPE

Correctness

Reliable implementation of OA detection algorithm

Capacity

We were able to detect an overriding assignment

Overhead

Complexity coupled to the original, major overhead comes from the framework

In the future...

More algorithms to detect conflicts beyond OA

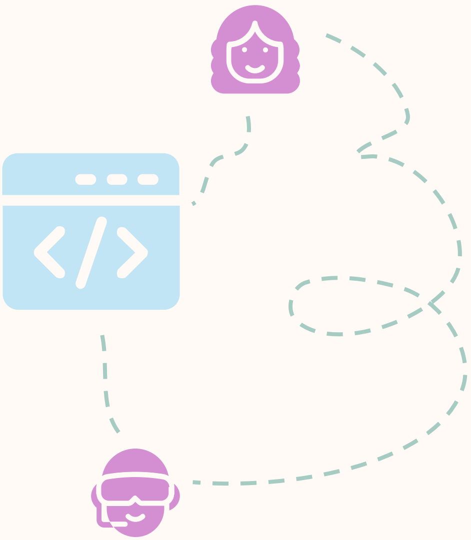
```
var arr = [0, 0, 0, 0, 0]
arr = [1, 1, 1, 1, 1, 1, 1]
console.log('hello')
arr[1] = 1
console.log('world!')
arr[2] = 2
console.log(arr)
```

In the future...

More algorithms to detect conflicts beyond OA

```
var arr = [0, 0, 0, 0, 0]
 arr = [1, 1, 1, 1, 1, 1, 1]
console.log('hello')
 arr[1] = 1
console.log('world!')
 arr[2] = 2
console.log(arr)
```

Plug the solution into code integration pipelines



Centro de Informática UFPE

```javascript
function aumentarContador() {
    ...
}

function iniciarProcesso() {
    ...
}

module.exports = {
    aumentarContador, iniciarProcesso
}
```

Expand analysis action on scripts instructions

Fuzzers, generating tests without assertions, or reusing parameters present in tests from the repositories

Amanda Moraes
CIn – UFPE – ascm@cin.ufpe.br

Paulo Borba
CIn – UFPE – phmb@cin.ufpe.br

Léuson Da Silva
Polytechnique Montreal Canada –
leuson-mario-pedro.da-silva@polymtl.ca

Thank you!

National Institute
of Science and Technology
in Software Engineering

Centro de
Informática
UFPE