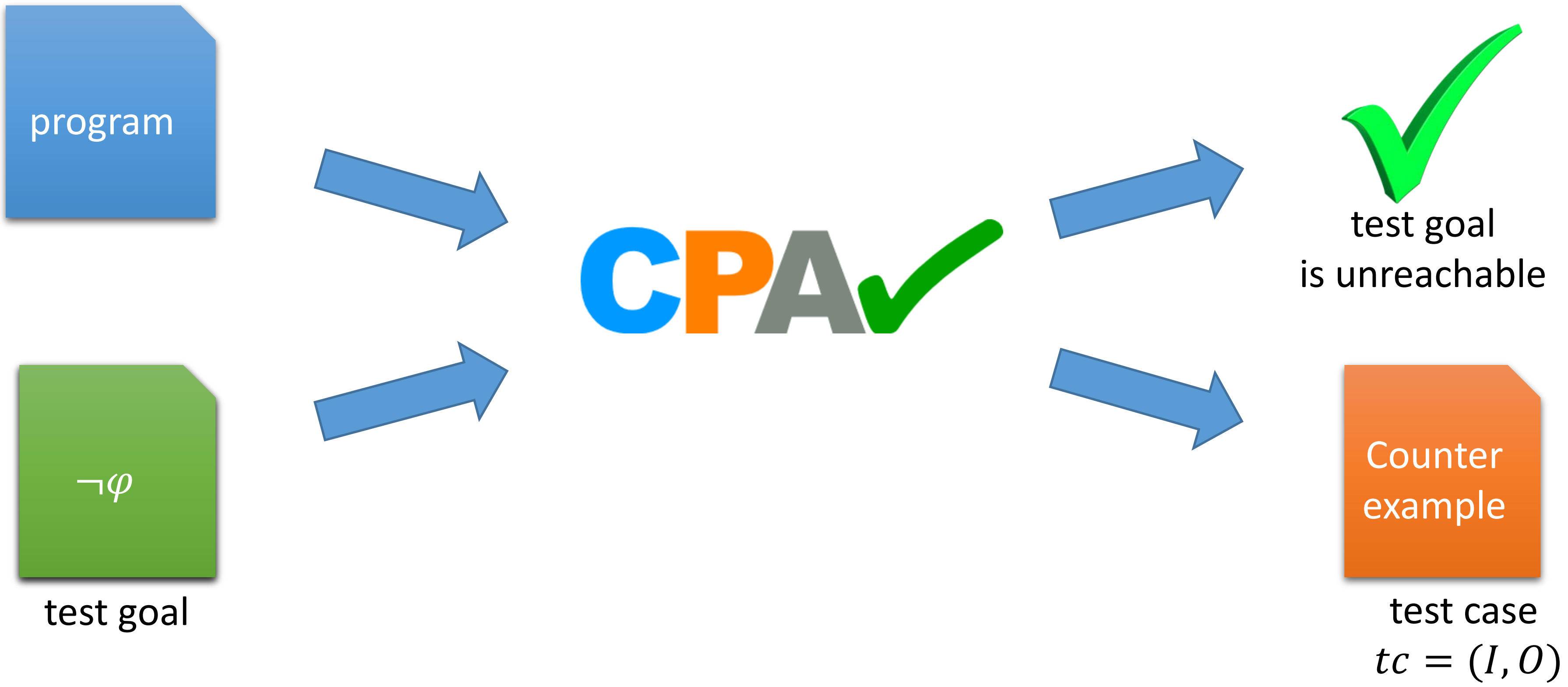
Configurable Test-Goal Set Partitioning for Directed Multi-Goal Test Generation

Malte Lochau (TU Darmstadt)

Joint work with:

Sven Apel, Johannes Bürdek, Dirk Beyer, Andreas Holzer, Andreas Stahlbauer

Test-Case Generation with Software Model Checking



program

$\neg\varphi$

test goal

CPA✓

test goal is unreachable

Counter example

test case

$tc = (I, O)$

Tiger Algorithm for Multi-Goal Test Coverage

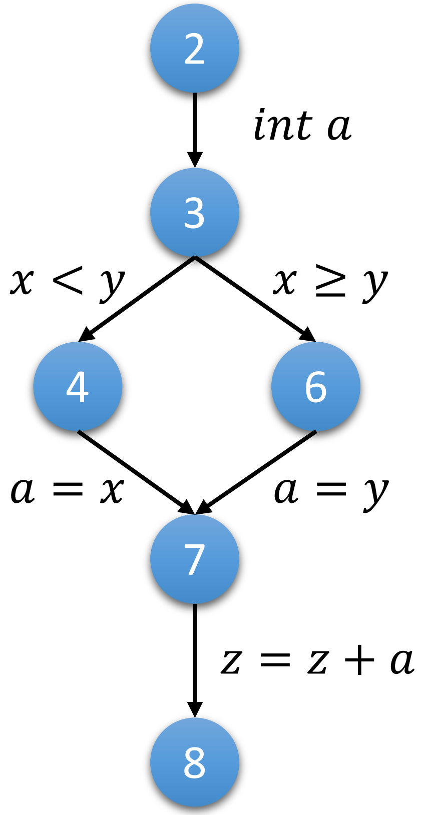
$\boldsymbol{Input}$: $CFA\ cfa, Set\ of\ Goals\ G$

$\boldsymbol{Output}$: $Test\ Suite\ TS\ covering\ G$

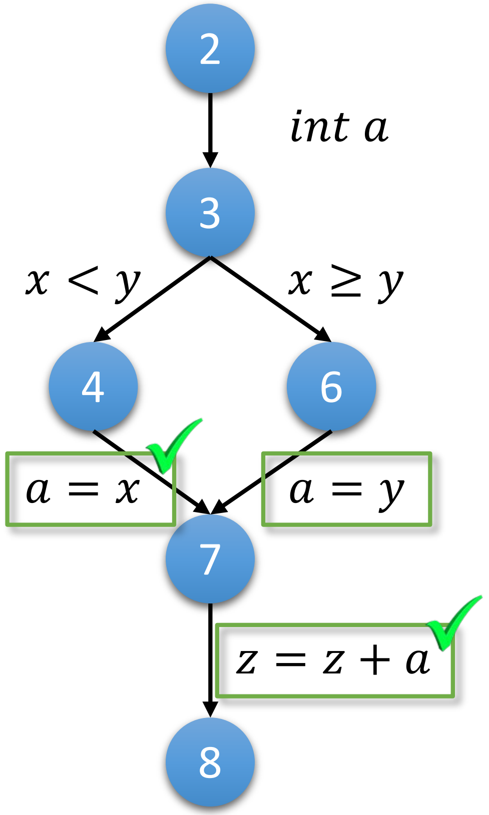
```
$TS := \{\}$
$G' \coloneqq G$
$while\ G' \neq \emptyset$
    $g \coloneqq pick\ and\ remove\ element\ from\ G'$
    if $\left(tc \coloneqq reachabilityAnalysis(cfa, g)\right)$
        $TS \coloneqq TS \cup (tc, g)$
        $foreach\ g' \in G'$
            $if\ tc\ covers\ g'$
                $TS \coloneqq TS \cup (tc, g')$
                $G` \coloneqq G' \backslash \mathrm{g}'$
```

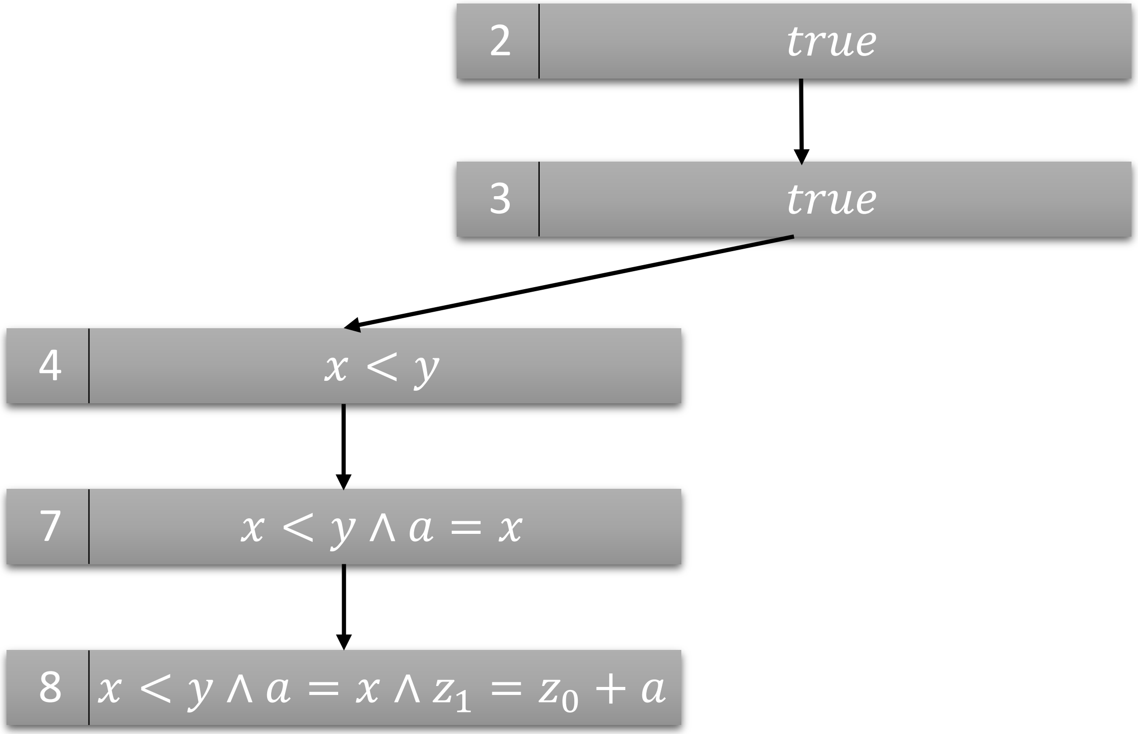
Example

```
1   int func-spl (int x, int y, int z) {
2      int a;
3      if (x < y)
4           a = x;
5      else
6           a = y;
7      z = z + a;
8      return z;
9   }
```



Example





2 $true$

3 $true$

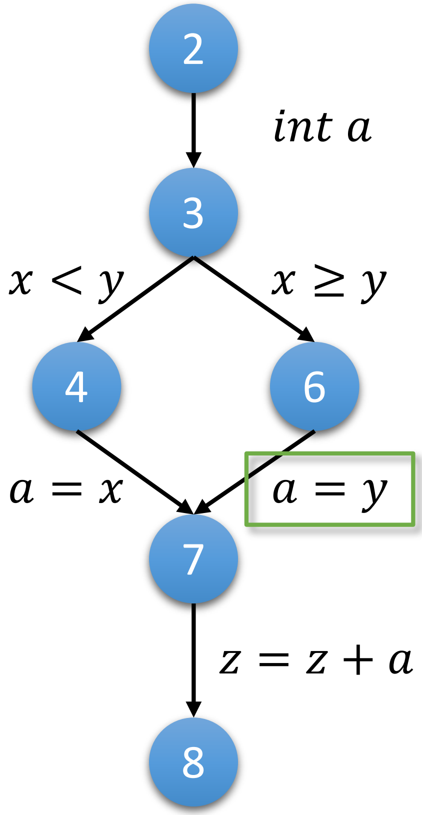
4 $x < y$

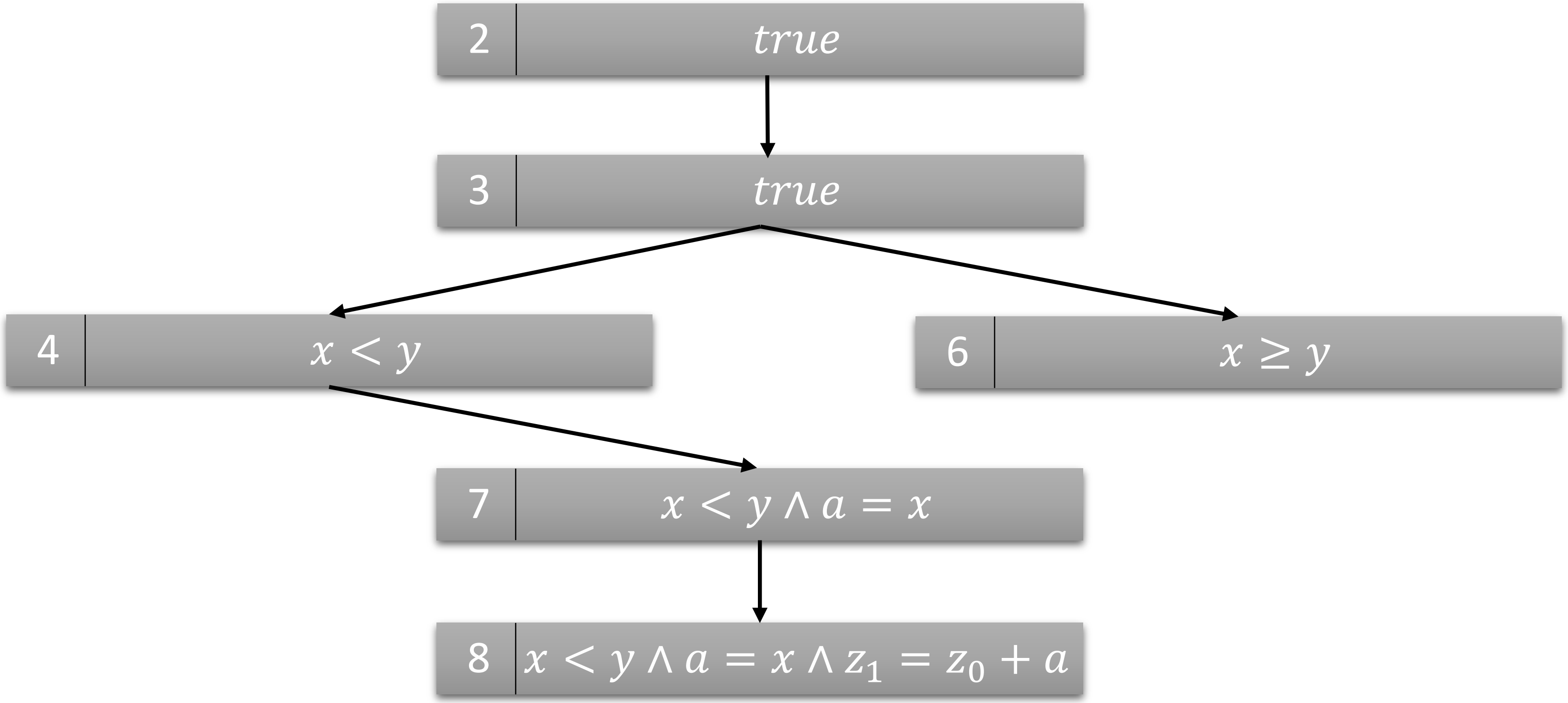
7 $x < y \wedge a = x$

8 $x < y \wedge a = x \wedge z_1 = z_0 + a$

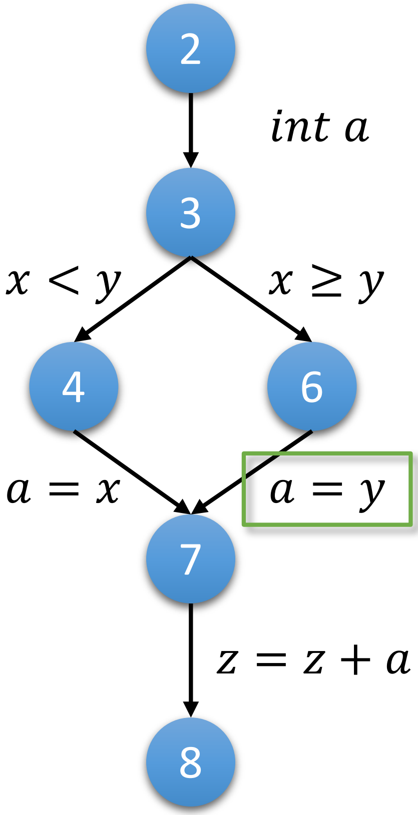
- tc = (x=0, y=1, z=0 | z=0)

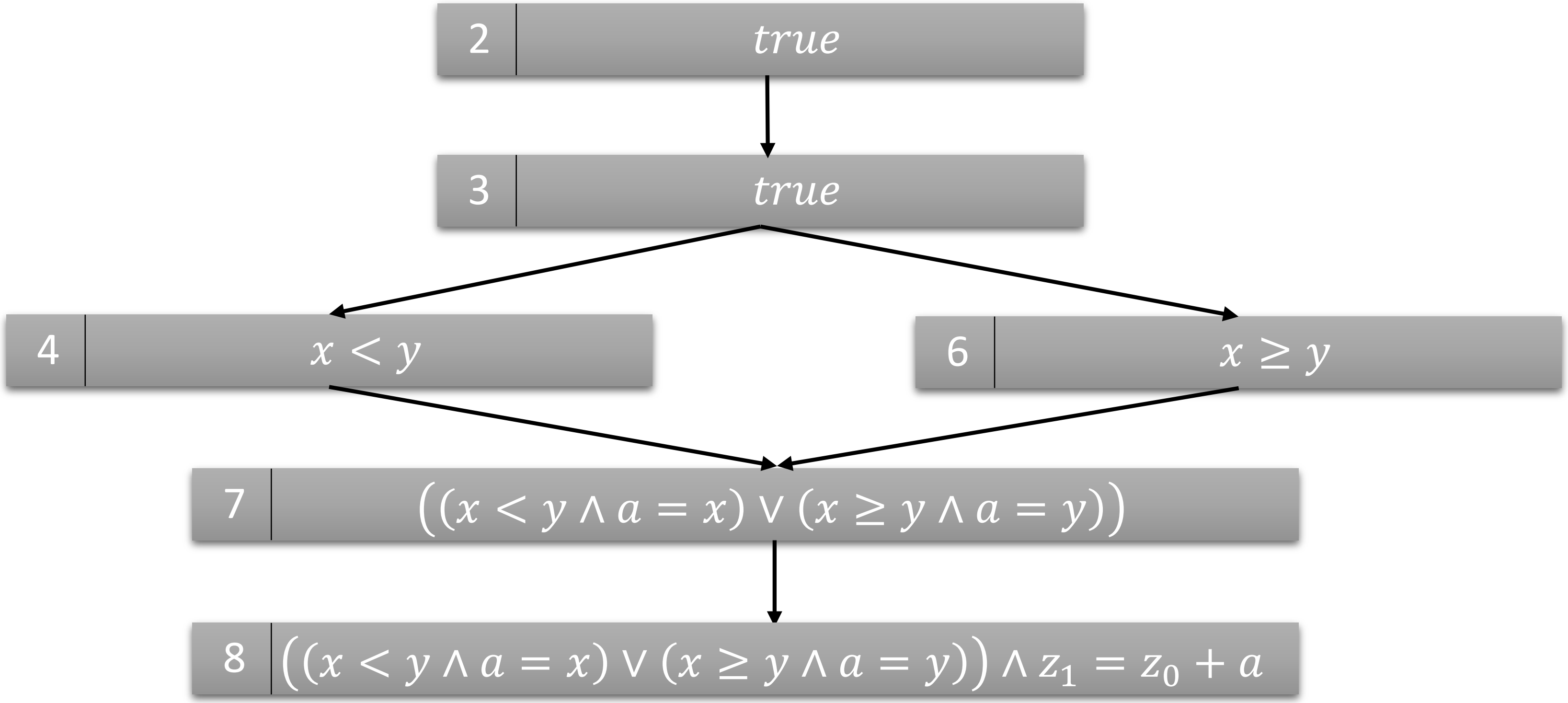
Example


2
$int\ a$
3
$x < y$
$x \geq y$
4
6
$a = x$
$a = y$
7
$z = z + a$
8


2 $true$
3 $true$
4 $x < y$
6 $x \geq y$
7 $x < y \wedge a = x$
8 $x < y \wedge a = x \wedge z_1 = z_0 + a$

Example





2 $true$

3 $true$

4 $x < y$

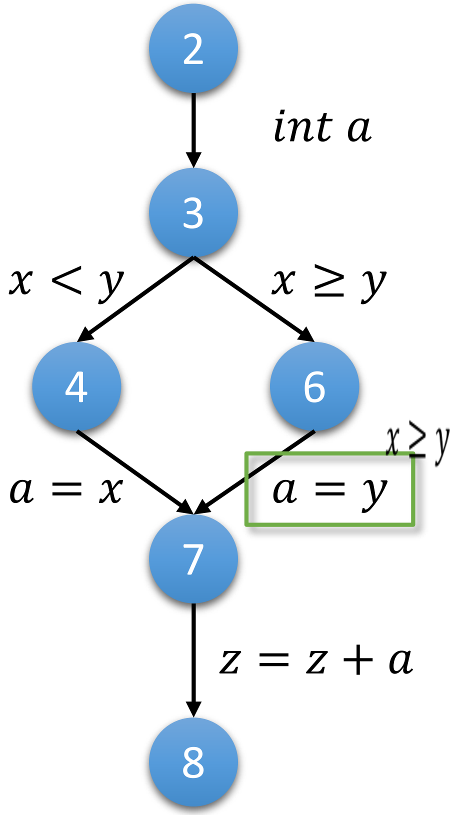
6 $x \geq y$

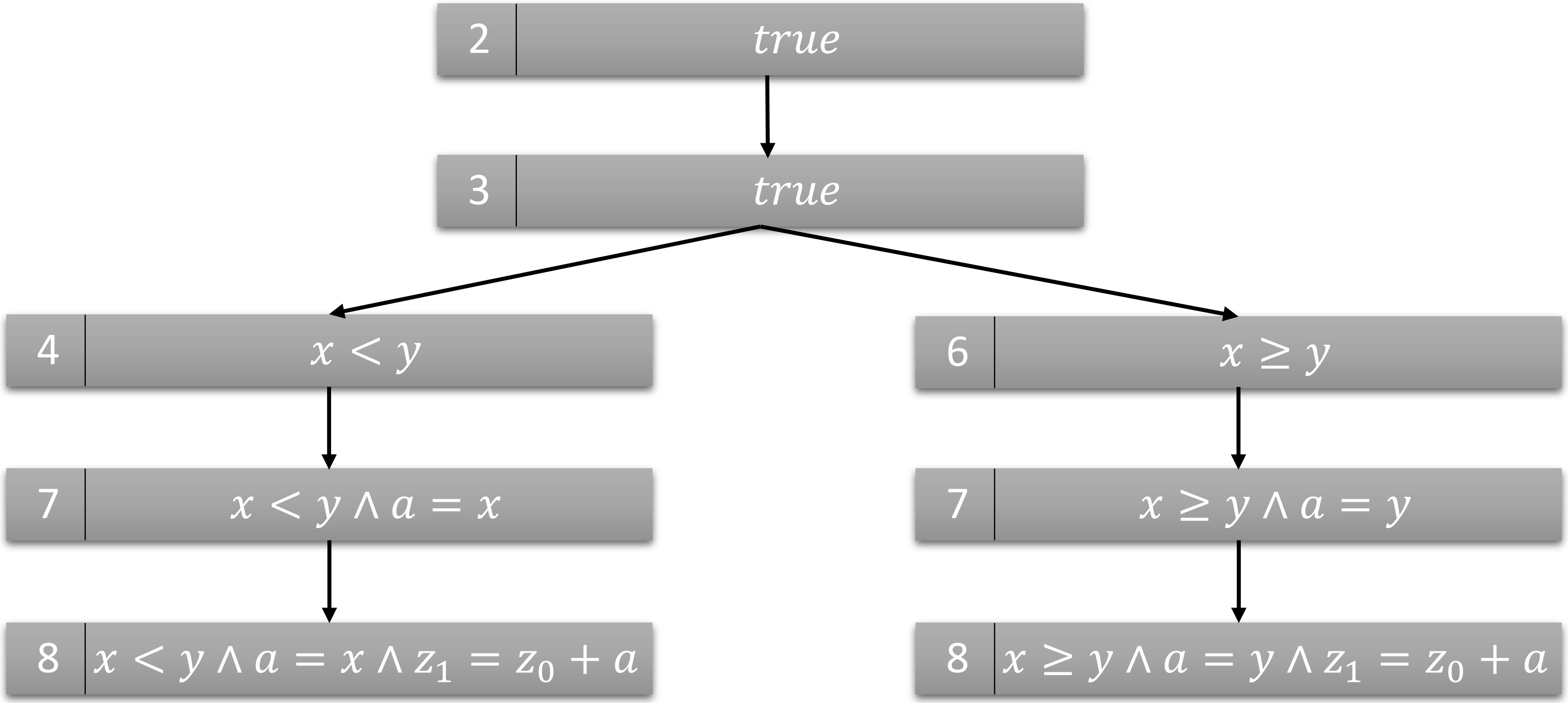
7 $\big((x < y \land a = x) \lor (x \geq y \land a = y)\big)$

8 $\big((x < y \land a = x) \lor (x \geq y \land a = y)\big) \land z_1 = z_0 + a$

- tc = ?

Example




2 $true$

3 $true$

4 $x < y$

7 $x < y \wedge a = x$

8 $x < y \wedge a = x \wedge z_1 = z_0 + a$

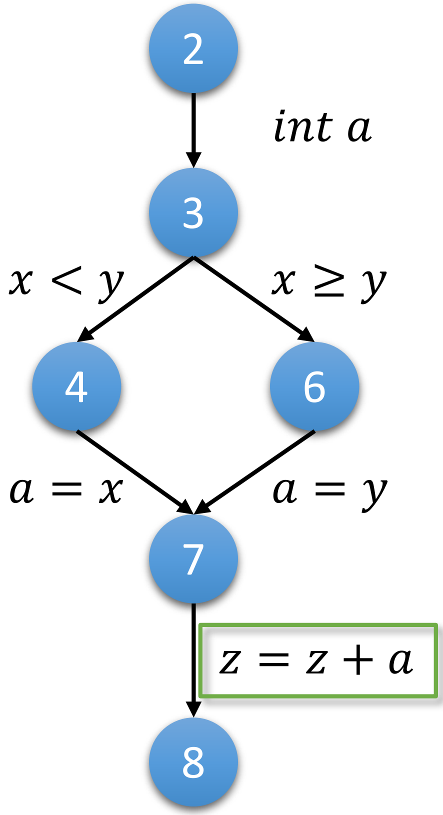
6 $x \geq y$

7 $x \geq y \wedge a = y$

8 $x \geq y \wedge a = y \wedge z_1 = z_0 + a$

- tc = (x=0, y=1, z=0 | z=0)
- tc = (x=0, y=0, z=0 | z=0)

Example



2

$int\ a$

3

$x < y$

$x \geq y$

4

6

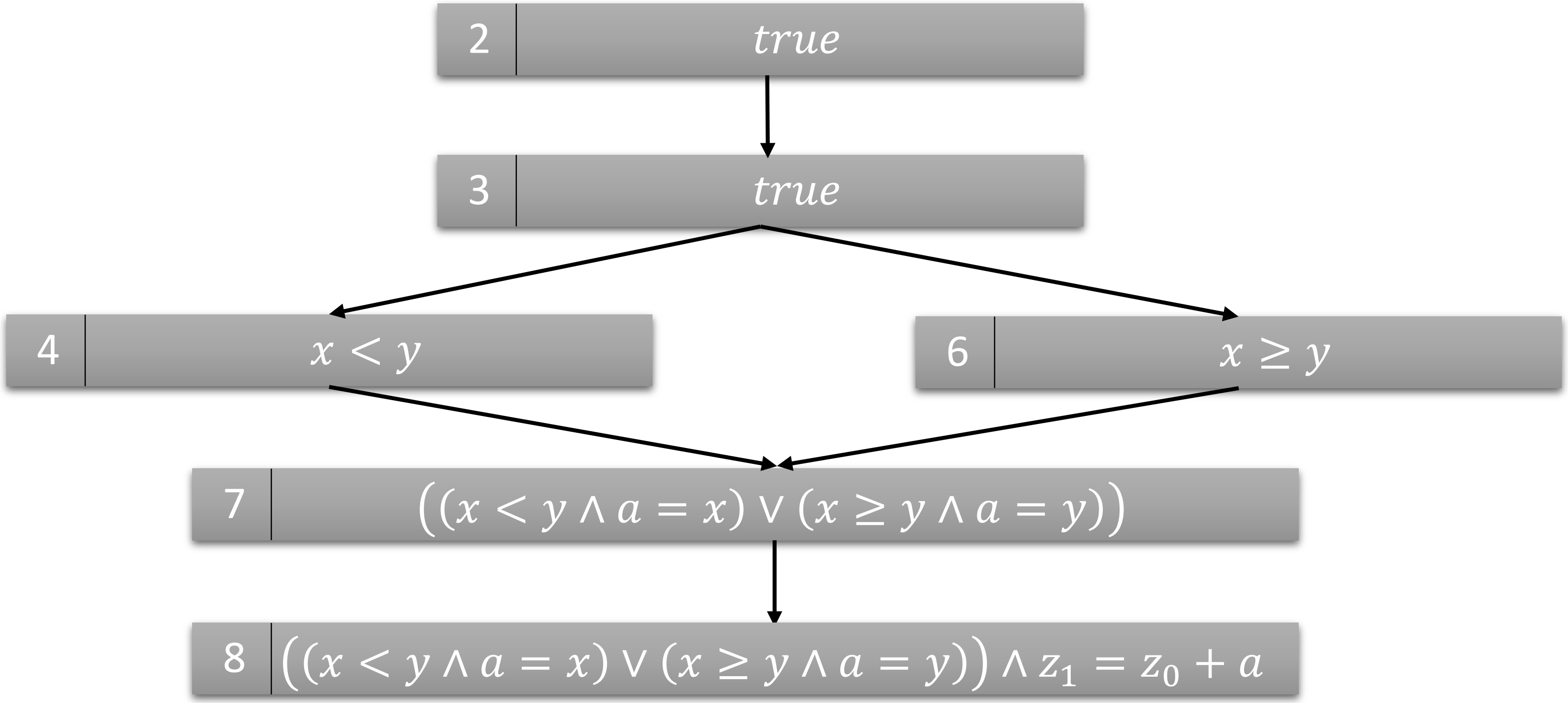
$a = x$

$a = y$

7

$z = z + a$

8



2 $true$

3 $true$

4 $x < y$

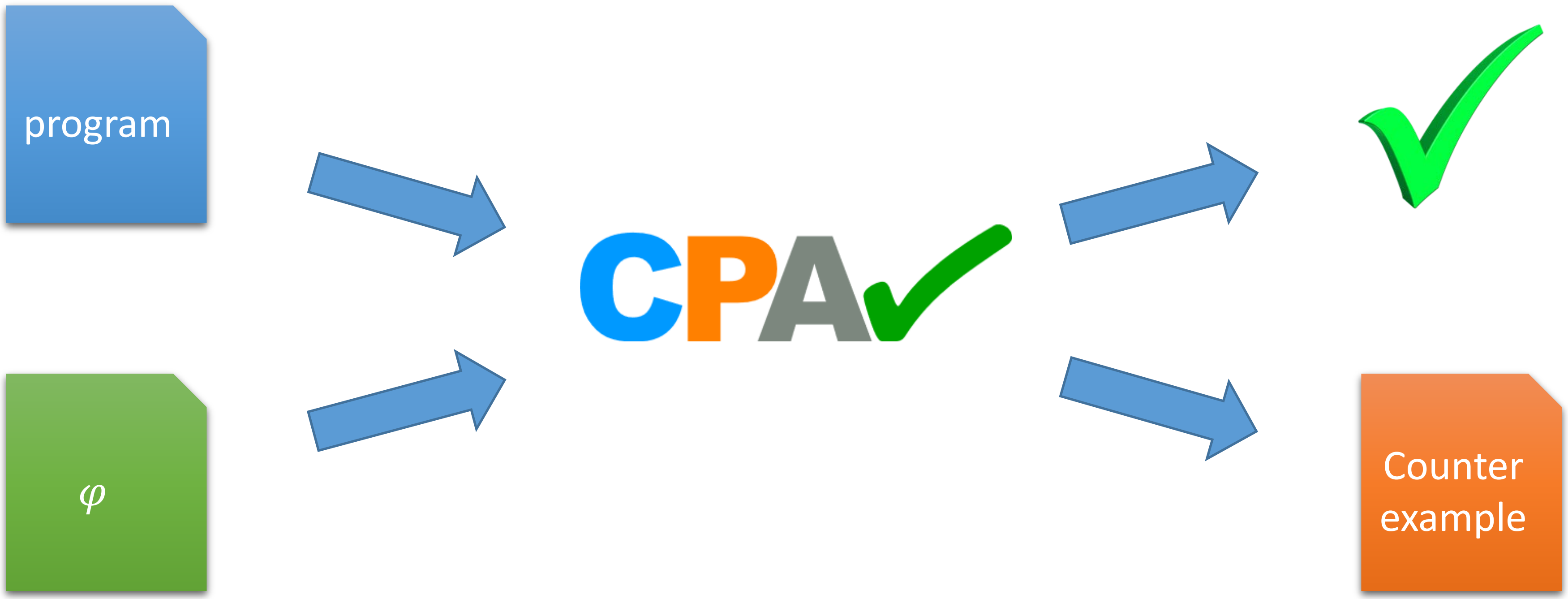
6 $x \geq y$

7 $\big((x < y \land a = x) \lor (x \geq y \land a = y)\big)$

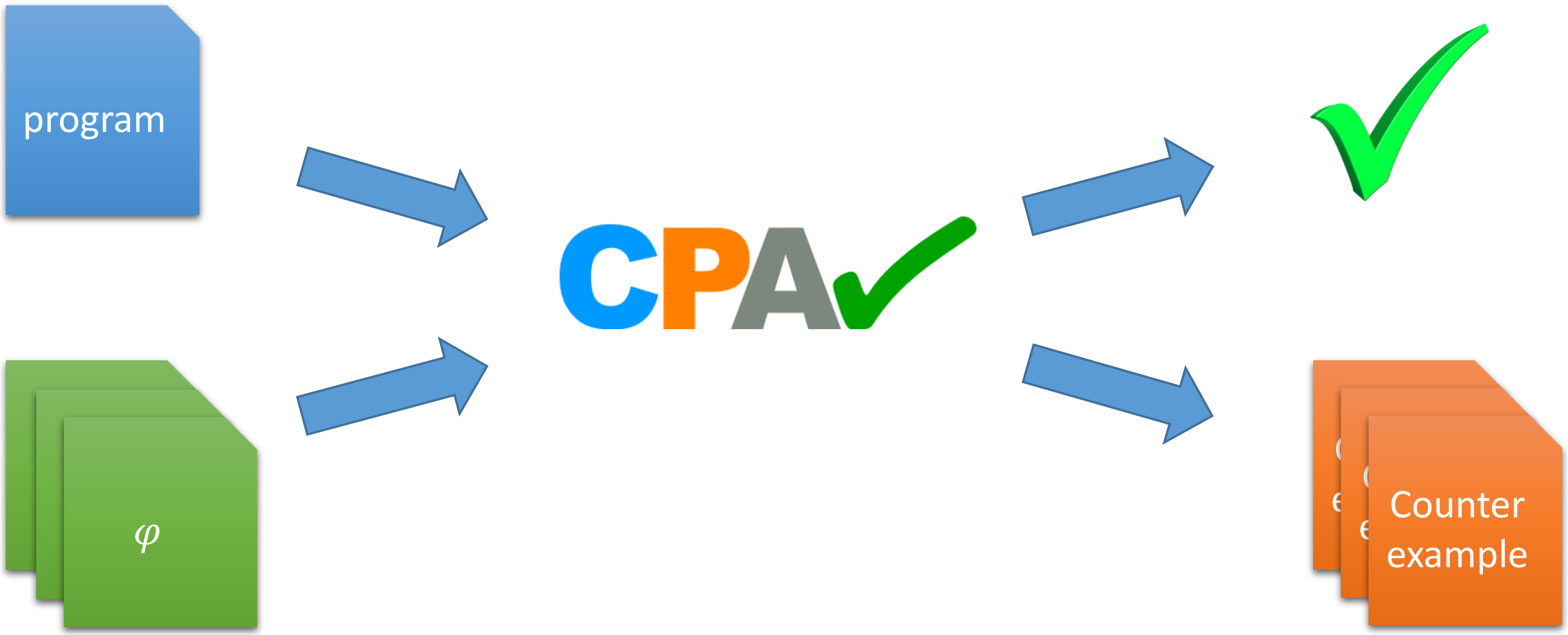
8 $\big((x < y \land a = x) \lor (x \geq y \land a = y)\big) \land z_1 = z_0 + a$

- ARG-state merge only feasible before reaching the test goal
- ARG is (re-)constructed for every test-goal

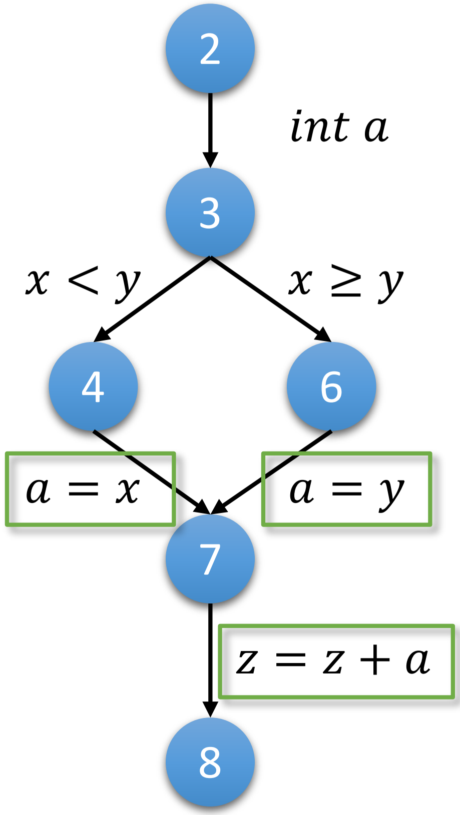
Model Checking for Test-Case Generation

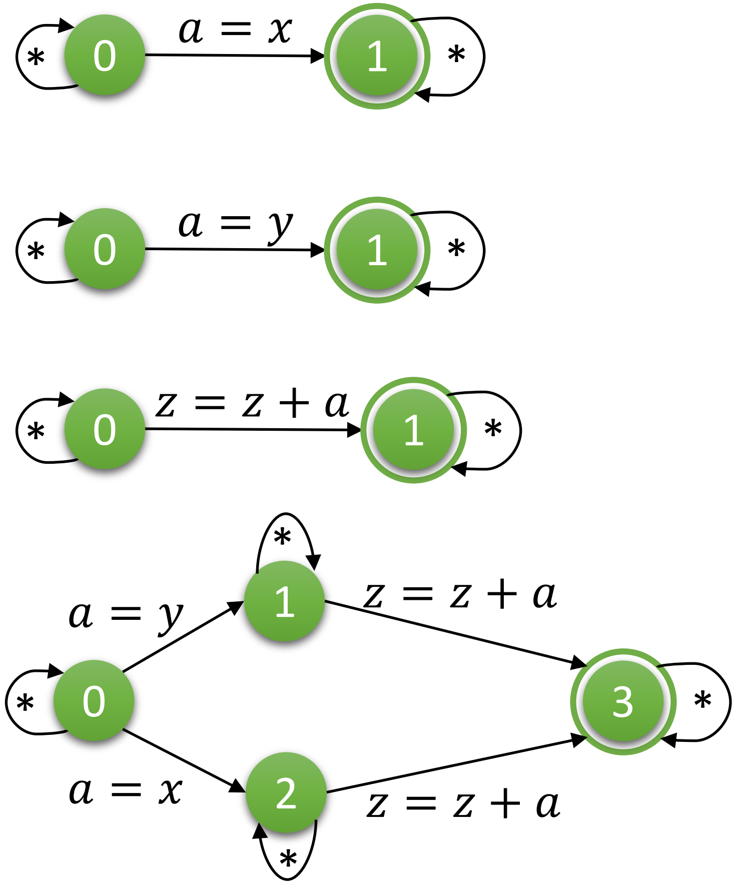


Multi-Property Checking for Multi-Goal Test-Suite Generation



Example: Multi-Goal Test-Generation





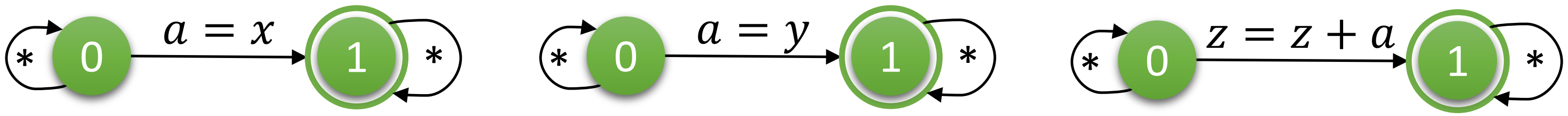
Test-Goal Automata

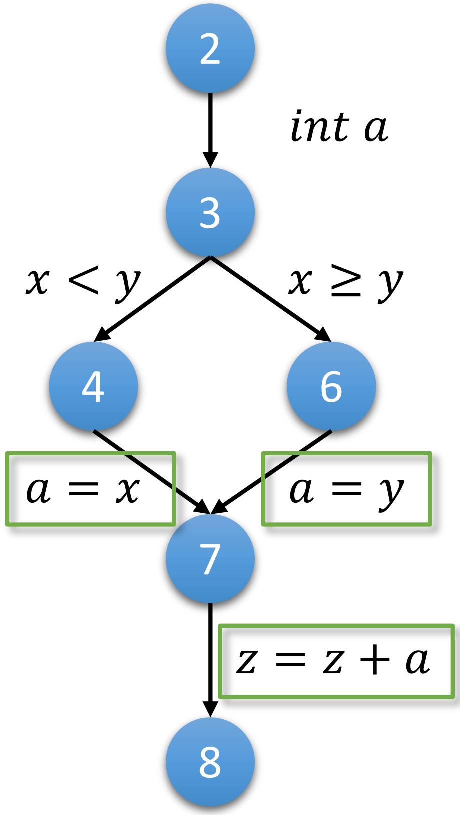
- represent sets of CFA paths
- non-deterministic
- parallel automata semantics: multiple simultanously active states

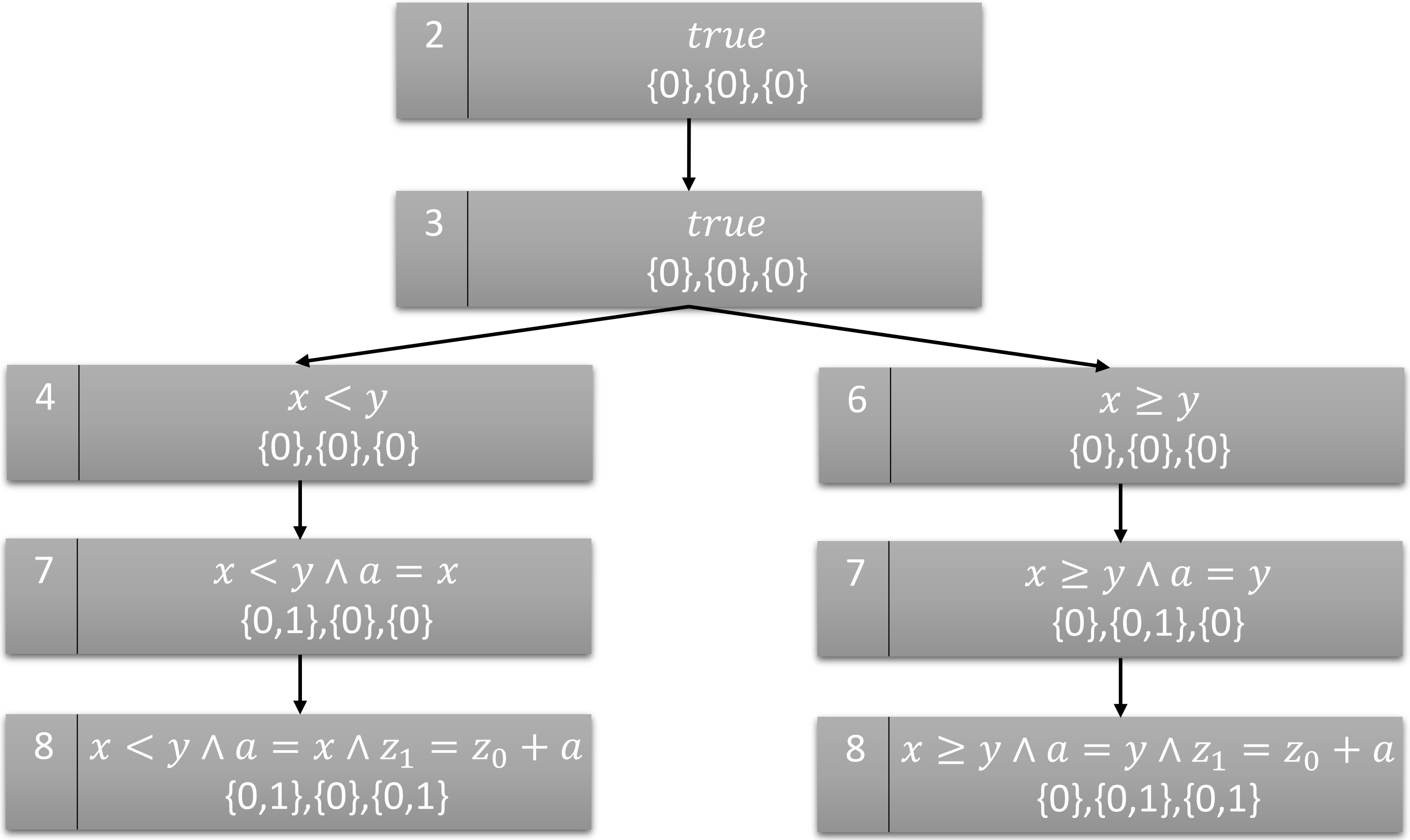
Test-Goal Set Partitioning

- Process subsets of test-goal automata simultaneously
- Partitioning criteria?

Example: Multi-Goal Test-Generation







2 $true$ {0},{0},{0}

3 $true$ {0},{0},{0}

4 $x < y$ {0},{0},{0}

6 $x \geq y$ {0},{0},{0}

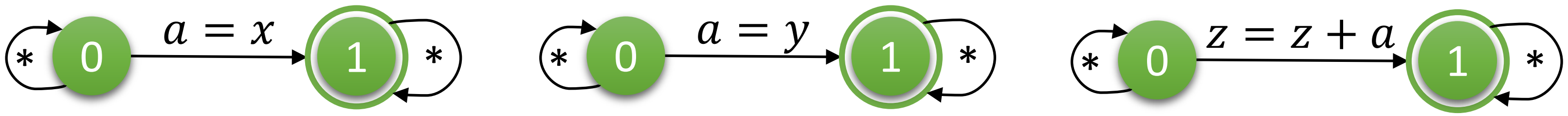
7 $x < y \wedge a = x$ {0,1},{0},{0}

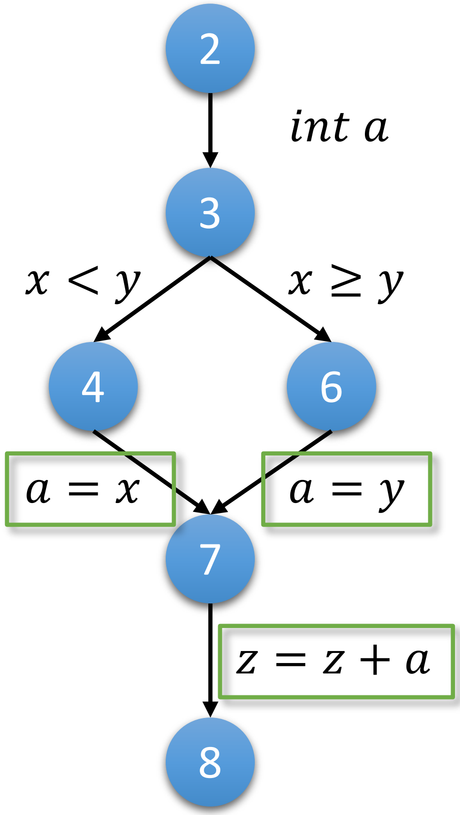
7 $x \geq y \wedge a = y$ {0},{0,1},{0}

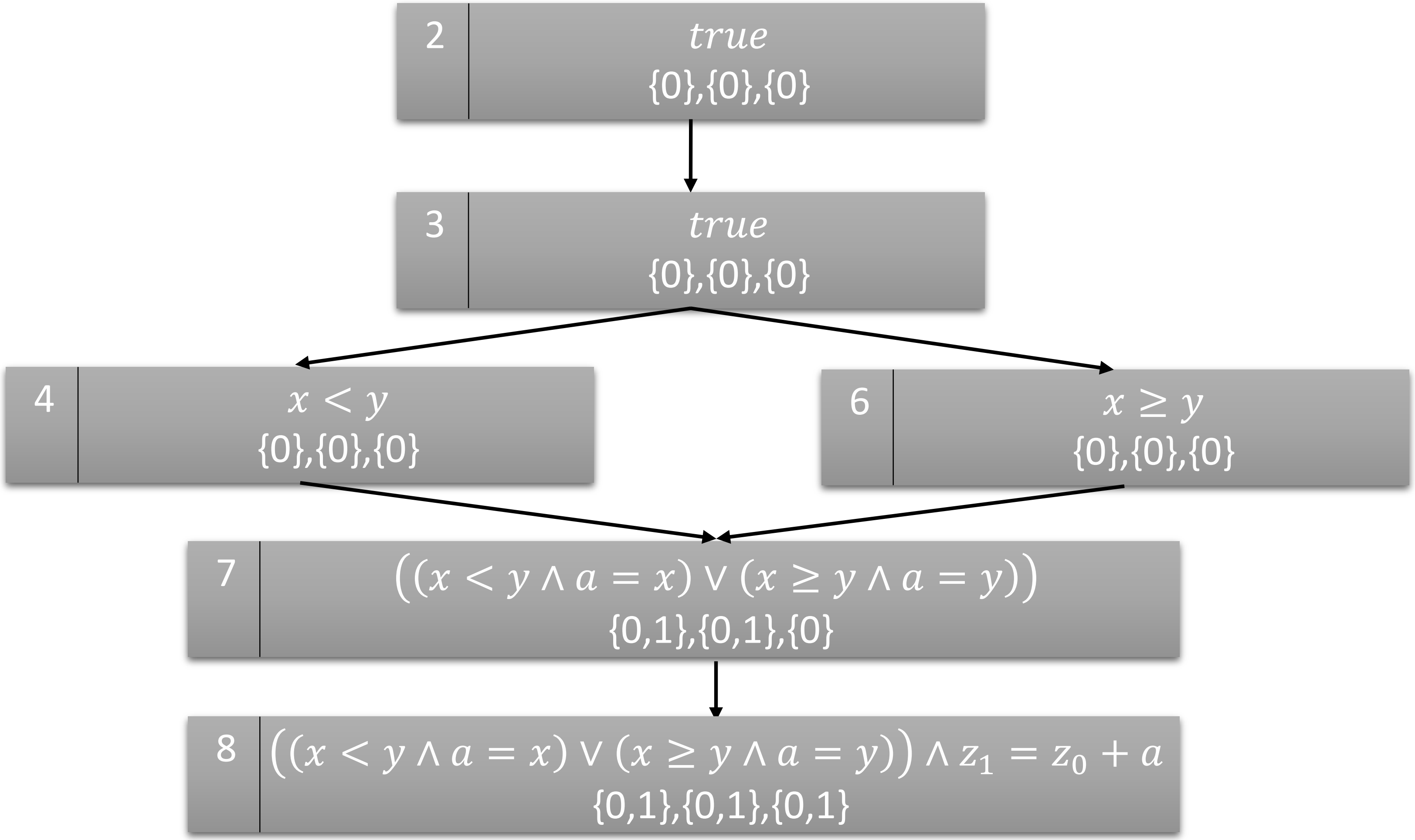
8 $x < y \wedge a = x \wedge z_1 = z_0 + a$ {0,1},{0},{0,1}

8 $x \geq y \wedge a = y \wedge z_1 = z_0 + a$ {0},{0,1},{0,1}

ARG-State Merge Fails!







2 $true$
{0},{0},{0}

3 $true$
{0},{0},{0}

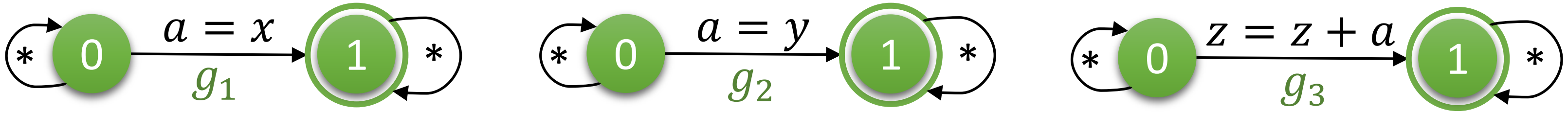
4 $x < y$
{0},{0},{0}

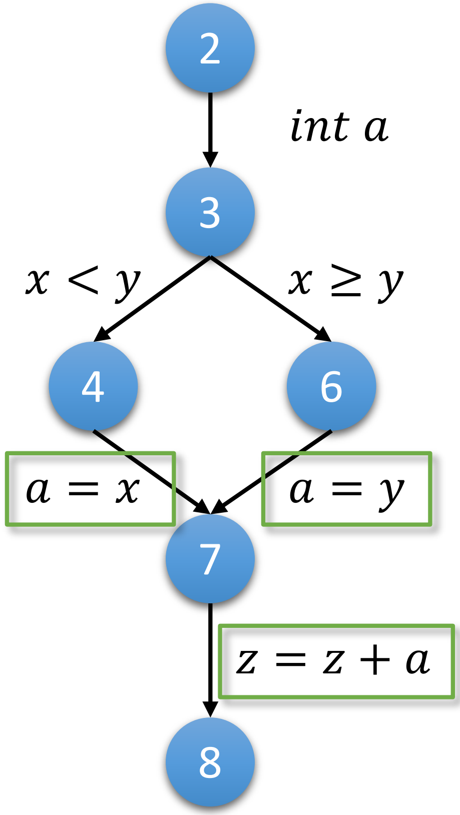
6 $x \geq y$
{0},{0},{0}

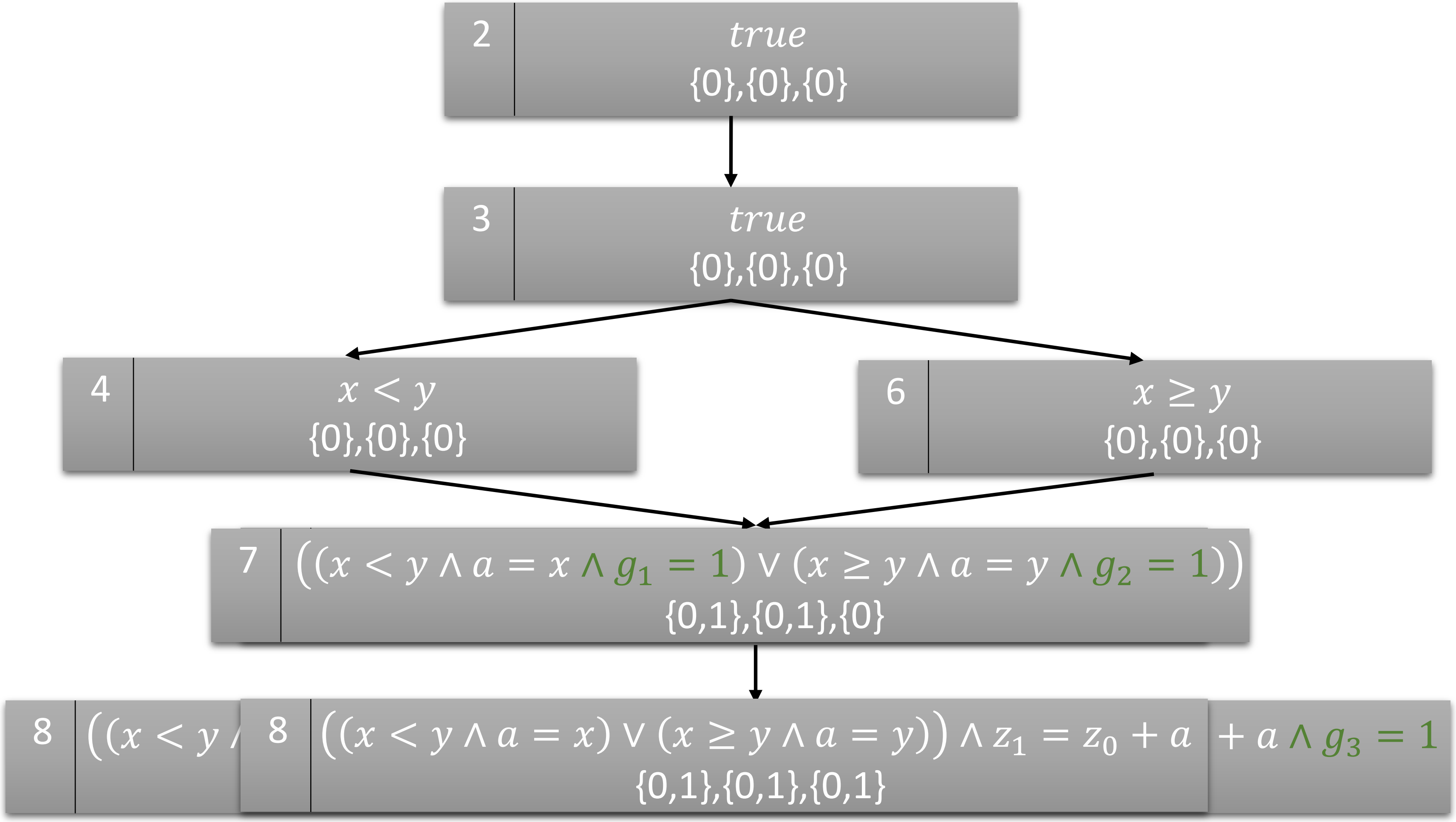
7 $\big((x < y \land a = x) \lor (x \geq y \land a = y)\big)$
{0,1},{0,1},{0}

8 $\big((x < y \land a = x) \lor (x \geq y \land a = y)\big) \land z_1 = z_0 + a$
{0,1},{0,1},{0,1}

Solution: Merge with (Test-Goal) Weaving







2 $true$
{0},{0},{0}

3 $true$
{0},{0},{0}

4 $x < y$
{0},{0},{0}

6 $x \geq y$
{0},{0},{0}

7 $\left((x < y \land a = x \land g_1 = 1) \lor (x \geq y \land a = y \land g_2 = 1)\right)$
{0,1},{0,1},{0}

8 $((x < y \land$ … $+ a \land g_3 = 1$

8 $\left((x < y \land a = x) \lor (x \geq y \land a = y)\right) \land z_1 = z_0 + a$
{0,1},{0,1},{0,1}

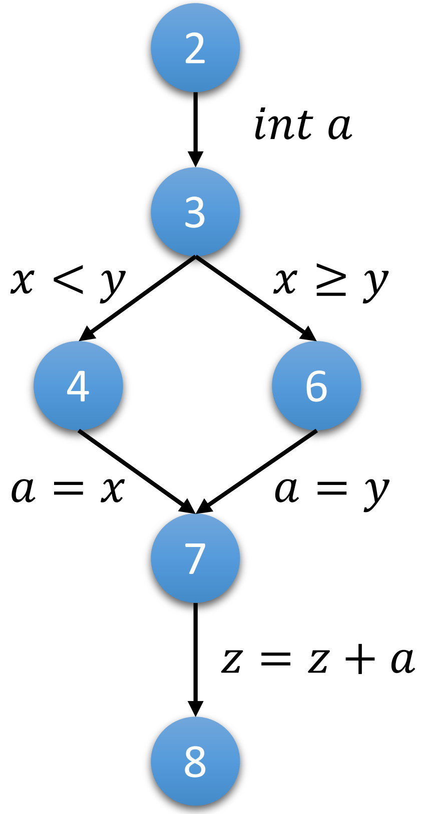
CEGAR vs. TGAR

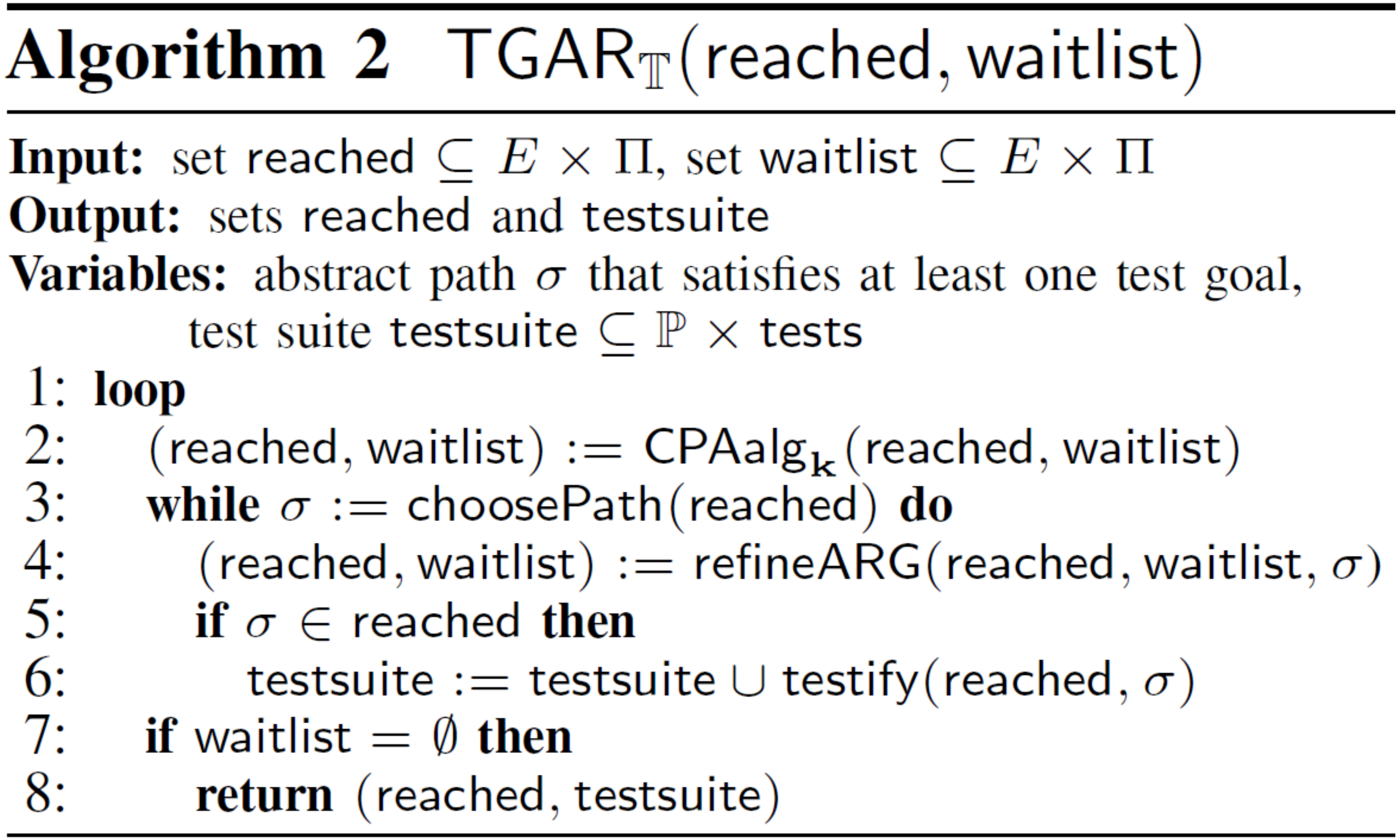
- CEGAR
  - Refinement after each covered goal

  => basic-block coverage and beyond on larger programs!?

- TGAR
  - Refinement after k covered goals

  => reduce (redundant) refinement steps





Algorithm 2 $\mathsf{TGAR}_{\mathbb{T}}(\mathsf{reached}, \mathsf{waitlist})$

Input: set $\mathsf{reached} \subseteq E \times \Pi$, set $\mathsf{waitlist} \subseteq E \times \Pi$

Output: sets $\mathsf{reached}$ and $\mathsf{testsuite}$

Variables: abstract path σ that satisfies at least one test goal, test suite $\mathsf{testsuite} \subseteq \mathbb{P} \times \mathsf{tests}$

```
loop
    $(\mathsf{reached}, \mathsf{waitlist}) := \mathsf{CPAalg}_{\mathbf{k}}(\mathsf{reached}, \mathsf{waitlist})$
    while $\sigma := \mathsf{choosePath}(\mathsf{reached})$ do
        $(\mathsf{reached}, \mathsf{waitlist}) := \mathsf{refineARG}(\mathsf{reached}, \mathsf{waitlist}, \sigma)$
        if $\sigma \in \mathsf{reached}$ then
            $\mathsf{testsuite} := \mathsf{testsuite} \cup \mathsf{testify}(\mathsf{reached}, \sigma)$
    if $\mathsf{waitlist} = \emptyset$ then
        return $(\mathsf{reached}, \mathsf{testsuite})$
```

Experiments

- Subject systems
  - 324 programs with 1541 LoC on average

- Coverage criteria
  - Basic-Block Coverage (C0)
  - Condition Coverage (C3)

- Test-goal set partition sizes
  - Fixed: 1 (DS), 10 (DM10), 20 (DM20), 50 (DM50), all (DA)
  - Percental: 25% (DS25%), 50 (DS50%), 75 (DS75%)

Research Questions

- RQ1 (Efficiency): Does multi-goal set partitioning (DM) increase testing efficiency, as compared to single-goal (DS) and all-goal (DA) processing?

- RQ2 (Effectiveness): How does multi-goal set partitioning (DM) influence testing effectiveness, as compared to single-goal (DS) and all-goal (DA) processing?

All-Goals (DA) vs. Single-Goal (DS)

- Avg. CPU time per program: 70.50s (DS) - 77.46s (DA)
- Avg. CPU time per test goal: 1.2s (DS) - 1.09s (DA)

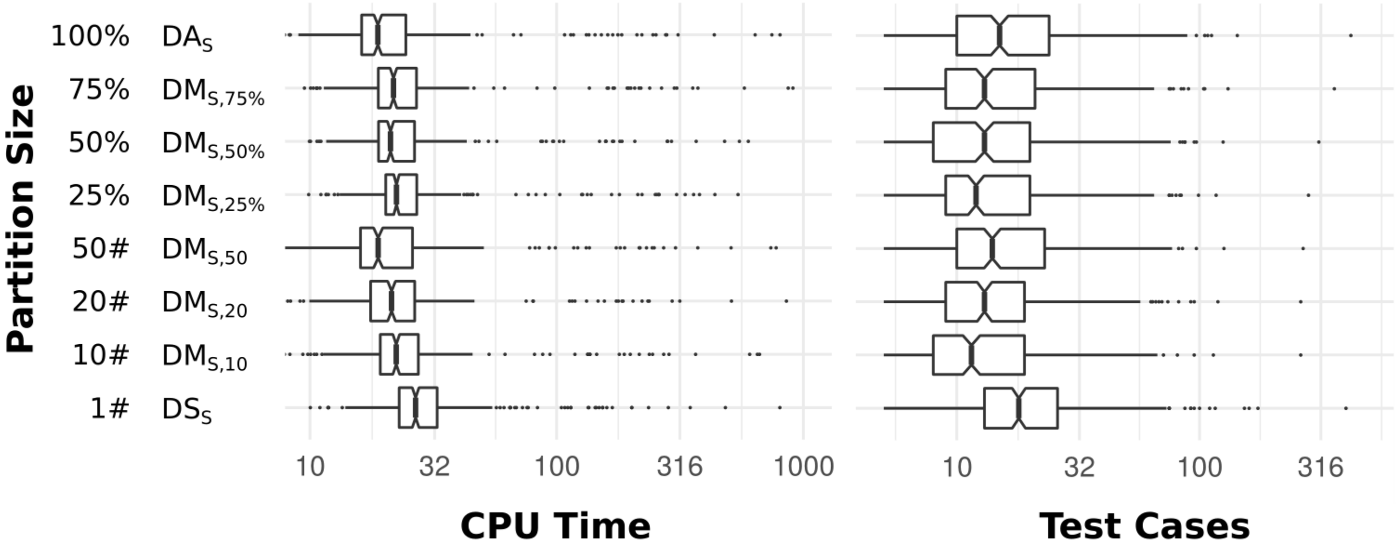
➢ Median Speedup by DA, as compared to DS: **1.4**

- Avg. Test-Suite Size: 24.9 (DS) - 21.4 (DS)
- Avg. Coverage Rate: 99.7% (DS) - 99.8% (DA)
- Avg. Fault-Detection Rate: 89.9% (DS) - 86.8% (DA)

➢ DA (slightly) improves testing efficiency under stable testing effectiveness, as compared to DS

Configurable Partition Sizes (DM)

- Multi-Goal improves testing efficiency and testing effectiveness, as compared to Single-Goal and All-Goal
- Depends on partition size
- We observe a critical number of test goals constituting the best trade-off between testing efficiency and testing effectiveness (i.e., goals covered per second)



Ongoing Work

- Criteria for choosing parition sizes and clusterings of test goals
  - Static: based on CFA (static slicing …)
  - Dynamic: on-the-fly (use information from previous CPAchecker runs …)