ISP RAS

Dynamic program analysis

Gerasimov Alexander

Dynamic program analysis –

program analysis while execution or using execution results

Dynamic analysis: applications

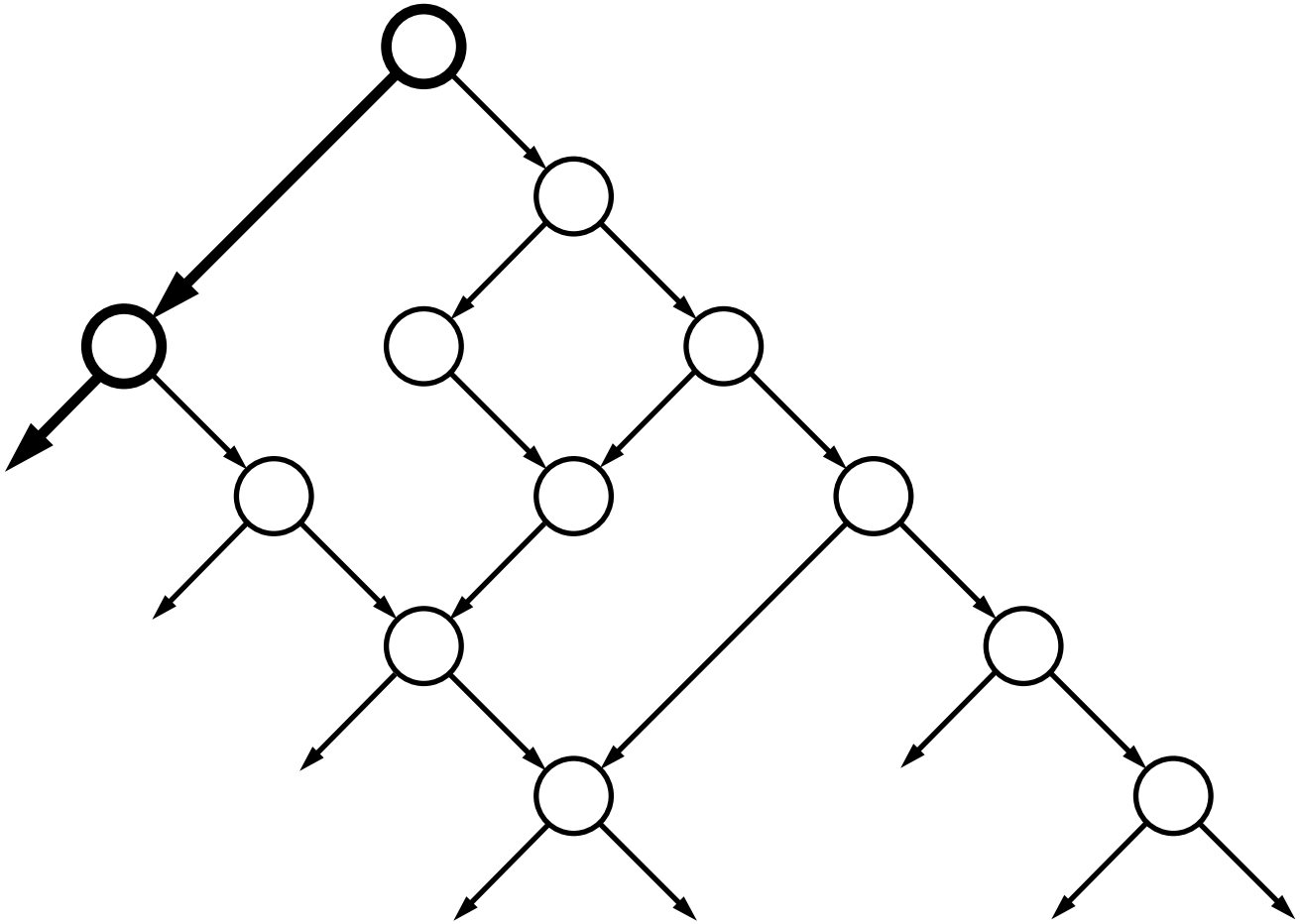
- Input data generation for code coverage purposes
- Defects and vulnerabilities detection
- Program profiling
  - Resource profiling (memory, files, …)
  - Performance profiling

Dynamic analysis: methods

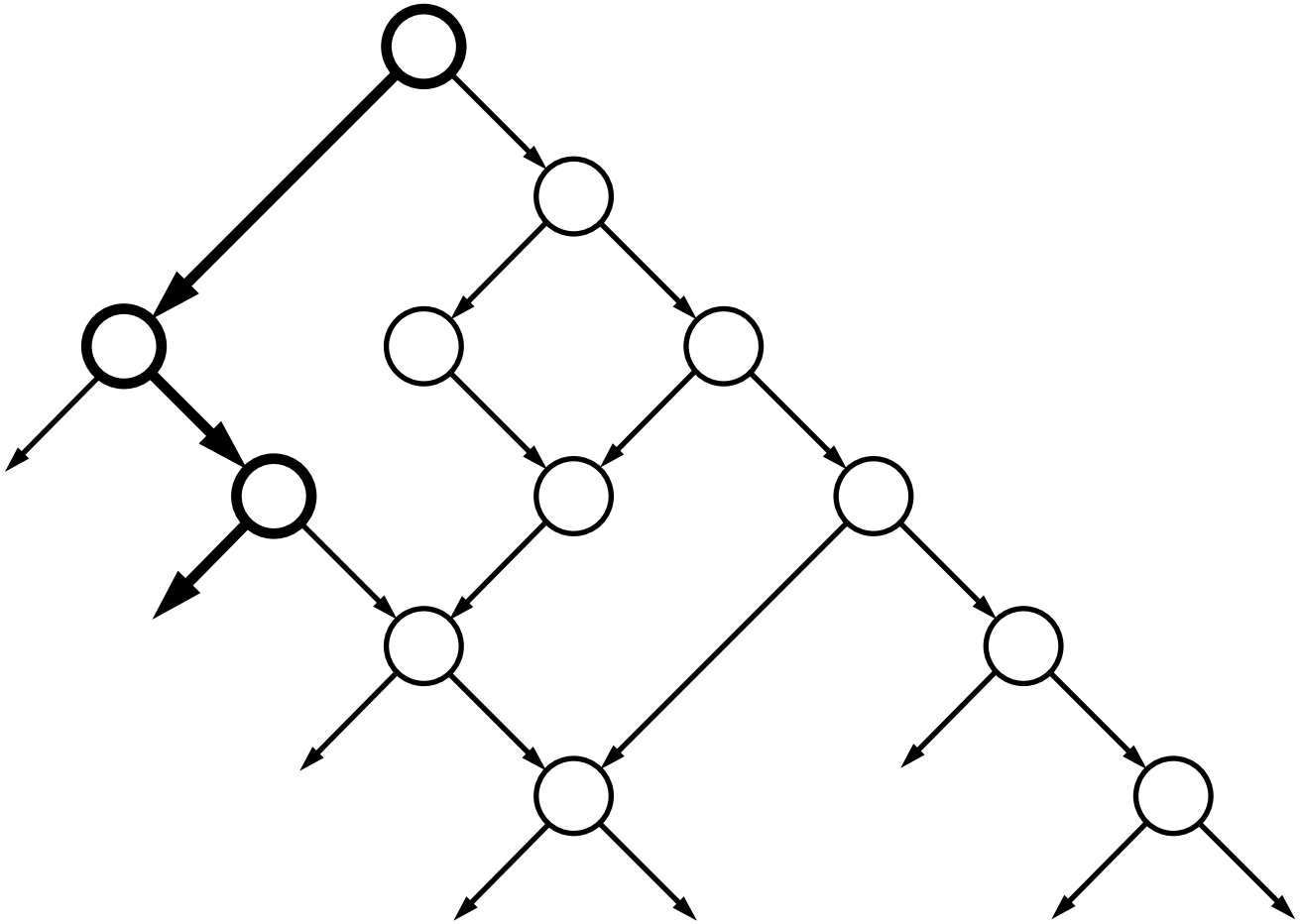
- Fuzzing
- Dynamic symbolic execution
- Special methods

Fuzzing

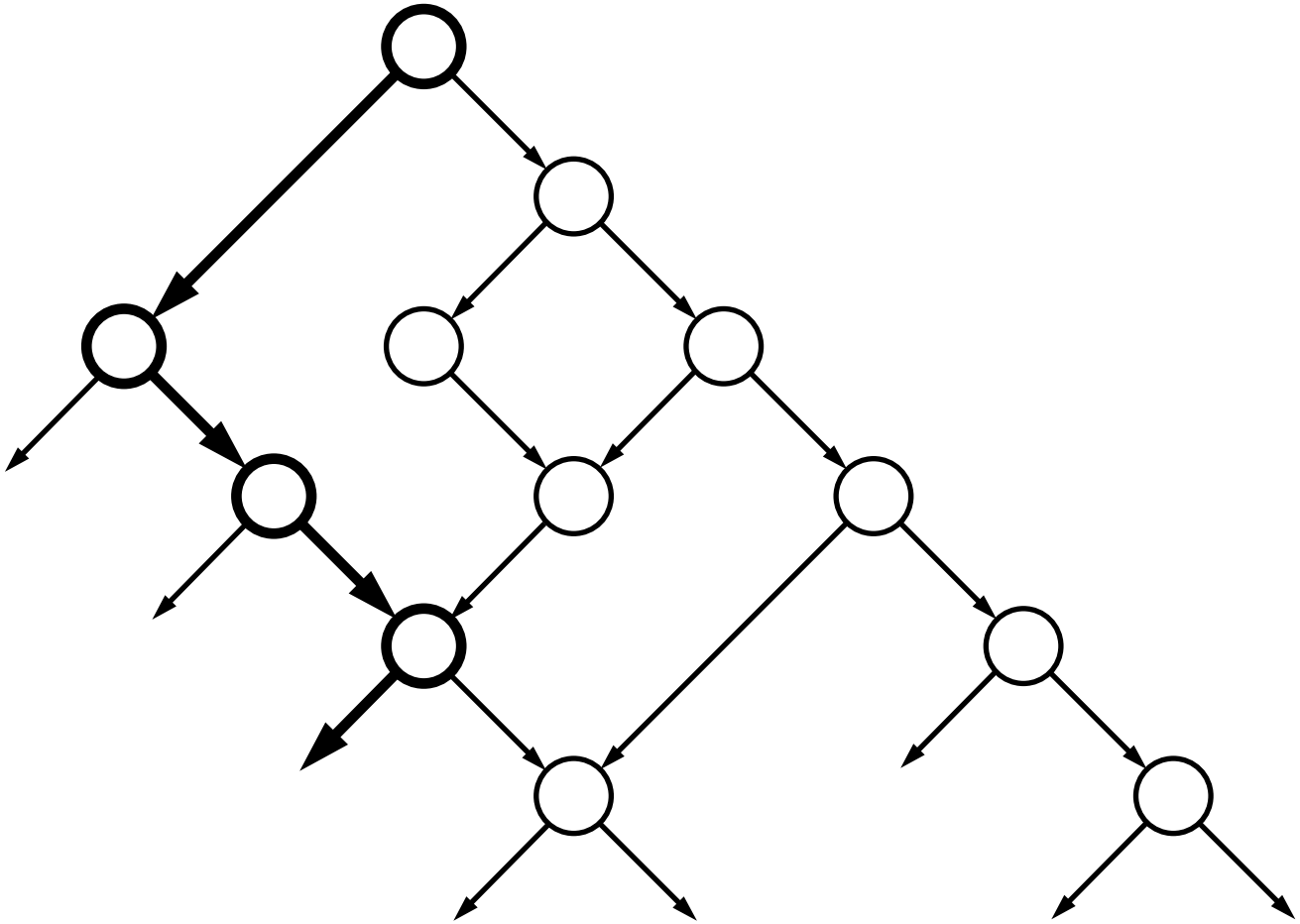
Fuzzing: Seeded run



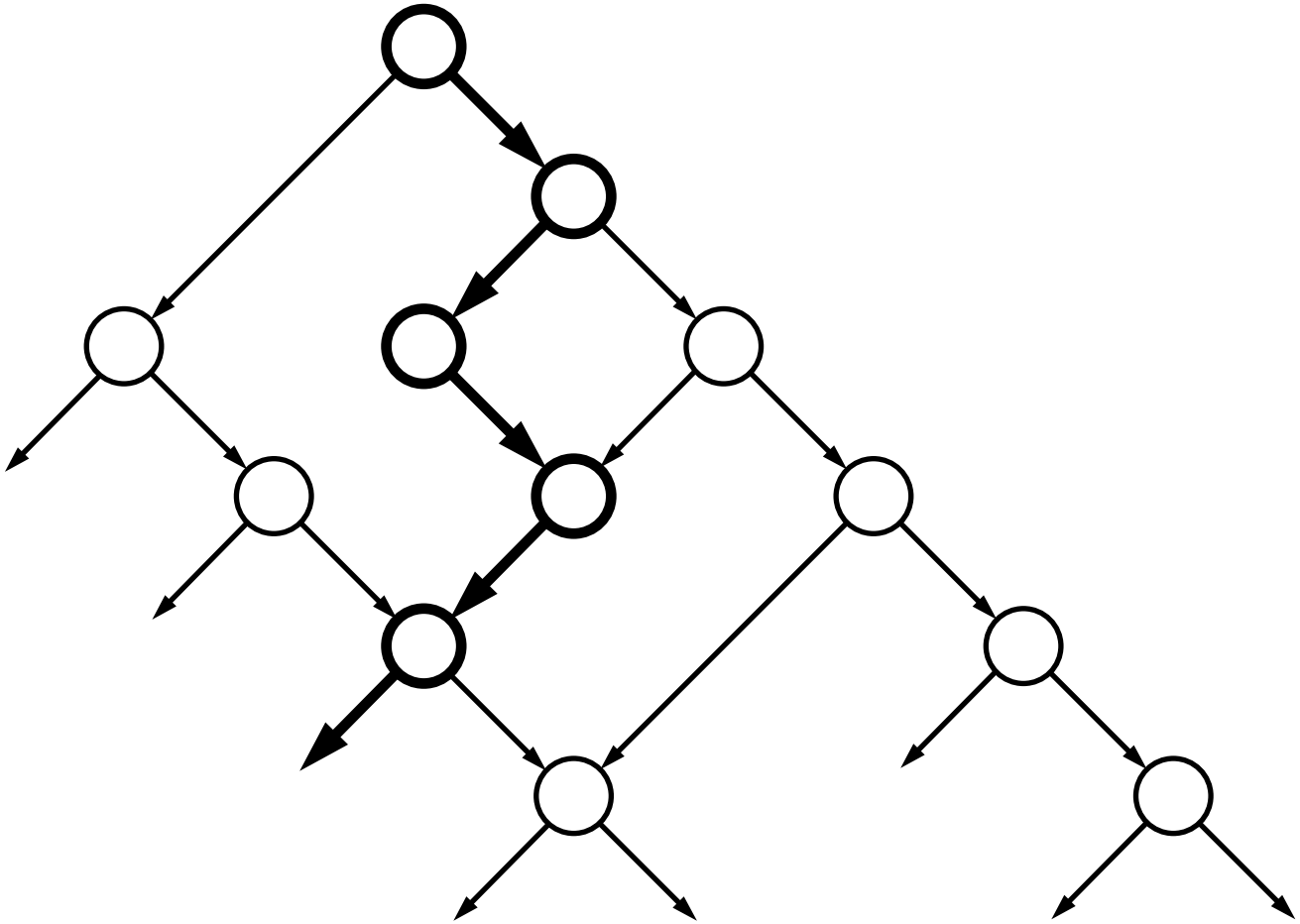
Fuzzing: Mutation 1



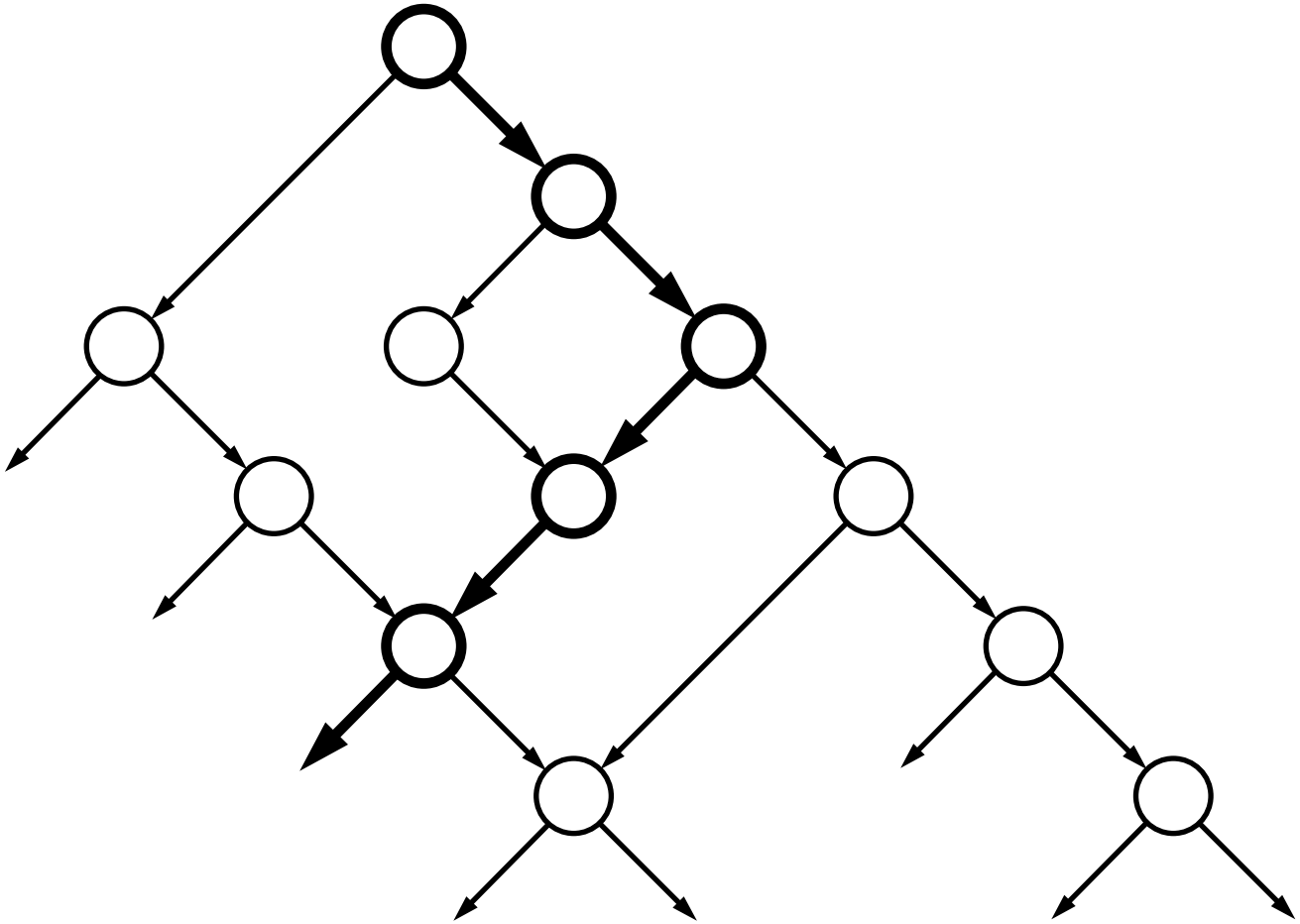
Fuzzing: Mutation 2



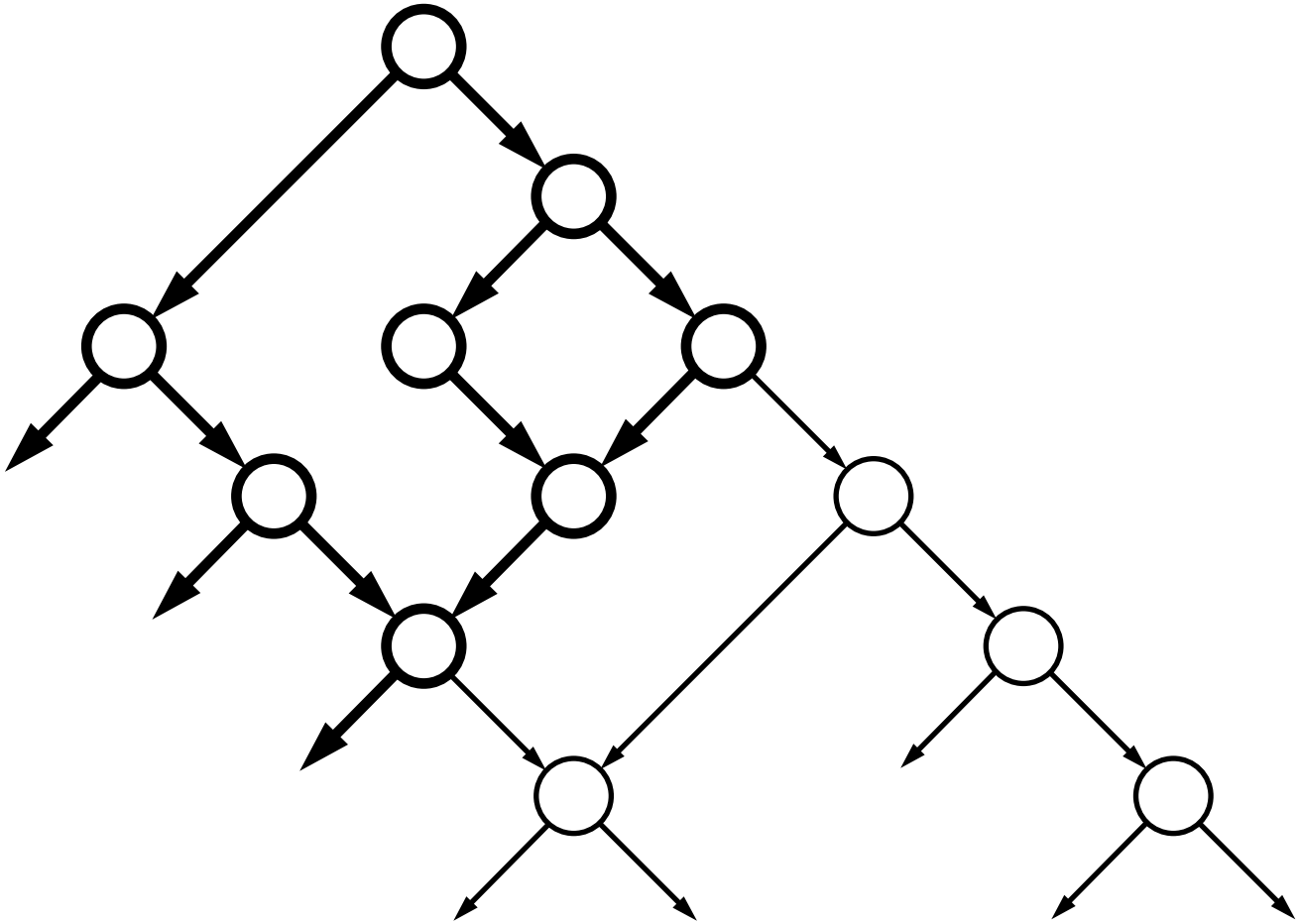
Fuzzing: Mutation 3



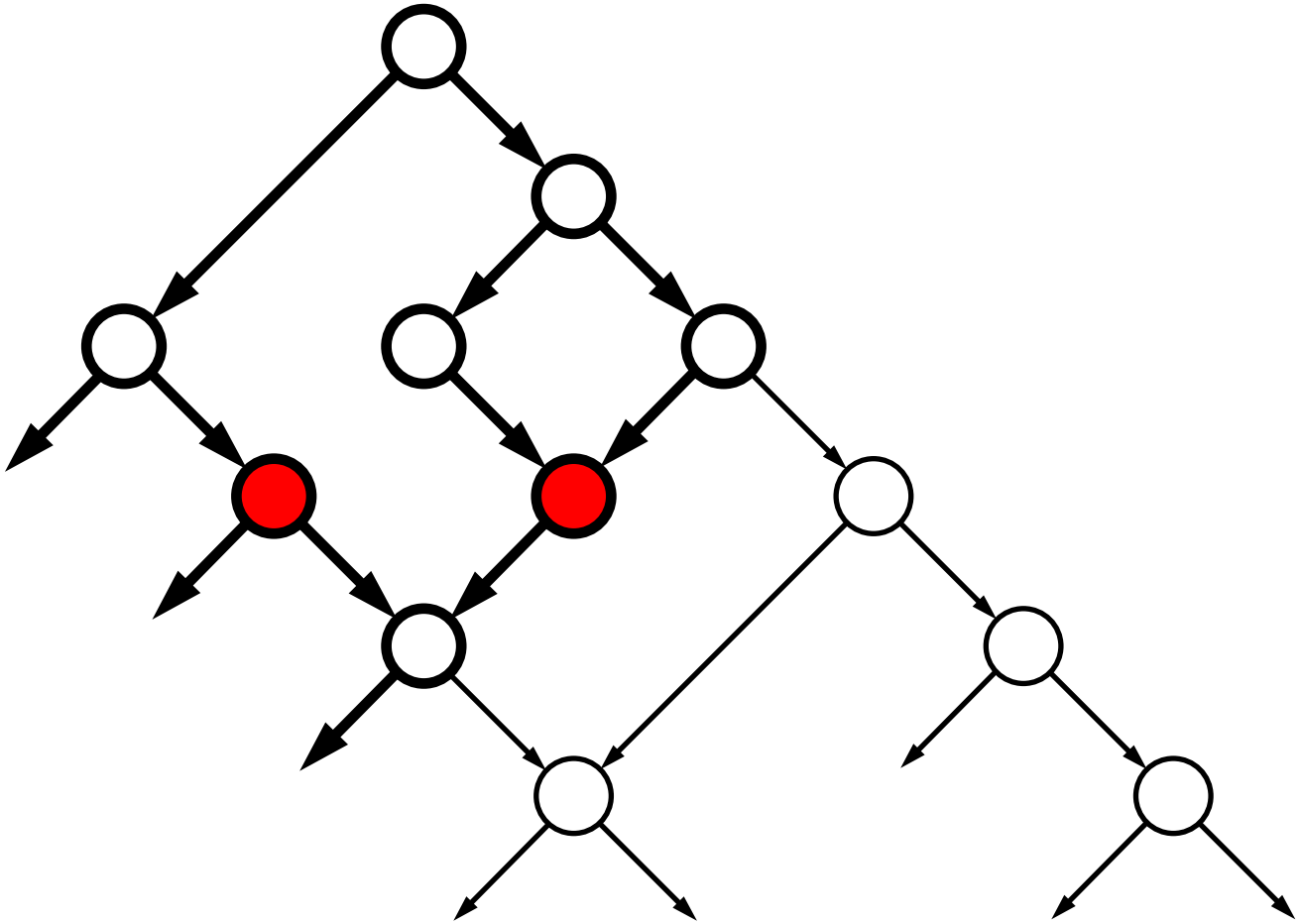
Fuzzing: Mutation 4



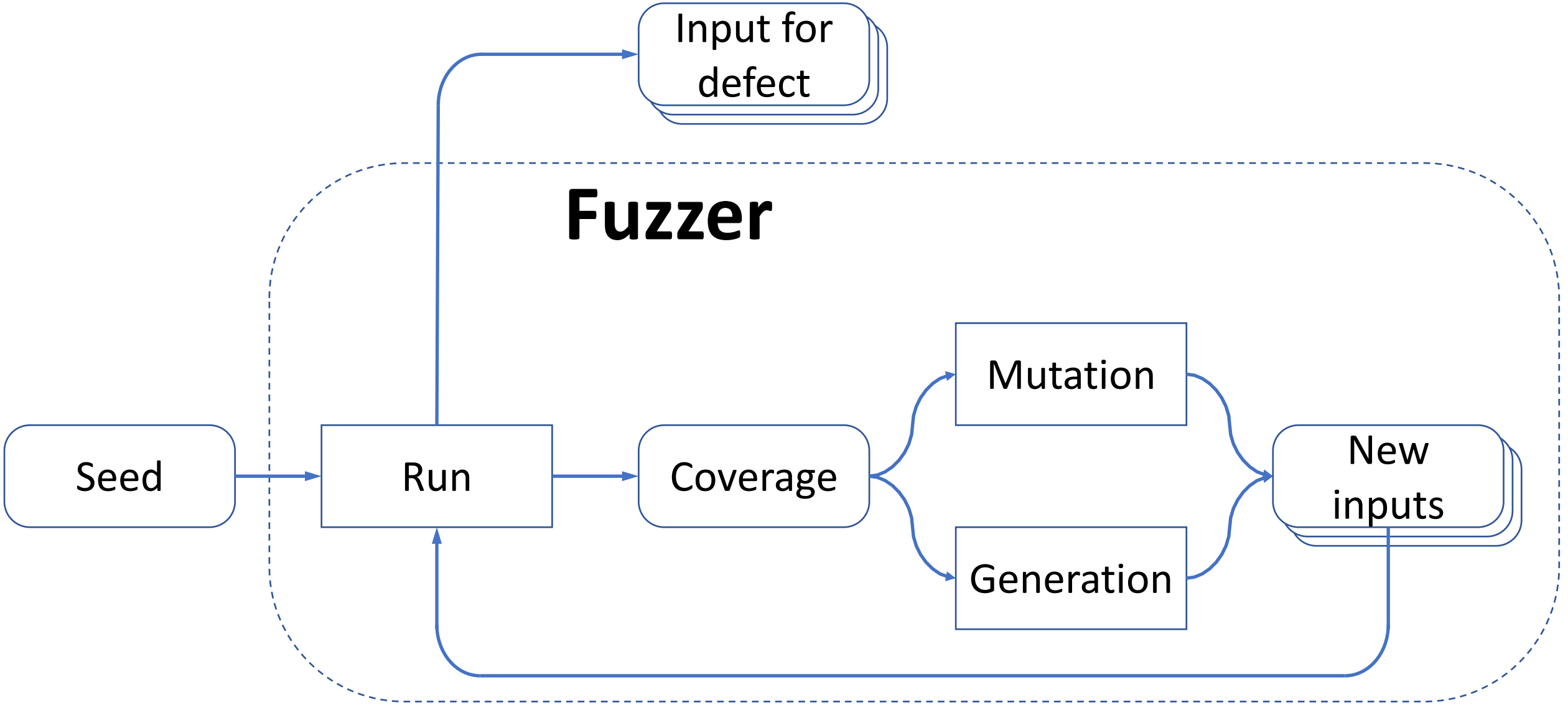
Fuzzing: analyzed compartment



Fuzzing: discovered defects



Fuzzing: tool structure



Fuzzing: kinds

- Black-box fuzzing
- Grey-box fuzzing
- White-box fuzzing

Fuzzing: gains

- most of CVEs excavated using fuzzing method
- there are commercial products

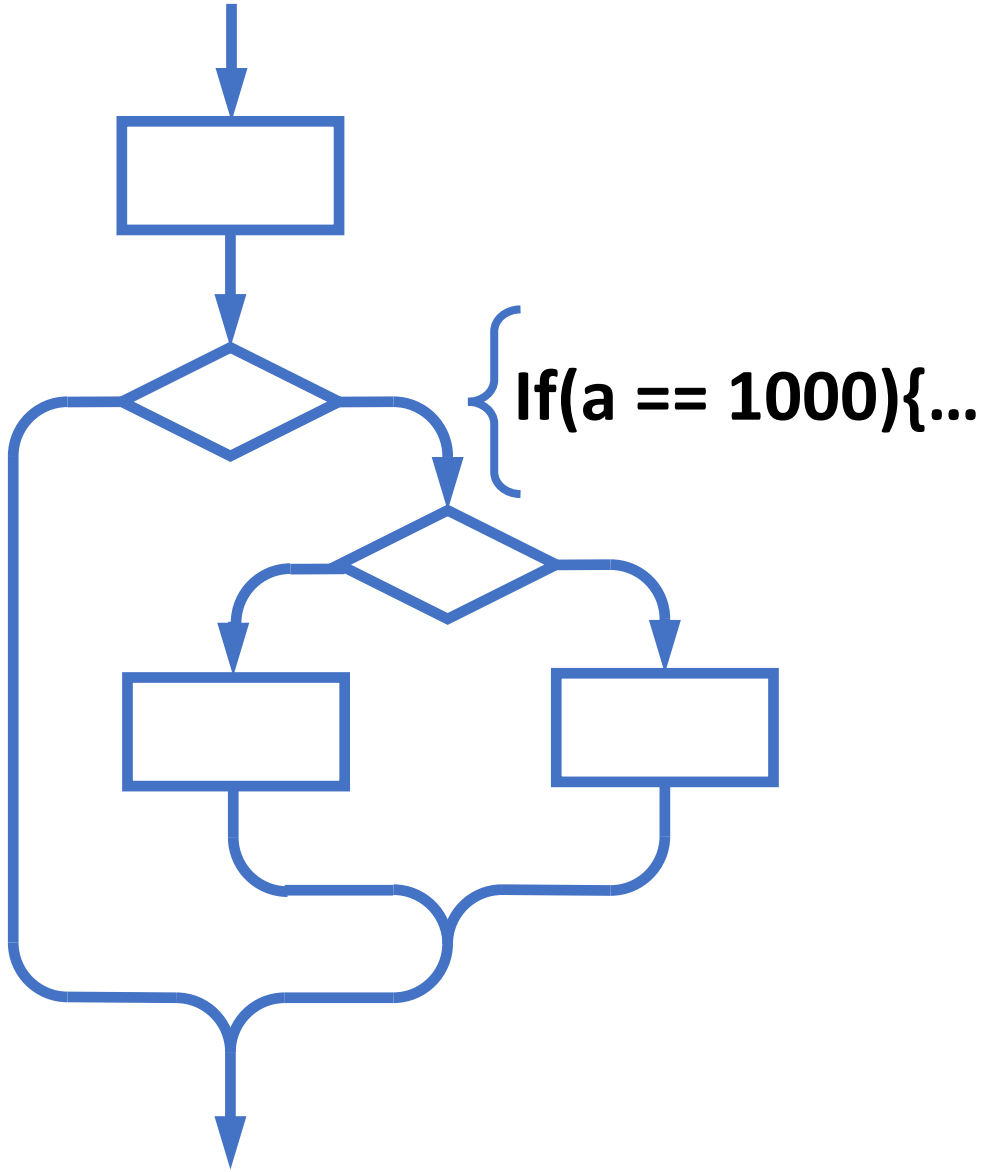
Fuzzer: problems of black-box/grey-box fuzzing

It is unlikely to guess

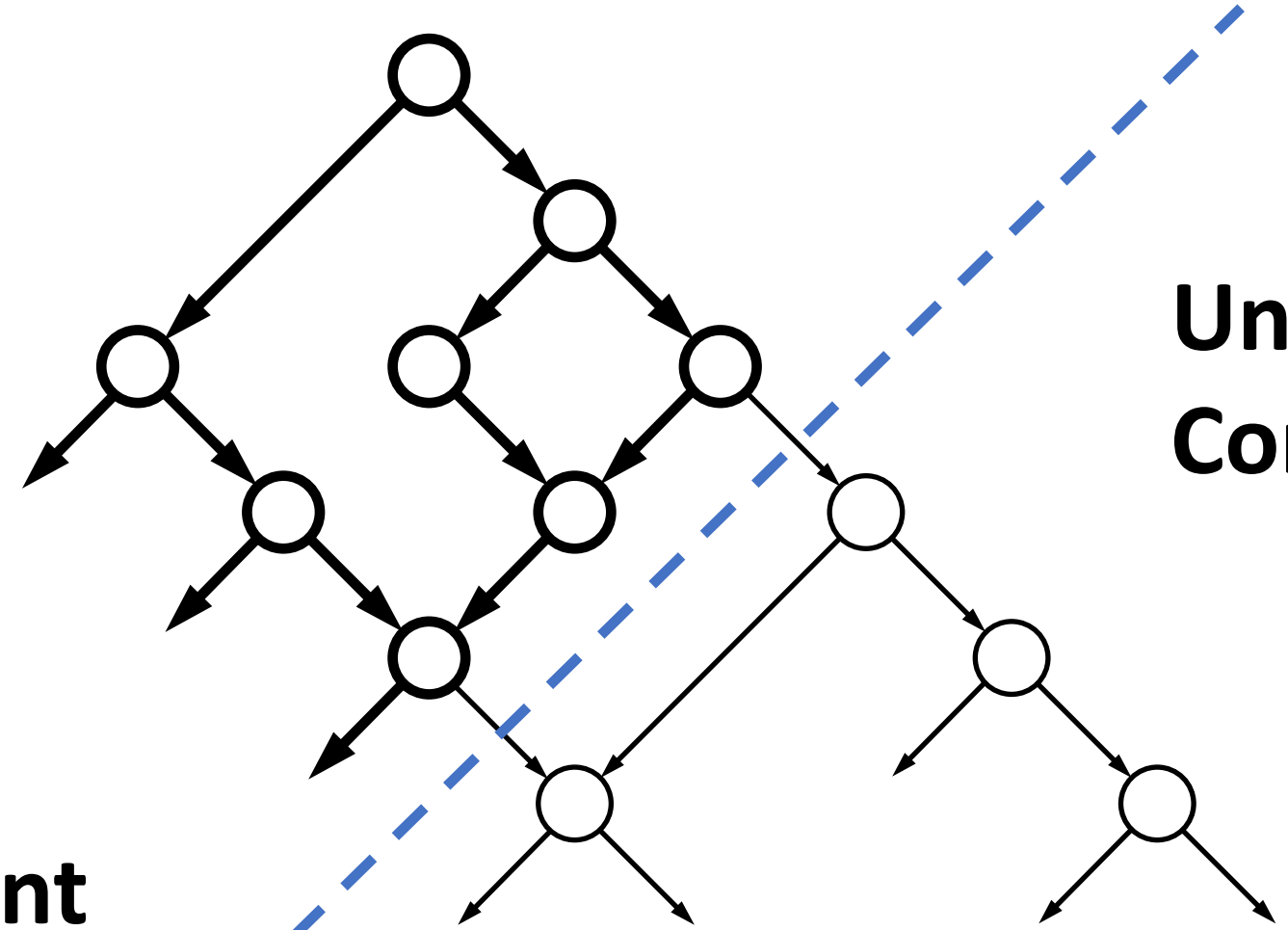
input to pass through

conditional jump with

constant comparison



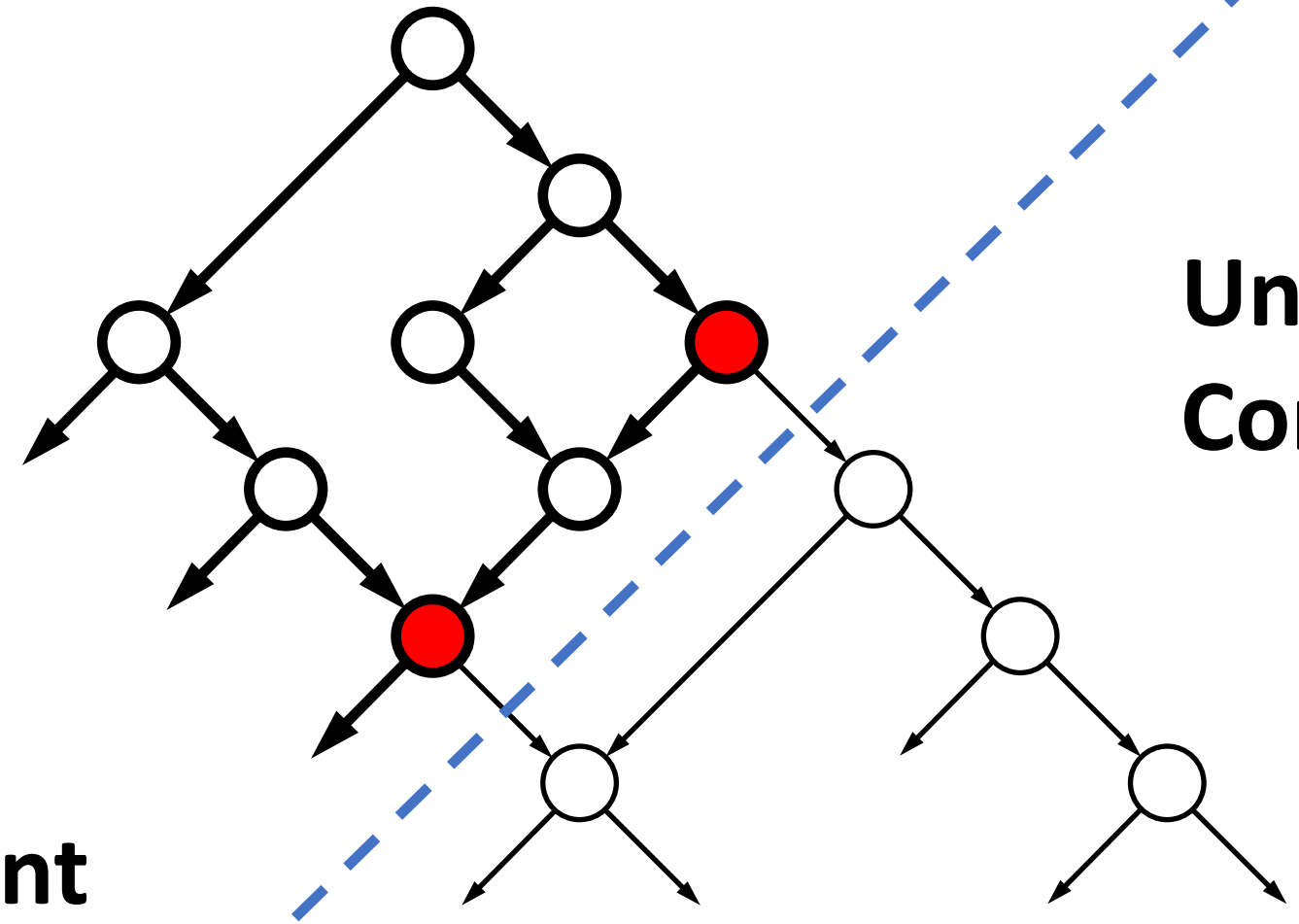
Fuzzing: compartments



Unanalyzed Compartment

Analyzed Compartment

Fuzzing: unlikely to guess input for constant comparisons



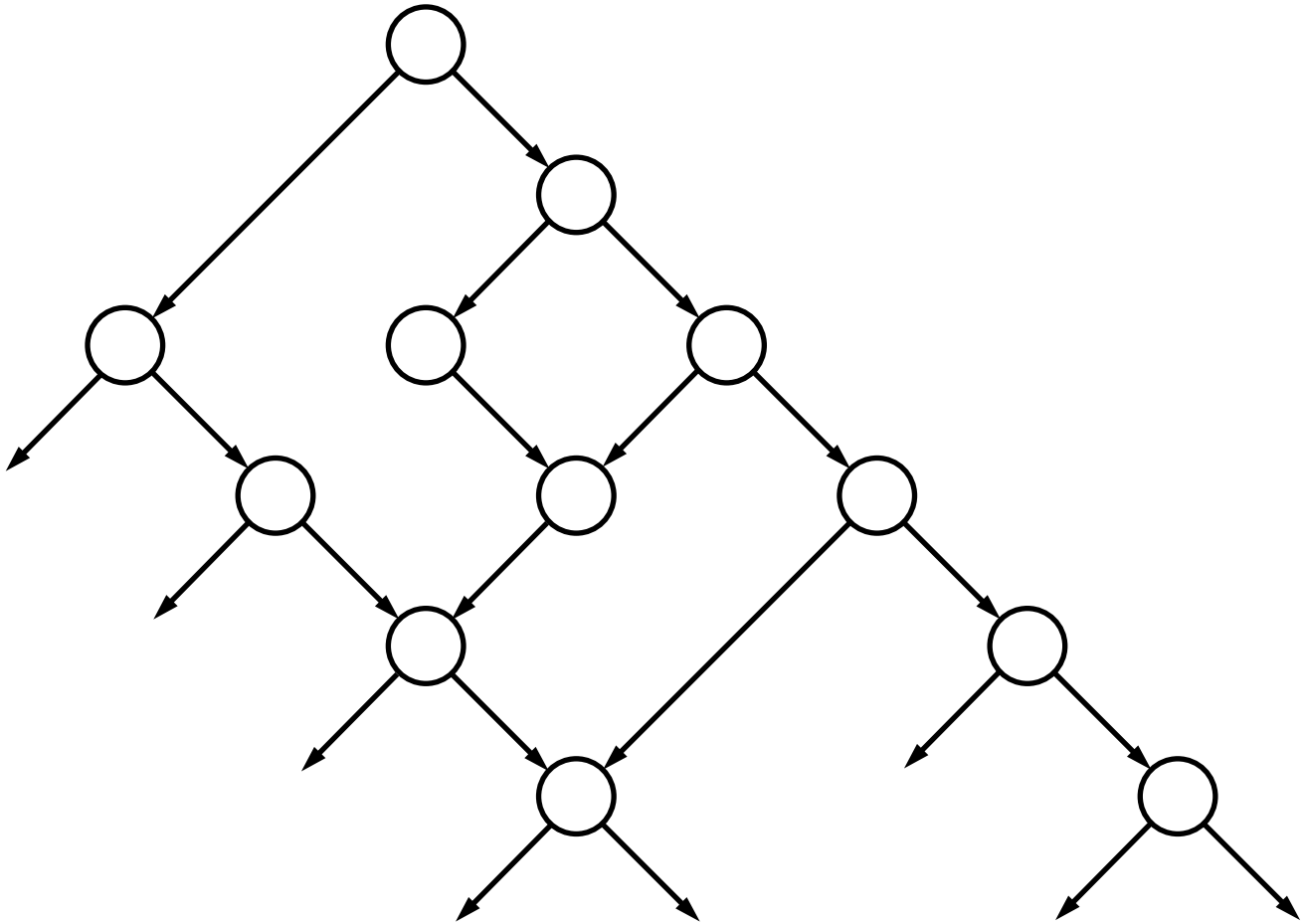
Unanalyzed Compartment

Analyzed Compartment

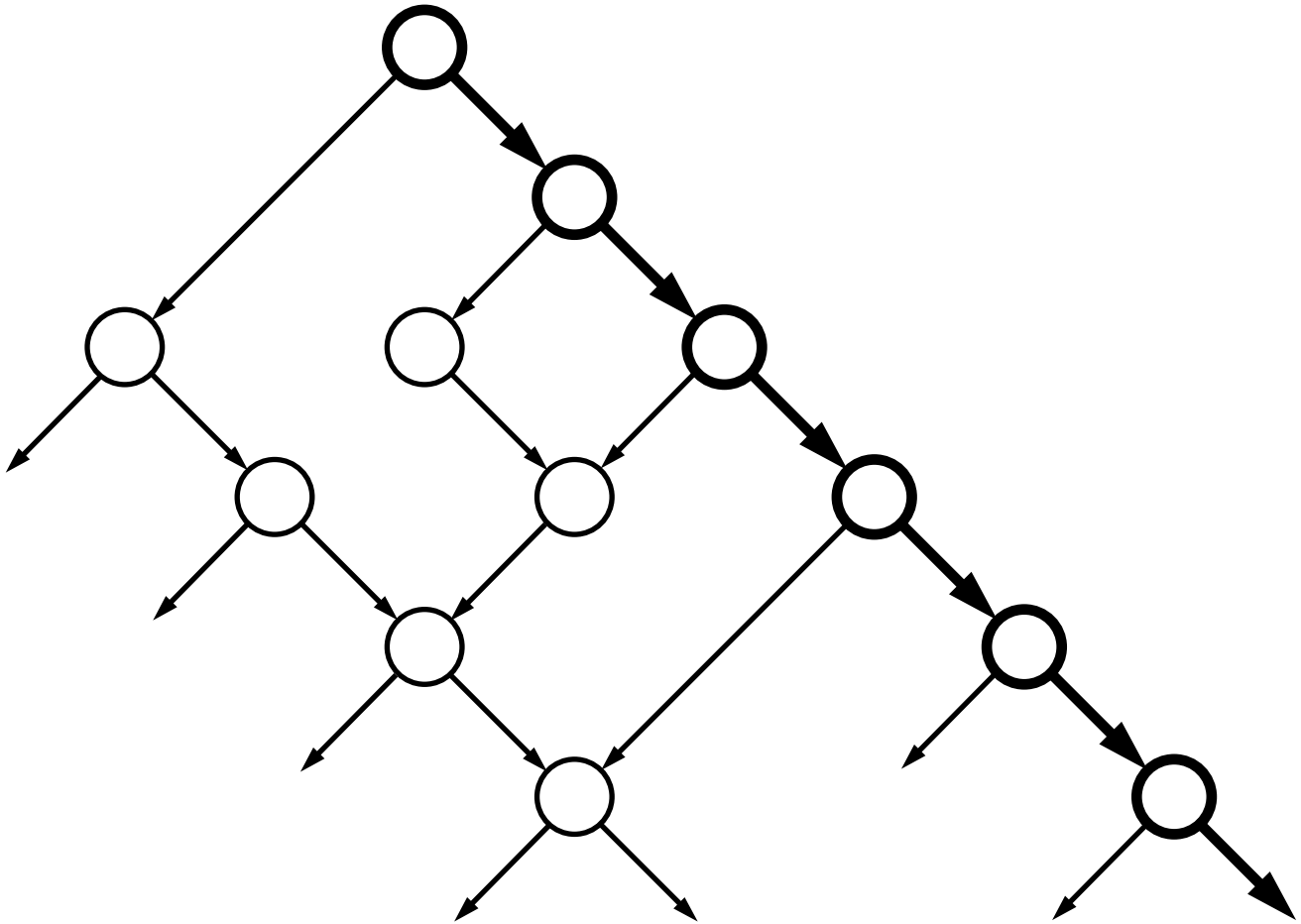
Dynamic symbolic execution

Coverage driven analysis

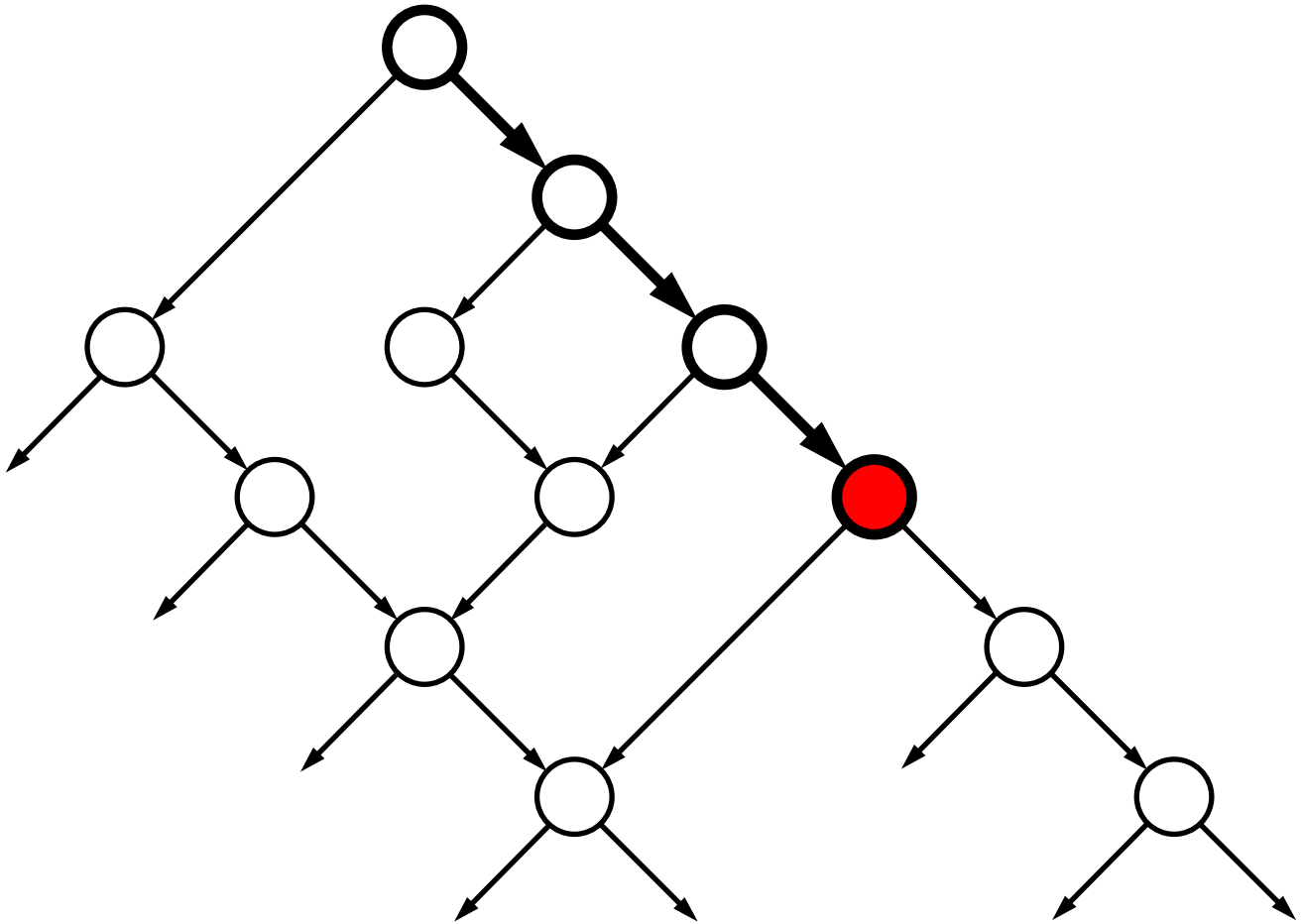
Dynamic program analysis: program jump map



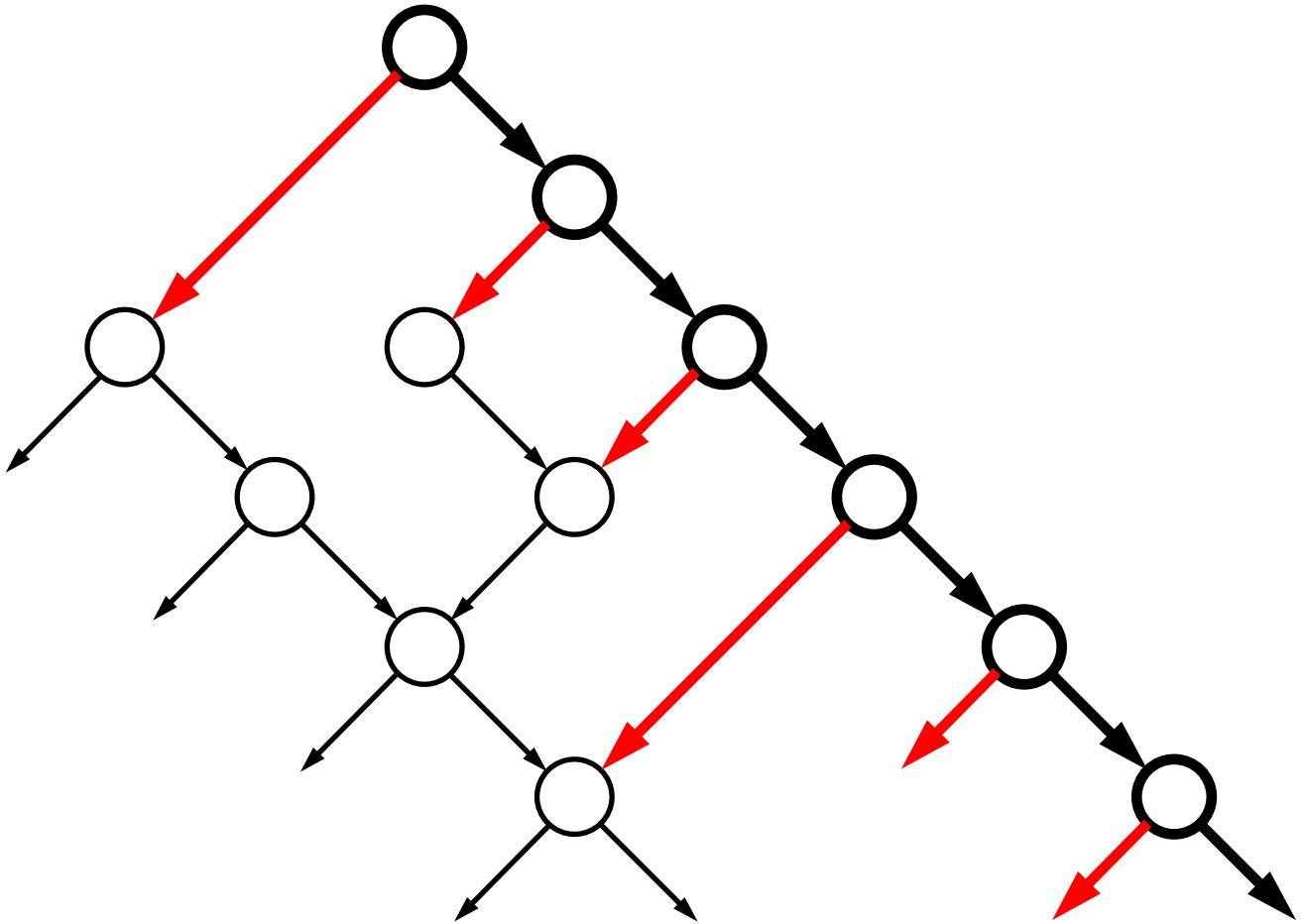
Dynamic program analysis: program run on some input



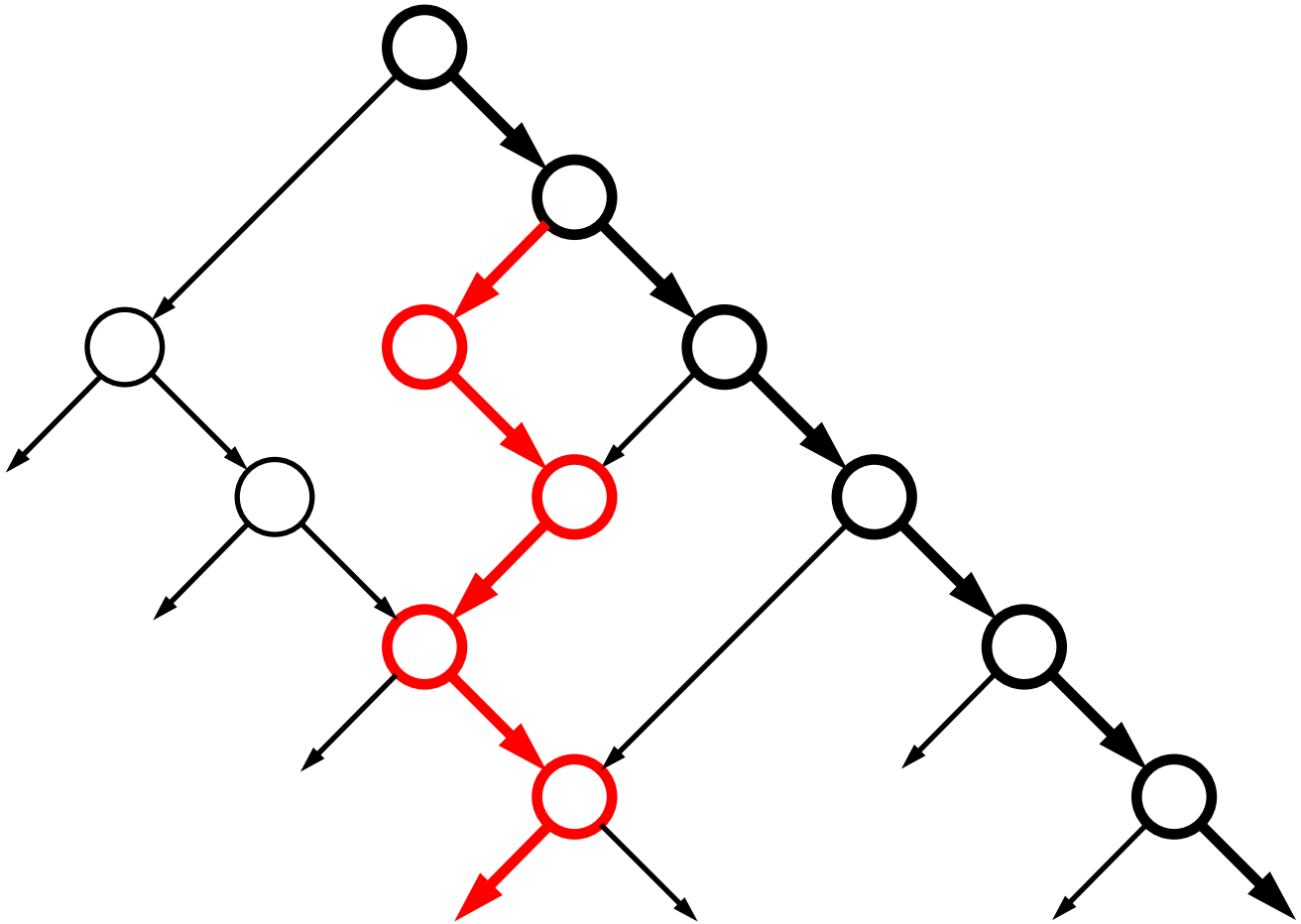
Dynamic program analysis: defect on execution trace



Dynamic program analysis: potential paths for further analysis



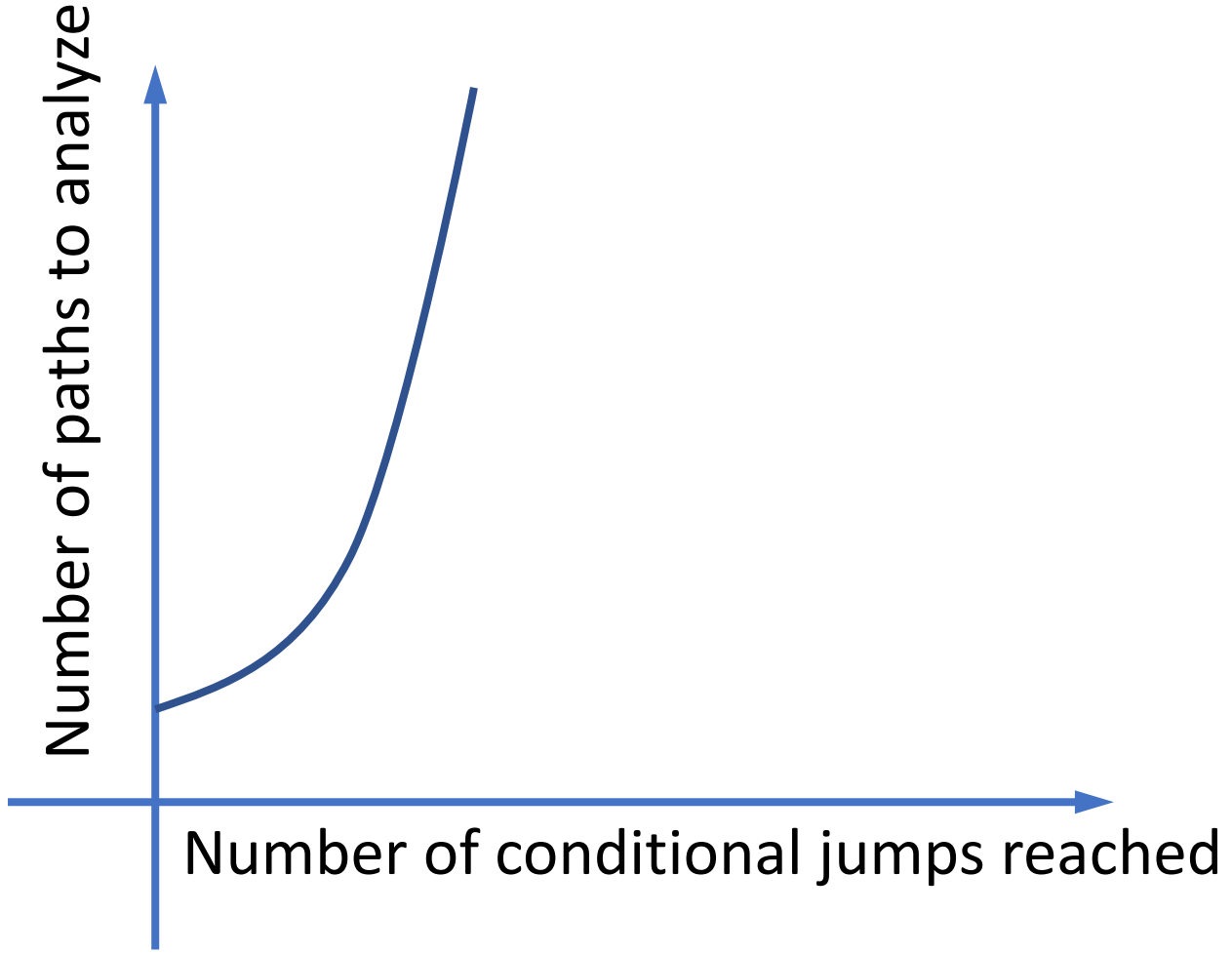
Dynamic program analysis: the most “promising” (maximum coverage growth) path for next analysis step



Problems of coverage driven dynamic symbolic execution

- Number of paths to analyze grows ~2^n of conditional jumps reached

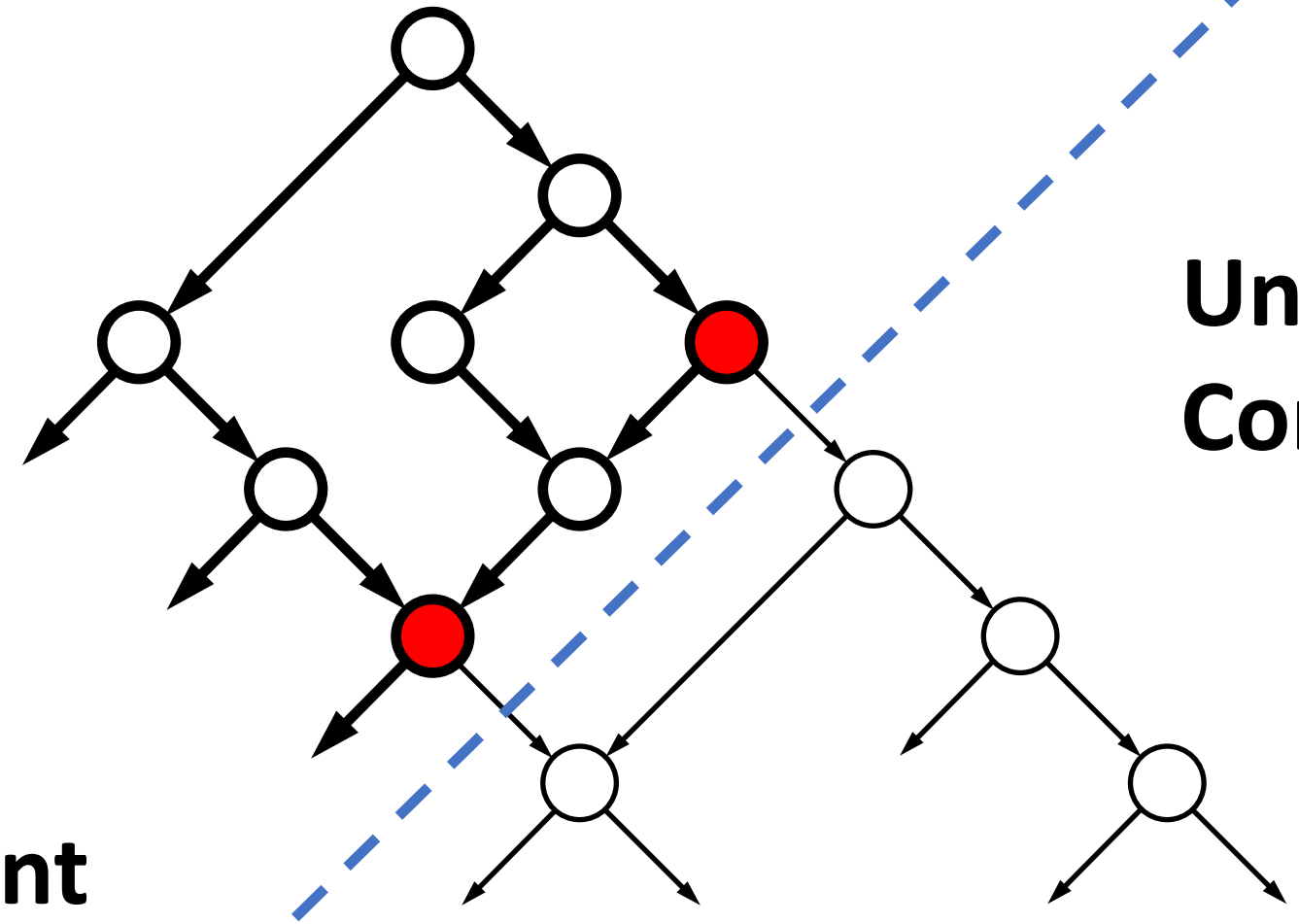
- Defects detected only on specific execution paths



Solution stay at the turn of methods

- **Fuzzer + Dynamic Symbolic Execution**
- **Static analysis + Dynamic Symbolic Execution**

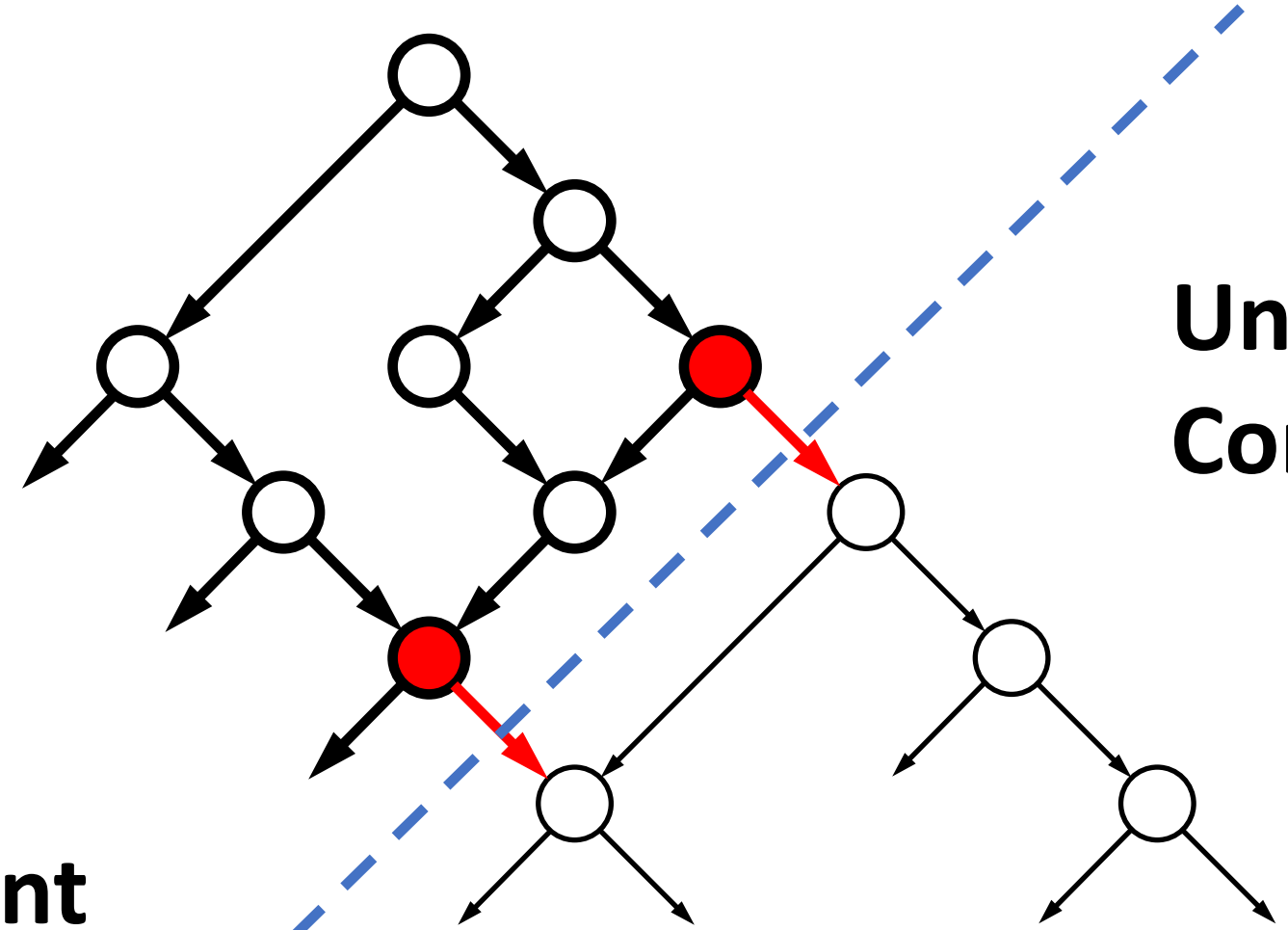
Fuzzing: unlikely to guess input for constant comparisons



Unanalyzed Compartment

Analyzed Compartment

Fuzzing: desired jumps



Unanalyzed Compartment

Analyzed Compartment

Combine Dynamic Symbolic Execution and Fuzzer

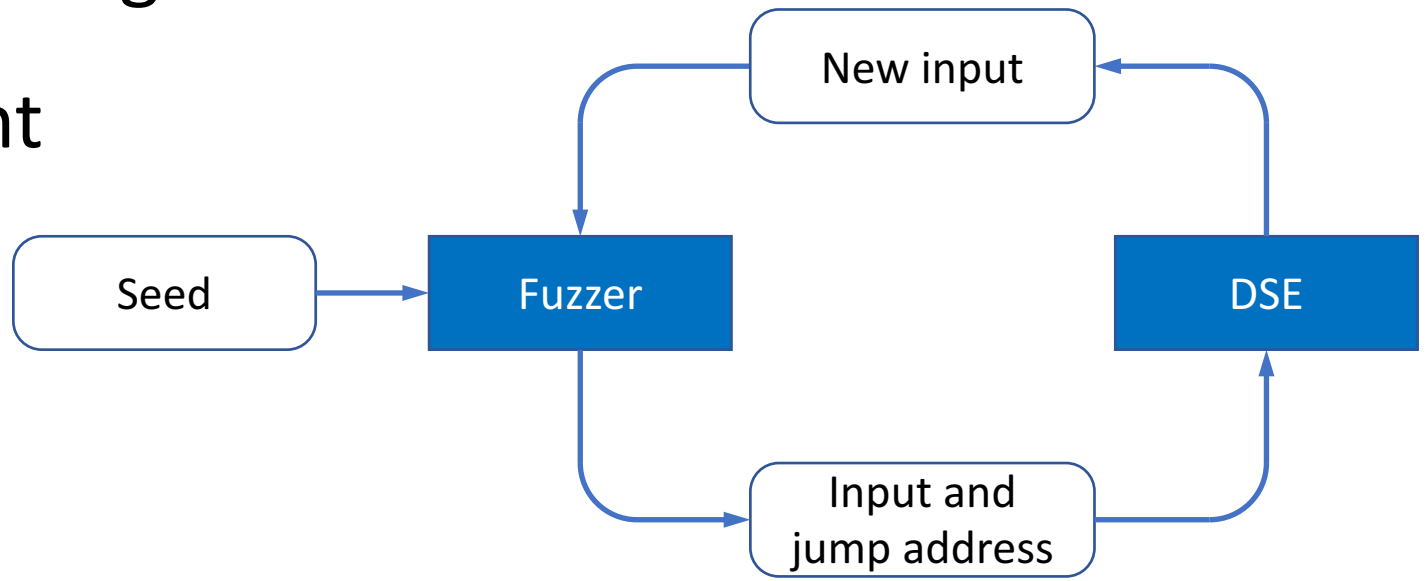
At the limit only one run of

Dynamic Symbolic Execution

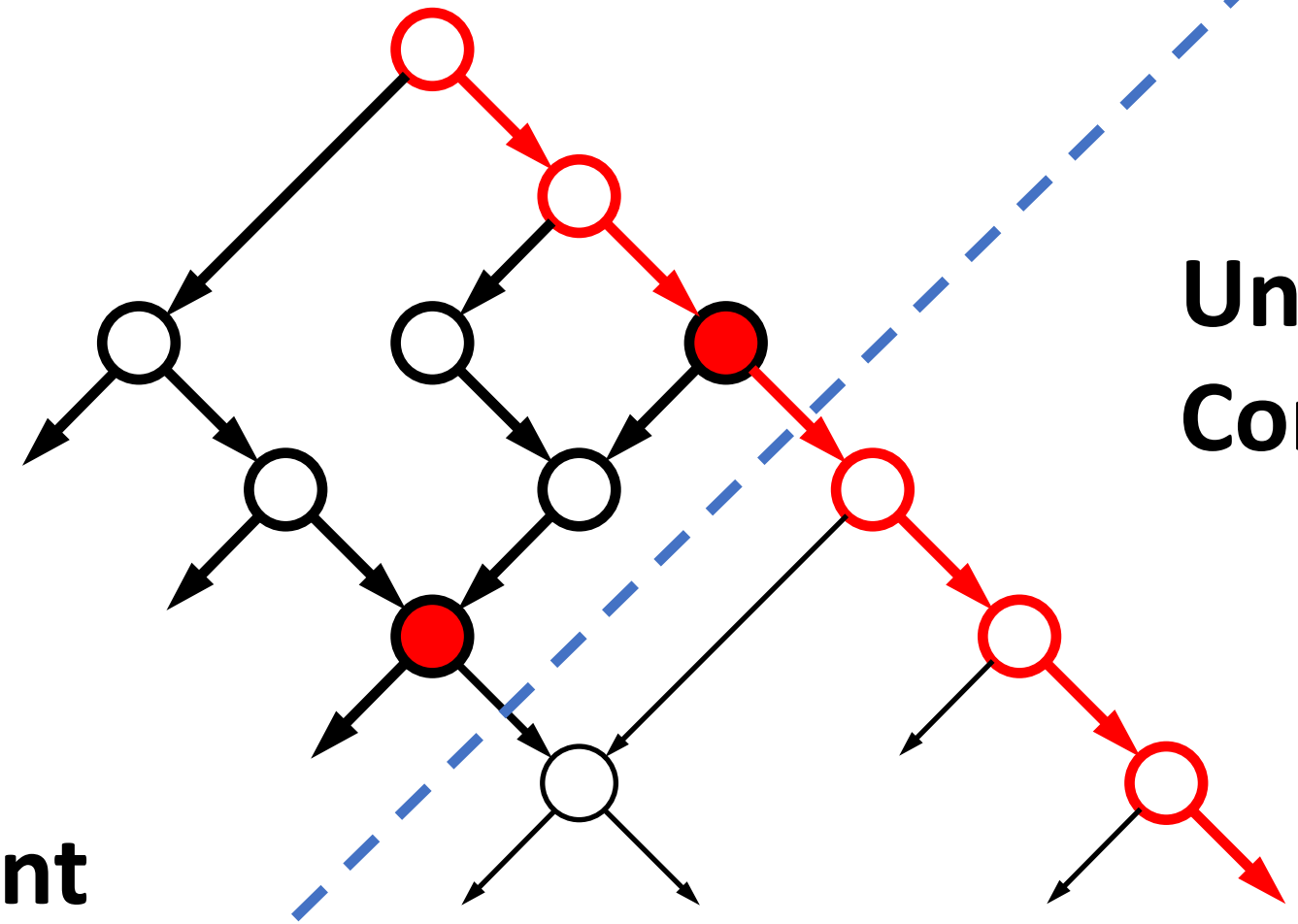
needed to pass through desired

conditional jump and break through

to an unanalyzed compartment



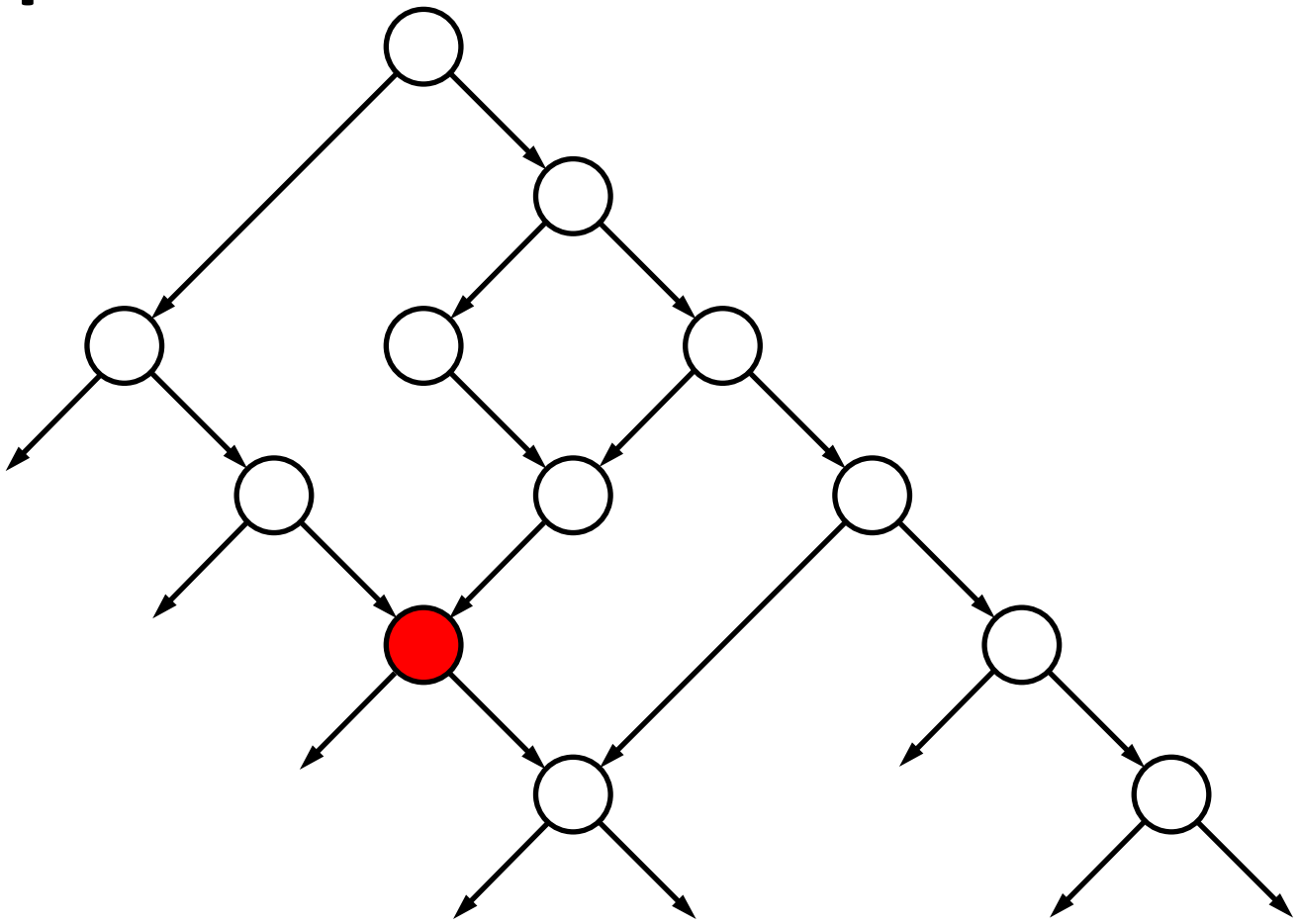
Input to pass through constant comparison jump



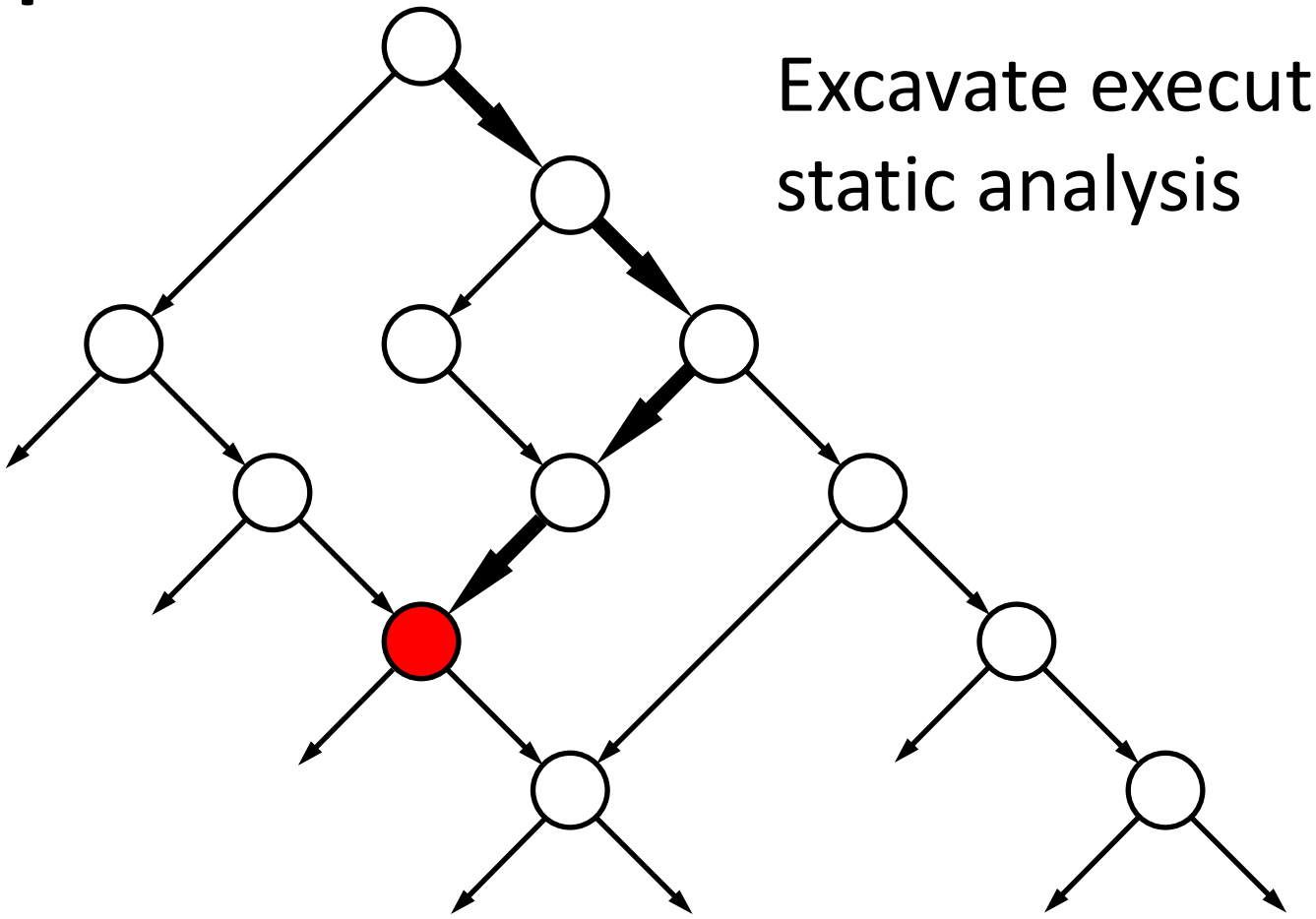
Unanalyzed Compartment

Analyzed Compartment

Combining dynamic and static program analysis for reaching specific instruction

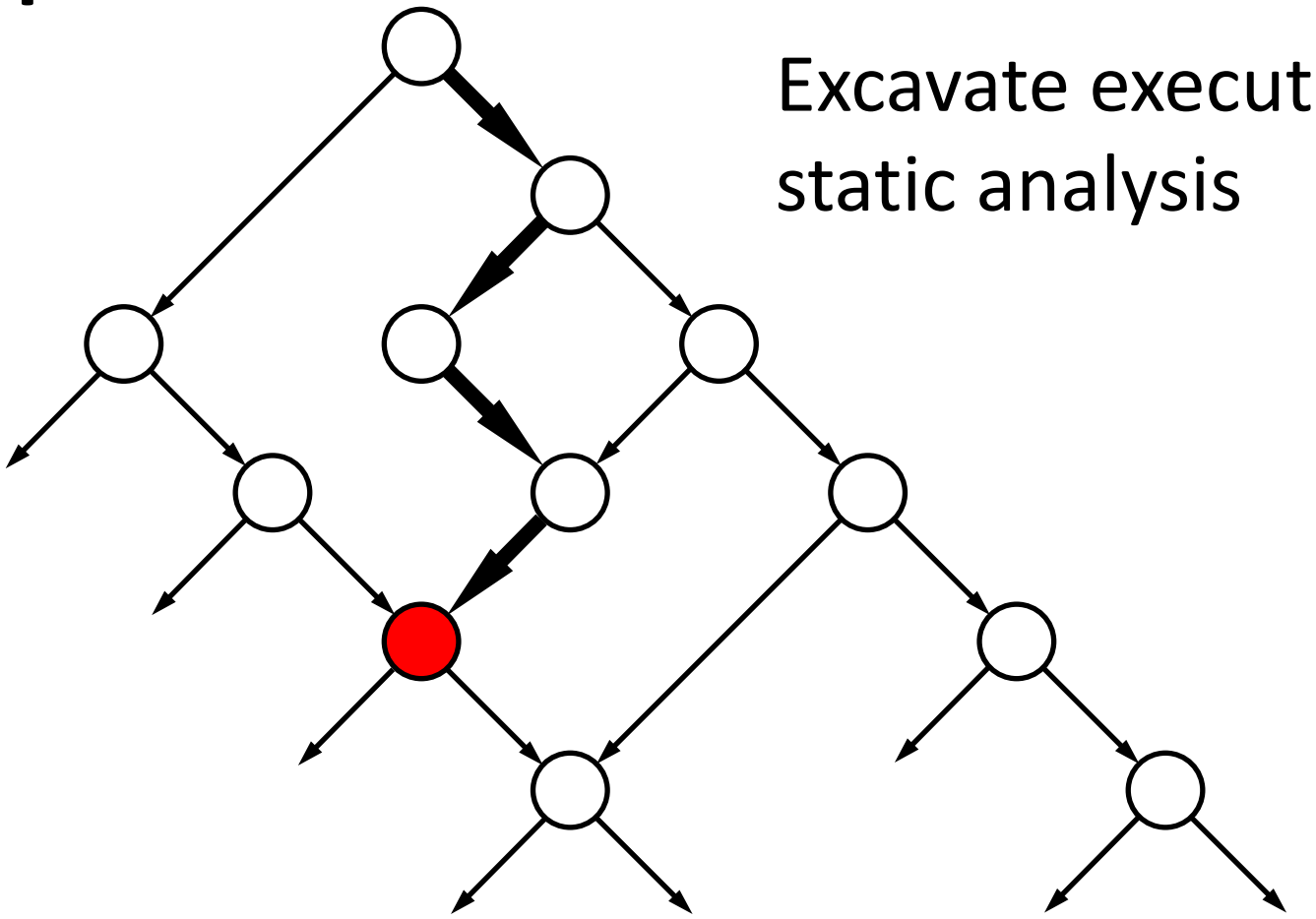


Combining dynamic and static program analysis for reaching specific instruction



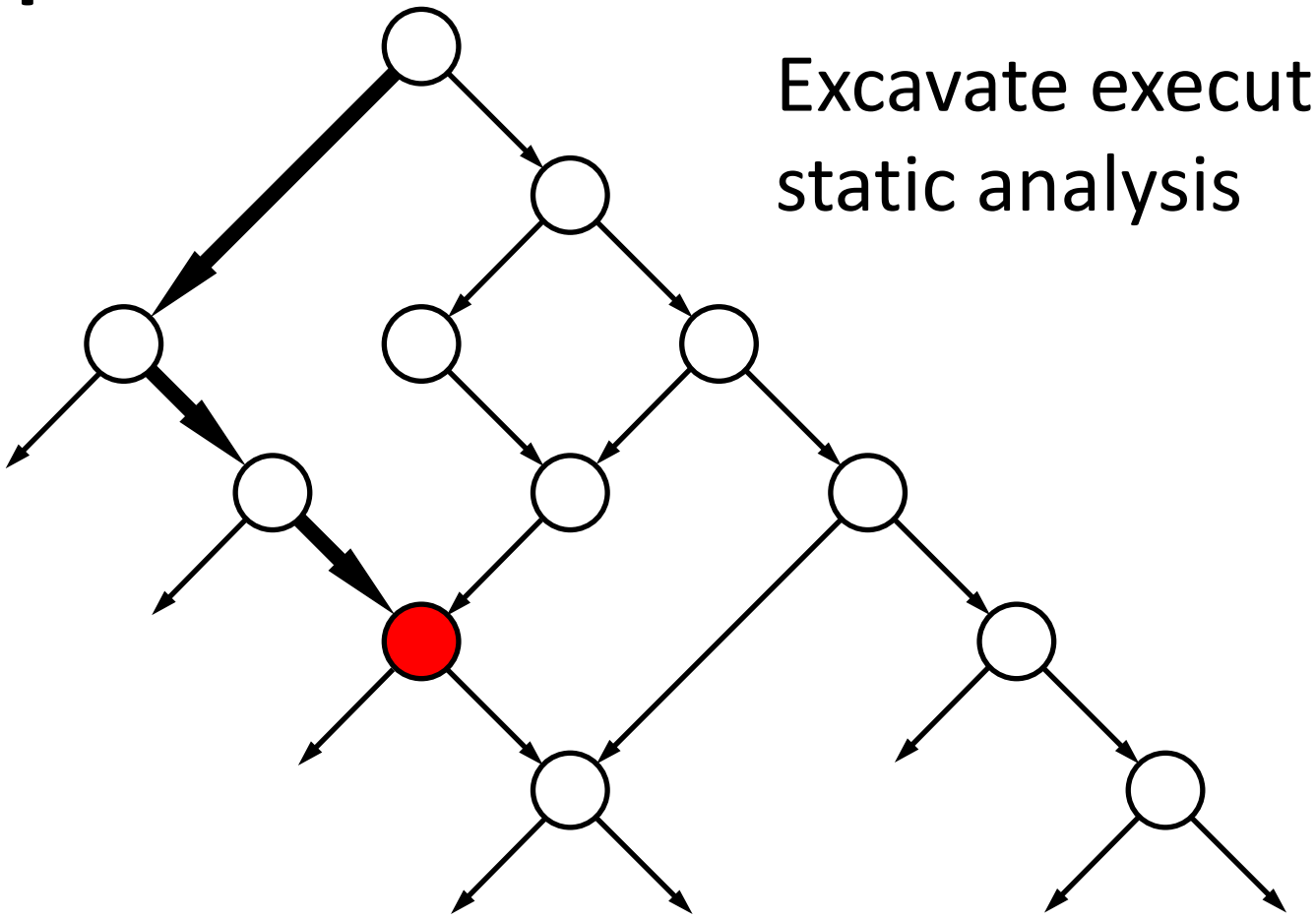
Excavate execution paths with static analysis

Combining dynamic and static program analysis for reaching specific instruction



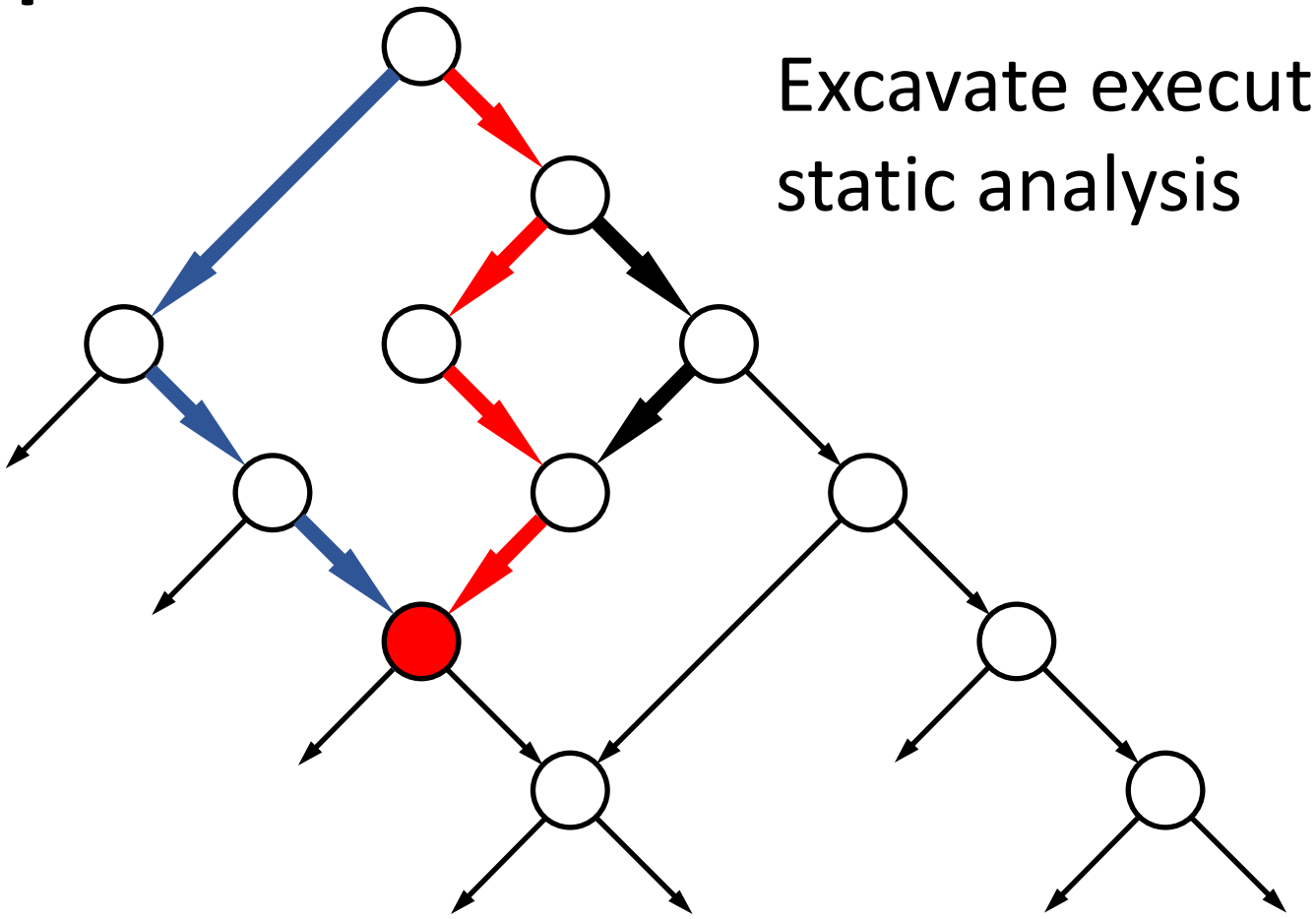
Excavate execution paths with static analysis

Combining dynamic and static program analysis for reaching specific instruction



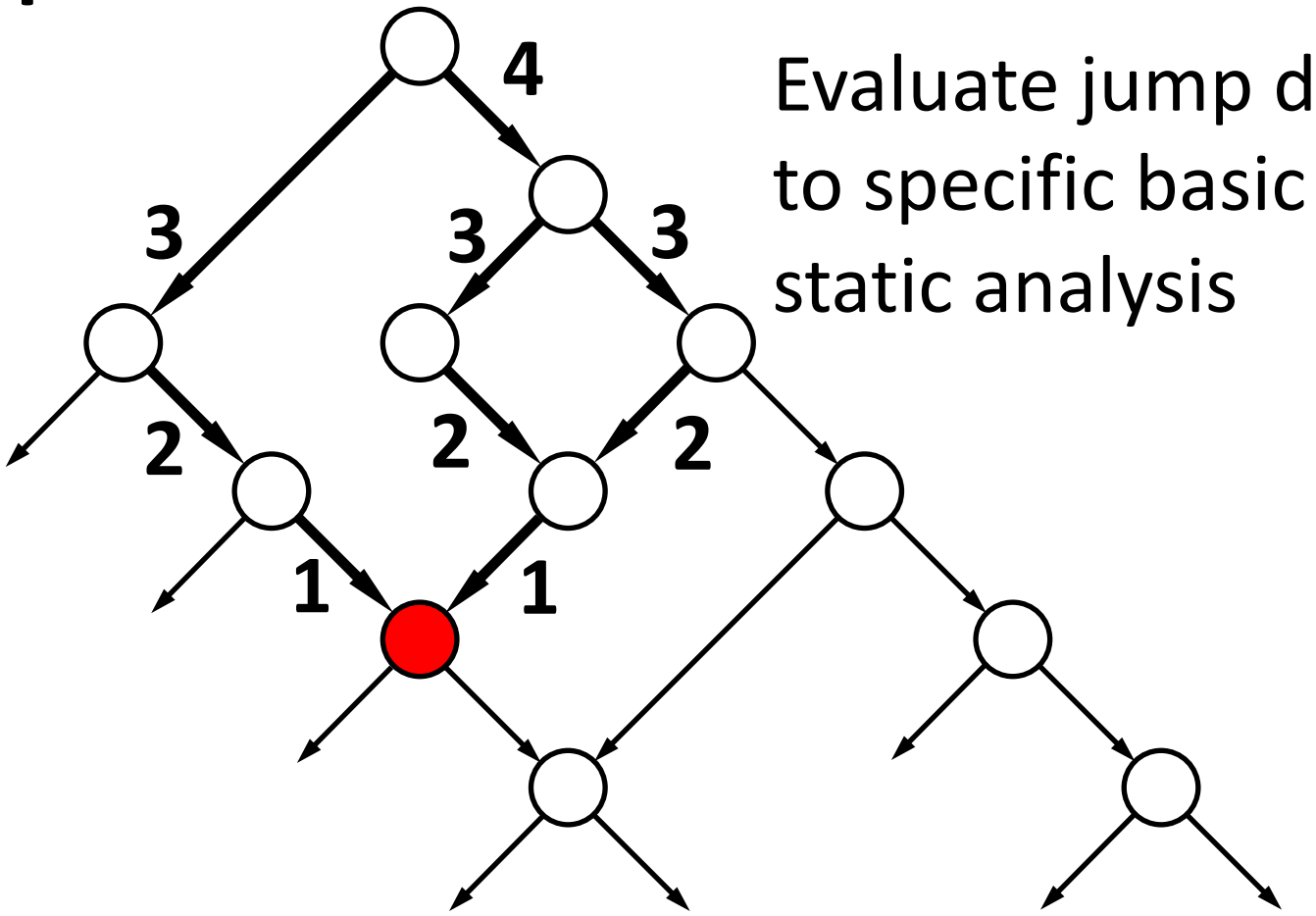
Excavate execution paths with static analysis

Combining dynamic and static program analysis for reaching specific instruction



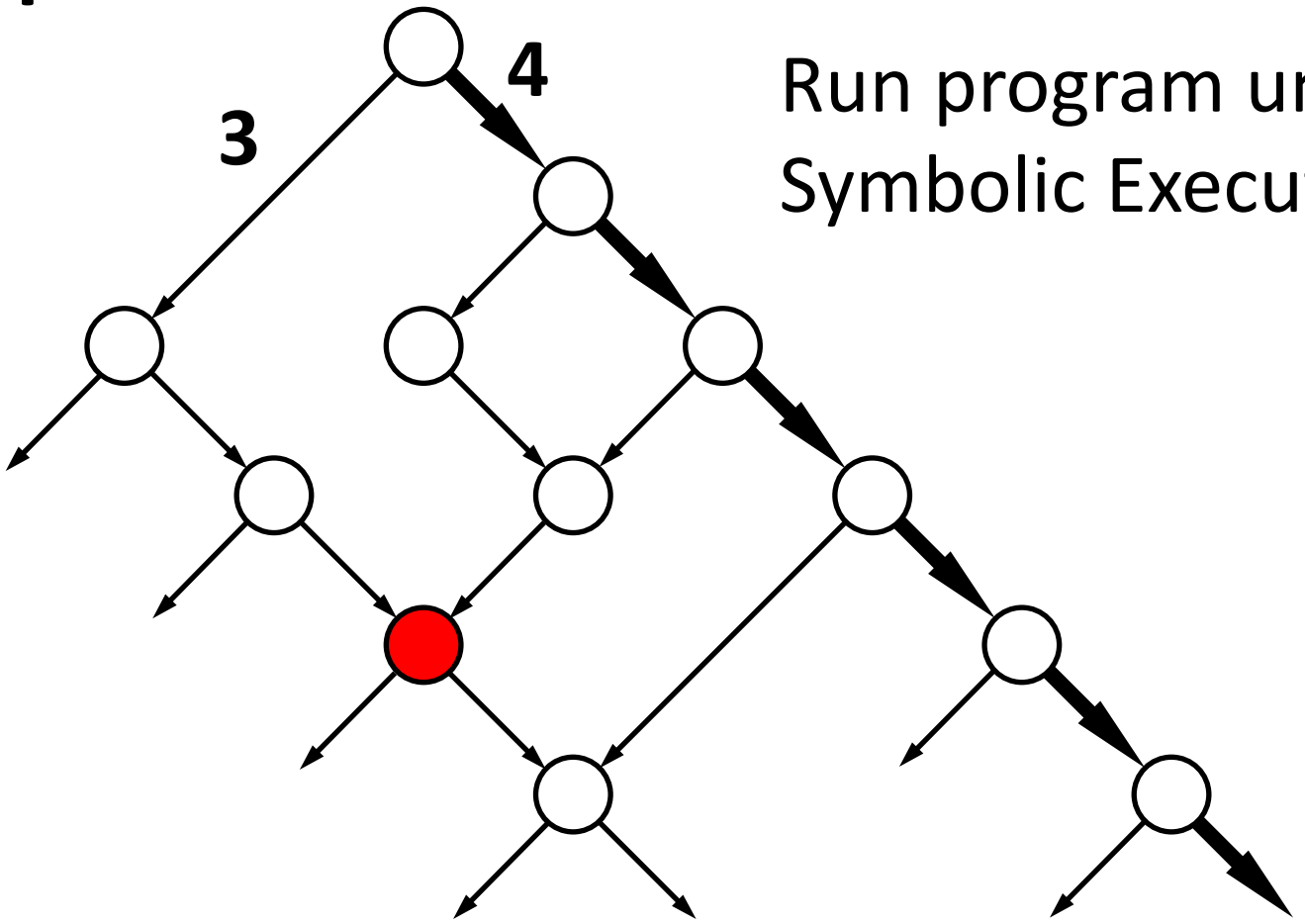
Excavate execution paths with static analysis

Combining dynamic and static program analysis for reaching specific instruction



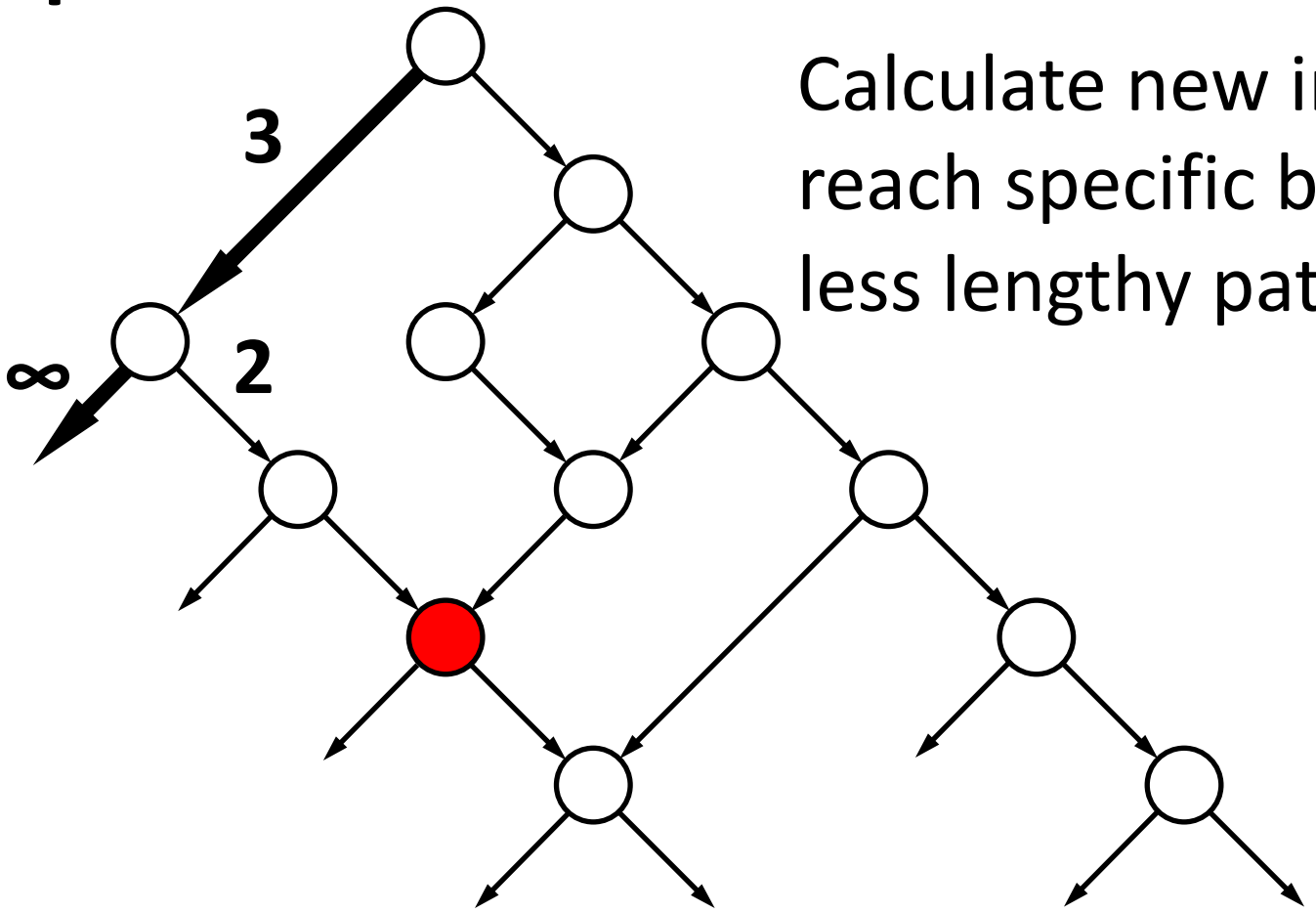
Evaluate jump direction length to specific basic block with static analysis

Combining dynamic and static program analysis for reaching specific instruction



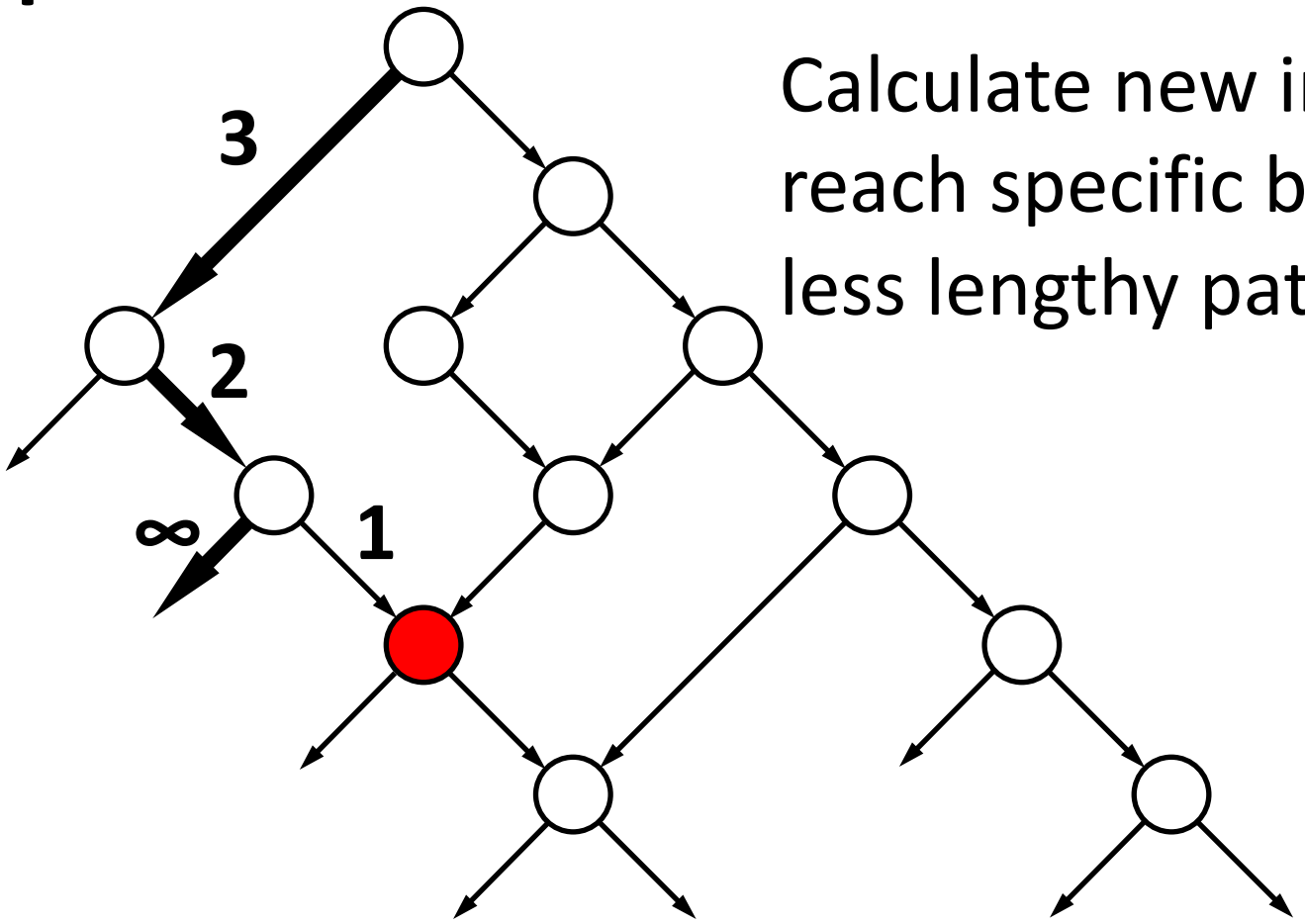
Run program under Dynamic Symbolic Execution

Combining dynamic and static program analysis for reaching specific instruction



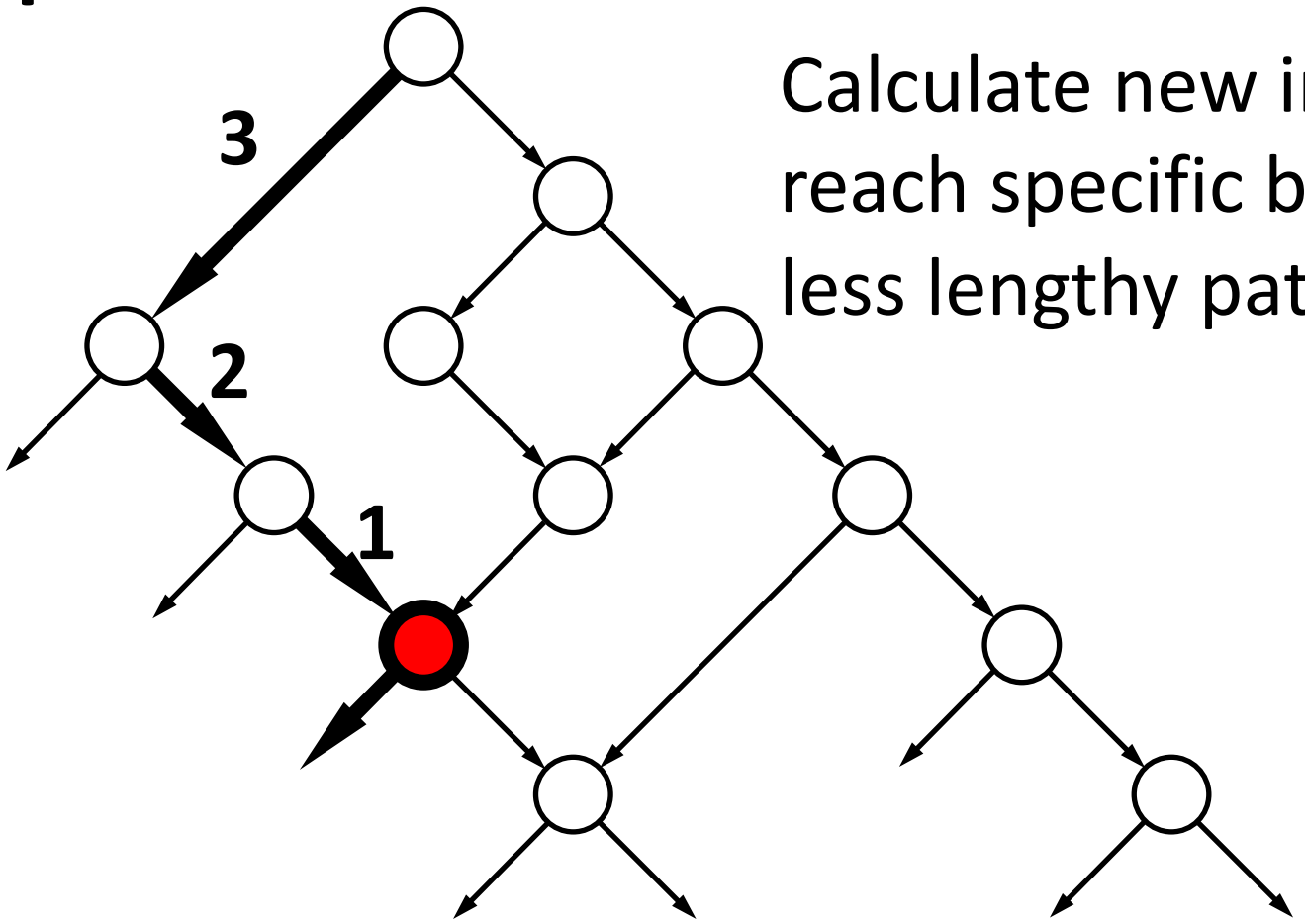
Calculate new input data to reach specific basic block by less lengthy path

Combining dynamic and static program analysis for reaching specific instruction



Calculate new input data to reach specific basic block by less lengthy path

Combining dynamic and static program analysis for reaching specific instruction



Calculate new input data to reach specific basic block by less lengthy path

Further research

- Indirect jumps
- Indirect taint data dependencies
- Defect formulae for SMT solver

Questions session