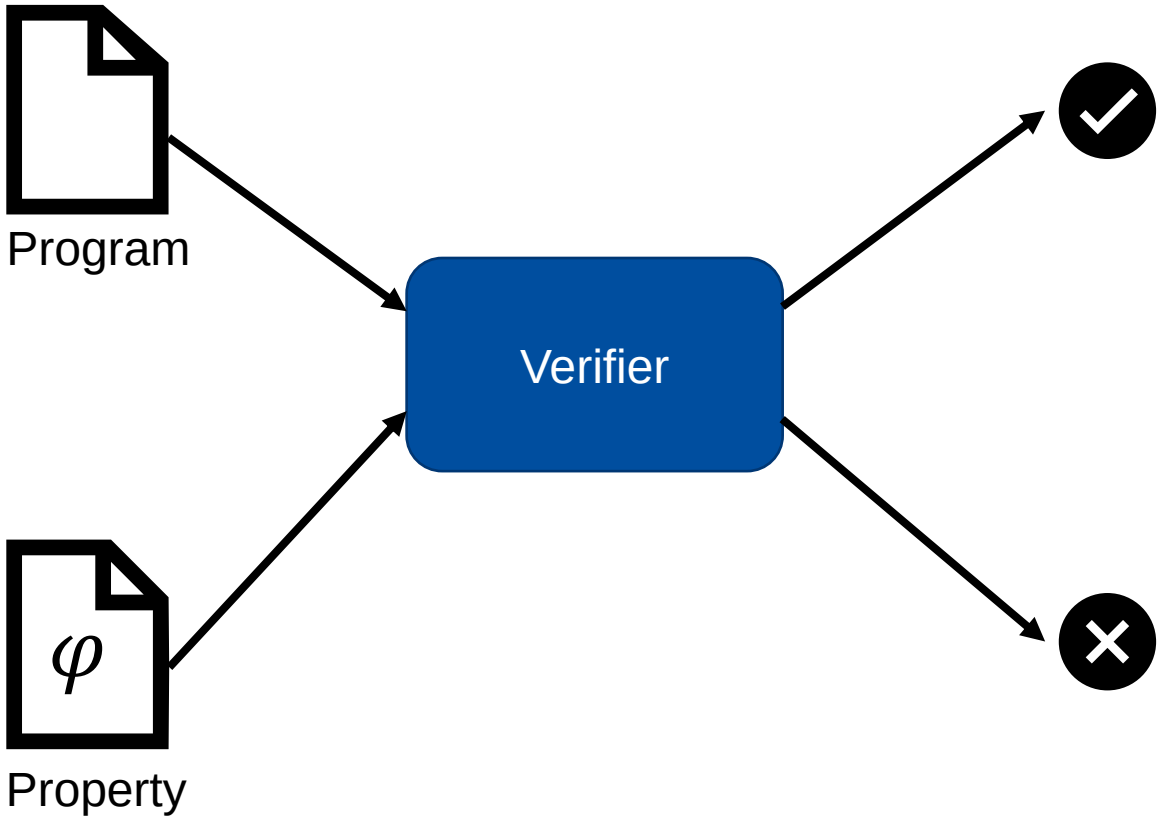
Carl von Ossietzky
Universität
Oldenburg

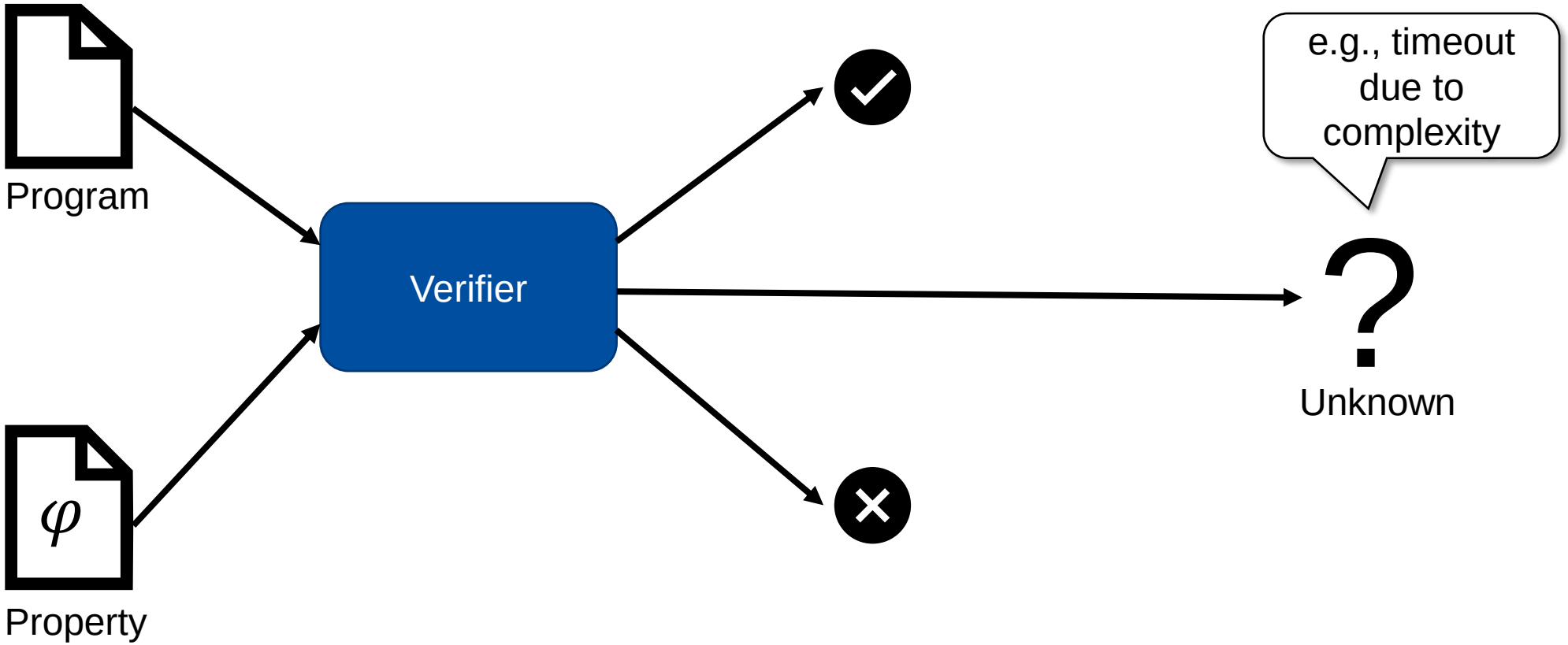
Cooperative Verification using Ranged Analysis

Jan Haltermann, Marie-Christine Jakobs,
Cedric Richter and Heike Wehrheim

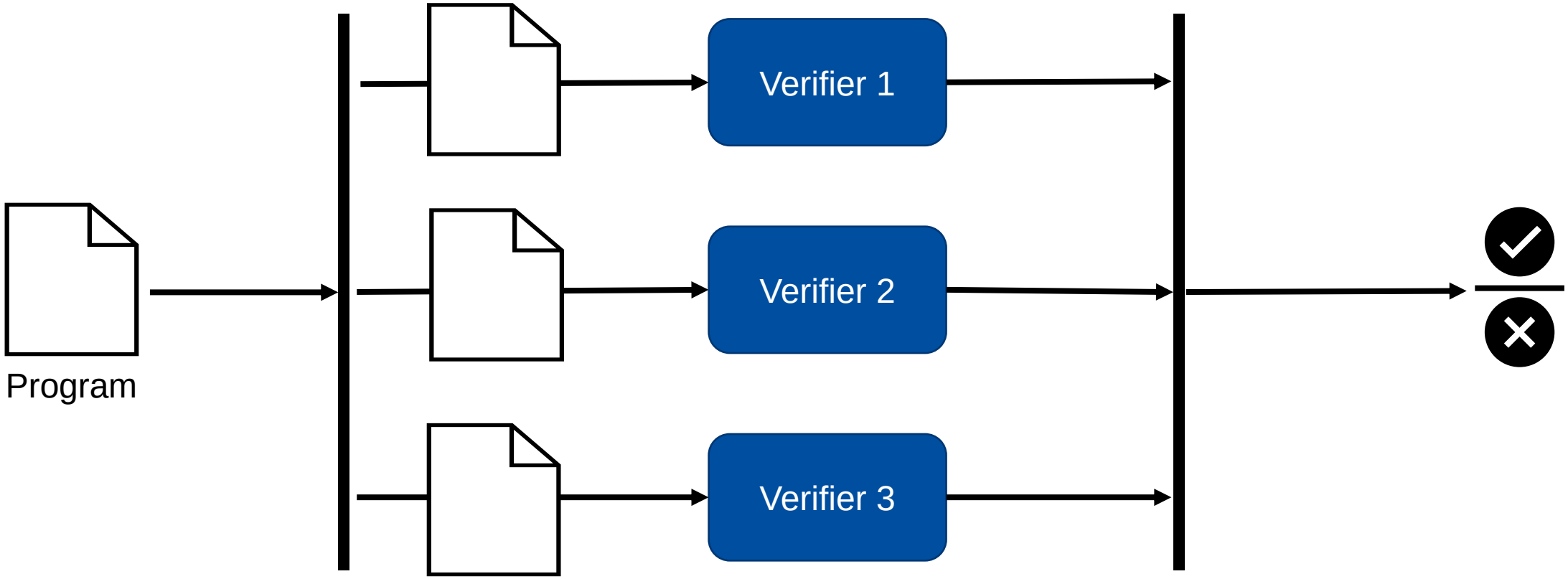
Software Verification



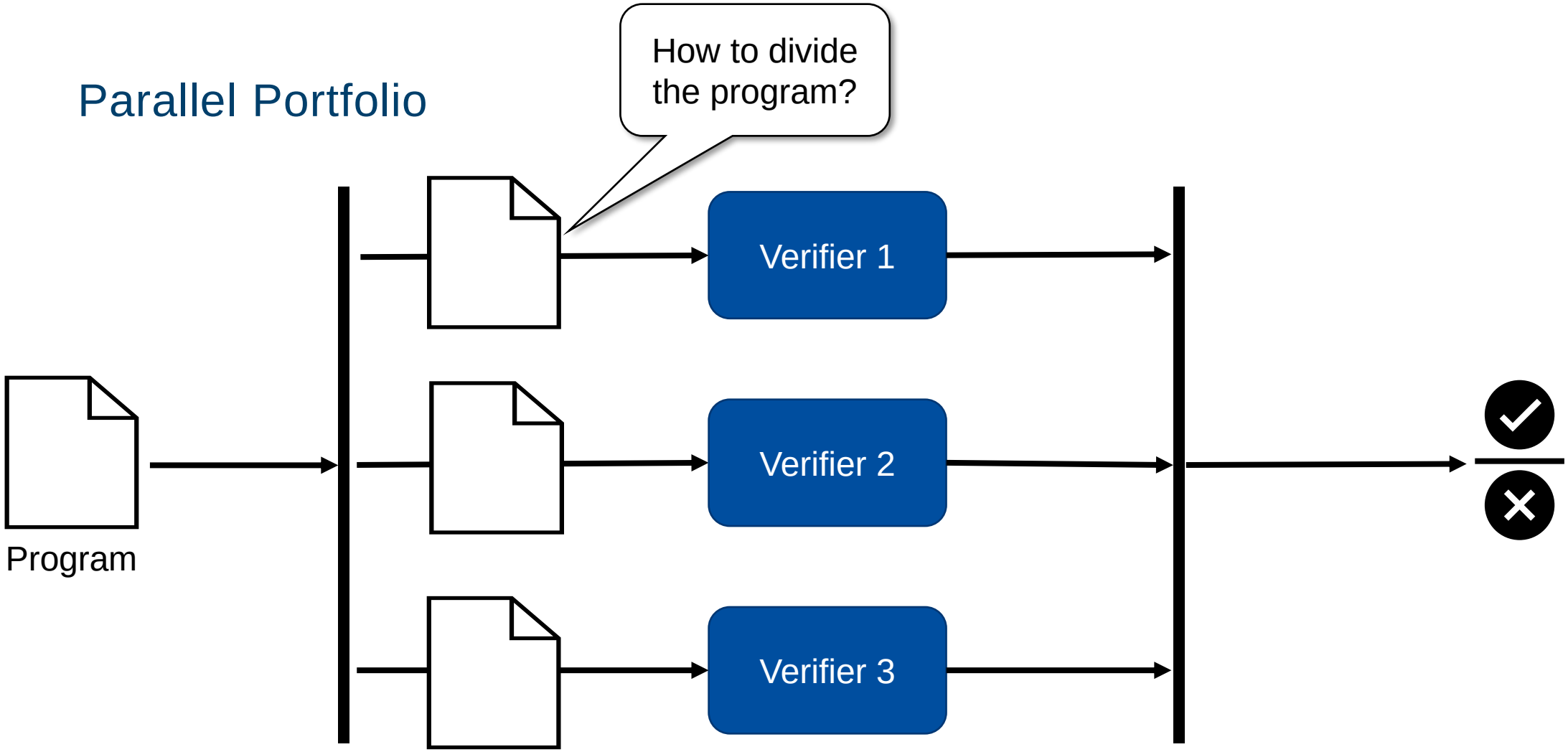
Software Verification



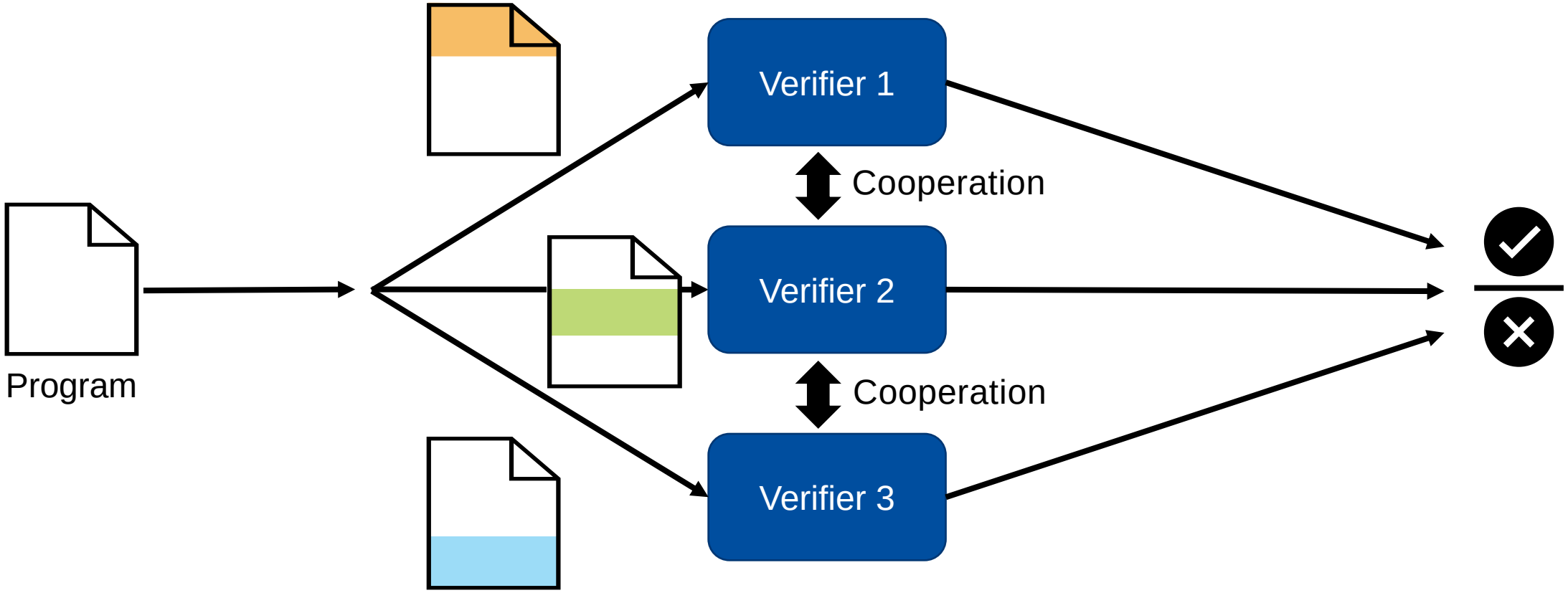
Parallel Portfolio



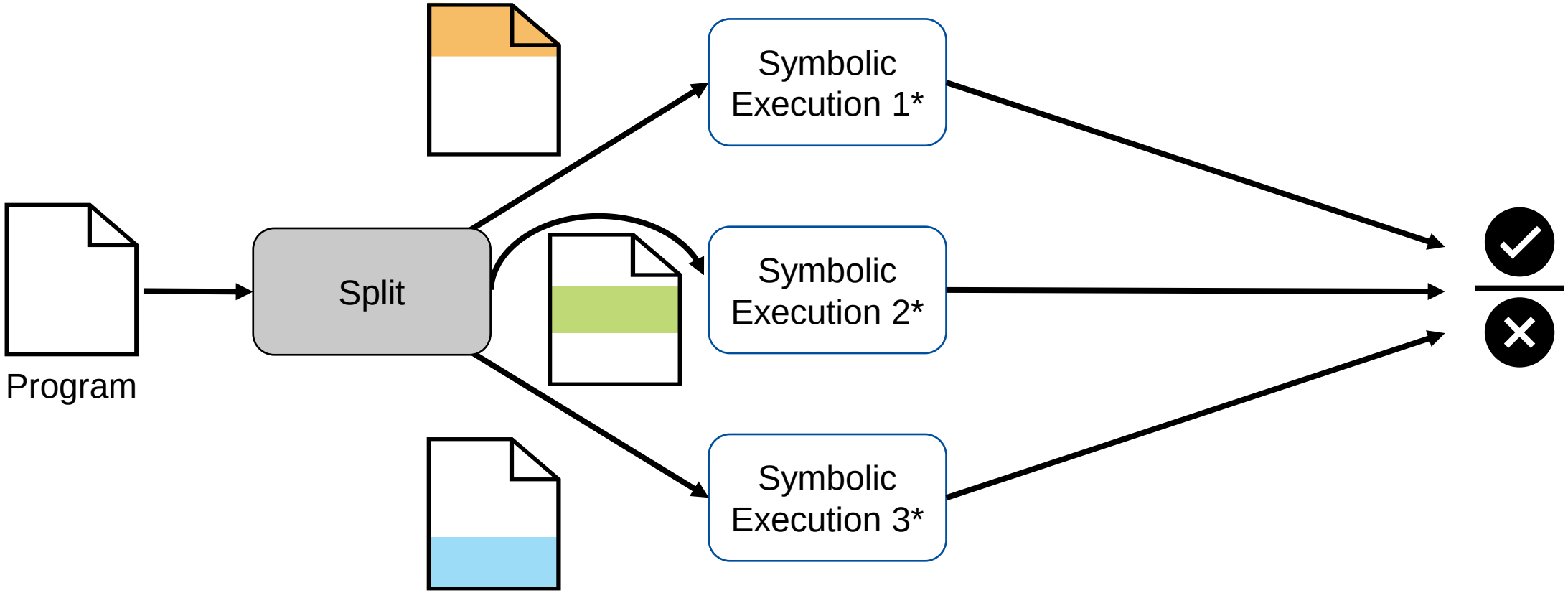
Parallel Portfolio



A Vision for Cooperative Verification



Ranged Symbolic Execution



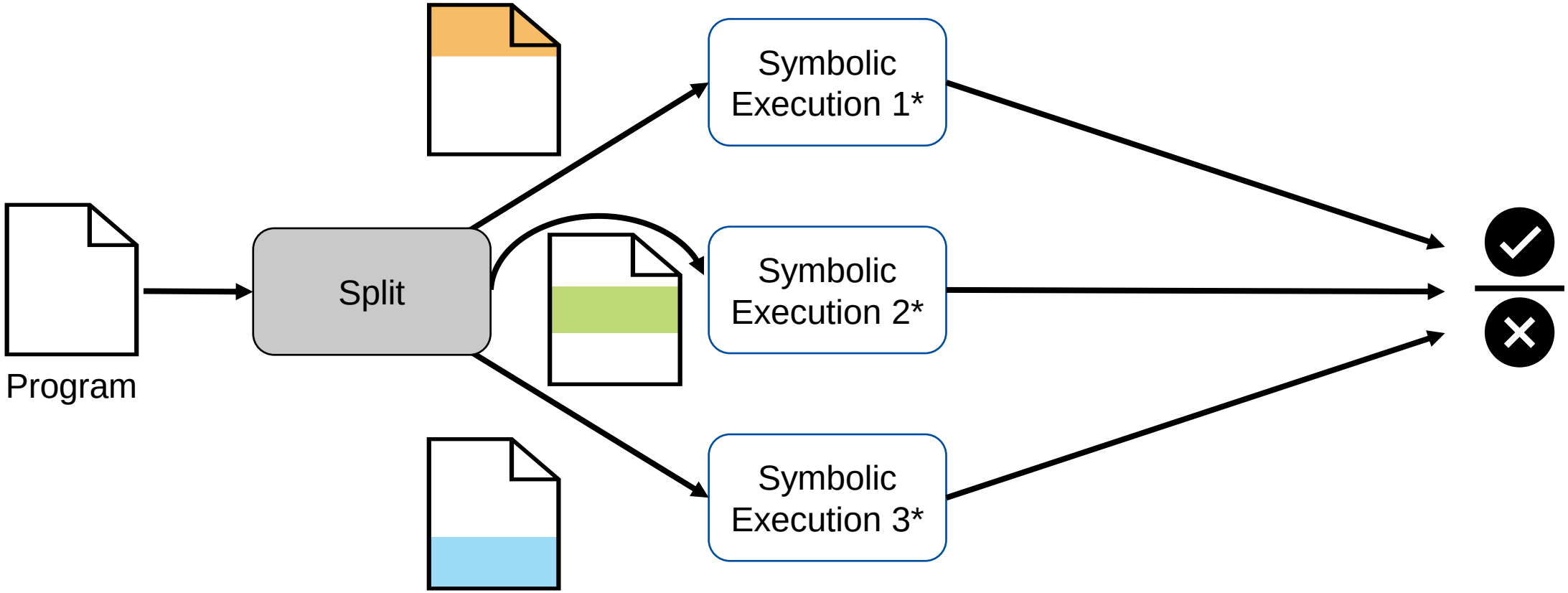
* Tools are modified

Cooperative Verification using Ranged Analysis
Jan Haltermann, Marie-Christine Jakobs, Cedric Richter and Heike Wehrheim

Siddiqui, J.H., Khurshid, S.: Scaling symbolic execution using ranged analysis. In Proc. SPLASH 2012

Ranged Symbolic Execution

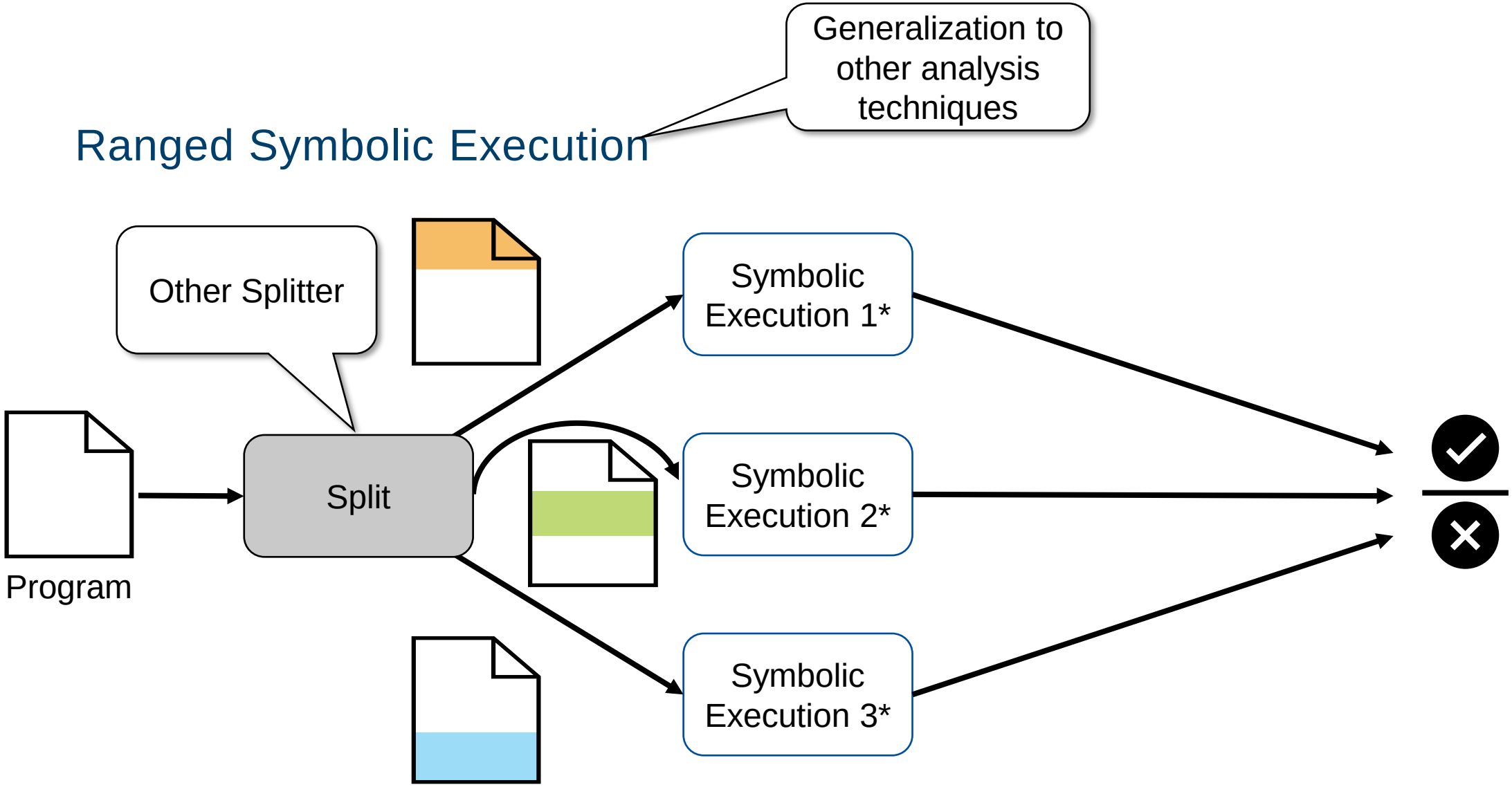
Generalization to other analysis techniques



* Tools are modified

Siddiqui, J.H., Khurshid, S.: Scaling symbolic execution using ranged analysis. In Proc. SPLASH 2012

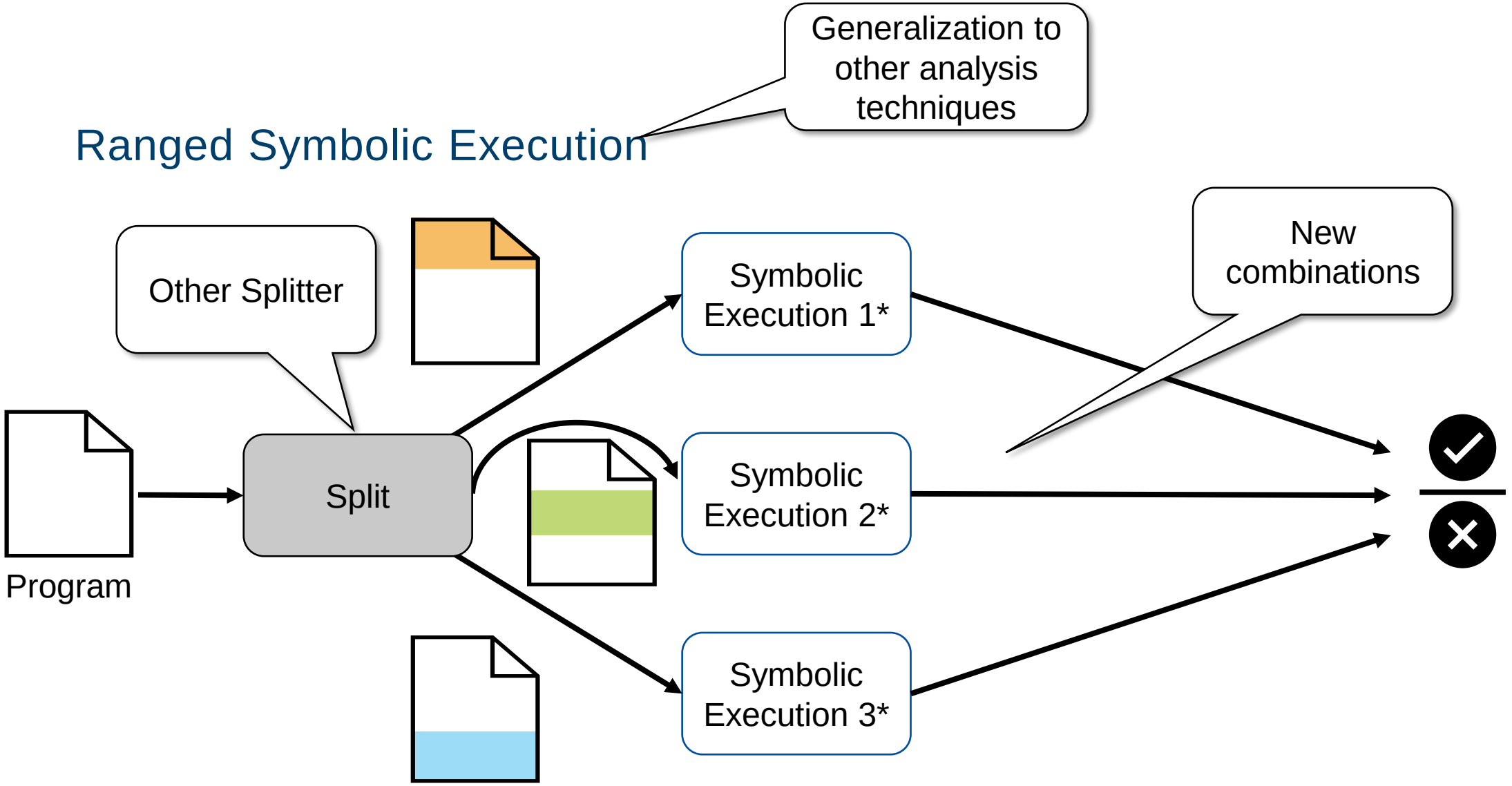
Ranged Symbolic Execution



* Tools are modified

Siddiqui, J.H., Khurshid, S.: Scaling symbolic execution using ranged analysis. In Proc. SPLASH 2012

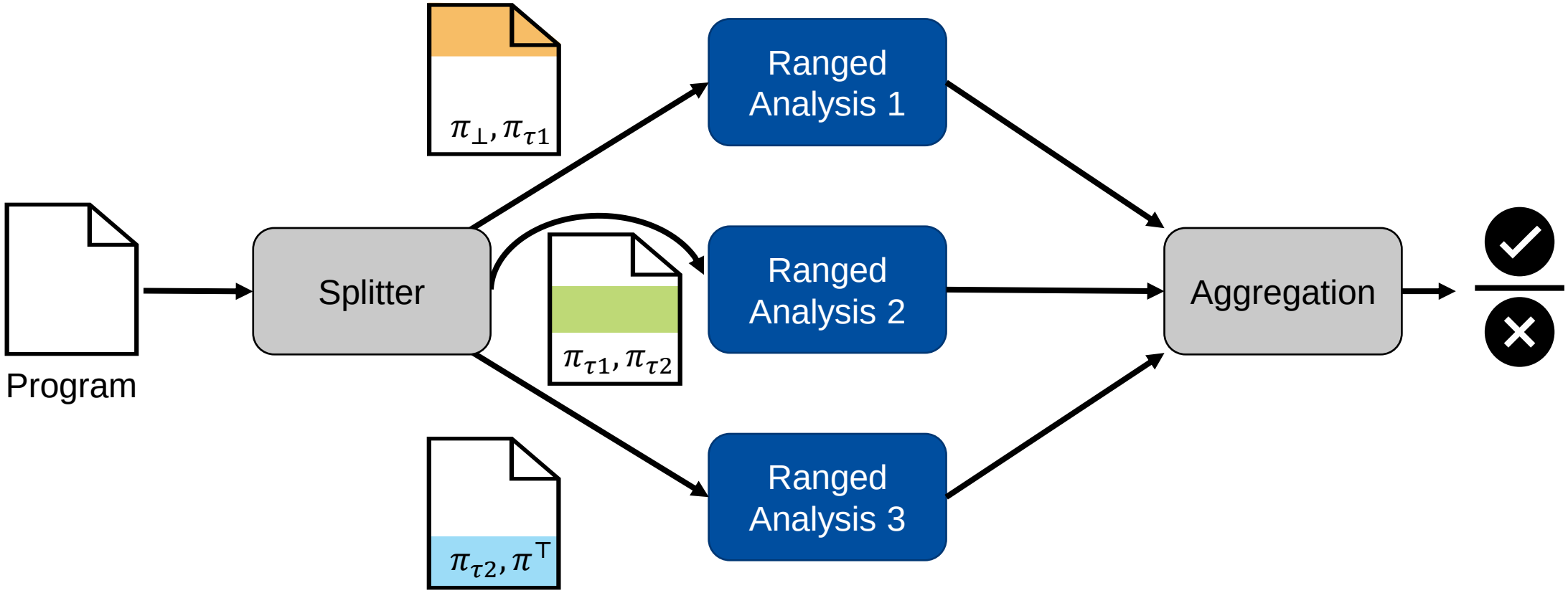
Ranged Symbolic Execution



* Tools are modified

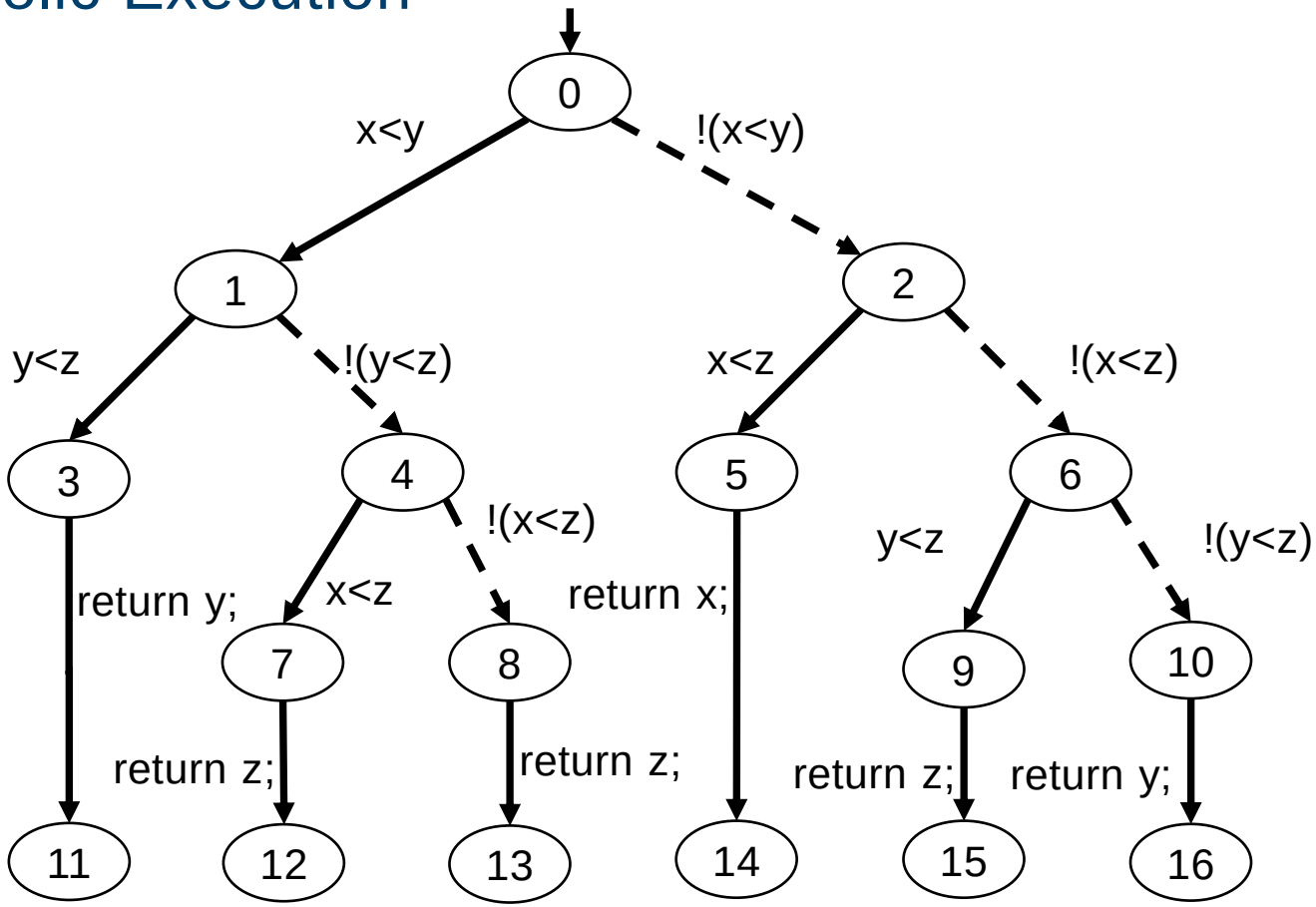
Siddiqui, J.H., Khurshid, S.: Scaling symbolic execution using ranged analysis. In Proc. SPLASH 2012

Idea of Parallel Program Analysis via Range Splitting



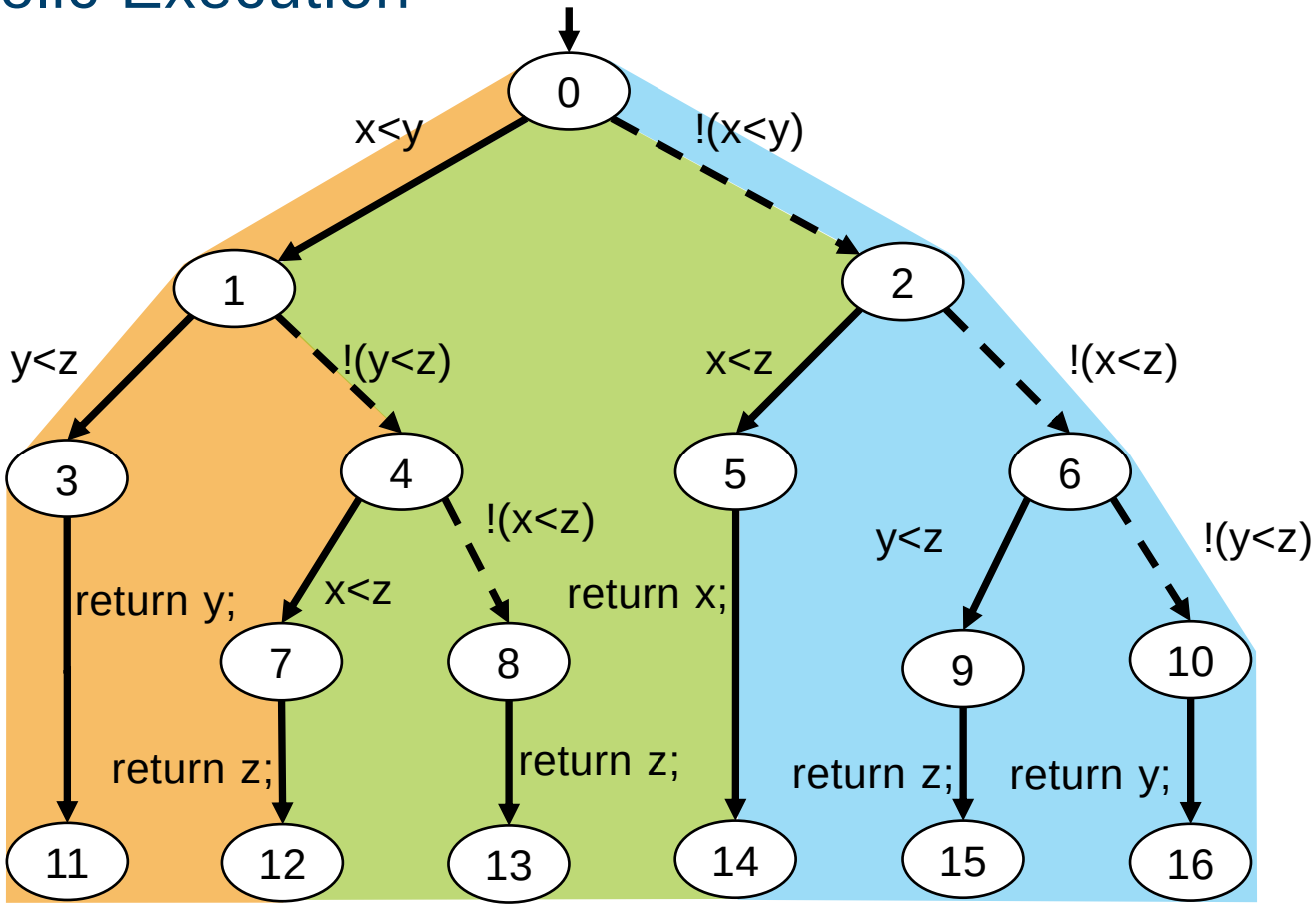
Example for Ranged Symbolic Execution

```
0  int mid(int x, int y, int z){
1    if (x < y){
2      if (y < z) return y;
3      if (x < z) return z;
4      else return x ;
5    }
6    if (x < z) return x;
7    if (y < z) return z;
8    else return y;
9  }
```

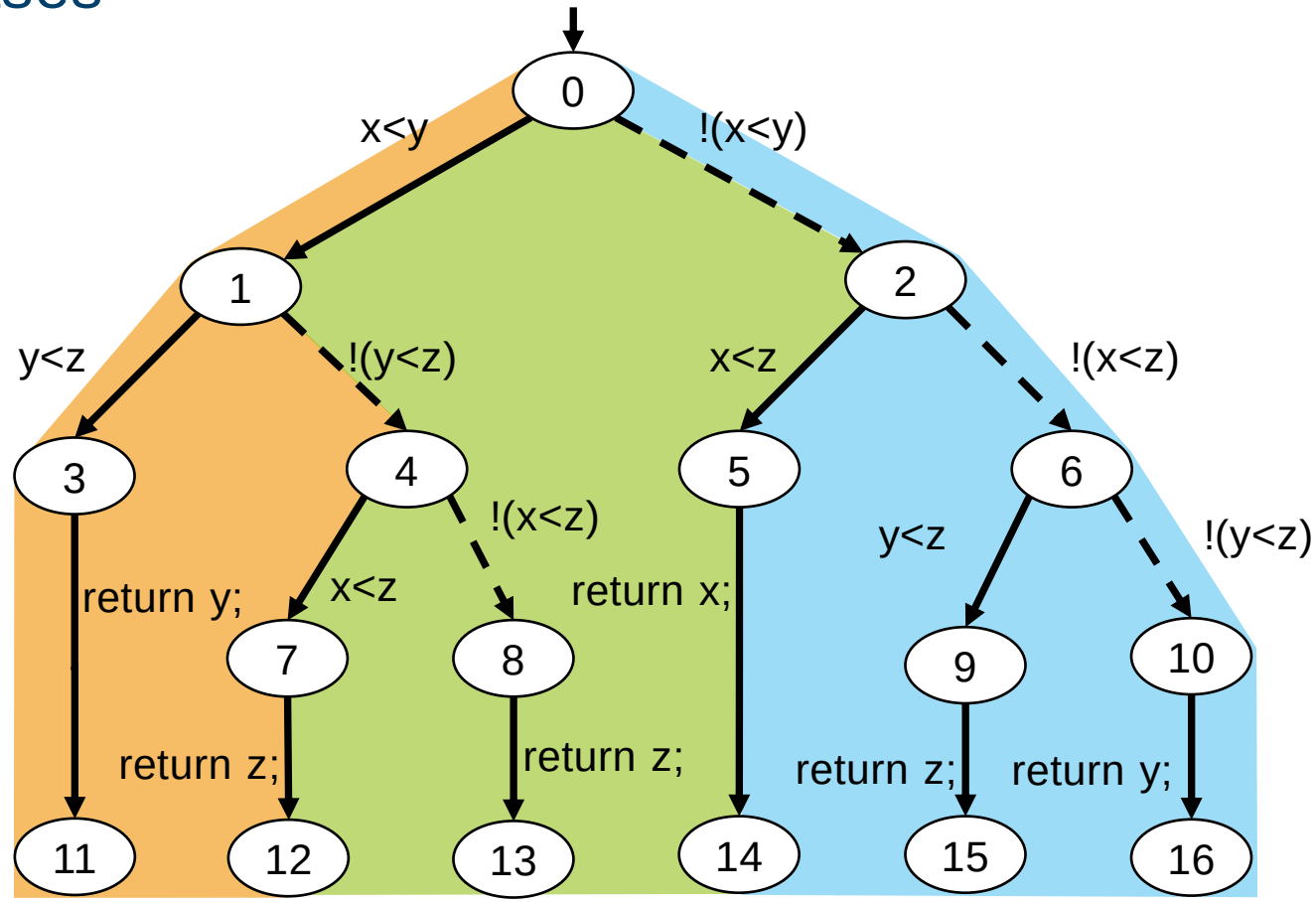


Example for Ranged Symbolic Execution

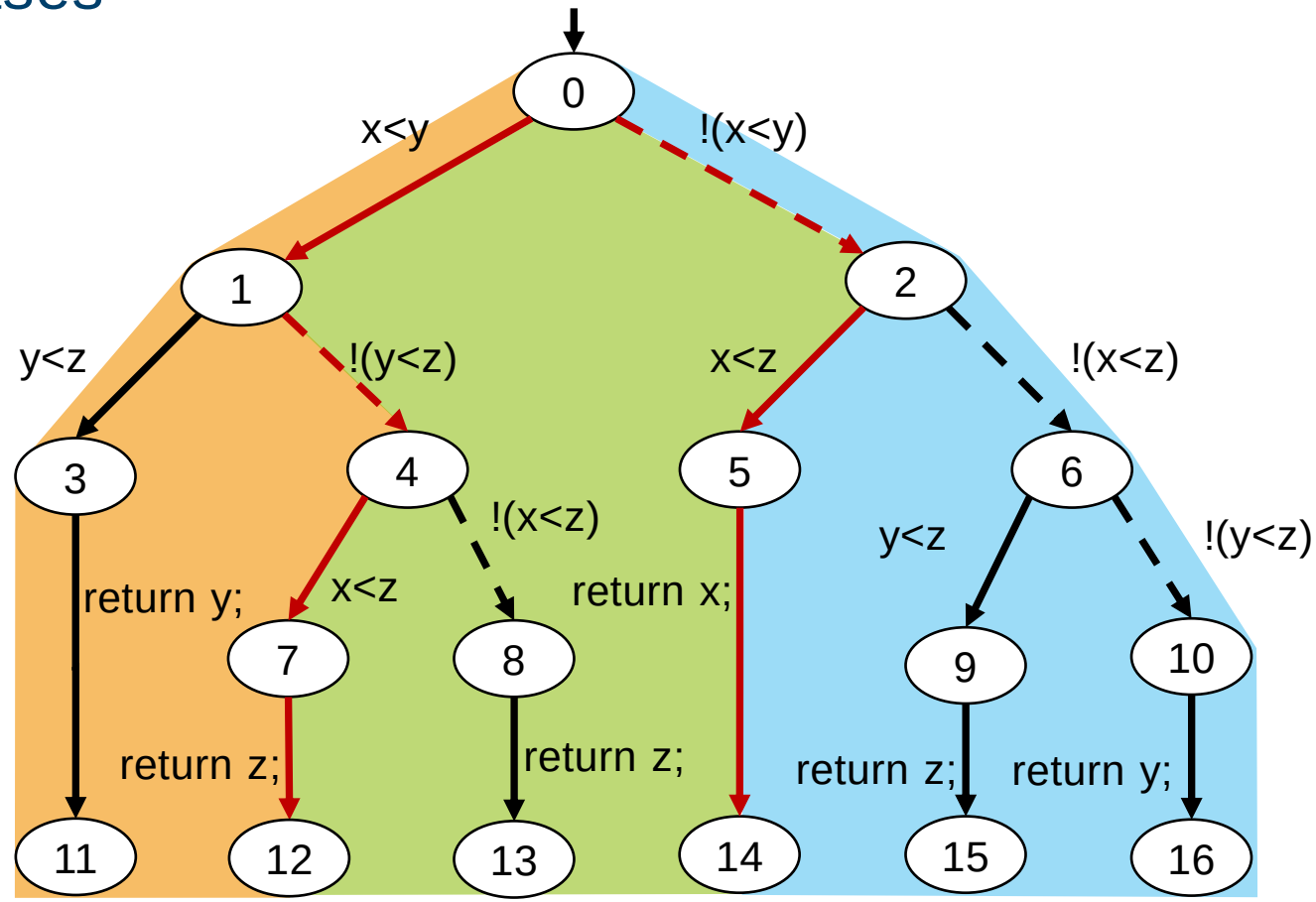
```
0  int mid(int x, int y, int z){
1    if (x < y){
2      if (y < z) return y;
3      if (x < z) return z;
4      else return x ;
5    }
6    if (x < z) return x;
7    if (y < z) return z;
8    else return y;
9  }
```



Paths Ordering and Testcases



Paths Ordering and Testcases

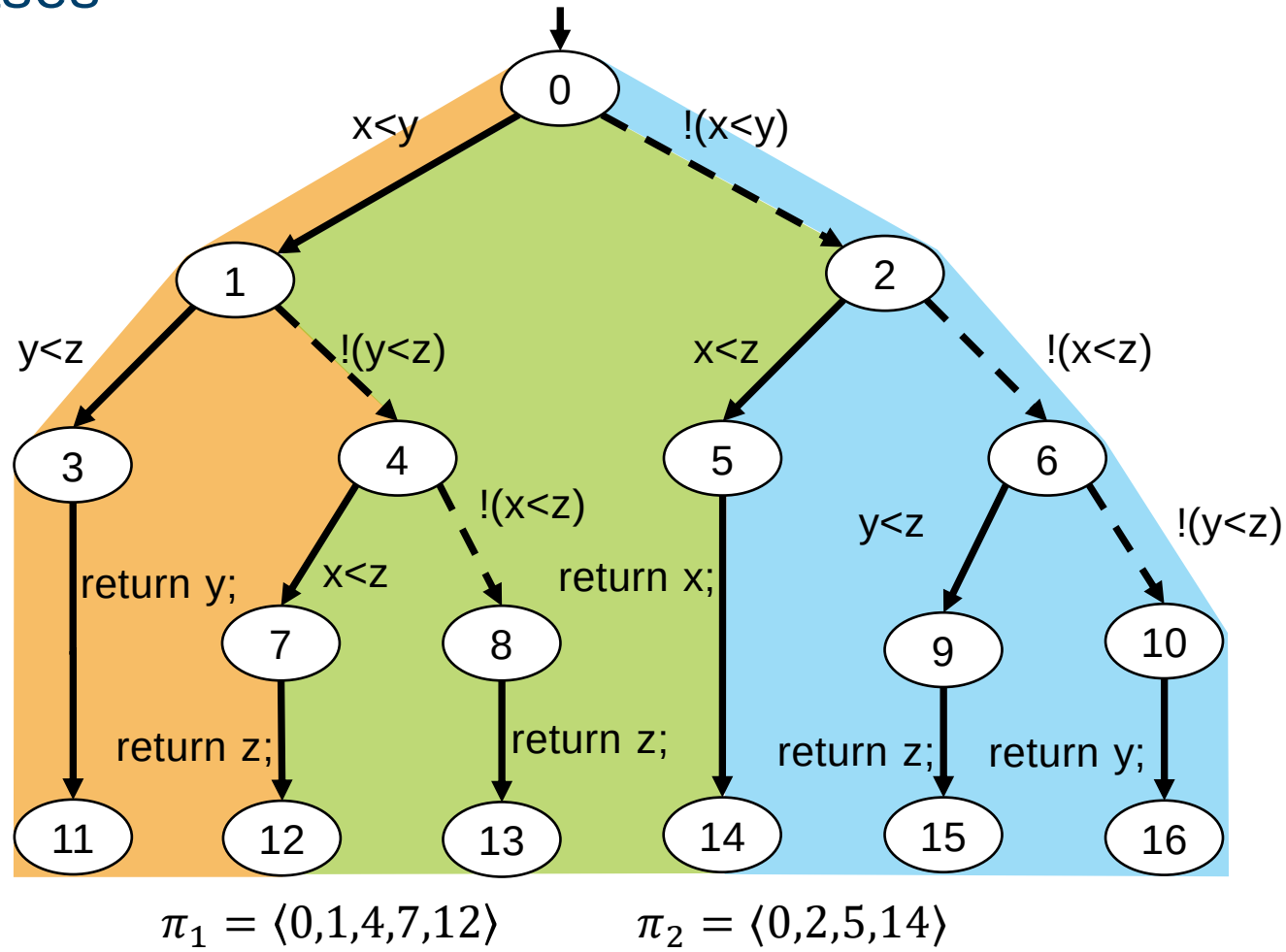


$\pi_1 = \langle 0,1,4,7,12 \rangle$ $\pi_2 = \langle 0,2,5,14 \rangle$

Paths Ordering and Testcases

– Ordering:

$$\pi_1 < \pi_2$$



$\pi_1 = \langle 0,1,4,7,12 \rangle$ $\pi_2 = \langle 0,2,5,14 \rangle$

Paths Ordering and Testcases

– Ordering:

$$\pi_1 < \pi_2$$

– Testcase:

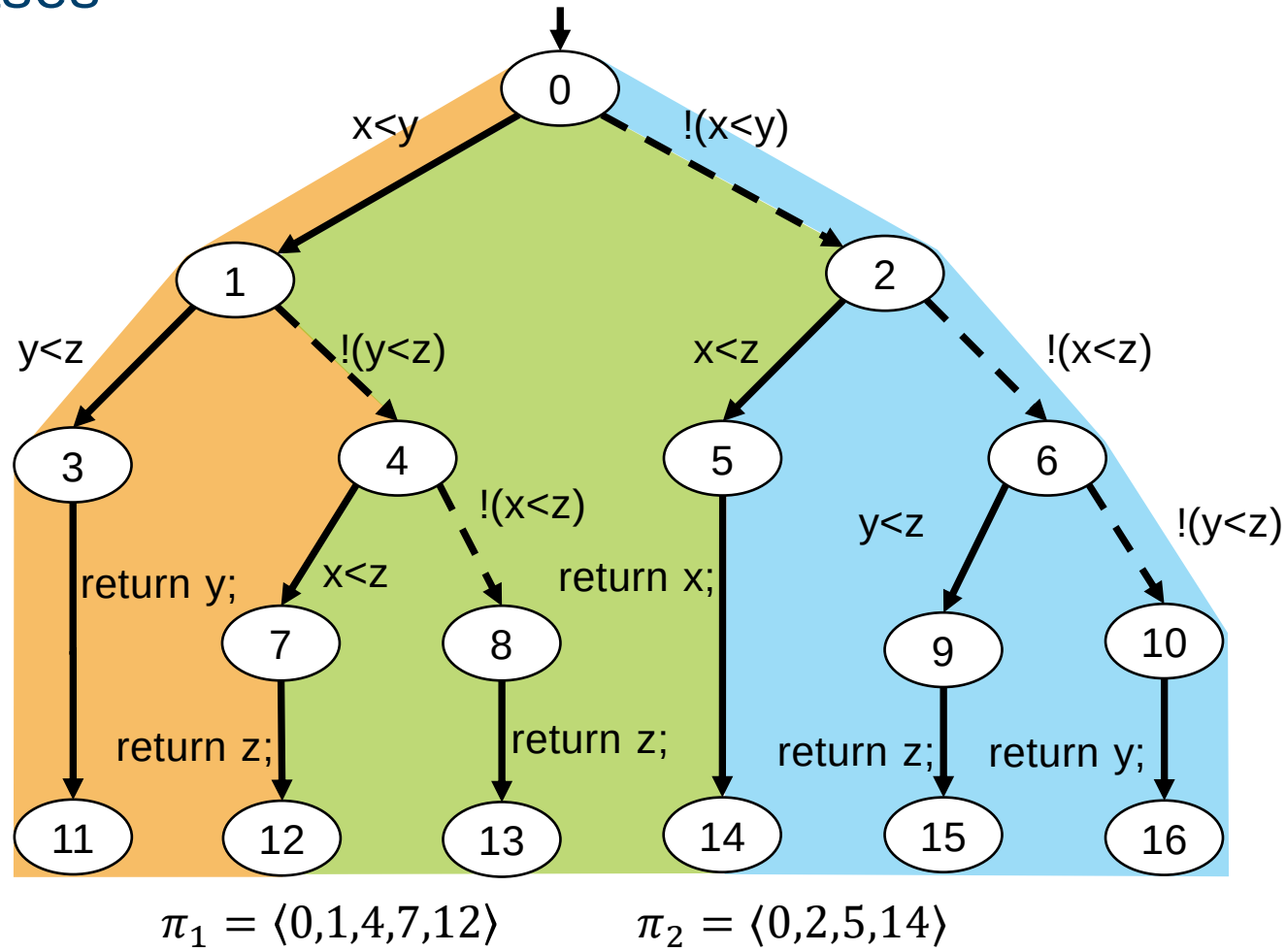
$$\tau = \{x: 0, y: 2, z: 1\}$$

– Path induced by testcase:

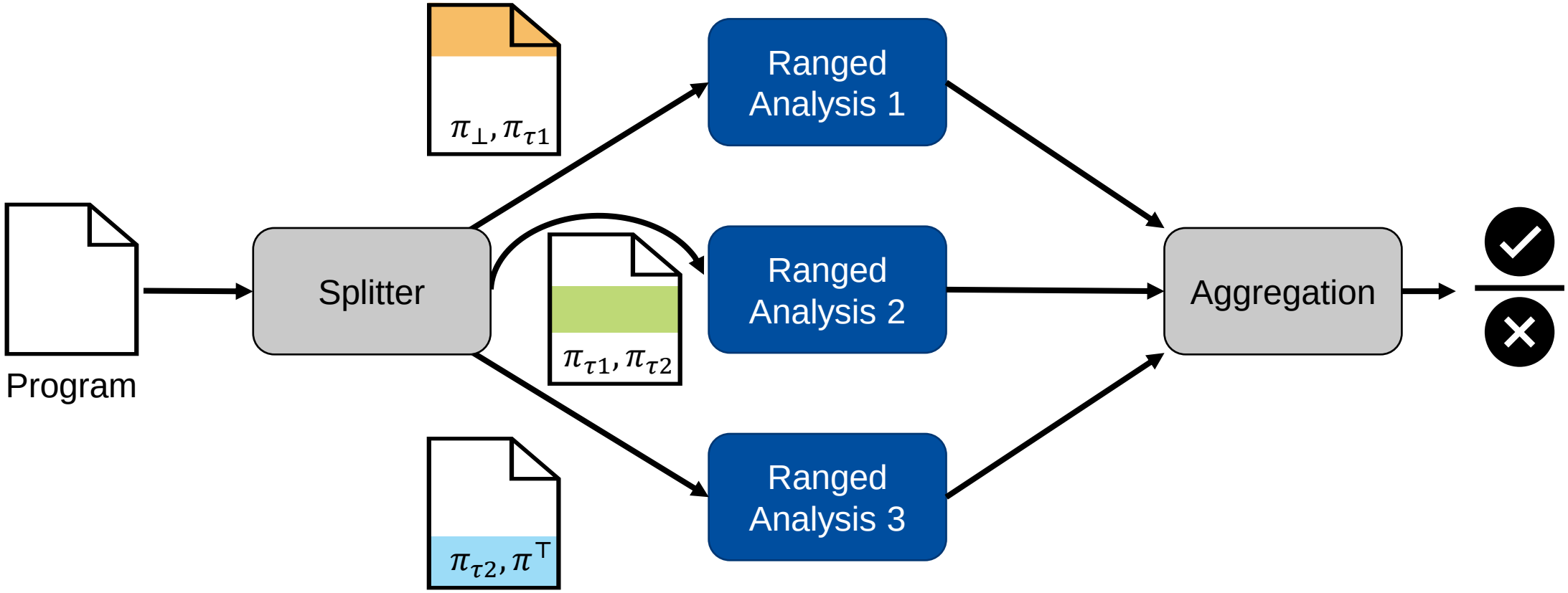
$$\pi_\tau = \langle 0,1,4,7,12 \rangle = \pi_1$$

– Smallest path: $\pi_\perp$

– Greatest path: $\pi^\top$



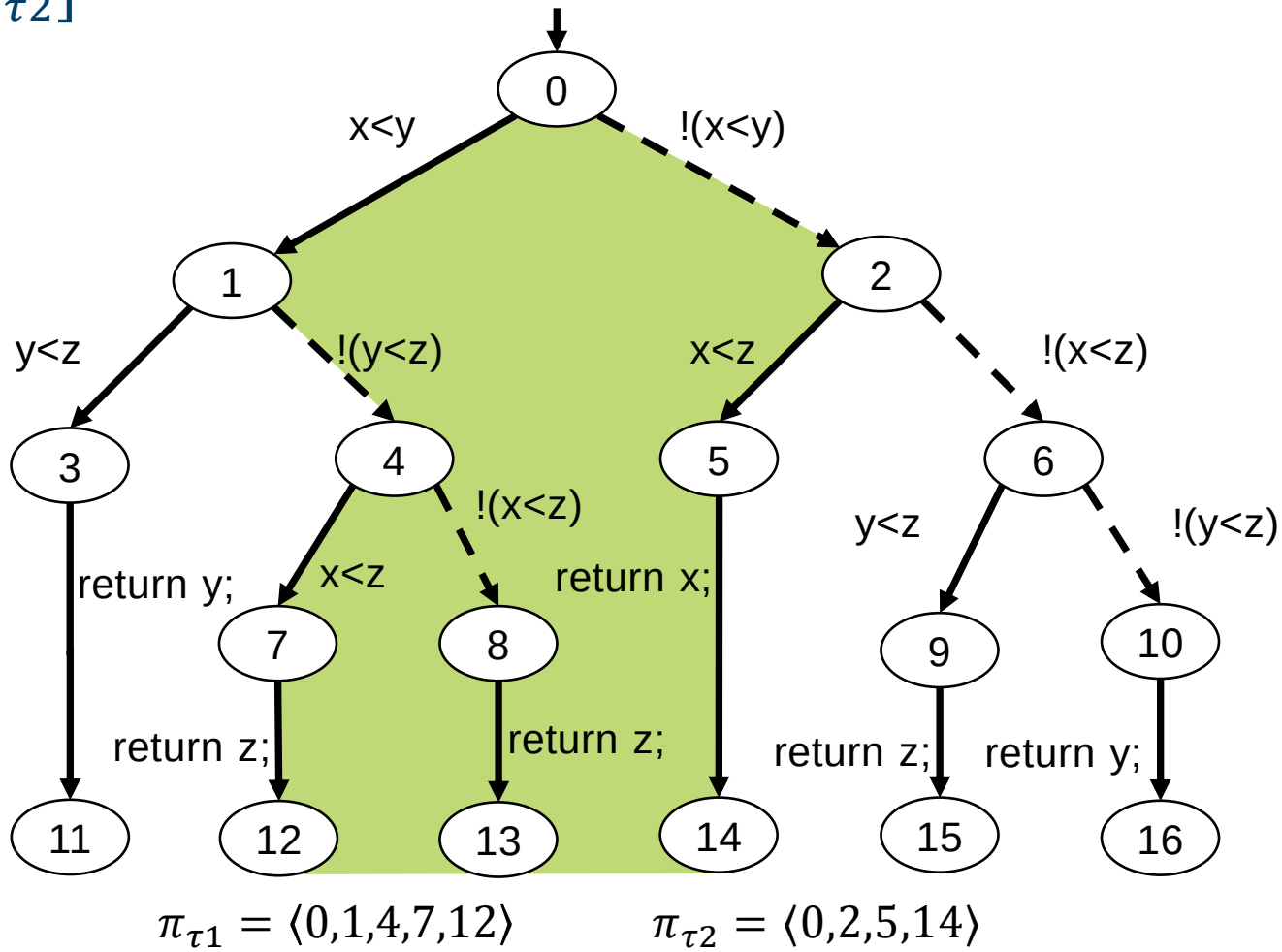
Idea of Parallel Program Analysis via Range Splitting



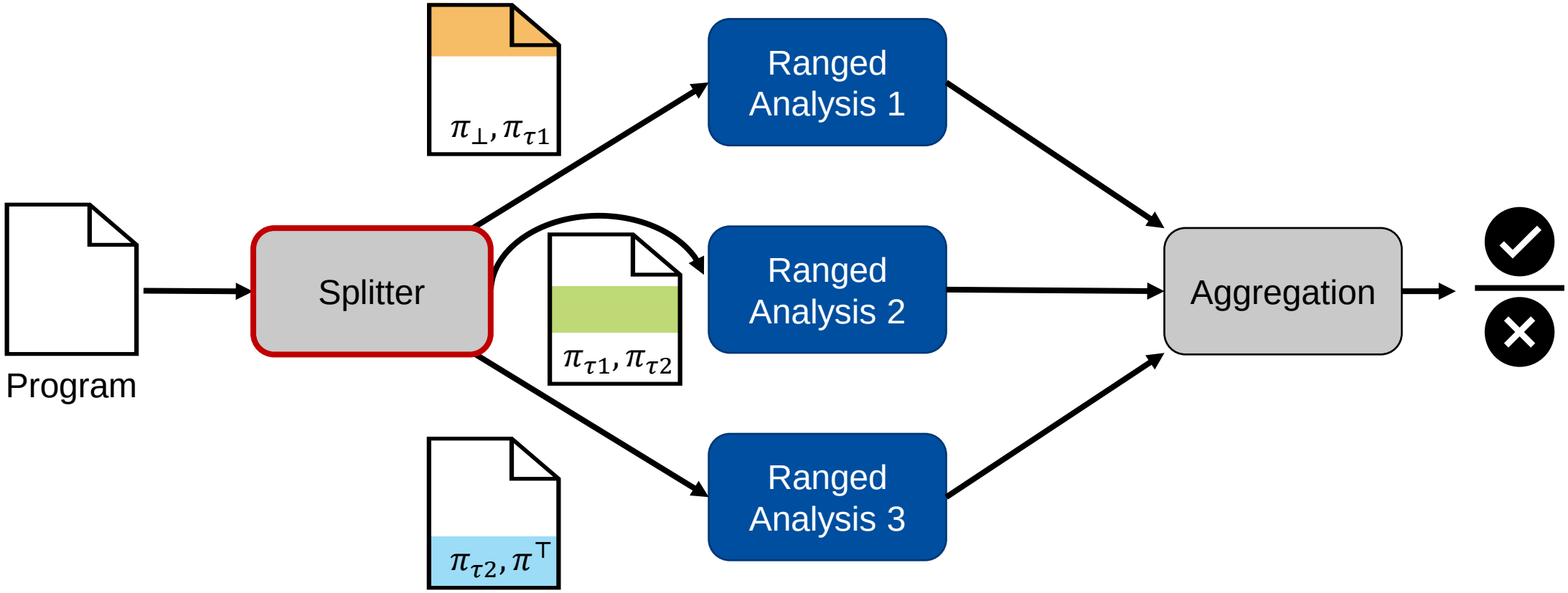
Ranged Analysis for $[\pi_{\tau 1}, \pi_{\tau 2}]$

Ranged Analysis

$\mathbb{R}_{[\pi_\perp, \pi_{\tau 2}]} \times \mathbb{A} \times \mathbb{R}_{[\pi_{\tau 1}, \pi_\top]}$

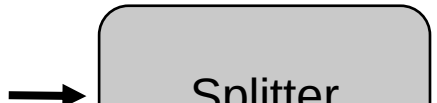


Splitting


$\pi_\perp, \pi_{\tau 1}$
Ranged Analysis 1
Program
Splitter
$\pi_{\tau 1}, \pi_{\tau 2}$
Ranged Analysis 2
Aggregation
$\pi_{\tau 2}, \pi^\top$
Ranged Analysis 3

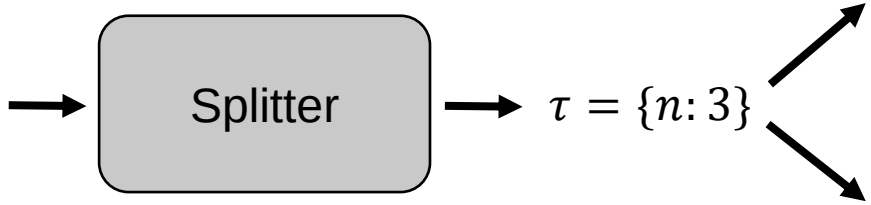
Splitting

```
1  int main(int n){
2    int x=n, y=0;
3    while(x>0){
4      x--; y++;
5    }
6    assert(y==n);
7  }
```

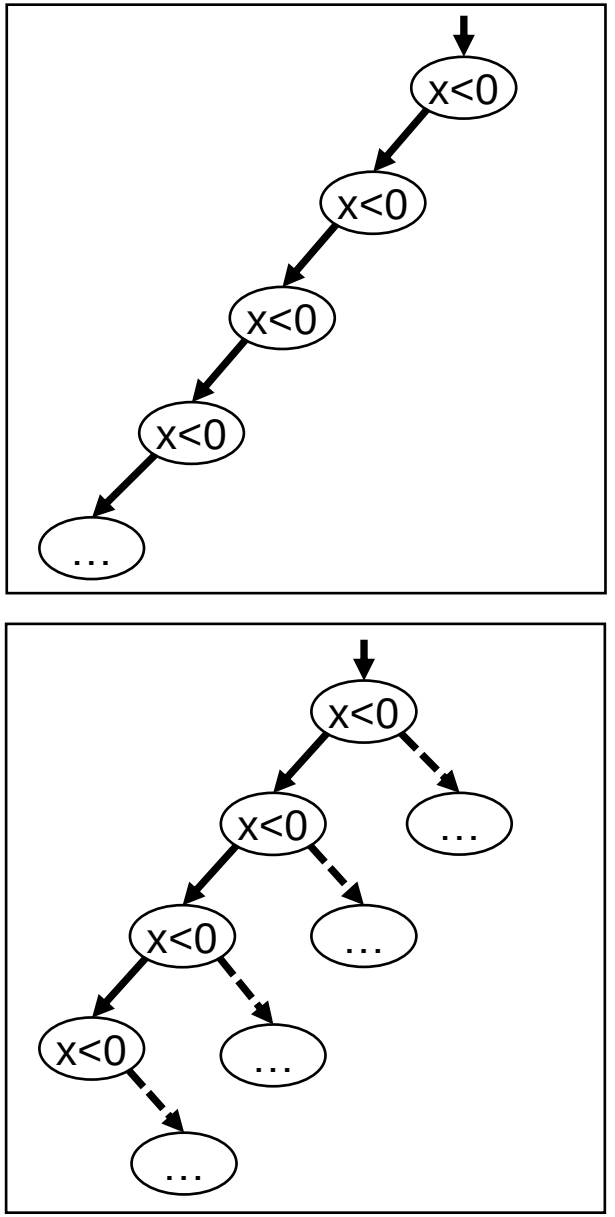


Splitting

```
1  int main(int n){
2    int x=n, y=0;
3    while(x>0){
4      x--; y++;
5    }
6    assert(y==n);
7  }
```

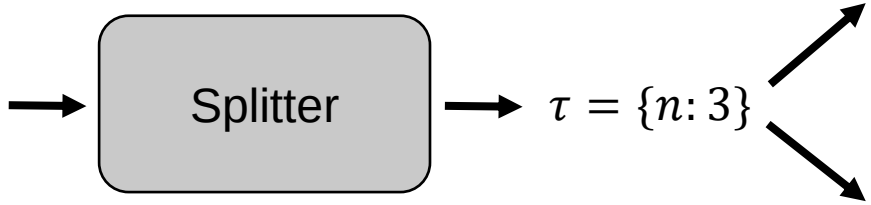


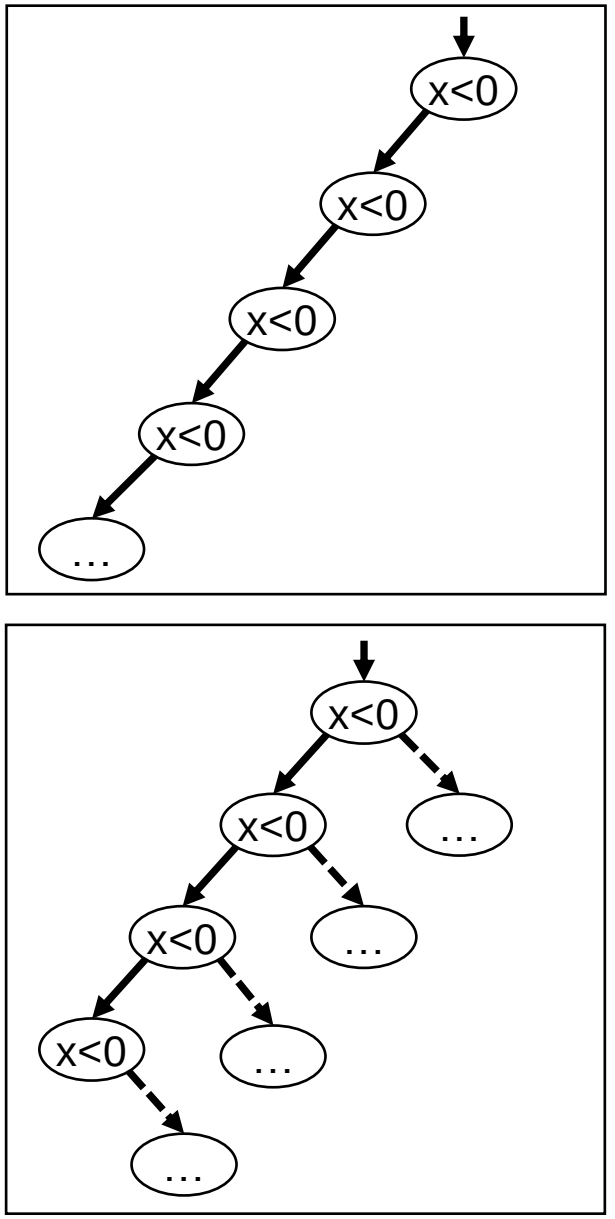
$\tau = \{n : 3\}$



Splitting

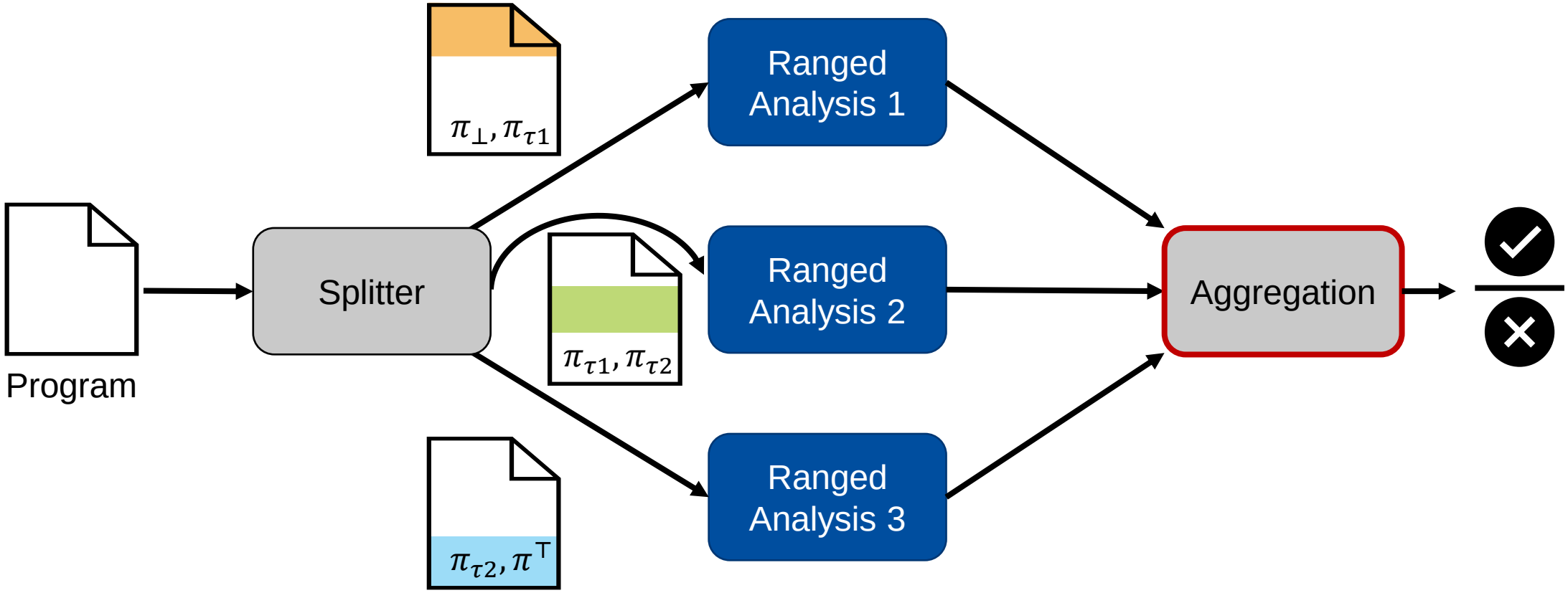
```
1  int main(int n){
2    int x=n, y=0;
3    while(x>0){
4      x--; y++;
5    }
6    assert(y==n);
7  }
```



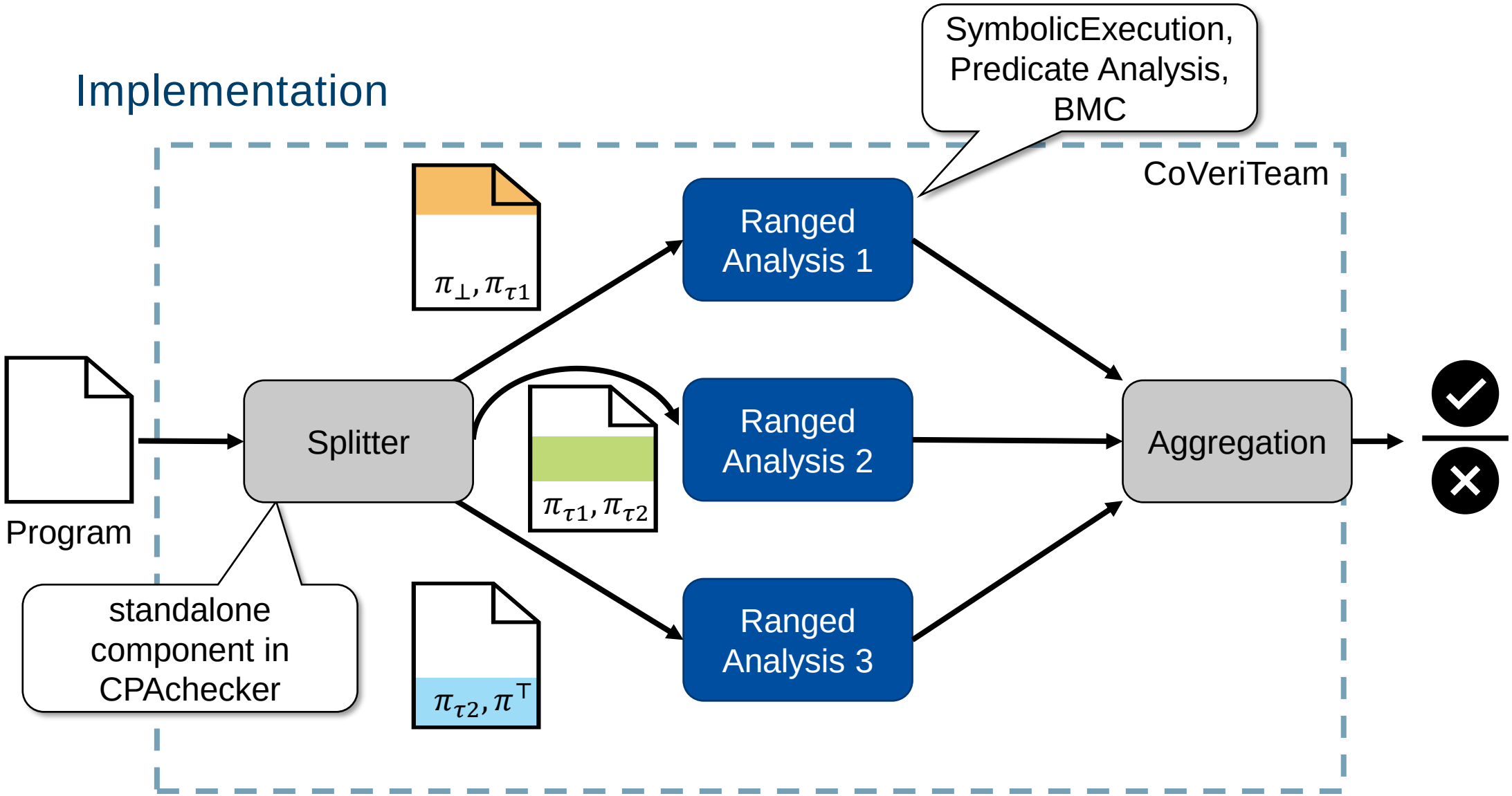


- Bounding number of Loop Unrollings (LB)
  - 3 (LB3) and 10 (LB10) iterations
- Randomly
  - 50%(RDM) and 90%(RDM9) prob. for true branch

Aggregation



Implementation



Evaluation

RQ1: Can composition of ranged analyses increase effectiveness and efficiency of symbolic execution?

RQ2: Can other analyses and combinations also benefit?

RQ1: Effect of Symbolic Analysis - Effectiveness

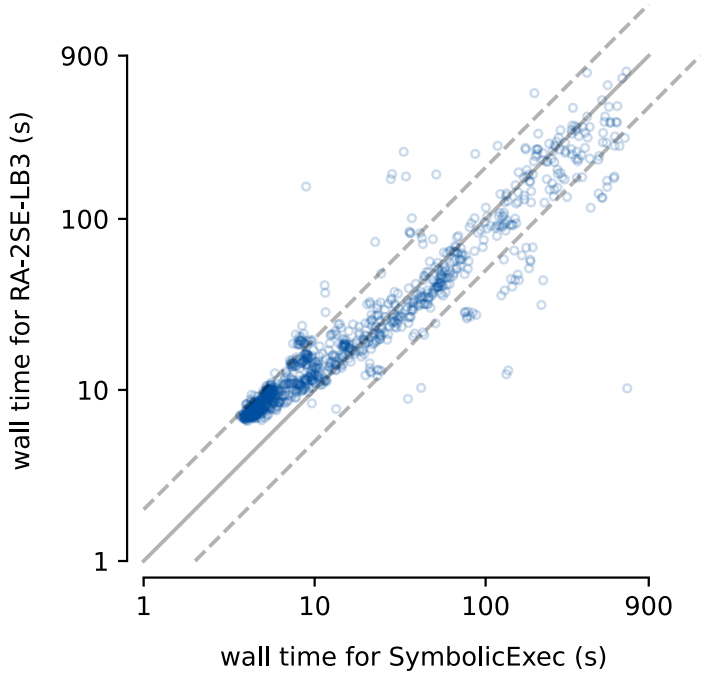
– Setup: 5400 C-Tasks, 15 min CPU-time, 15GB RAM per Tool

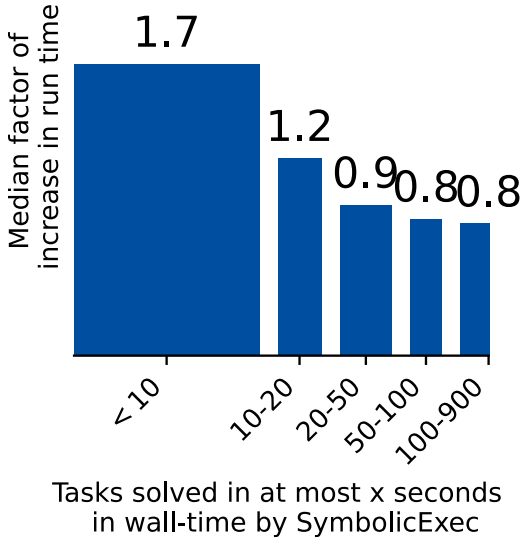
	correct alarm	correct proof	**correct overall**
SymbExec	821	565	1386
Ra-2Se-Lb3	921	566	1487
Ra-3Se-Lb	969	563	1532

+ 146 correct answers
+ 181 uniques

→ Effectiveness increases for finding violations

RQ1: Effect of Symbolic Analysis - Efficiency





→ Reduced real overall time consumed for complex or large tasks

RQ2: Effect for other (combination of) analyses

	correct proof	correct alarm	**correct overall**
BMC	930	1604	2534
RA-2BMC-RDM9	932	1573	2505

- 29 correct answers, but:
+ 48 uniques

→ Effectiveness increases partially for BMC,

RQ2: Effect for other (combination of) analyses

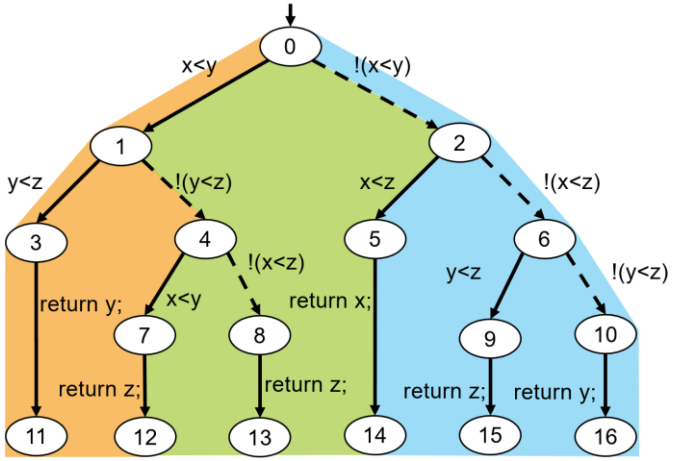
	correct proof	correct alarm	**correct overall**
BMC	930	1604	2534
RA-2BMC-RDM9	932	1573	2505
SYMBEXEC	565	821	1386
PRED	1107	1147	2254
RA-SE-PRED-LB3	913	1108	2021

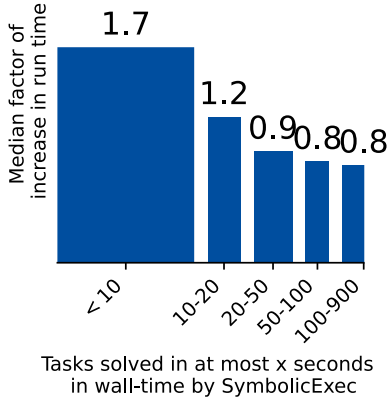
- 29 correct answers, but:
+ 48 uniques

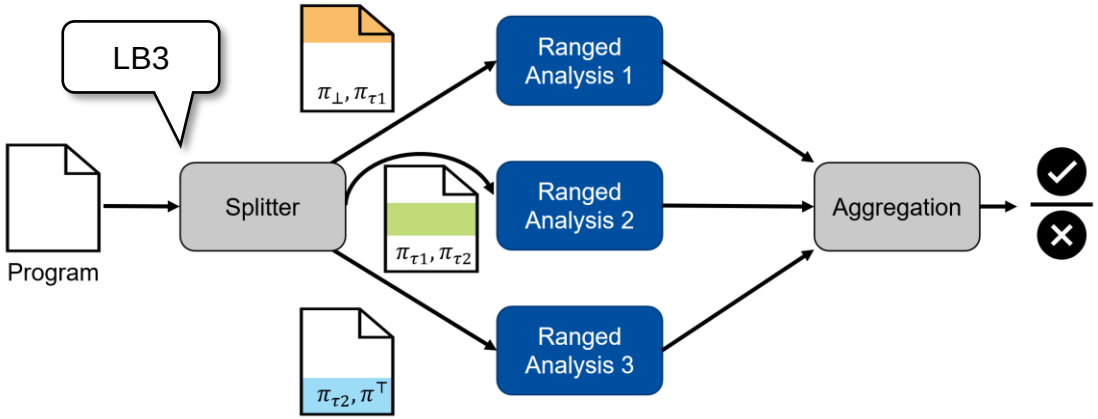
+ 635 correct answers
+ 30 uniques (vs. SYMBEXEC and PRED)

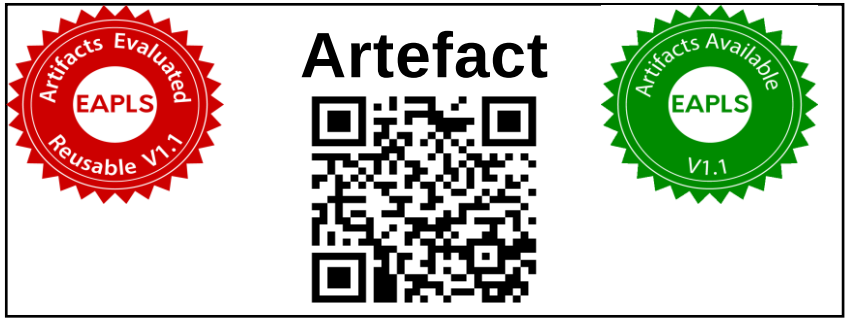
→ Effectiveness increases partially for BMC,
→ Effectiveness drastically increased for SYMBEXEC and slightly decreased for PRED
→ Efficiency comparable for complex or large task

Summary









Future and ongoing work:

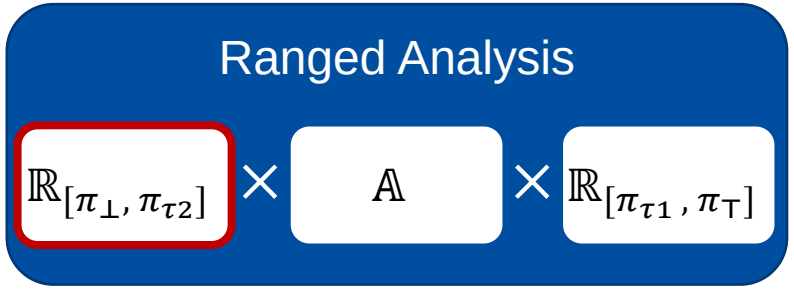
- New tool combinations
- Work Stealing
- Usage of non-CPA-based verifiers
- Joining Witnesses
- Usage in parallelized setting
  → Tackle scaling issues
- Novel splitting strategies

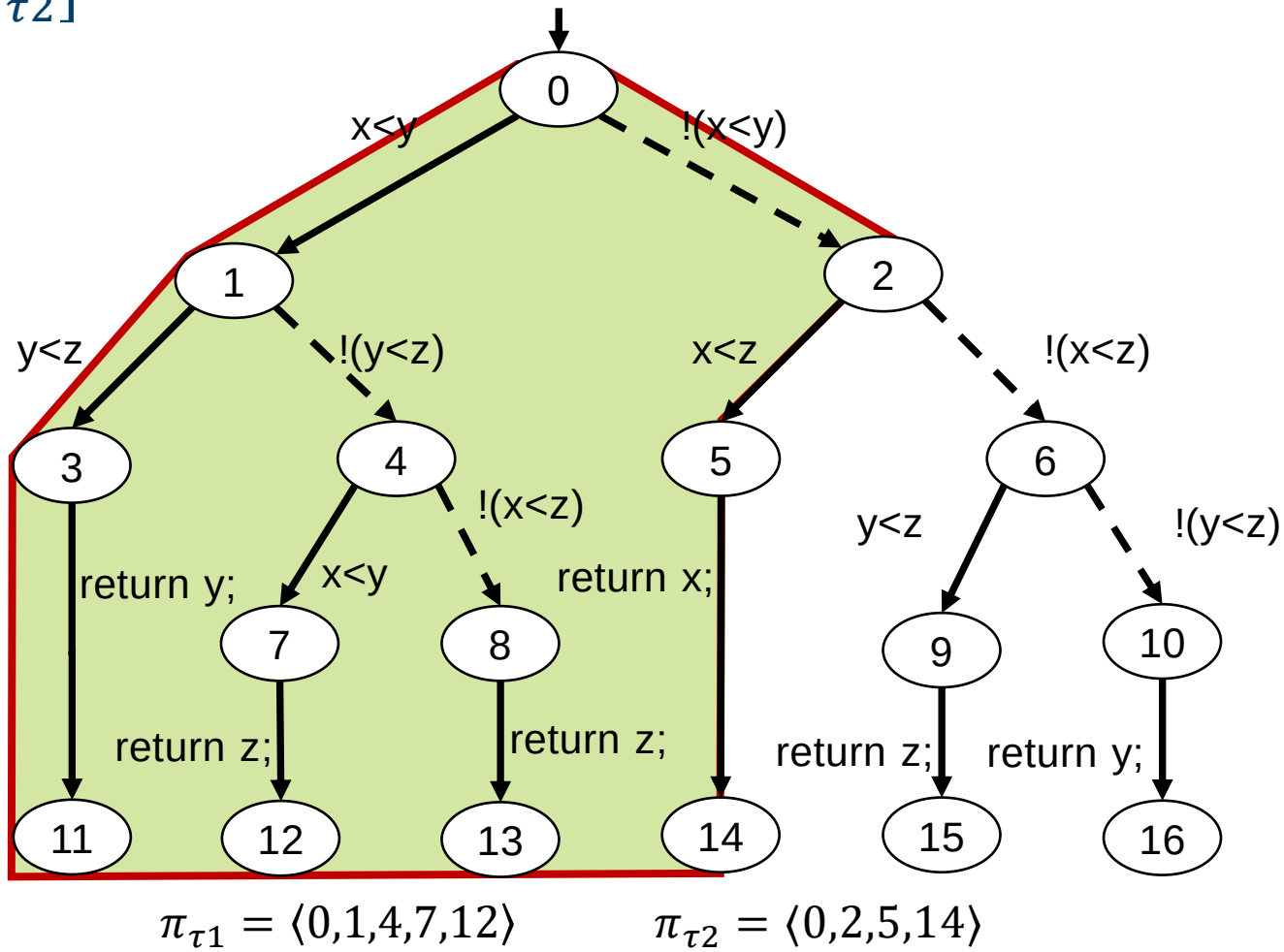
Cooperative Verification using Ranged Analysis
Jan Haltermann, Marie-Christine Jakobs, Cedric Richter and Heike Wehrheim

Carl von Ossietzky
Universität
Oldenburg

Backup

Ranged Analysis for $[\pi_{\tau 1}, \pi_{\tau 2}]$

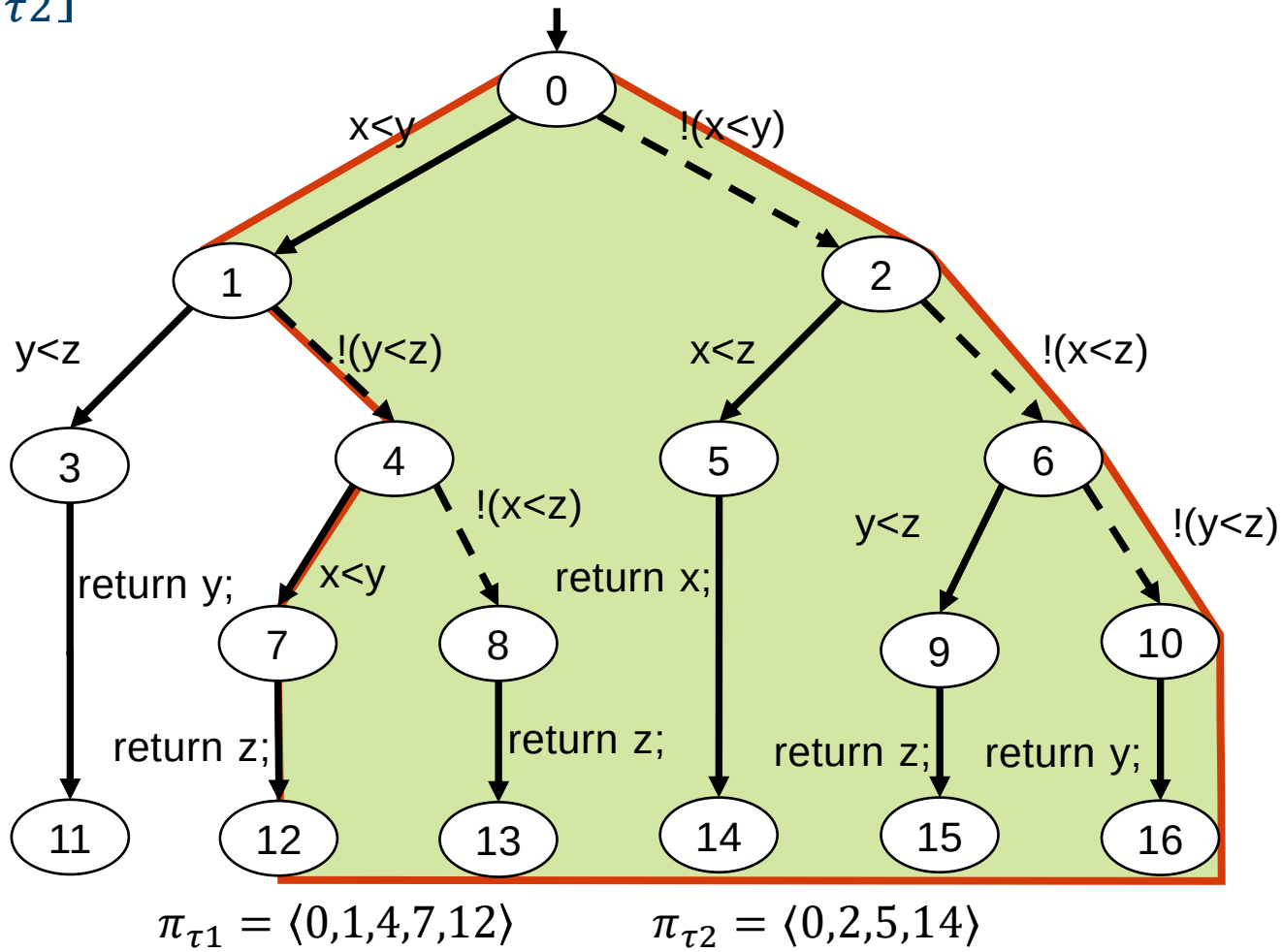




Ranged Analysis for $[\pi_{\tau 1}, \pi_{\tau 2}]$

Ranged Analysis

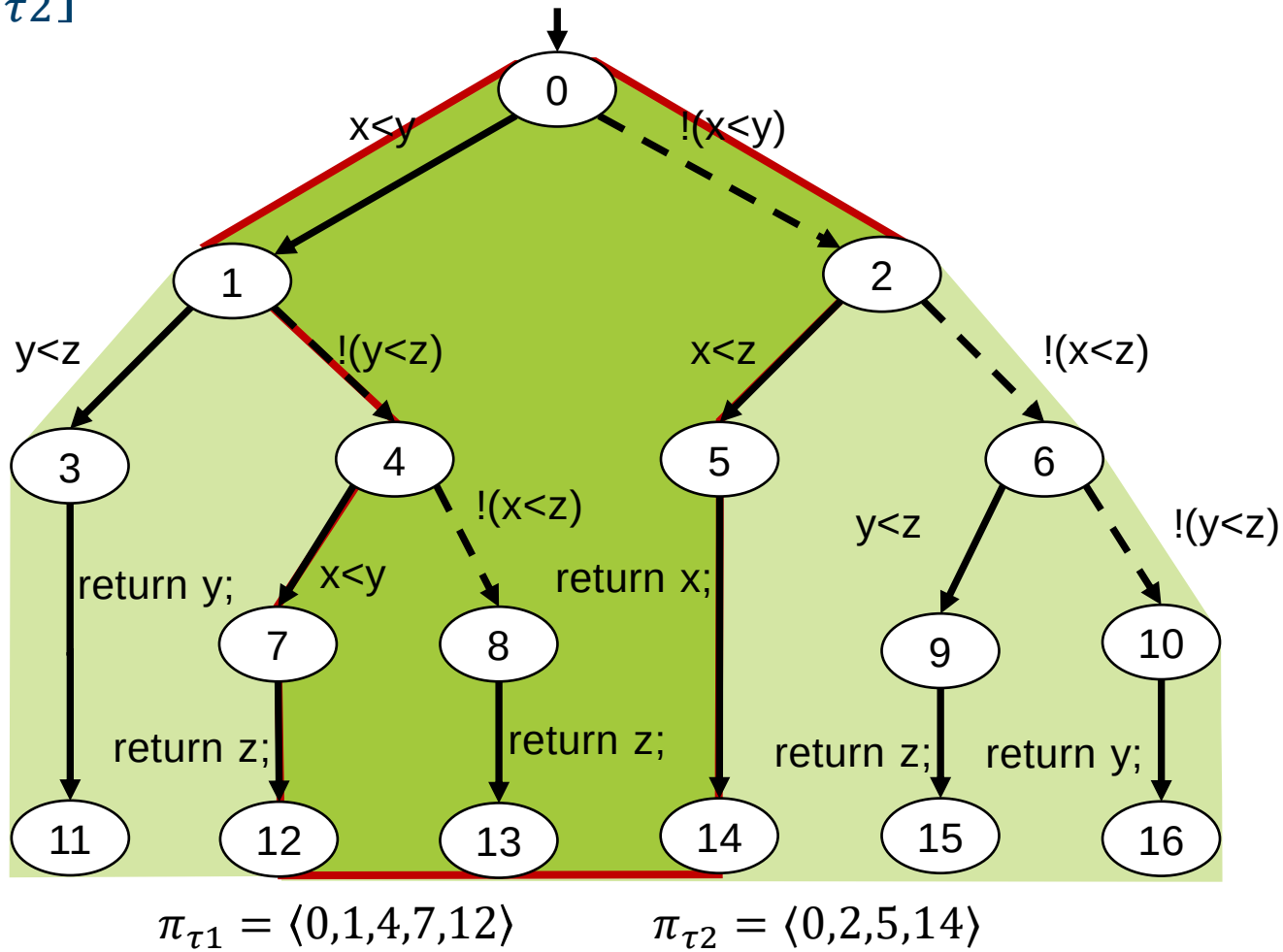
$\mathbb{R}_{[\pi_\perp, \pi_{\tau 2}]} \times \mathbb{A} \times \mathbb{R}_{[\pi_{\tau 1}, \pi_\top]}$



Ranged Analysis for $[\pi_{\tau 1}, \pi_{\tau 2}]$

Ranged Analysis

$\mathbb{R}_{[\pi_\perp, \pi_{\tau 2}]} \times \mathbb{A} \times \mathbb{R}_{[\pi_{\tau 1}, \pi_\top]}$

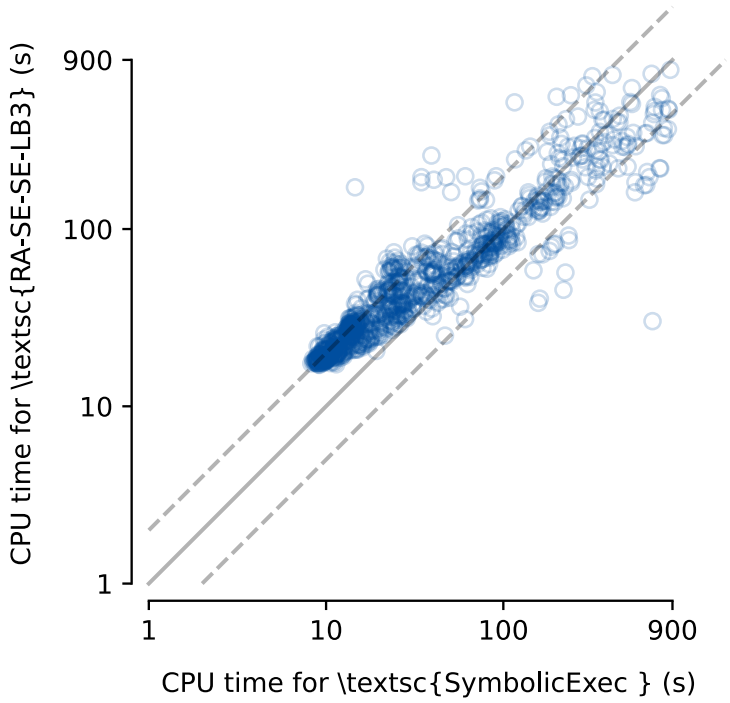


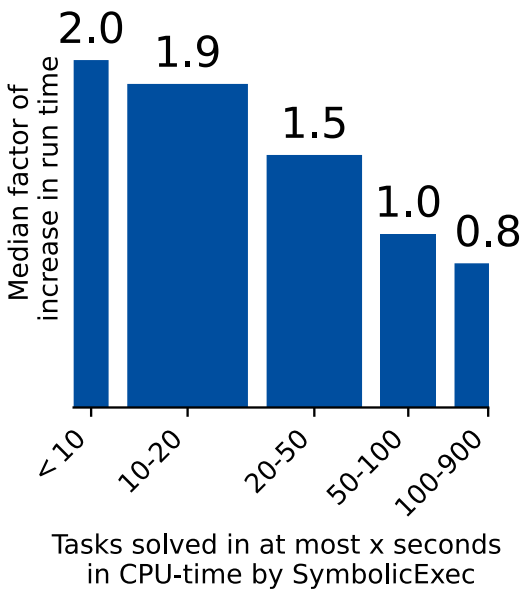
$\pi_{\tau 1} = \langle 0,1,4,7,12 \rangle$ $\pi_{\tau 2} = \langle 0,2,5,14 \rangle$

RQ1: Effect of Symbolic Analysis

	correct				incorrect	
	overall	**proof**	**alarm**	**add.**	**Proof**	**Alarm**
SymbExec	1386	565	821	-	1	31
Ra-2Se-Lb3	1487	566	921	116	1	36
Ra-2Se-Lb10	1464	567	897	92	1	37
Ra-2Se-Rdm	1464	569	895	97	1	39
Ra-2Se-Rdm9	1408	565	843	45	1	34
Ra-3Se-Lb	1532	563	969	181	1	35
Ra-3Se-Rdm	1505	565	940	160	1	55
Ra-3Se-Rdm9	1416	556	860	80	1	53

RQ1: Efficiency (CPU time)



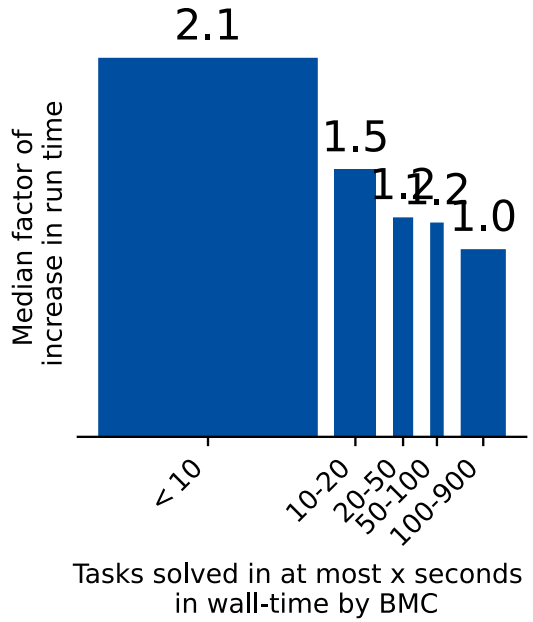


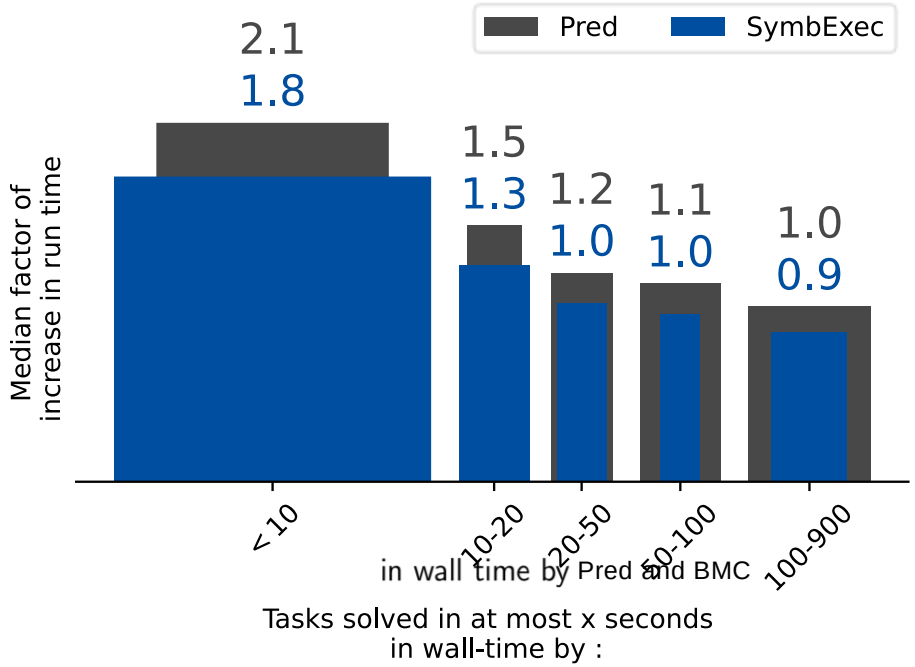
→ Reduced CPU time for complex or large tasks

RQ2: Effect for other (combination of) analyses

	correct				incorrect	
	overall	**proof**	**alarm**	**add.**	**Proof**	**Alarm**
BMC	2534	930	1604	-	0	68
RA-2BMC-LB3	2437	925	1512	36	0	68
RA-2BMC-LB10	2445	925	1519	43	0	70
RA-2BMC-RDM	2457	926	1532	48	0	69
RA-2BMC-RDM9	2505	932	1573	48	0	68
SYMBEXEC	1386	565	821	-	1	31
PRED	2254	1107	1147	-	0	38
RA-SE-PRED-LB3	2021	913	1108	30	0	40
RA-SE-PRED-LB10	2021	911	1110	36	0	38
RA-SE-PRED-RDM	1643	534	1109	27	1	40
RA-SE-PRED-RDM9	1795	715	1080	17	1	38

RQ2: Efficiency





→ Comparable real time for complex or large tasks