Finding the cracks between the analyses

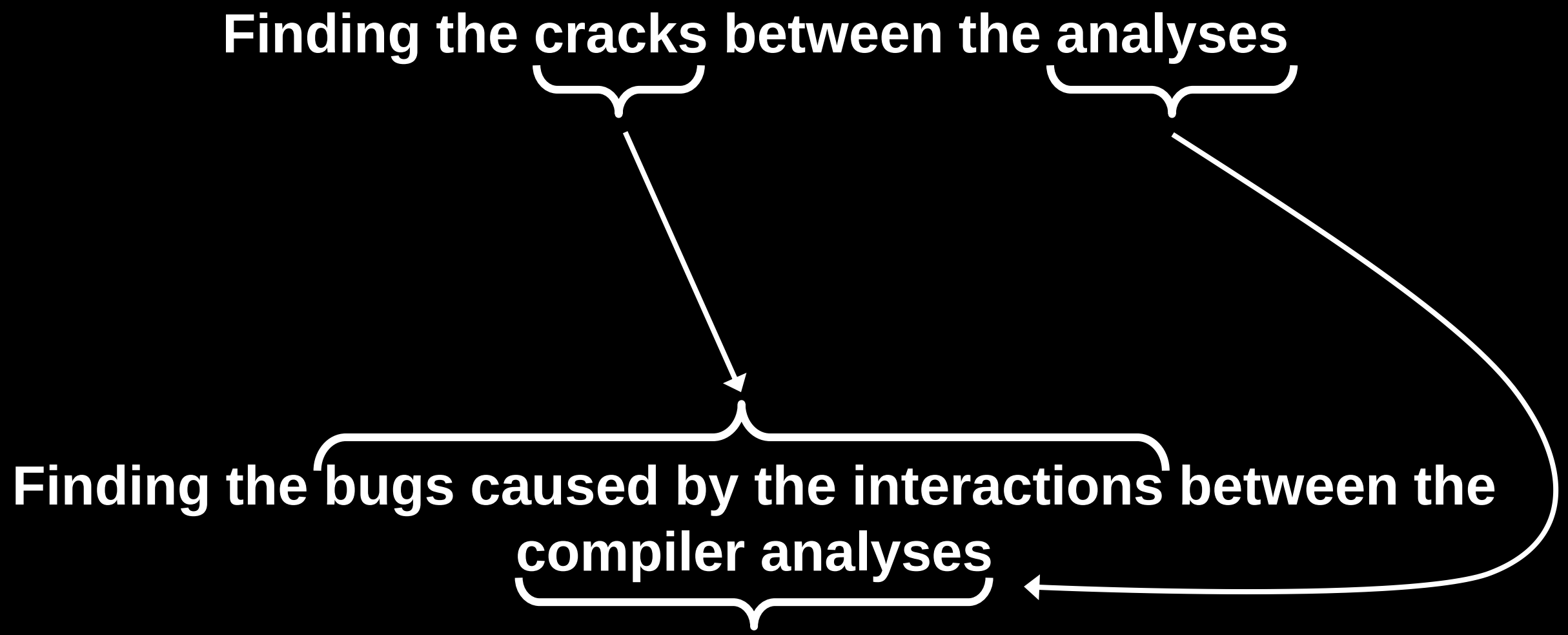
Reshabh Sharma

Finding the cracks between the analyses

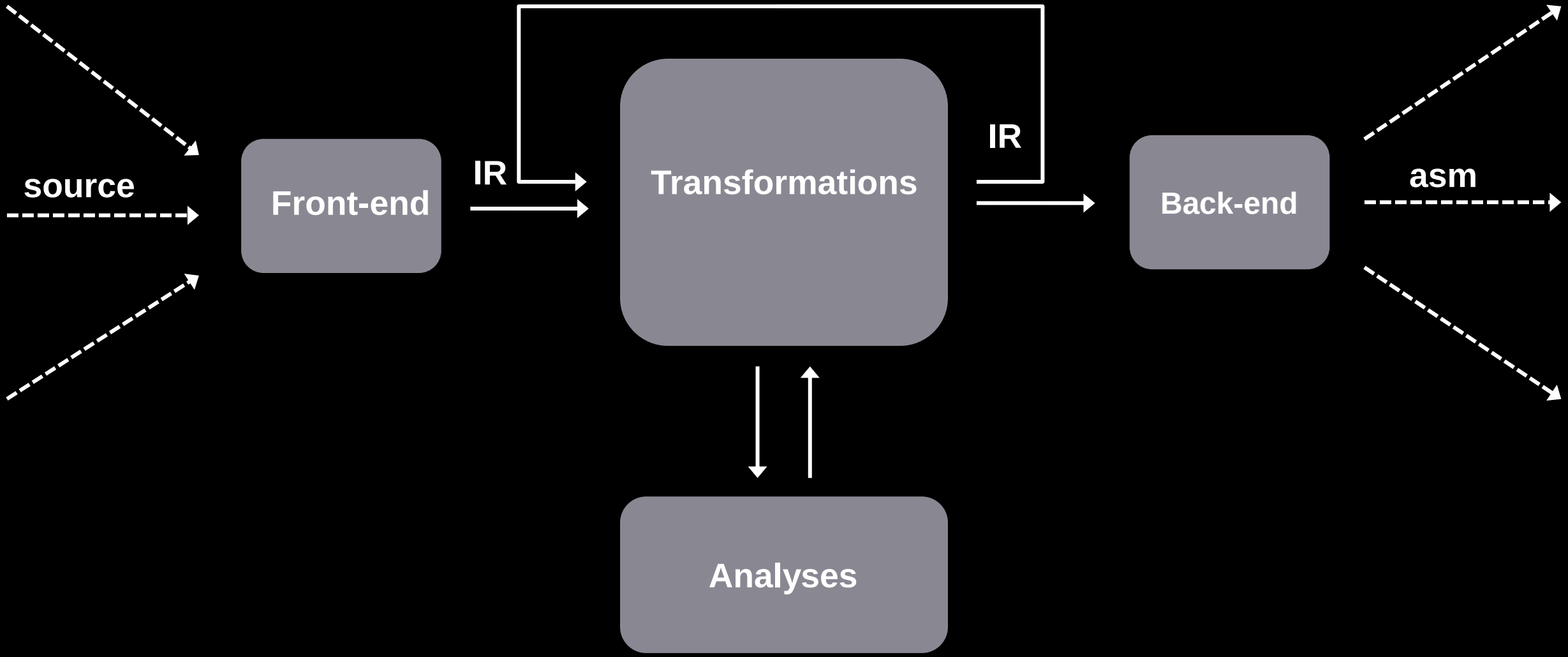
Finding the bugs caused by the interactions between the compiler analyses

Finding the cracks between the analyses

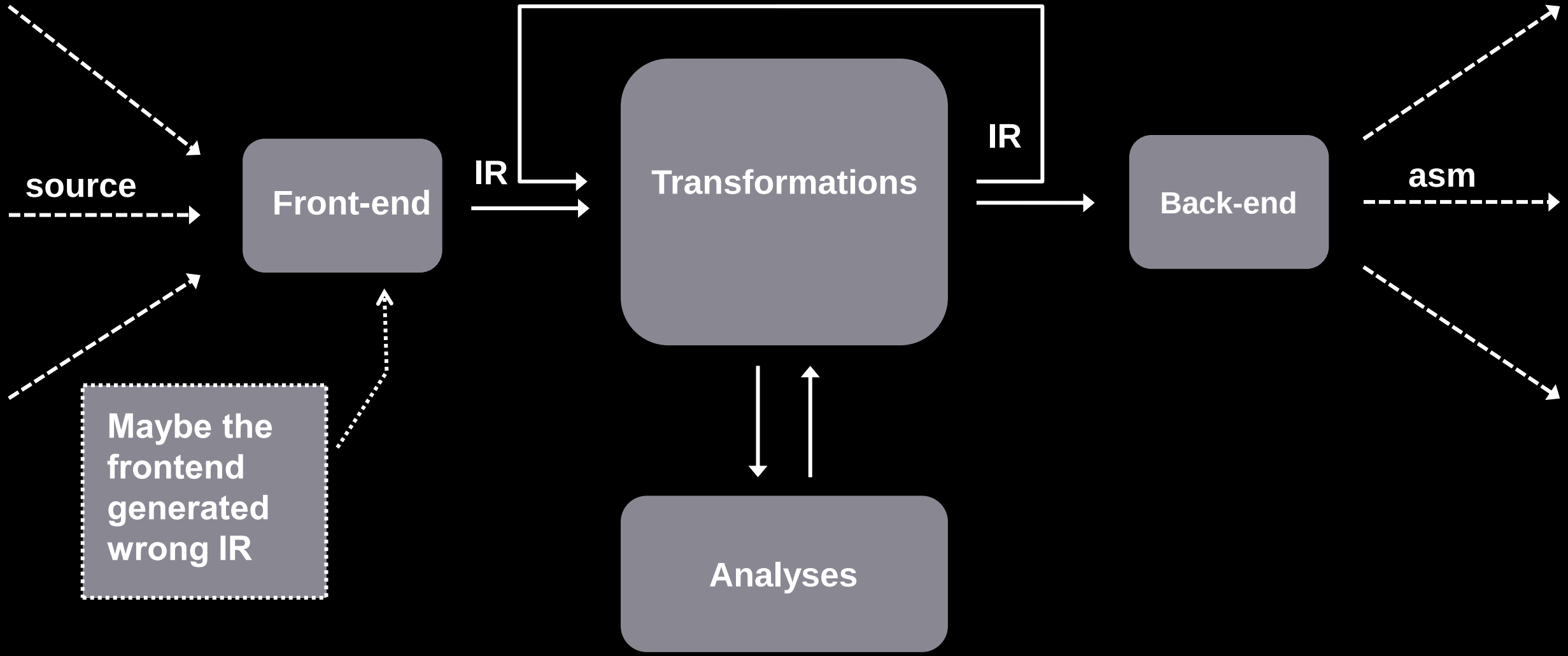
Finding the bugs caused by the interactions between the compiler analyses


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

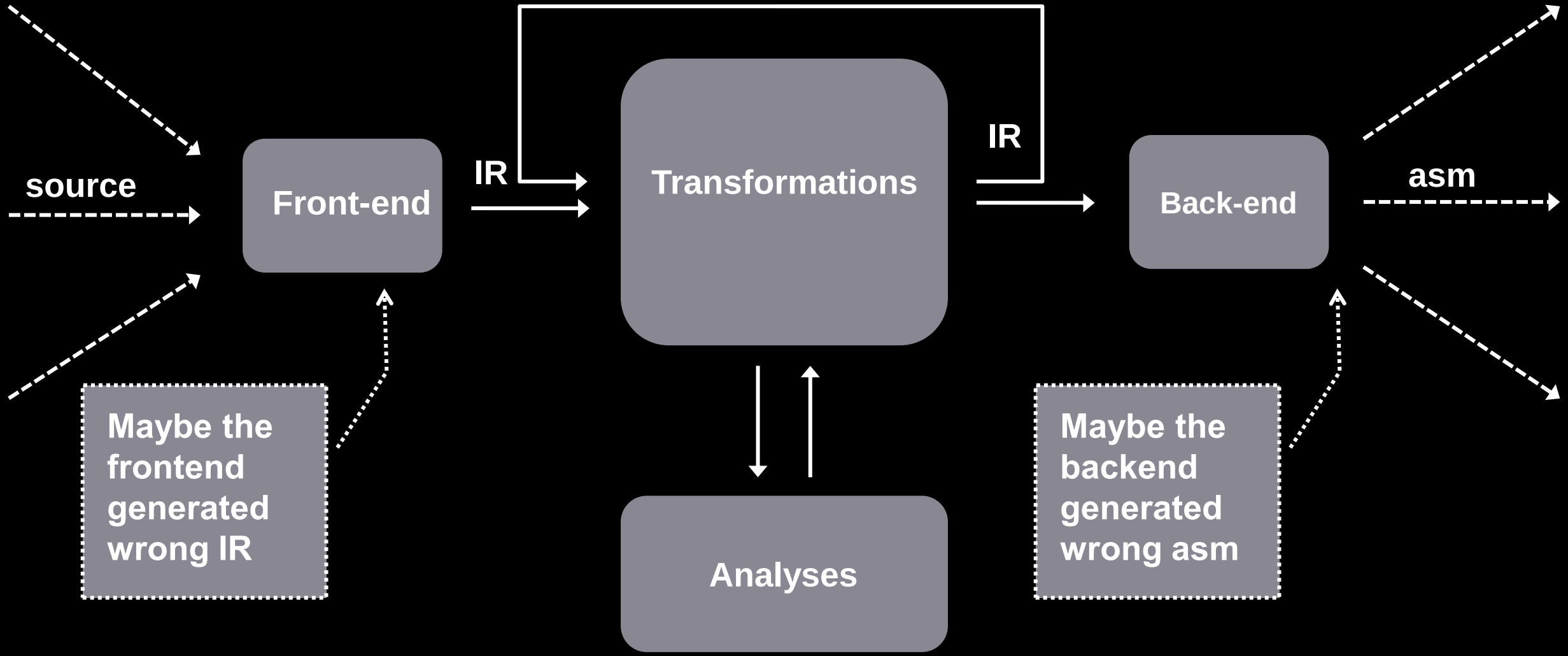
Revisiting how LLVM optimizes the IR



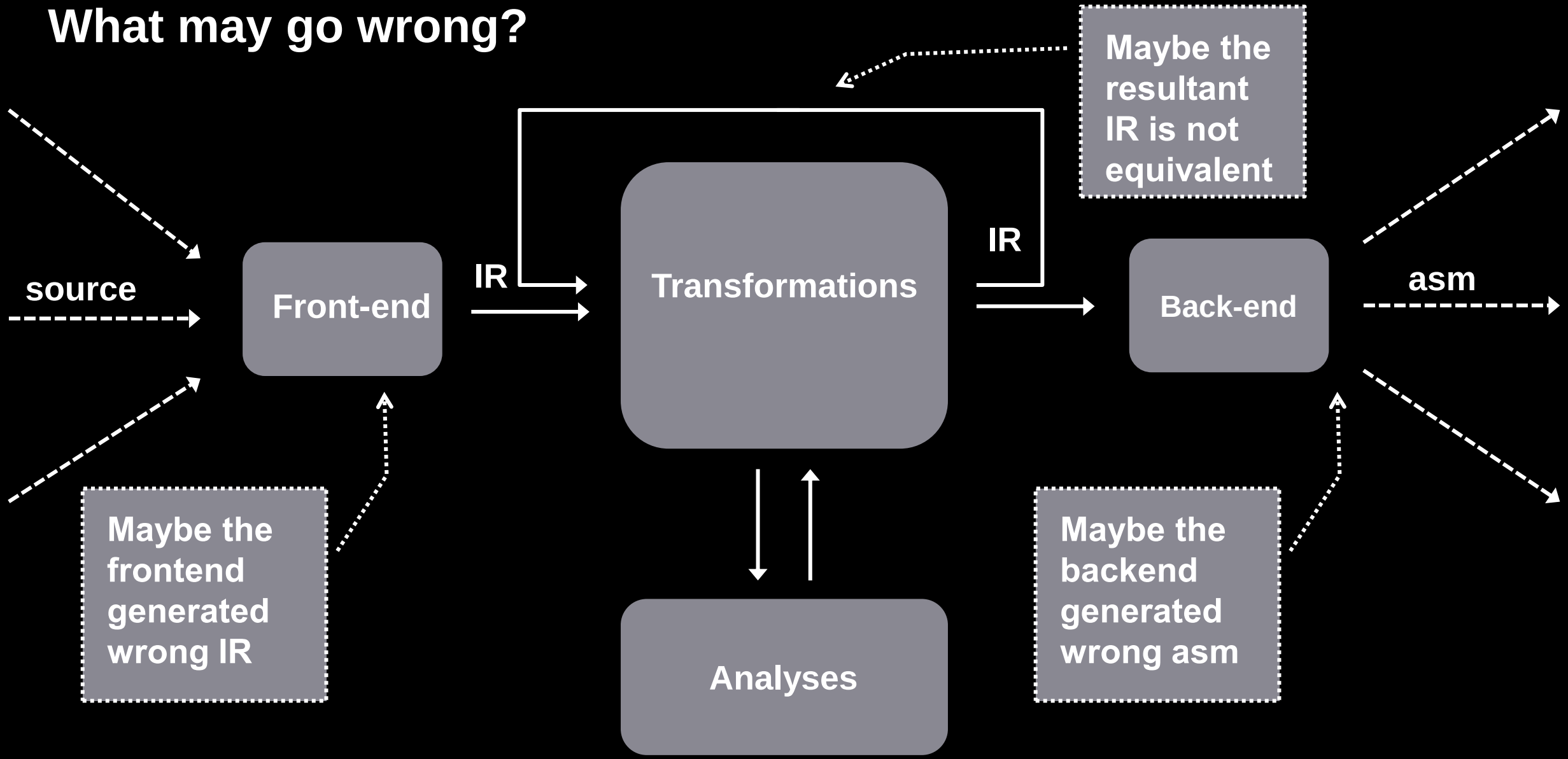
What may go wrong?



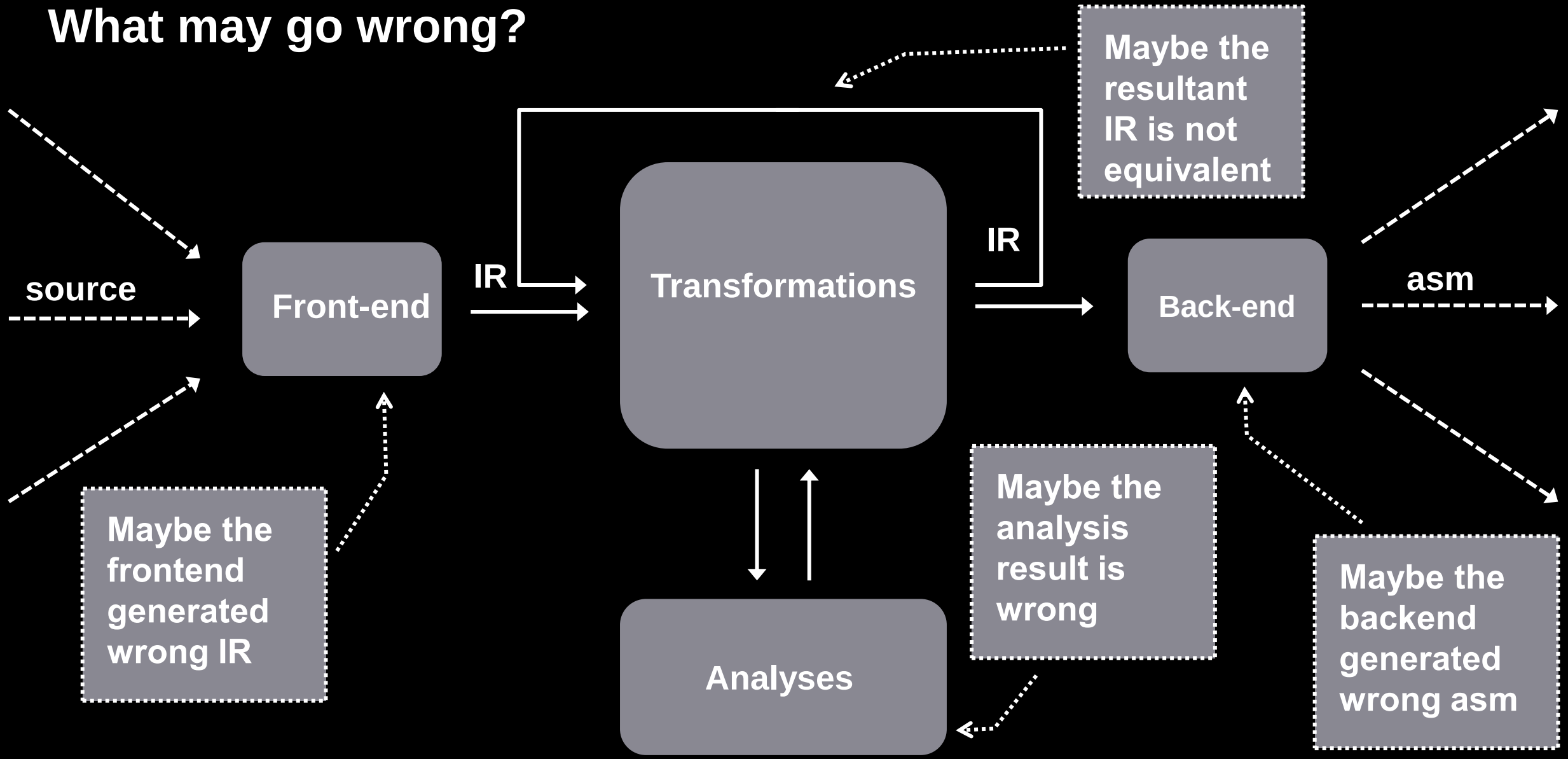
What may go wrong?



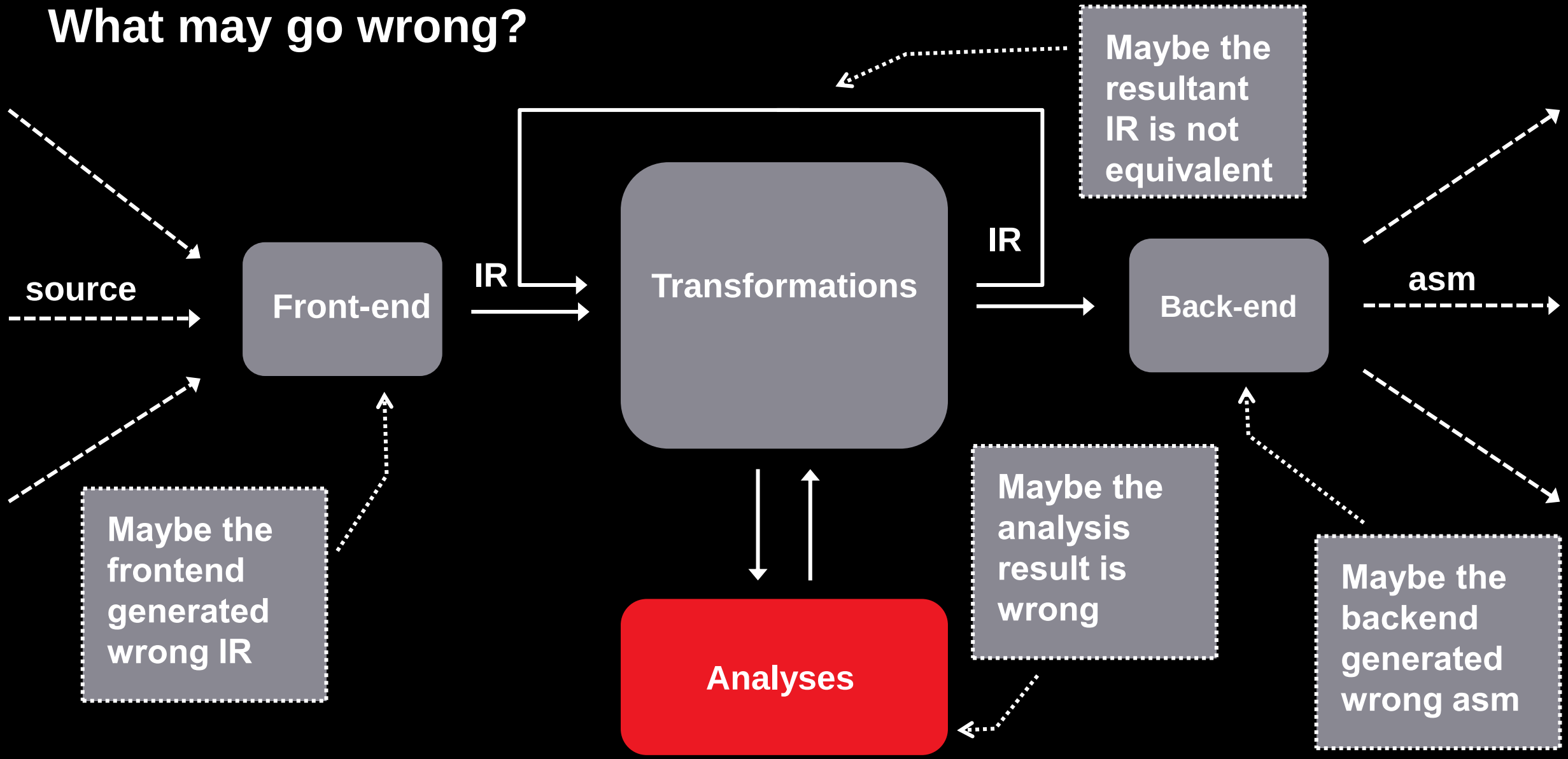
What may go wrong?



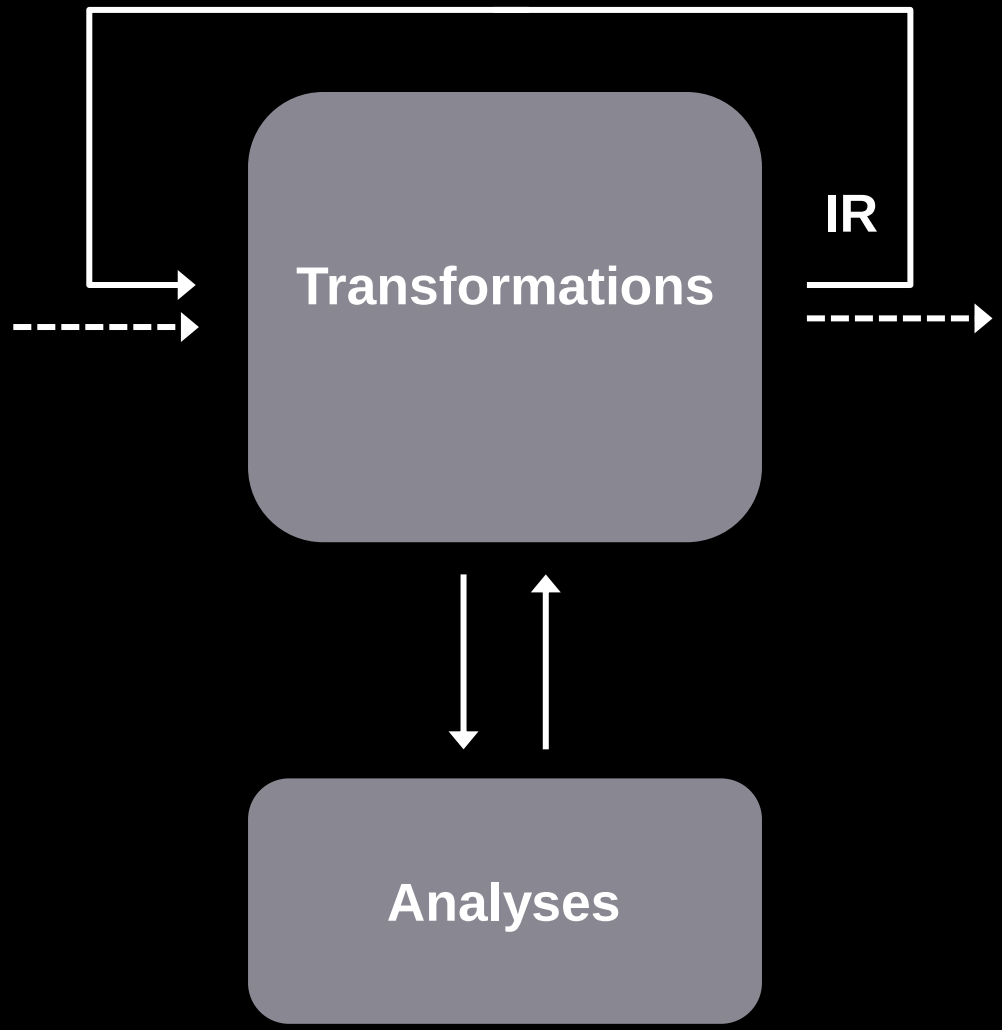
What may go wrong?



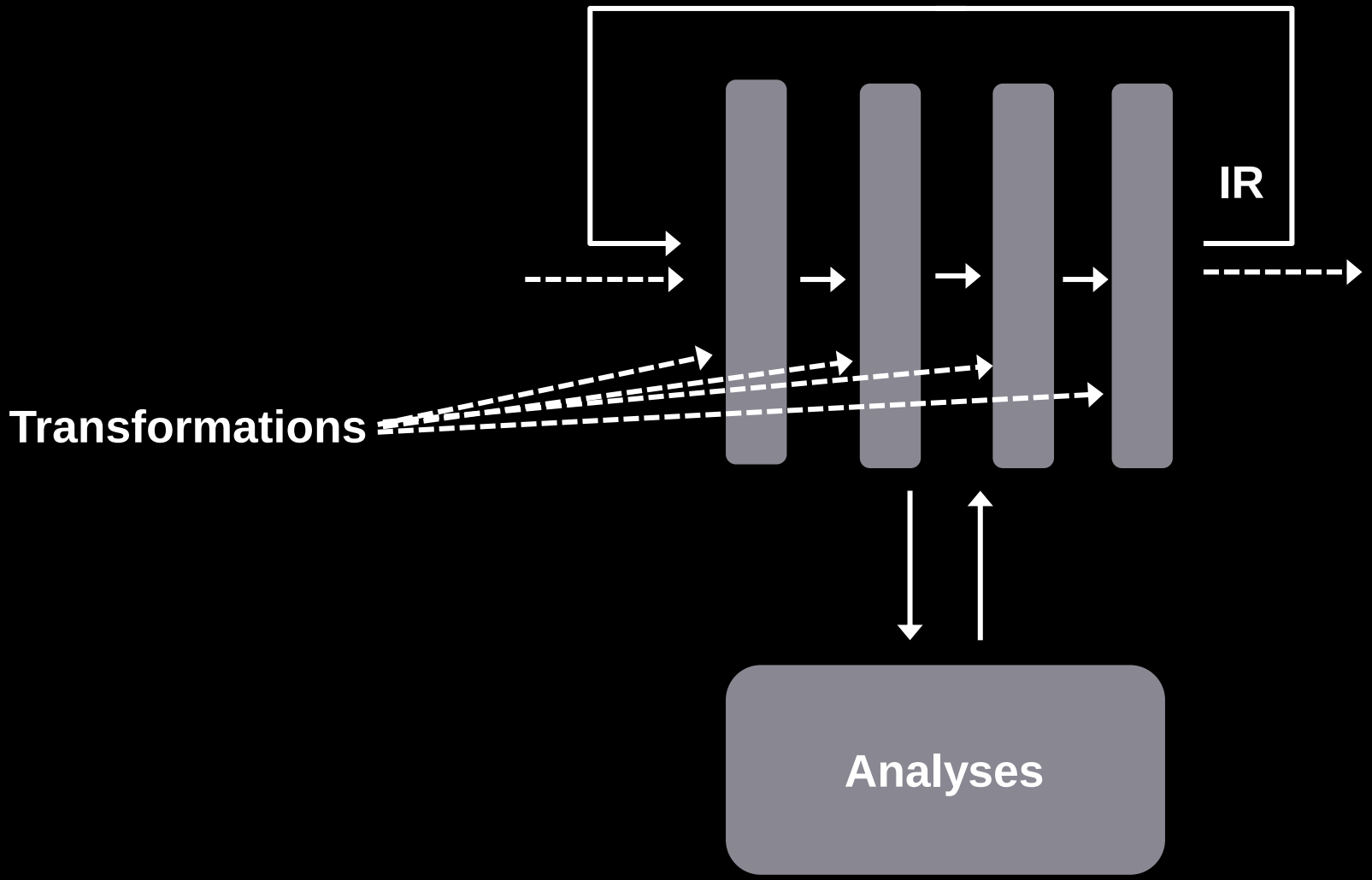
What may go wrong?



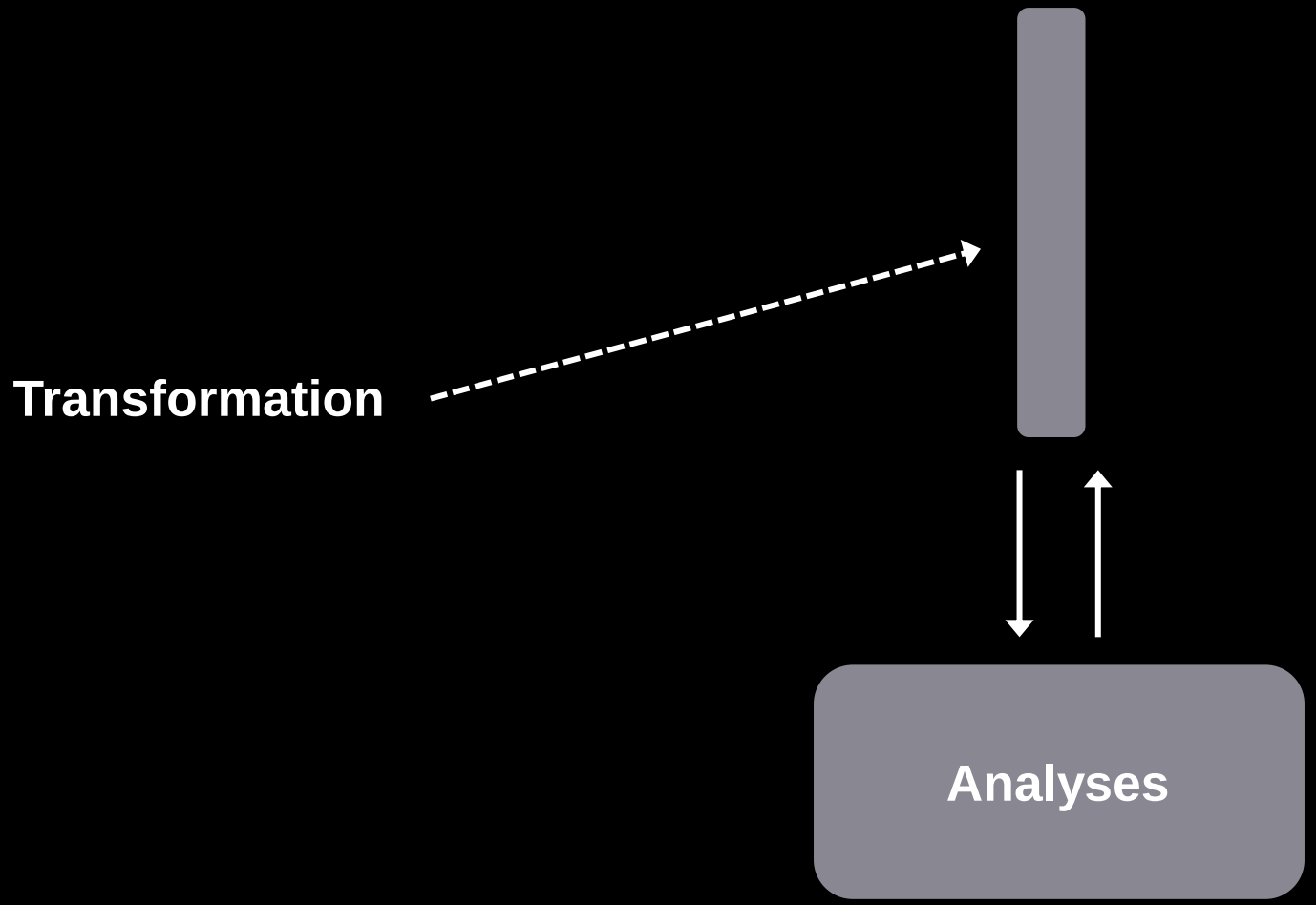
Transformations use analyses



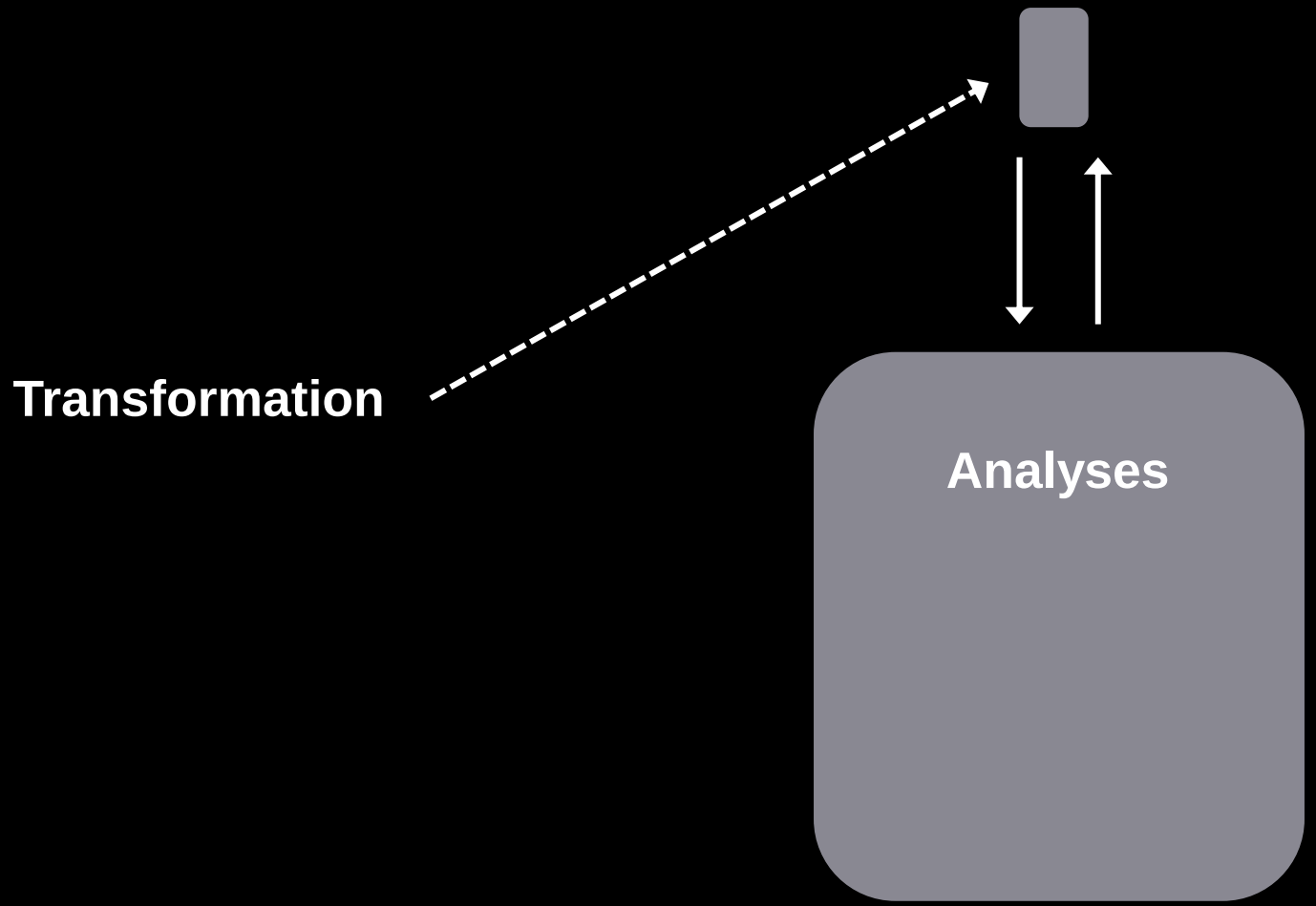
Transformations use analyses



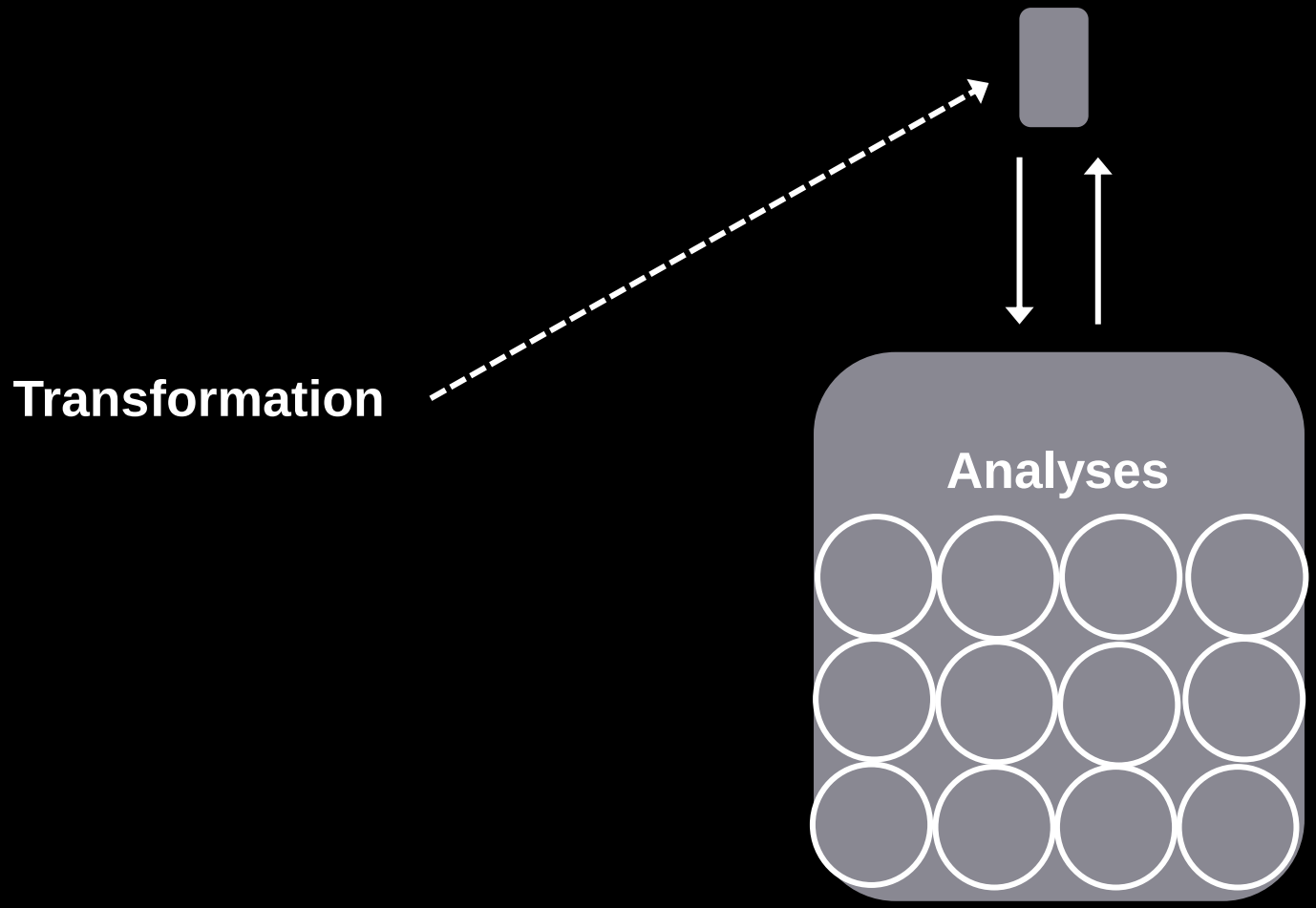
Transformations use analyses



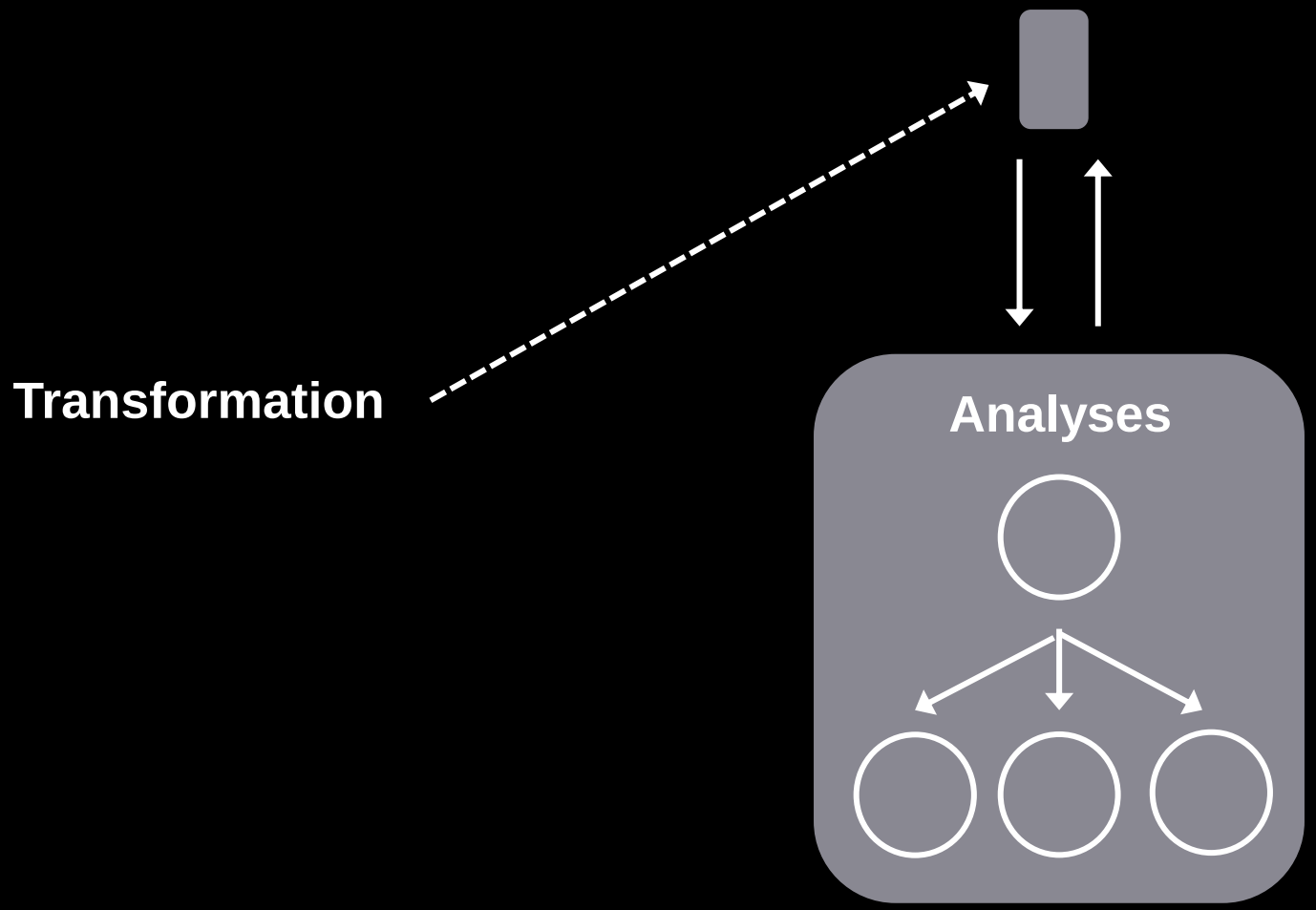
Transformations use analyses



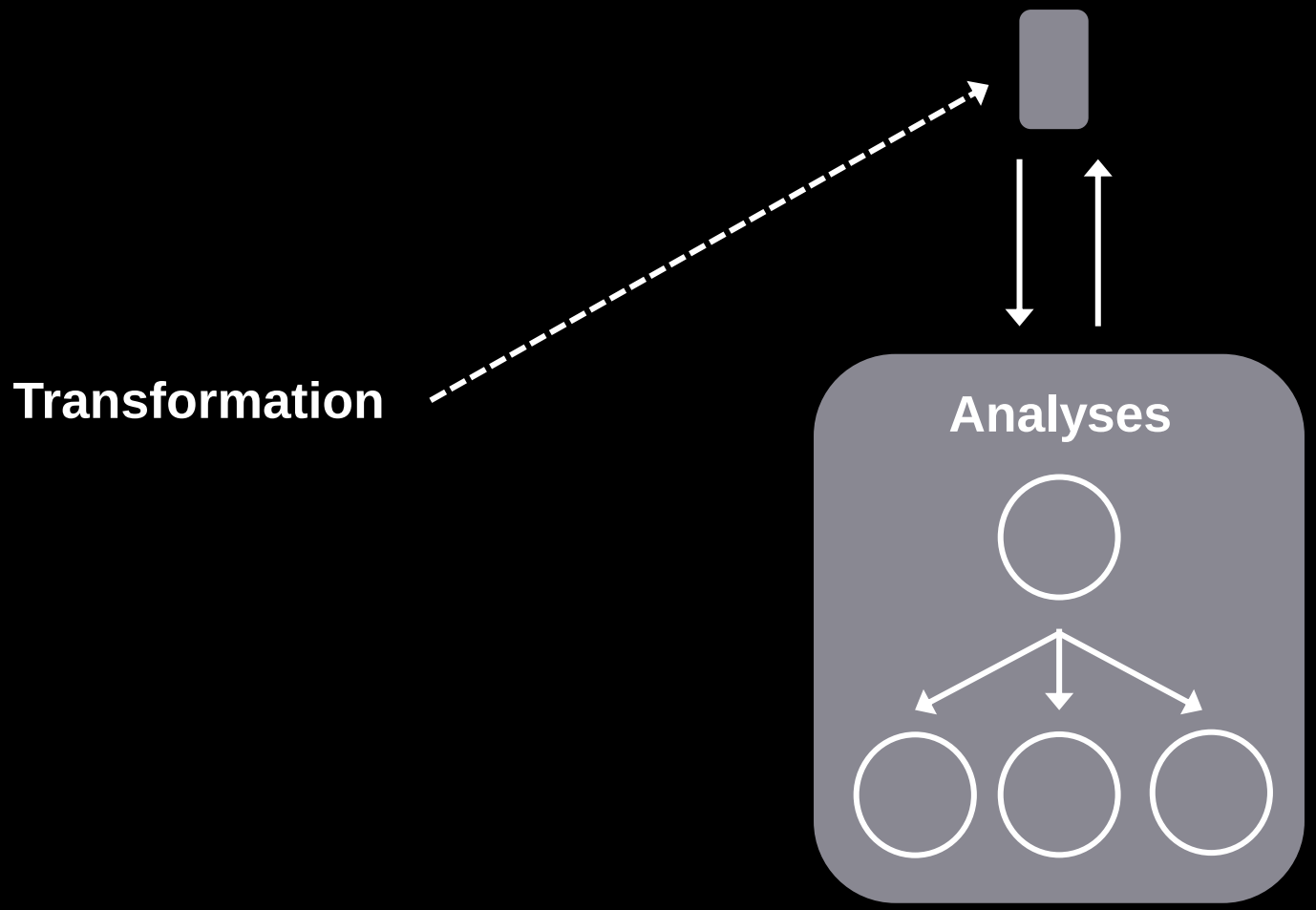
Transformations use analyses



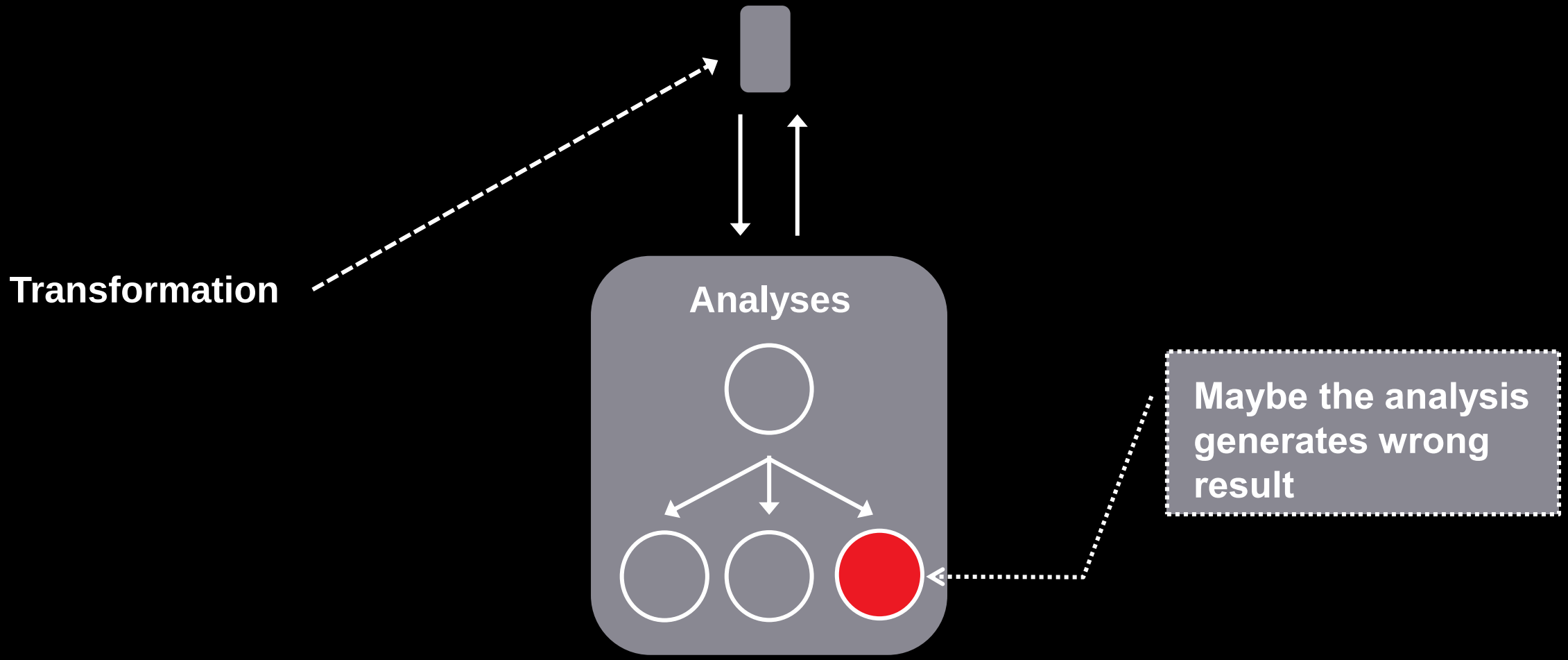
Transformations use analyses



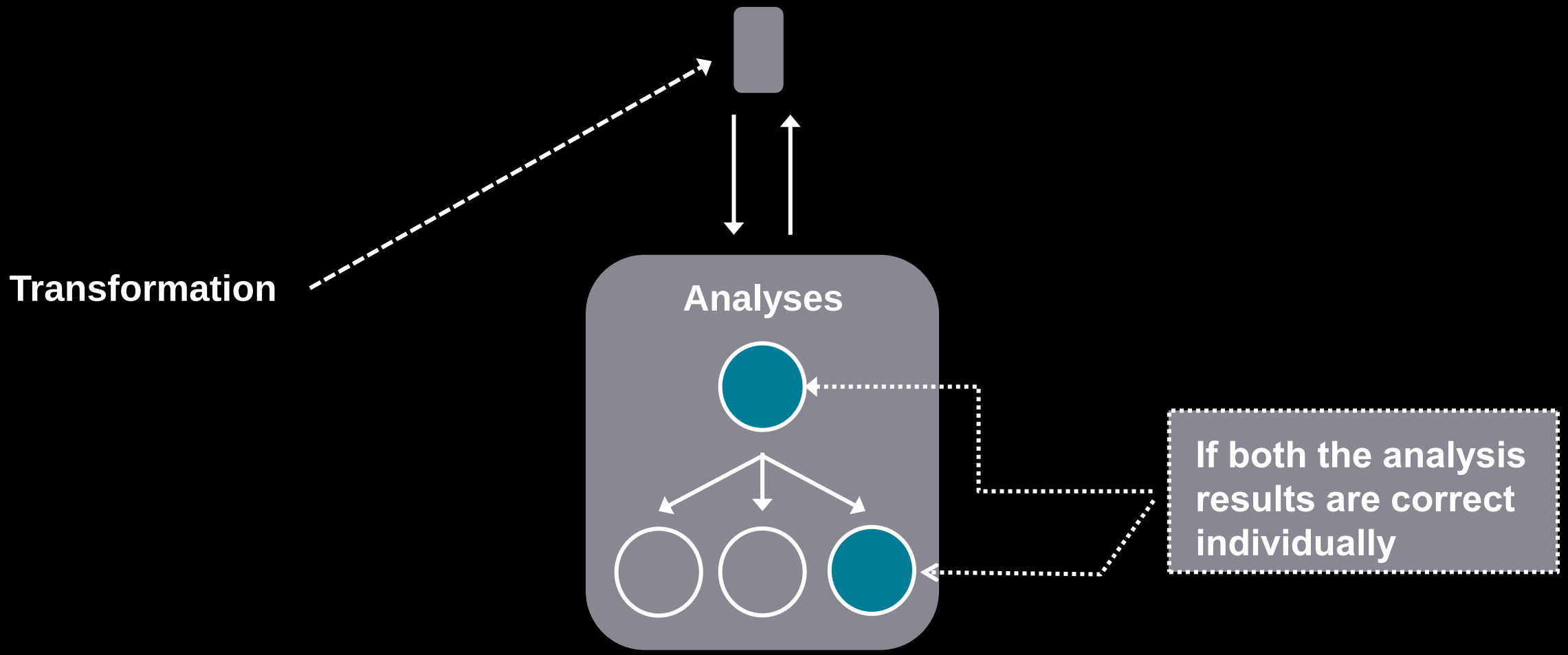
What may go wrong?



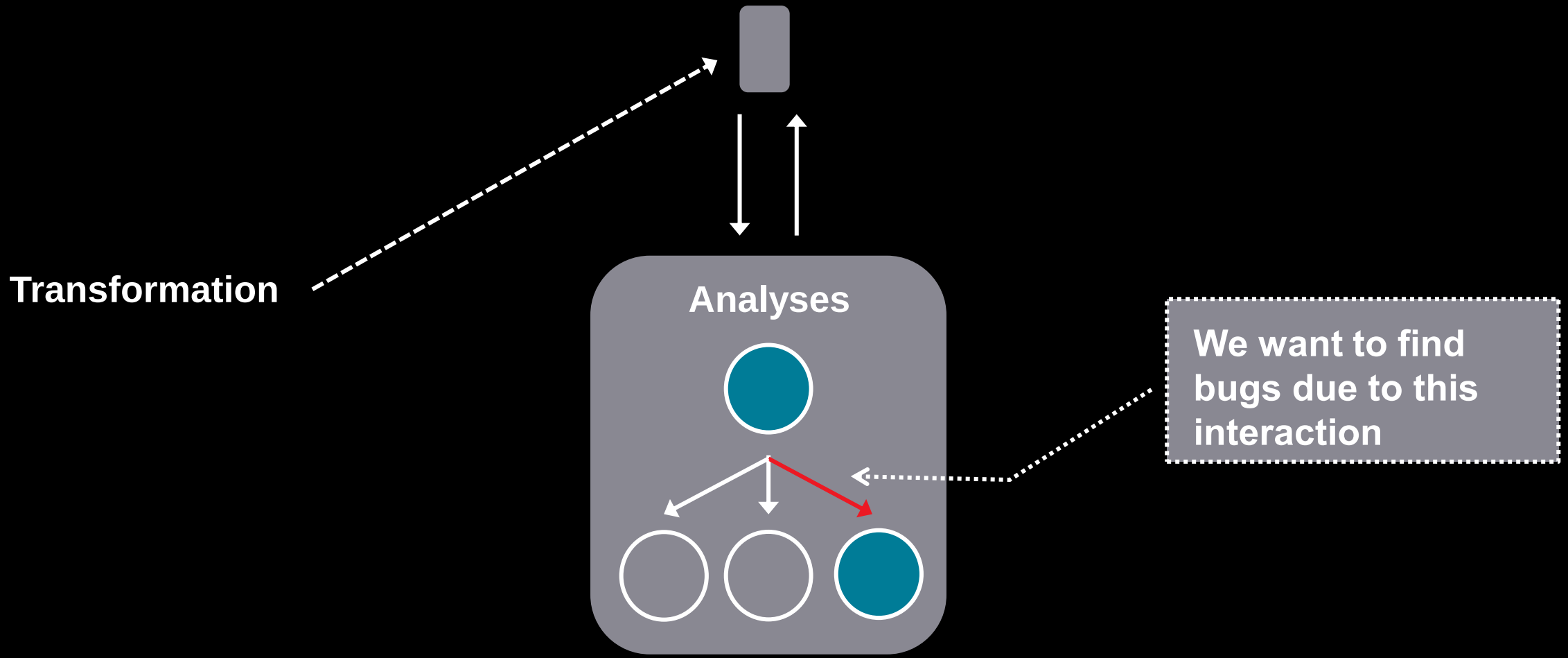
What may go wrong?



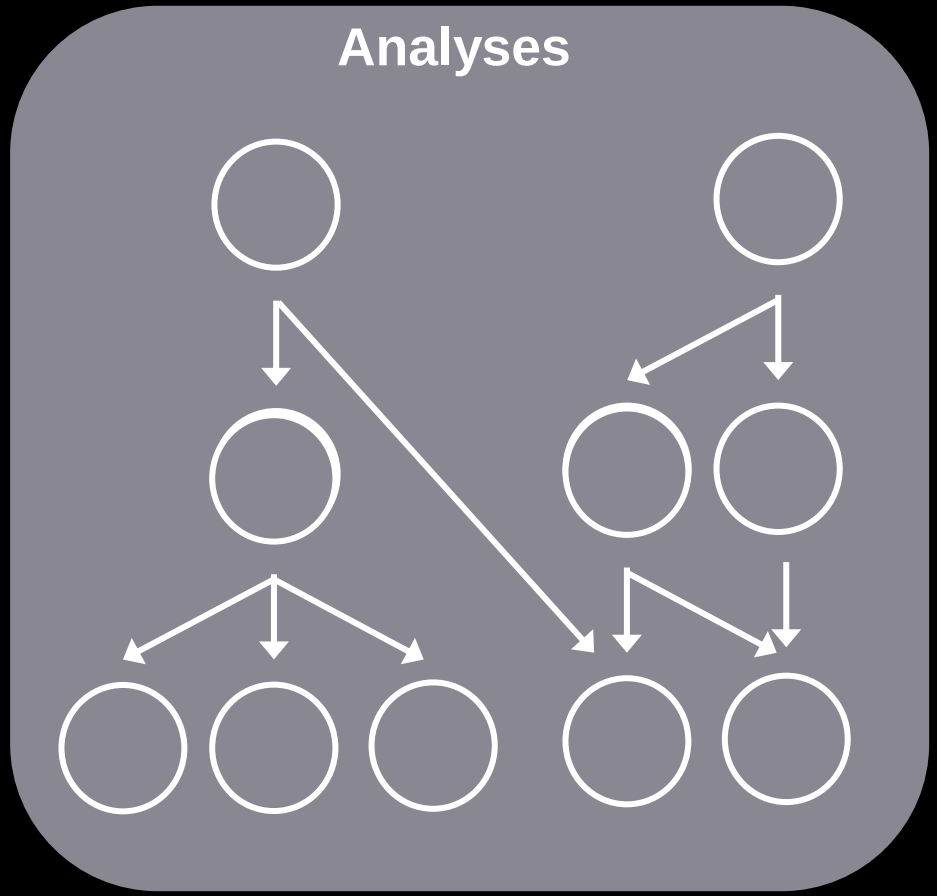
Interaction between the analyses



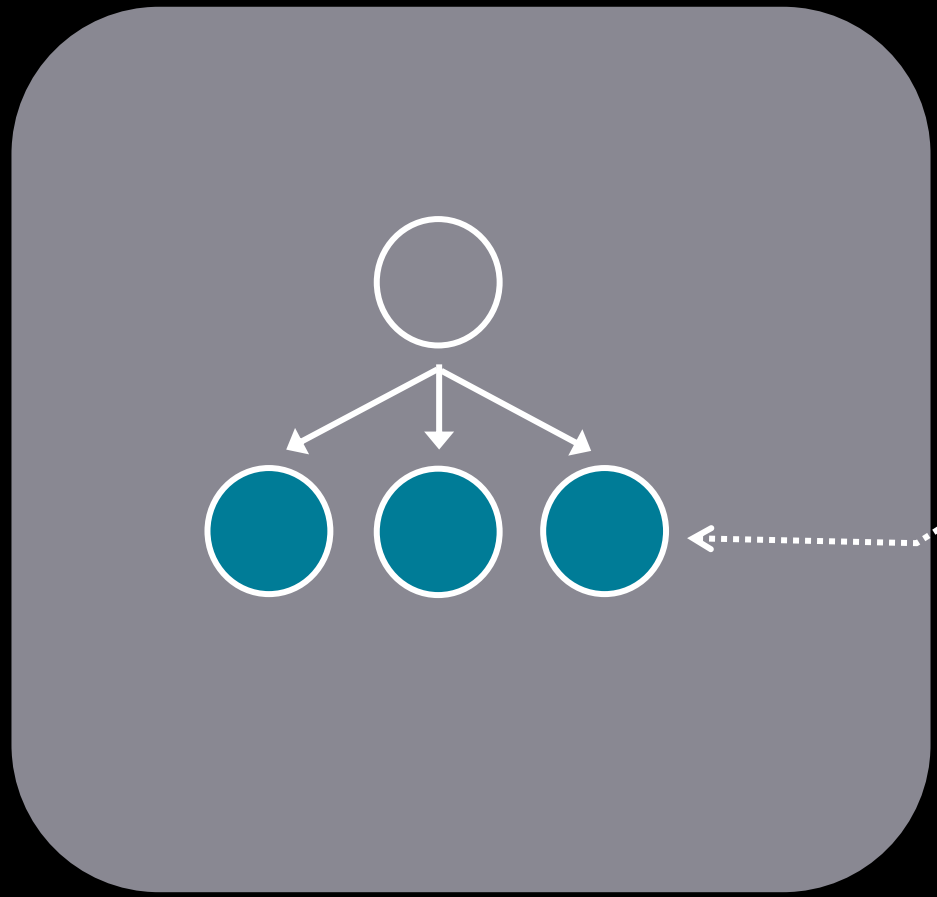
Interaction between the analyses



Analyses

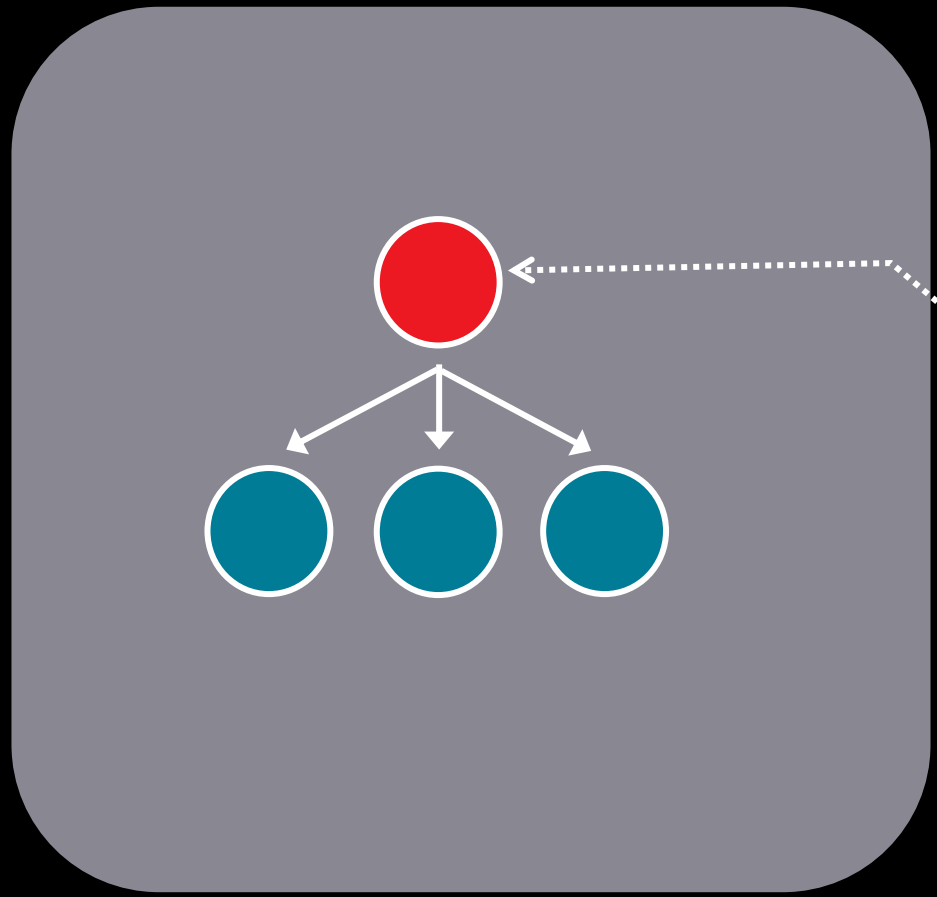


Fundamental Analyses



These analyses do not depend on any other analysis

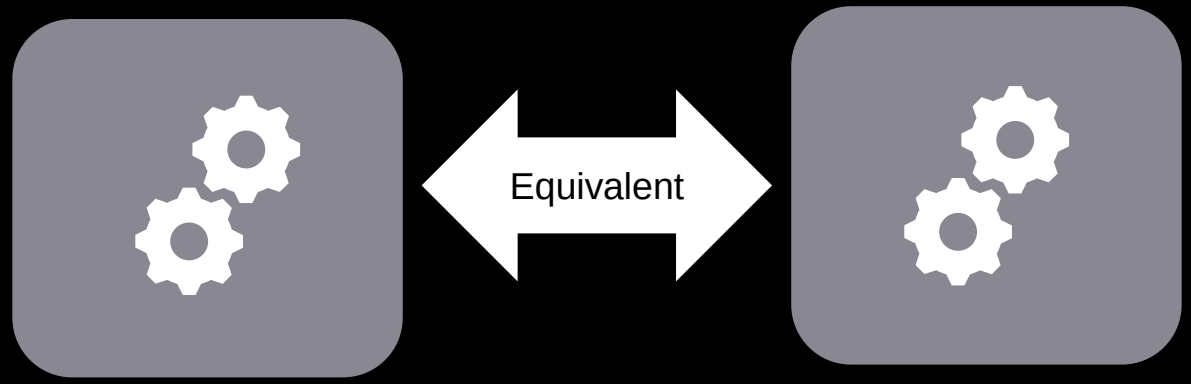
Derived or dependent Analyses



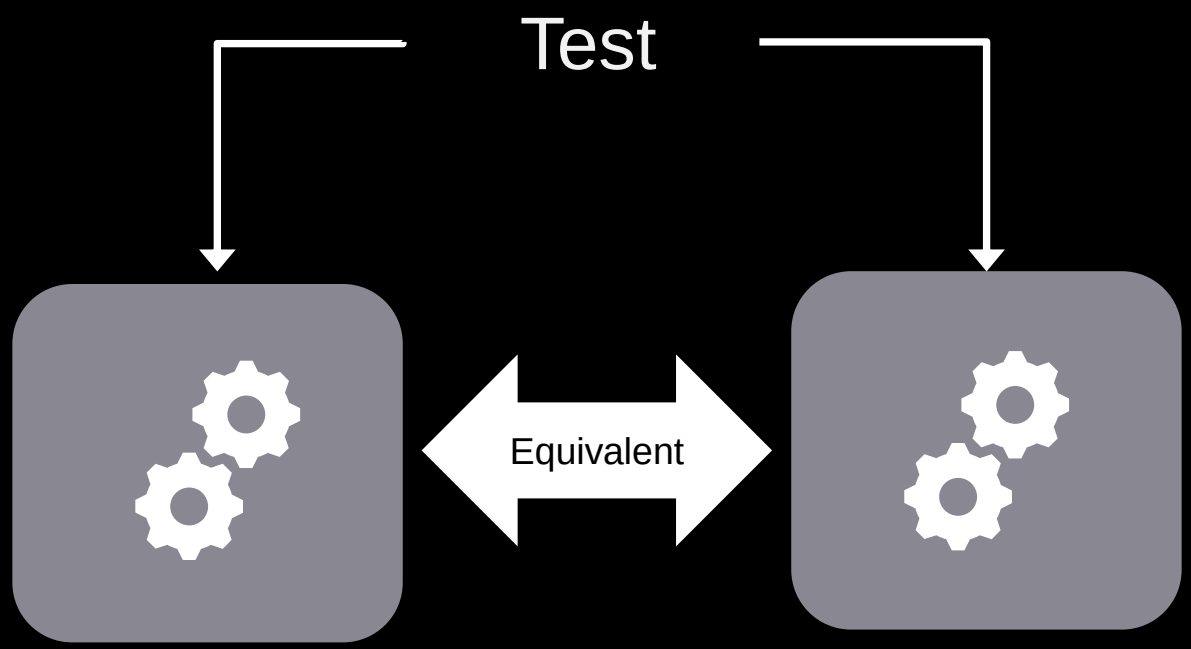
This analysis depends on the results from other analyses

Testing differentially

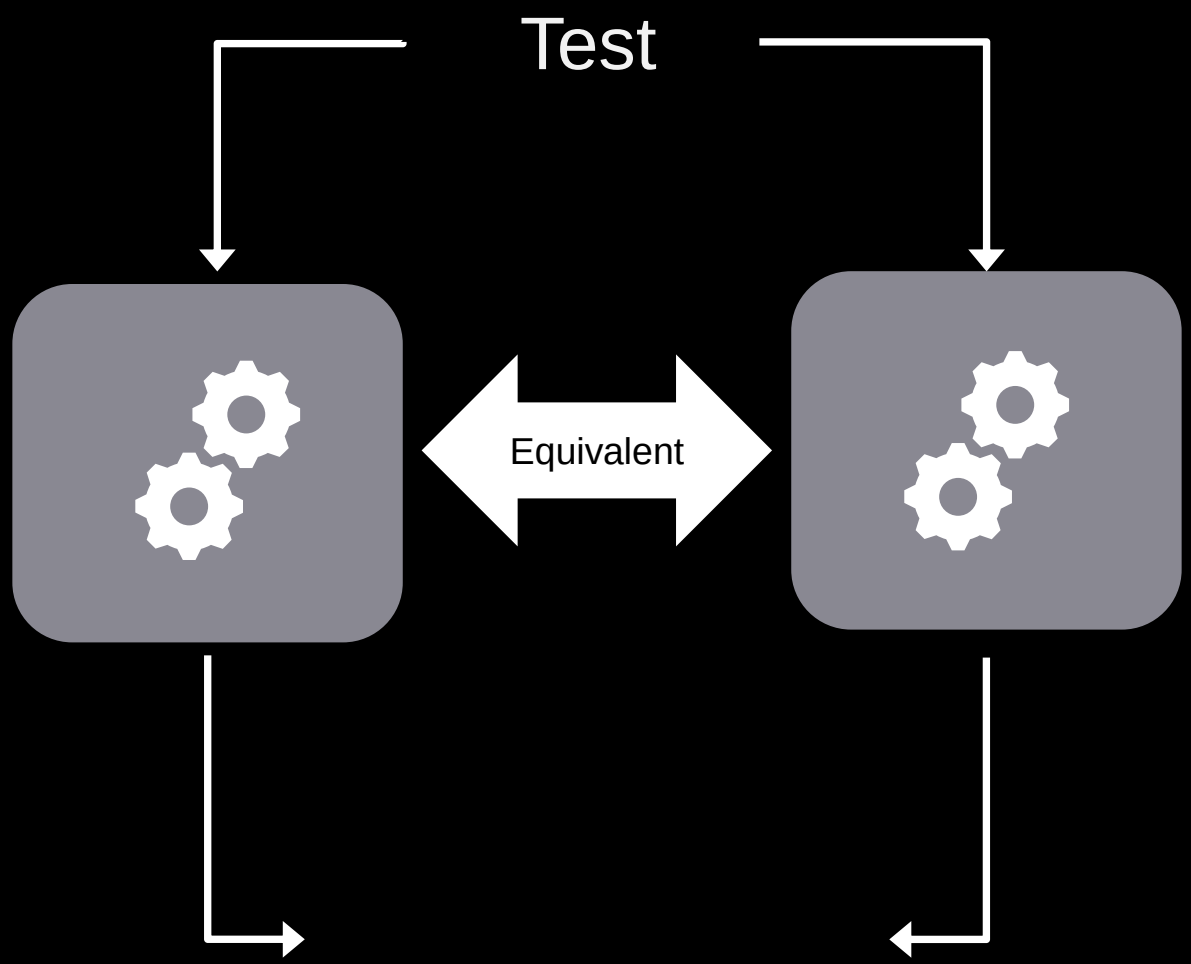
Testing differentially



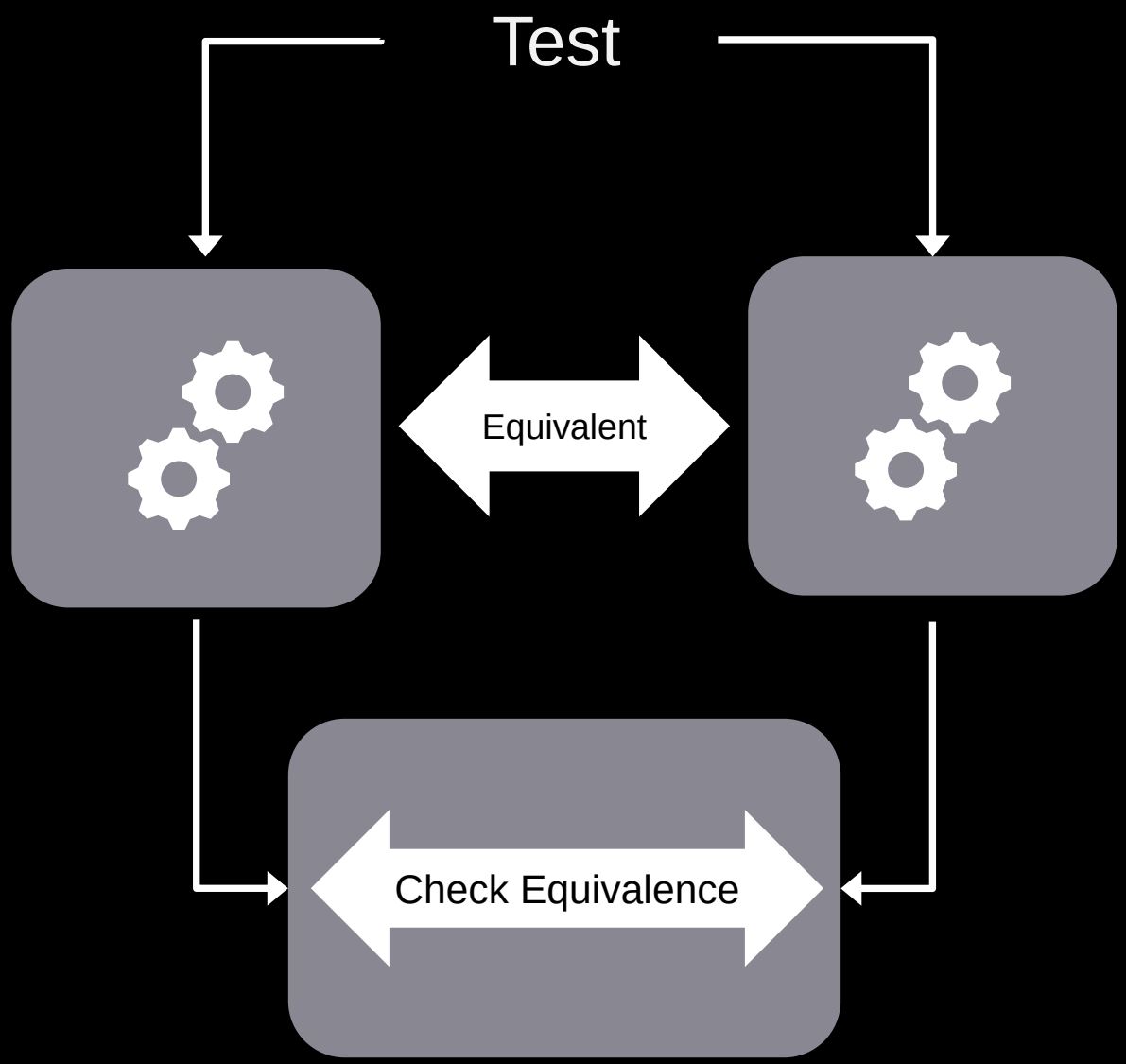
Testing differentially



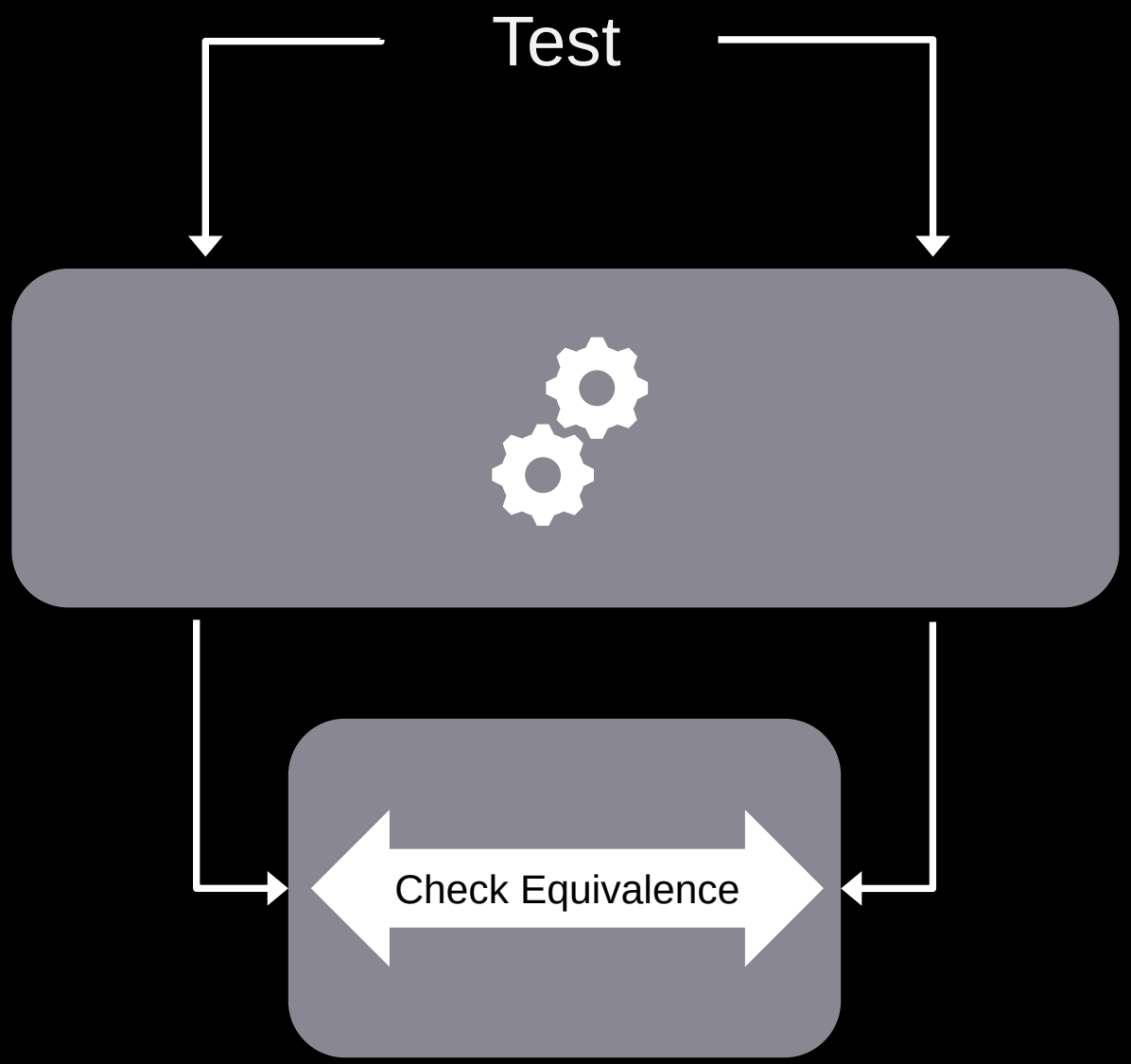
Testing differentially



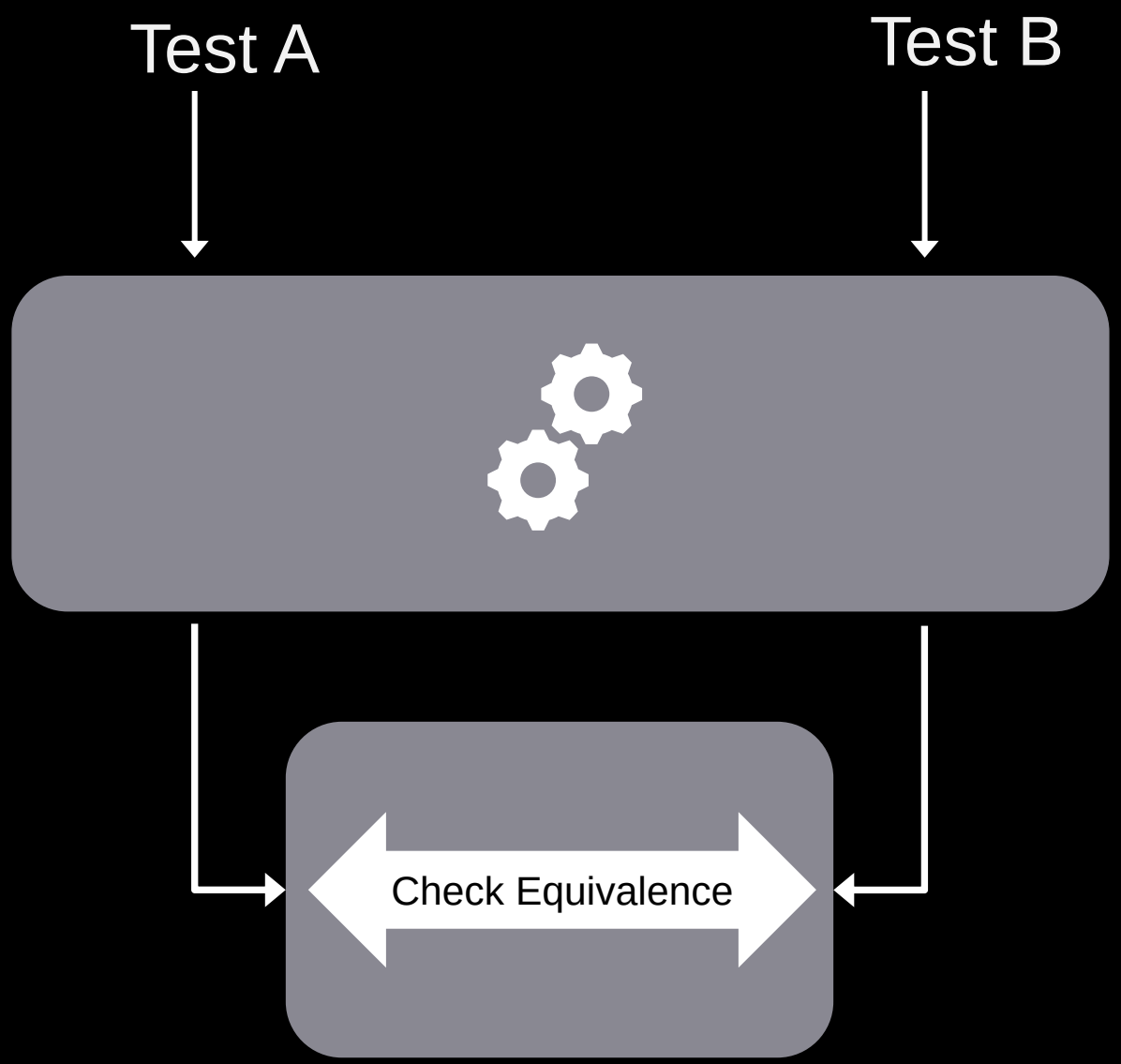
Testing differentially



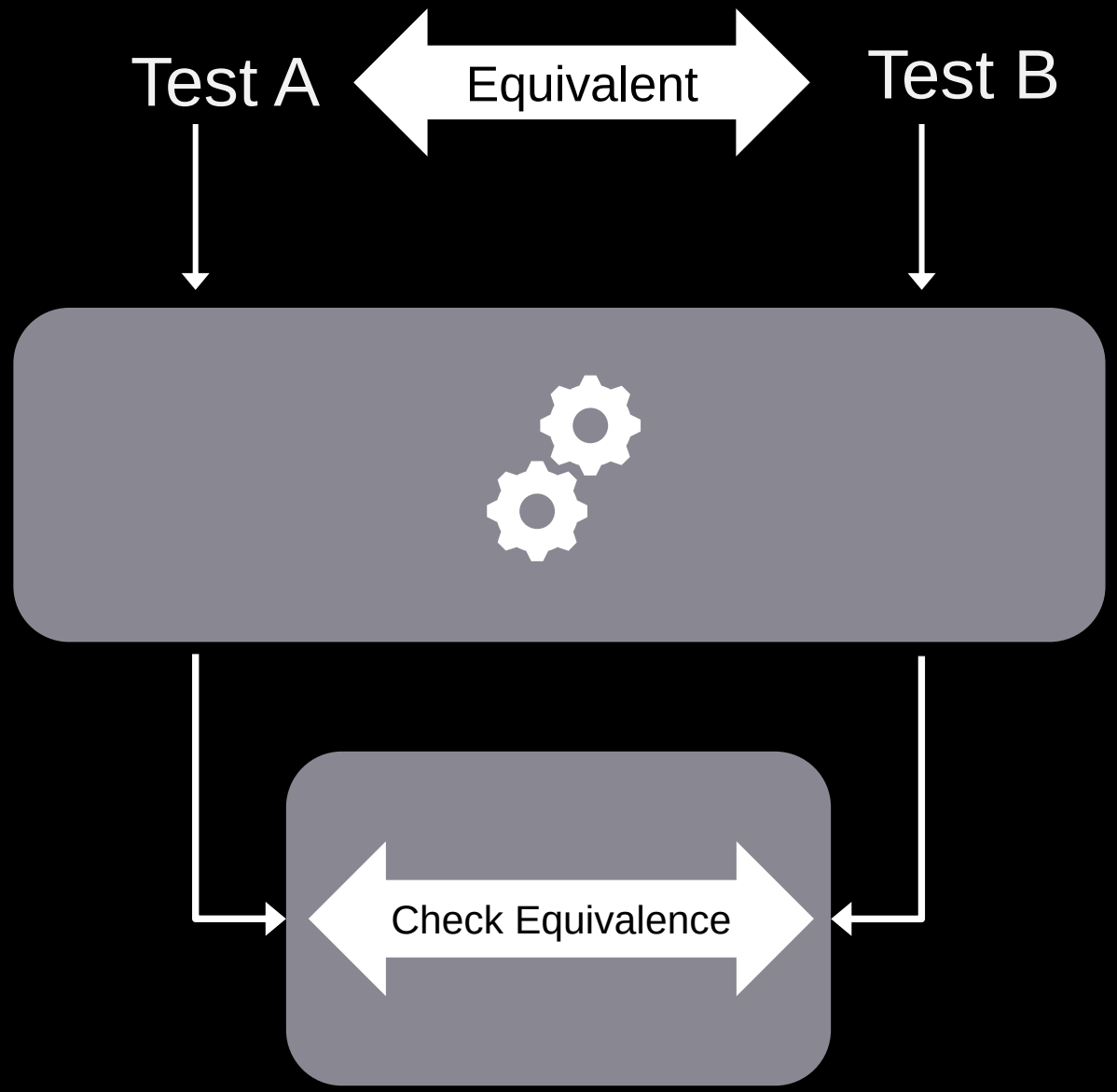
Testing differentially



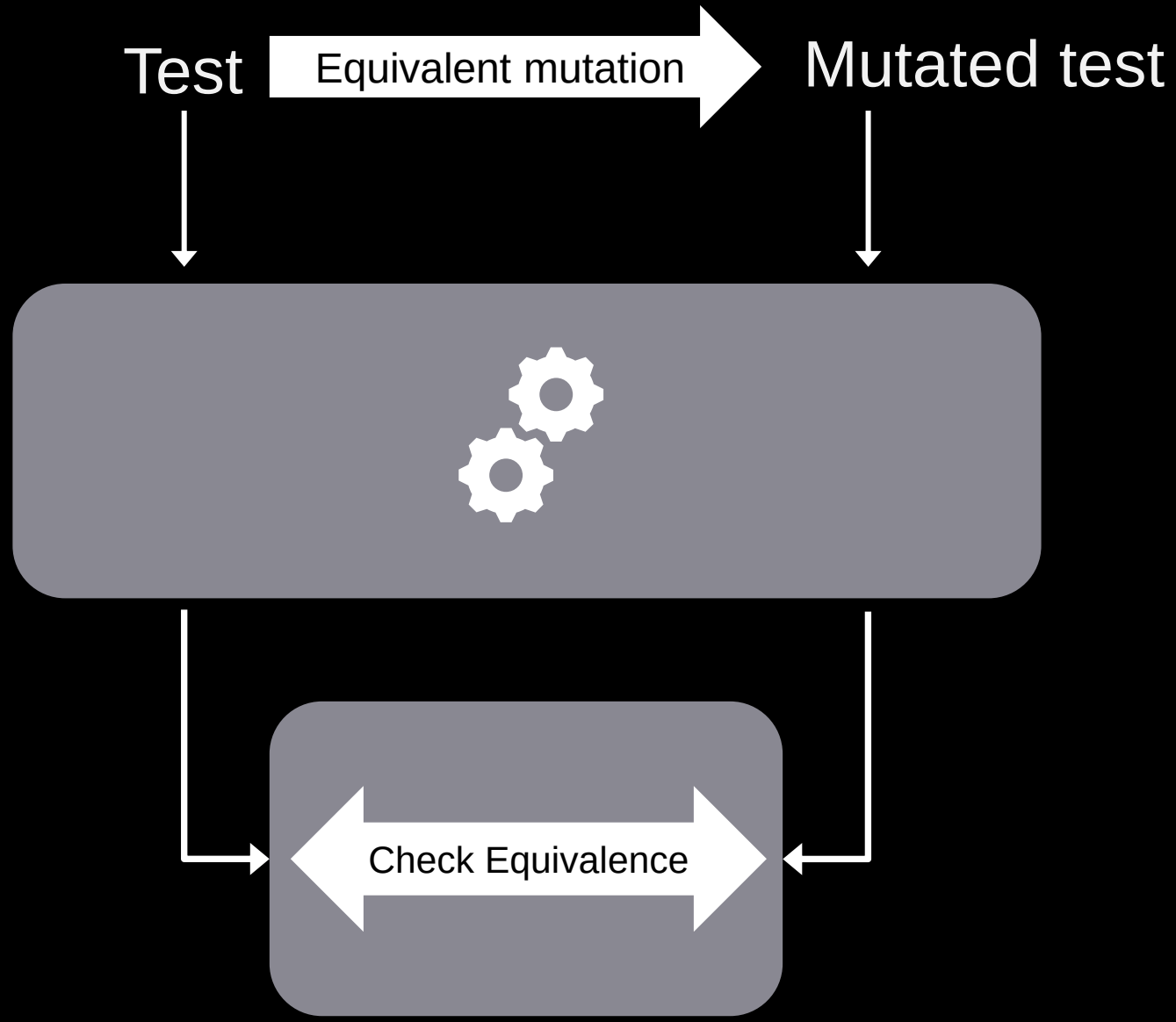
Testing differentially



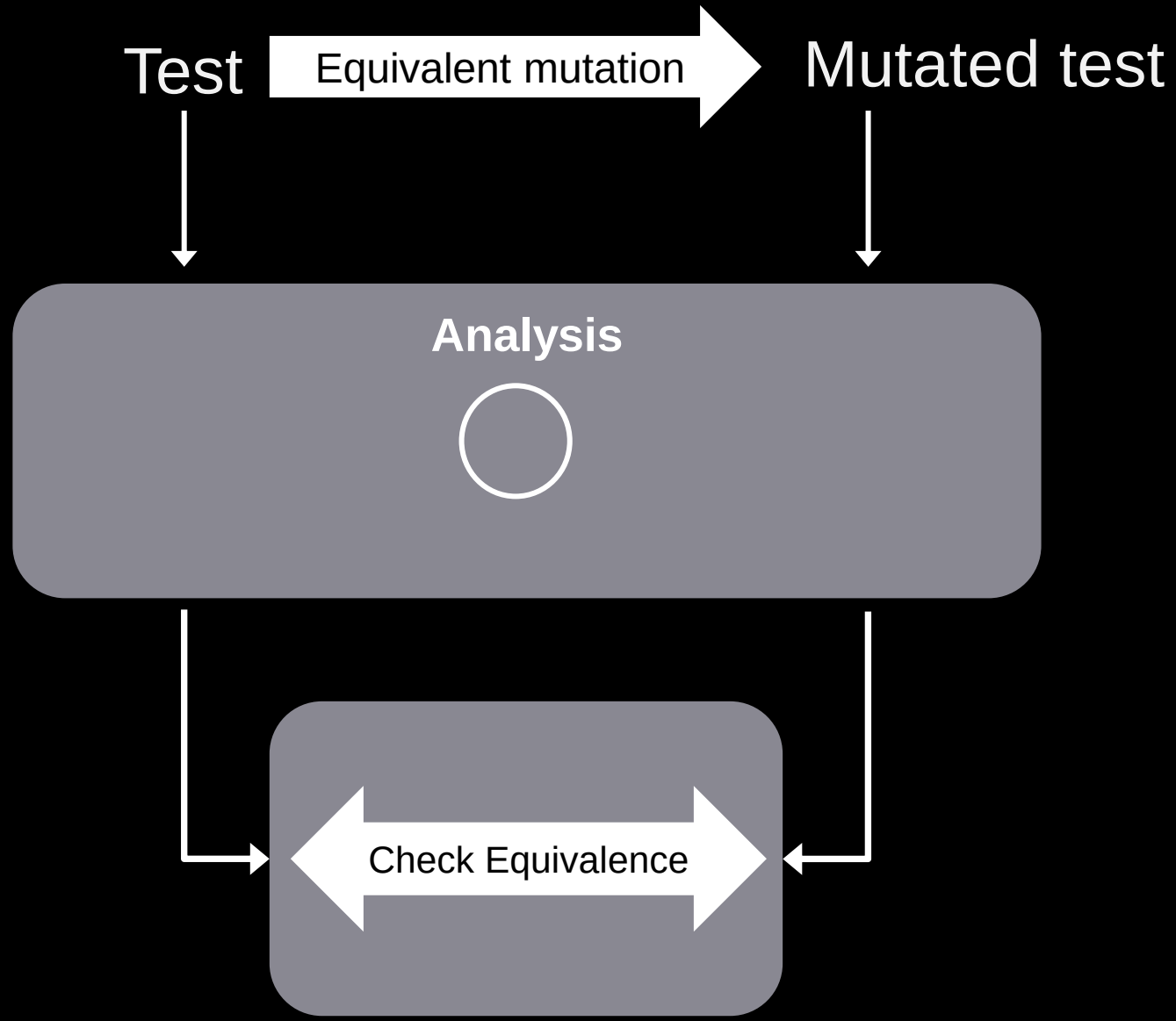
Testing differentially



Testing differentially

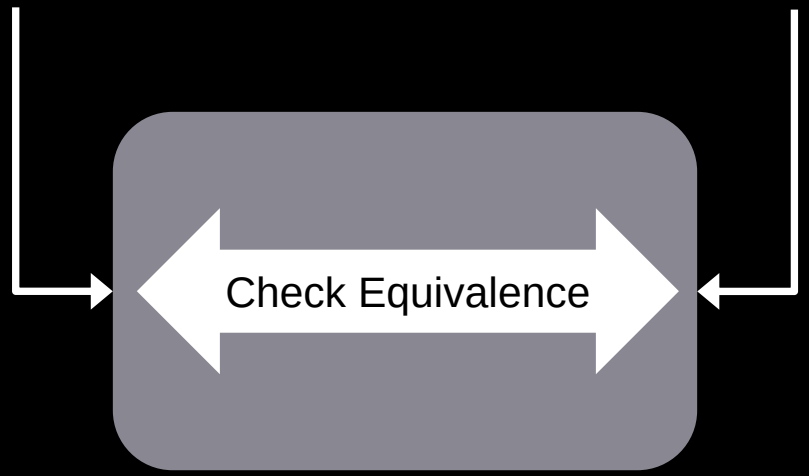


Testing analysis differentially



Checking equivalence of analyses result

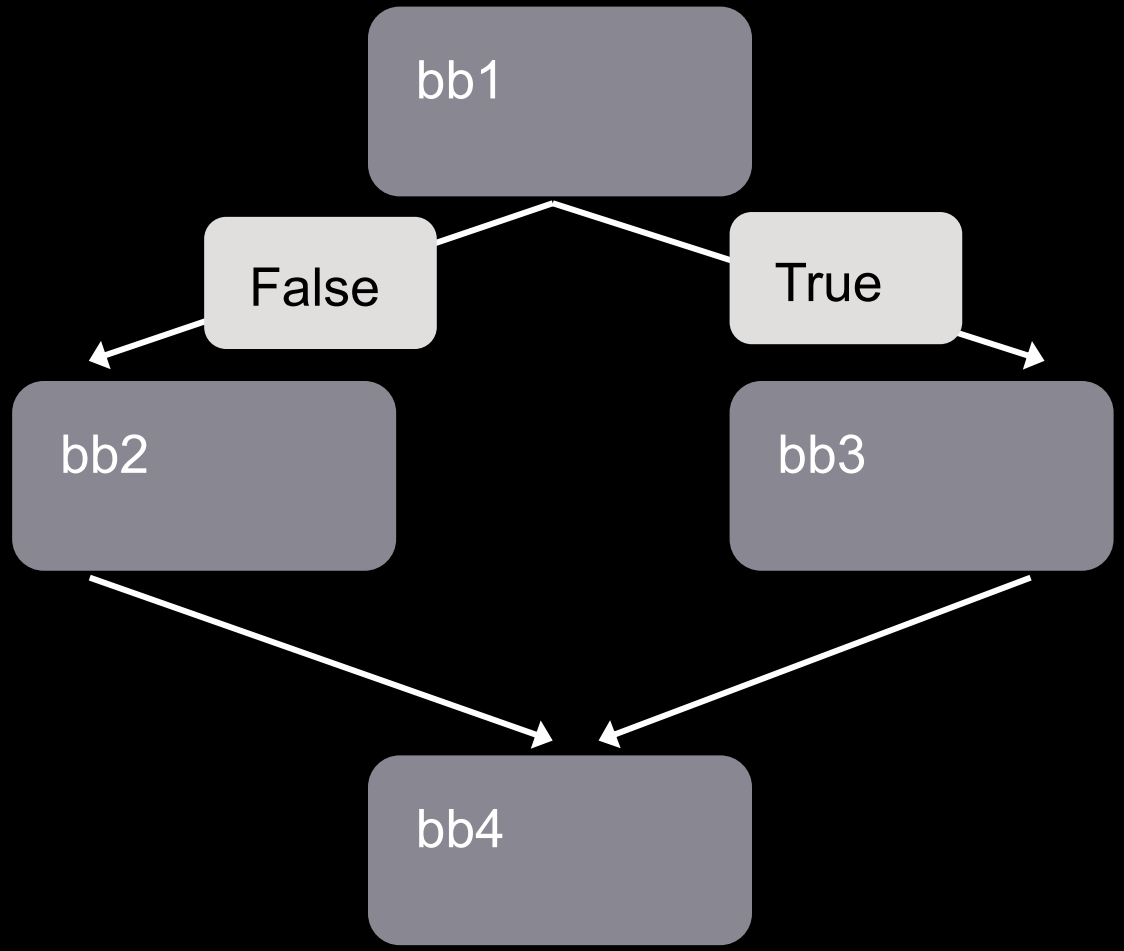
It is not straight forward, and a notion of equivalence needs to be established for each analysis result



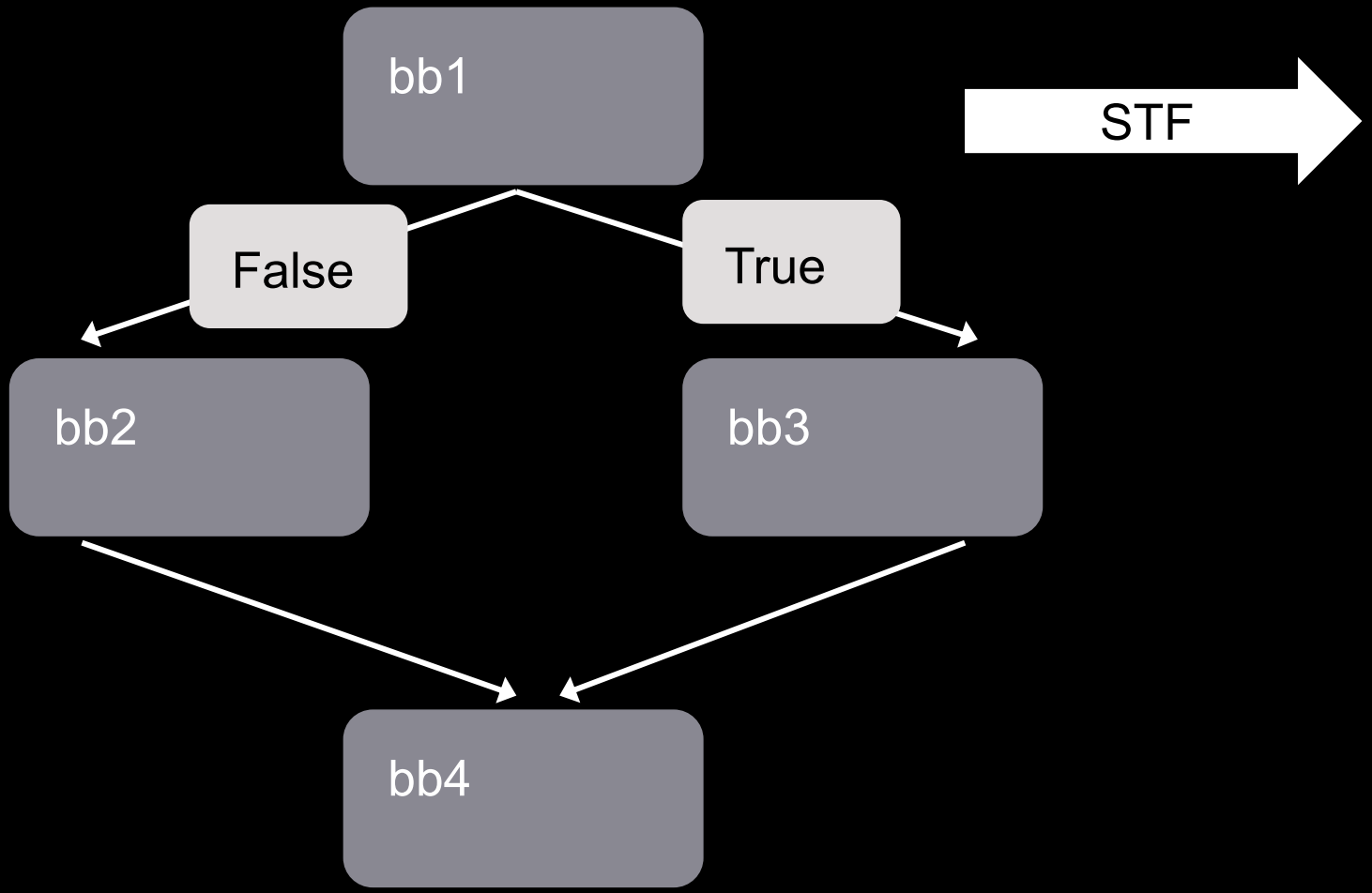
Case study: SwapTrueFalse (STF)

- STF is a structural mutation
- It swaps true and false branches

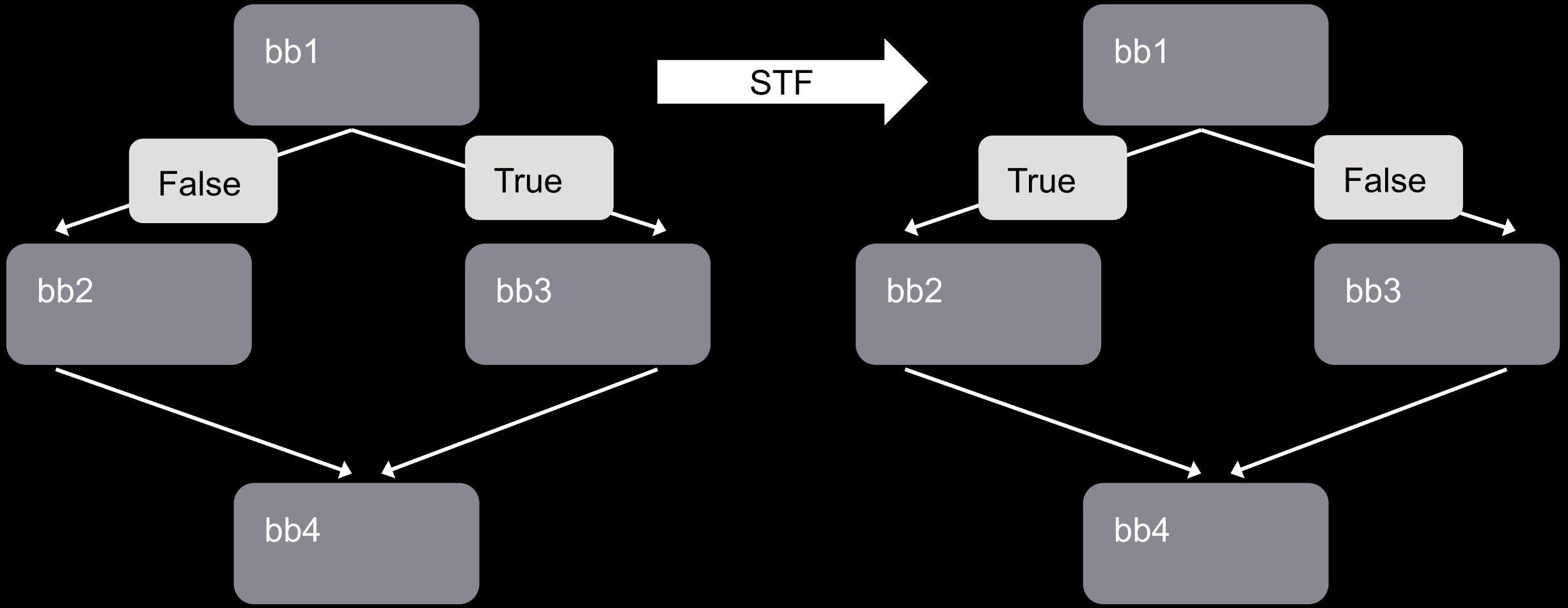
Case study: SwapTrueFalse (STF)



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

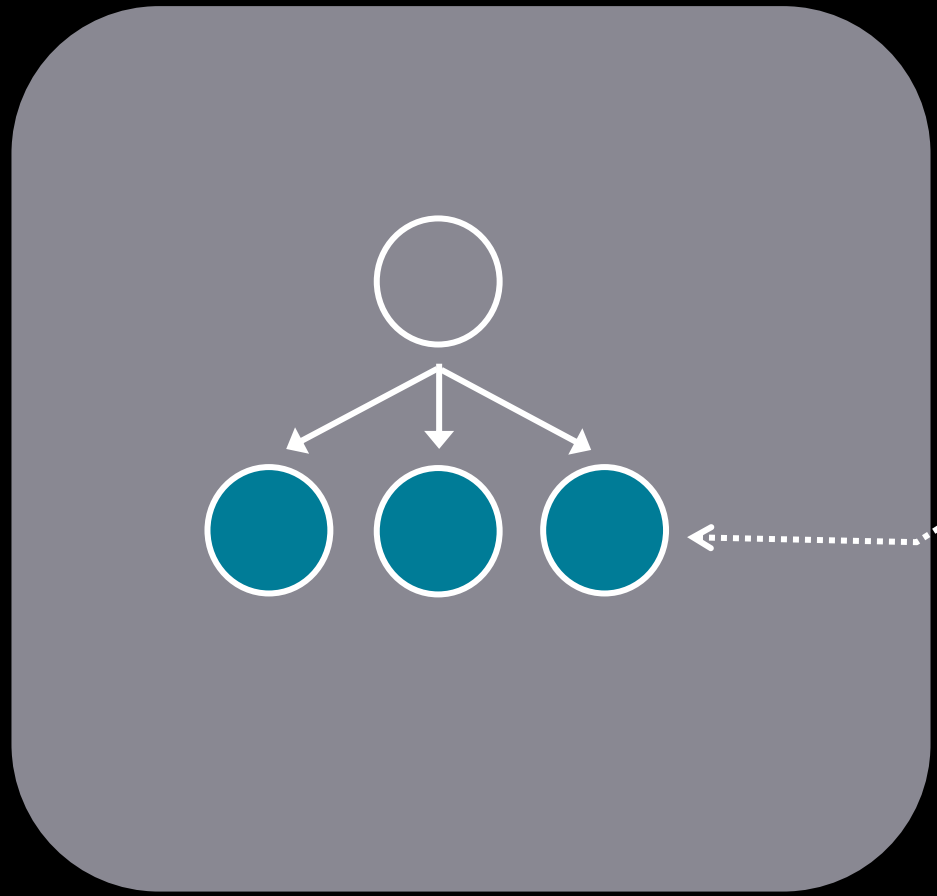
STF does not change the structure of the control-flow graph so can be used for analysis that are independent of CFG

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



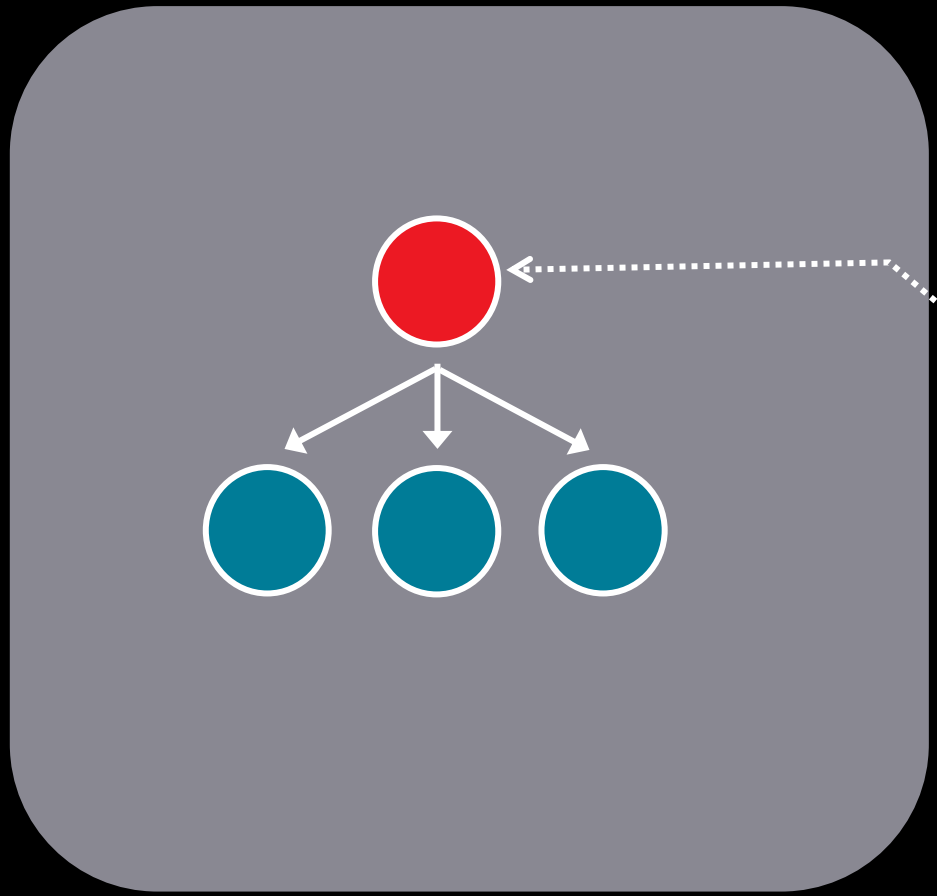
These analyses do not depend on any other analysis

Source of bug

The bug in any derived analysis can be caused by itself

Source of bug

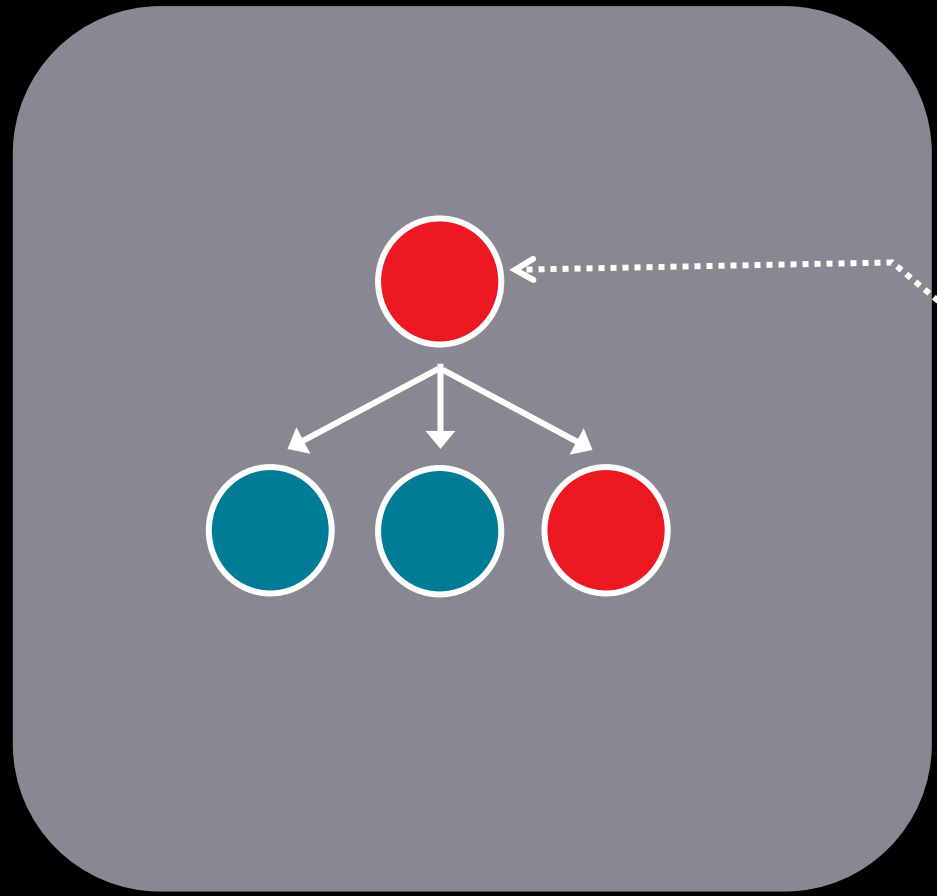
The bug in any derived analysis can be caused by itself



This analysis depends on the results from other analyses

Source of bug

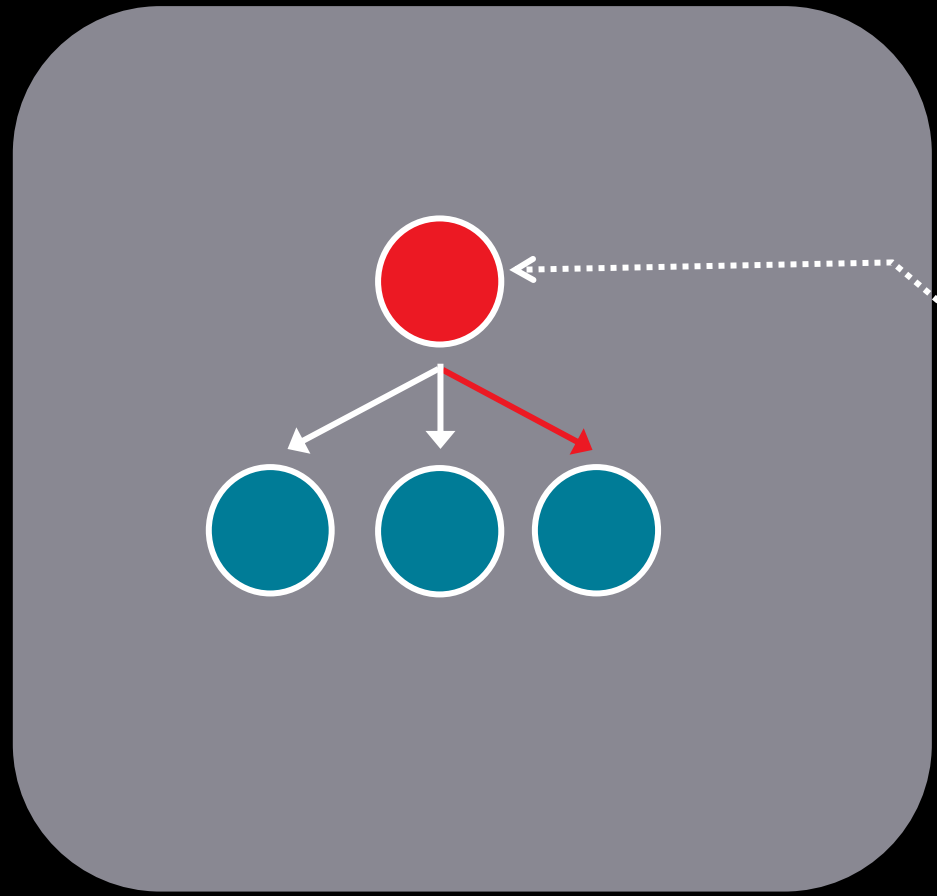
The bug in any derived analysis can be caused by any dependent analysis



This analysis depends on the results from other analyses

Source of bug

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



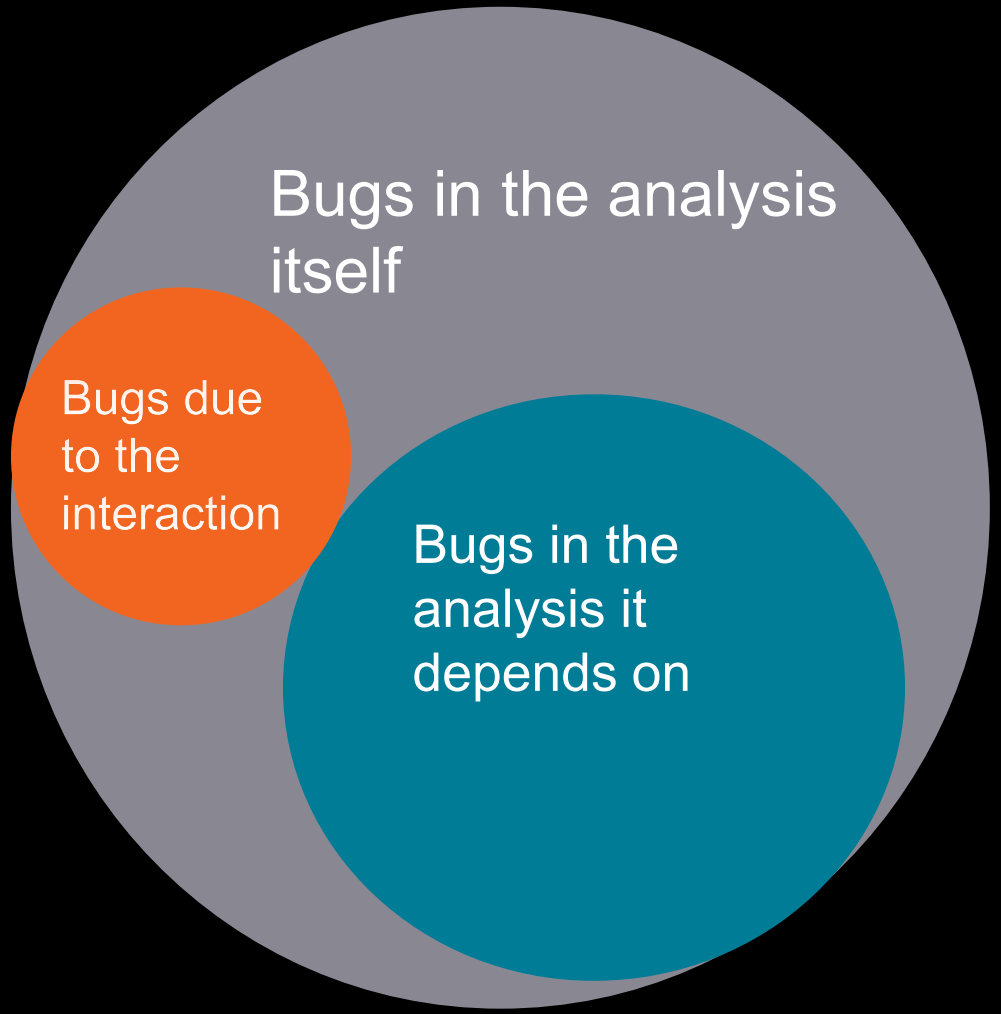
This analysis depends on the results from other analyses

Source of bug

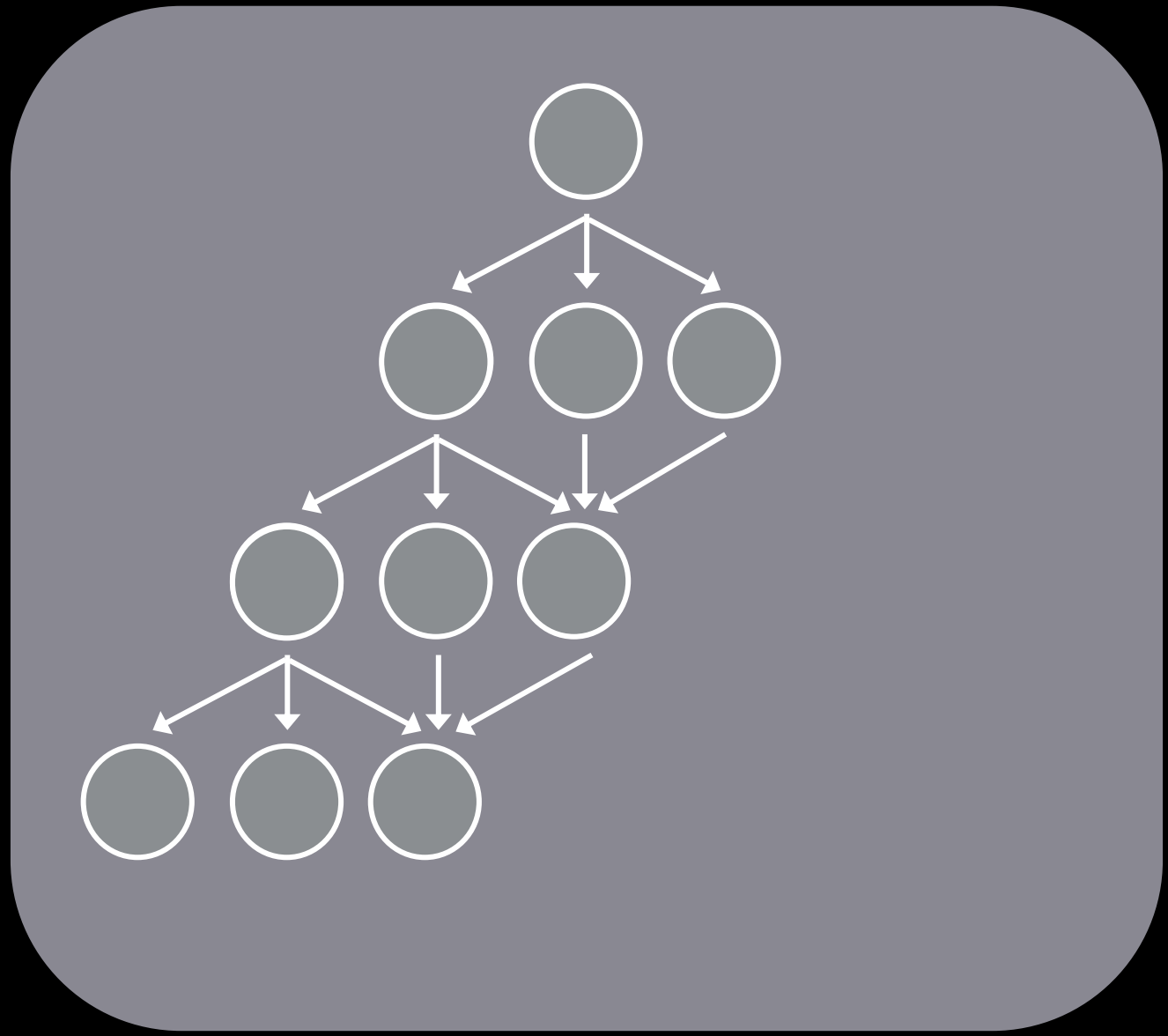
The bug in any derived analysis can be caused by the following,

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

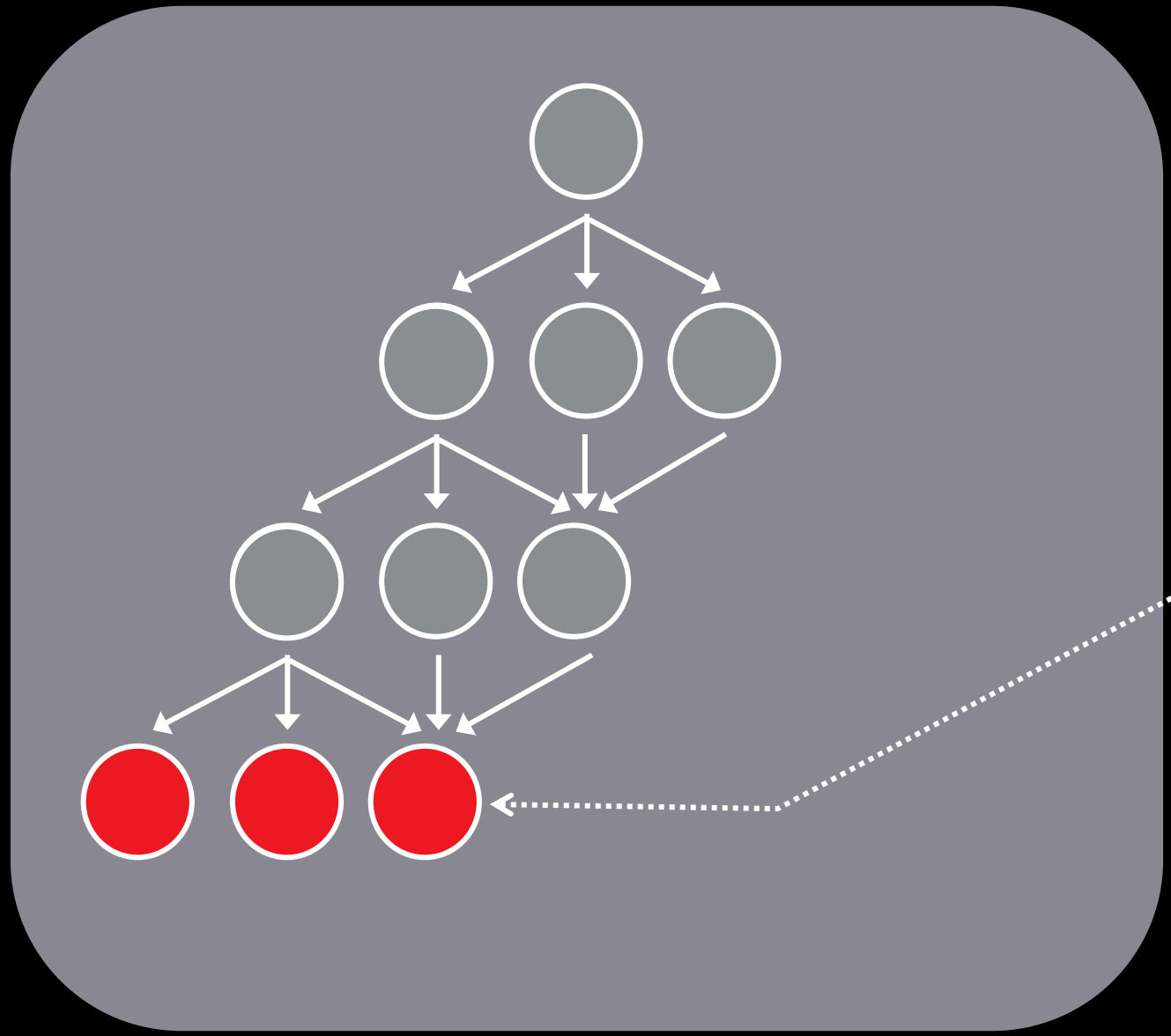
Source of bug



Source of trust

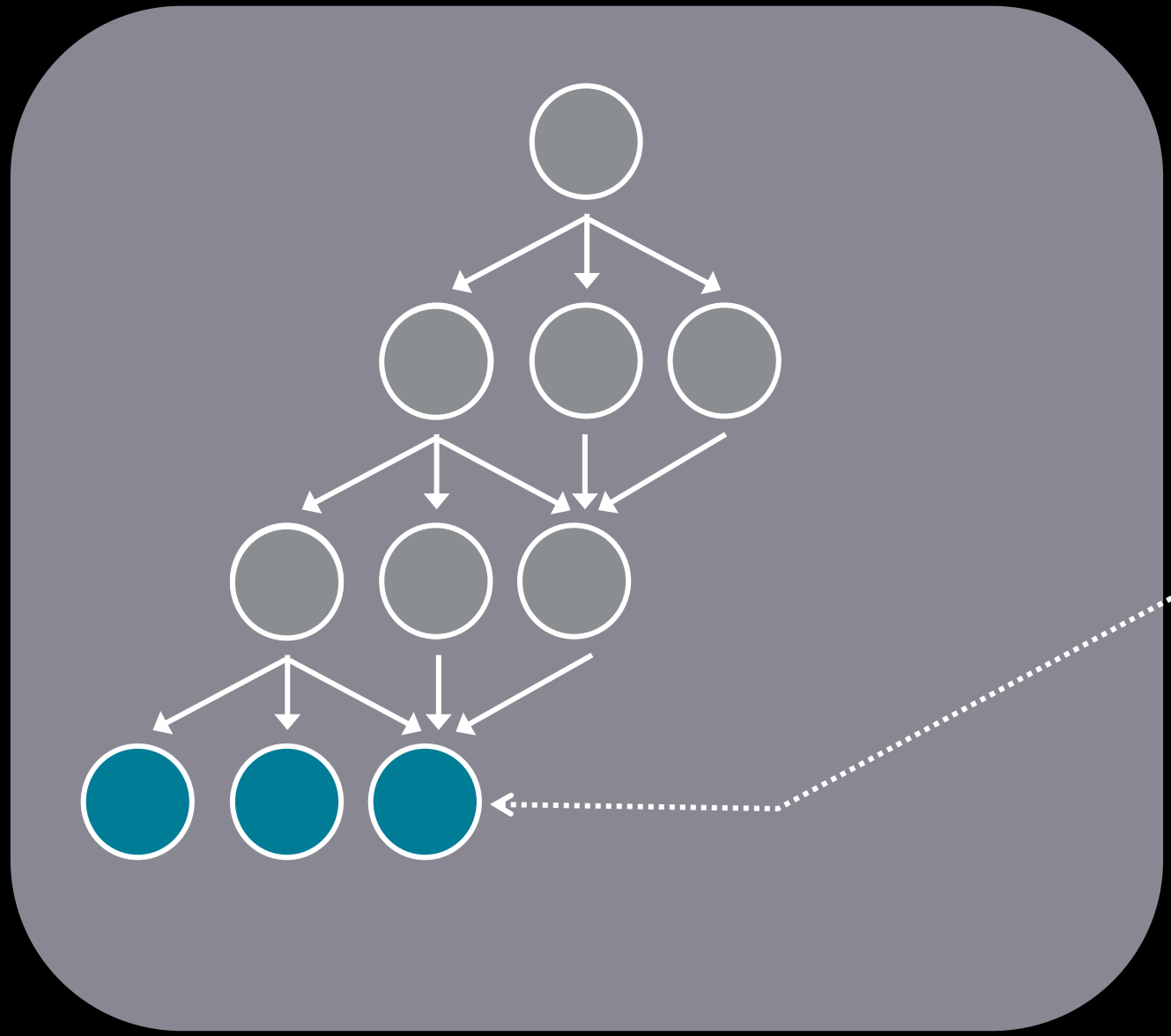


Source of trust



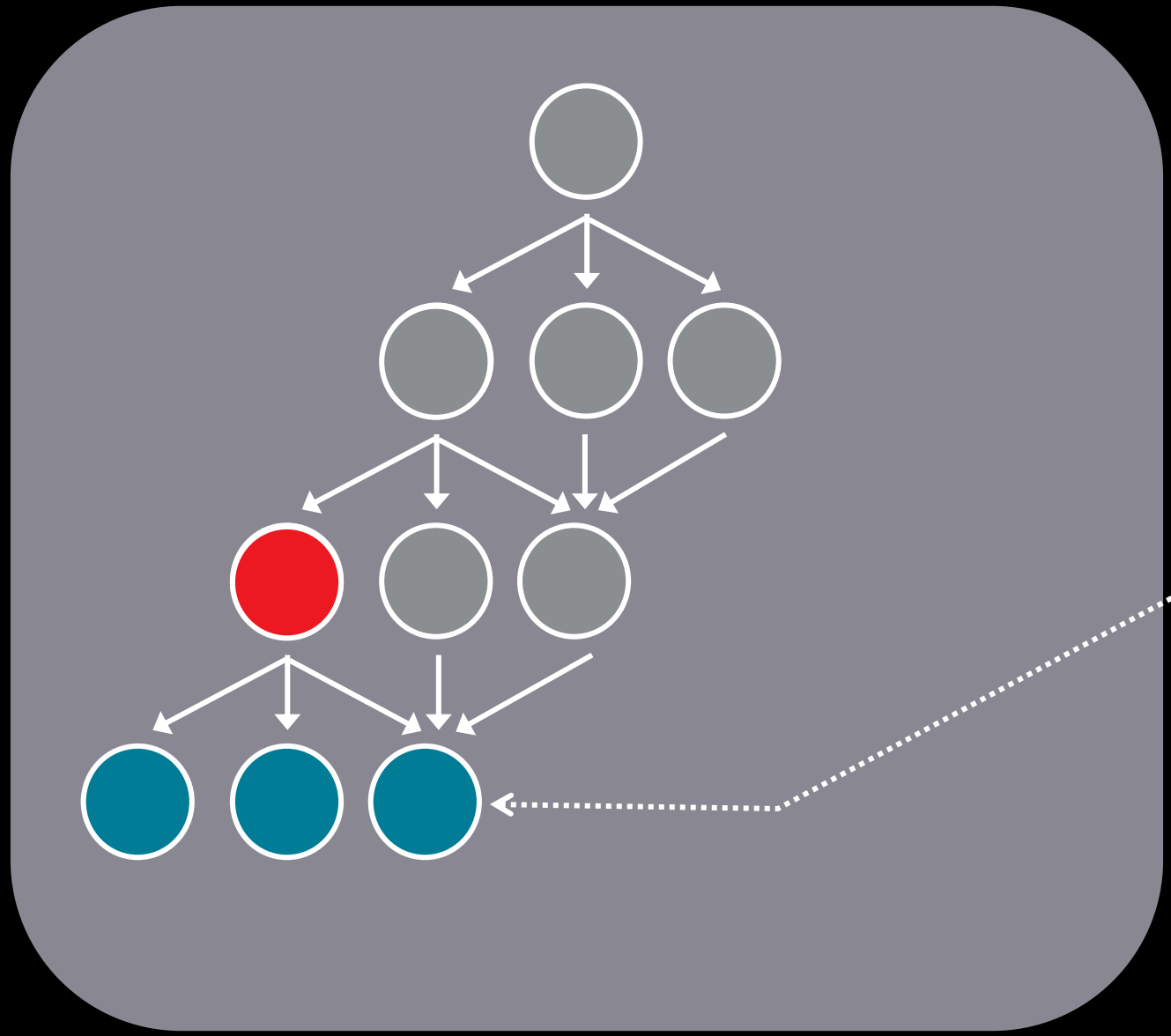
Fundamental analysis are the easiest to reason about

Source of trust



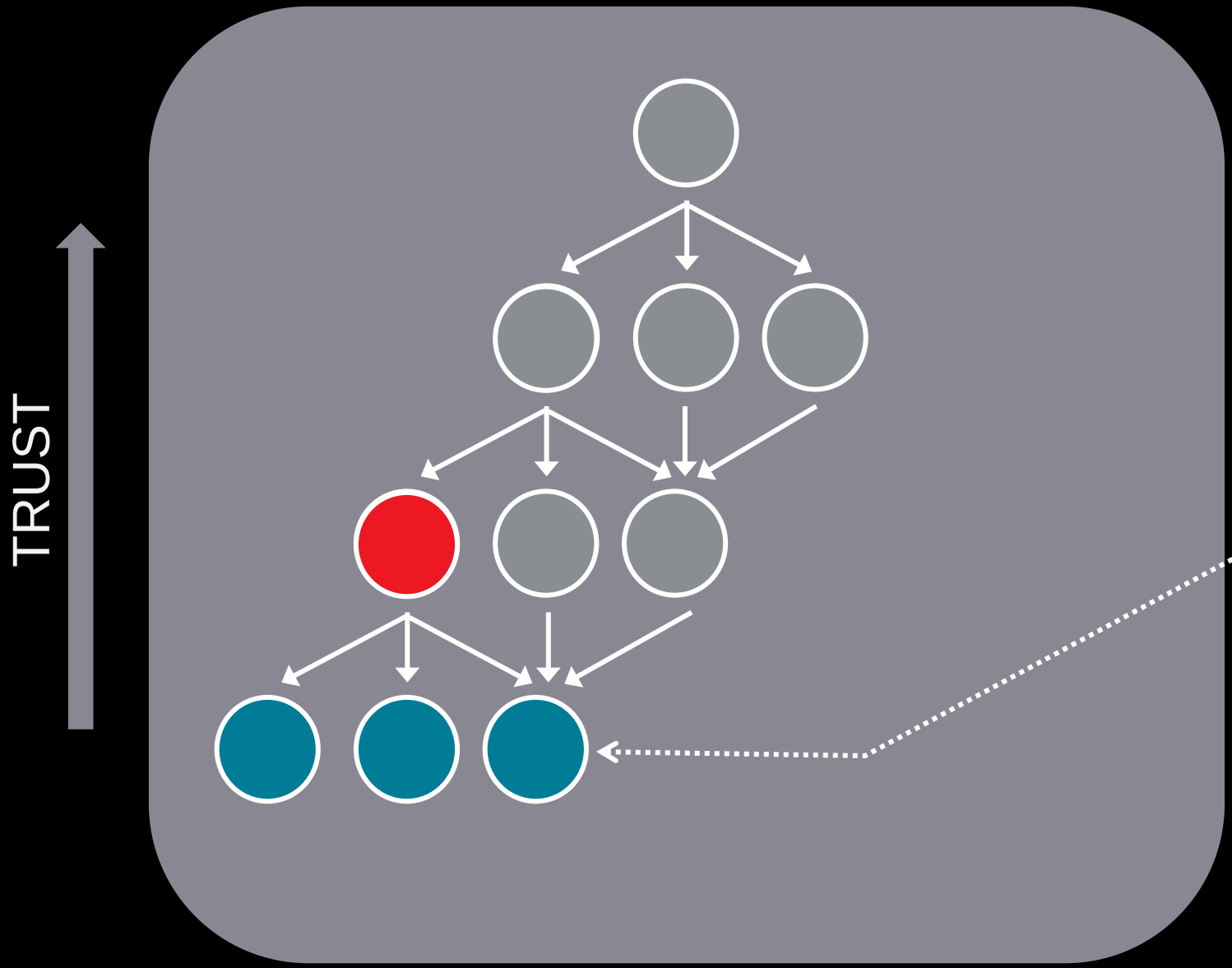
Fundamental analysis are the easiest to reason about

Source of trust



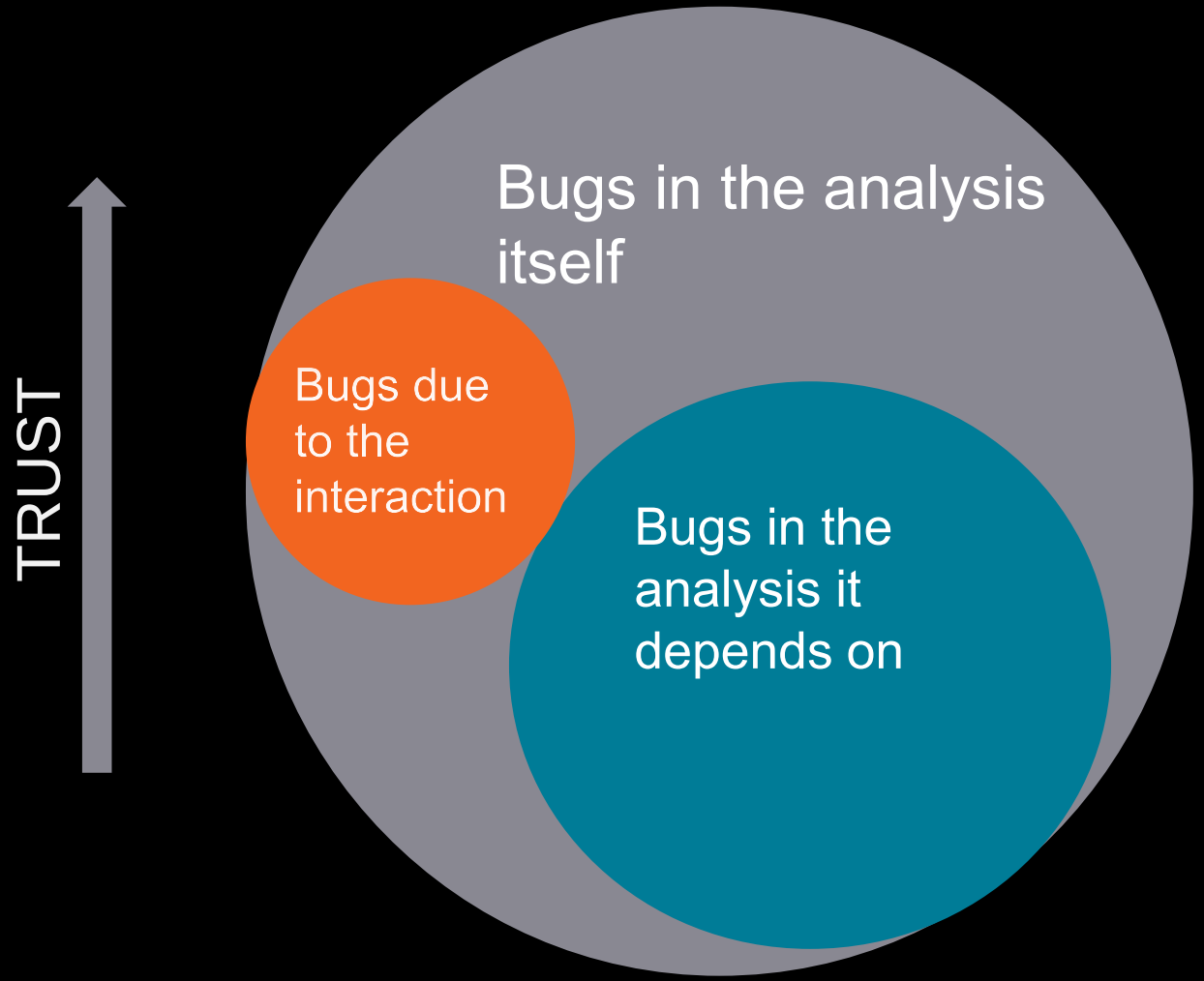
Fundamental analysis are the easiest to reason about

Source of trust

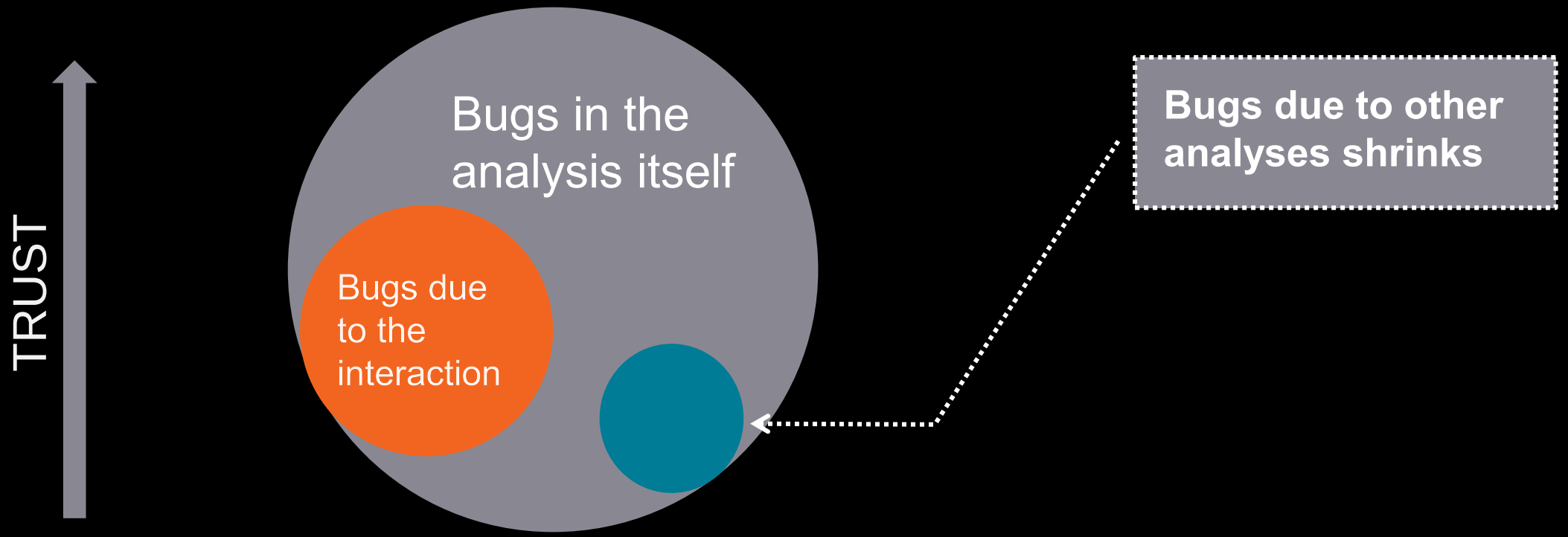


Fundamental analysis are the easiest to reason about

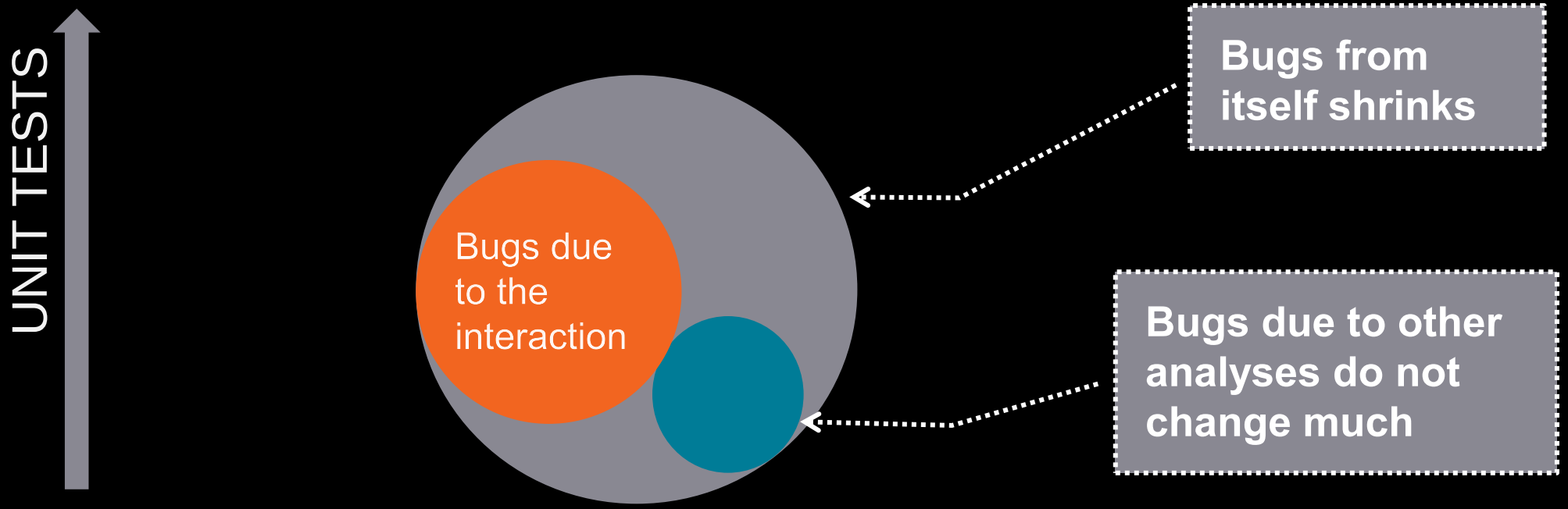
Source of trust



Source of trust



Source of trust

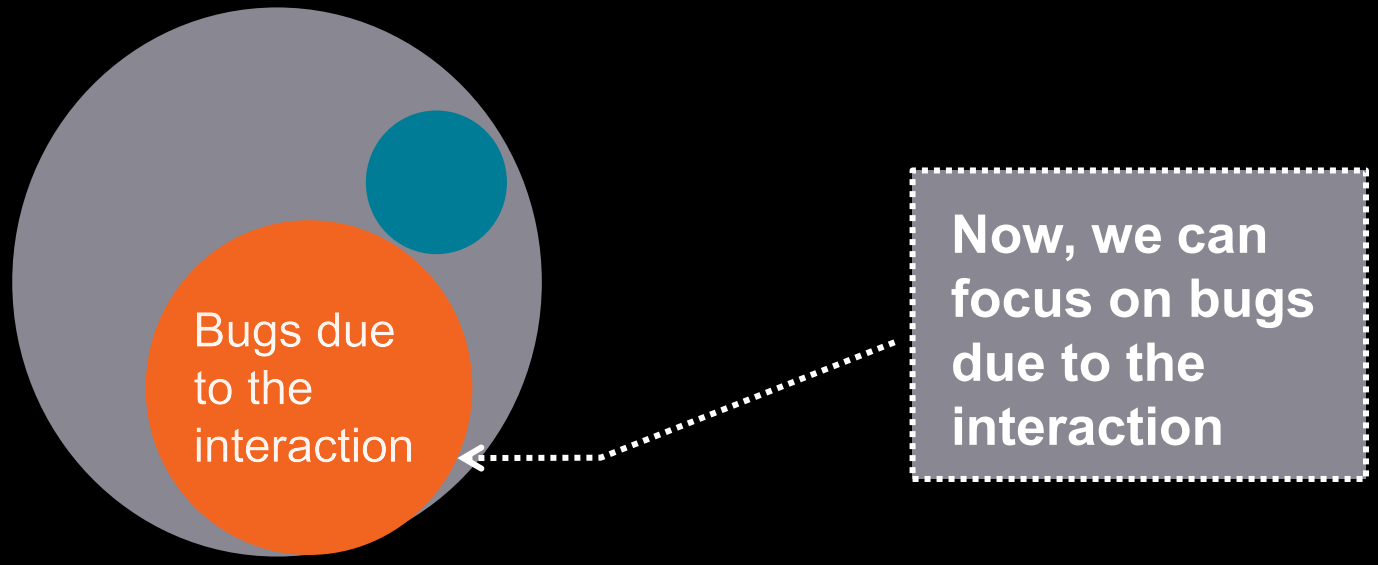


Source of trust

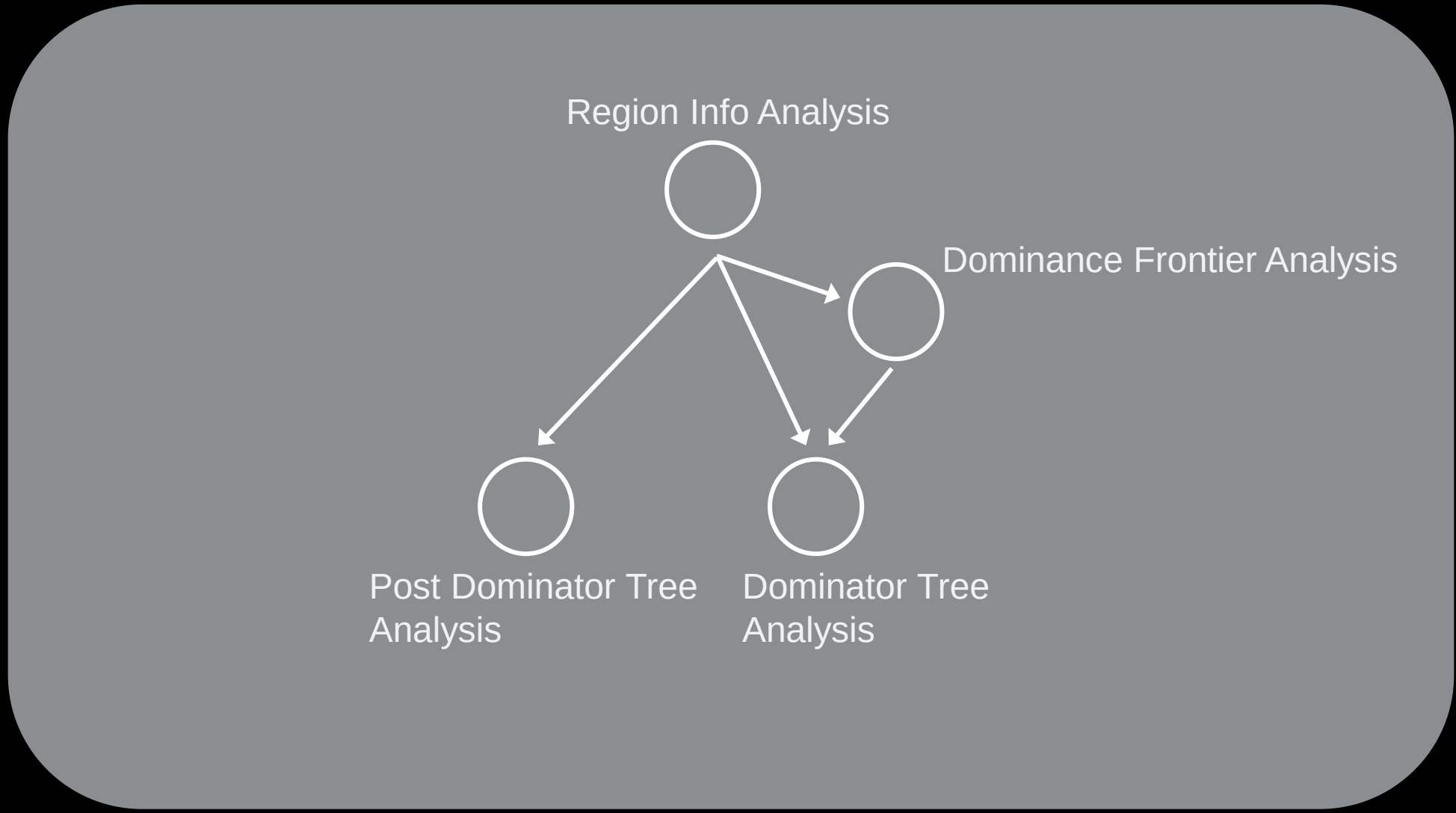
If we choose mutations which are inert to the analysis in question and the analyses it uses

Source of trust

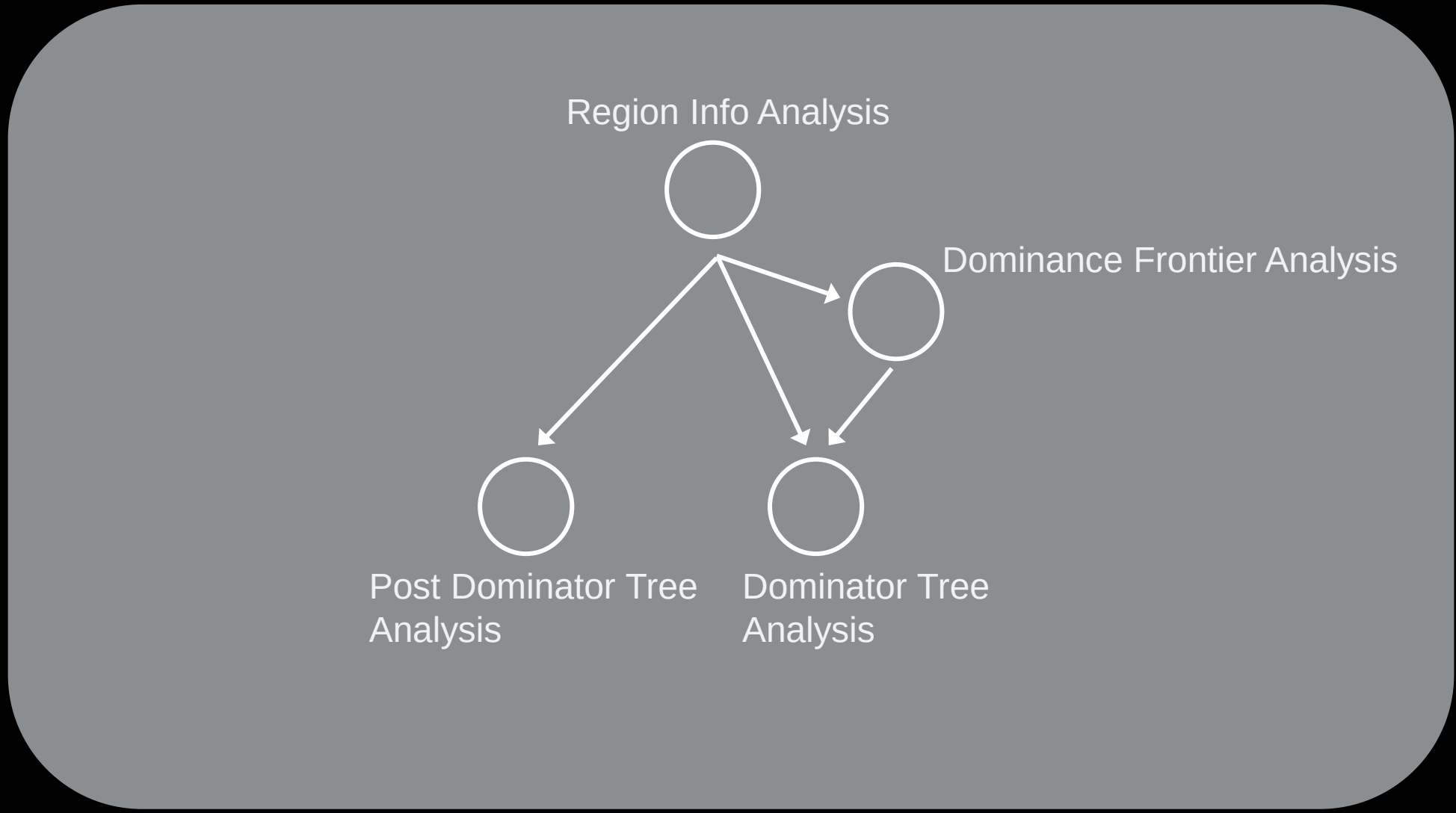
If we choose mutations which are inert to the analysis in question and the analyses it uses



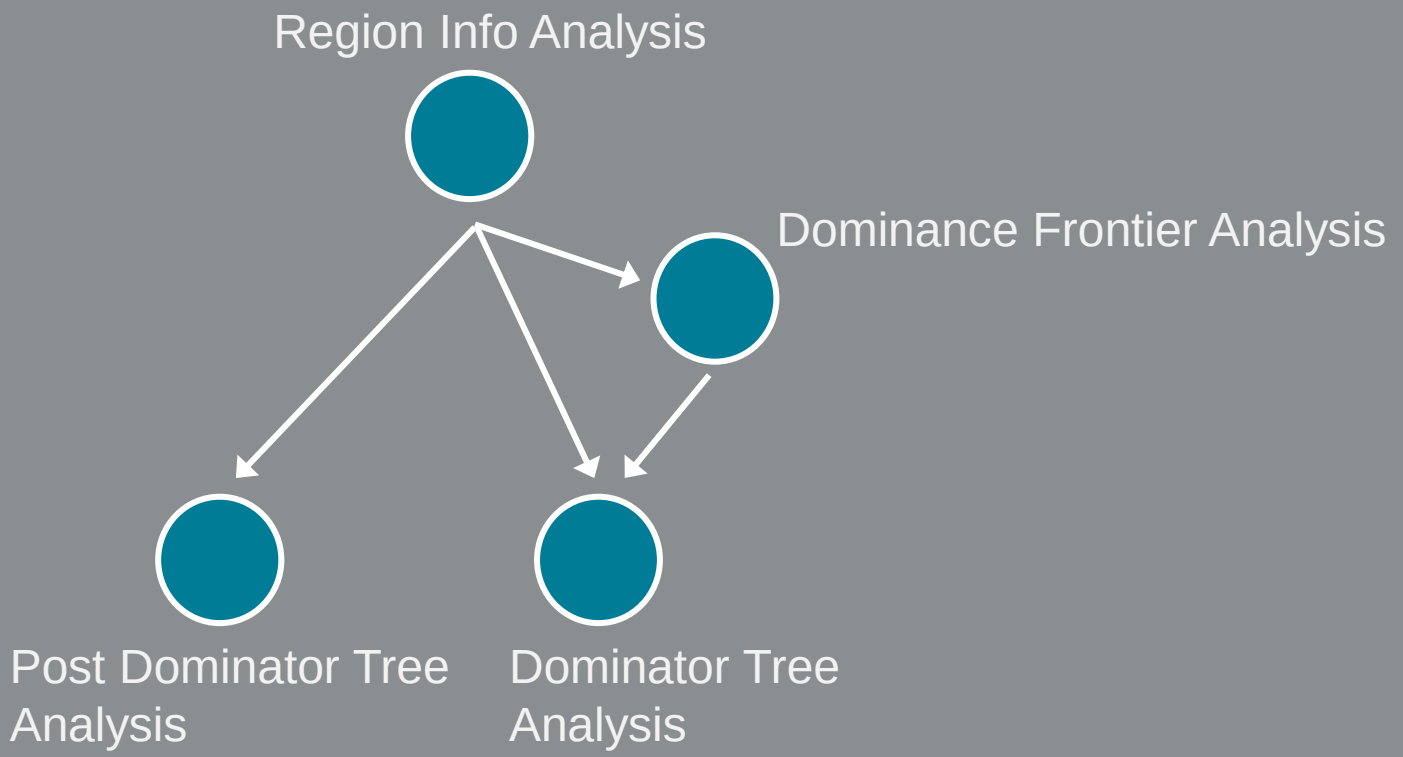
Example: Region Info Analysis



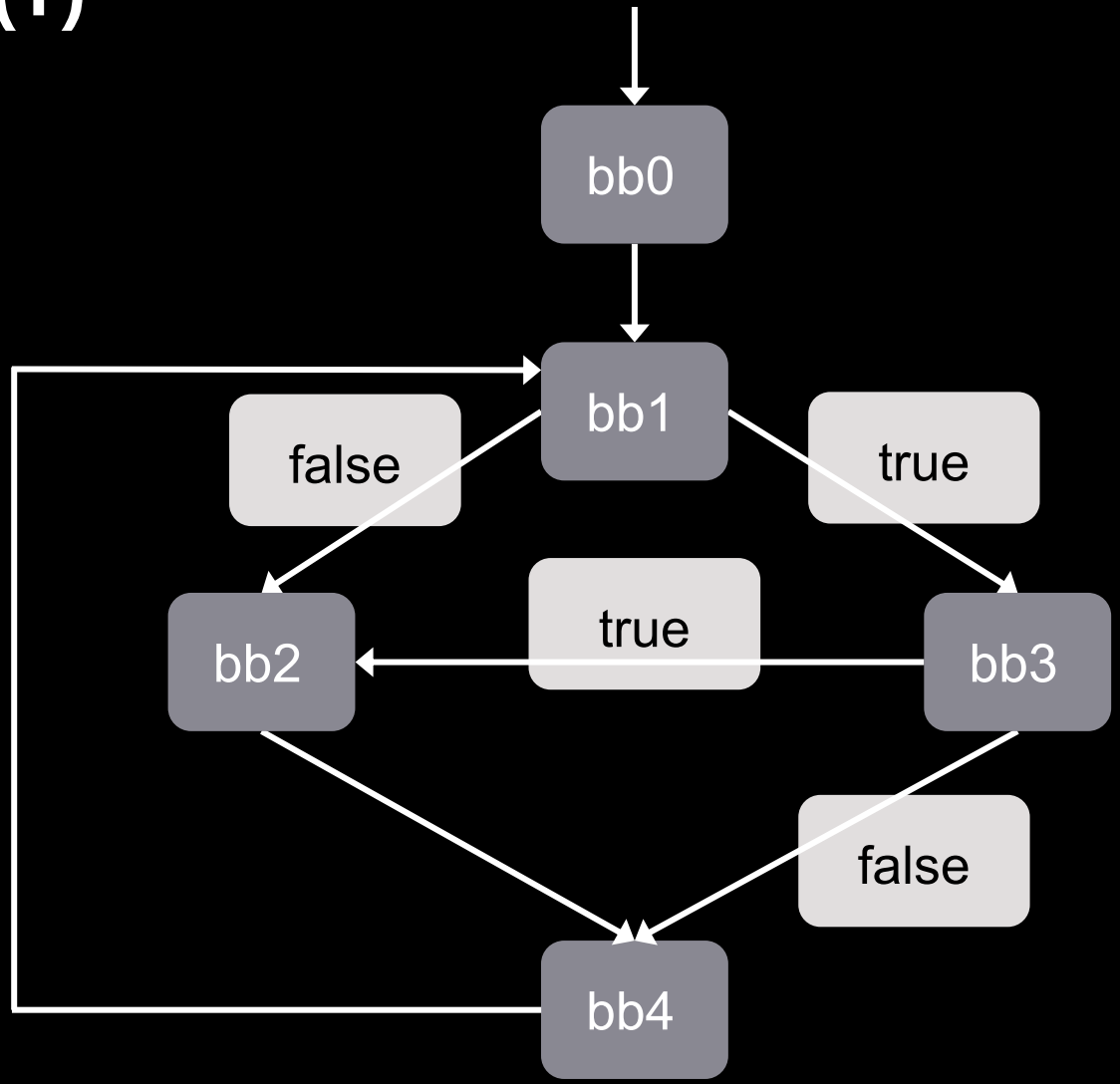
Mutation: SwapTrueFalse (STF)



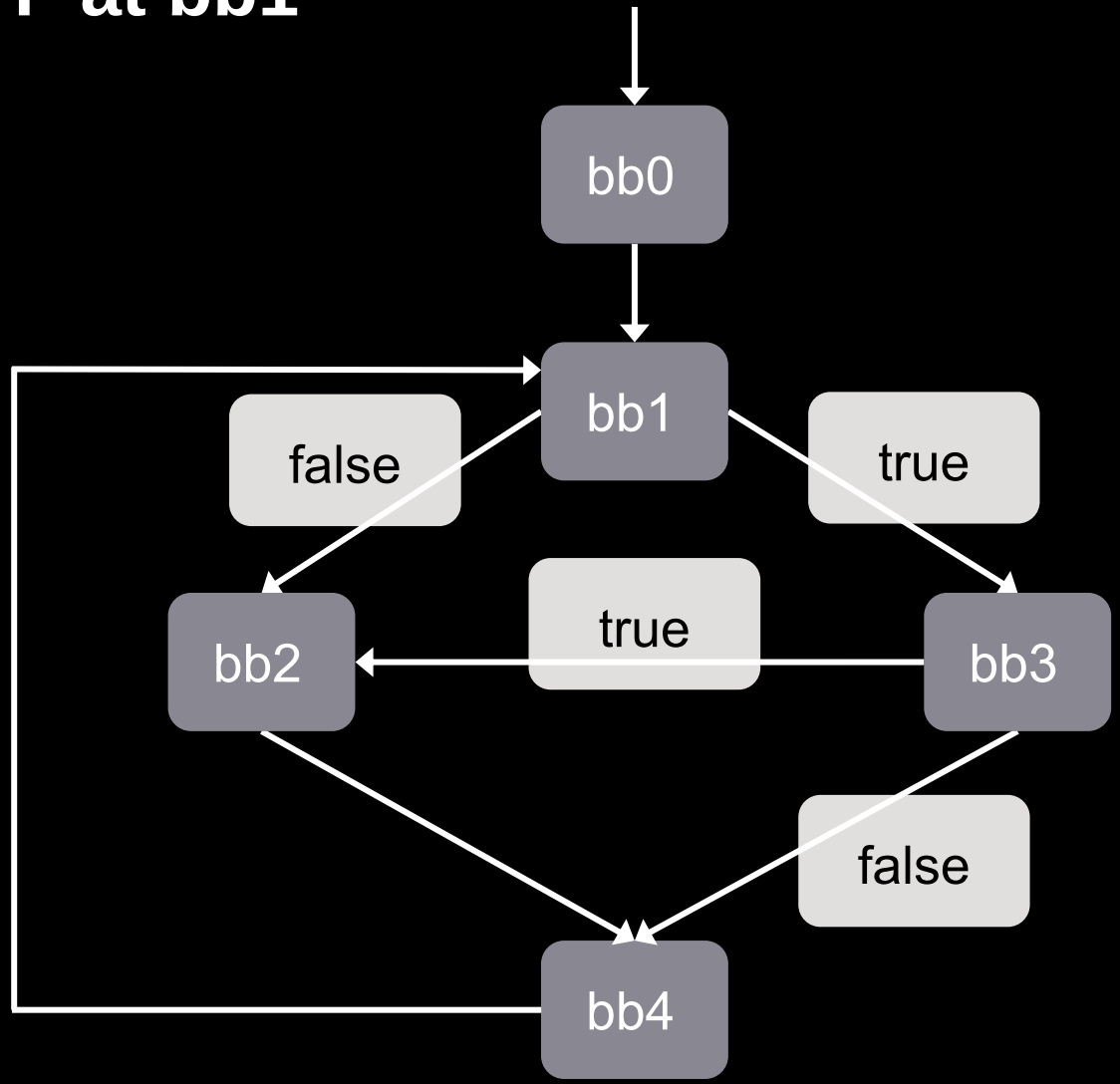
Mutation: SwapTrueFalse (STF)



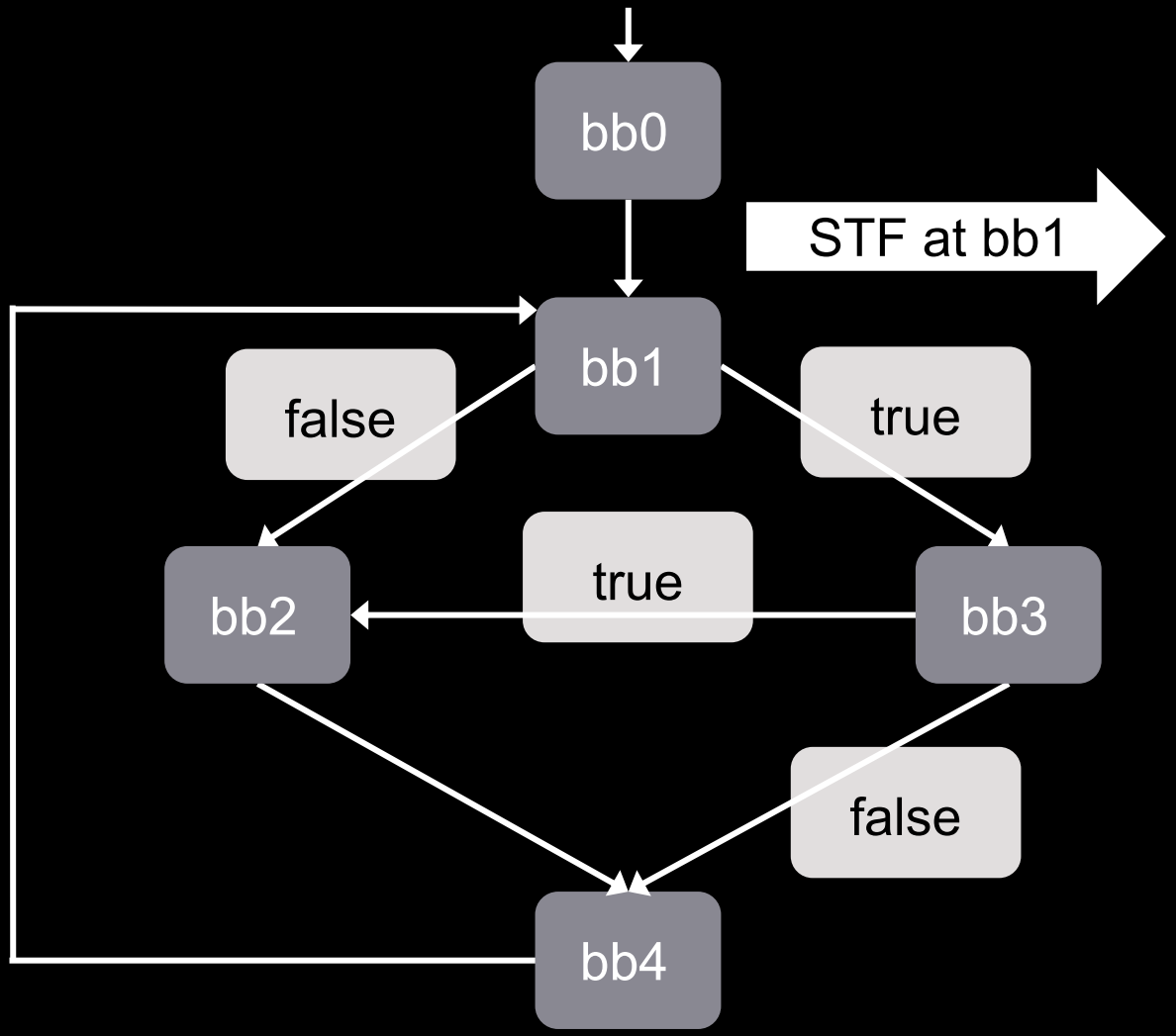
A test case (T)



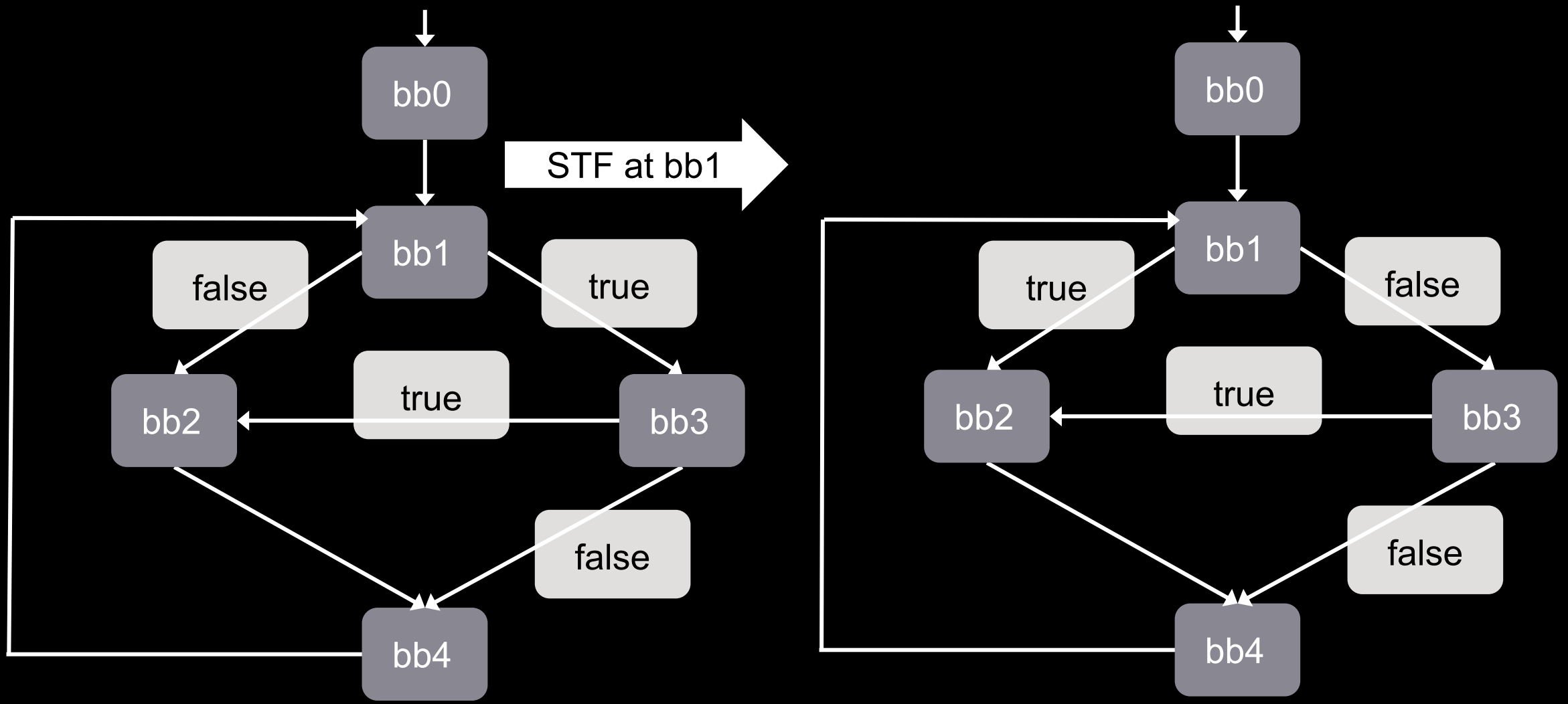
Applying STF at bb1



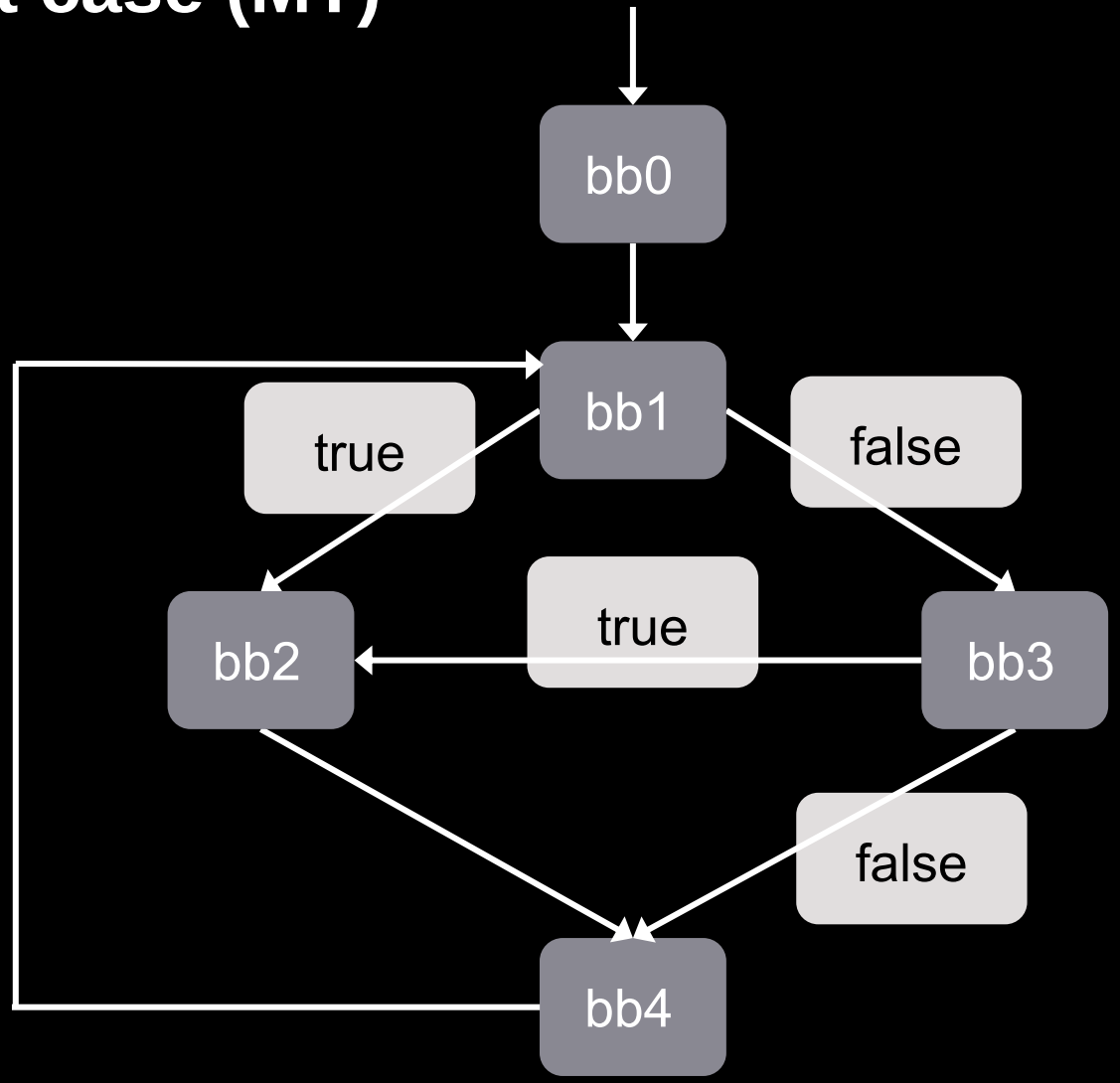
Applying STF at bb1



Applying STF at bb1



Mutated test case (MT)

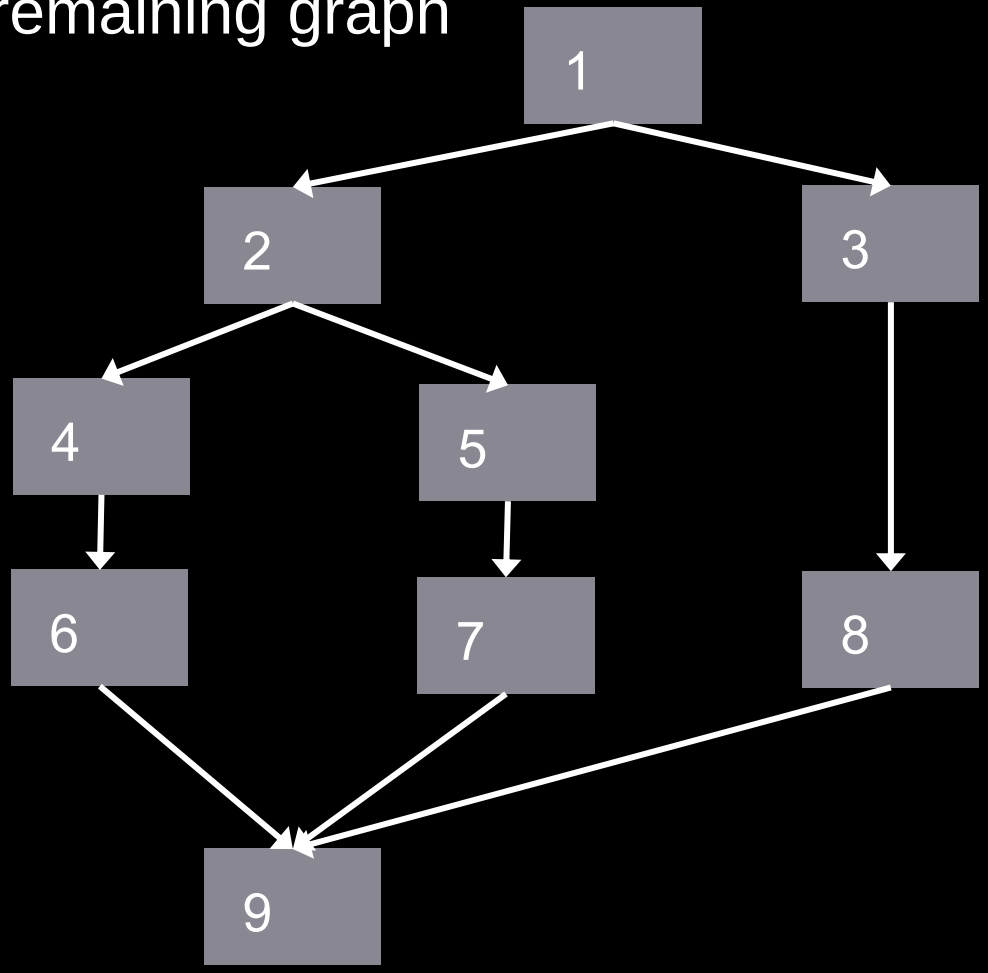


Region Info Analysis

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

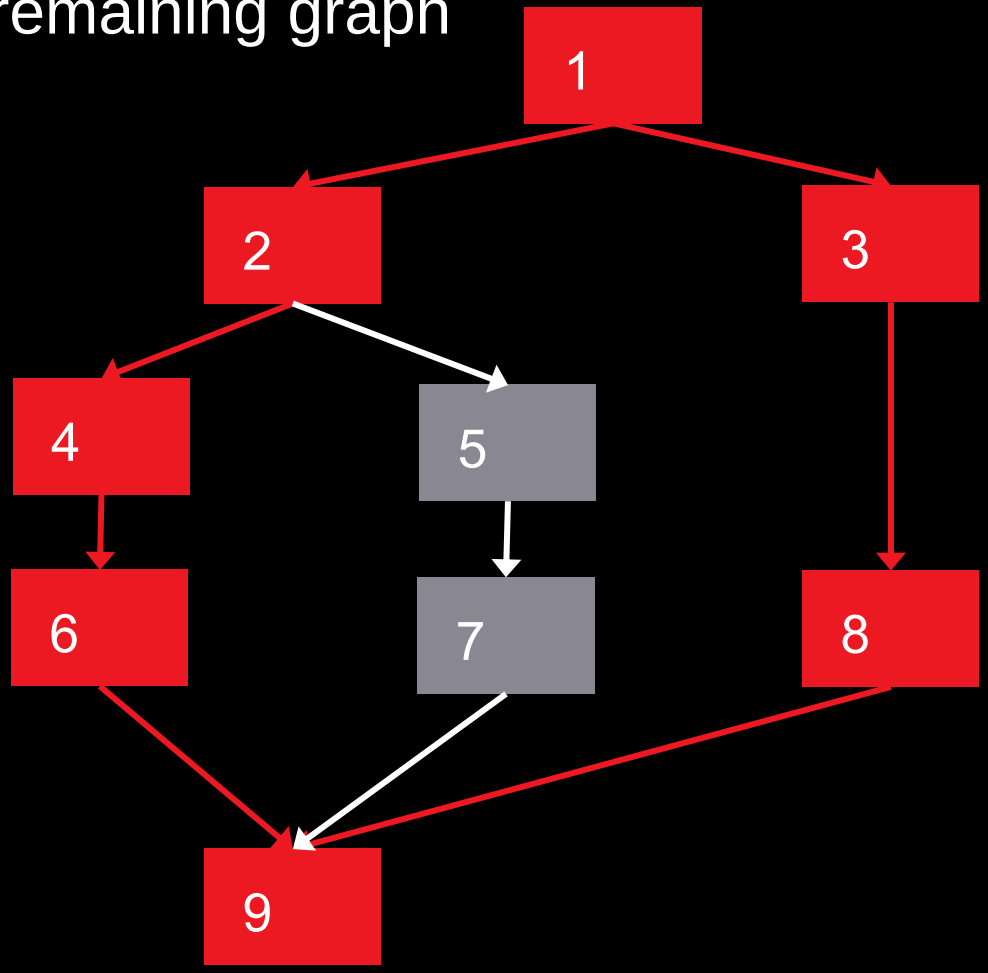
Region Info Analysis

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



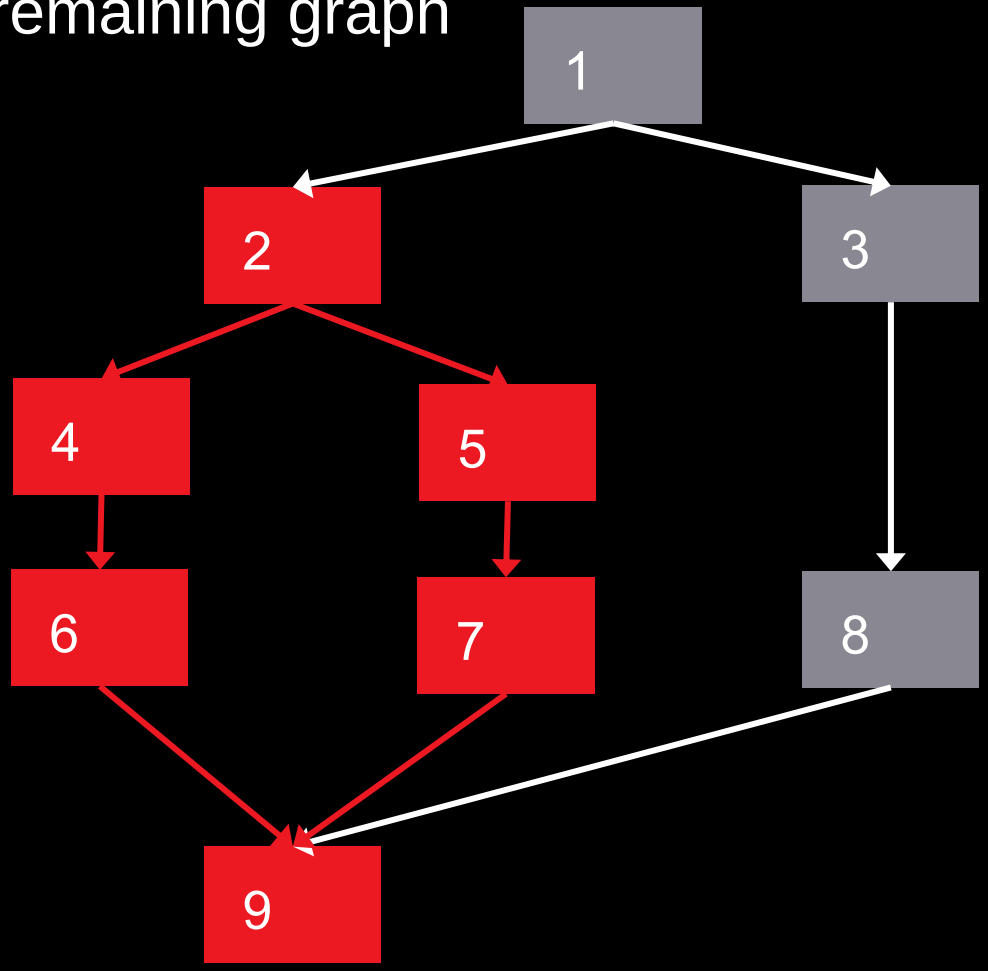
Region Info Analysis

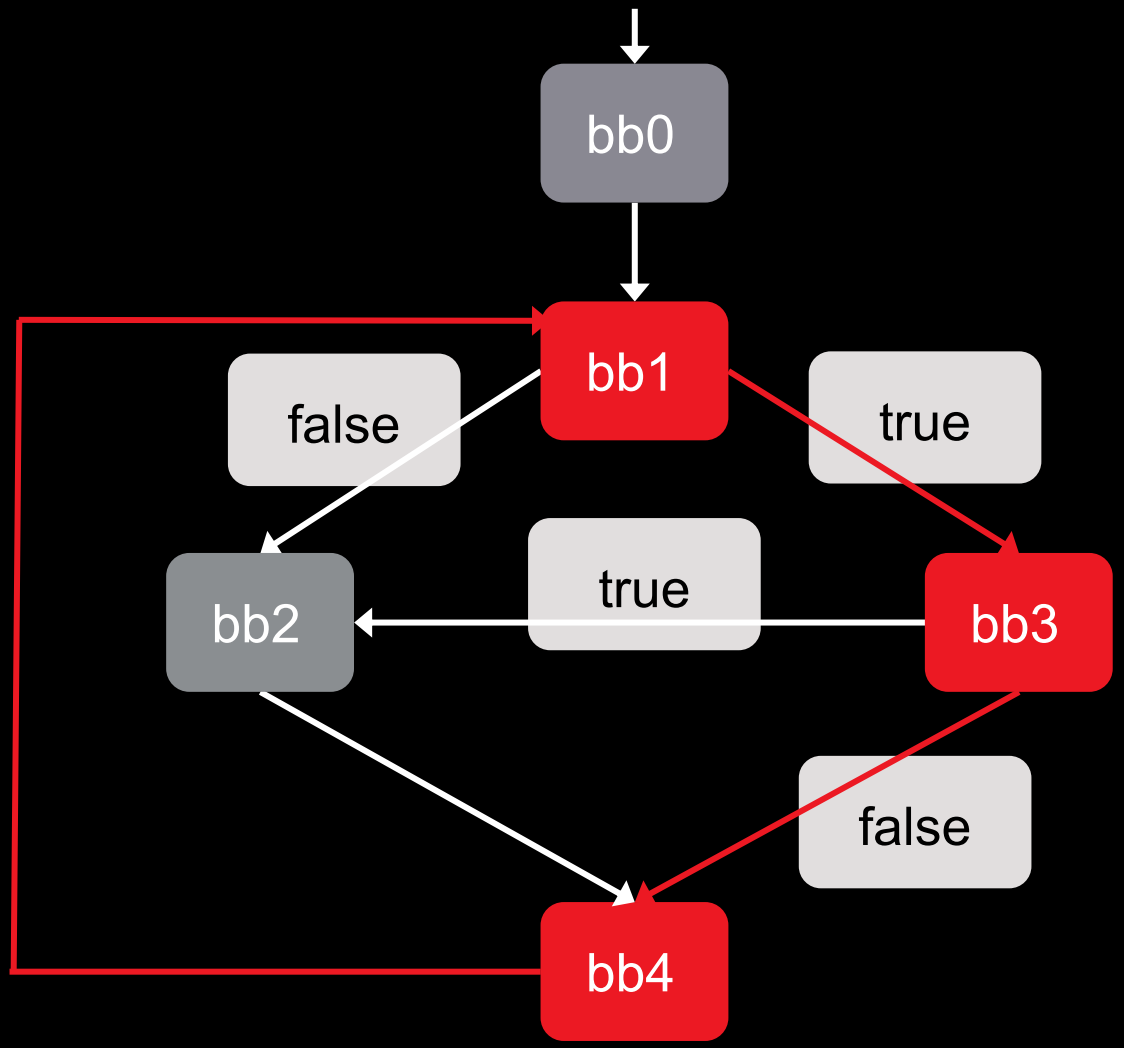
A Region is a connected subgraph of a control flow graph that has exactly two connections to the remaining graph

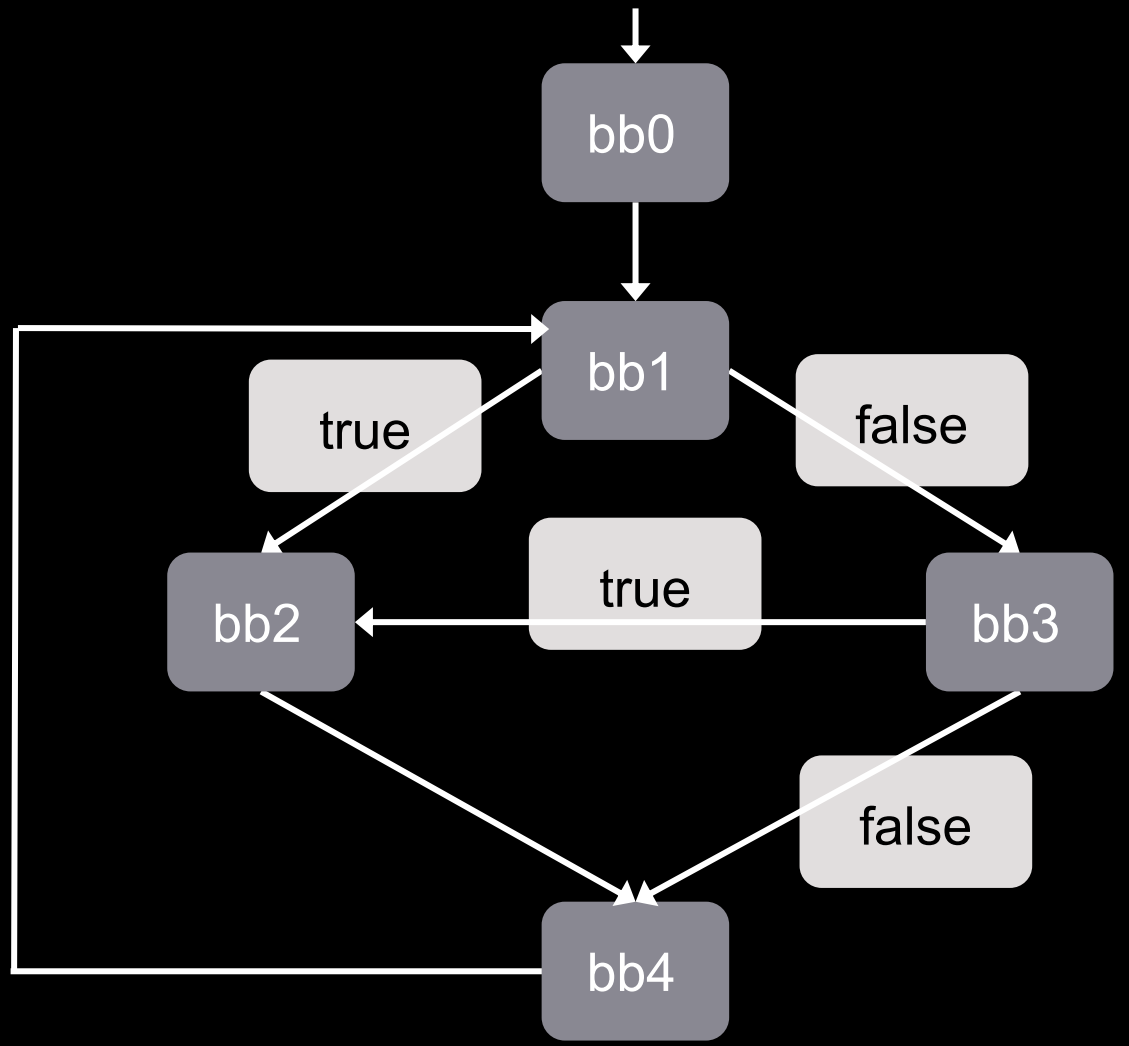


Region Info Analysis

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




bb0
bb1
false
true
bb2
true
bb3
false
bb4


bb0
bb1
true
false
bb2
true
bb3
false
bb4

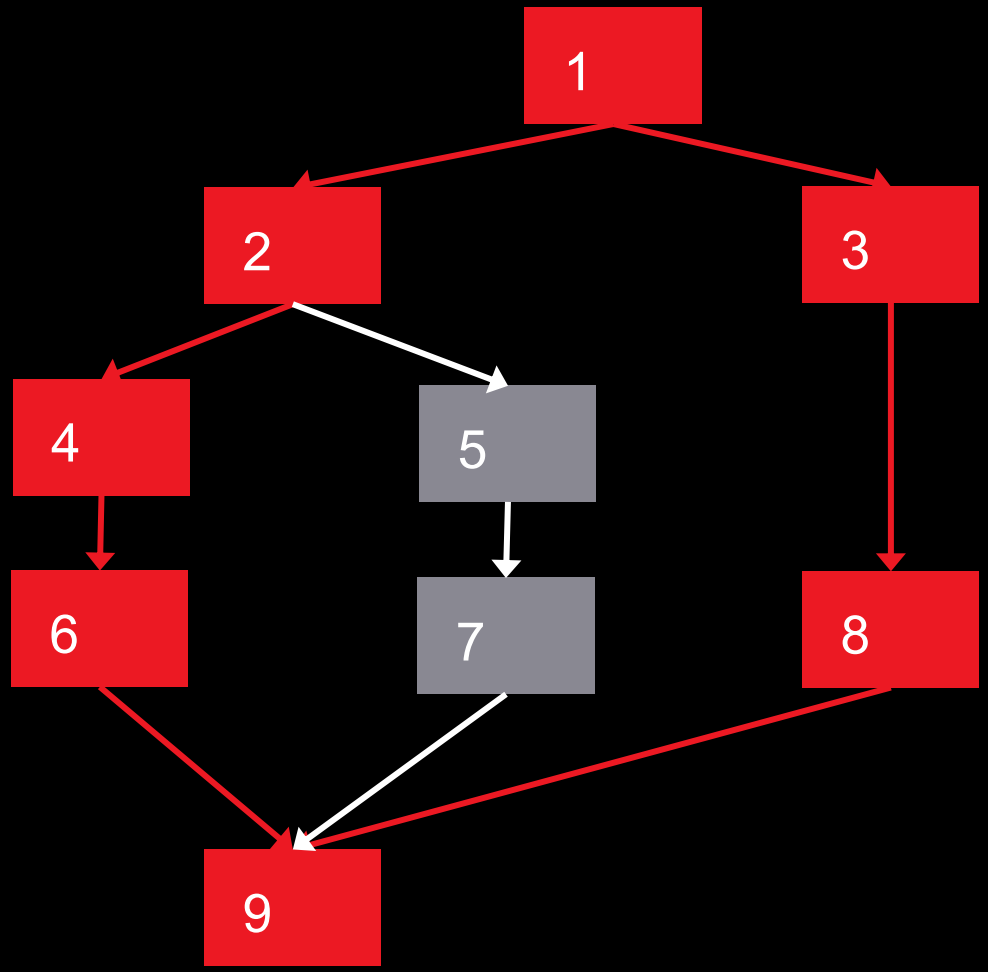
How bad is missing a region?

There is a region pass in the legacy pass manager

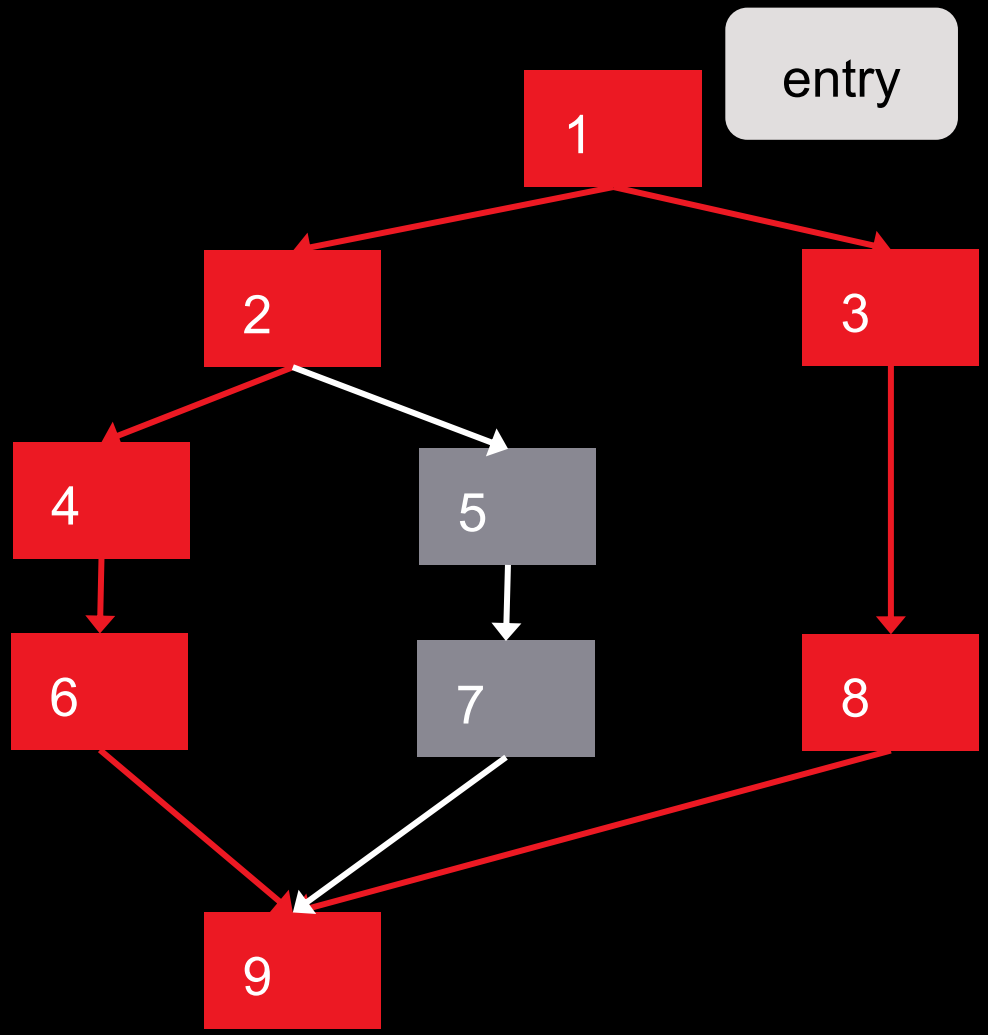
Was it really an interaction bug?

How does region info analysis works

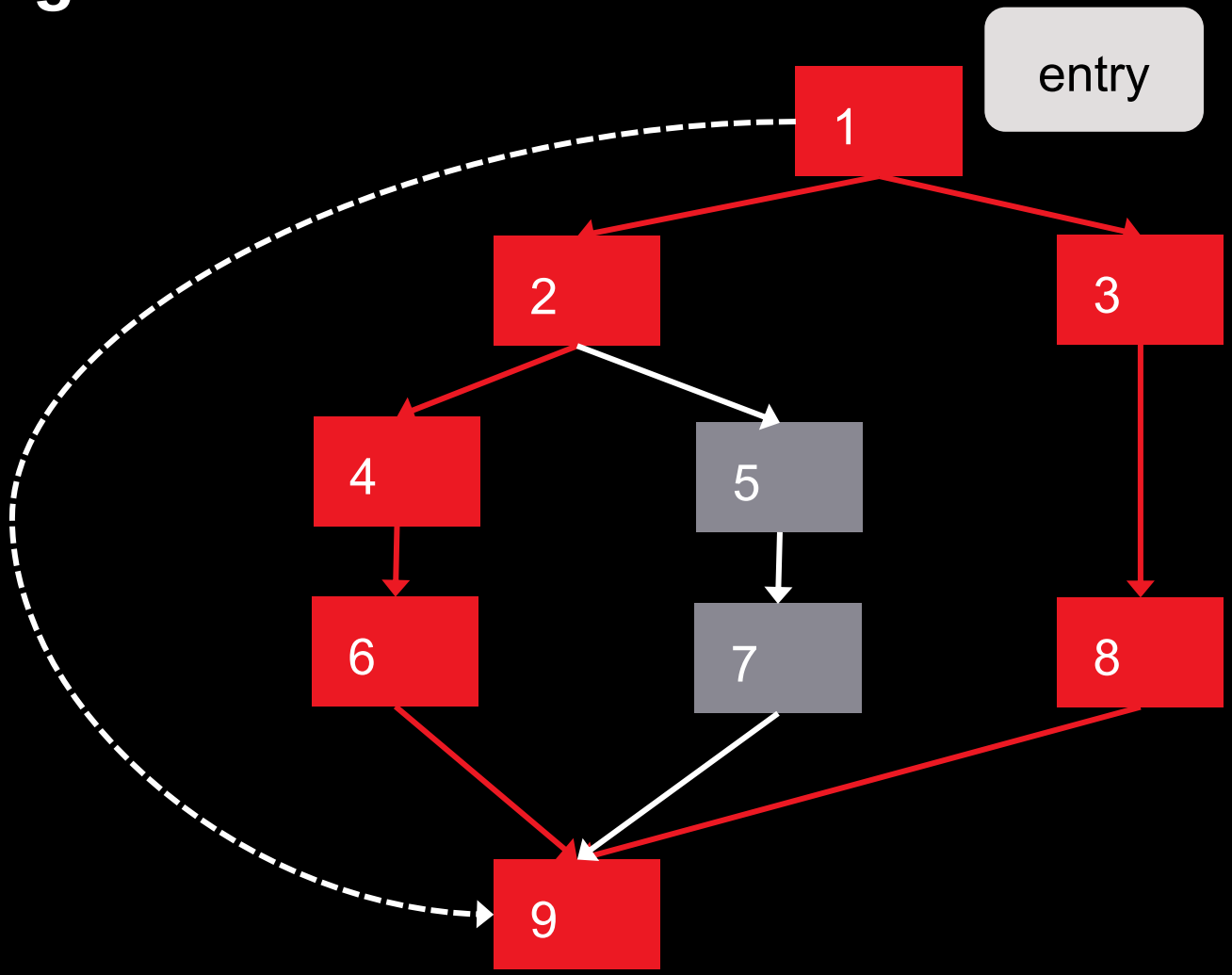
Is it a region?



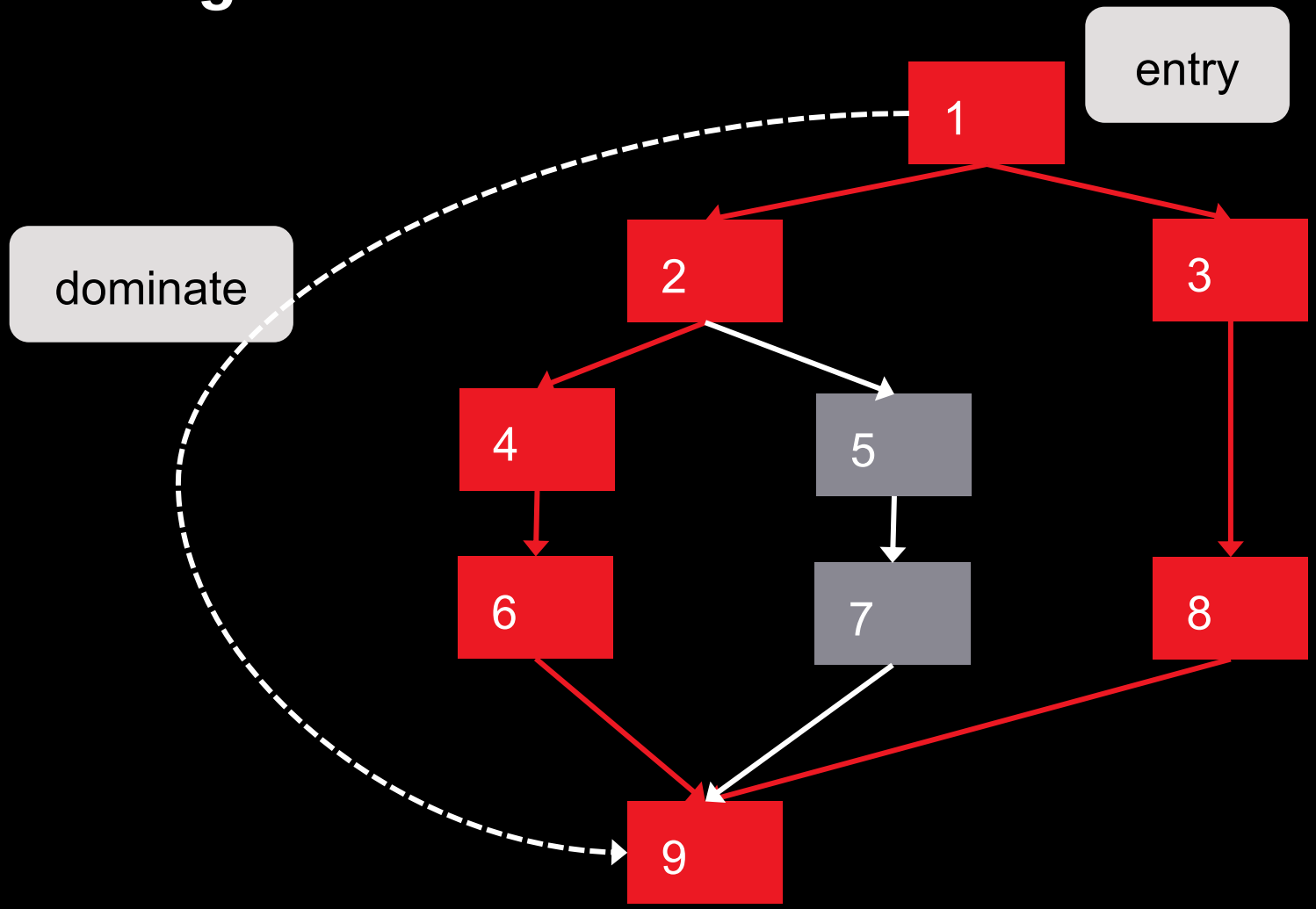
Is it a region?



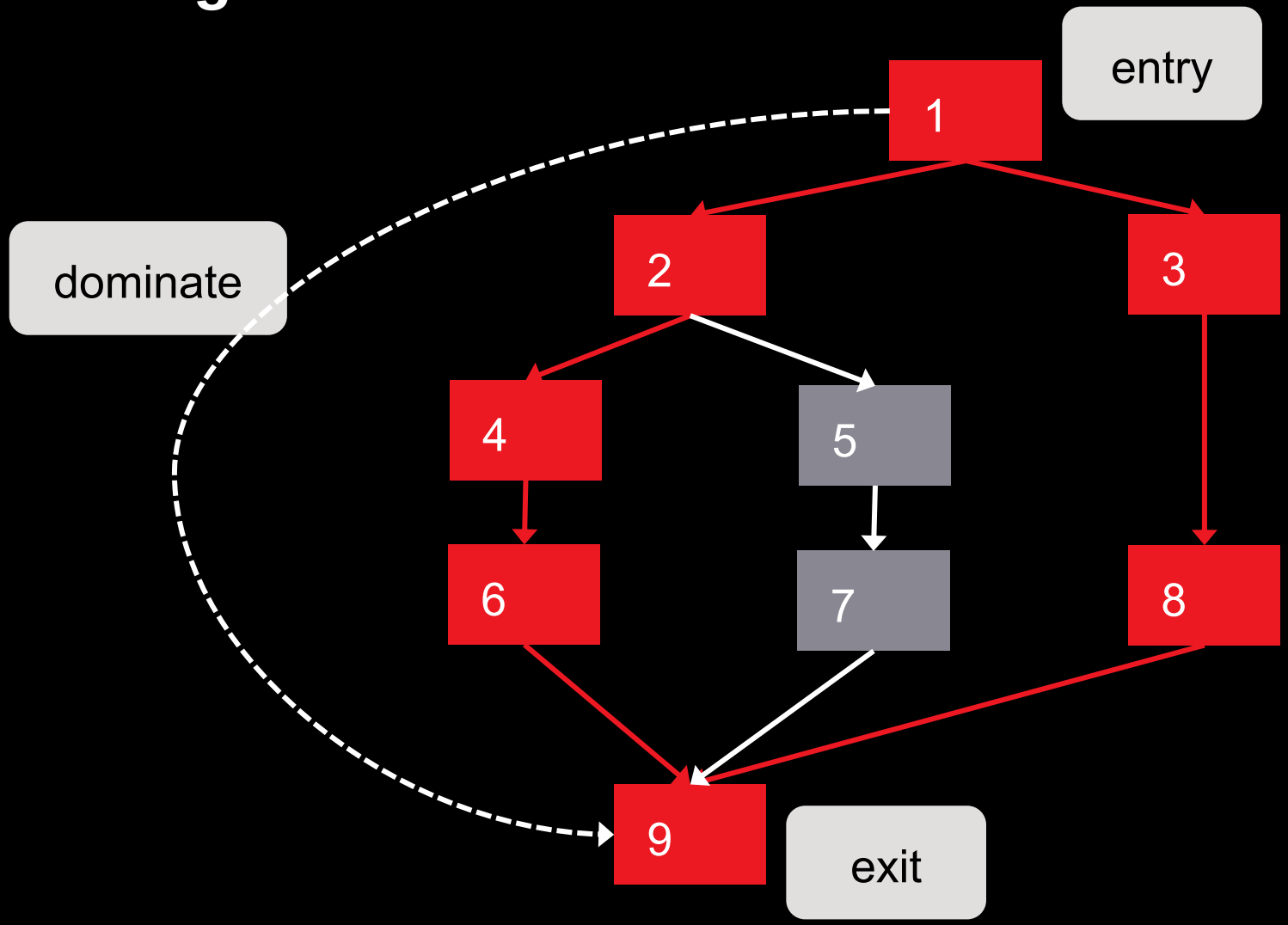
Is it a region?



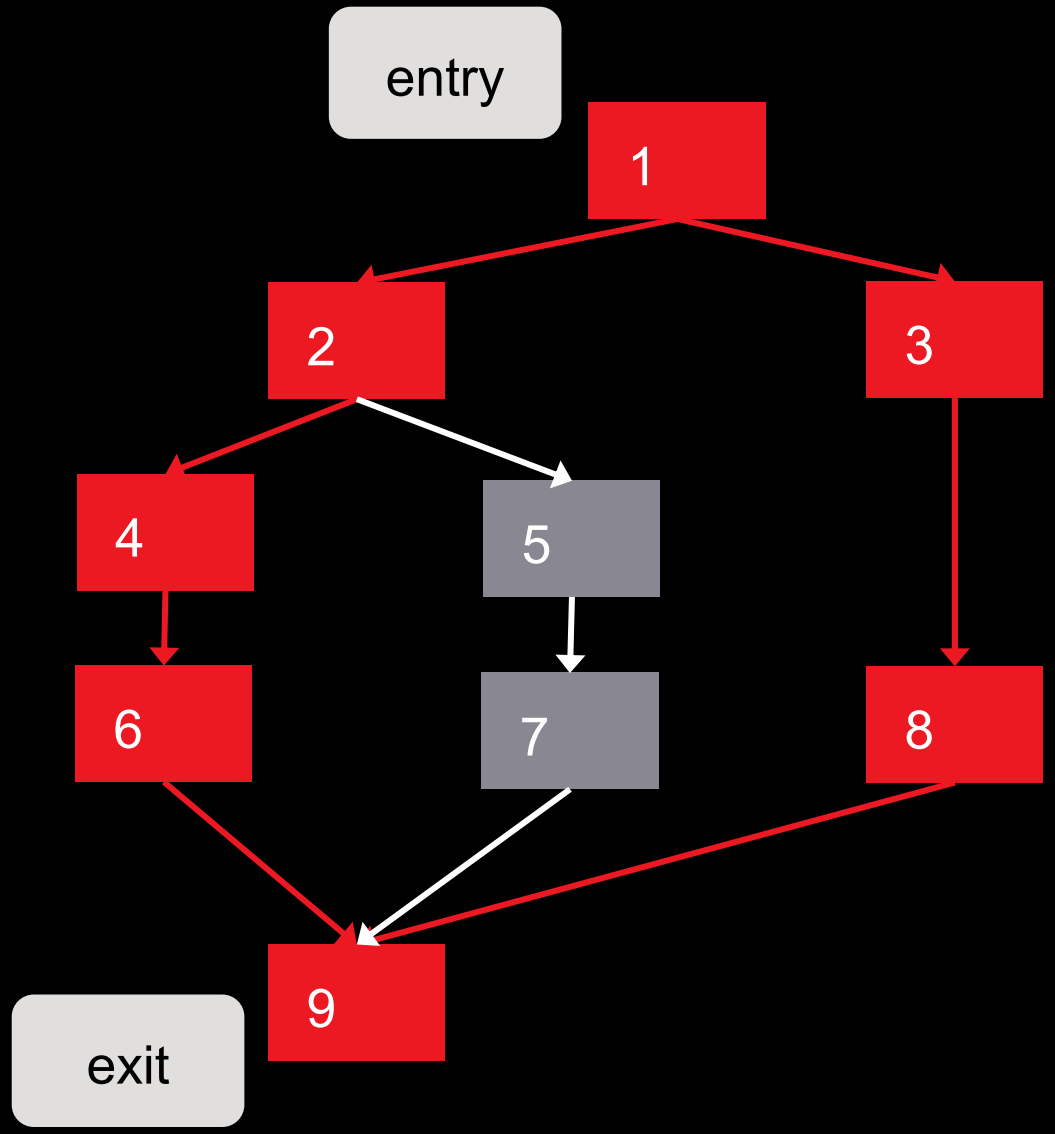
Is it a region?



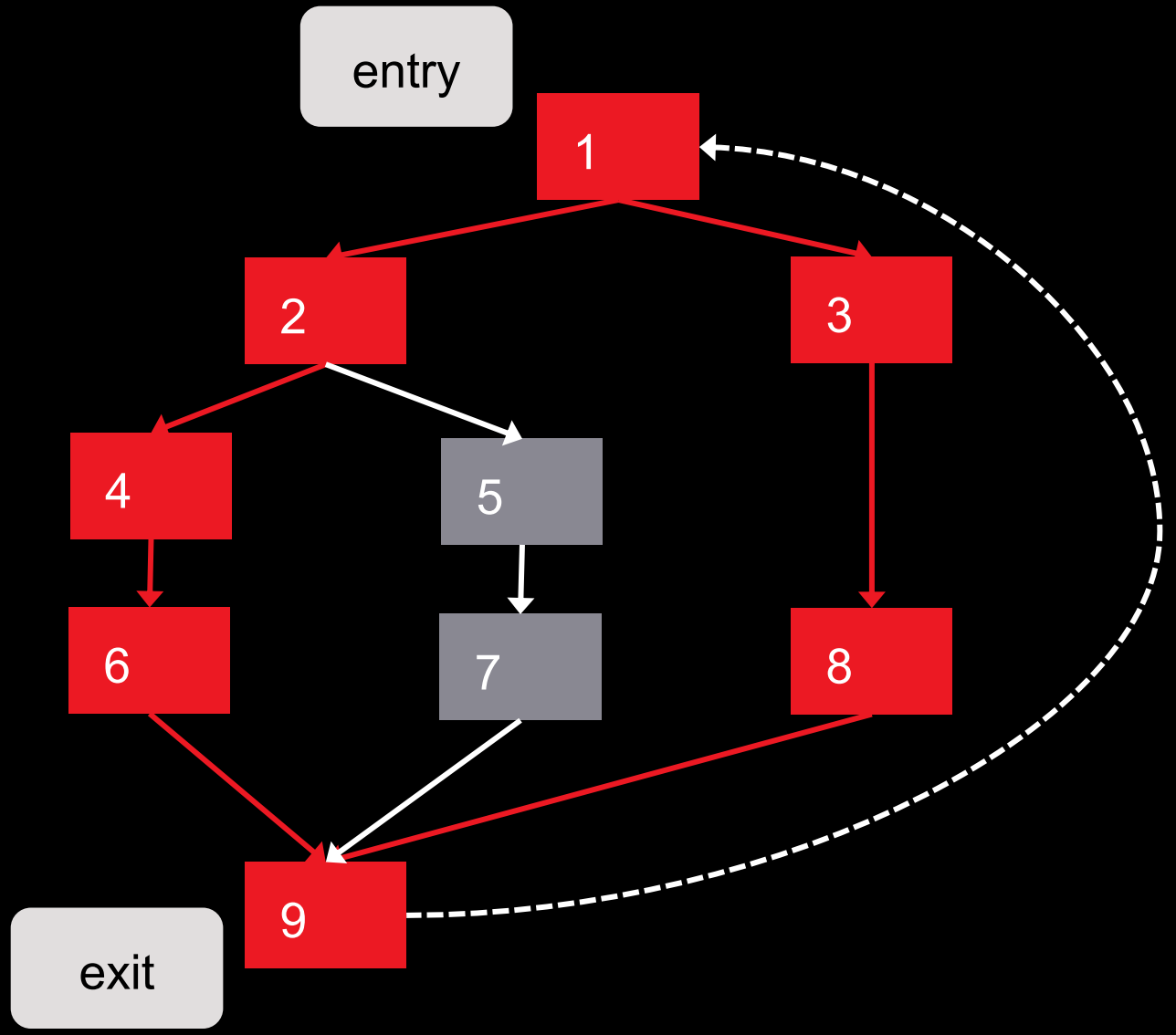
Is it a region?



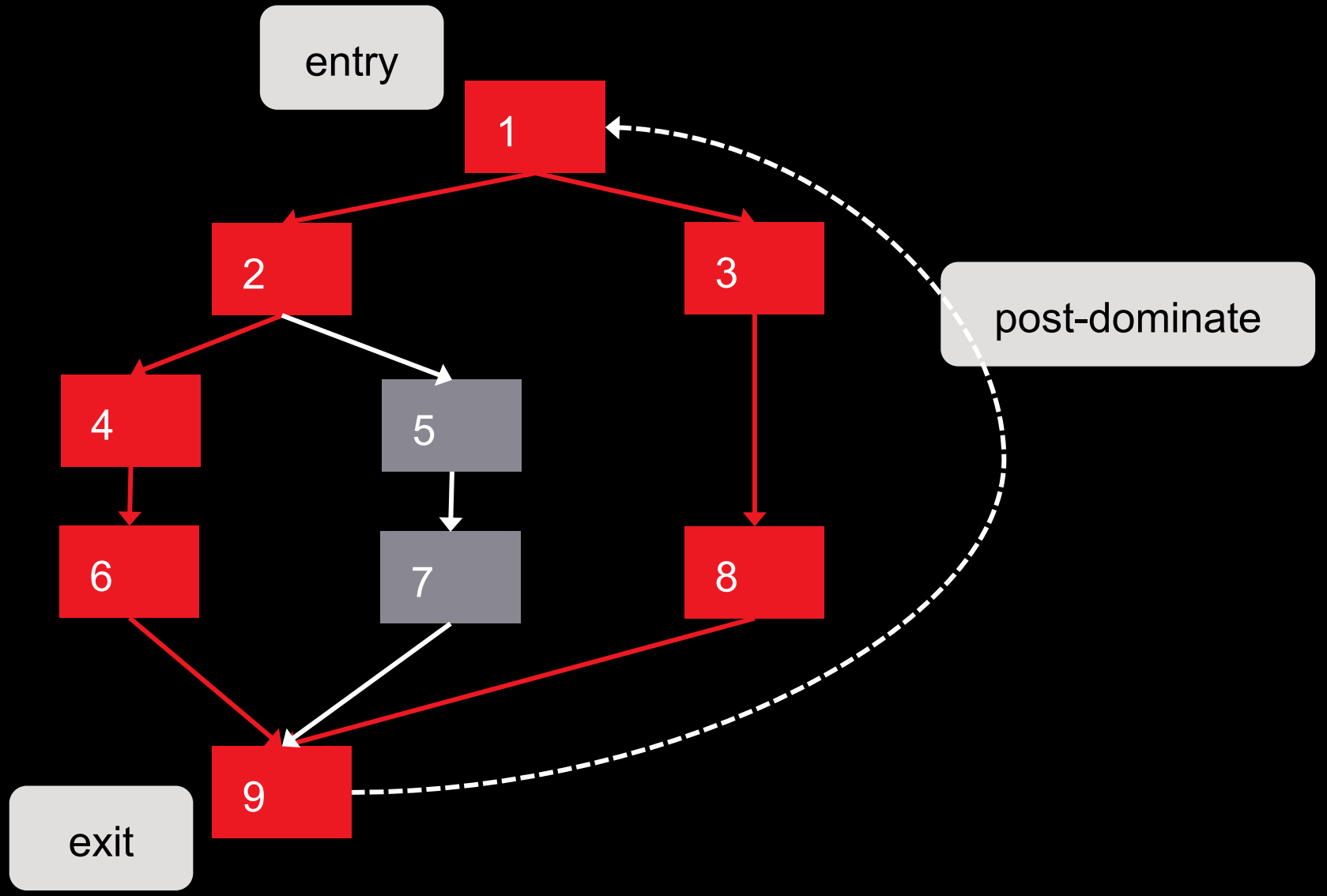
Is it a region?



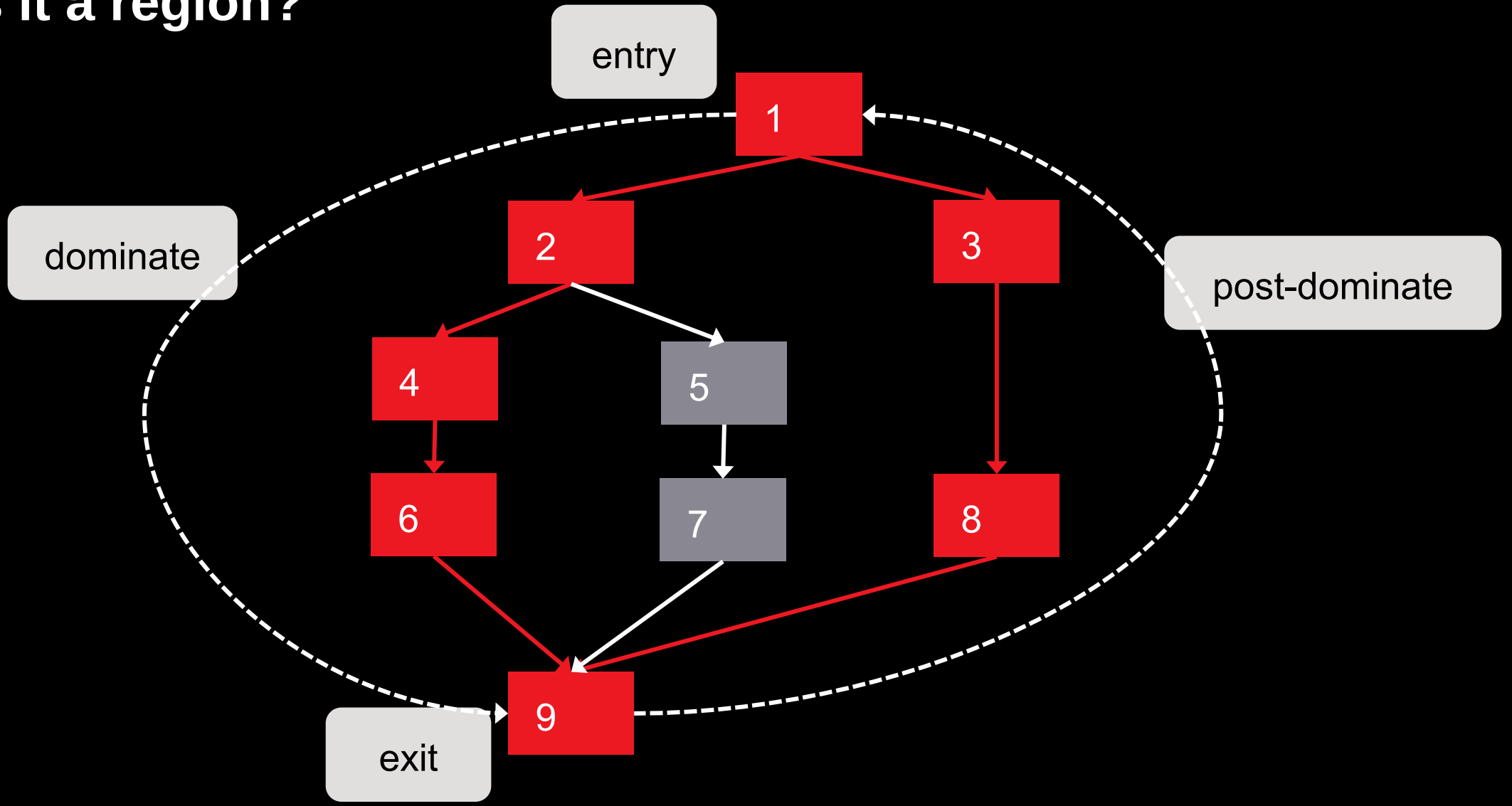
Is it a region?



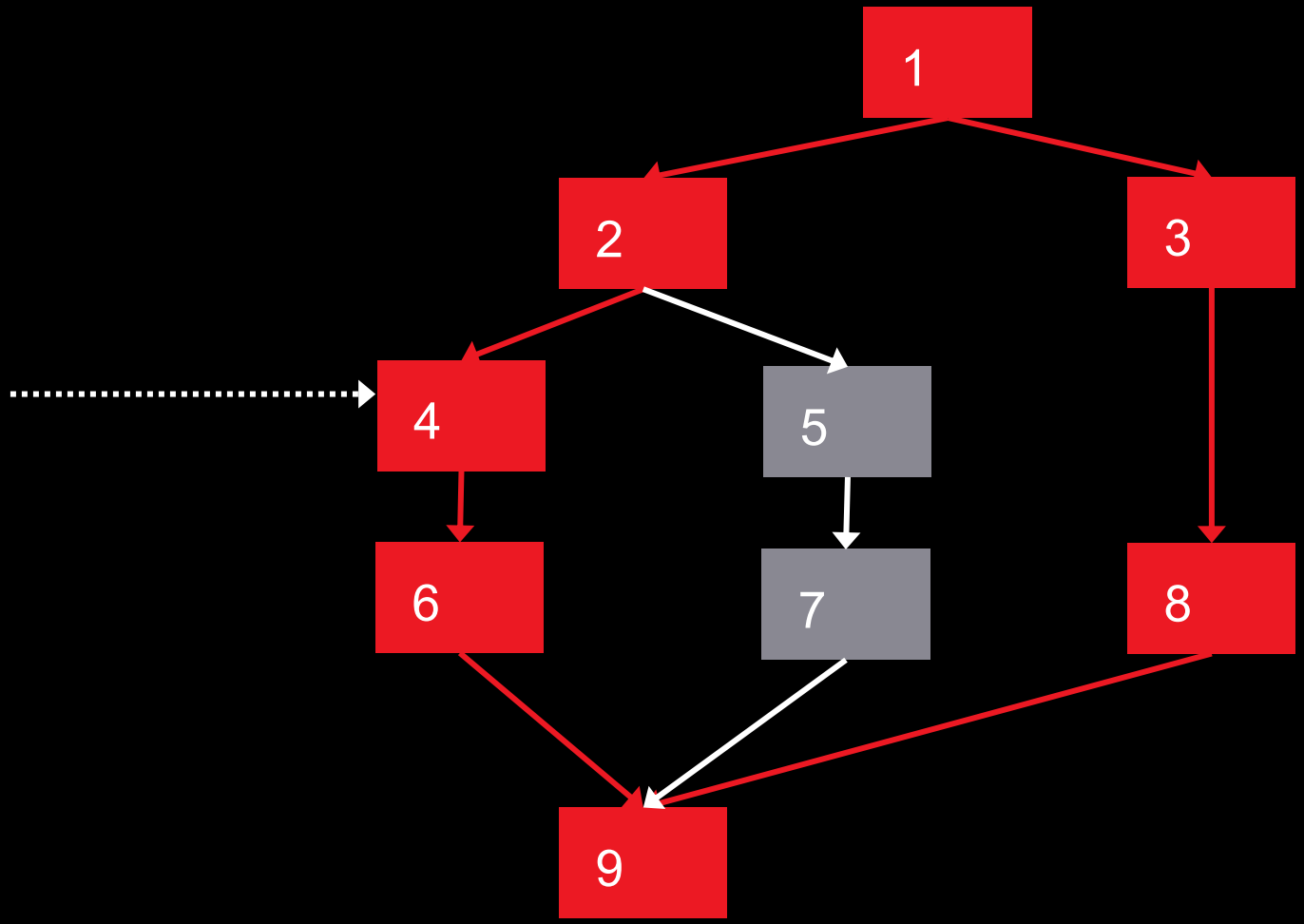
Is it a region?



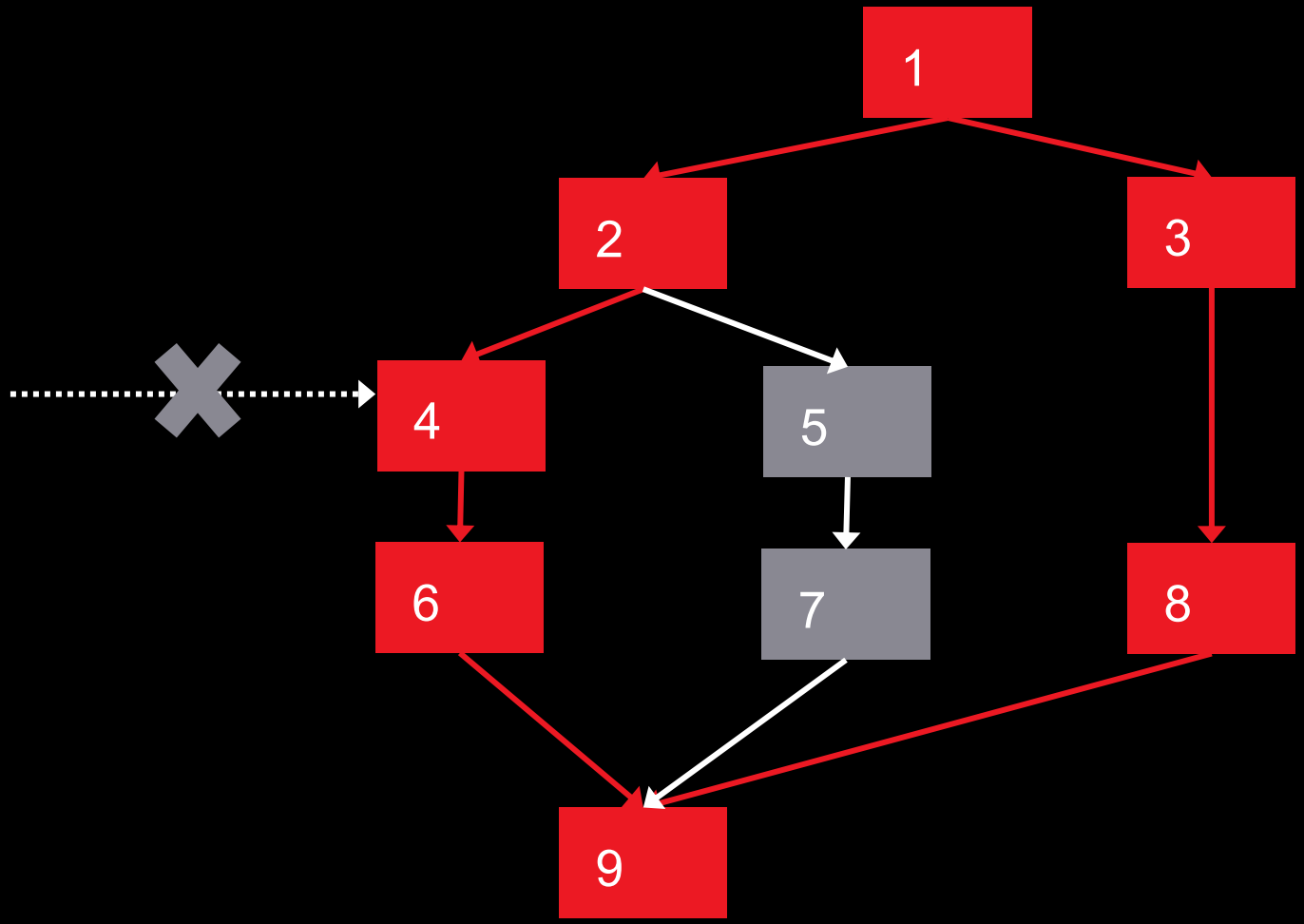
Is it a region?



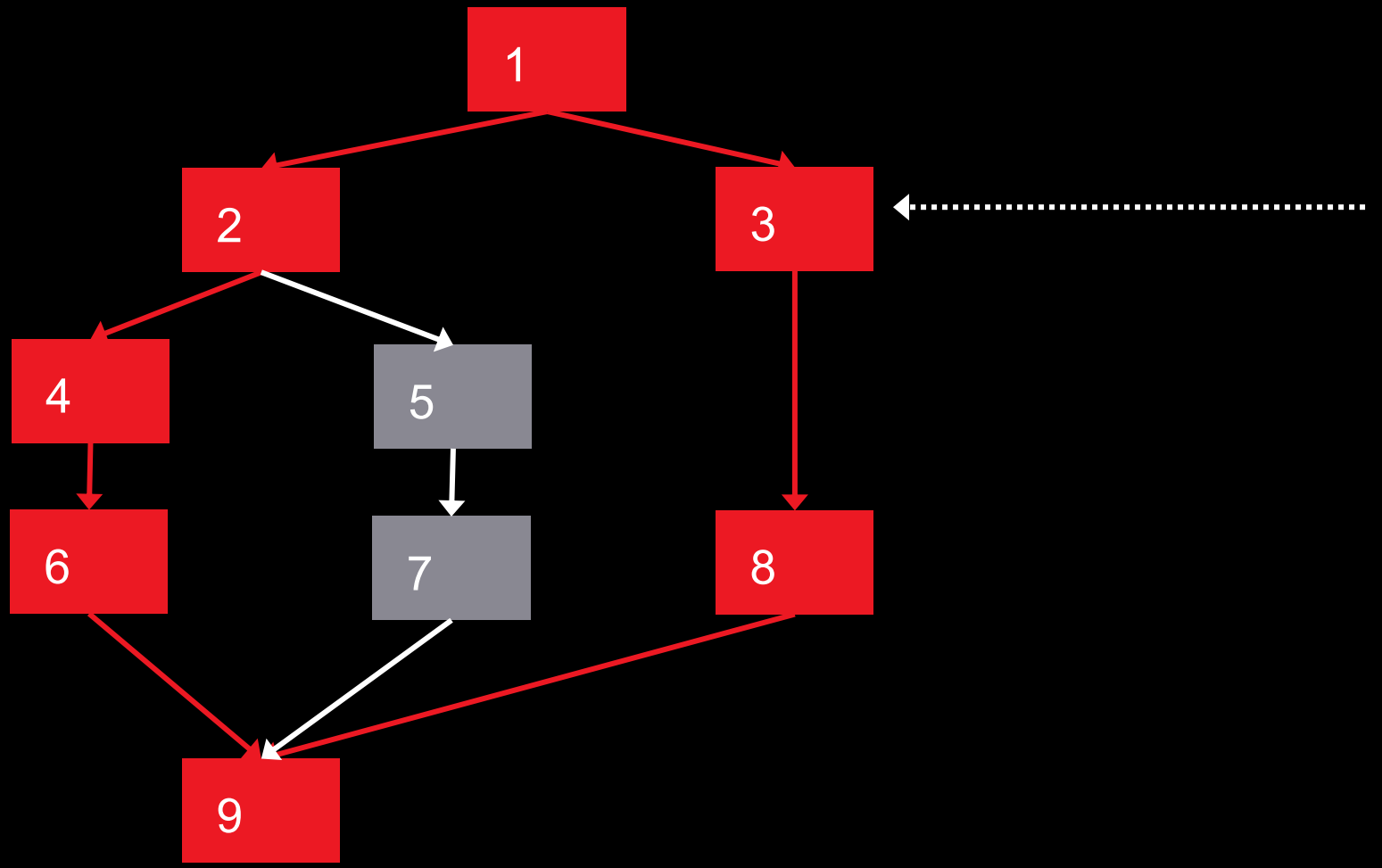
Is it a region?



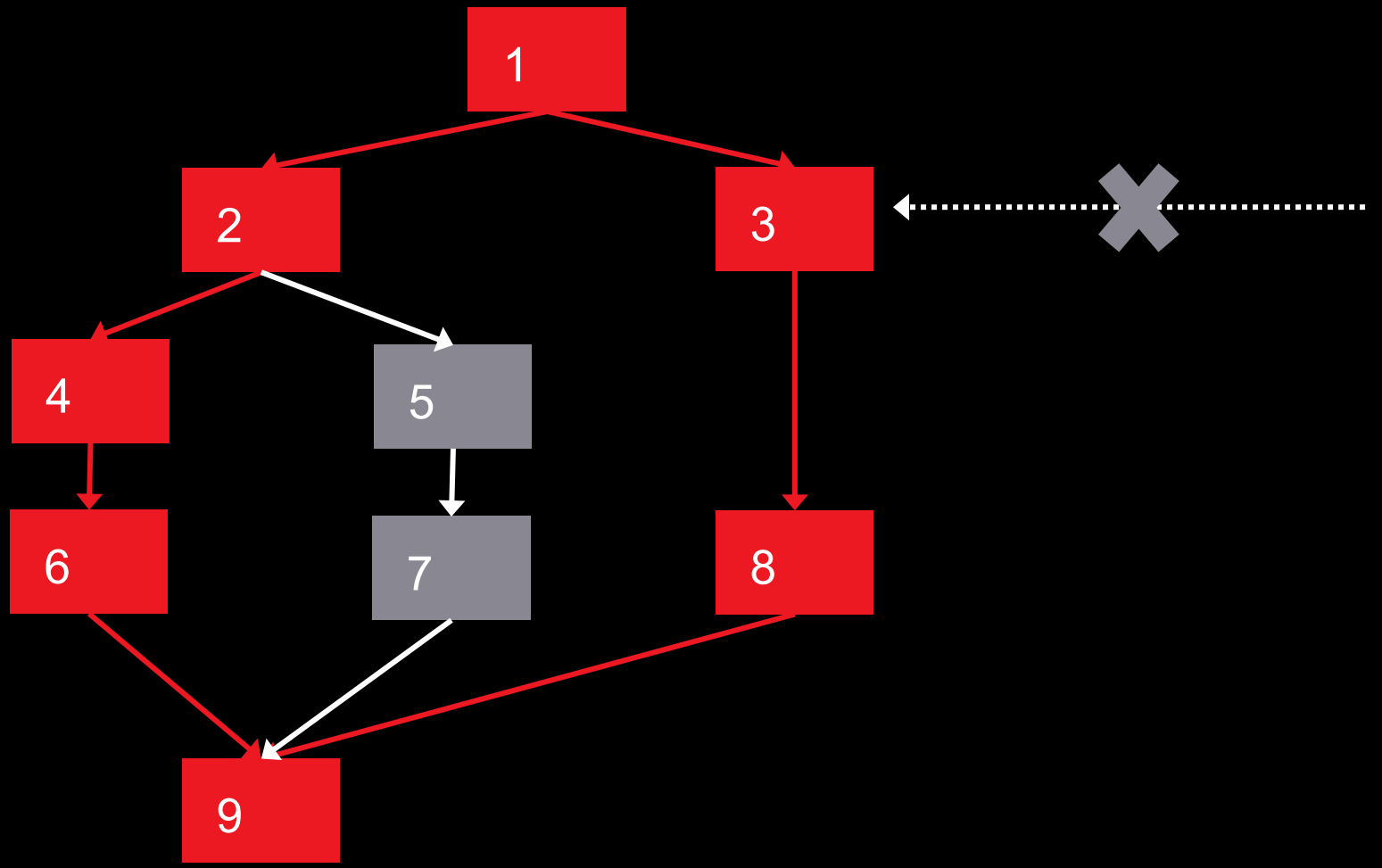
Is it a region?



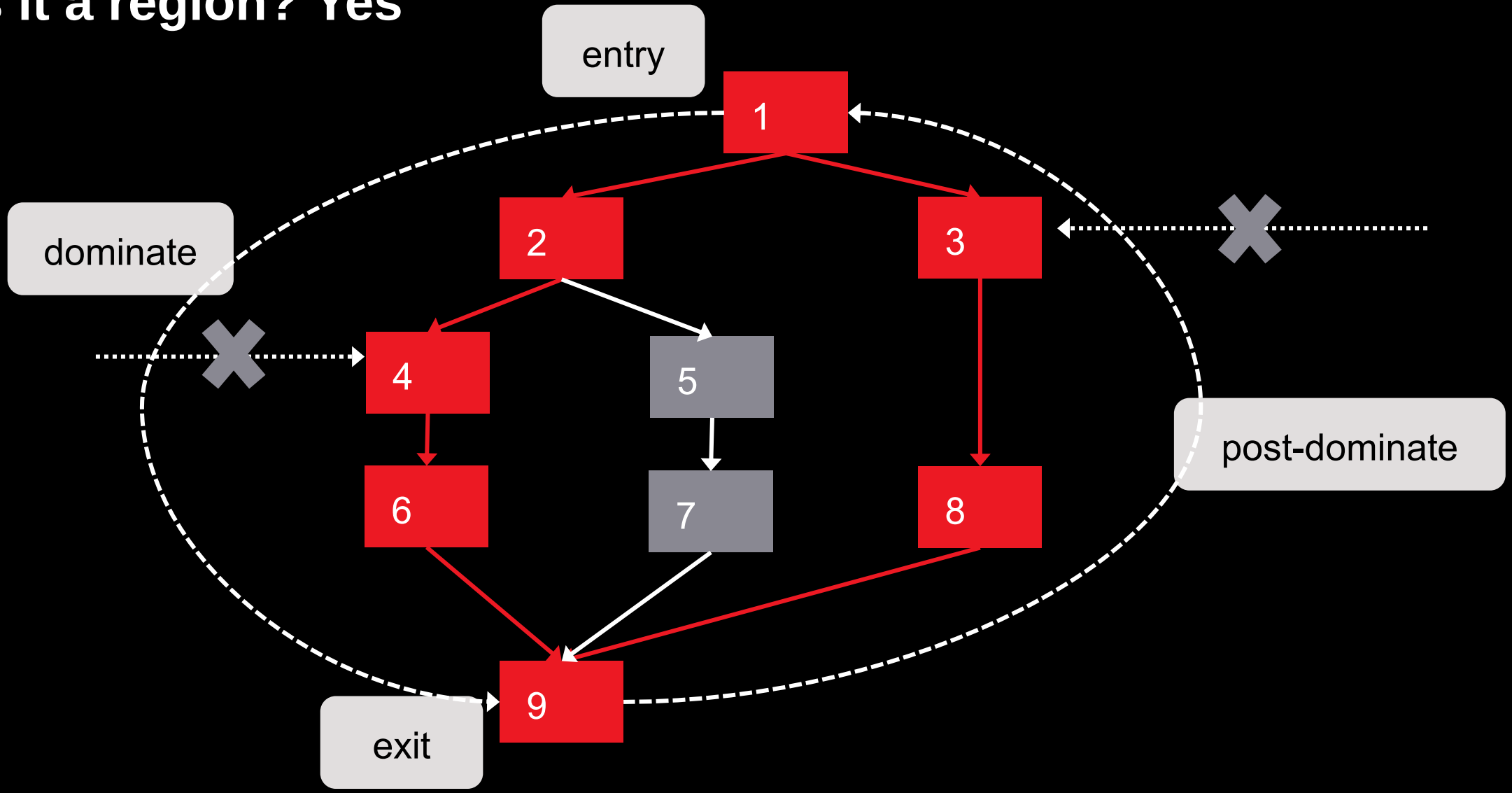
Is it a region?



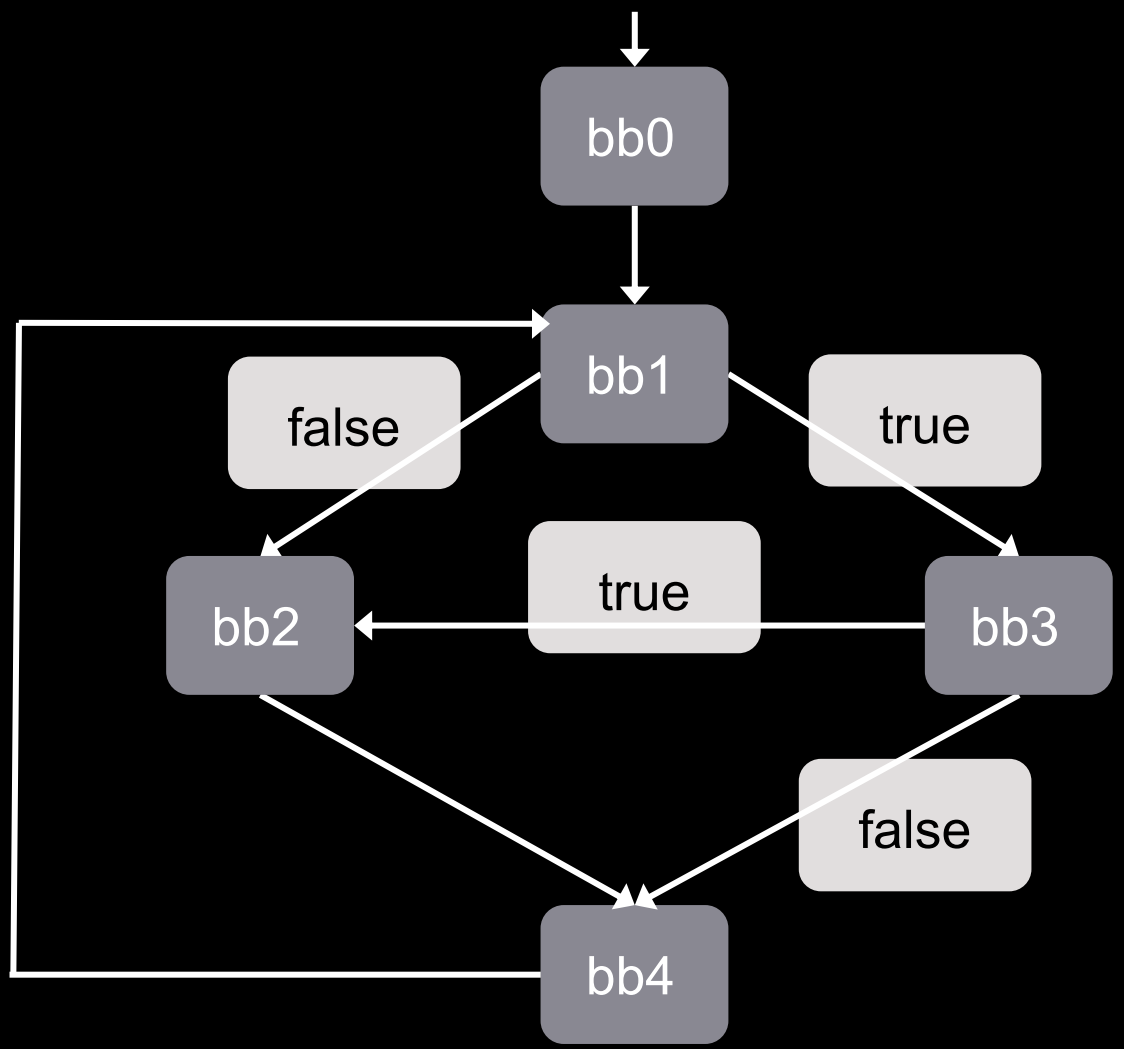
Is it a region?

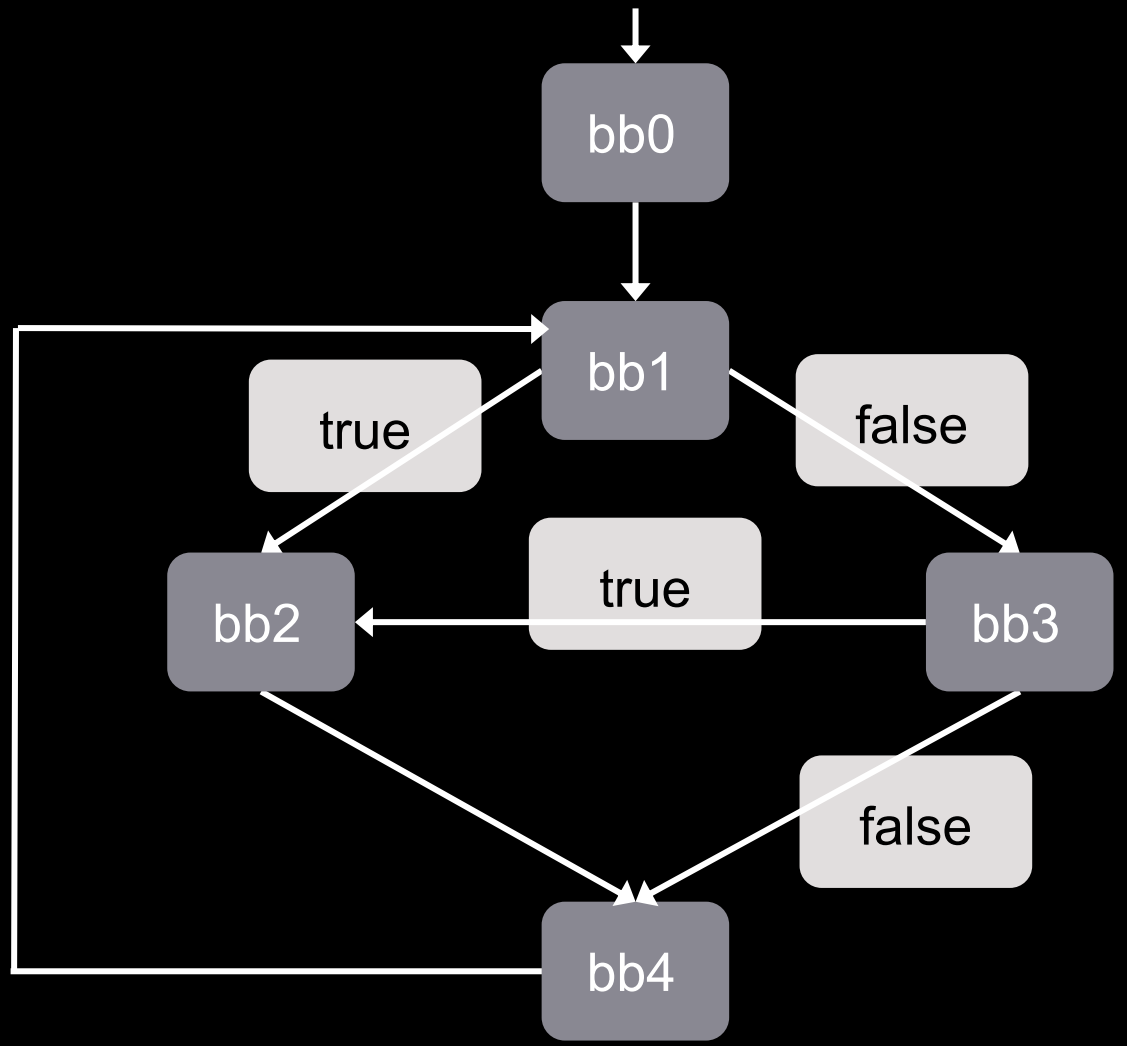


Is it a region? Yes

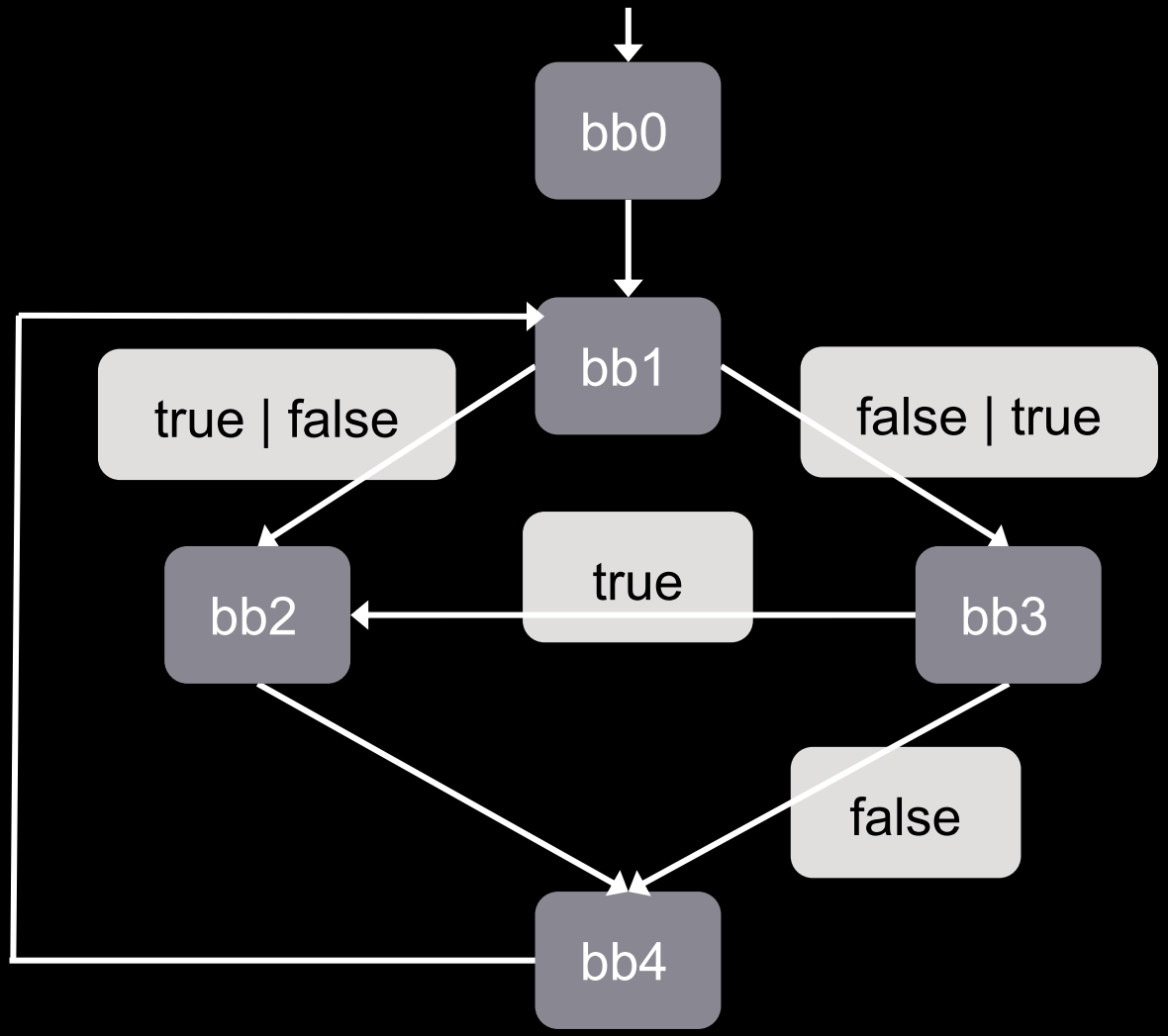


The test case and the analyses results

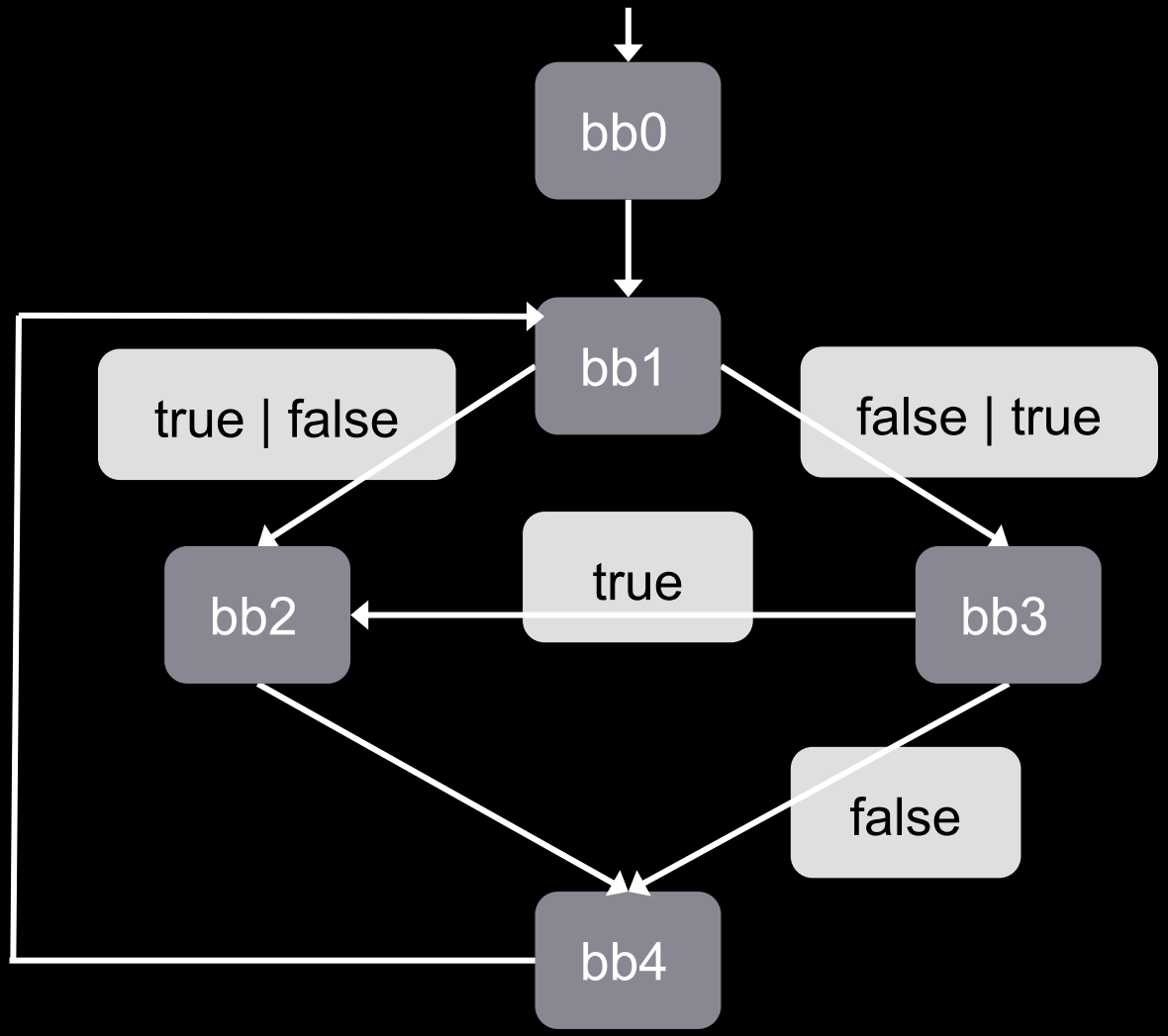




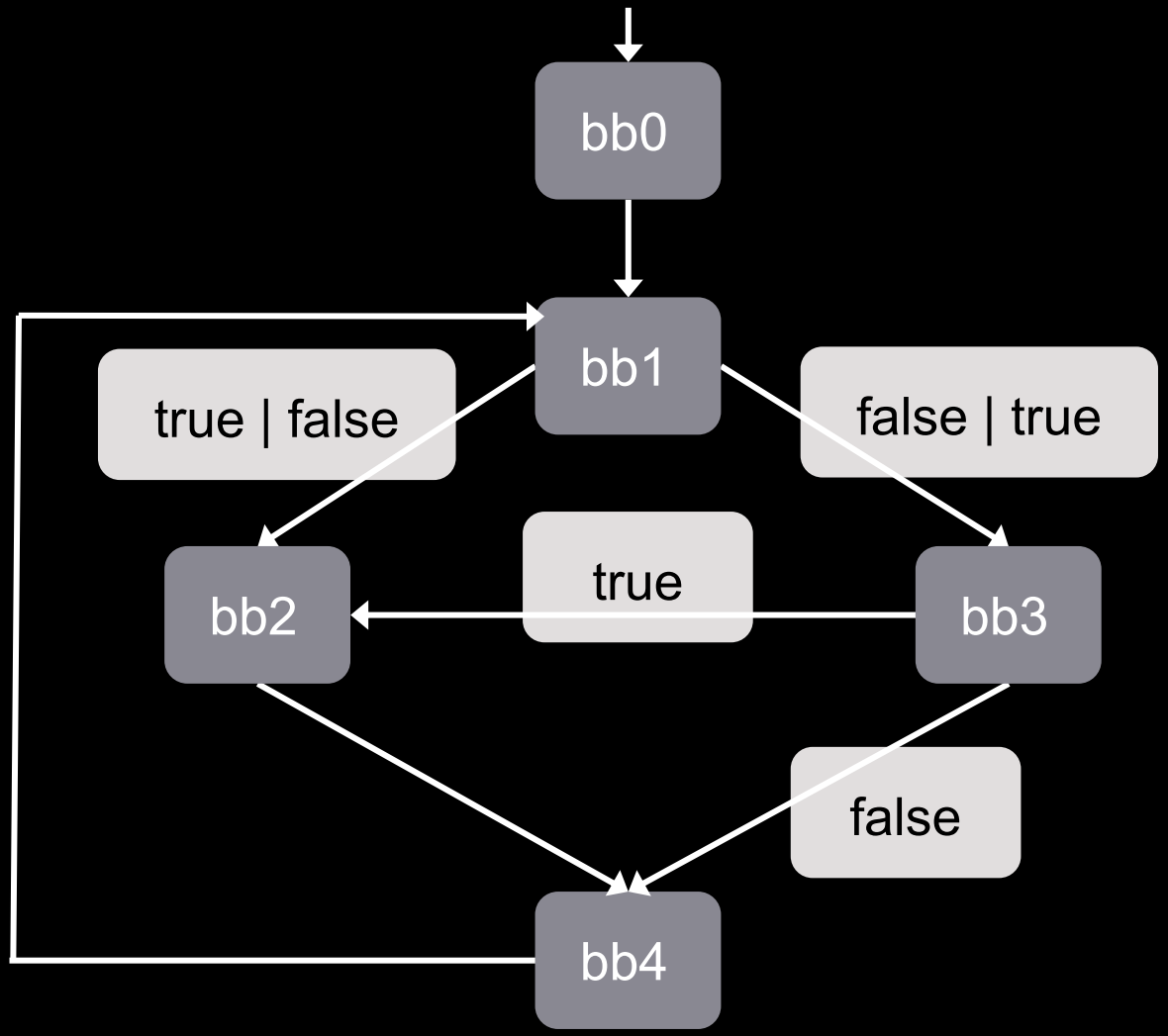
The test case and the analyses results

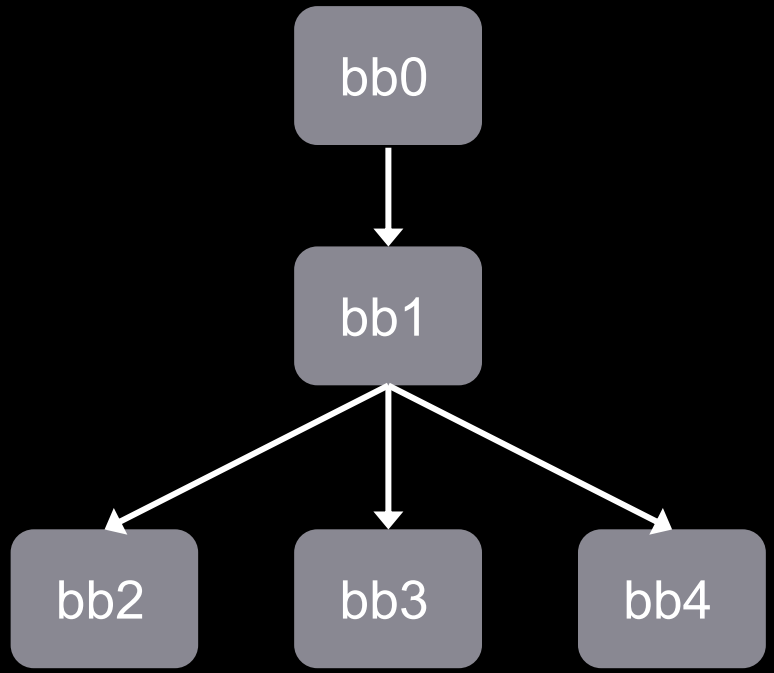


The test case and the dominator tree

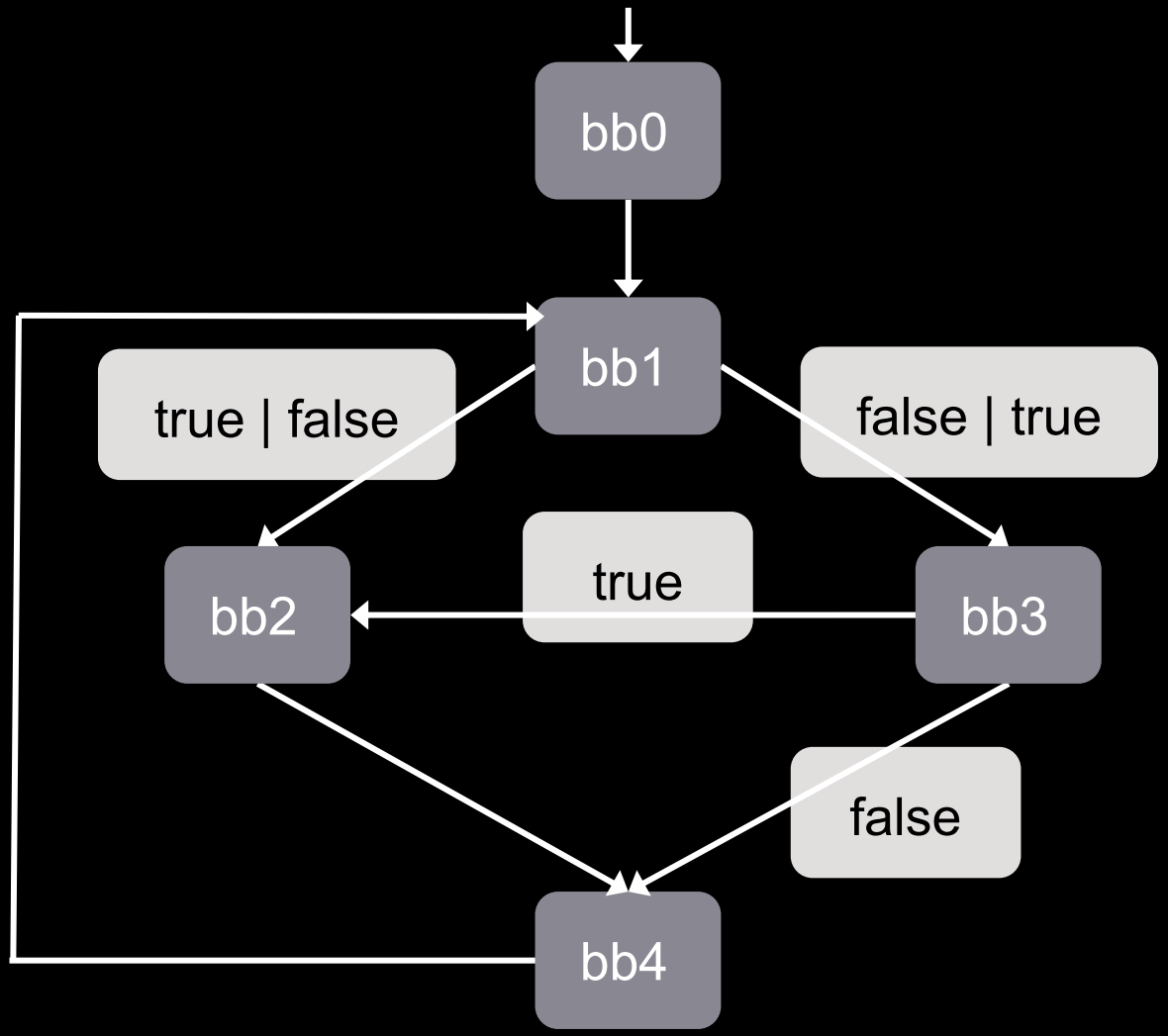


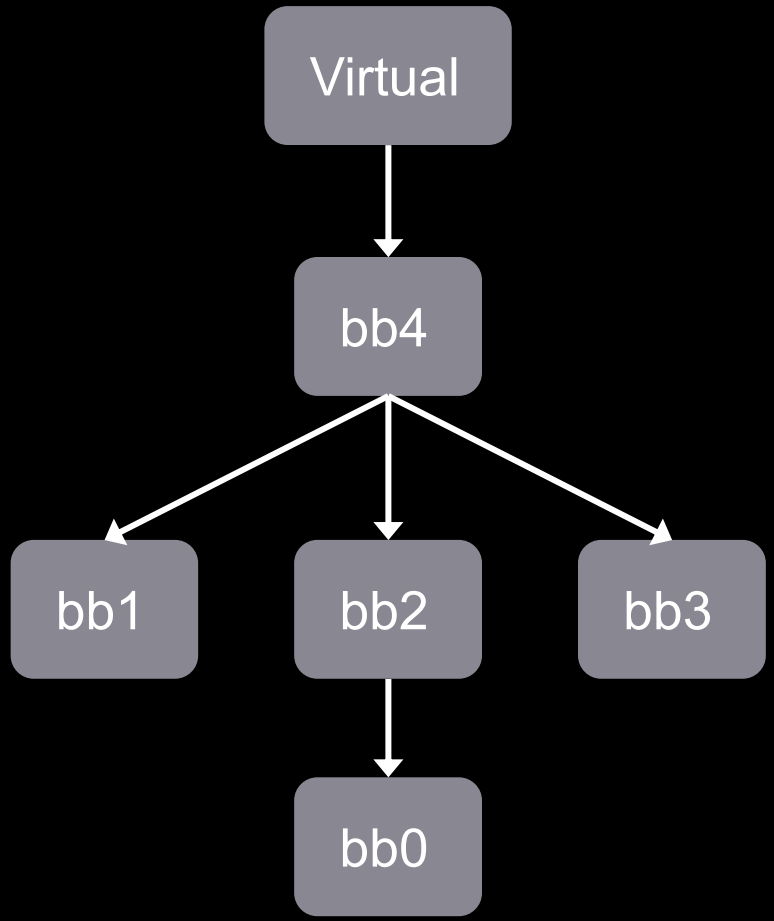
The test case and the dominator tree



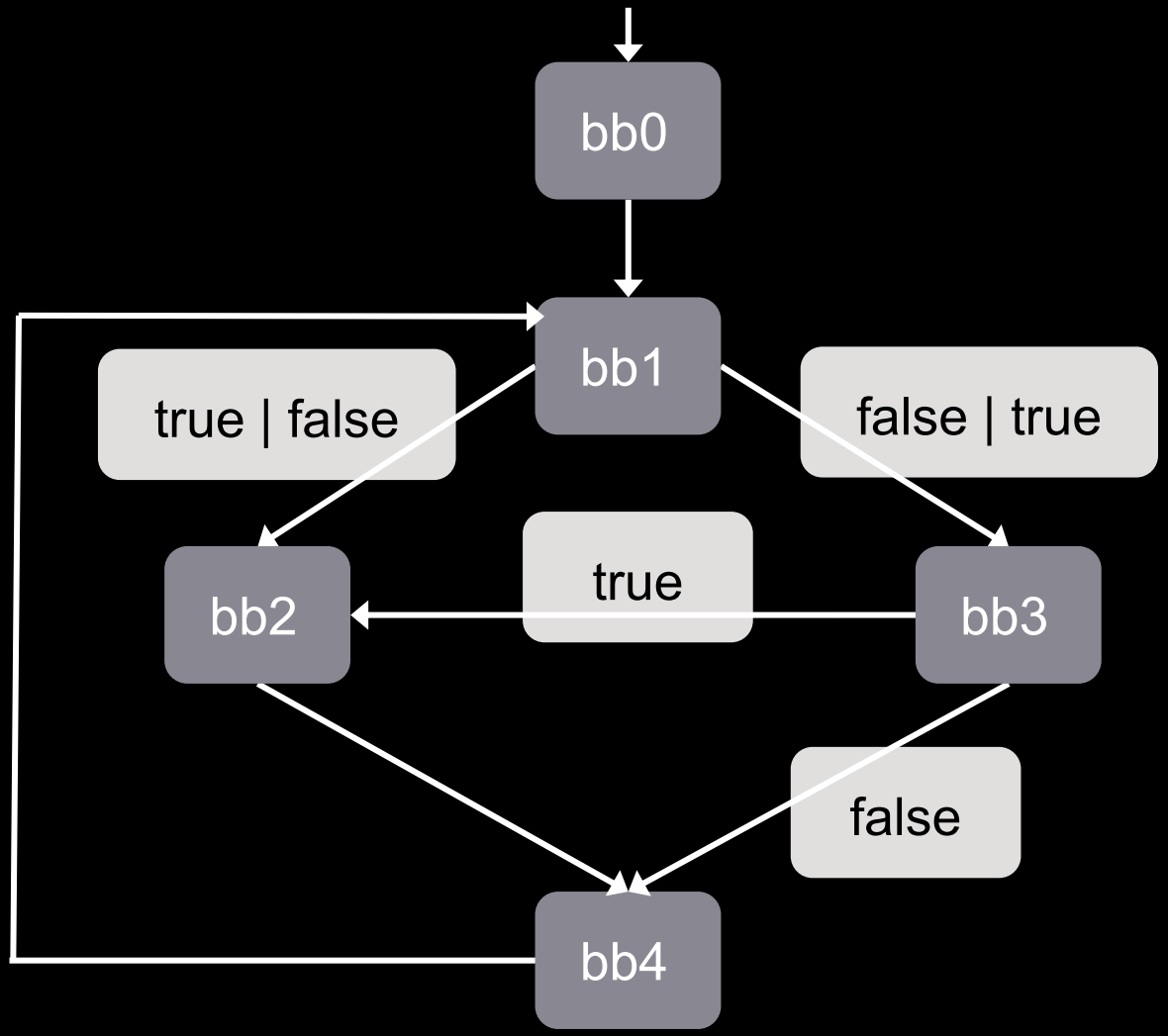


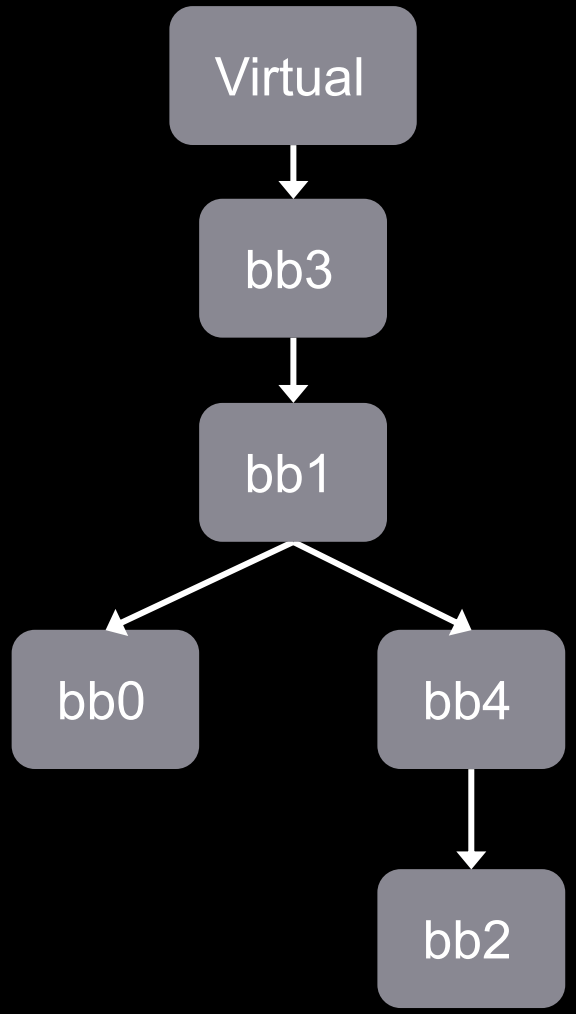
The test case and the post dominator tree



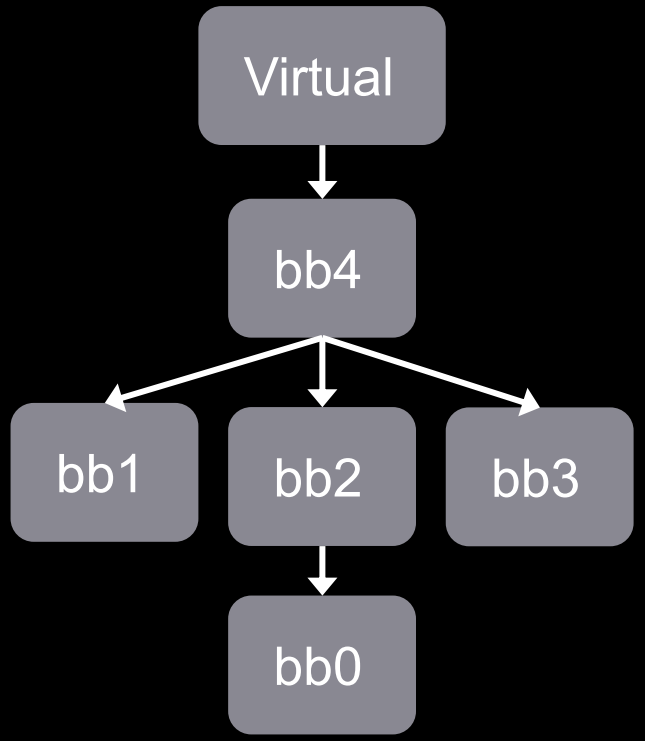


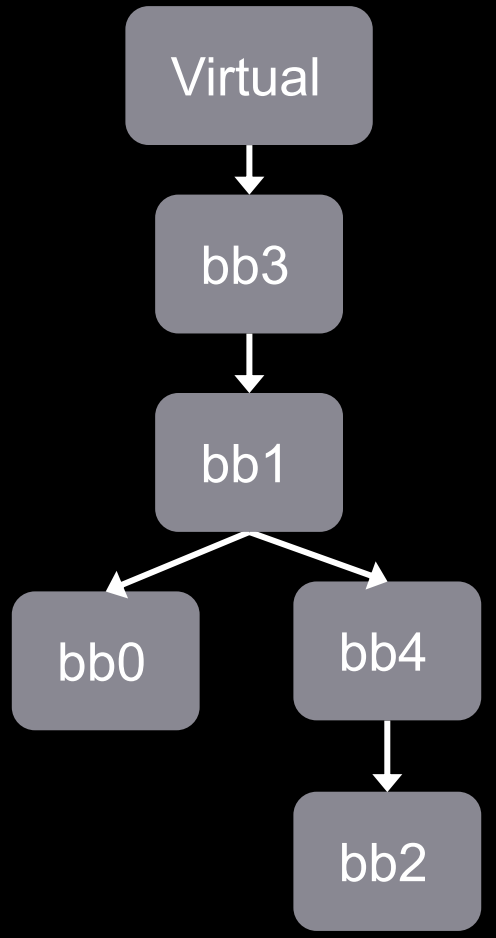
The test case and the post dominator tree (again)



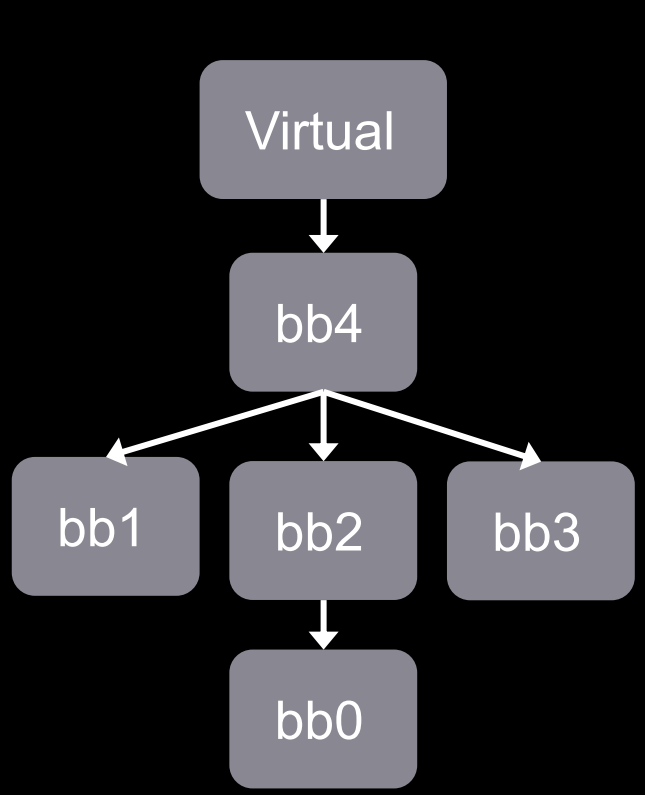


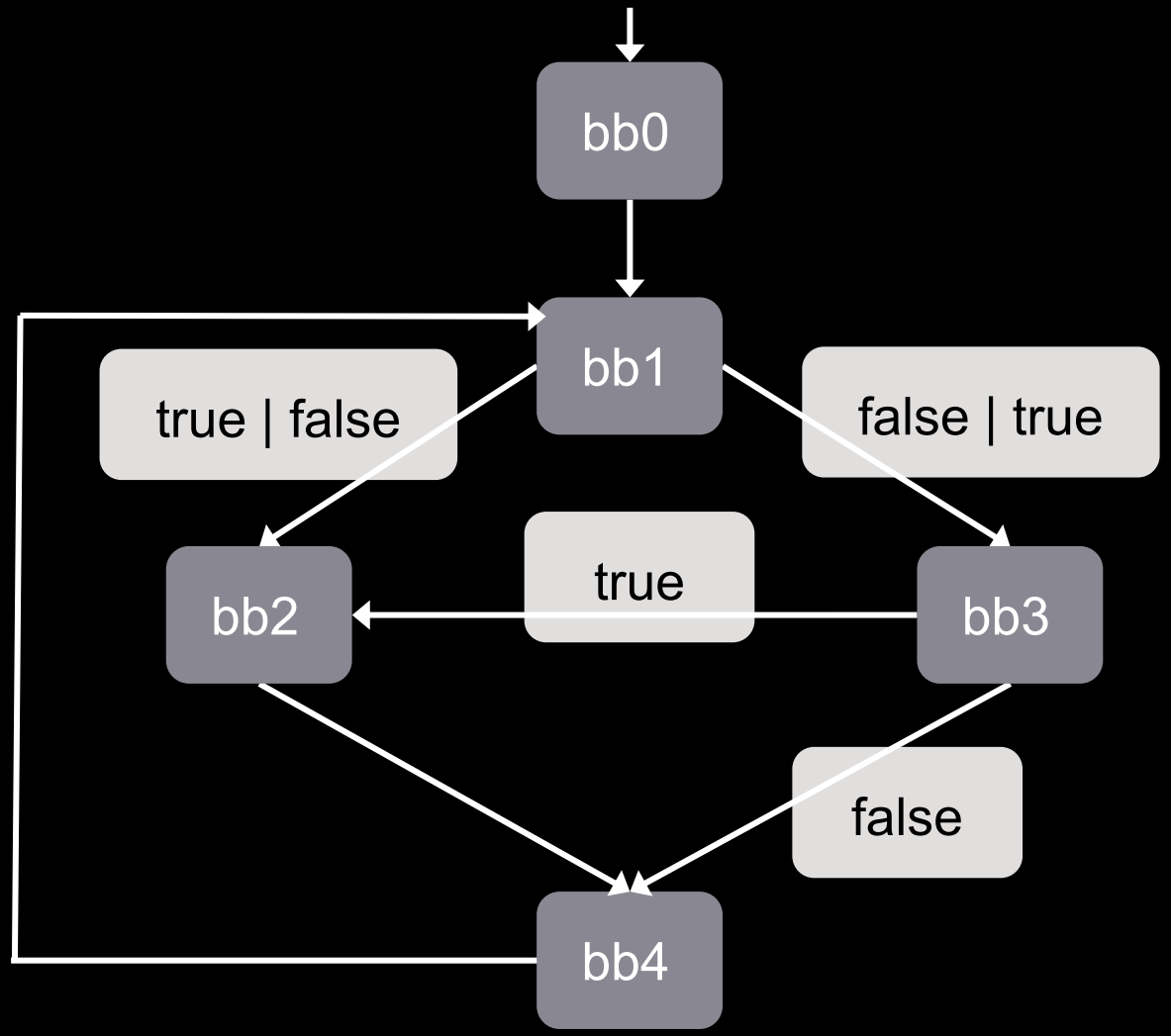
Two legal post dominator trees

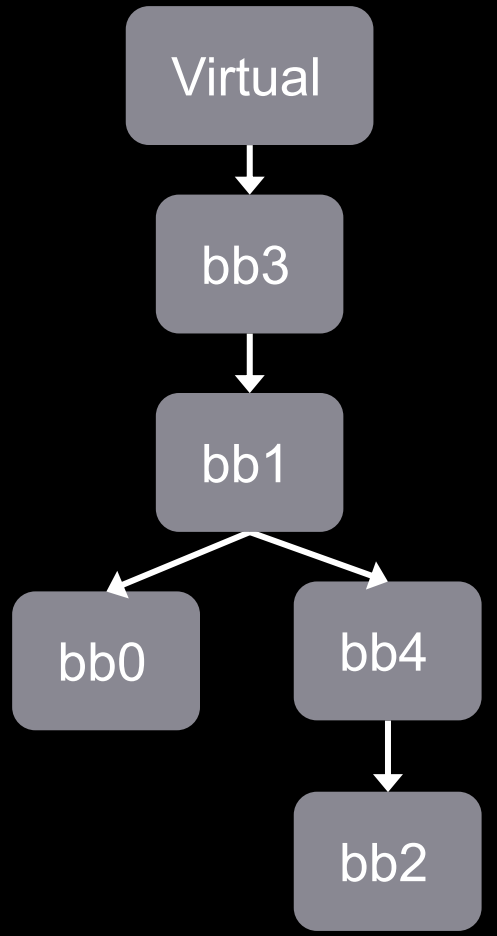




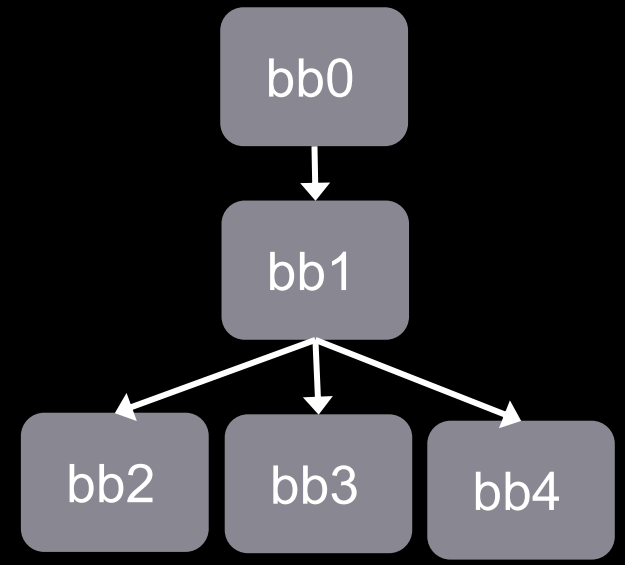
Two legal post dominator trees



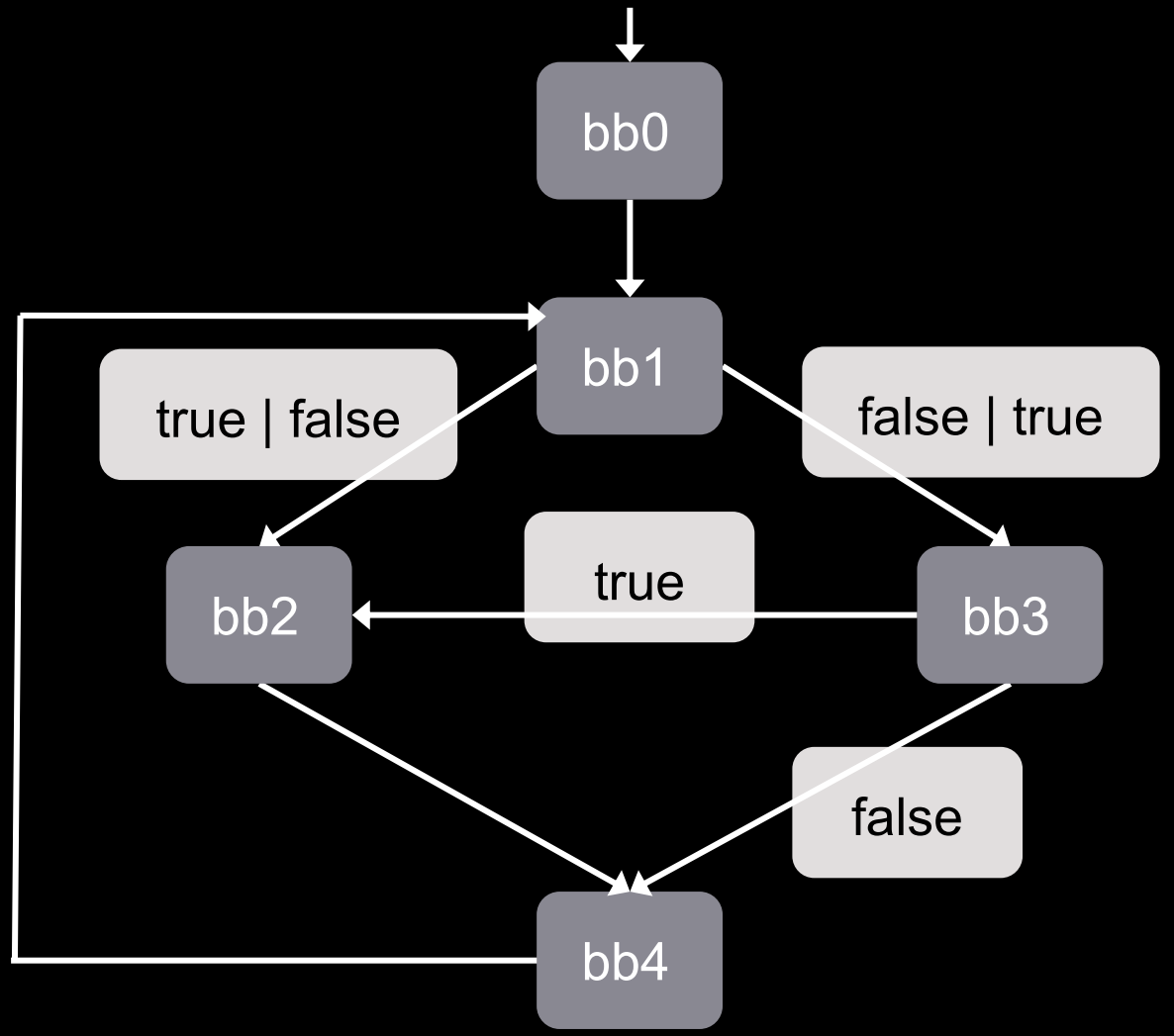


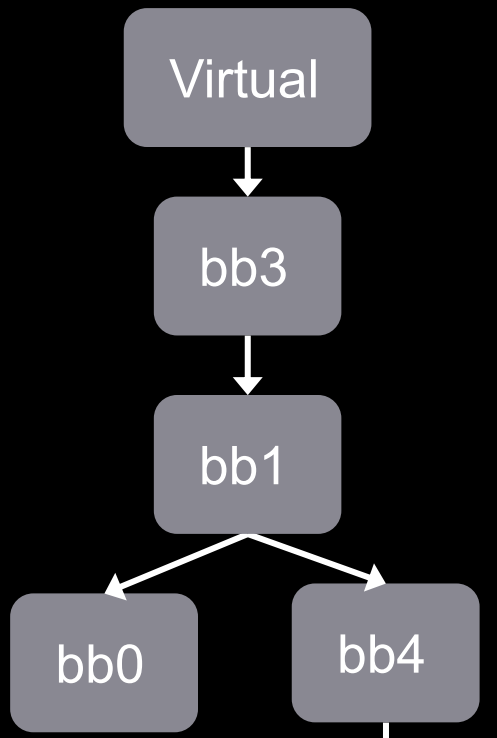


Test cases and analyses



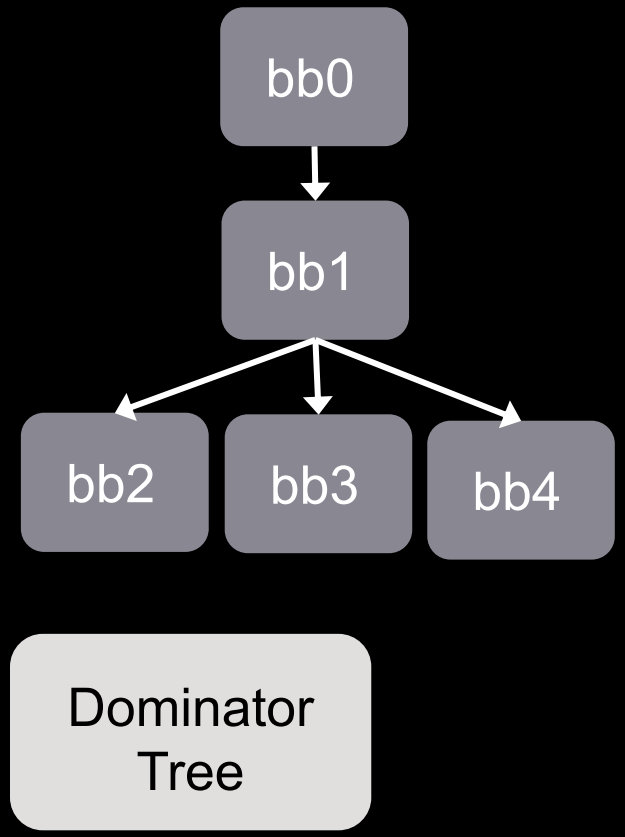
Dominator Tree

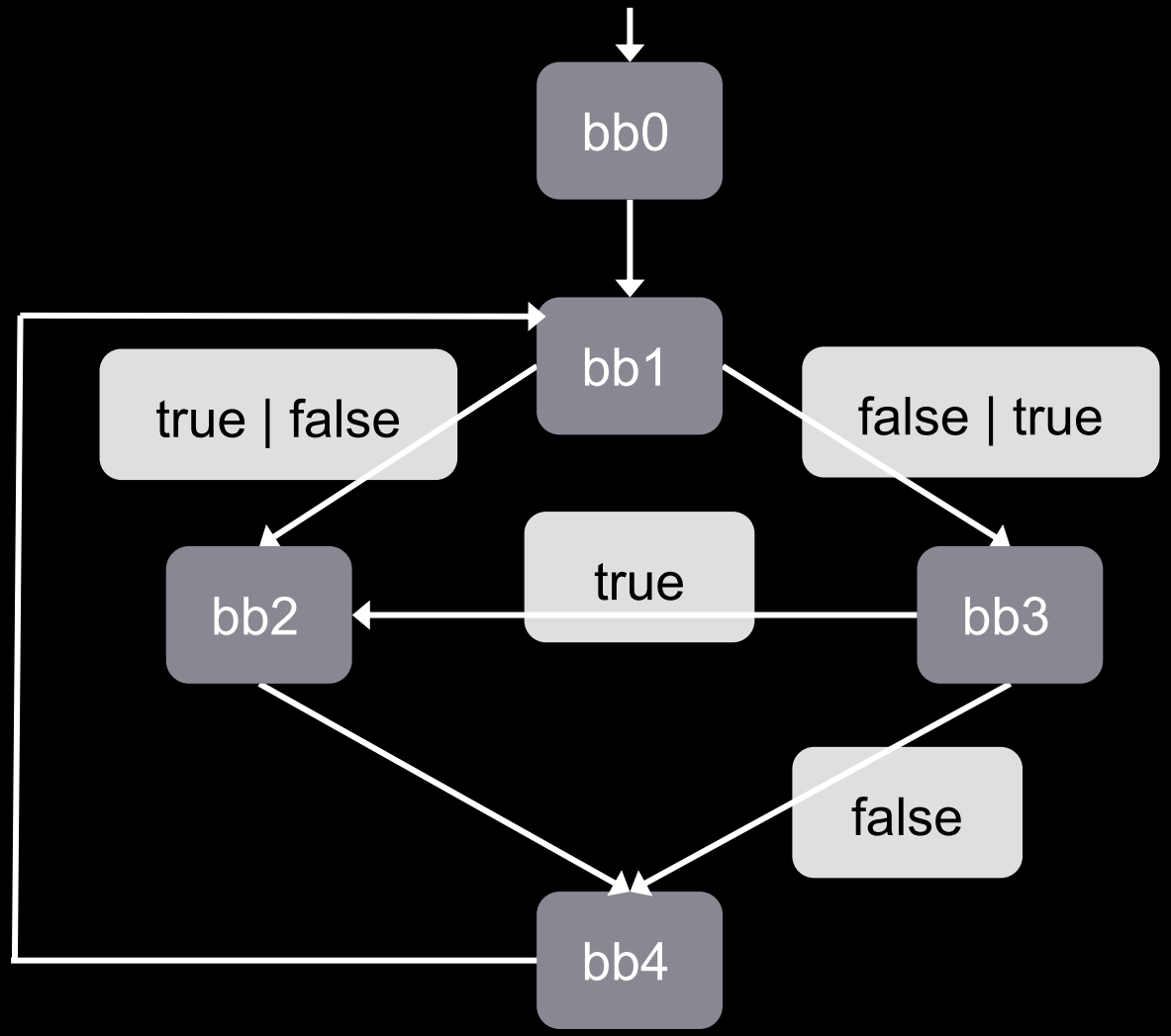


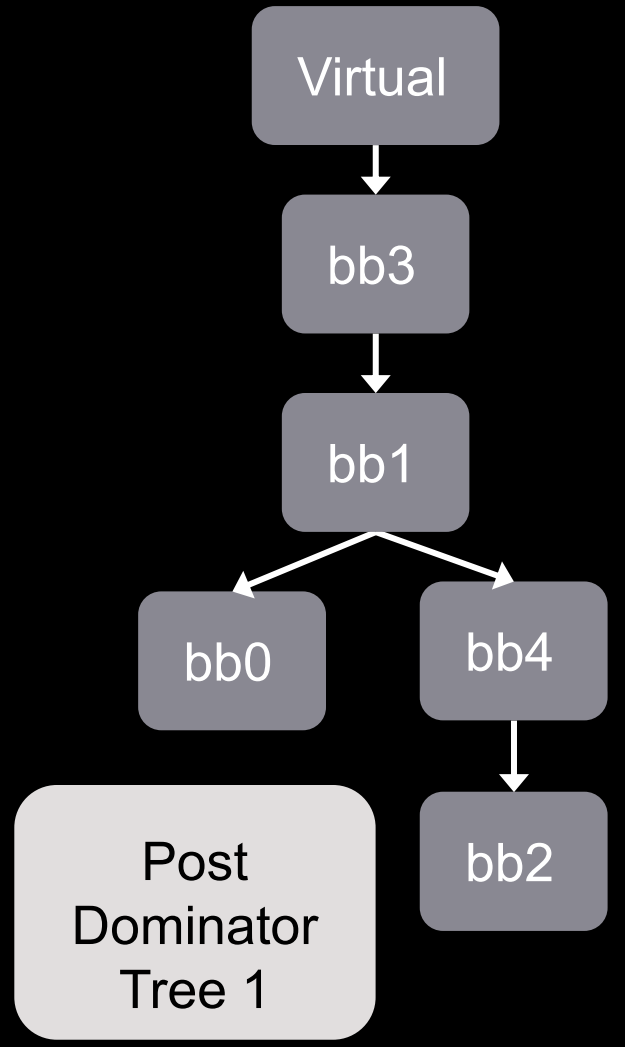


Post Dominator Tree 1

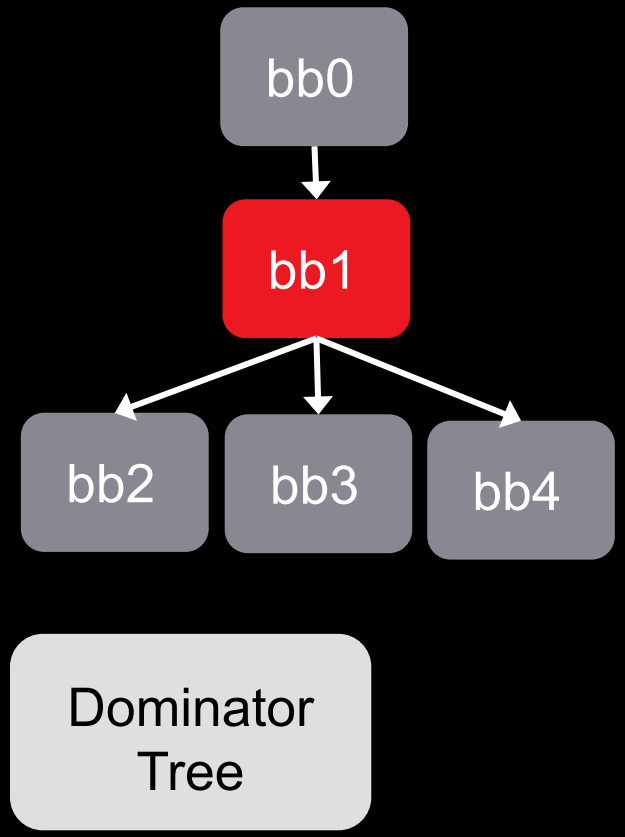
Let’s find regions

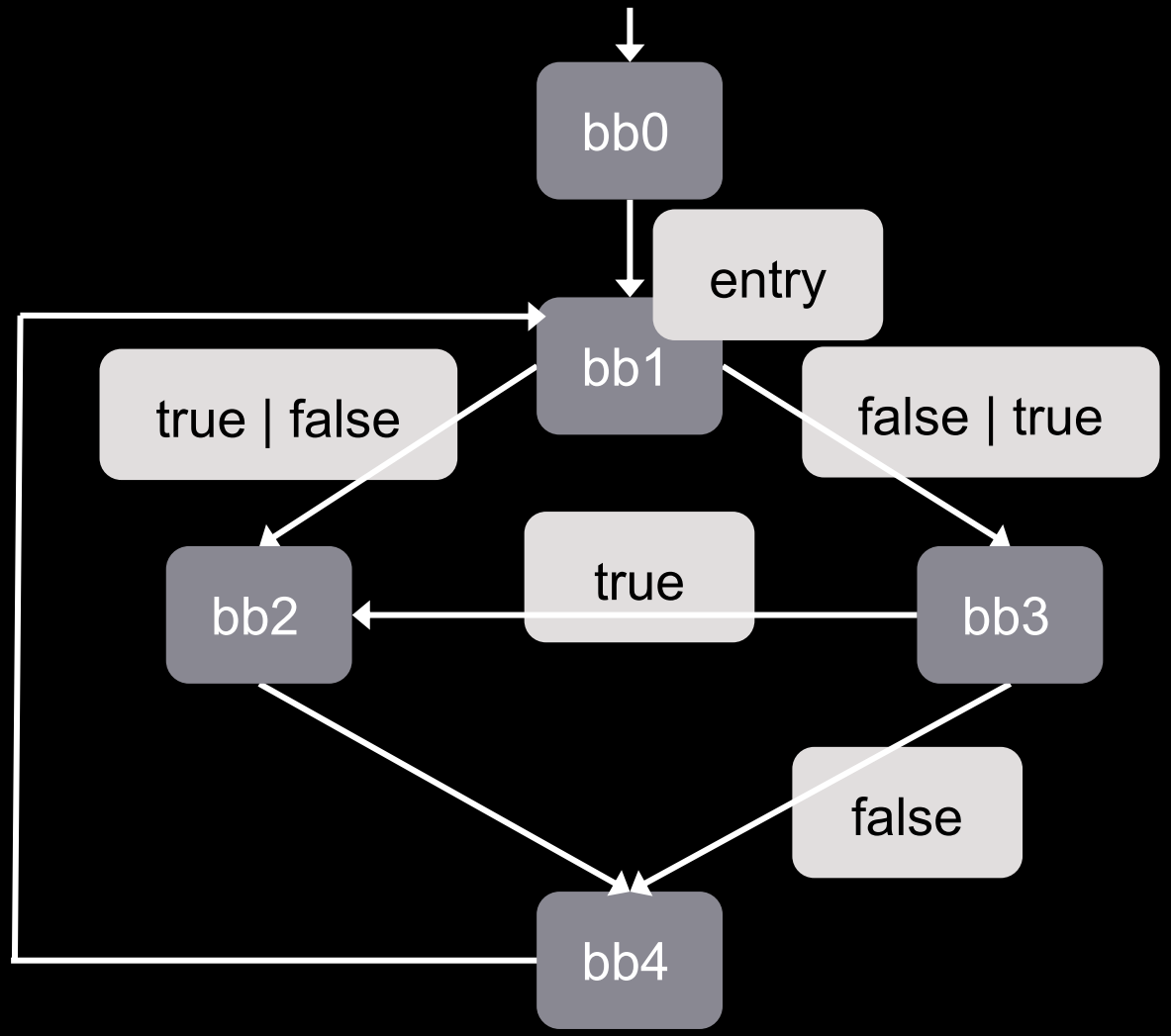


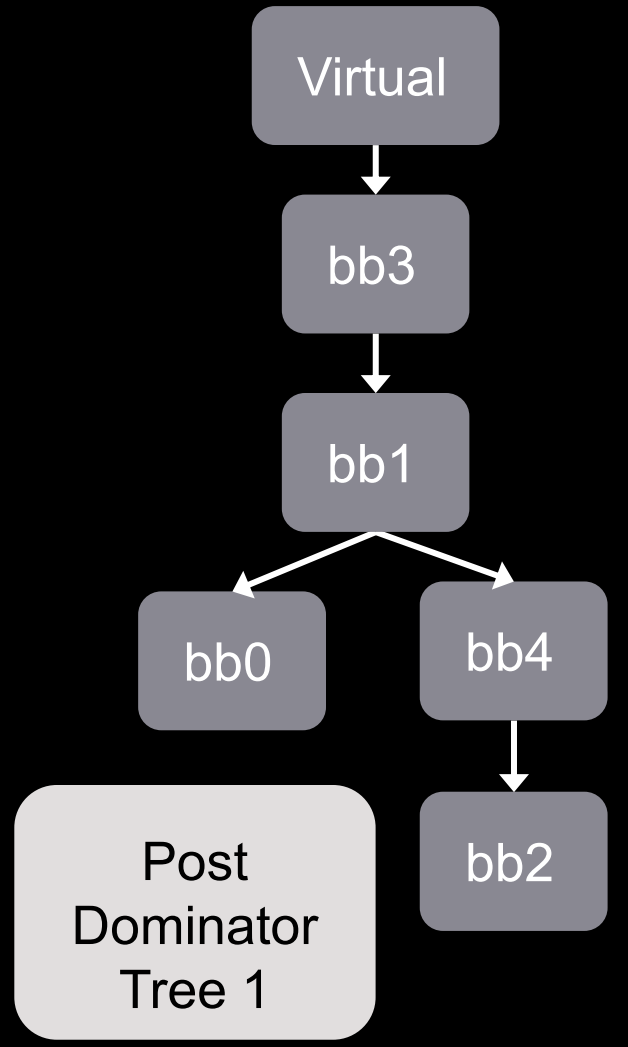




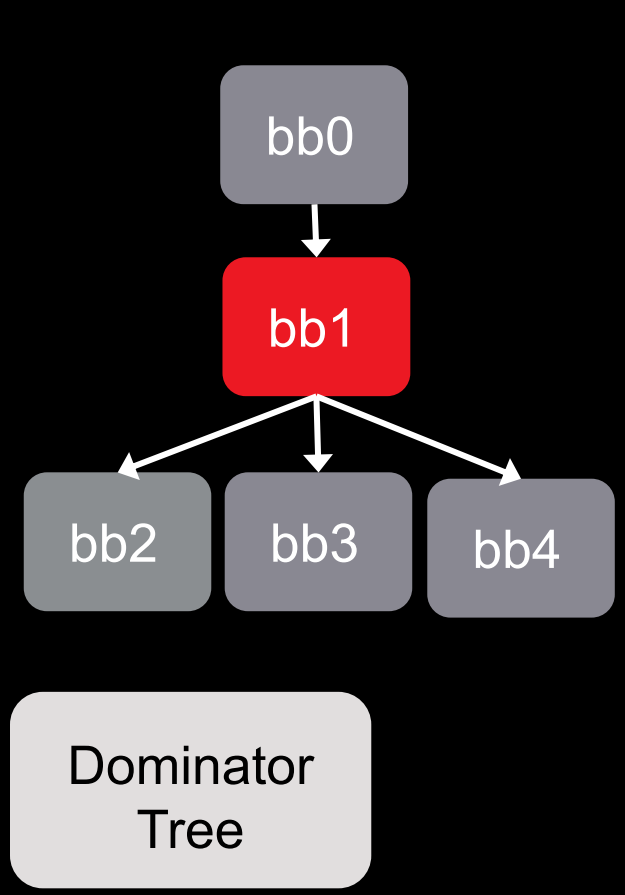
Let’s find regions

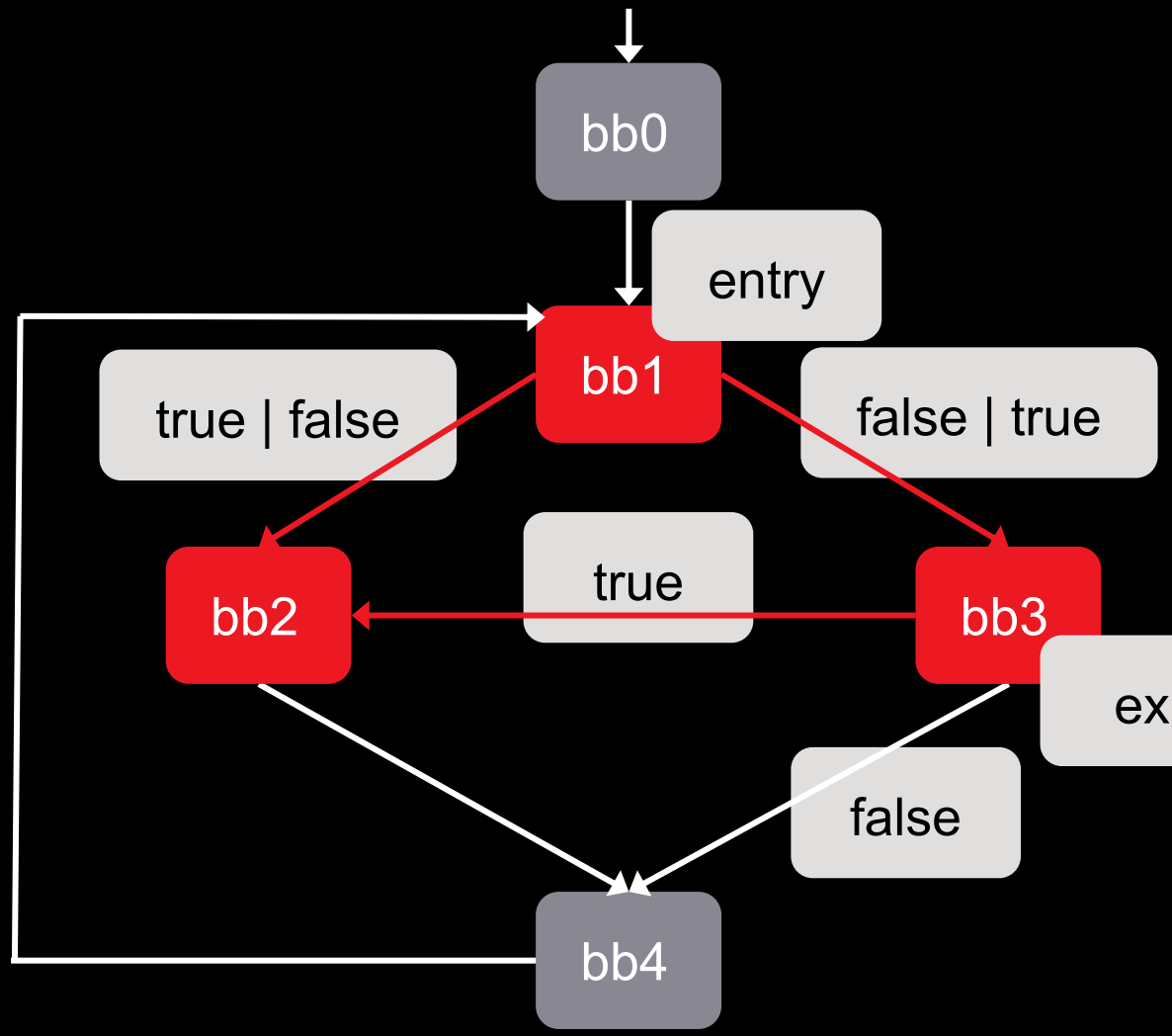


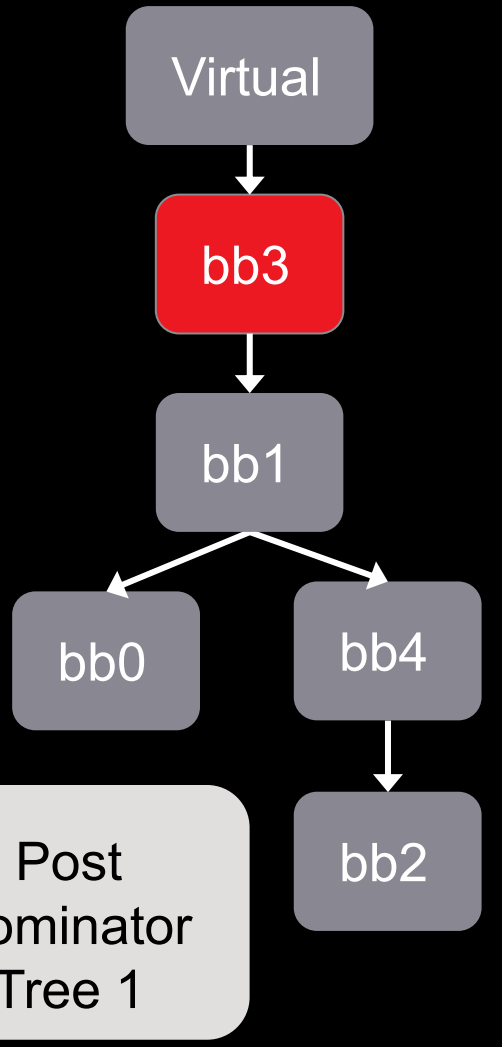




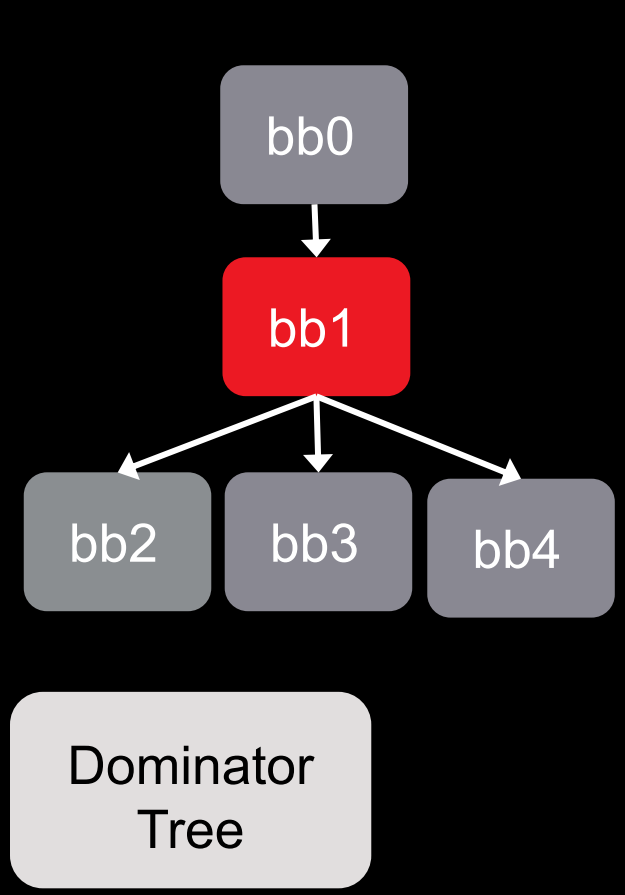
Let’s find regions

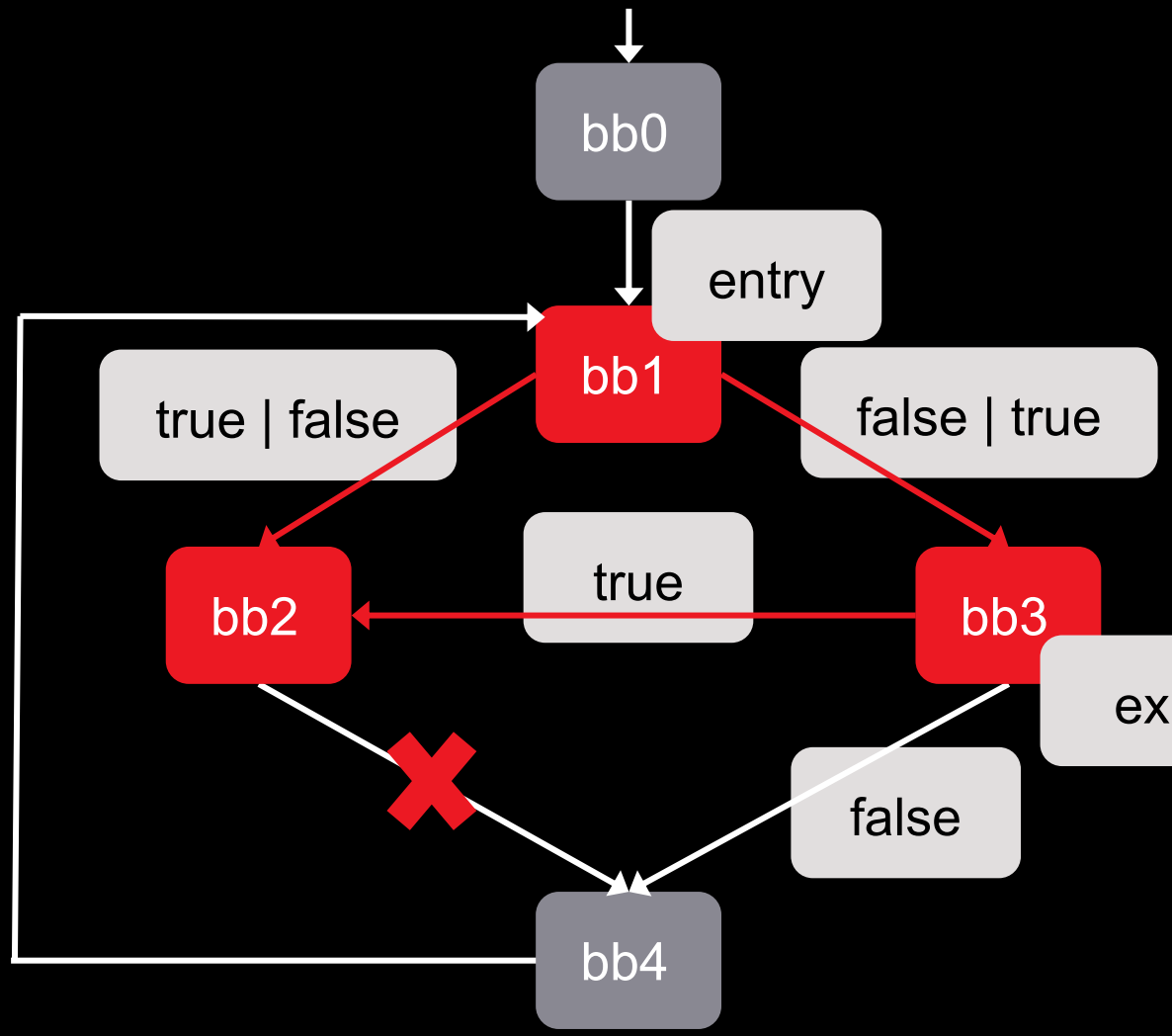


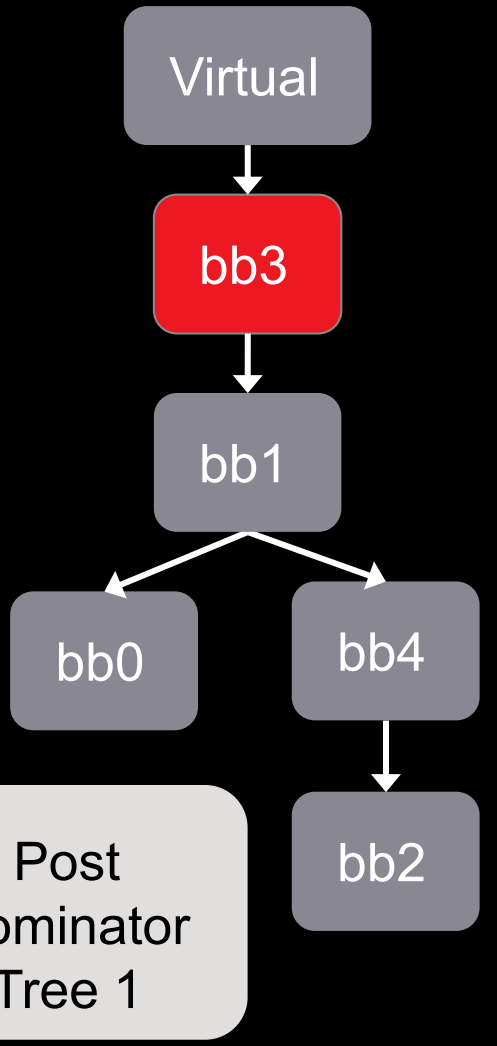




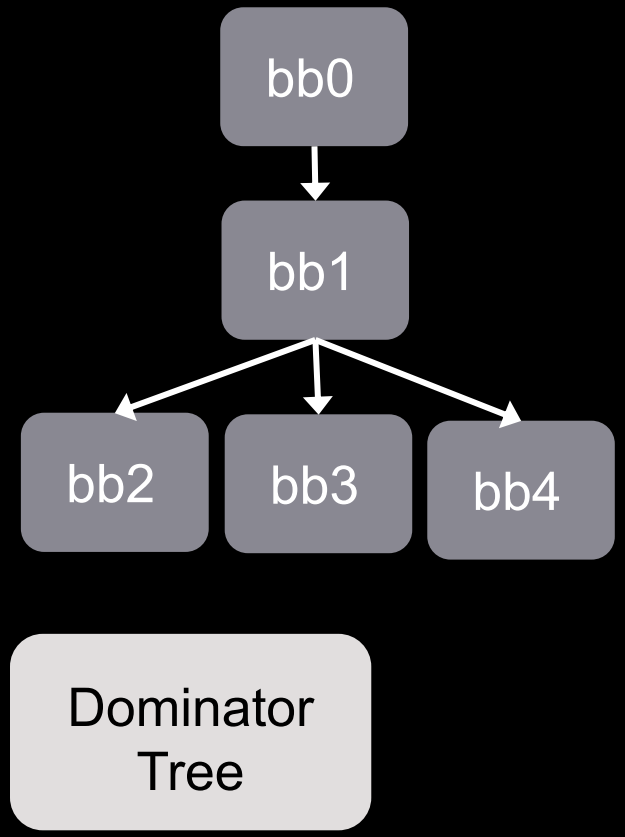
Let’s find regions

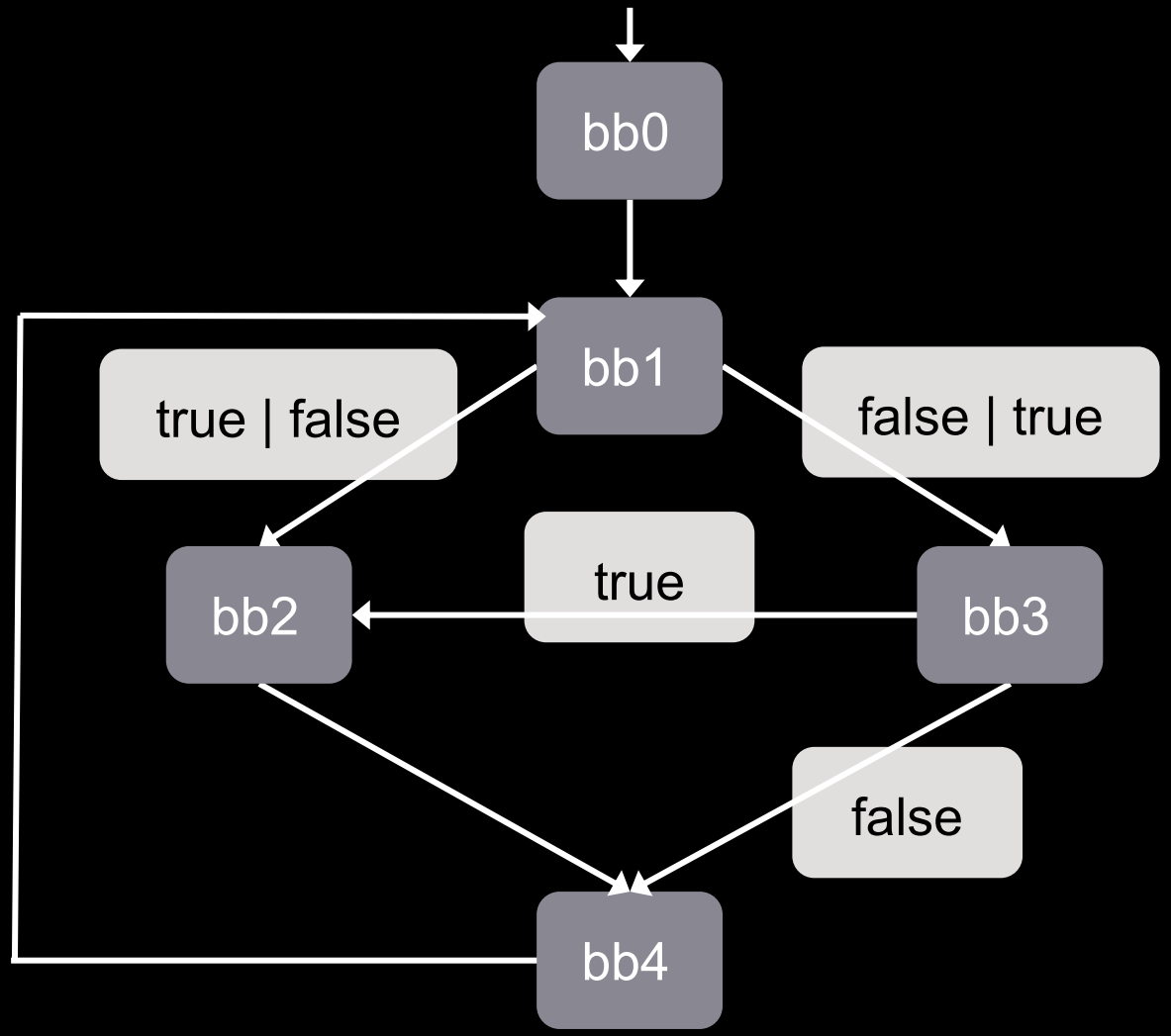


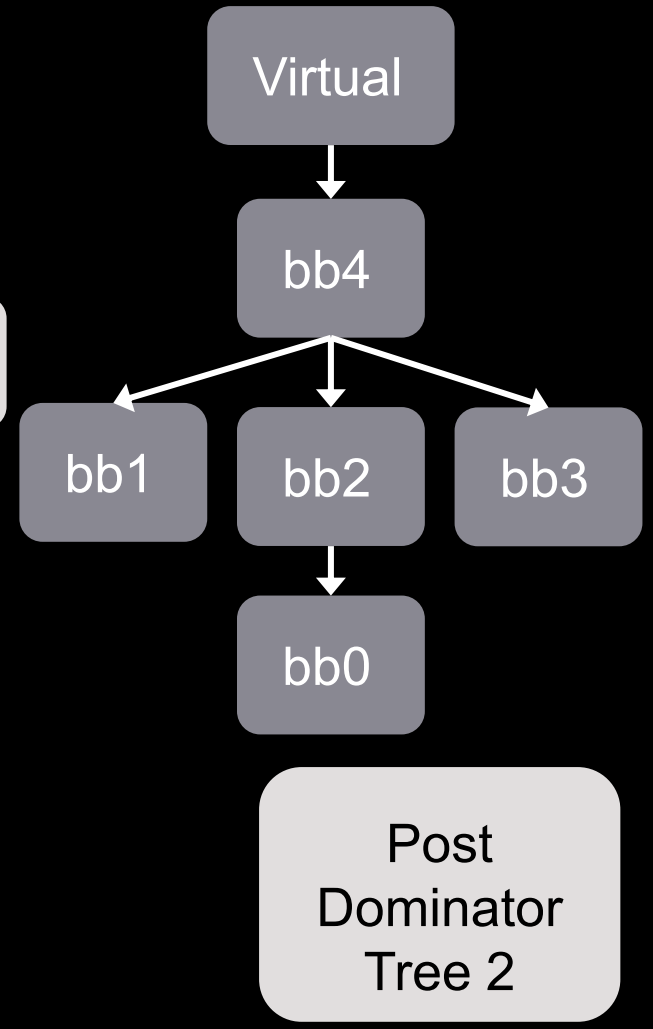




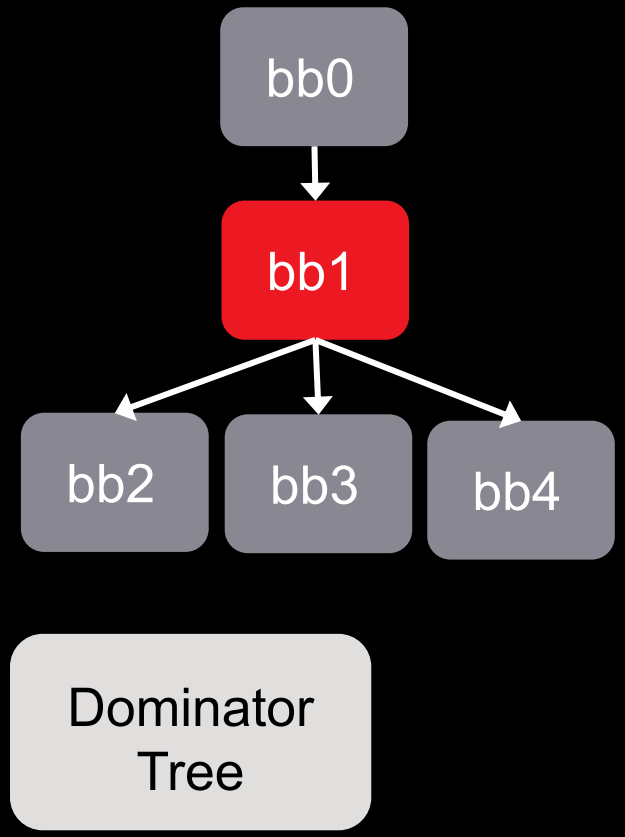
Let’s find regions

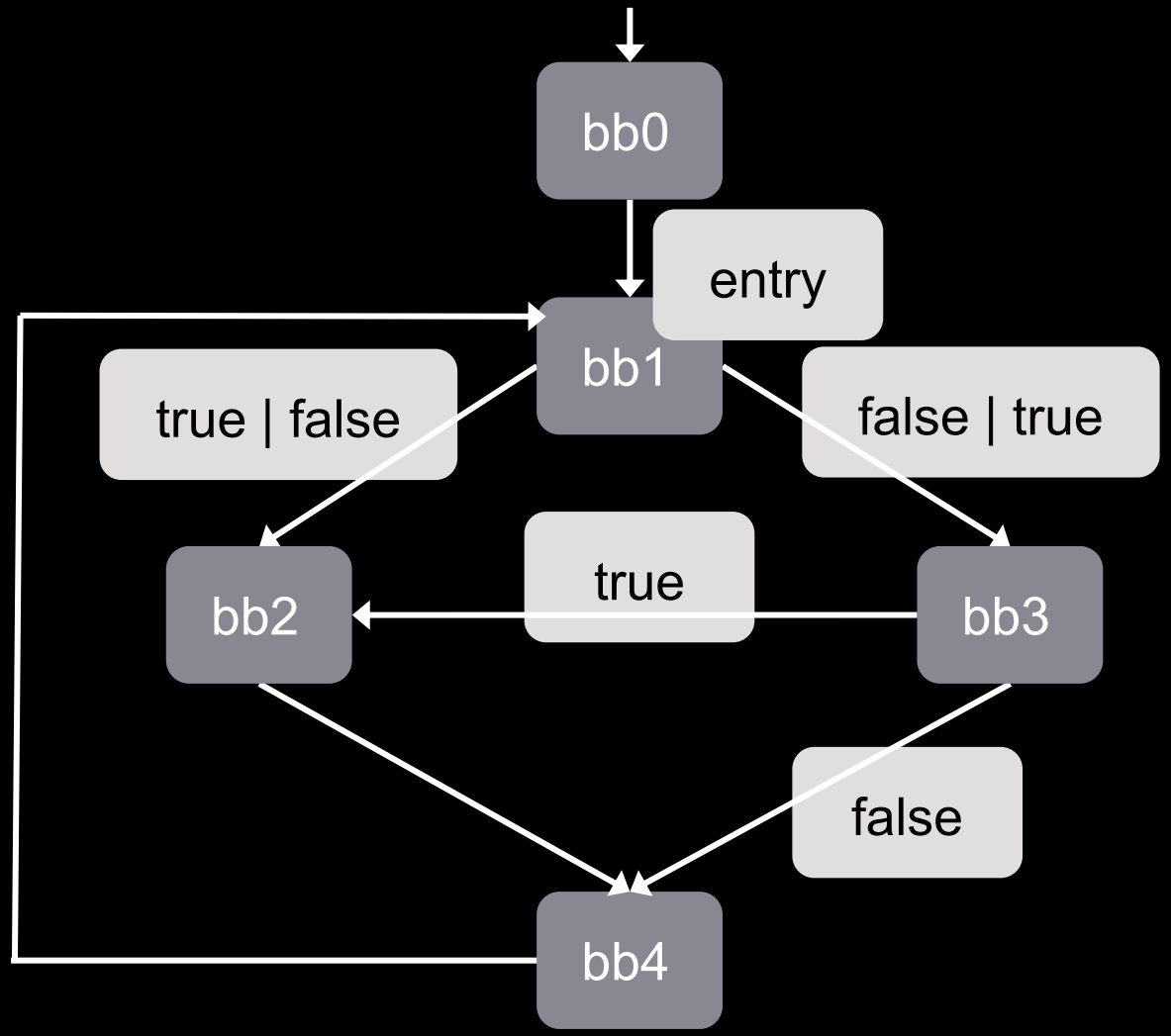


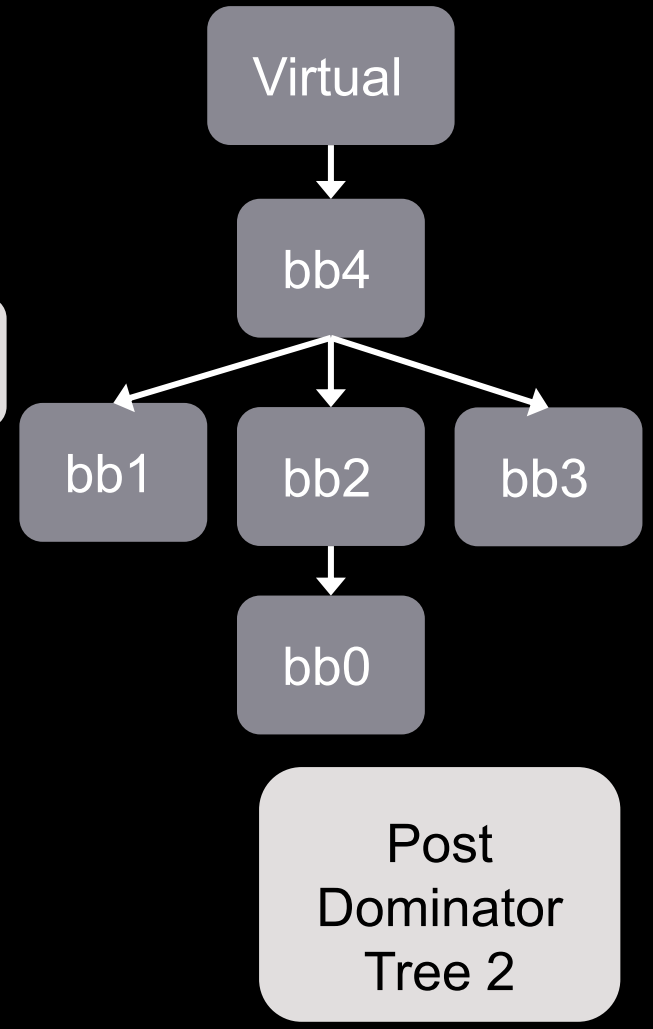




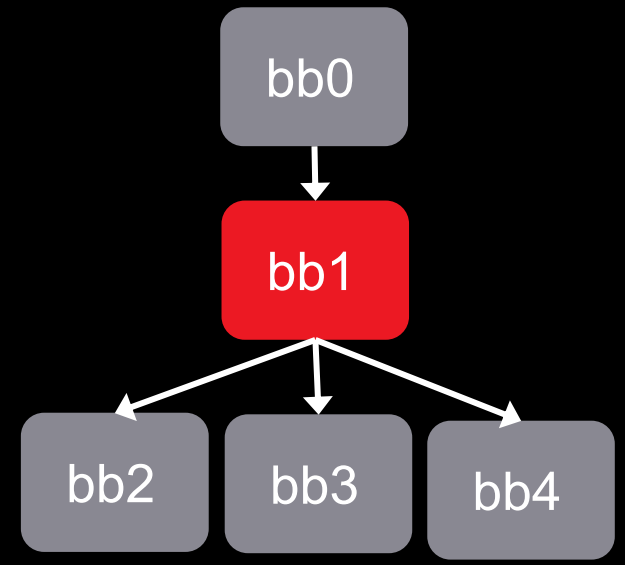
Let’s find regions



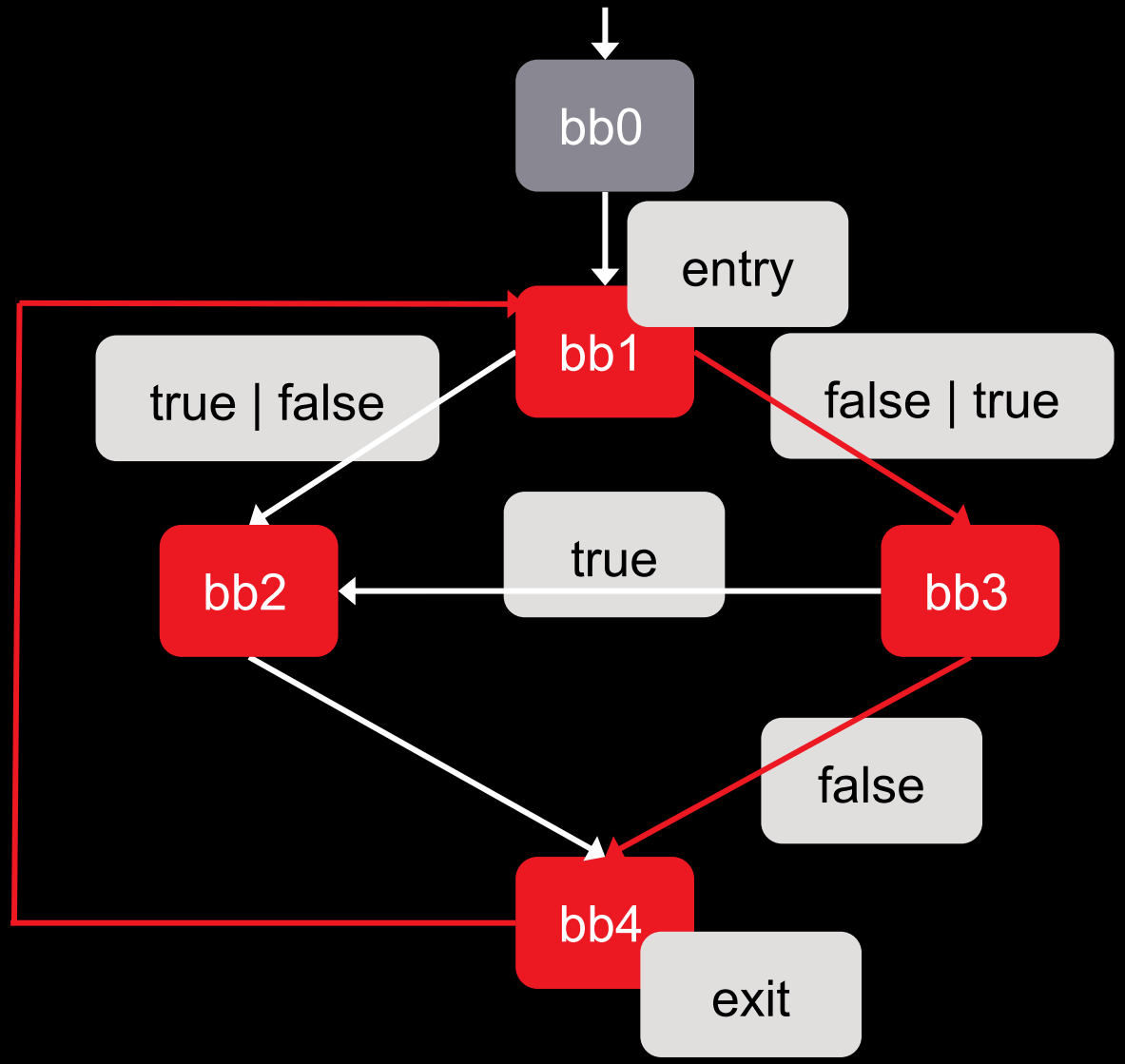


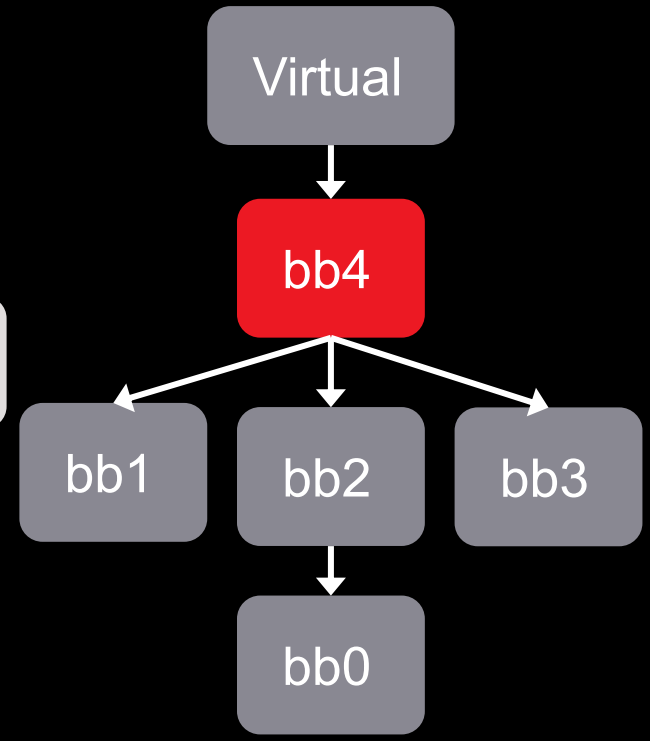


Let’s find regions



Dominator Tree





Post Dominator Tree 2

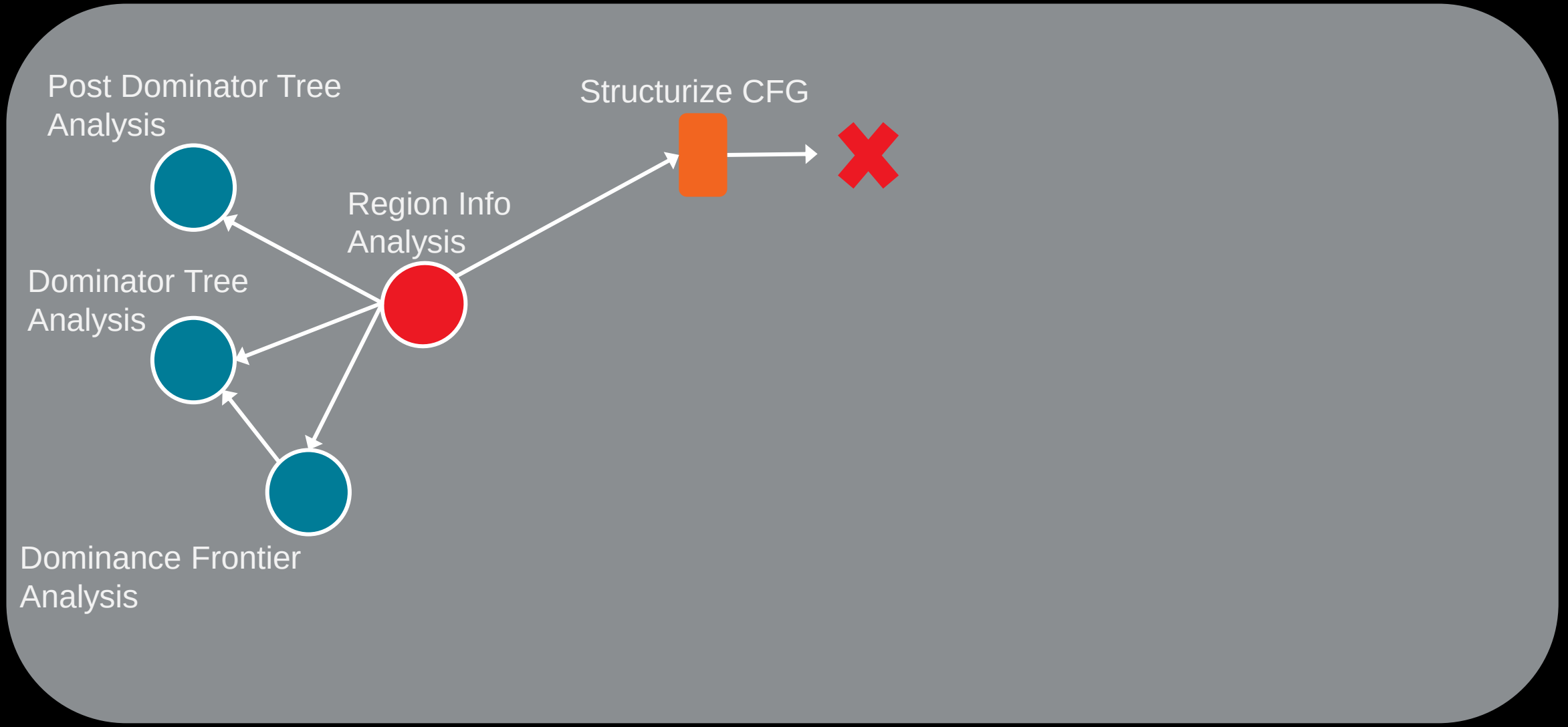
AMD

What went wrong?

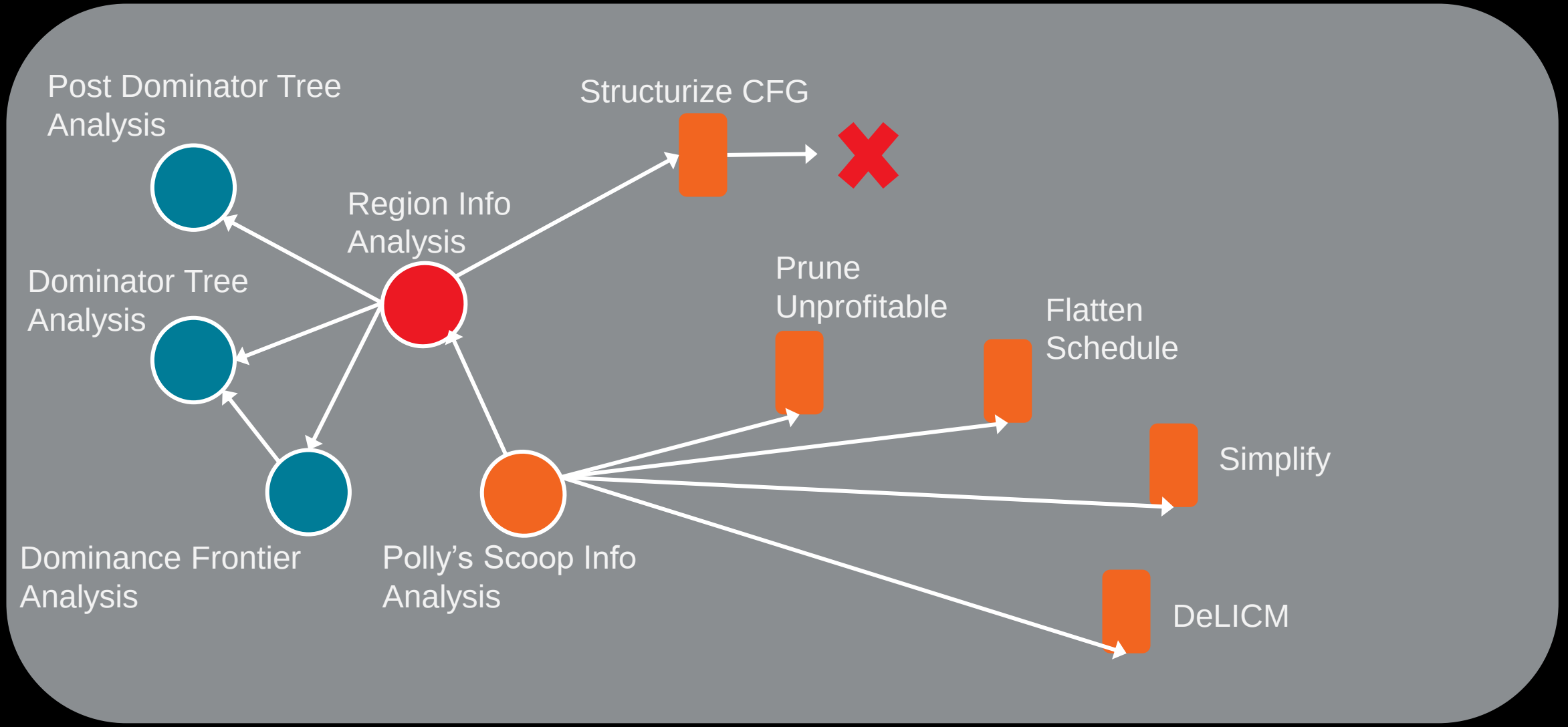
Region analysis did the right thing at its end

Post dominator analysis also did the right thing at its end

How bad is missing a region?



How bad is missing a region?



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!

DISCLAIMER

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18


AMD