Mutation Testing and Automated Program Improvement

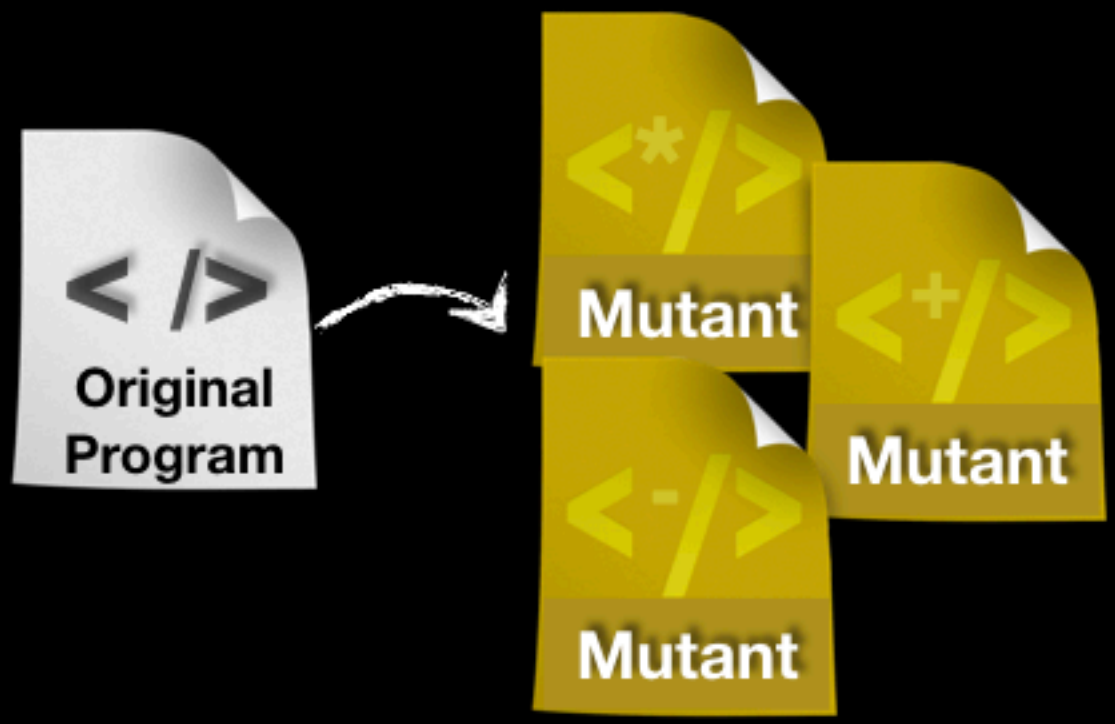
Mike Papadakis
University of Luxembourg
SnT Center

58th CREST Open Workshop

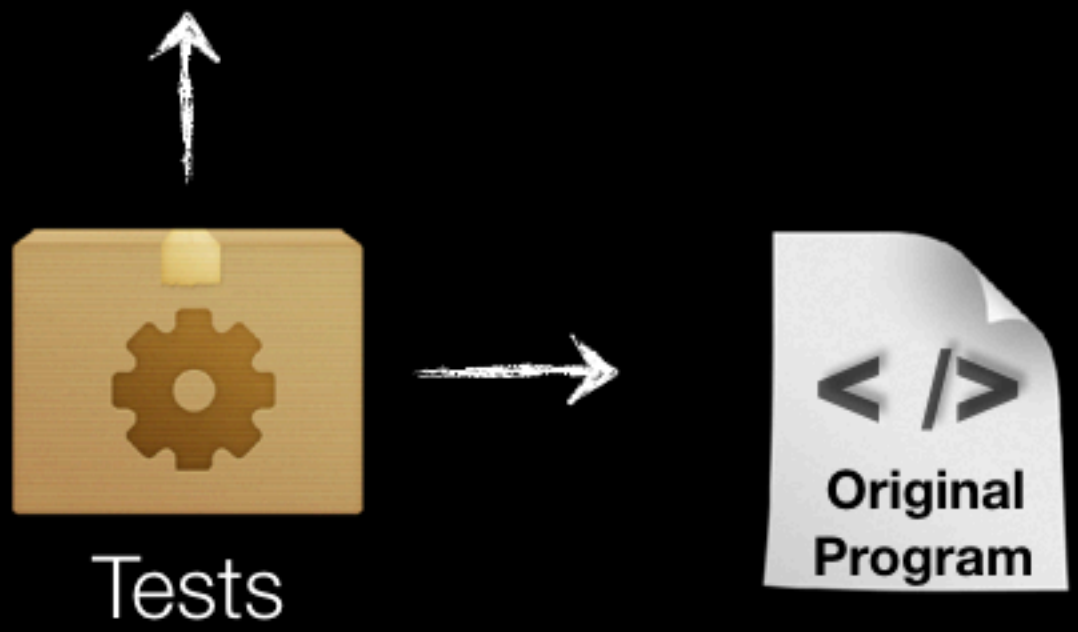
*Part of the presentation were made by Dr. Yue Jia

Mutation Testing

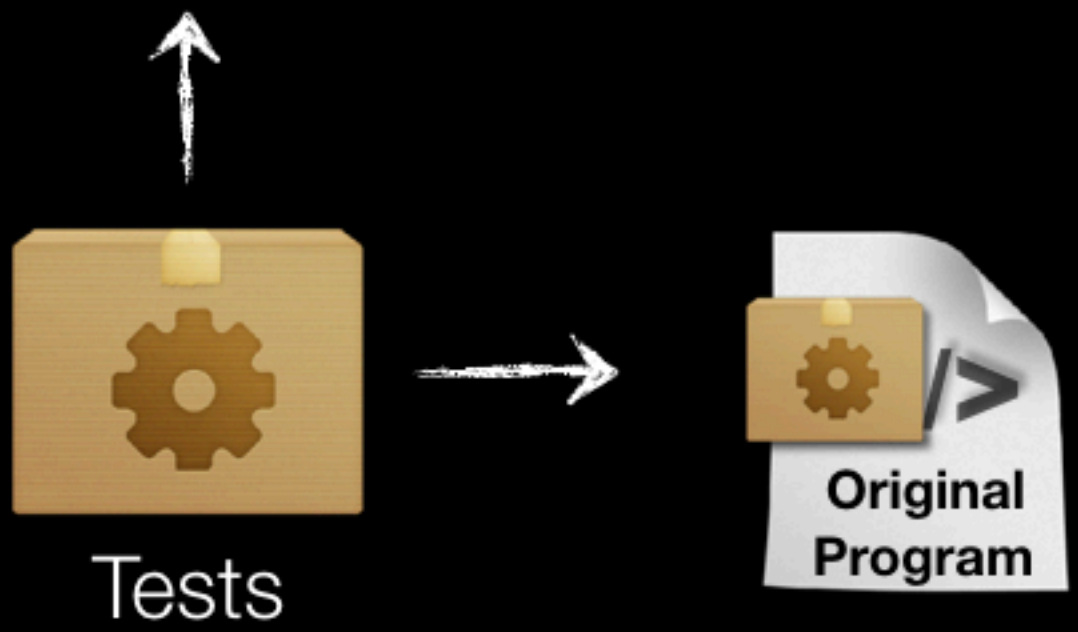
Mutation Testing



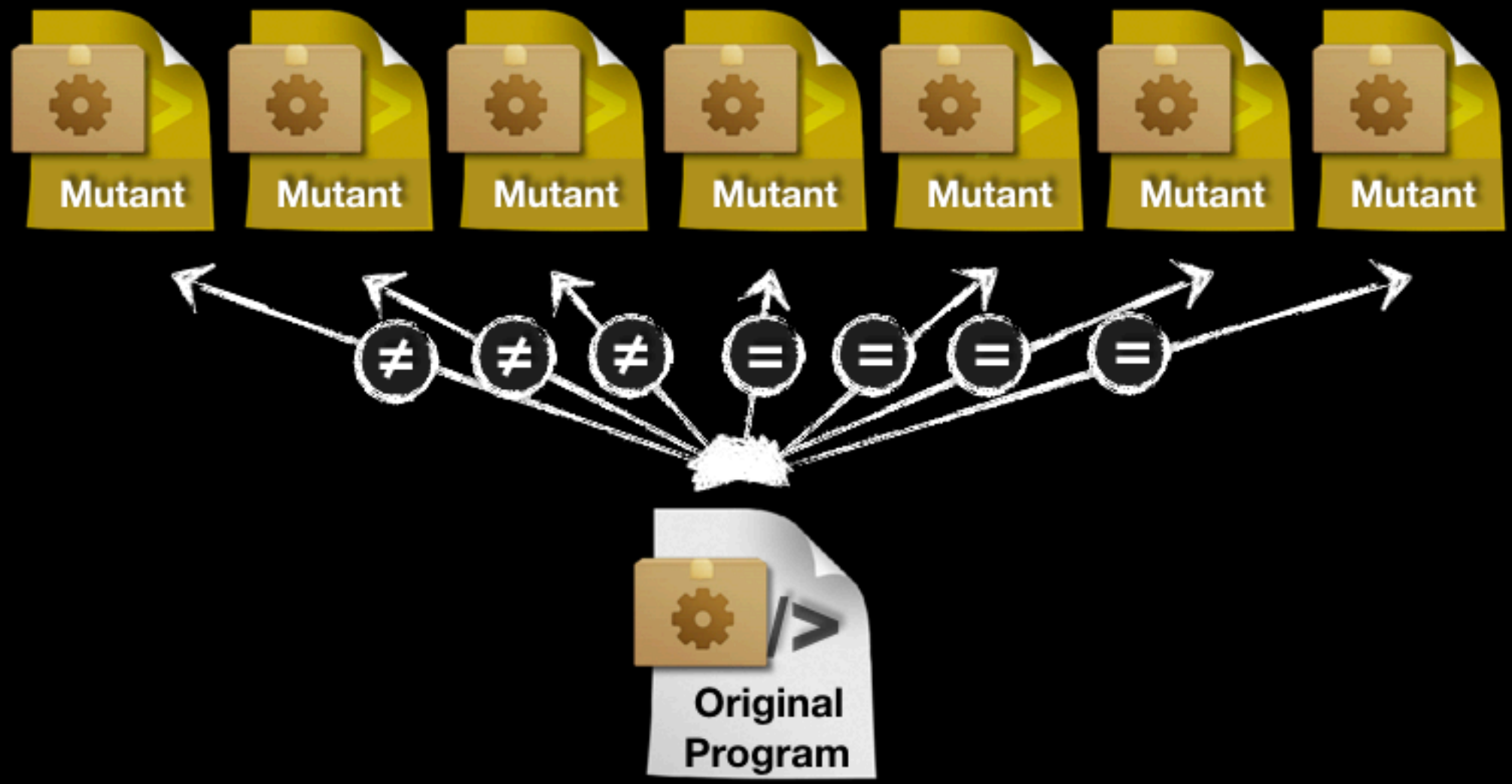
Code-based Mutation Testing

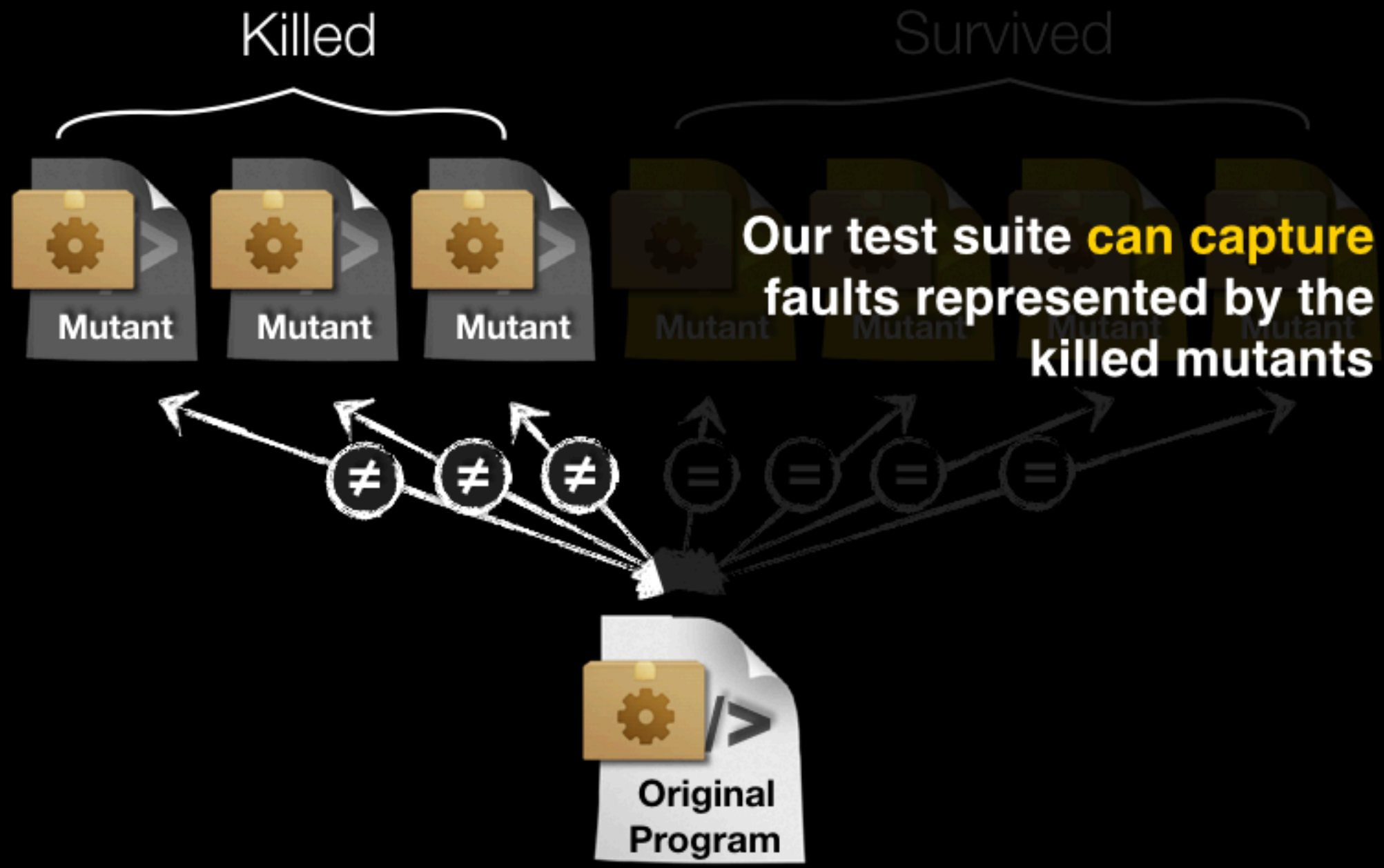


Code-based Mutation Testing



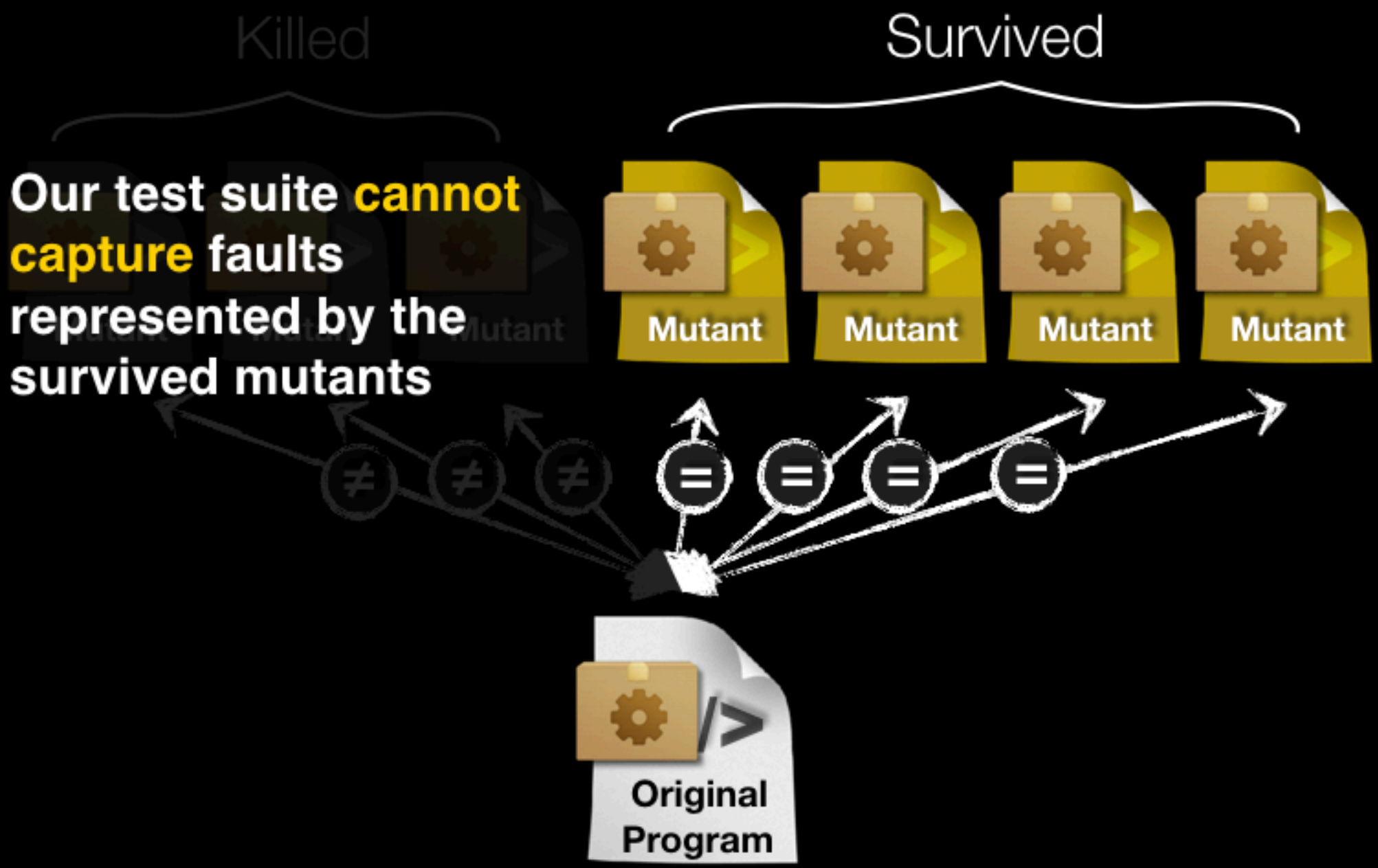
Code-based Mutation Testing



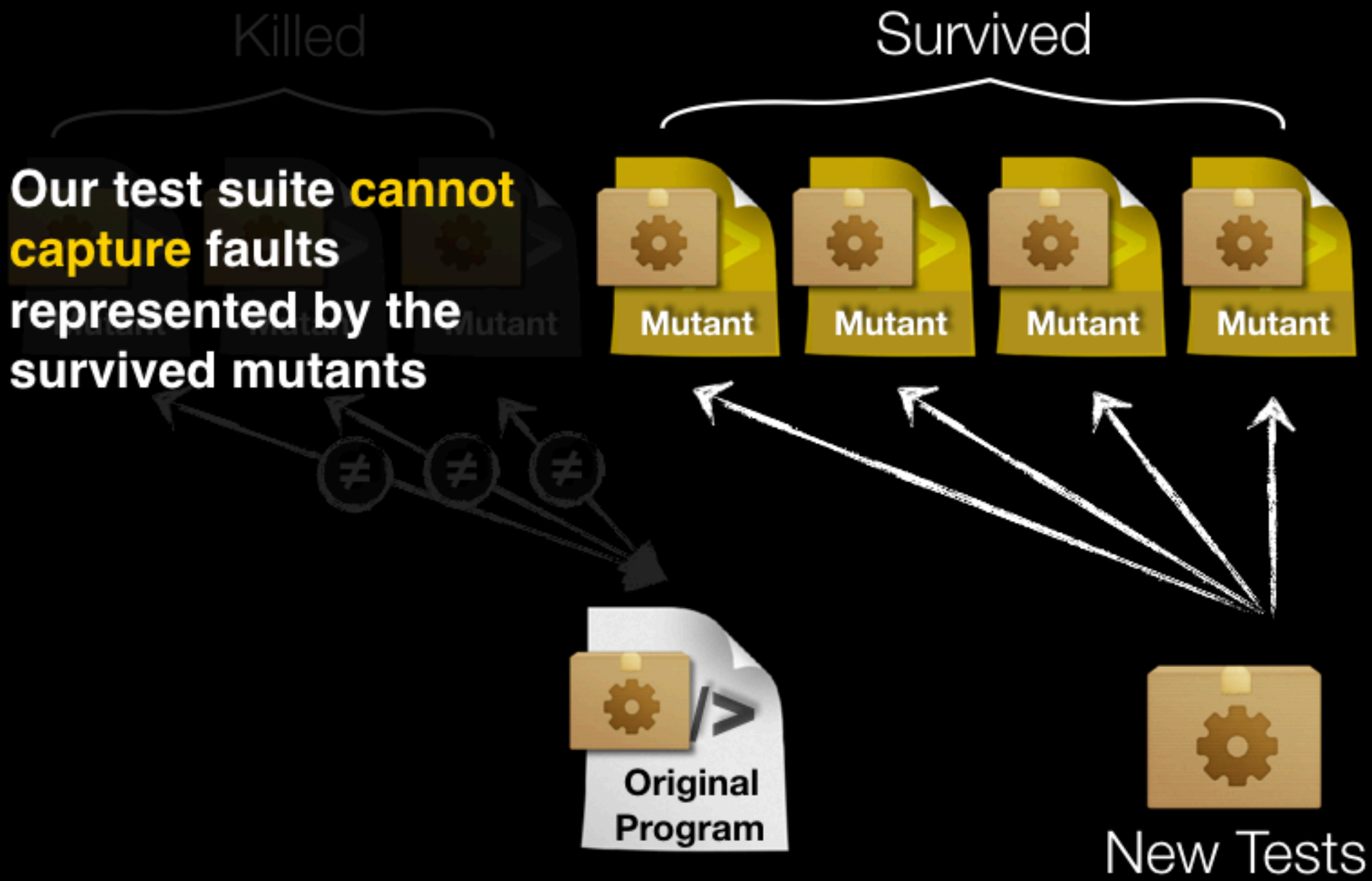


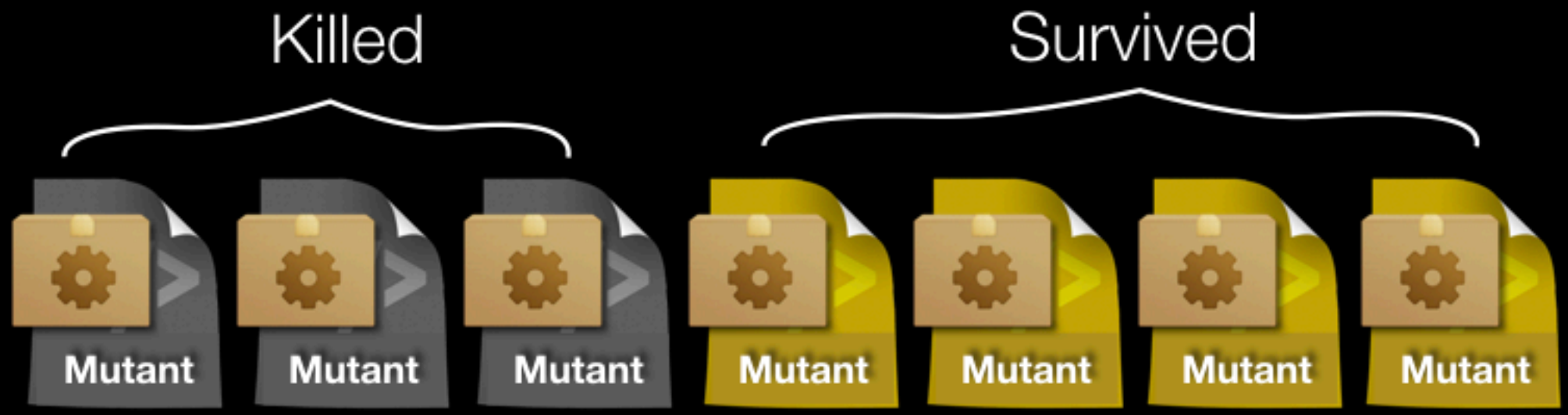
Our test suite can capture faults represented by the killed mutants

Our test suite **cannot capture** faults represented by the survived mutants



Our test suite **cannot capture** faults represented by the survived mutants





$$\text{Mutation Score} \approx \frac{\#\ \text{Killed}}{\#\ \text{All}}$$

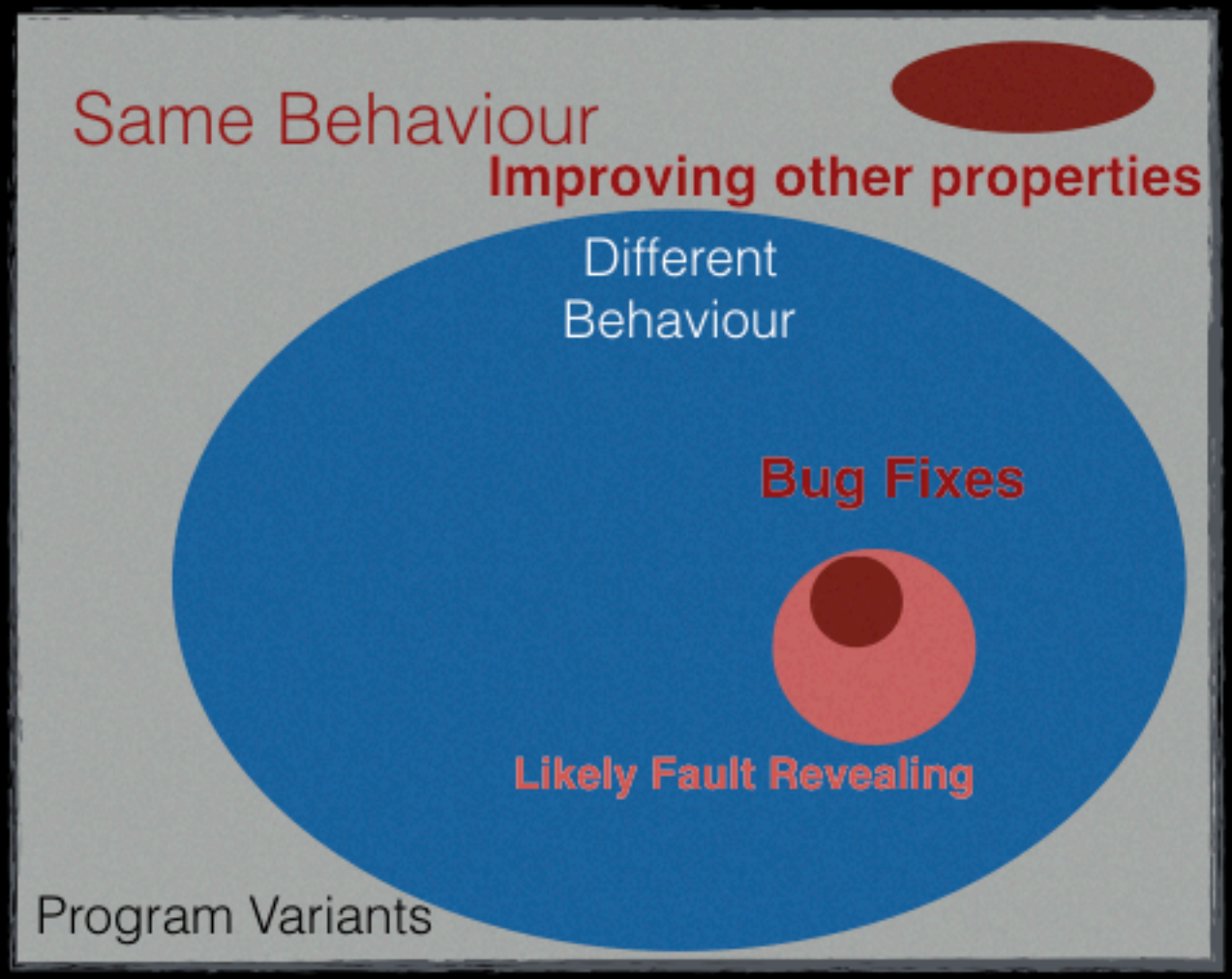
$$\approx 0.43$$

Mutation Testing: Focuses on the behaviour differences between program variants

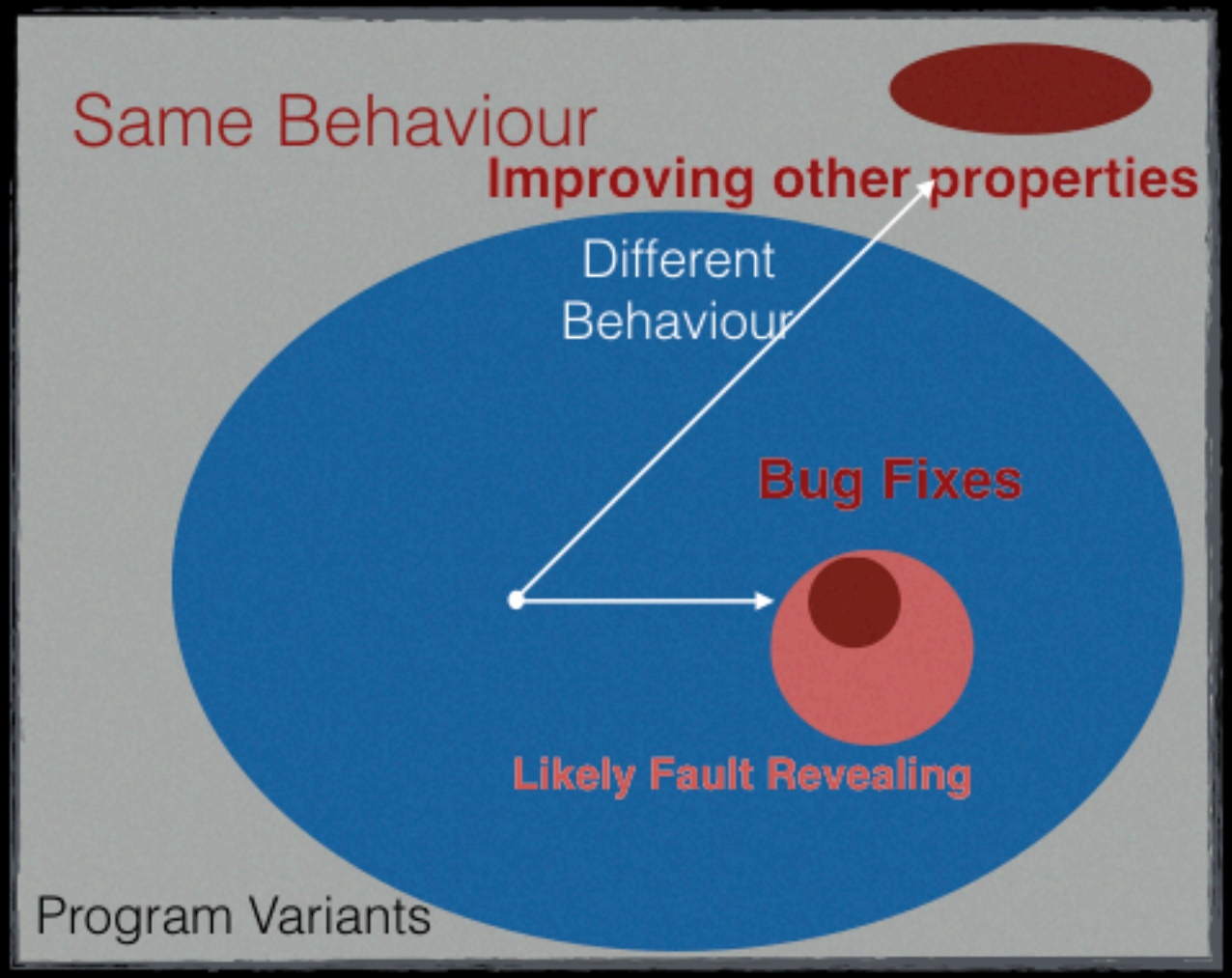
A **good** mutant is one that **leads** to test cases that reveal **erroneous** behaviour (of the original program)

Program Improvement: Focuses on the program variants with the “same” or “similar” behaviour

Mutation Testing and Automated Program Improvement

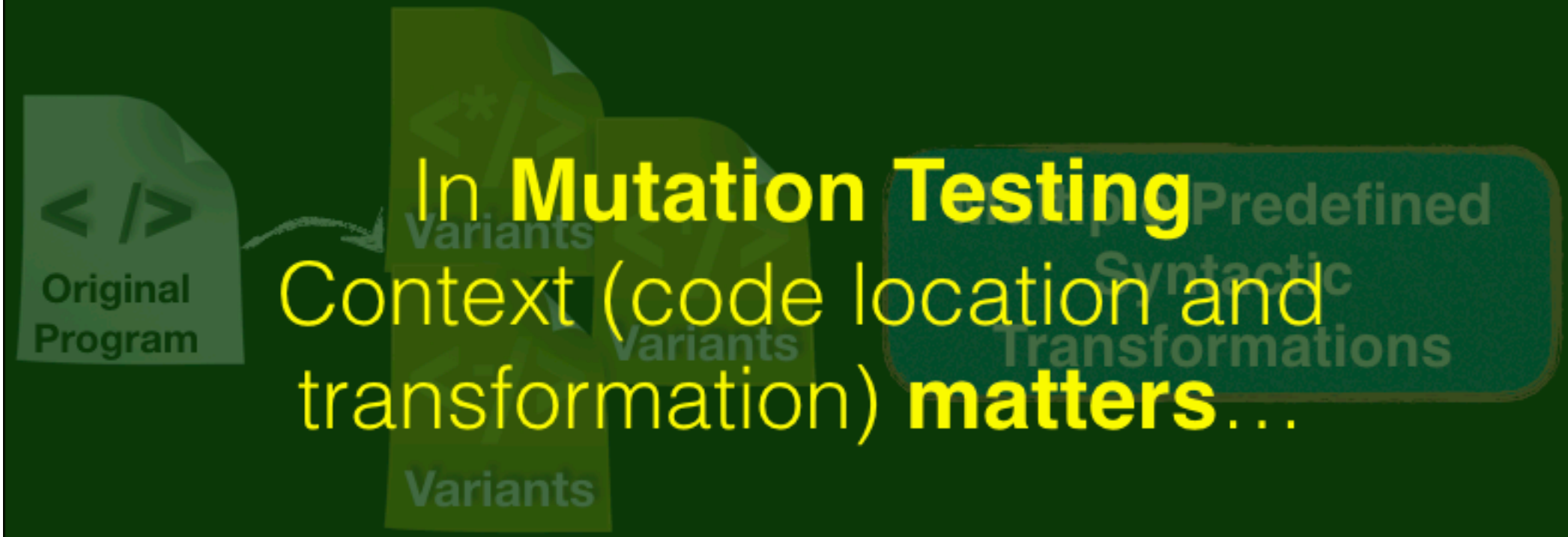


Mutation Testing and Automated Program Improvement



Feasibility and **Scalability** issues are the same!

Automated Program Improvement



In **Mutation Testing** Context (code location and transformation) **matters**…

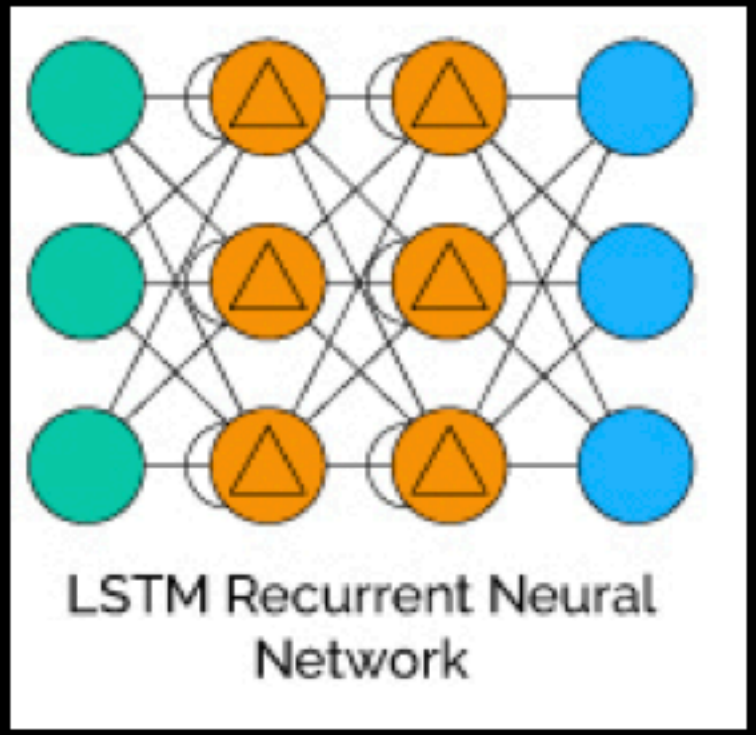
Improbable or surprising code is likely to be problematic

Improbable code snipets

More probable code snipets

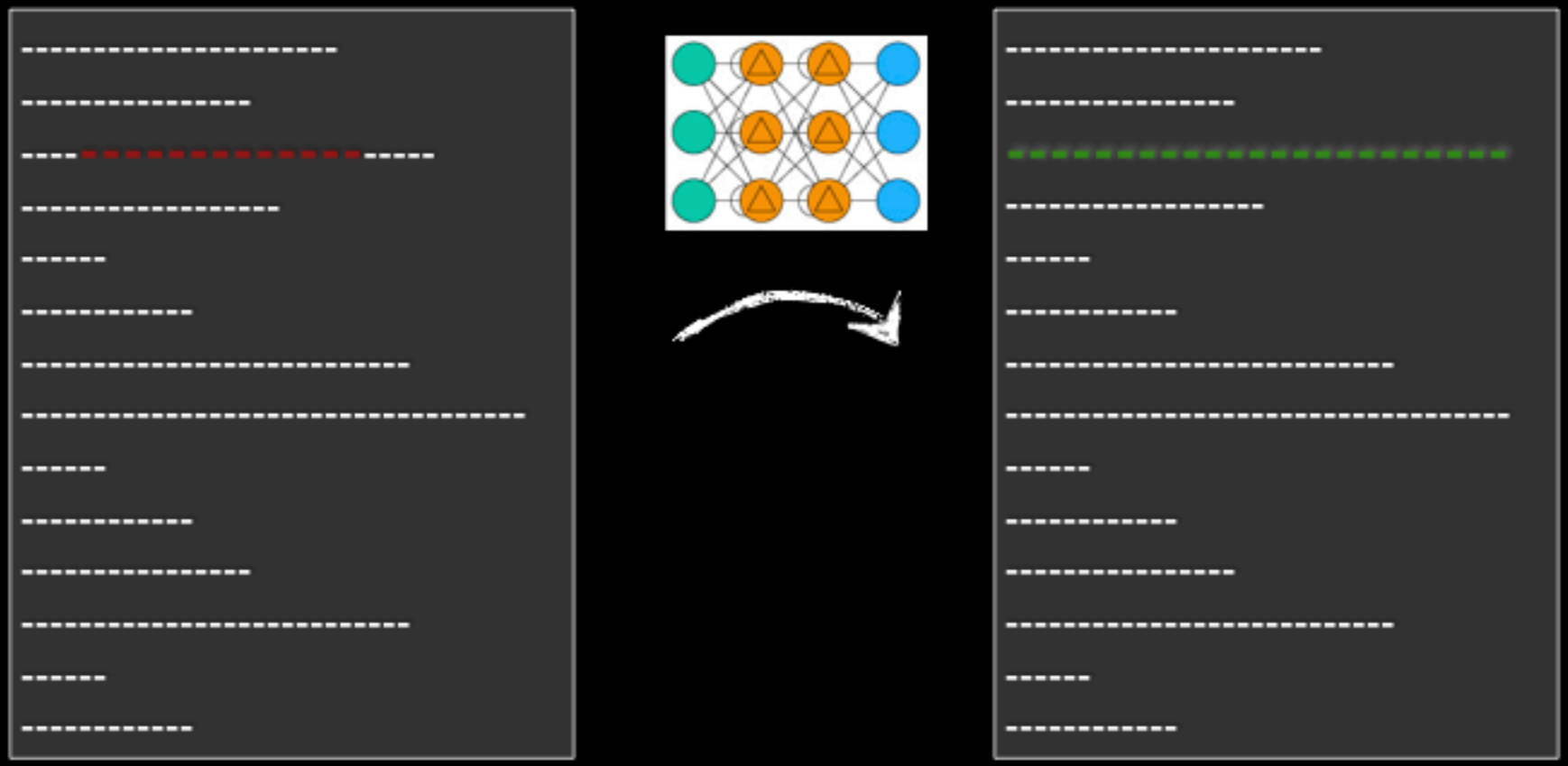
Ray et al. **On the "naturalness" of buggy code** ICSE'16

Build a model (learn from existing code)



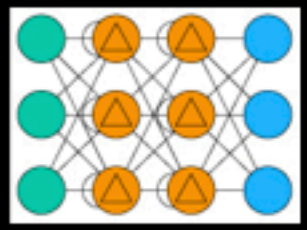
Mutations-Transformations

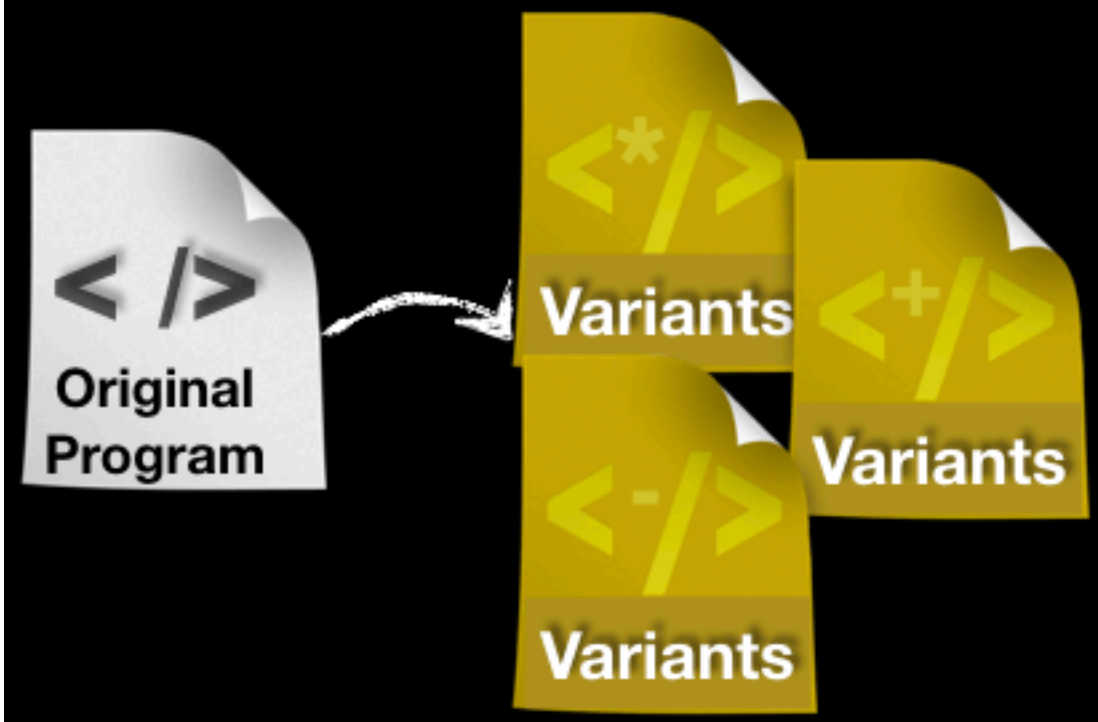
Mutations-Transformations



Mutations-Transformations

Mutations-Transformations

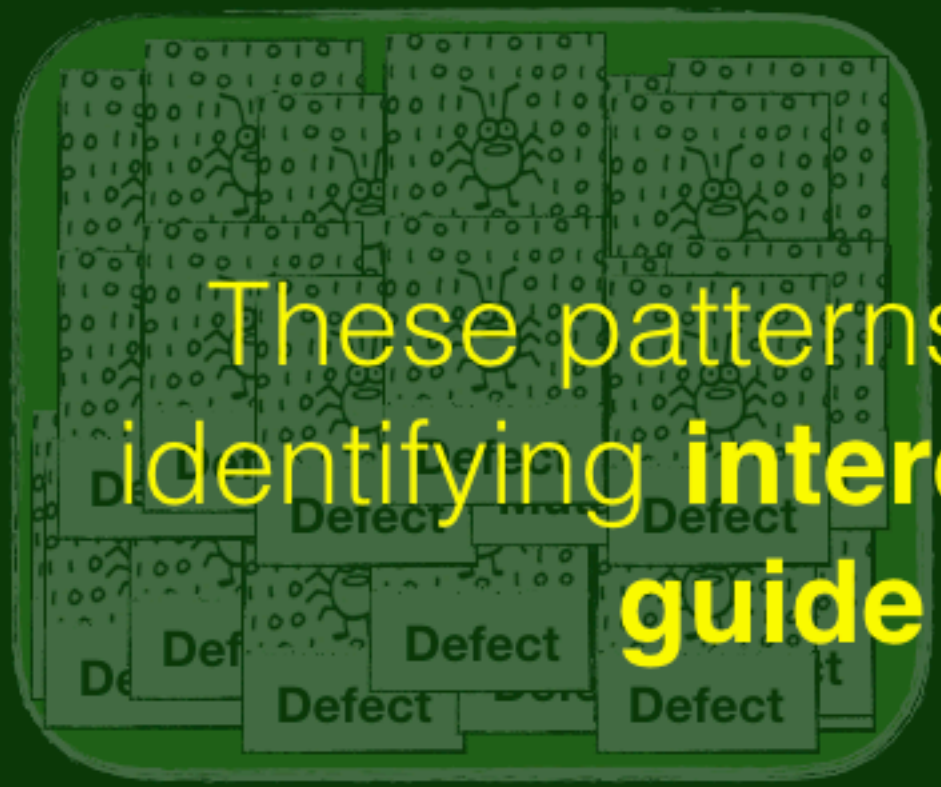




Multiple Predefined Syntactic Transformations

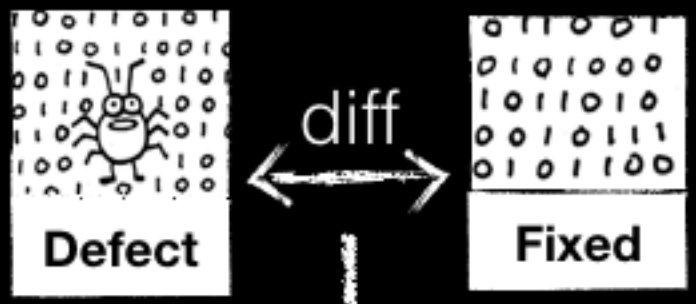
Tailored Mutations

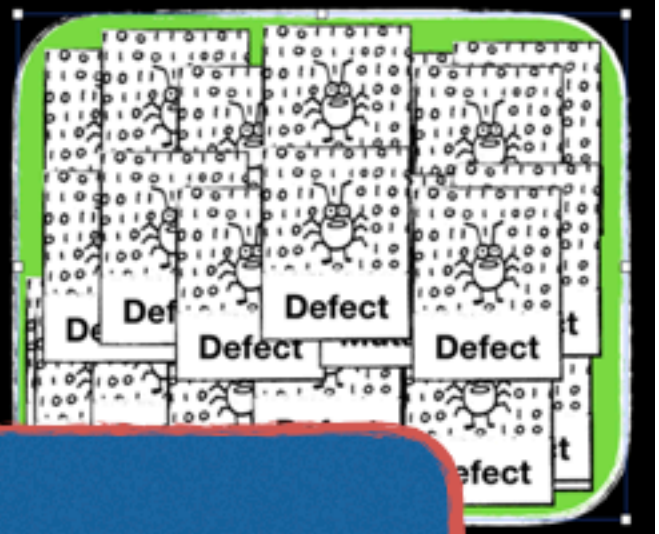
Code Locations & Guidance…



These patterns can be used for identifying **interesting locations** & **guide** evolution

Patterns

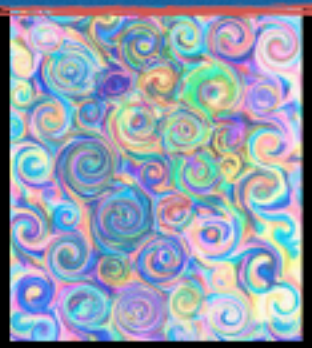




Patterns are described in terms of

Control & Data Dependencies

AST Graph elements



Pattern

GP

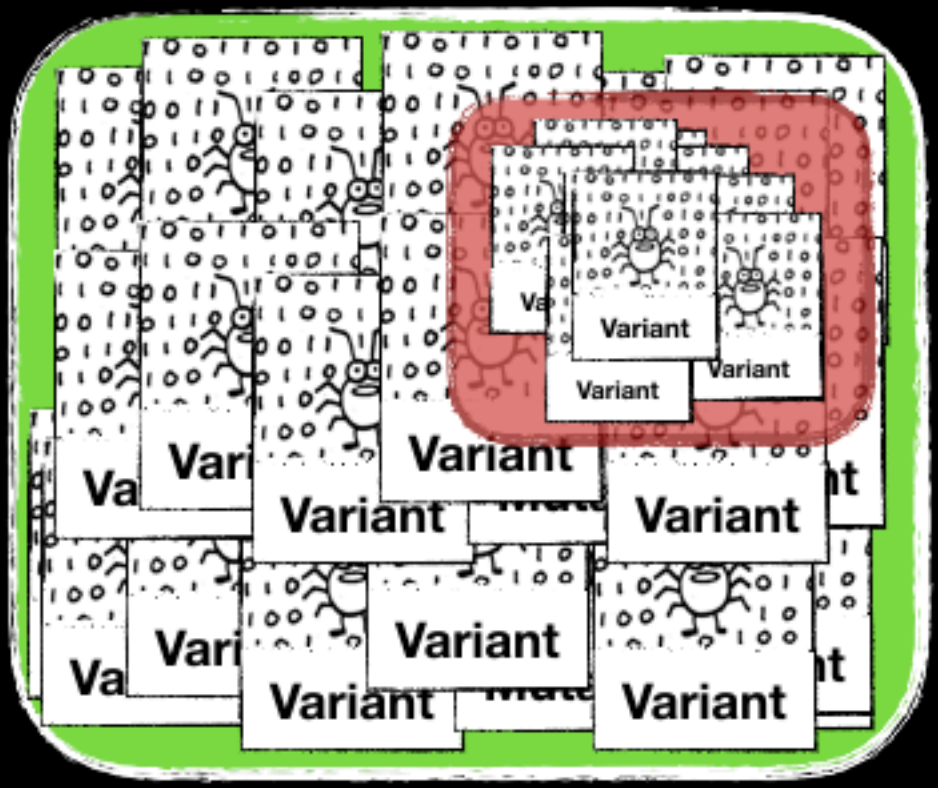
Selection Problem

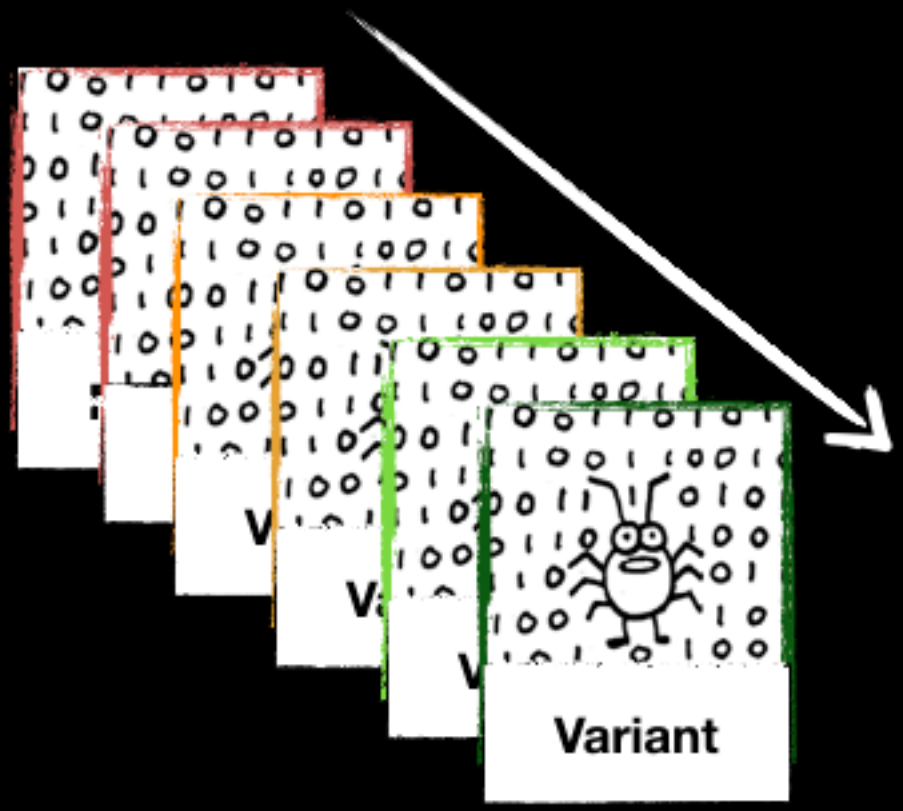
In mutation testing only **a few mutants** (approximately 3%) are interesting…

Problem

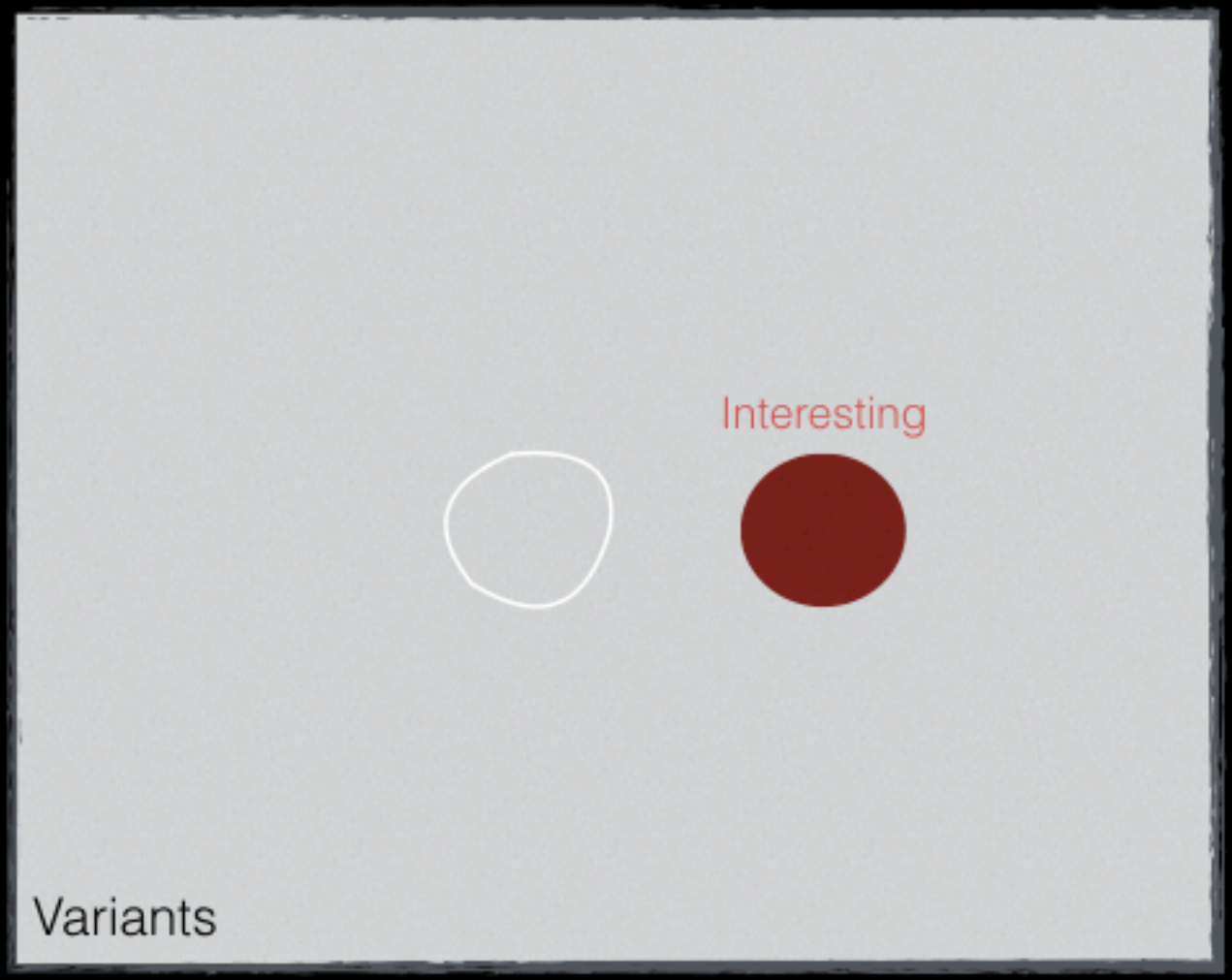
Set selection problem

Prioritisation Problem



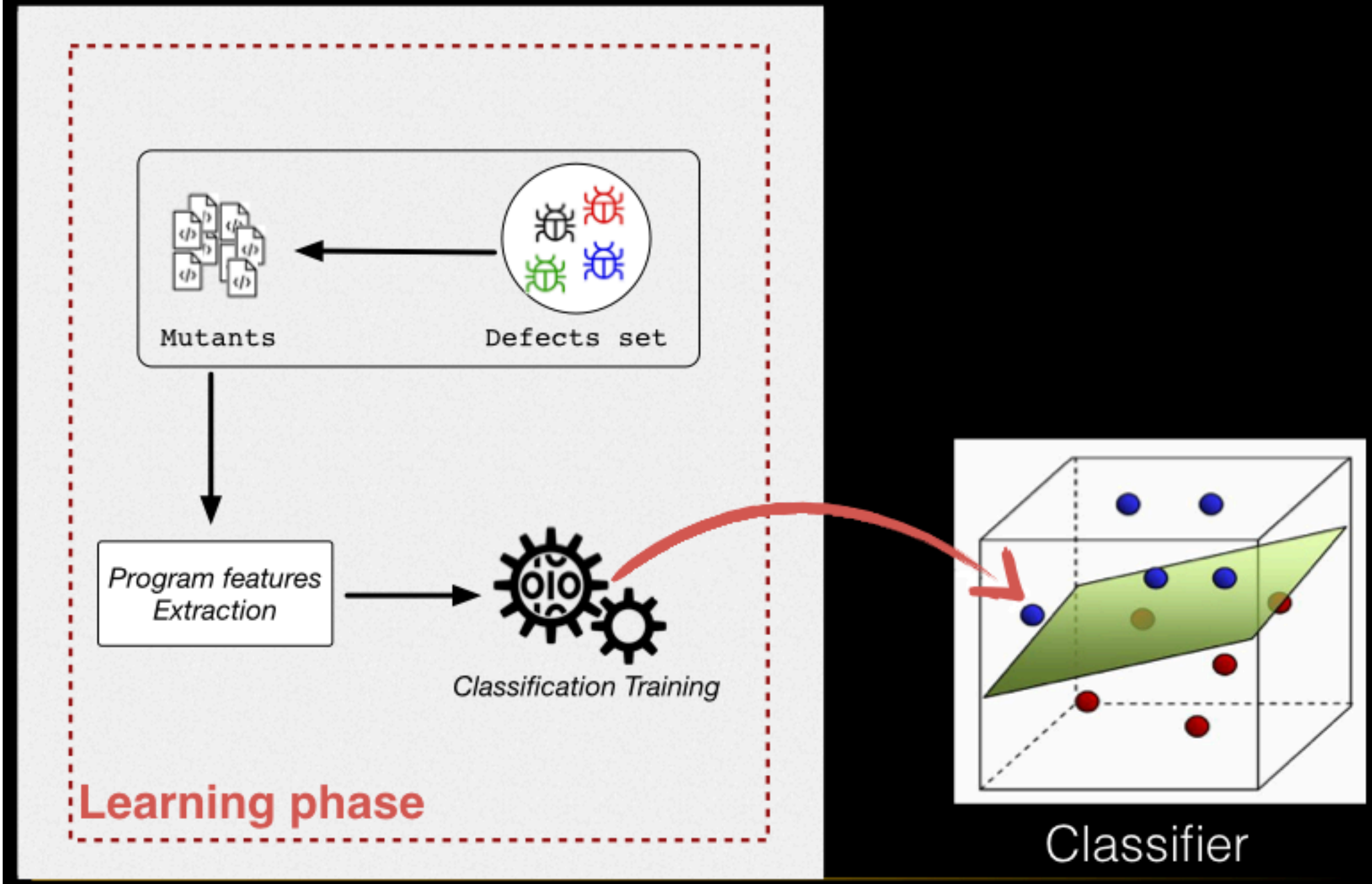


Selection Problem

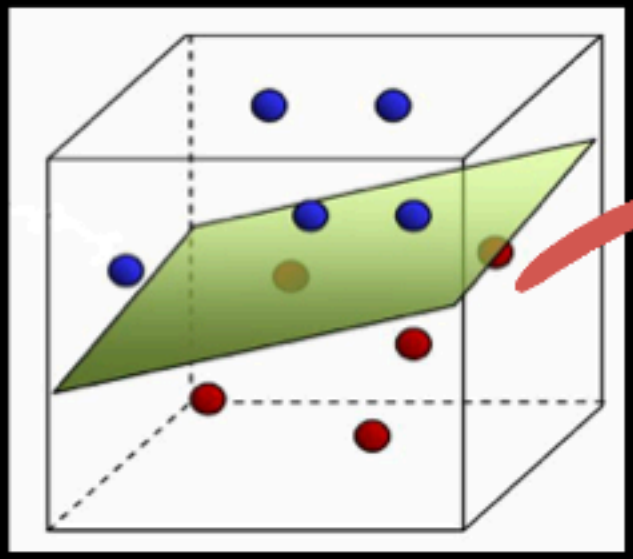


Prediction Modelling

Machine Learning



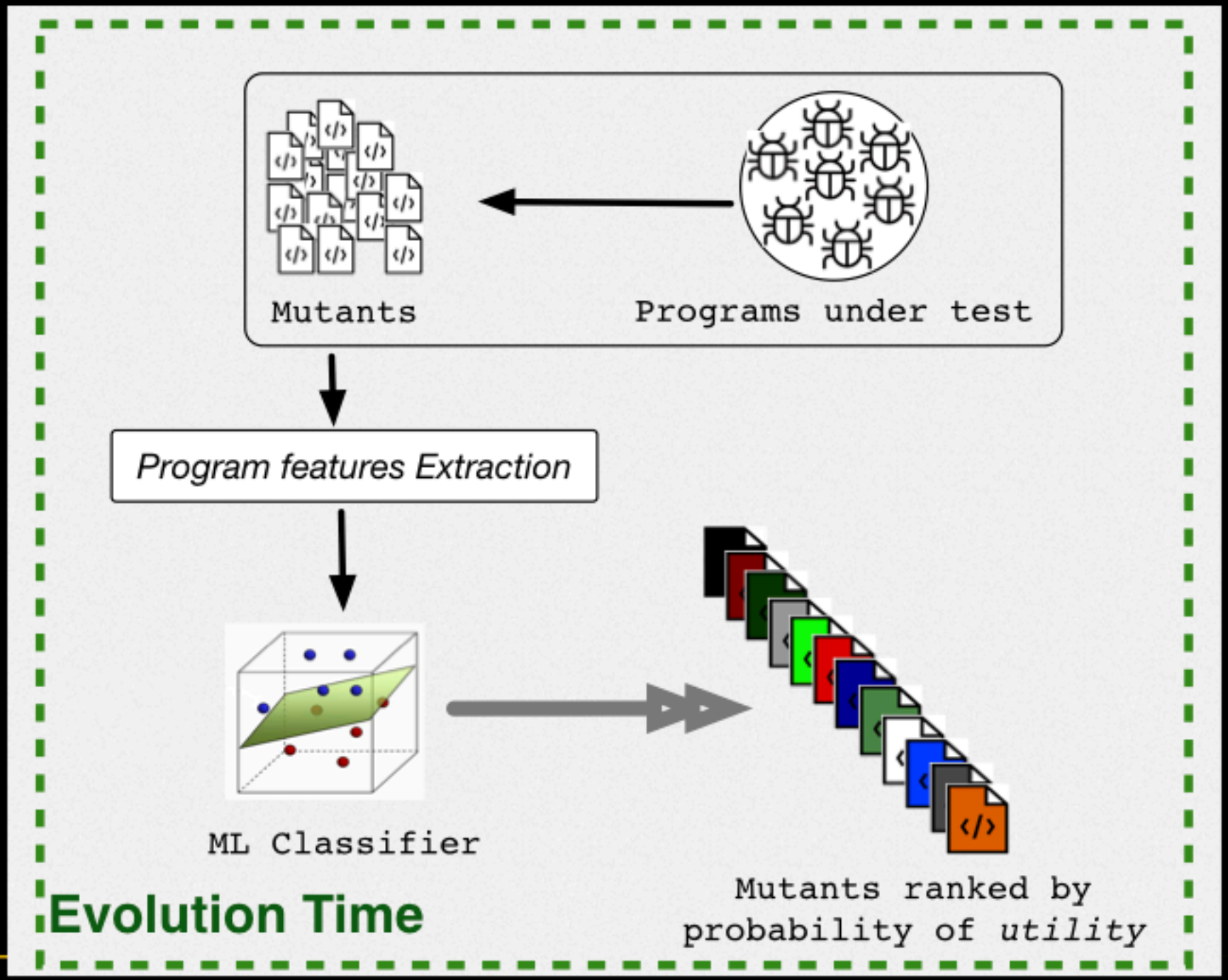
Classifier that predicts mutants' Utility



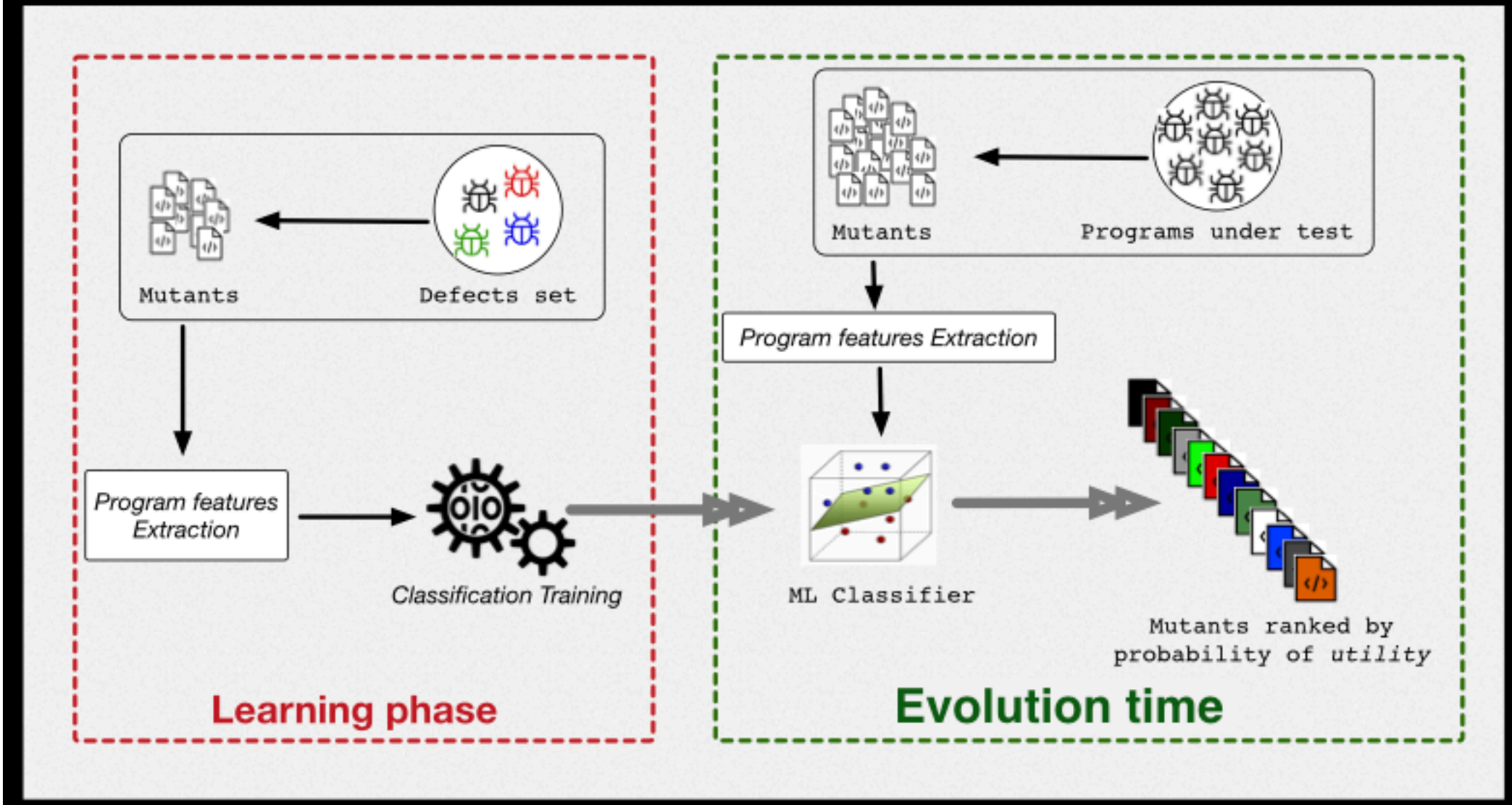
Classifier

Evolution

Learning-to-rank Mutants



Learning-to-rank Mutants



Features

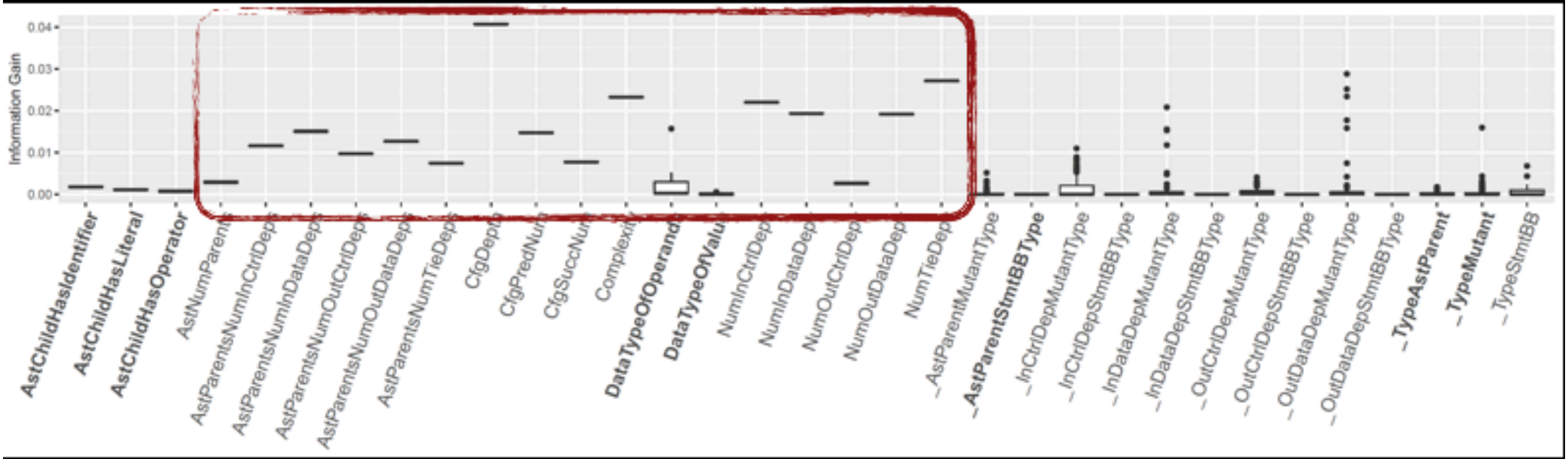
- Depth in the CFG of ***B***
- Complexity of **S** as its number of mutants
- Mutant type of **M**
- Types of mutants on P_S
- AST type of P_S
- Number of predecessor/successor Basic Blocks of ***B*** on the CFG
- Number of AST parents of ***S***
- Type of ***B***
- Type of P_S' basic block
- Number of mutants on ***S***
- Number of mutants of statement control out-dependent to **S**
- Number of mutants of statement control in-dependent to **S**
- Number of mutants of statement data out-dependent to **S**
- Number of mutants of statement data in-dependent to **S**
- Number of mutants of statement control out-dependent to P_S
- Number of mutants of statement control in-dependent to P_S
- Number of mutants of statement data out-dependent to P_S
- Number of mutants of statement data in-dependent to P_S
- Mutant type and statement type of statement control out-dependent to **S**
- Mutant type and statement type of statement control in-dependent to **S**
- Mutant type and statement type of statement data out-dependent to **S**
- Mutant type and statement type of statement data in-dependent to **S**

M denotes a mutant on statement S in basic block B. P_S denotes the AST parent (of S)

Learning to Rank Interesting Mutants

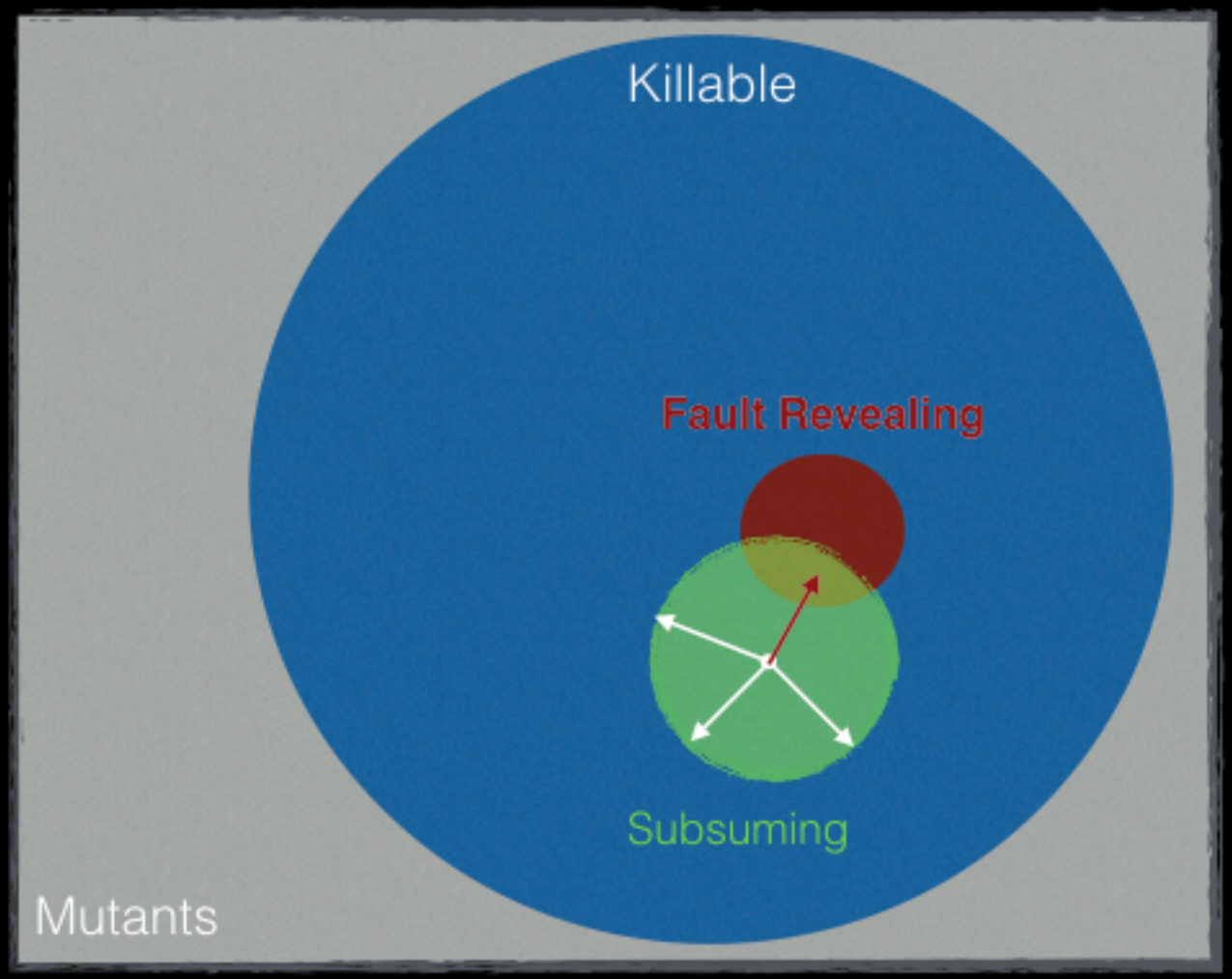
Use **Gradient Boosted Decision** Trees

LLVM-based mutation tool

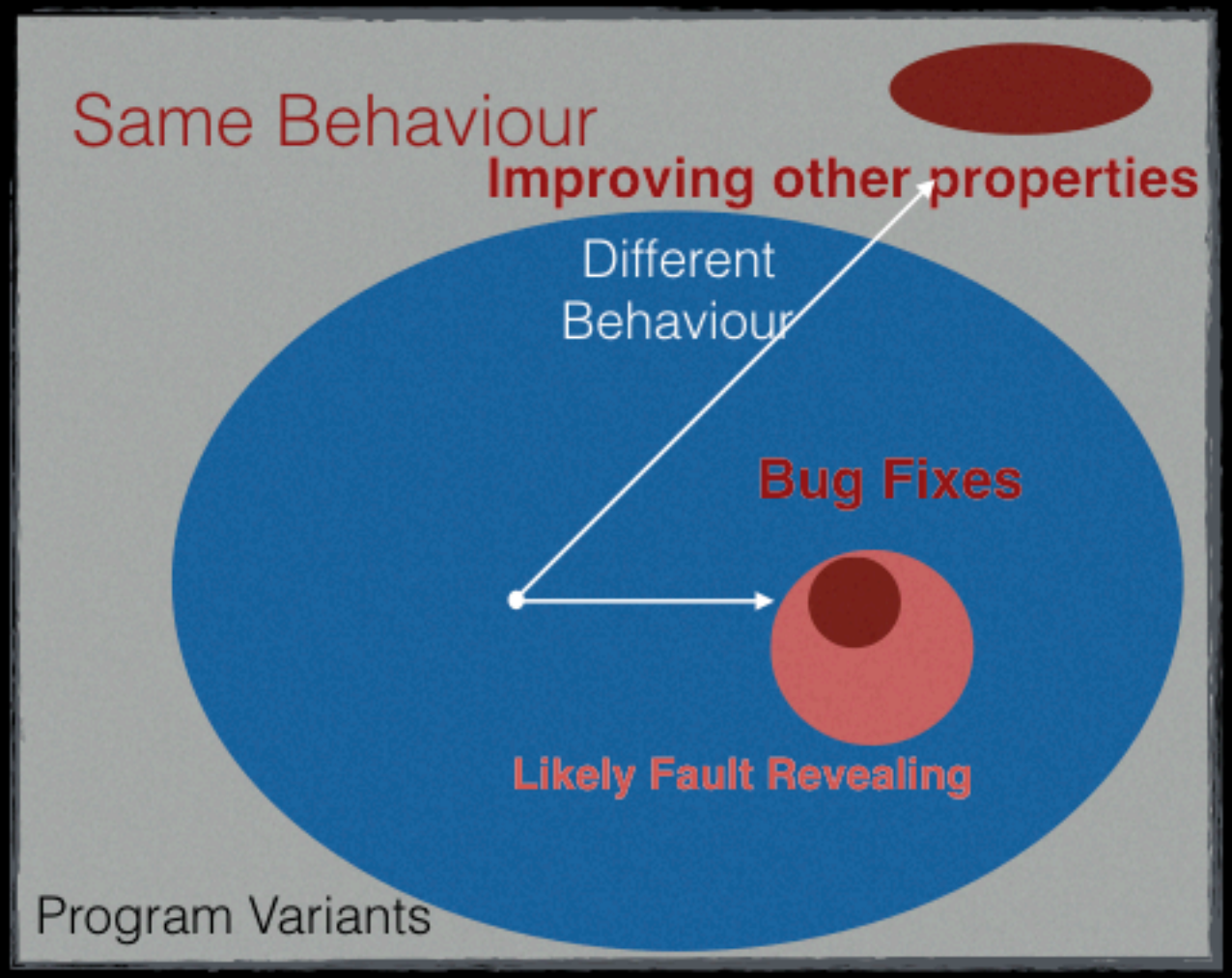

Information Gain
0.04
0.03
0.02
0.01
0.00
AstChildHasIdentifier
AstChildHasLiteral
AstChildHasOperator
AstNumParents
AstParentsNumInCtrlDeps
AstParentsNumInDataDeps
AstParentsNumOutCtrlDeps
AstParentsNumOutDataDeps
AstParentsNumTieDeps
CfgDepth
CfgPredNum
CfgSuccNum
Complexity
DataTypeOfOperand
DataTypeOfValue
NumInCtrlDeps
NumInDataDeps
NumOutCtrlDeps
NumOutDataDeps
NumTieDeps
_AstParentMutantType
_AstParentStmtBBType
_InCtrlDepMutantType
_InCtrlDepStmtBBType
_InDataDepMutantType
_InDataDepStmtBBType
_OutCtrlDepMutantType
_OutCtrlDepStmtBBType
_OutDataDepMutantType
_OutDataDepStmtBBType
_TypeAstParent
_TypeMutant
_TypeStmtBB

Control and **Data Dependencies** are the most **informative** Features

Subsuming mutants and Faults

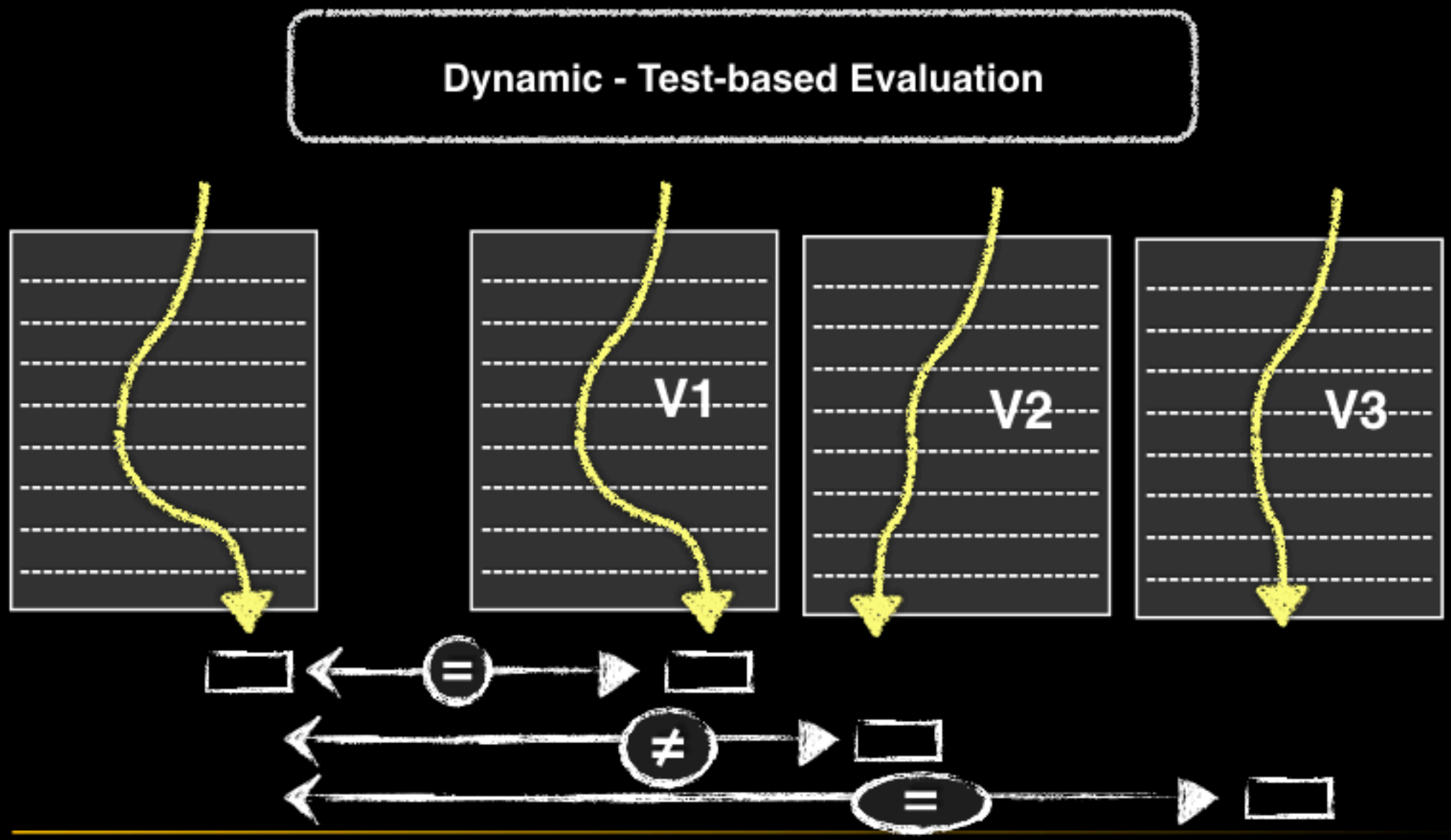


Mutation Testing and Automated Program Improvement



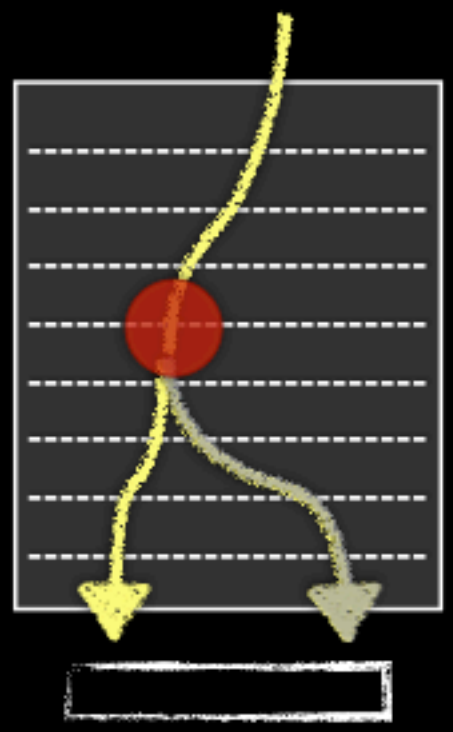
Plausibility & Evaluation

Evaluate Solutions



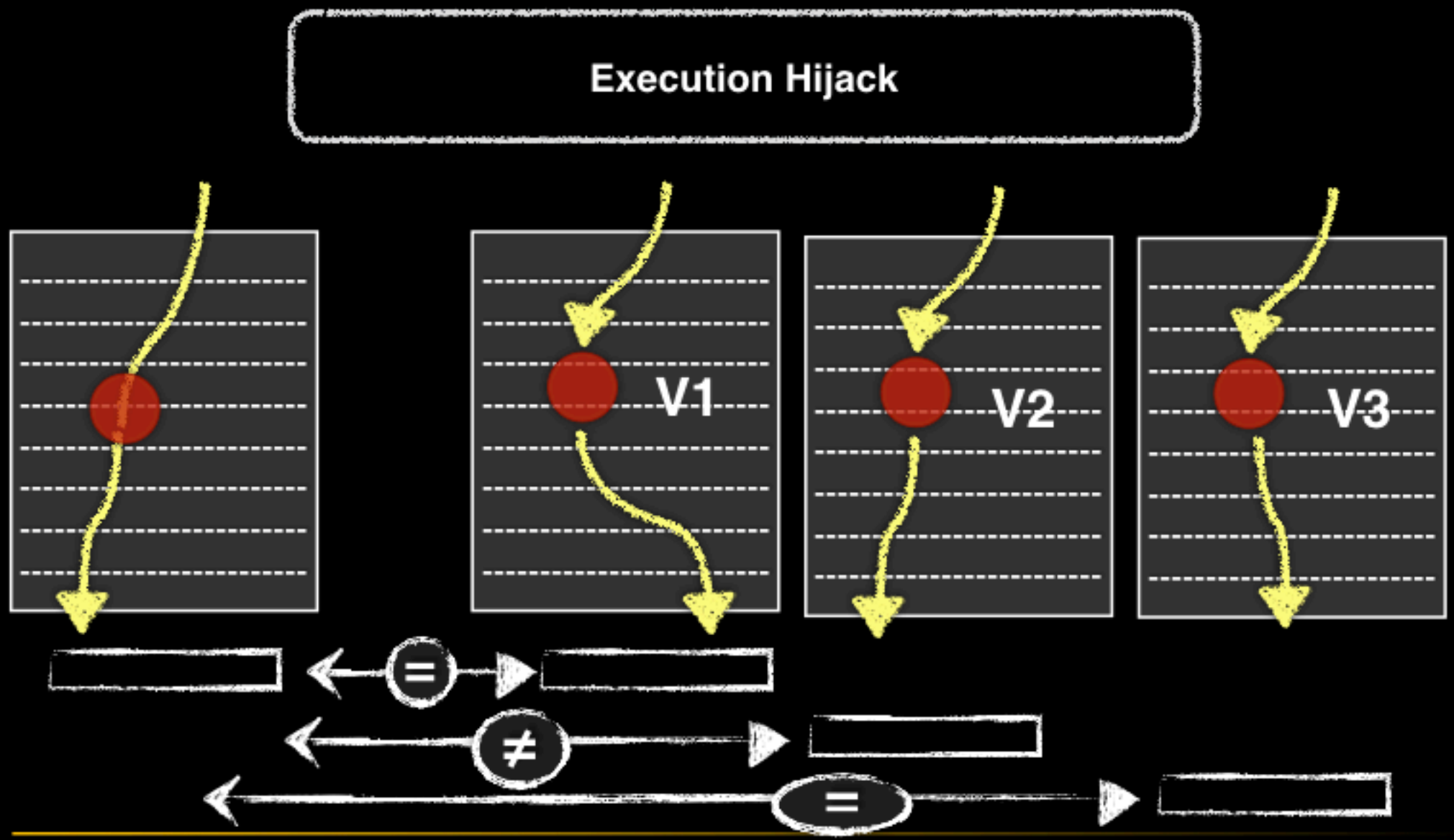
Evaluate Solutions

Execution Hijack



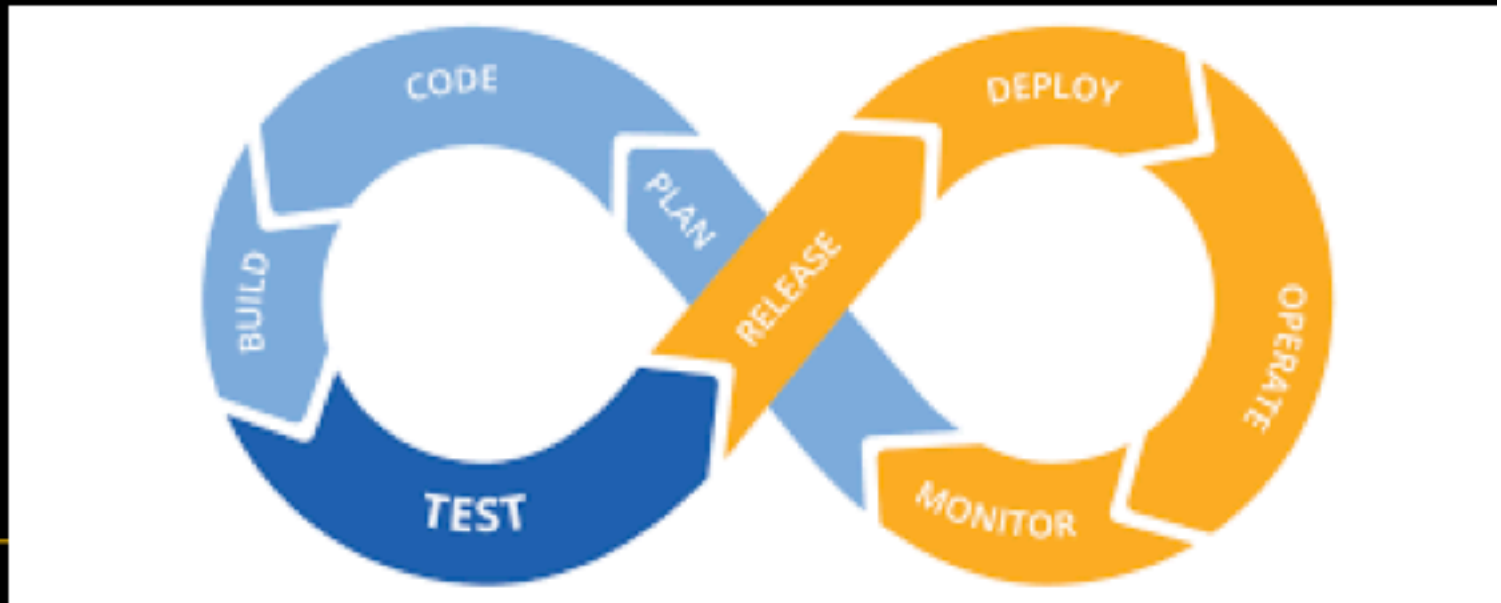
Tsankov et al. **Execution Hijacking: Improving Dynamic Analysis by Flying off Course** ICST'11

Evaluate Solutions



Tsankov et al. **Execution Hijacking: Improving Dynamic Analysis by Flying off Course** ICST'11

Proposing & Documenting Changes



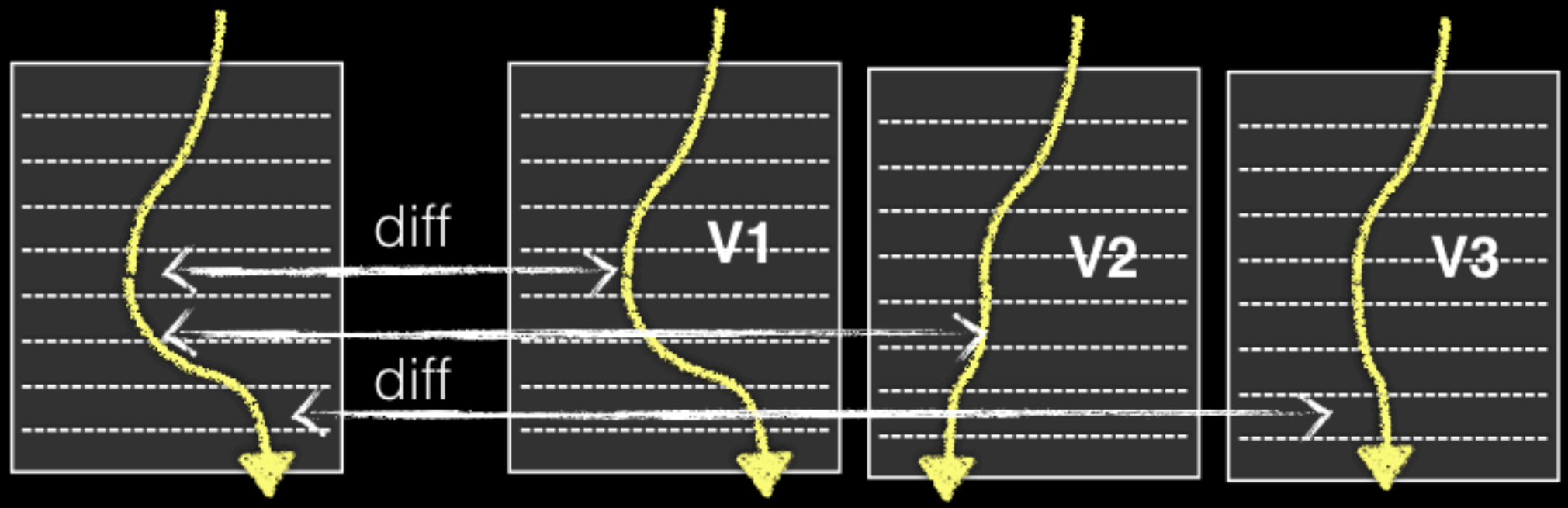
Present the Results

V1

V2

V3

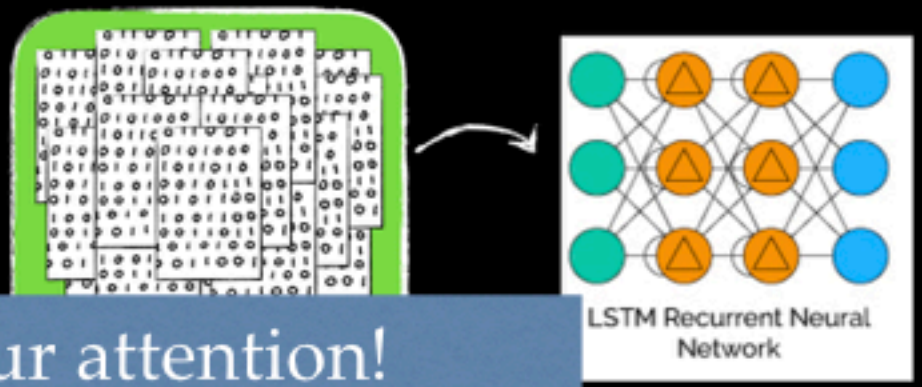
Differences in the data states



Differences Synthesise conditions-hints

Mutation Testing and Automated Program Improvement

Build a model (learn from existing code)

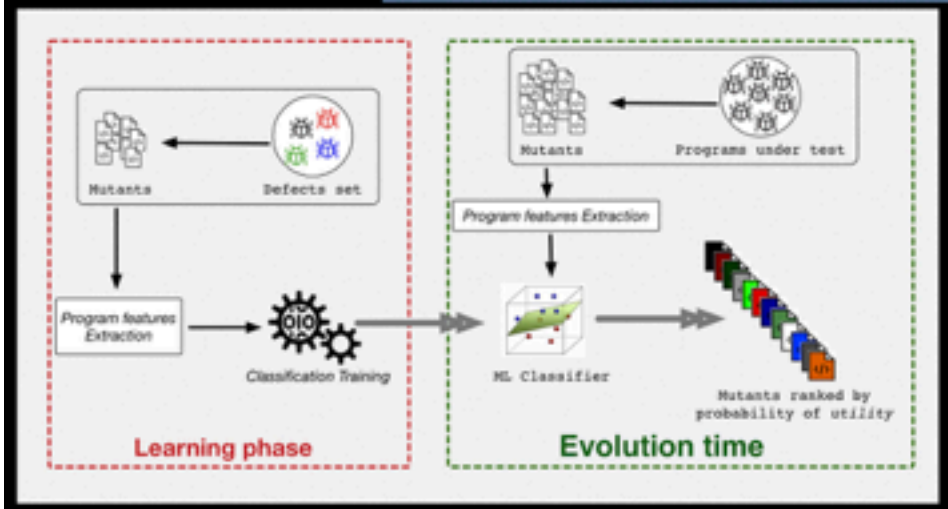


Thank you for your attention!

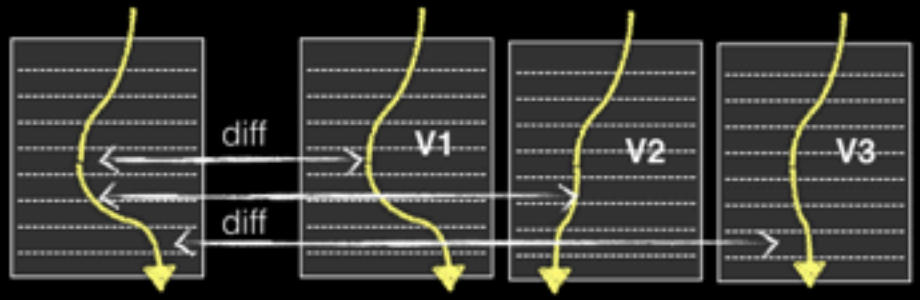
https://sites.google.com/site/mikepapadakis/

michail.papadakis@uni.lu

Learning-to-ran



Differences in the data states



Differences Synthesise conditions-hints