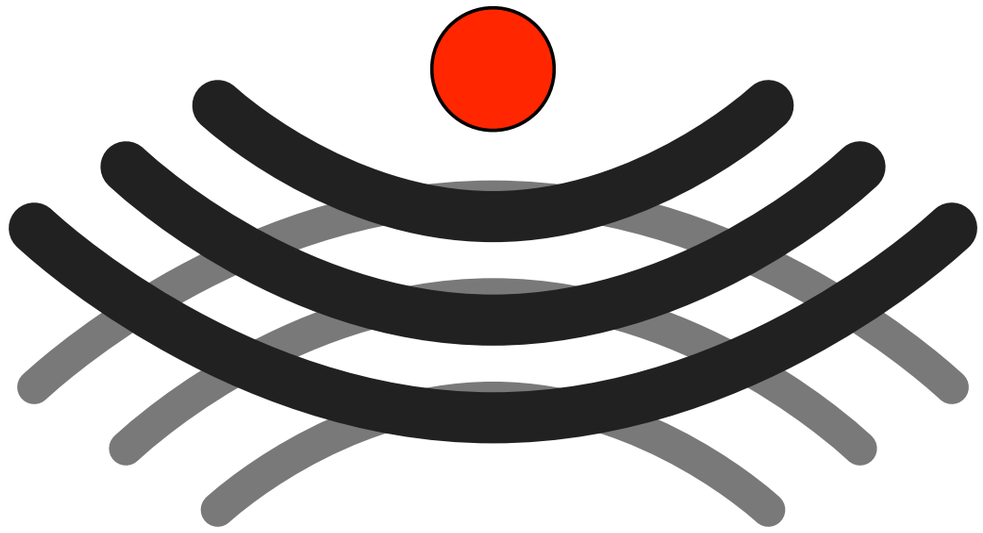
KLEE's Sonar-Search

Reviewing in the context of Greybox Fuzzing

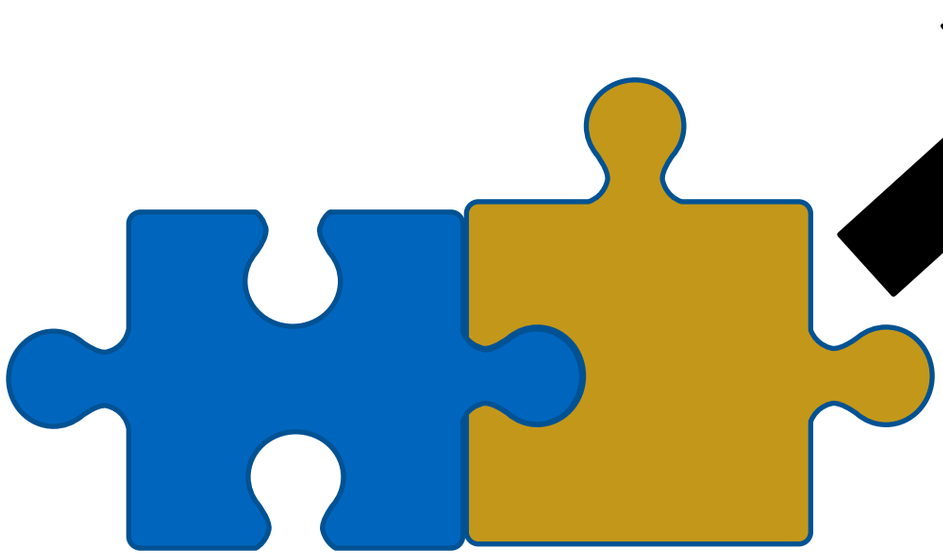
Saahil Ognawala, Alexander Pretschner, Thomas Hutzelmann,
Eirini Psallida, Ricardo Nales

Technical University of Munich, Germany

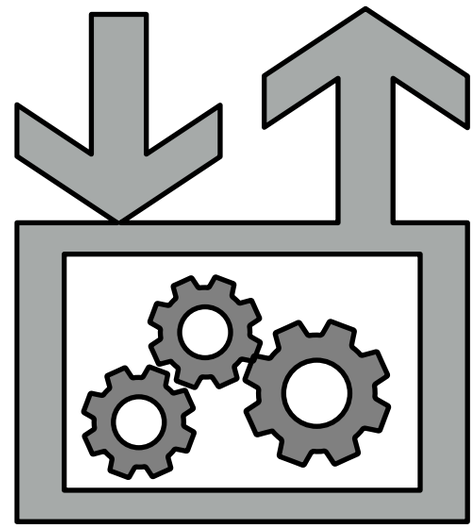
Agenda



Sonar-search



Compositional analysis



Greybox Fuzzing

Motivation

- Symbolic execution and fuzzing - insufficient coverage => vulnerabilities not discovered
- Idea - Analyze individual functions (C programs) for vulnerabilities
  - Do reachability analysis for vulnerabilities using symbolic execution
- But what about path explosion?

Targeted symbolic execution

- To tackle path-explosion
- Set targets (function entry points or vulnerable instructions in functions)
- Terminate states that do not reach targets-of-interest

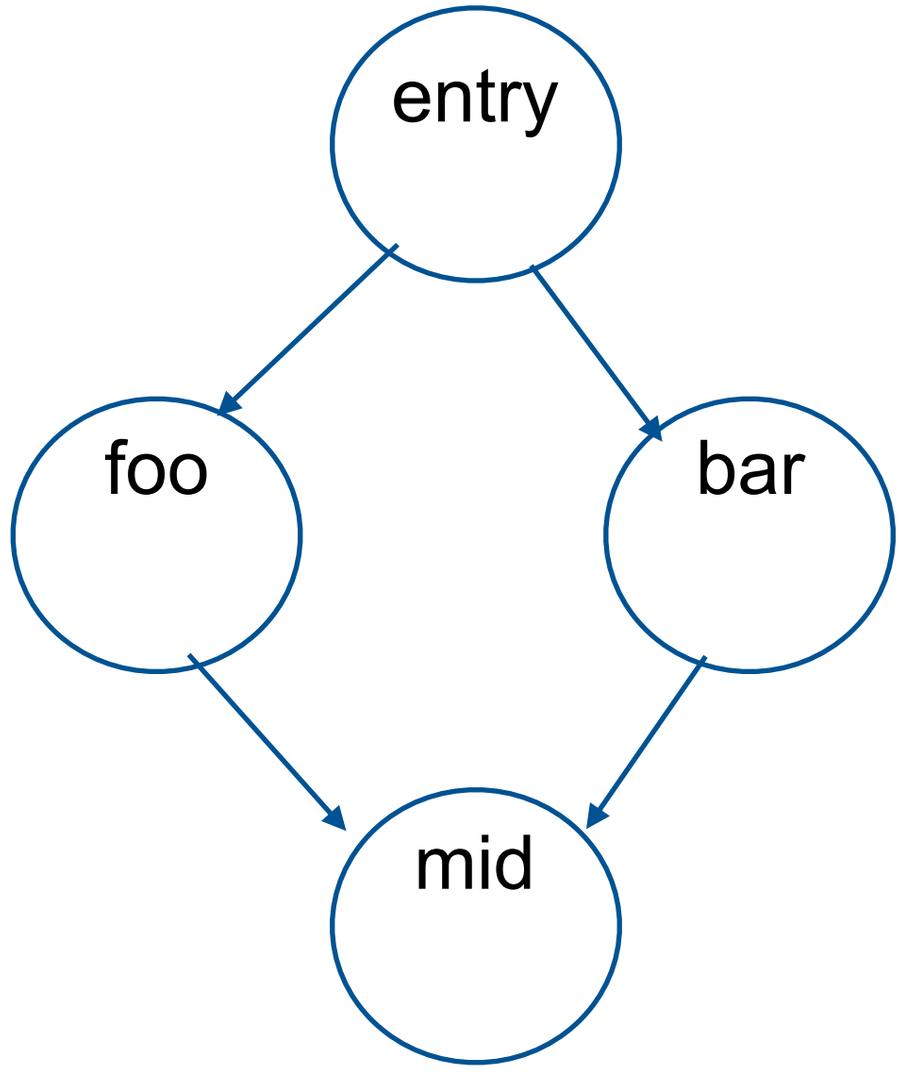
Sonar-search

- Implementation of targeted-search in KLEE — *KLEE22*
- Target may be *function-call*, *function-return, LLVM bb* or *klee_assert*
- Nested searcher
  - nurs:covnew when target is reached

Sonar-search

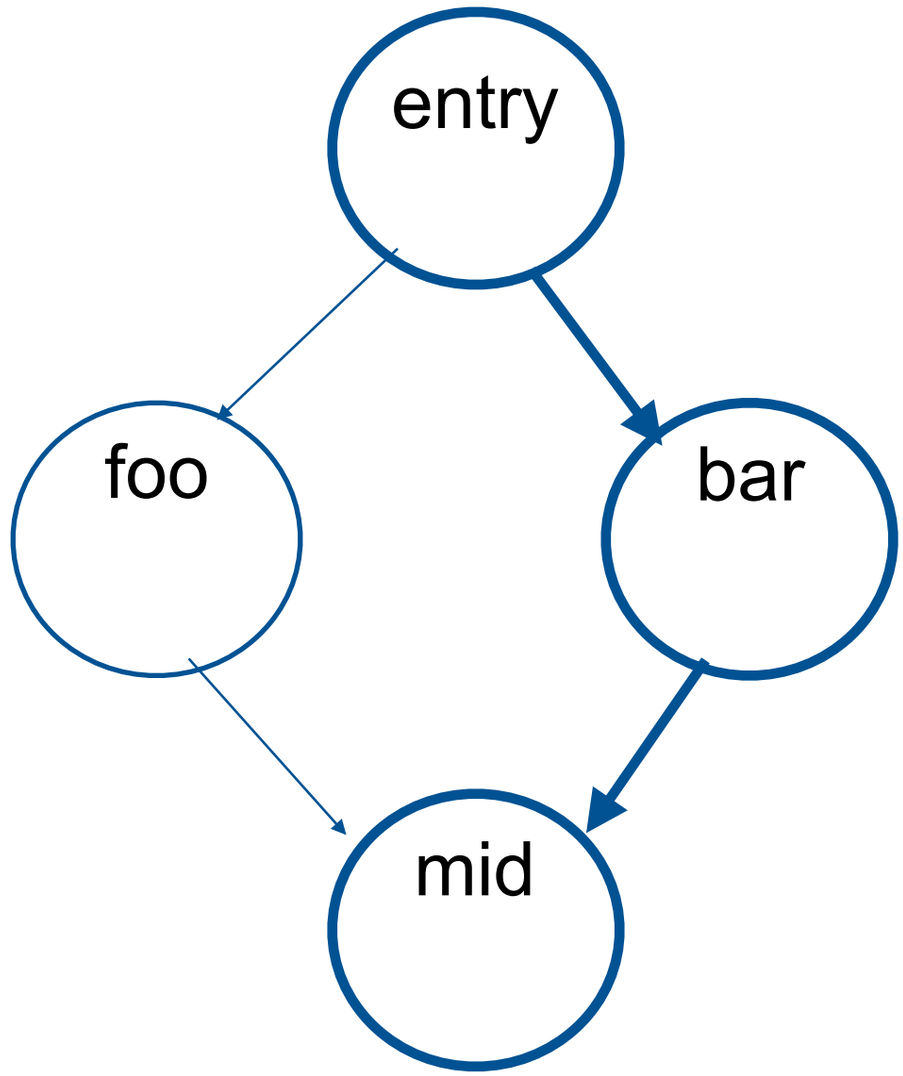
```
void mid() {
  // <- position of the current execution state
  return;
}

void foo() {
  mid();
  // <- target
}

void bar() {
  mid();
}

void entry(int a) {
  if (a > 0) // target reachable
    foo();
  else // target unreachable
    bar();
}
```

Three cases for sonar-search to consider



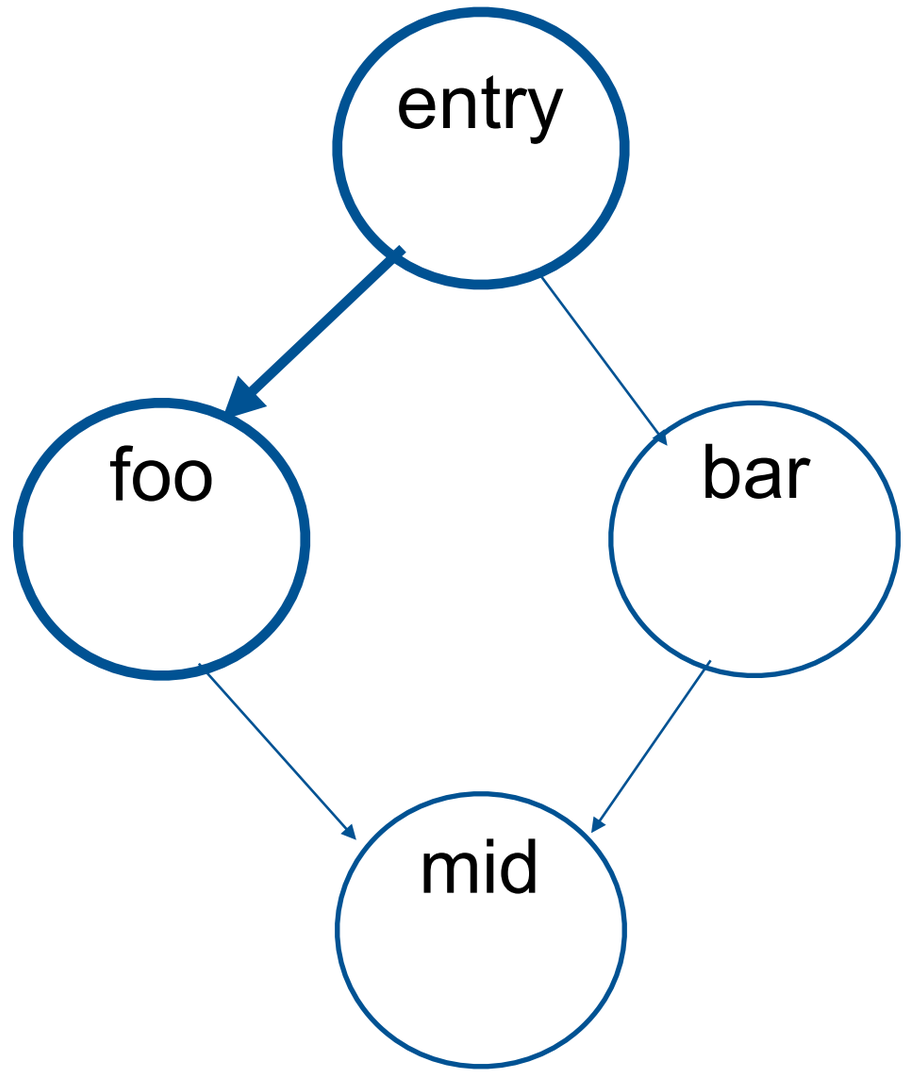
Sonar-search

```
void mid() {
  // <- position of the current execution state
  return;
}

void foo() {
  mid();
  // <- target
}

void bar() {
  mid();
}

void entry(int a) {
  if (a > 0) // target reachable
    foo();
  else // target unreachable
    bar();
}
```



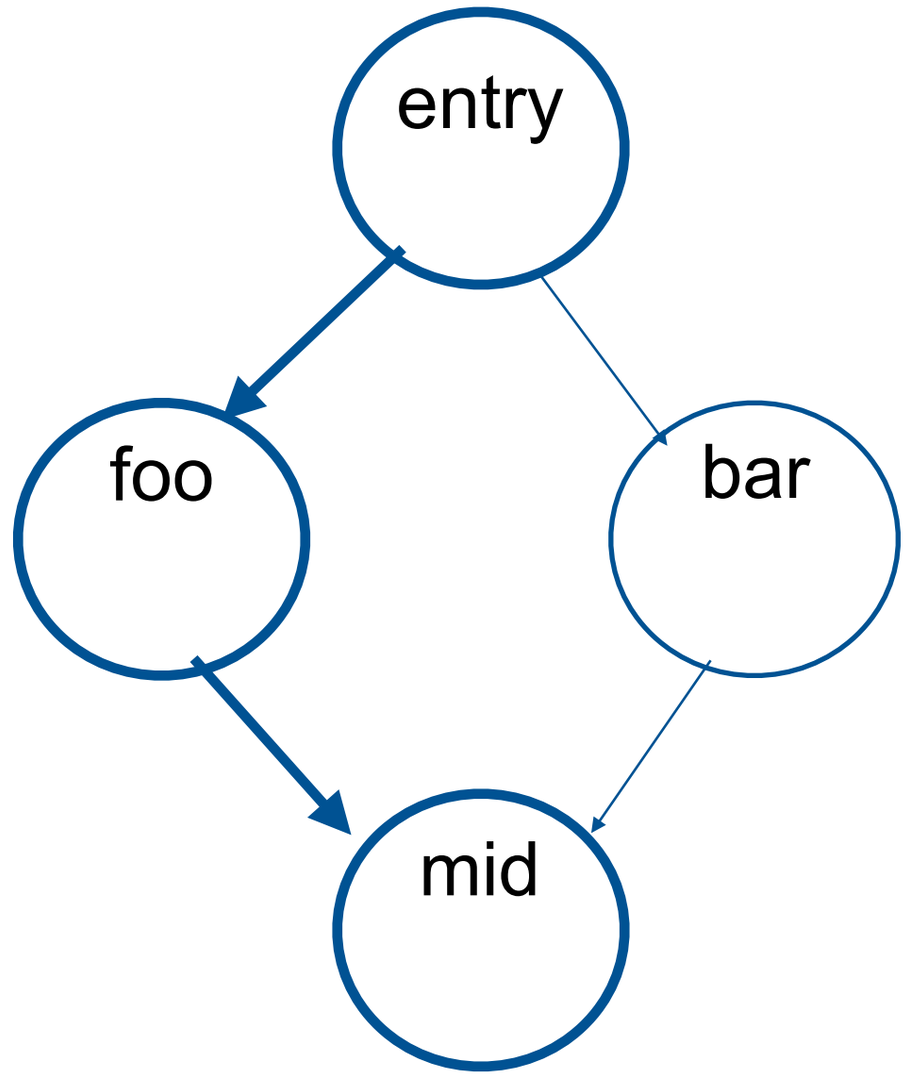
Sonar-search

```
void mid() {
  // <- position of the current execution state
  return;
}

void foo() {
  mid();
  // <- target
}

void bar() {
  mid();
}

void entry(int a) {
  if (a > 0) // target reachable
    foo();
  else // target unreachable
    bar();
}
```



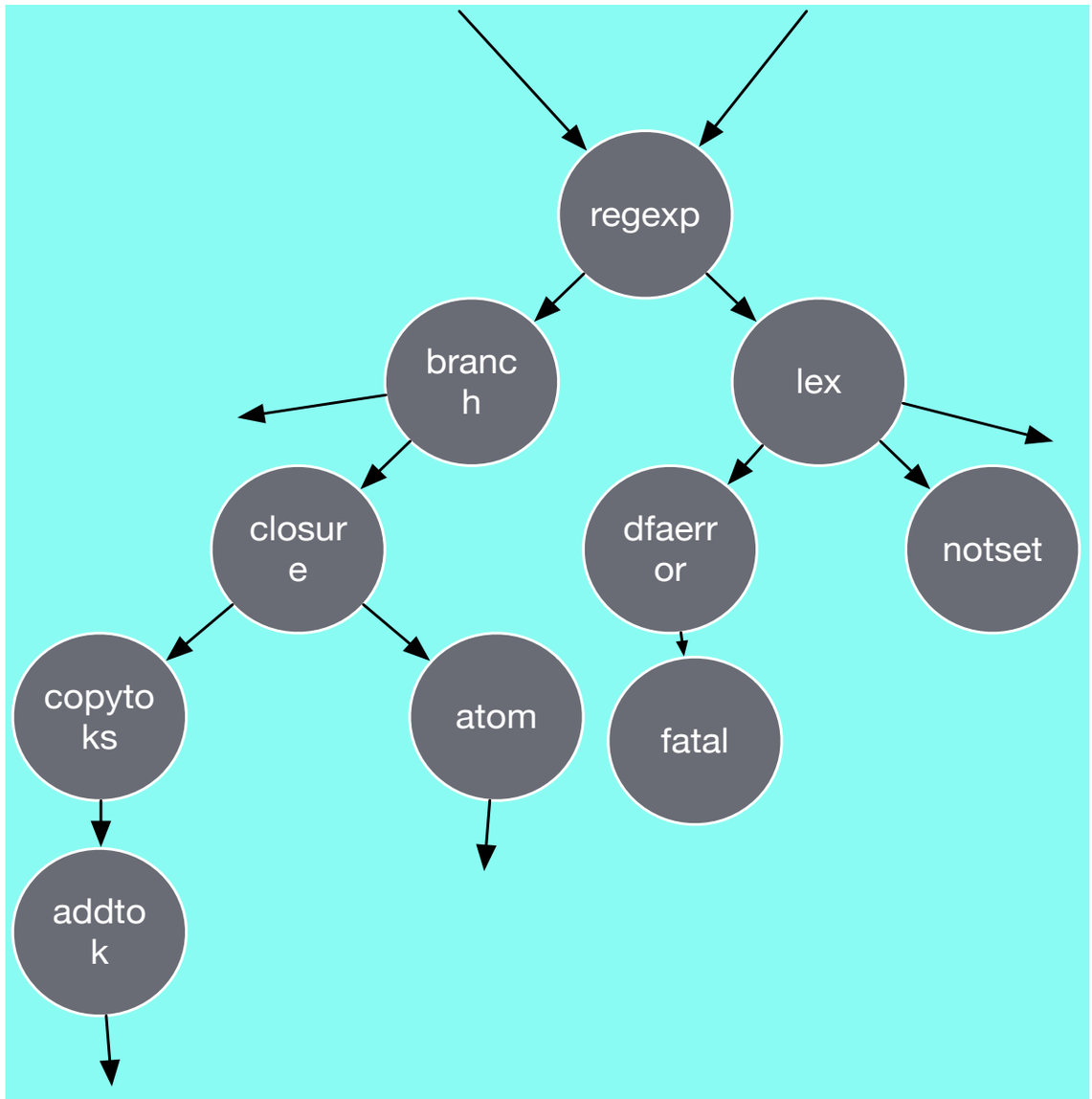
Sonar-search

```
void mid() {
  // <- position of the current execution state
  return;
}

void foo() {
  mid();
  // <- target
}

void bar() {
  mid();
}

void entry(int a) {
  if (a > 0) // target reachable
    foo();
  else // target unreachable
    bar();
}
```



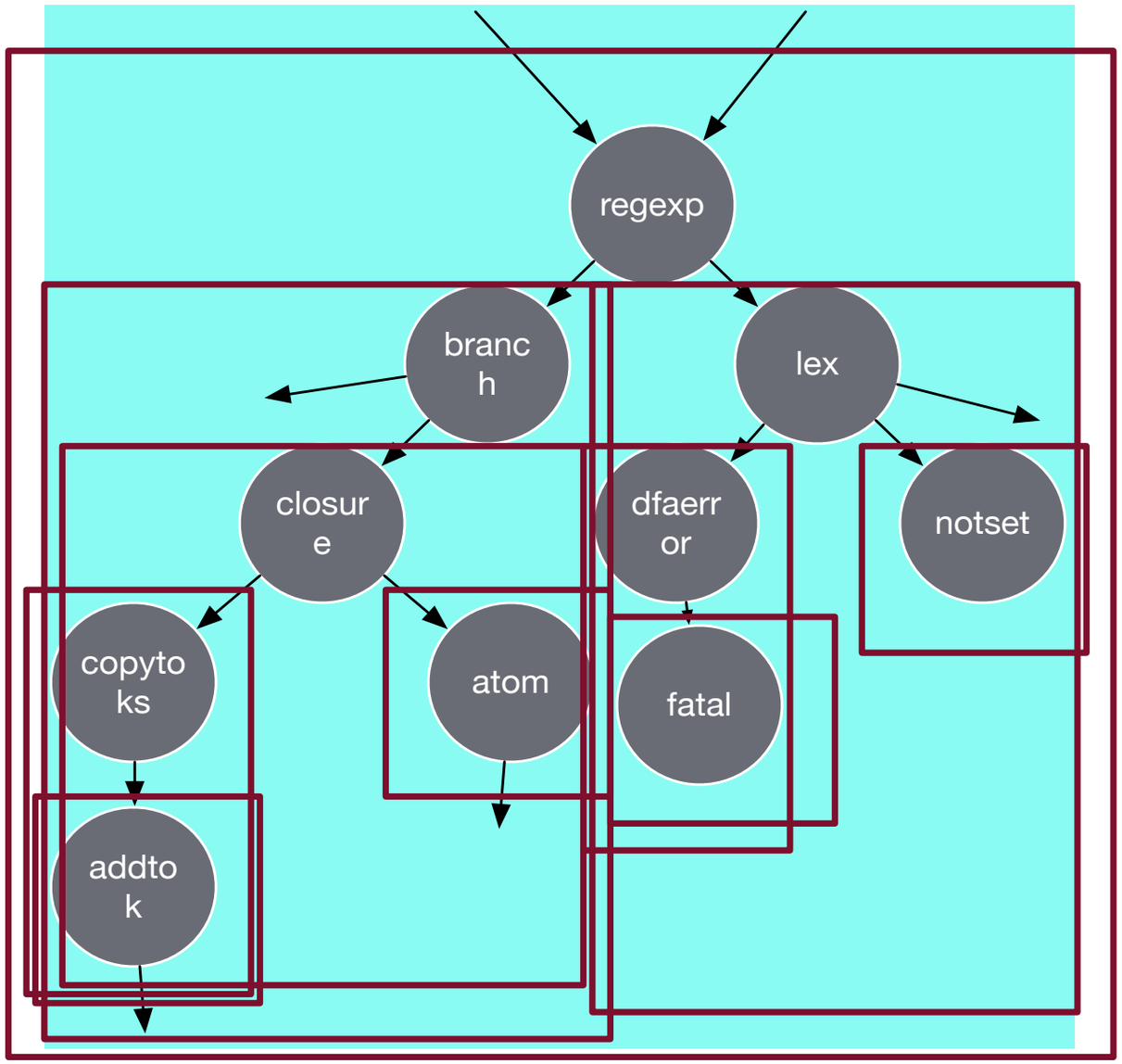
Compositional symbolic execution with *Macke*

Real-life programs too big



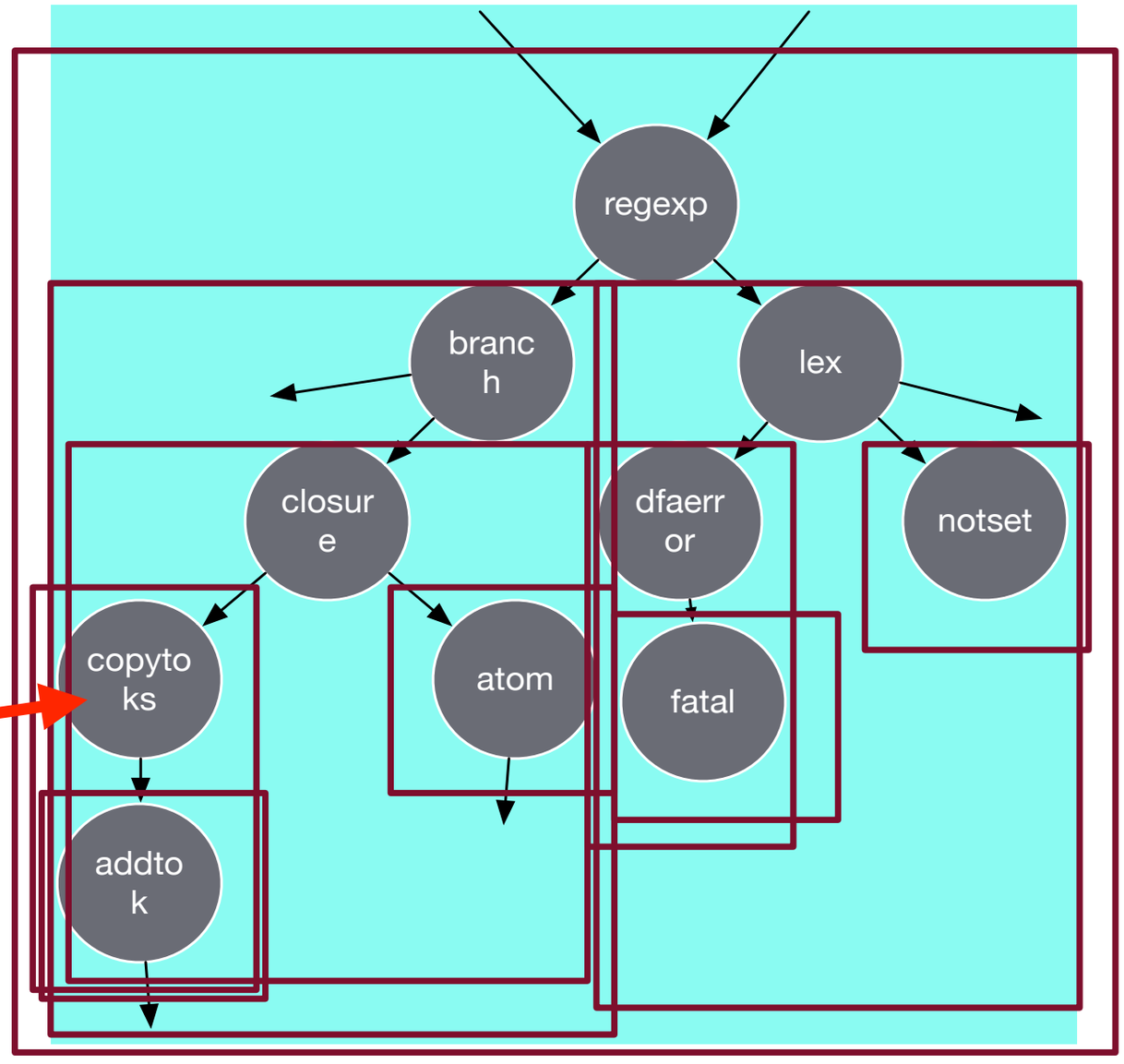
Partial call-graph of Grep 2.25

Symbolic execution of isolated functions



Partial call-graph of Grep 2.25

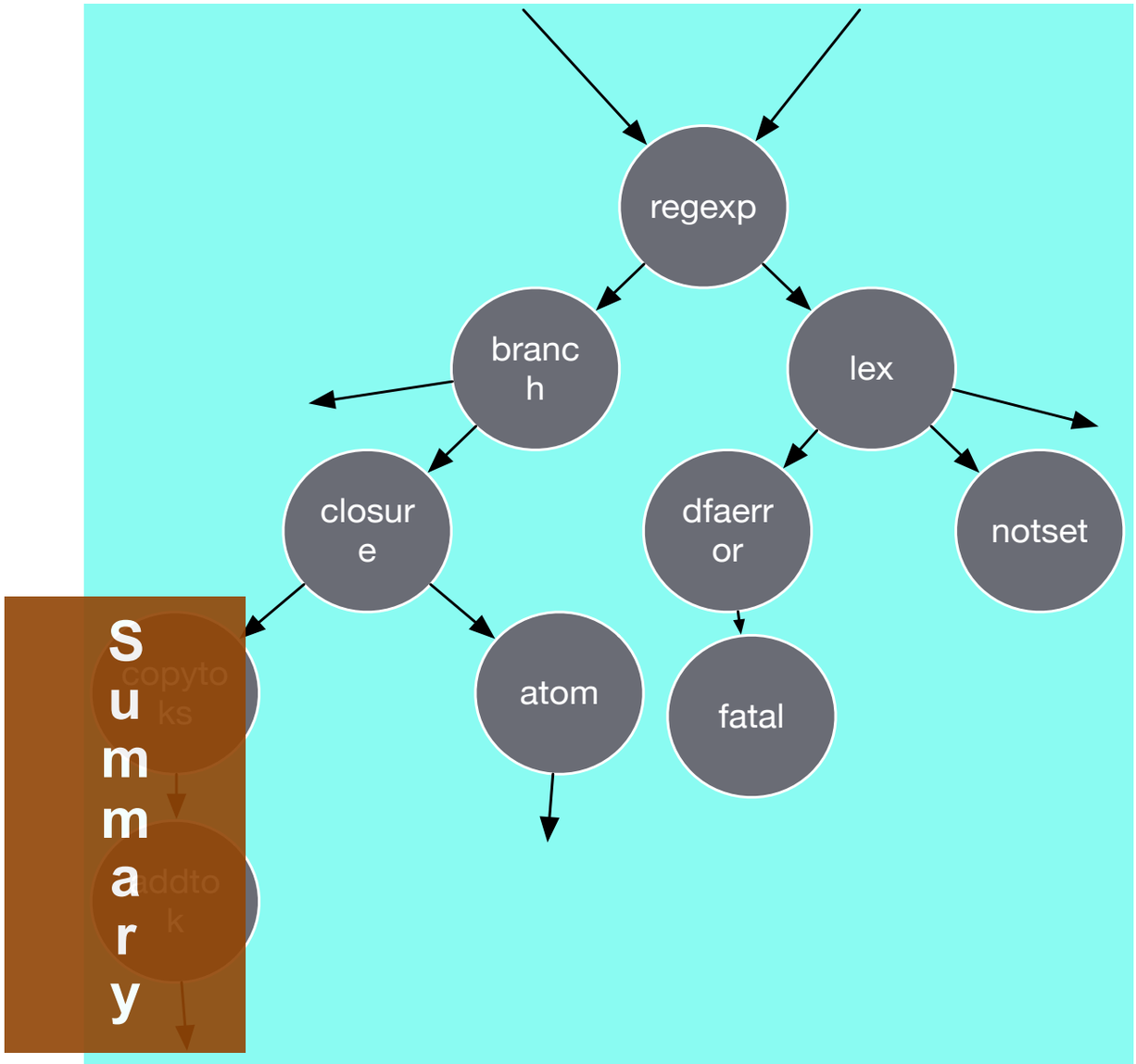
Symbolic execution of isolated functions



Vulnerability here

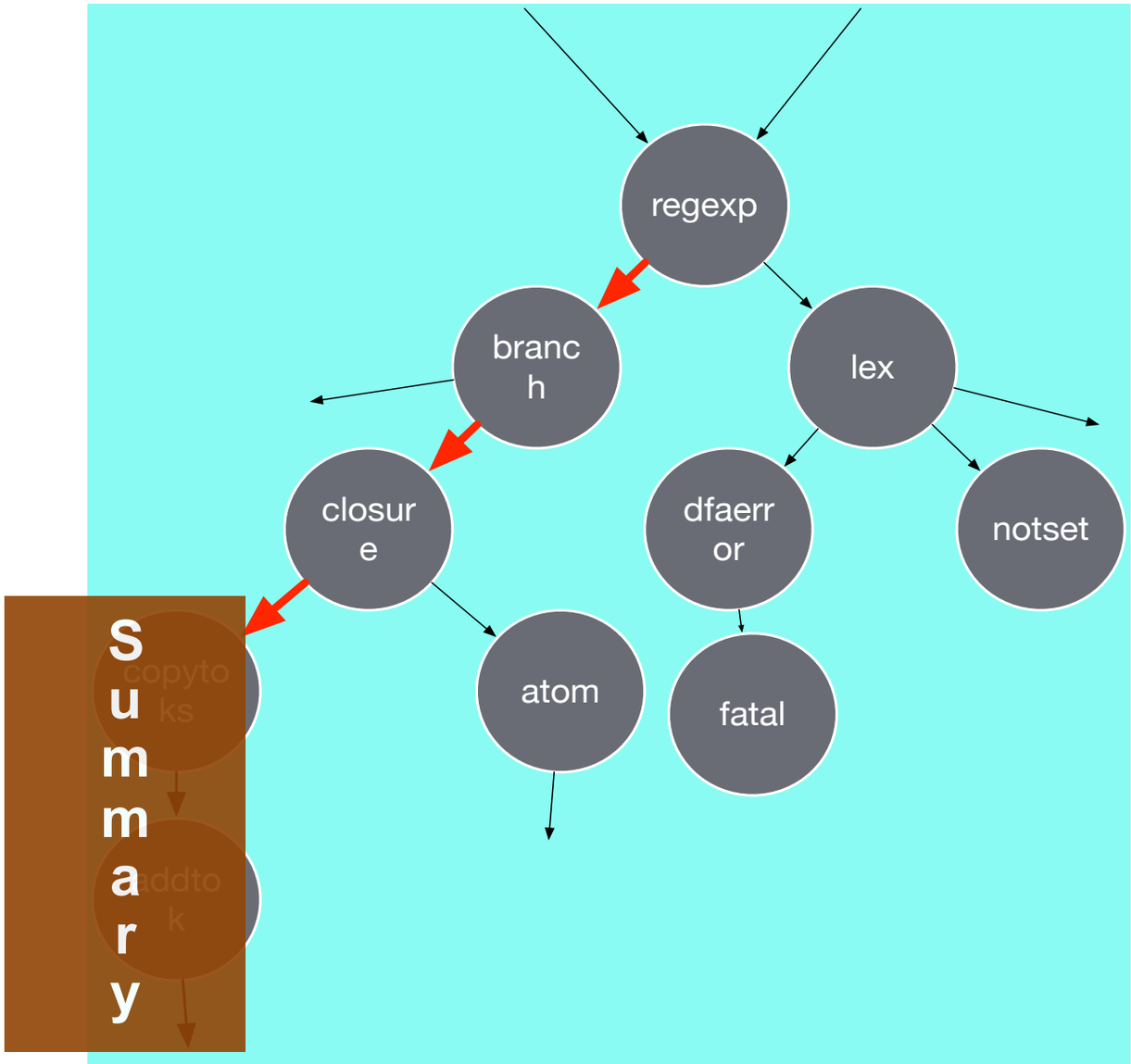
Partial call-graph of Grep 2.25

Summarization of vulnerability as PC (or PC solution)



Partial call-graph of Grep 2.25

Sonar-search to vulnerable functions



Macke - Results

Program	MACKE only	both	KLEE only
bc 1.06	5	0	0
bison 3.0.4	1	0	0
bzip2 1.0.6	0	0	0
coreutils 6.10	24	28	0
coreutils 8.25	34	4	0
diff 3.4	0	1	0
flex 2.6.0	0	0	0
flex SIR	2	3	0
goahead 3.6.3	0	0	0
grep 2.25	0	0	0
grep SIR	0	2	0
jq 1.5	0	0	0
less 481	0	1	0
lz4 r131	0	1	0
ngircd 23	1	0	0
sed 4.2.2	0	0	0
tar 1.29	0	1	0
zopfli 1.0.1	0	0	0
in total	67	41	0

Increasing function coverage with *Munch*

Symbolic execution vs. Fuzzing

- Path explosion and constraint solver in symbolic execution -> Low path coverage in “deep” parts of program
- Weak input (mutation) strategy with fuzzing -> Low path coverage everywhere.
- Idea: Greybox fuzzing (symbolic execution + fuzzing)

Munch - Greybox Fuzzer for Function Coverage

FS hybrid

- Fuzzing for limited time
- List uncovered functions
- *Sonar-search* to uncovered functions

SF hybrid

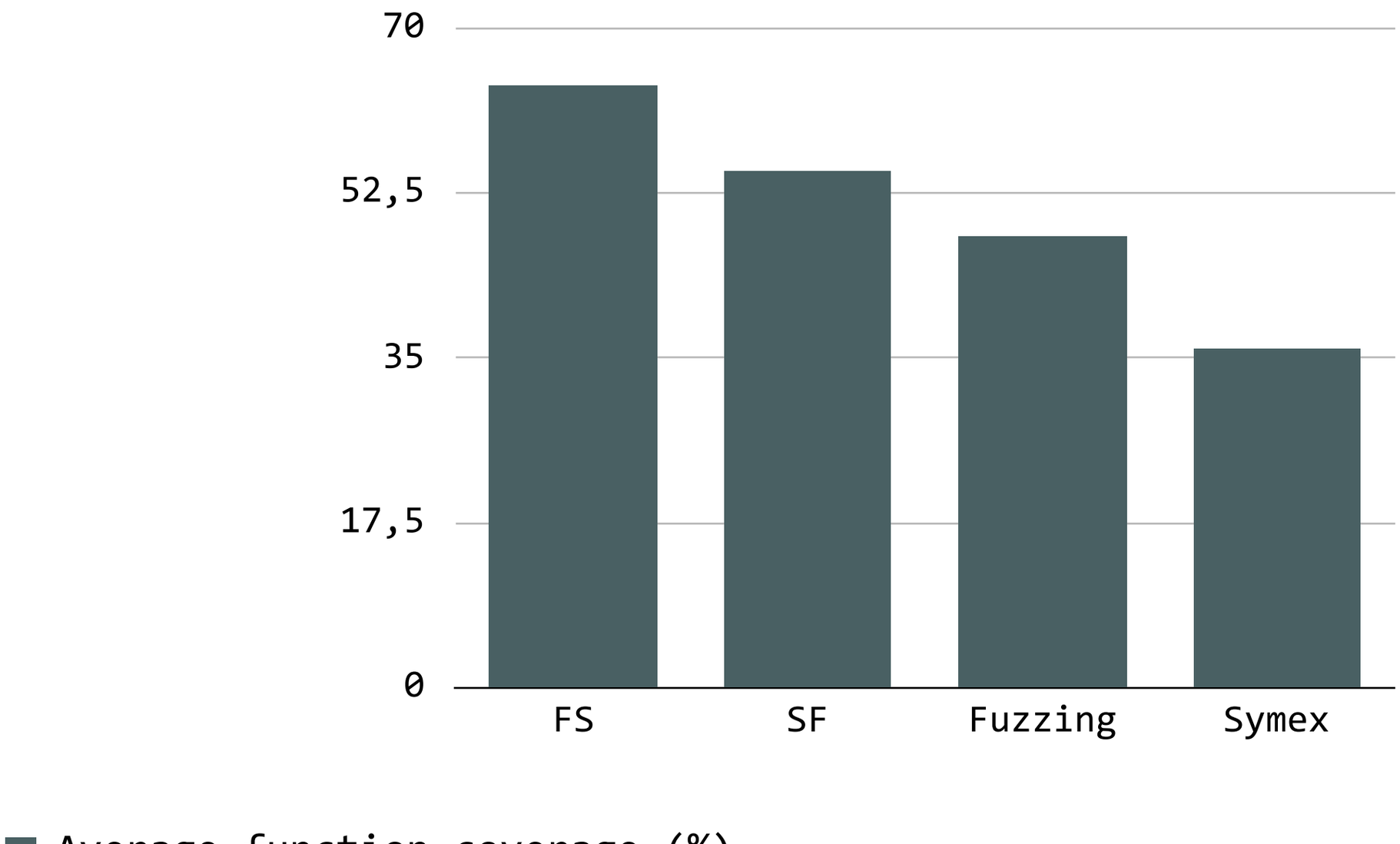
- Symbolic execution (KLEE default strategy) for limited time
- Fuzzing with seed inputs generated by KLEE

Munch - Results

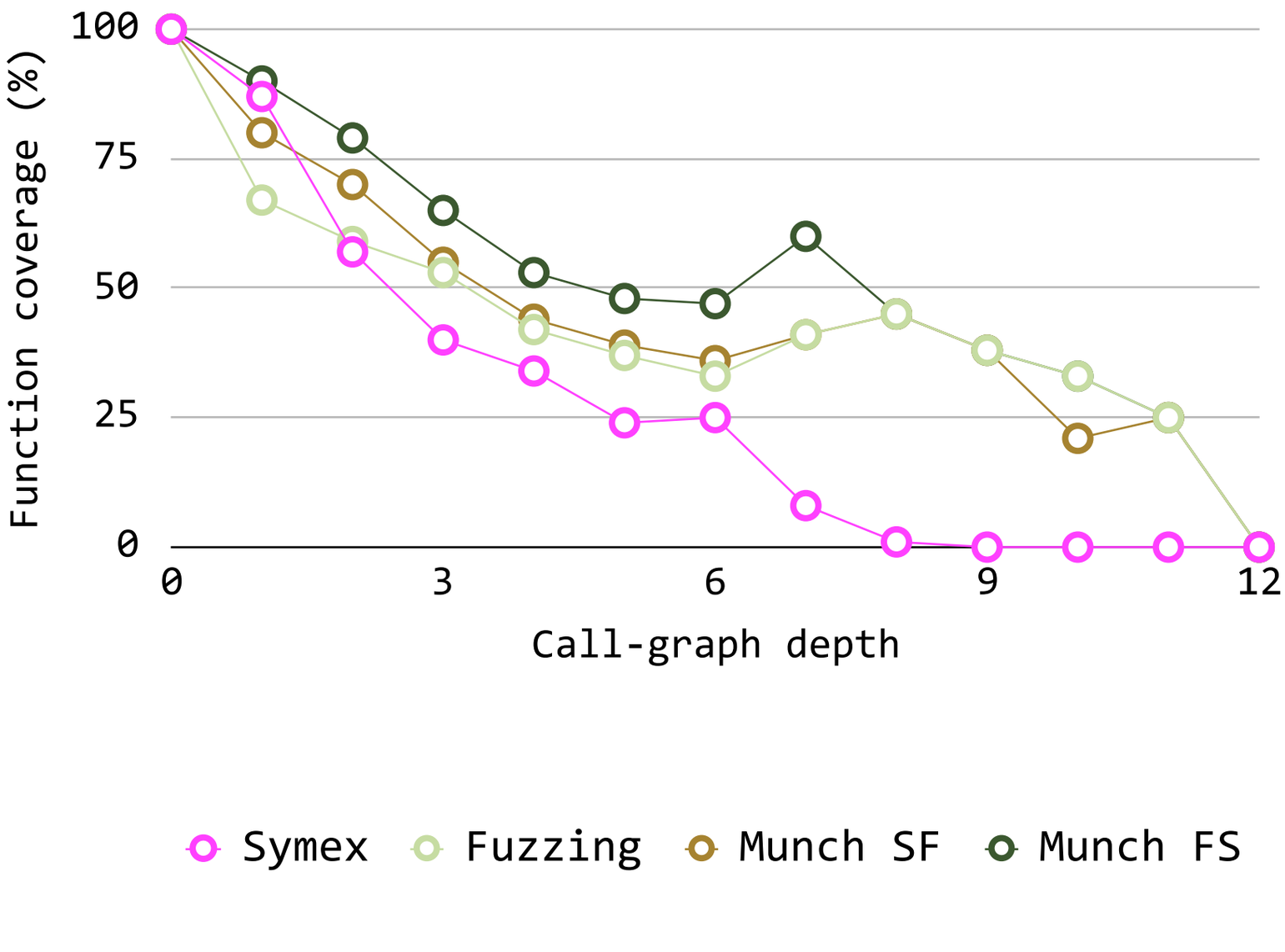
- 9 C-programs (⊂ Macke)

Technique	Time
Munch FS	~ 2 hours
Munch SF	2 hours
Symbolic execution	5 hours
Fuzzing	5 hours

Munch - Results

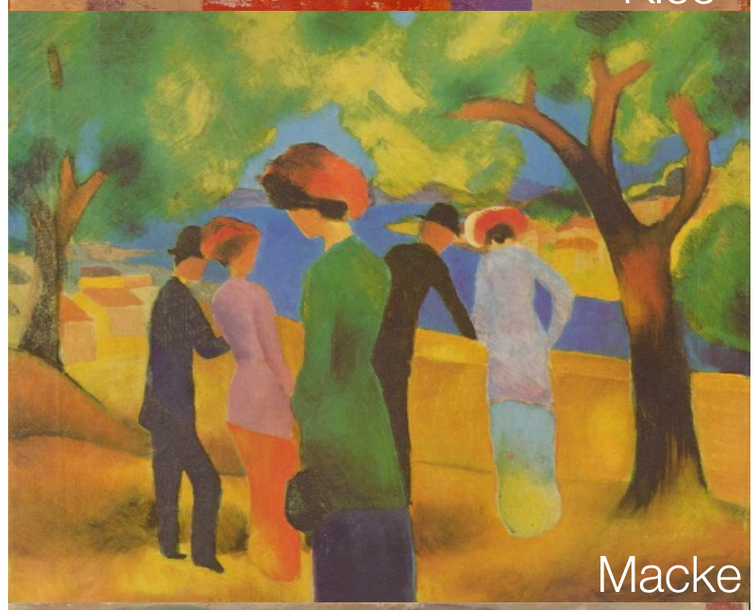


Munch - Results



Conclusion

- Sonar-search (KLEE22) for targeted symbolic execution
- Two use-cases of sonar-search
  - Compositional analysis of C programs
  - Greybox fuzzing to increase function coverage
- Compositional analysis performs better in terms of vulnerability discovery than symbolic execution
- Greybox fuzzing performs better in terms of function coverage, than symbolic execution or blackbox fuzzing.



References

- Ognawala, S., Ochoa, M. et al. "MACKE: Compositional analysis of low-level vulnerabilities with symbolic execution." *Automated Software Engineering (ASE), 2016*
  - https://github.com/tum-i22/macke
- Ognawala, S., Hutzelmann, T. et al. "Improving Function Coverage with Munch: A Hybrid Fuzzing and Directed Symbolic Execution Approach." *Symposium for Applied Computing (ACM SAC), 2018*
  - https://github.com/tum-i22/munch
- KLEE22 with Sonar-search - https://github.com/tum-i22/klee22 (branch "*sonar*")
- Severity assessment tool - https://vmpretschner18.informatik.tu-muenchen.de/
- ognawala@in.tum.de