Finding the cracks between the analyses

Reshabh Sharma
Finding the cracks between the analyses

Finding the bugs caused by the interactions between the compiler analyses
Finding the cracks between the analyses

Finding the bugs caused by the interactions between the compiler analyses
Finding the cracks between the analyses

Finding the bugs caused by the interactions between the compiler analyses
Revisiting how LLVM optimizes the IR
What may go wrong?

Source → Front-end → IR → Transformations → IR → Analyses → IR → Back-end → asm

Maybe the frontend generated wrong IR
What may go wrong?

- Source to Front-end
- IR to Transformations
- Transformations to Back-end
- Back-end to IR

- Maybe the frontend generated wrong IR
- Maybe the backend generated wrong asm

Analyses
What may go wrong?

- Front-end
- Back-end
- Transformations
- Analyses
- IR
- source
- asm
- Maybe the resultant IR is not equivalent
- Maybe the frontend generated wrong IR
- Maybe the backend generated wrong asm
What may go wrong?

- Maybe the frontend generated wrong IR
- Maybe the resultant IR is not equivalent
- Maybe the analysis result is wrong
- Maybe the backend generated wrong asm
What may go wrong?

- Maybe the frontend generated wrong IR
- Maybe the resultant IR is not equivalent
- Maybe the analysis result is wrong
- Maybe the backend generated wrong asm
Transformations use analyses

Transformations

Analyses

IR
Transformations use analyses
Transformations use analyses
Transformations use analyses
Transformations use analyses
Transformations use analyses
What may go wrong?

Transformation

Analyses
What may go wrong?

Transformation

Maybe the analysis generates wrong result
Interaction between the analyses

Transformation

Analyses

If both the analysis results are correct individually
Interaction between the analyses

We want to find bugs due to this interaction
Analyses
Fundamental Analyses

These analyses do not depend on any other analysis
Derived or dependent Analyses

This analysis depends on the results from other analyses.
Testing differentially
Testing differentially
Testing differentially

Test

Equivalent
Testing differentially

Equivalent

Test
Testing differentially

Test

Equivalent

Check Equivalence
Testing differentially

Test

Check Equivalence
Testing differentially

Test A

Check Equivalence

Test B
Testing differentially

Test A ← Equivalent → Test B

Check Equivalence
Testing differentially

Test → Equivalent mutation → Mutated test

Check Equivalence
Testing analysis differentially

Test → Equivalent mutation → Mutated test

Analysis

Check Equivalence
Checking equivalence of analyses result

It is not straightforward, and a notion of equivalence needs to be established for each analysis result.
Case study: SwapTrueFalse (STF)

- STF is a structural mutation
- It swaps true and false branches
Case study: SwapTrueFalse (STF)
Case study: SwapTrueFalse (STF)
Case study: SwapTrueFalse (STF)
Case study: SwapTrueFalse (STF)

We can use a mutation to differentially test an analysis that does not react to it.

STF does not change the structure of the control-flow graph so can be used for analysis that are independent of CFG.
Source of bug

The bug in any fundamental analysis will only be caused by itself
Source of bug

The bug in any fundamental analysis will only be caused by itself

These analyses do not depend on any other analysis
Source of bug

The bug in any derived analysis can be caused by itself
Source of bug

The bug in any derived analysis can be caused by itself

This analysis depends on the results from other analyses
Source of bug

The bug in any derived analysis can be caused by any dependent analysis.
Source of bug

The bug in any derived analysis can be caused by the interaction between the analyses.

This analysis depends on the results from other analyses.
Source of bug

The bug in any derived analysis can be caused by the following,

- Can be caused by itself
- Can be caused by any other analysis it depends on
- Can be caused by the interaction between the analyses
Source of bug

- Bugs in the analysis itself
- Bugs due to the interaction
- Bugs in the analysis it depends on
Source of trust
Source of trust

Fundamental analysis are the easiest to reason about
Source of trust

Fundamental analysis are the easiest to reason about
Source of trust

Fundamental analysis are the easiest to reason about
Source of trust

Fundamental analysis are the easiest to reason about
Source of trust

- Bugs in the analysis itself
- Bugs due to the interaction
- Bugs in the analysis it depends on
Source of trust

- Bugs in the analysis itself
  - Bugs due to the interaction
  - Bugs due to other analyses shrinks
Source of trust

 UNIT TESTS

Bugs due to the interaction

Bugs from itself shrinks

Bugs due to other analyses do not change much
Source of trust

If we choose mutations which are inert to the analysis in question and the analyses it uses
Source of trust

If we choose mutations which are inert to the analysis in question and the analyses it uses

Bugs due to the interaction

Now, we can focus on bugs due to the interaction
Example: Region Info Analysis

Region Info Analysis

- Dominance Frontier Analysis
- Post Dominator Tree Analysis
- Dominator Tree Analysis
Mutation: SwapTrueFalse (STF)
Mutation: SwapTrueFalse (STF)
A test case (T)
Applying STF at bb1
Applying STF at bb1

- bb0
- true
  - true
    - bb1
    - false
    - bb2
    - false
    - bb4
  - true
    - bb3

Applying STF at bb1
Mutated test case (MT)

Diagram:
- bb0
- bb1
- bb2
- bb3
- bb4

bb0 -> bb1
bb1 -> bb2, true, false
bb2 -> true, bb3, false
bb3 -> true, bb2, false
bb4 -> false, bb3
Region Info Analysis

A Region is a connected subgraph of a control flow graph that has exactly two connections to the remaining graph.
Region Info Analysis

A Region is a connected subgraph of a control flow graph that has exactly two connections to the remaining graph.
Region Info Analysis

A Region is a connected subgraph of a control flow graph that has exactly two connections to the remaining graph.
Region Info Analysis

A Region is a connected subgraph of a control flow graph that has exactly two connections to the remaining graph.
How bad is missing a region?

There is a region pass in the legacy pass manager
Was it really an interaction bug?
How does region info analysis works
Is it a region?
Is it a region?
Is it a region?
Is it a region?

- 1 (entry)
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

- dominate
Is it a region?

entry

dominate

exit
Is it a region?
Is it a region?
Is it a region?
Is it a region?
Is it a region?
Is it a region?
Is it a region?
Is it a region?
Is it a region? Yes

entry

dominate

post-dominate

exit
The test case and the analyses results
The test case and the analyses results

```
bb0 -> bb1
  |                       |
  v                       v
true | false  <- true      false | true
  |                       |
  v                       v
bb2 <- true  -> bb3
  |                       |
  v                       v
false
  |                       |
  v                       v
bb4
```
The test case and the dominator tree
The test case and the dominator tree

```plaintext
bb0
  ↓
  bb1
false | true
  ↓
  bb2
true
  ↓
  bb3
false
  ↓
  bb4
```

```plaintext
bb0
  ↓
  bb1
  ↓
  bb2
  ↓
  bb3
  ↓
  bb4
```
The test case and the post dominator tree
The test case and the post dominator tree (again)
Two legal post dominator trees
Two legal post dominator trees
Test cases and analyses

Dominator Tree

Virtual

Post Dominator Tree 1
Let’s find regions

Dominator Tree

Virtual

Post Dominator Tree 1
Let’s find regions

Dominator Tree

Virtual

Post Dominator Tree 1
Let’s find regions

Dominator Tree

Virtual

Post Dominator Tree 1
Let’s find regions
Let’s find regions

- Dominator Tree
- Virtual
  - bb4
  - bb1
  - bb2
  - bb3
  - bb0
- Post Dominator Tree 2
- bb0
- bb1
- bb2
- bb3
- bb4

- true | false
- false | true
- true
- false
Let’s find regions

Dominator Tree

Virtual

Post Dominator Tree 2
What went wrong?

Region analysis did the right thing at its end

Post dominator analysis also did the right thing at its end
How bad is missing a region?

- Post Dominator Tree Analysis
- Dominator Tree Analysis
- Dominance Frontier Analysis
- Region Info Analysis
- Structurize CFG

[Diagram showing relationships between different analysis stages]
How bad is missing a region?

Post Dominator Tree Analysis
Dominator Tree Analysis
Dominance Frontier Analysis
Region Info Analysis
Polly’s Scoop Info Analysis
Structurize CFG
Prune Unprofitable
Schedule
Unprofitable
Simplify
DeLICM
Conclusion

Compilers are complex 😊
Conclusion

Compilers are complex 😊

Special cases for each analysis should be kept in mind while using them

New techniques to suggest tests based on interaction between the analyses
Thank You!