Conditional Testing

Off-the-Shelf Combination of Test-Case Generators

Thomas Lemberger
Joint work with Dirk Beyer

LMU Munich, Germany

2019-10-29, ATVA 2019

ESBMC AFL
KLUZZER
Gramfuzz
Redqueen
zzuff
QSYM AFL-fast
NEUZZ
Symbiotic
AFLGo
CoVeriTest
DART CPA/Tiger-MGP
VUzzer
PathCrawler Driller
PRTest libFuzzer FShell
Legion FairFuzz CREST
Tracer-X KLEE
VeriFuzz

Motivation

- Automated test generation is prospering
- But:
  - Different strengths and weaknesses
  - Proprietary interfaces
  - No cooperation
  - Lock-in

$\Rightarrow$ Missed potential

ESBMC AFL
KLUZZER
Gramfuzz Redqueen zzuff
QSYM AFL-fast NEUZZ
Symbiotic
AFLGo CoVeriTest
DART CPA/Tiger-MGP
VUzzer
PathCrawler Driller
PRTest libFuzzer FShell
Legion FairFuzz CREST
Tracer-X KLEE
VeriFuzz

ESBMC
AFL
KLUZZER
Gramfuzz
Redqueen
zzuf
NEUZZ
QSYM
AFL-fast
AFLGo
DART
Driller
PRTest
FShell
Legion
FairFuzz
CREST
Tracer-X
KLEE
VeriFuzz

Example

```
int i = input();

if (i != 1017) {
  while (i > 1017) {
    // branch 1.1
    i--;
  }
  // branch 1.2
  // ...
} else {
  // branch 2
  // ...
}
```

Example

```
int i = input();

if (i != 1017) {
  while (i > 1017) {
    // branch 1.1
    i--;
  }
  // branch 1.2
  // ...
} else {
  // branch 2
  // ...
}
```

- Random testing: may not find i = 1017
- Symbolic execution: may hang in while-loop

Example

```
int i = input();

if (i != 1017) {
  while (i > 1017) {
    // branch 1.1
    i--;
  }
  // branch 1.2
  // ...
} else {
  // branch 2
  // ...
}
```

Random Testing

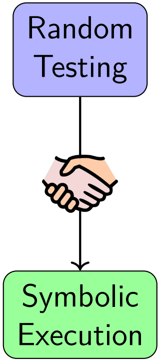
Example

```
int i = input();

if (i != 1017) {





} else {
  // branch 2
  // ...
}
```

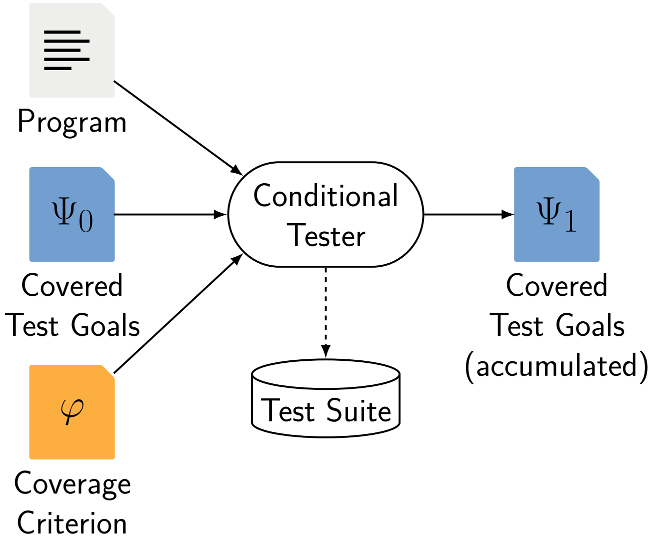
Symbolic Execution

Example

```
int i = input();

if (i != 1017) {
  while (i > 1017) {
    // branch 1.1
    i--;
  }
  // branch 1.2
  // ...
} else {
  // branch 2
  // ...
}
```

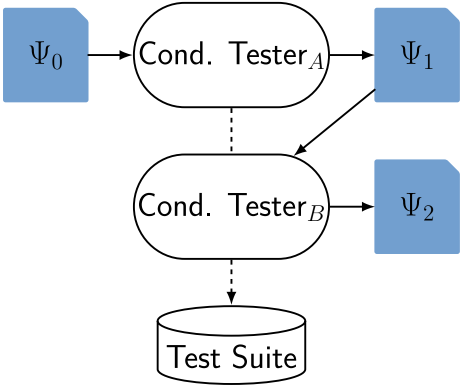


Conditional Tester



Combinations

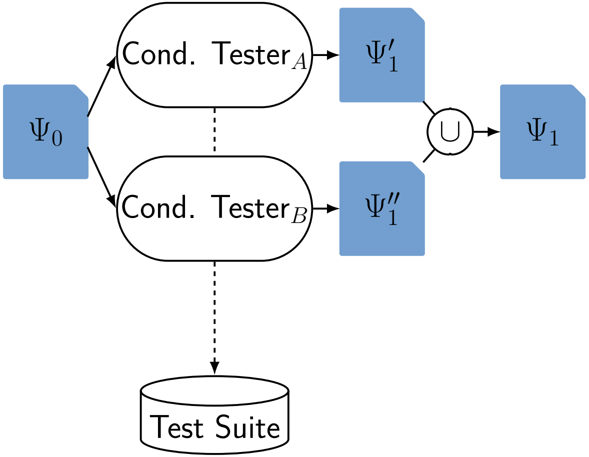
- Sequential



Omitting input program and coverage criterion for simplicity

Combinations

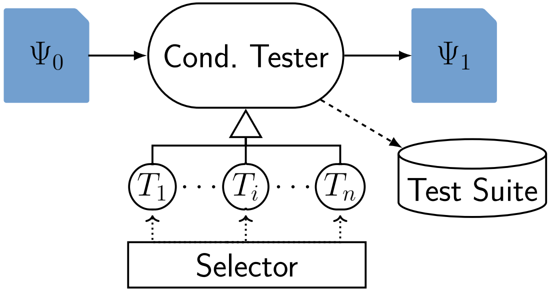
- Sequential
- Parallel



Omitting input program and coverage criterion for simplicity

Combinations

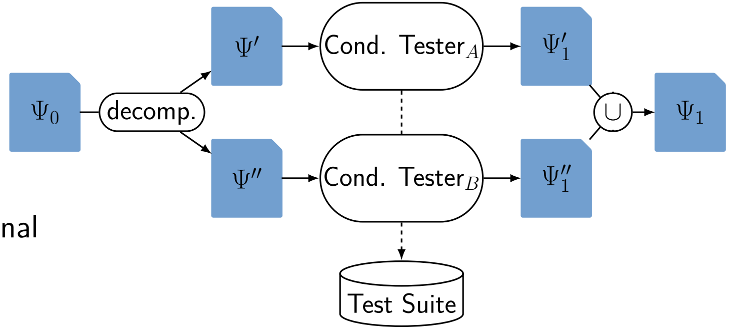
- Sequential
- Parallel
- Strategy selection



Omitting input program and coverage criterion for simplicity

Combinations

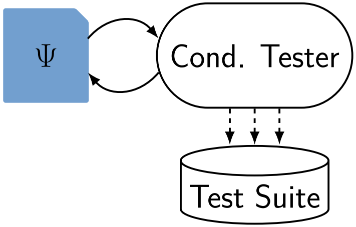
- Sequential
- Parallel
- Strategy selection
- Compositional



Omitting input program and coverage criterion for simplicity

Combinations

- Sequential
- Parallel
- Strategy selection
- Compositional
- Cyclic



Omitting input program and coverage criterion for simplicity

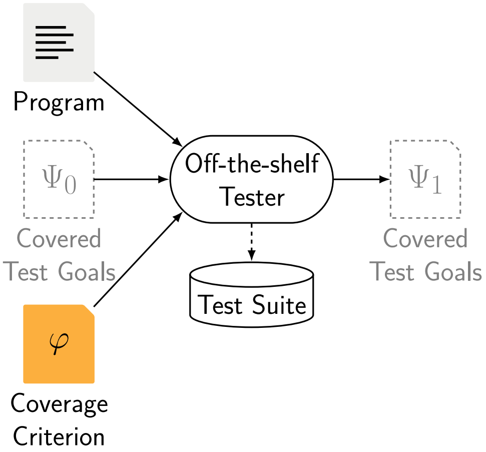
Combinations

- Sequential
- Parallel
- Strategy selection
- Compositional
- Cyclic
- . . .

Omitting input program and coverage criterion for simplicity

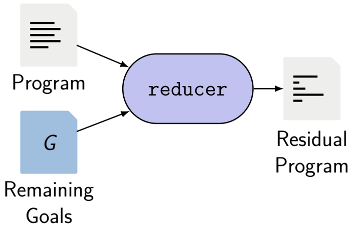
Off-the-shelf Tester

- Current testers do not support conditions



Reducer

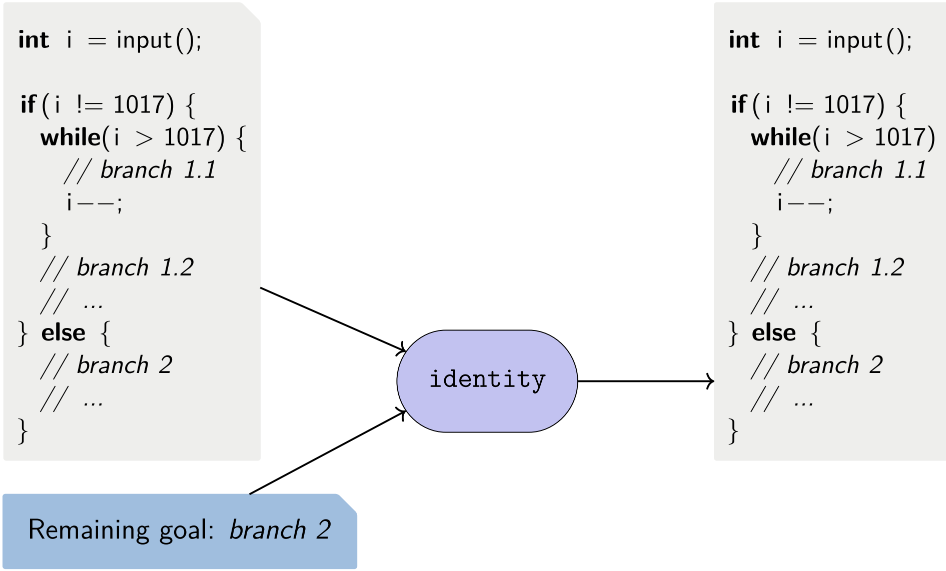
- Input: Program P, remaining test goals *G*
- Output: Residual program P'
- Required property: P' *reachability-equivalent* to P with regard to *G*



Reachability equivalence

Each program input that reaches a test goal of *G* in P' reaches the same test goal in P

Reducer Example: Identity



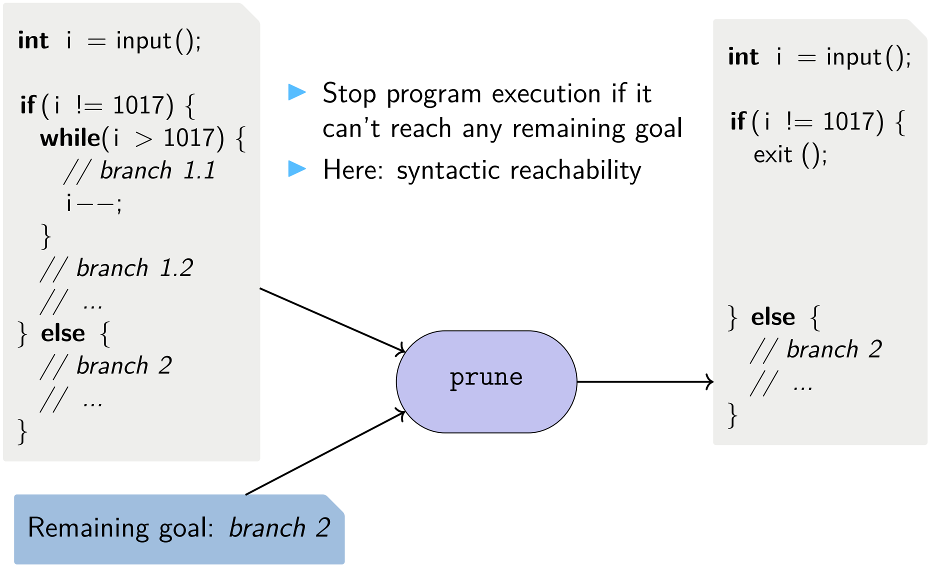
```
int i = input();

if (i != 1017) {
  while(i > 1017) {
    // branch 1.1
    i--;
  }
  // branch 1.2
  // ...
} else {
  // branch 2
  // ...
}
```

```
int i = input();

if (i != 1017) {
  while(i > 1017)
    // branch 1.1
    i--;
  }
  // branch 1.2
  // ...
} else {
  // branch 2
  // ...
}
```

Remaining goal: *branch 2*

Reducer Example: Pruning

- Stop program execution if it can't reach any remaining goal
- Here: syntactic reachability



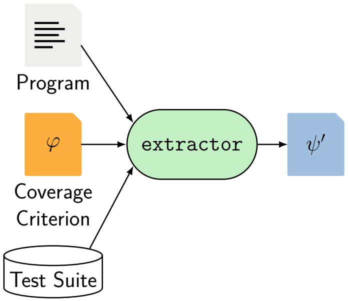
```
int i = input();

if (i != 1017) {
  while(i > 1017) {
    // branch 1.1
    i--;
  }
  // branch 1.2
  // ...
} else {
  // branch 2
  // ...
}
```

Remaining goal: *branch 2*

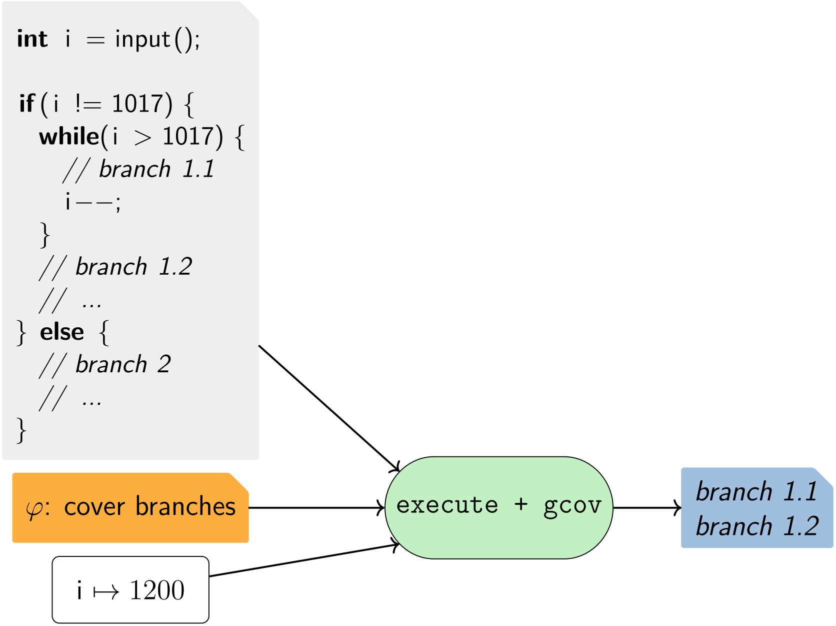
```
int i = input();

if (i != 1017) {
  exit();



} else {
  // branch 2
  // ...
}
```

Test-goal Extractor

- Input: Program P,
  coverage criterion φ, test suite S
- Output: Test goals Ψ covered by S

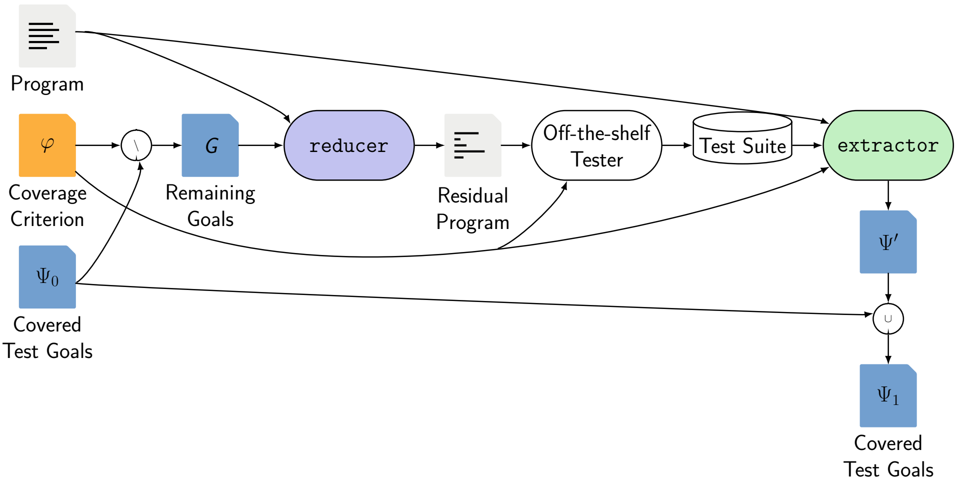


Extractor Example: Test Executor



```
int i = input();

if(i != 1017) {
  while(i > 1017) {
    // branch 1.1
    i--;
  }
  // branch 1.2
  // ...
} else {
  // branch 2
  // ...
}
```

Off-the-shelf → Conditional



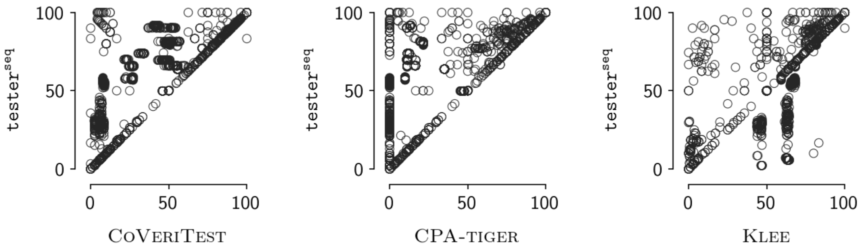
Implementation

- CondTest
  https://gitlab.com/sosy-lab/software/conditional-testing

1. Test-Comp tester $\rightarrow$ conditional tester
2. SV-COMP verifier $\rightarrow$ conditional tester
3. Sequential combination
4. Cyclic combination

- Plug-and-play through existing tool descriptions

Evaluation (I)



- Branch coverage of created test suites (%), per task
- Tool standalone, 900 s (x-axis)
- `tester`$^{\texttt{seq}}$: CPA-tiger + CoVeriTest + Klee, 300 s each (y-axis)

Evaluation (II)

- CPA-TIGER + COVERITEST + KLEE, 300 s each
- `id`: no info exchange `prune`: info exchange

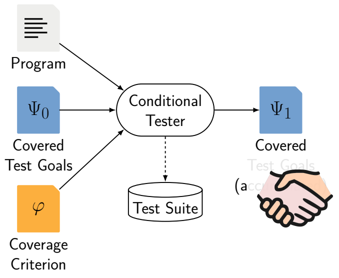
task	branch coverage (%)		
	`id`	`prune`	increase
mod3.c.v+sep-reducer	75.0	80.0	+ 5.00
Problem07_label35	52.0	54.0	+ 2.00
Problem07_label37	54.2	56.2	+ 1.97
Problem04_label35	79.5	81.3	+ 1.79
Problem06_label02	57.0	58.7	+ 1.70
Problem06_label27	57.5	58.6	+ 1.09
Problem04_label02	80.2	81.3	+ 1.06
Problem06_label18	57.5	58.6	+ 1.05
Problem04_label16	79.1	80.1	+ 1.01
Problem04_label34	80.2	81.2	+ 0.99

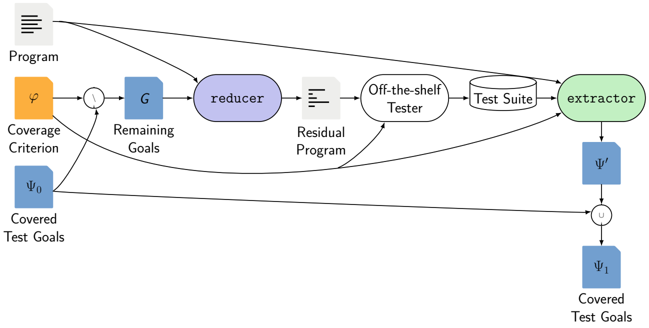
Evaluation (III)

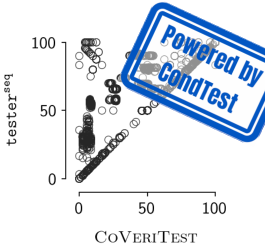
- `prune`: CPA-tiger + CoVeriTest + Klee, 300 s each
- `vb`: CPA-tiger + CoVeriTest + Klee, 200 s each + Esbmc 300 s

task	branch coverage (%)		
	`prune`	`vb`	increase
Problem08_label30	5.72	62.0	+56.2
Problem08_label32	5.72	61.9	+56.1
Problem08_label06	5.72	61.8	+56.1
Problem08_label35	5.72	61.7	+56.0
Problem08_label00	5.72	61.6	+55.9
Problem08_label11	5.72	61.5	+55.8
Problem08_label19	5.72	61.5	+55.7
Problem08_label29	5.67	61.4	+55.7
Problem08_label22	5.72	61.5	+55.7
Problem08_label56	5.72	61.5	+55.7

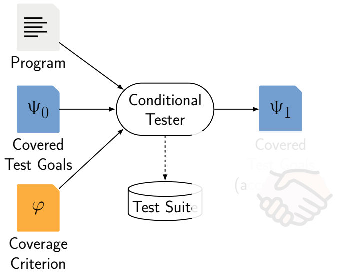
Contributions



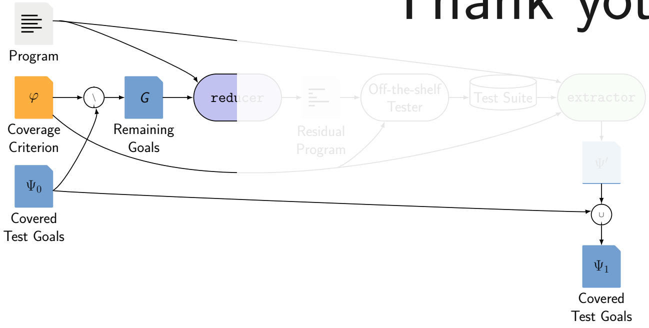


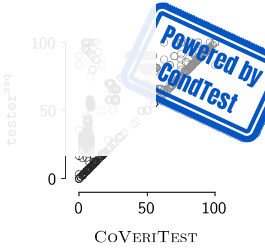


Contributions



Thank you!





References

[1] D. Beyer, T. A. Henzinger, M. E. Keremoglu, and P. Wendler.
Conditional model checking: A technique to pass information between verifiers.
In *Proc. FSE*. ACM, 2012.

[2] M. Harman, L. Hu, R. M. Hierons, J. Wegener, H. Sthamer, A. Baresel, and M. Roper.
Testability transformation.
IEEE Trans. Software Eng., 30(1):3–16, 2004.