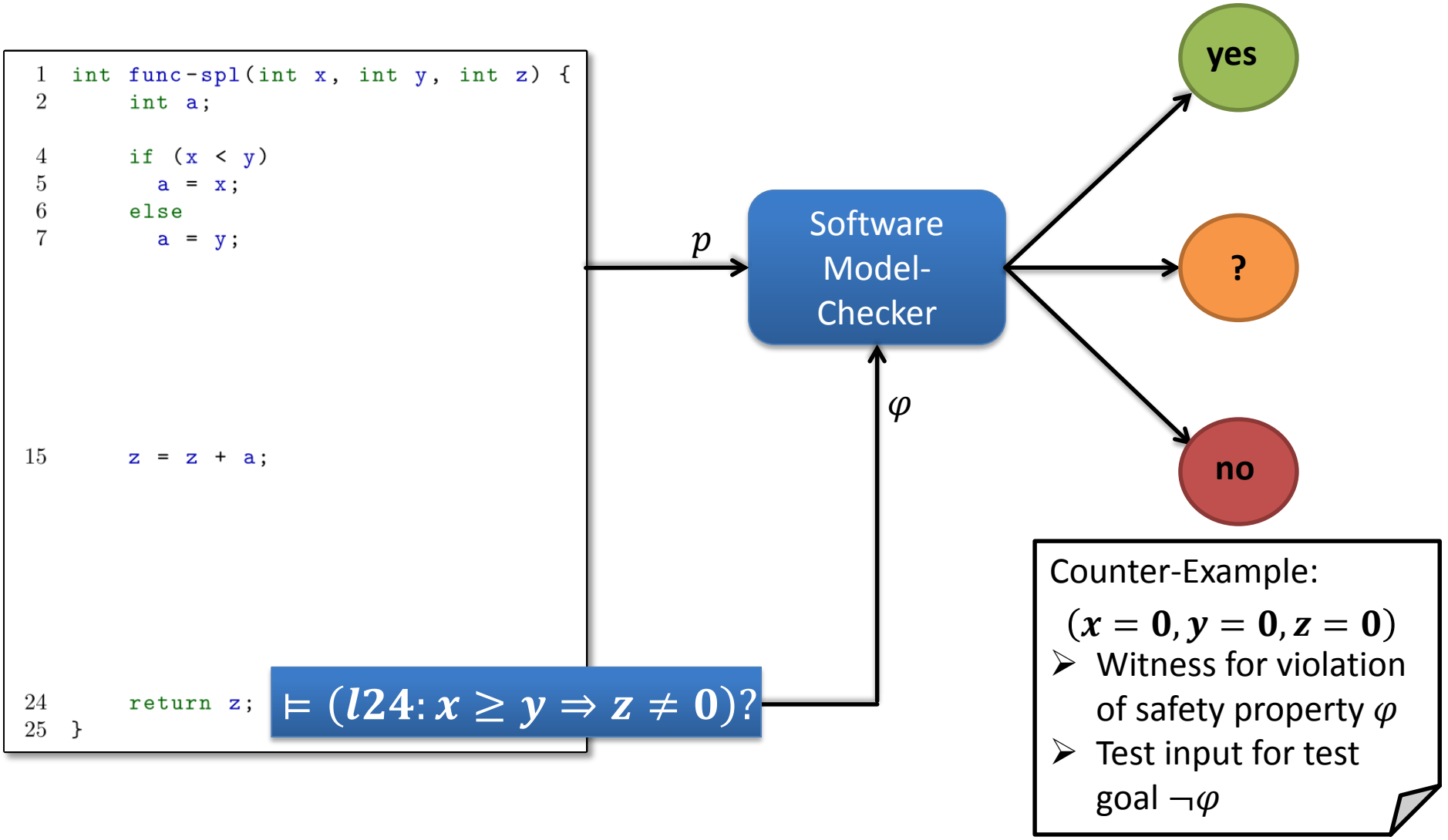
N-way Program Merging for Efficient Test Coverage of Configurable Software

Malte Lochau (TU Darmstadt)

(Joint Work with Dennis Reuling and Johannes Bürdek)

FOSD Meeting 2017

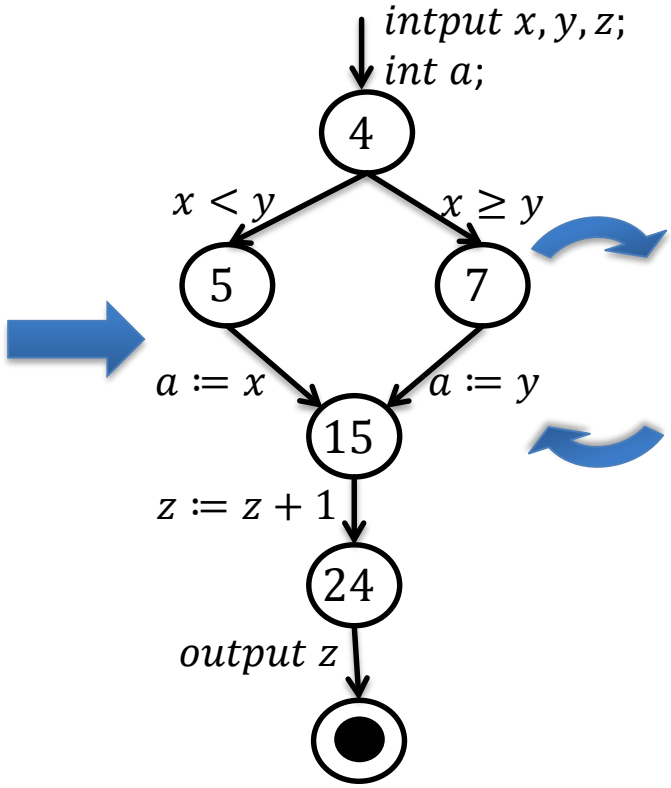
Program Analysis

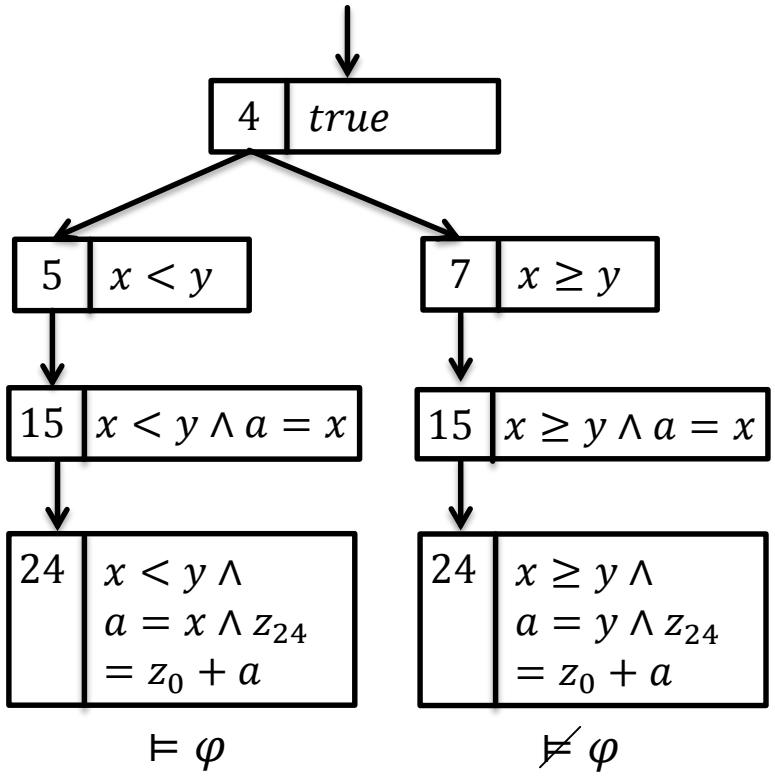


Counter-Example Guided Abstraction Refinement

[Clarke et al. 2004]

```
1  int func-spl(int x, int y, int z) {
2      int a;
4      if (x < y)
5        a = x;
6      else
7        a = y;
15     z = z + a;
24     return z;
25 }
```



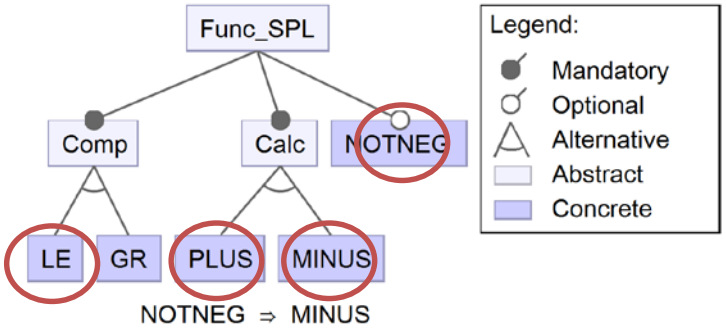


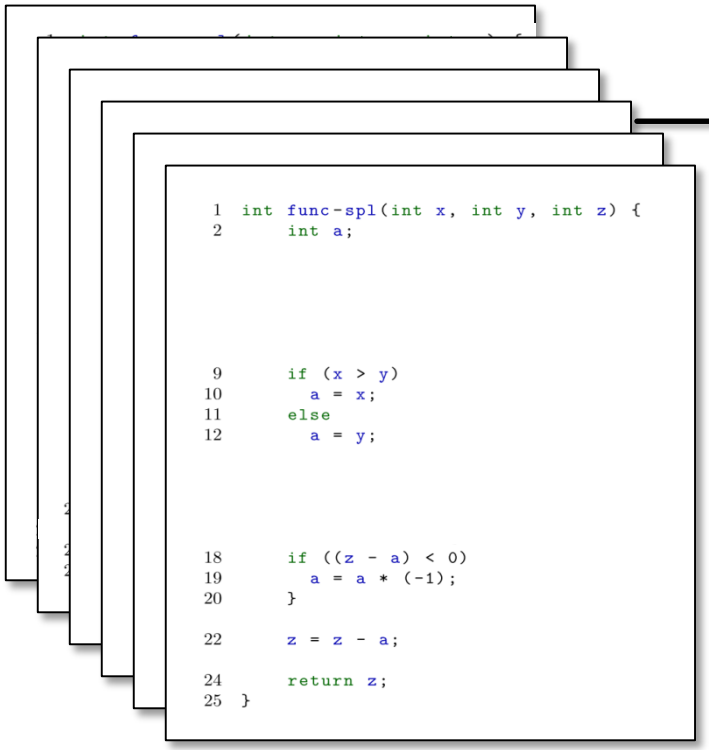
Program P

Control-Flow Automaton (CFA)

Abstract Reachability Graph (ARG)

Product-based Product-Line Analysis





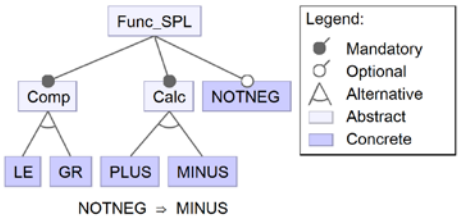
```
1  int func-spl(int x, int y, int z) {
2      int a;

9      if (x > y)
10       a = x;
11     else
12       a = y;

18     if ((z - a) < 0)
19       a = a * (-1);
20     }

22     z = z - a;

24     return z;
25 }
```

Software Model-Checker

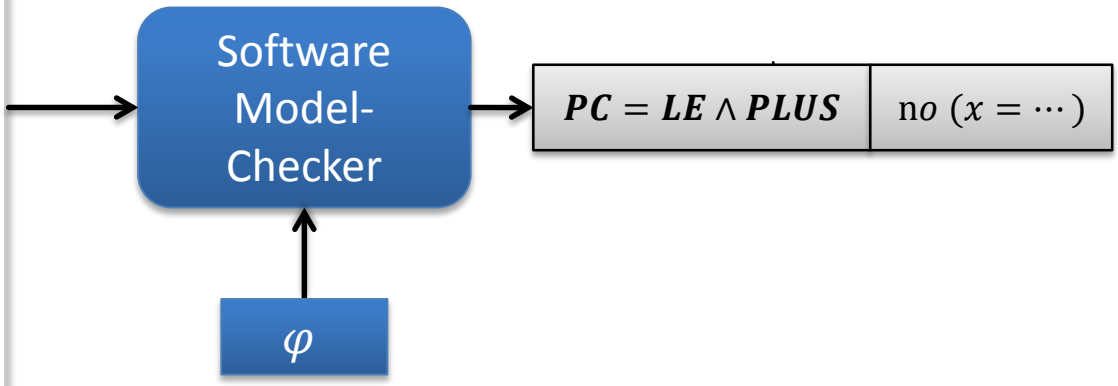
φ

P1	yes
P2	$no\ (x = \cdots)$
P3	$no\ (x = \cdots)$
P4	$no\ (x = \cdots)$
P5	$no\ (x = \cdots)$
P6	yes

Family-based Product-Line Analysis



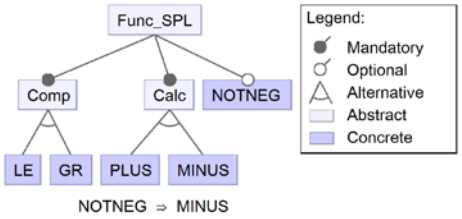

```
1  int func-spl(int x, int y, int z) {
2      int a;
3      #ifdef LE
4      if (x < y)
5        a = x;
6      else
7        a = y;
8      #elsif GR
9      if (x > y)
10       a = x;
11     else
12       a = y;
13     #endif
14     #ifdef PLUS
15     z = z + a;
16     #elsif MINUS
17     #ifdef NOTNEG {
18     if ((z - a) < 0)
19       a = a * (-1);
20     }
21     #endif
22     z = z - a;
23     #endif
24     return z;
25 }
```



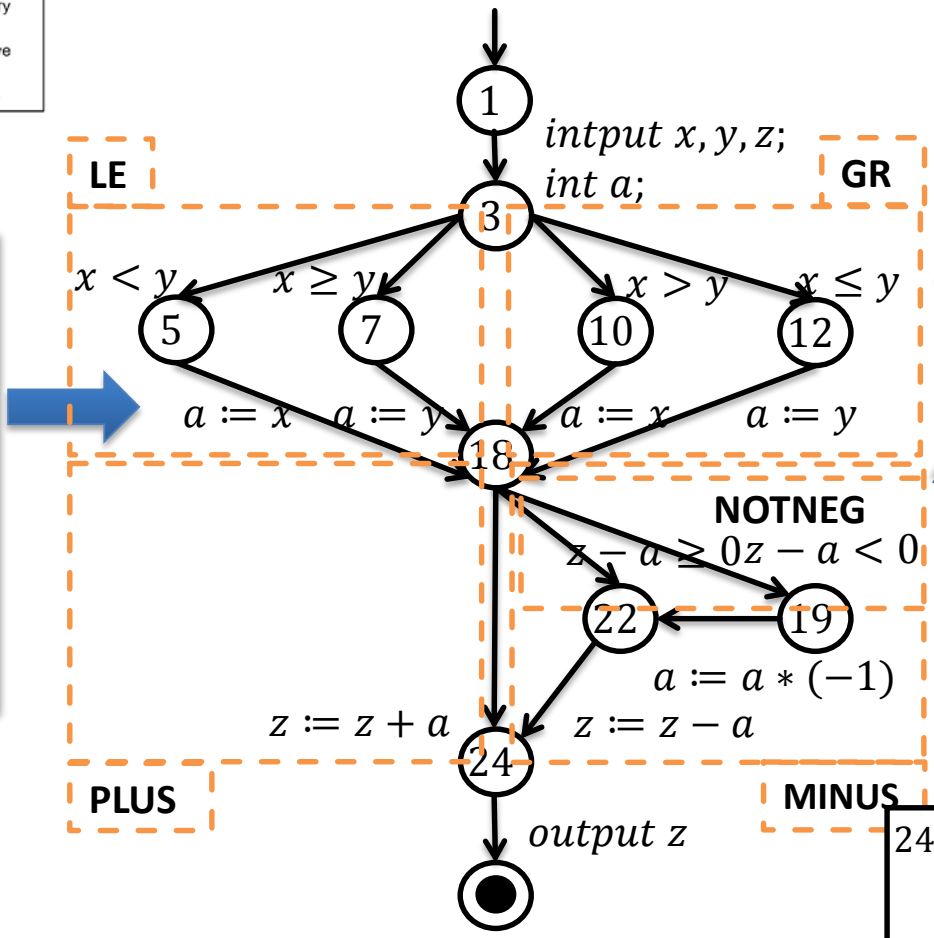
[Apel et al., 2013]

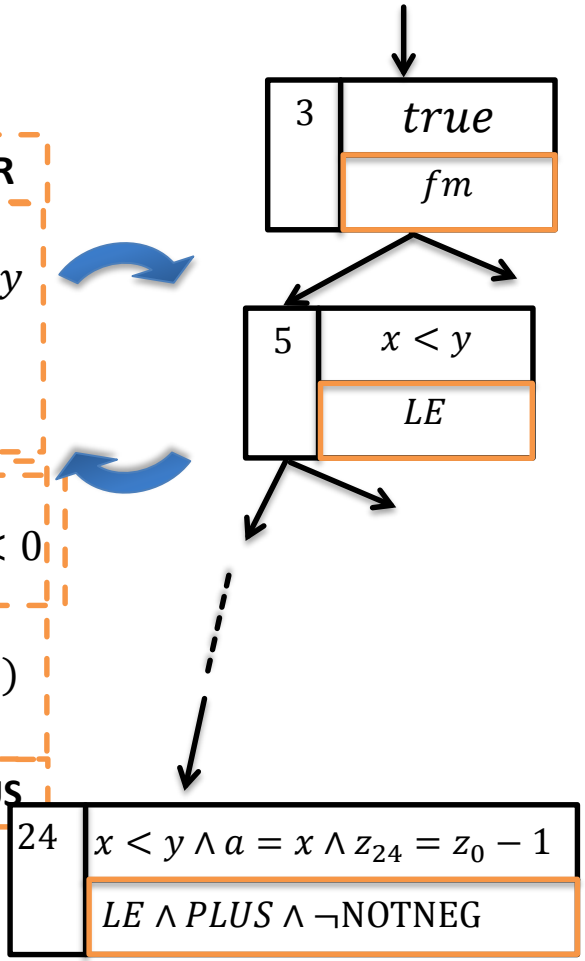
[Bürdek et al., 2015]

Family-based Product-Line Analysis




```
1  int func-spl(int x, int y, int z) {
2      int a;
3      #ifdef LE
4      if (x < y)
5        a = x;
6      else
7        a = y;
8      #elsif GR
9      if (x > y)
10       a = x;
11     else
12       a = y;
13     #endif
14     #ifdef PLUS
15     z = z + a;
16     #elsif MINUS
17     #ifdef NOTNEG {
18     if ((z - a) < 0)
19       a = a * (-1);
20     }
21     #endif
22     z = z - a;
23     #endif
24     return z;
25 }
```





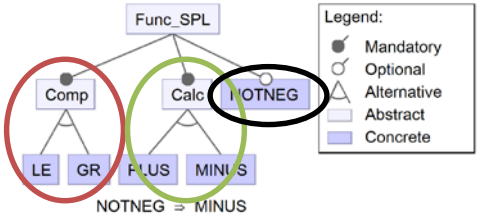
SPL Implementation **Superimposed CFA** **Featured ARG**

Challenges

- Scalability / Precision Trade-offs
- Partial / Incomplete / Evolving SPL Implementations

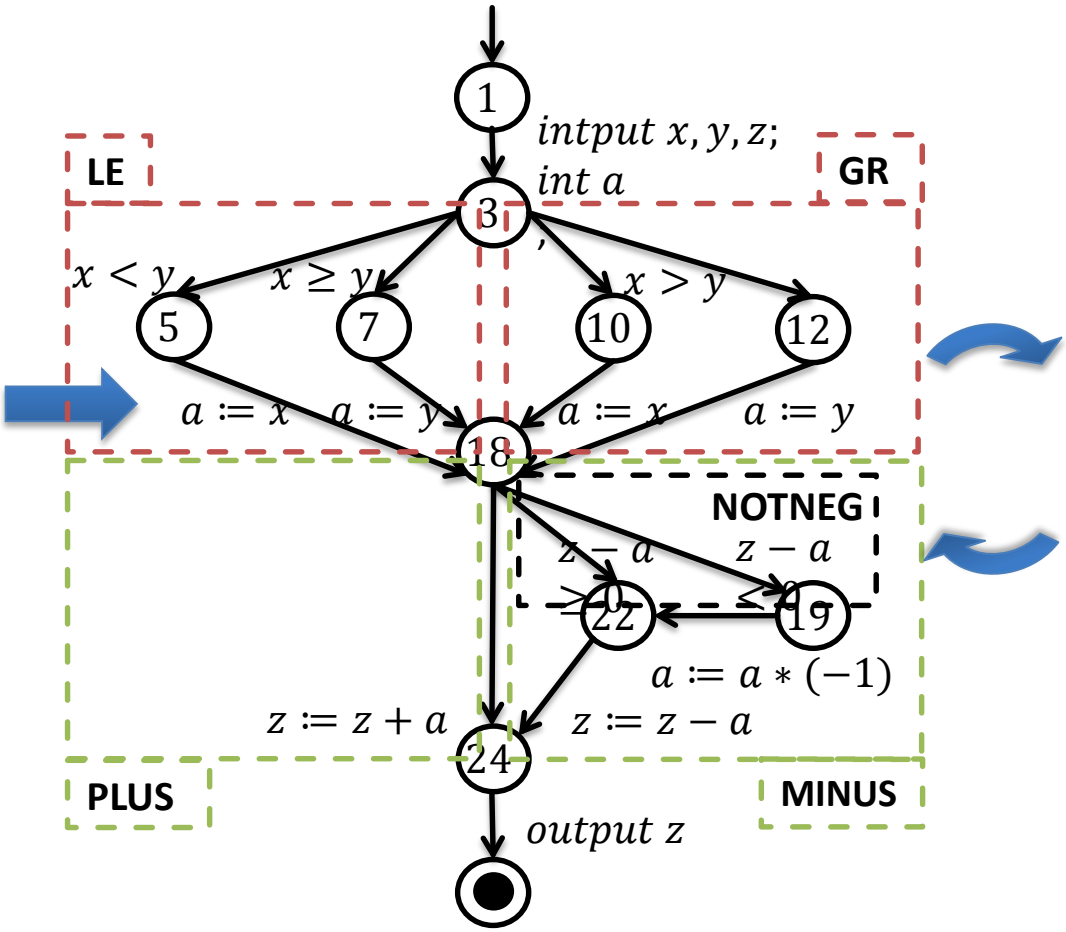
⇨ Product/Family-based SPL Analysis

Product/Family-based Analysis




```
1  int func-spl(int x, int y, int z) {
2      int a;
3      #ifdef LE
4      if (x < y)
5        a = x;
6      else
7        a = y;
8      #elsif GR
9      if (x > y)
10       a = x;
11     else
12       a = y;
13     #endif
14     #ifdef PLUS
15     z = z + a;
16     #elsif MINUS
17     #ifdef NOTNEG {
18     if ((z - a) < 0)
19       a = a * (-1);
20     }
21     #endif
22     z = z - a;
23     #endif
24     return z;
25 }
```

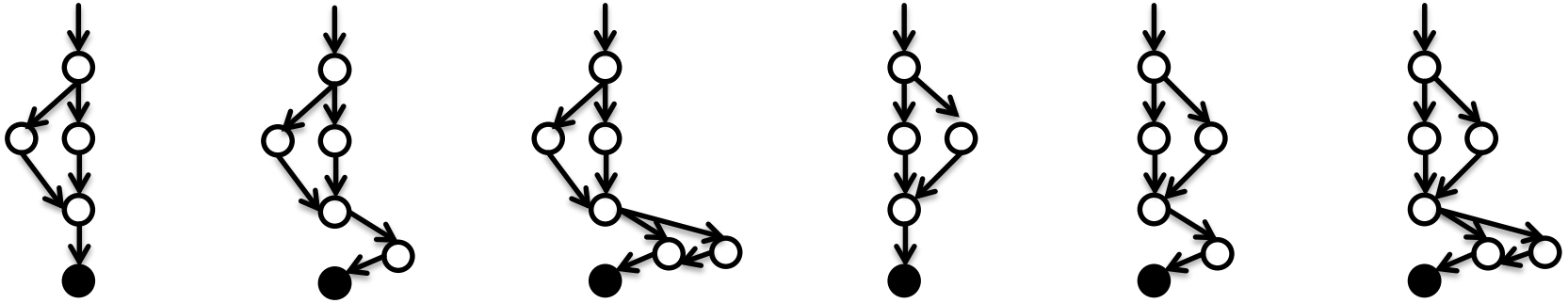
Partial/Evolving/Incomplete SPL Implementation Artifacts



(Continuously) Superimposed CFA

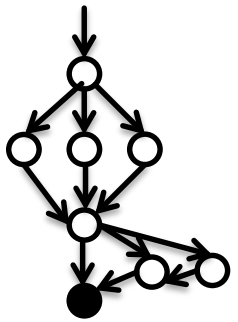
N-Way Model-Merging

- Given: N input models M_i



CFA_1 CFA_2 CFA_3 CFA_4 CFA_5 CFA_6

➡ Find: a <u>correct</u> and <u>good</u> family model M

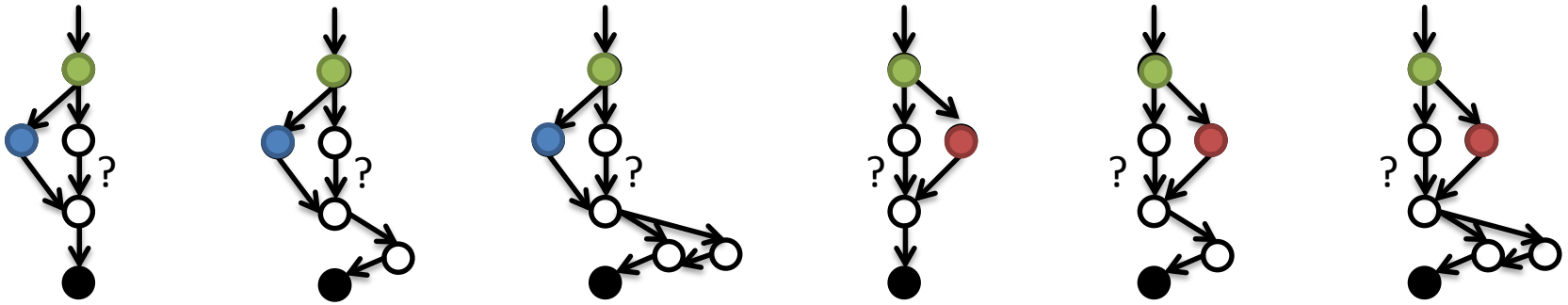


$Superimposed\ CFA$

Compare, …

[Rubin and Chechik, 2013]

- N-tuple t $= (e_1, \ldots, e_k) \in T, 1 \leq k \leq N,$ such that no two elements belong to the same input model

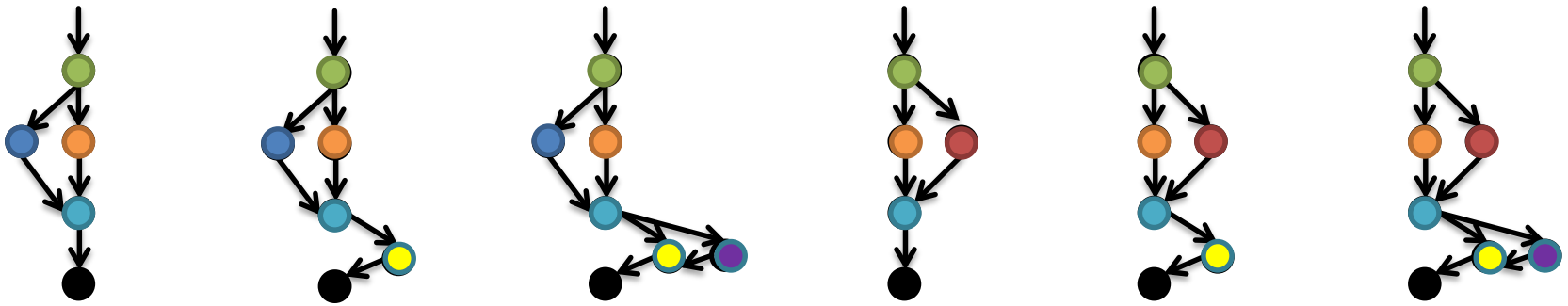


- $compare : T \rightarrow [0,1]$

…, Match, …

[Rubin and Chechik, 2013]

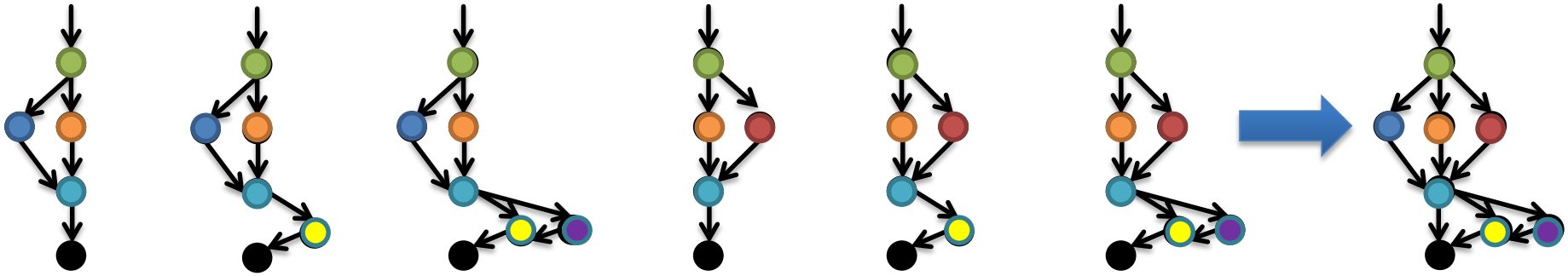
- Subset $T' \subseteq T$ is a (complete) *match* iff each model element occurs in exactly one N-tuple $t \in T'$
- Match T' is *minimal* iff for every match T'' it holds that $\sum_{t' \in T'} compare(t') \geq \sum_{t'' \in T''} compare(t'')$



… and Merge

[Rubin and Chechik, 2013]

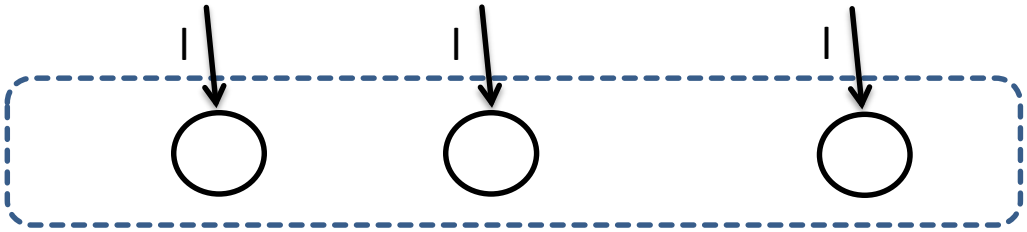
- Compose N models into one by integrating matched elements



Compare/Match/Merge has Problems

- Three **hard problems** to be solved at once
  1. Enumerate N-tuples: „combinatorial explosion“
  2. Compare locations: „path explosion“
  3. Find minimal match: „knapsack problem“
- **Configuration information** is not preserved
- Match/merge may produce ill-formed models as model elements are **untyped**
- Metrics for measuring **quality** of merging results?

Location Prepartitioning

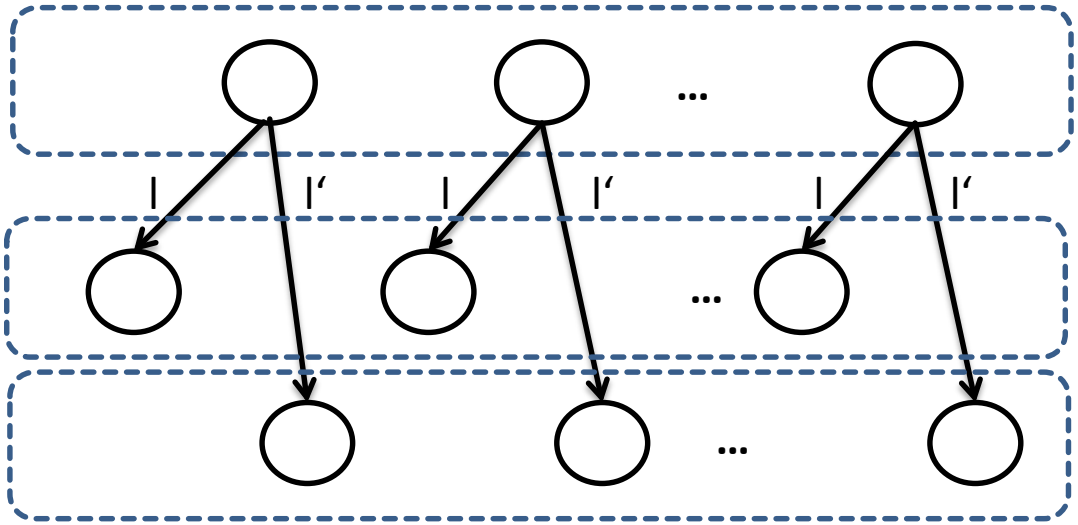


Group similar locations with…

- same type (initial, termination, sequence, branch, …)
- at least one equally labeled incoming edge
- similar block nestings

=> Parameter: **Minimum Group Size (MGS)**

Incremental Location Matching

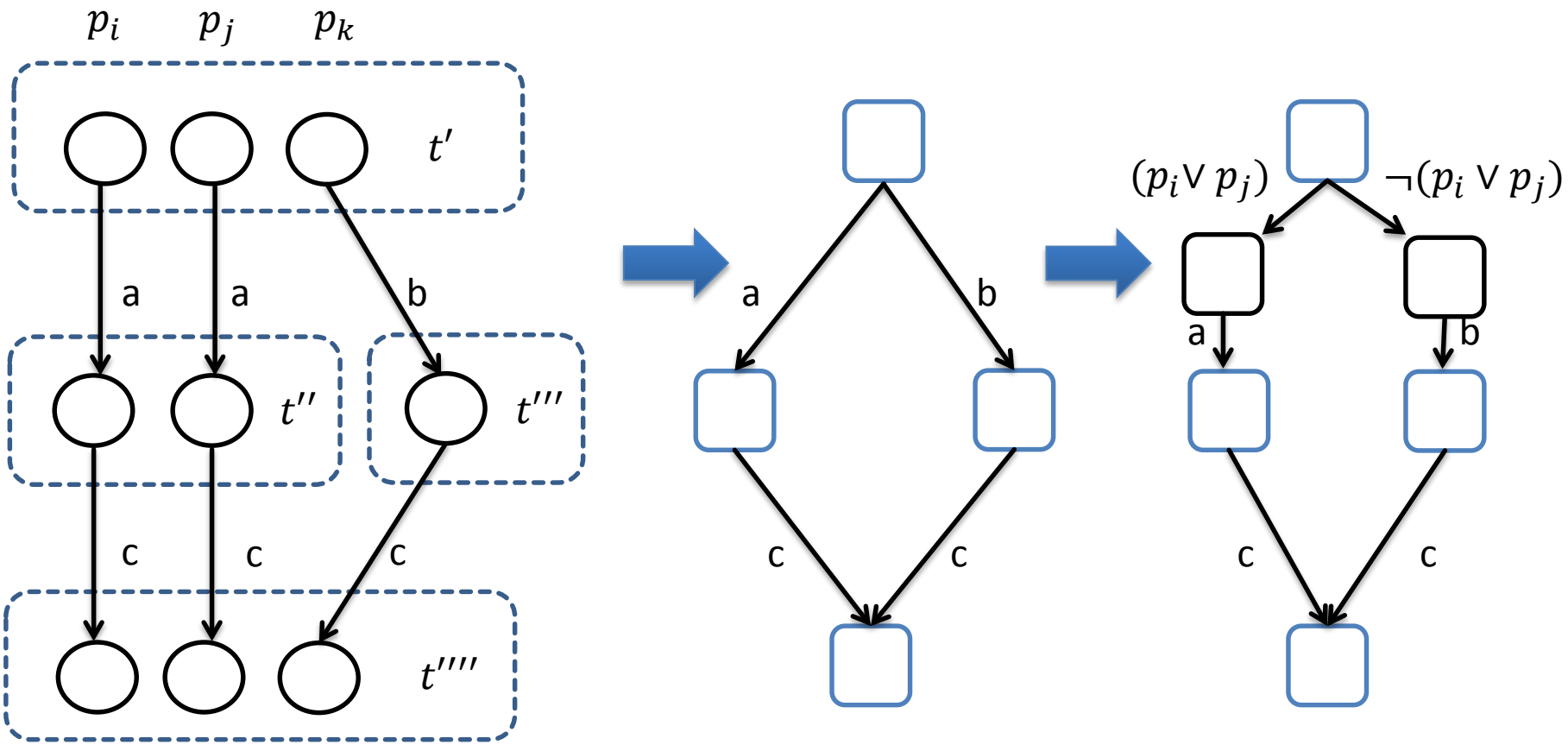


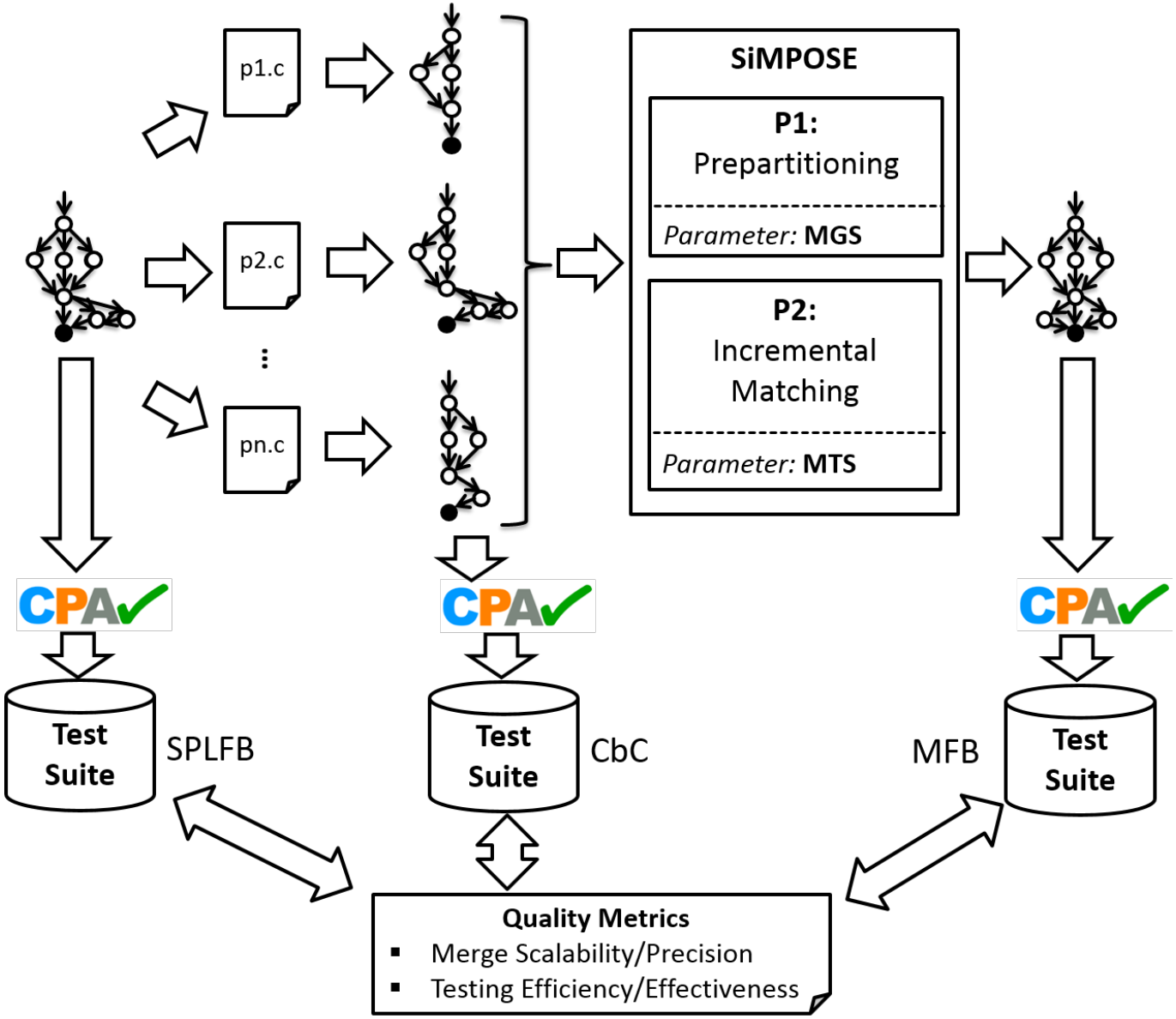
Match grouped locations

- Combination of depth-first and bredth-first traversal
- Interleaved with randomized matching attempts

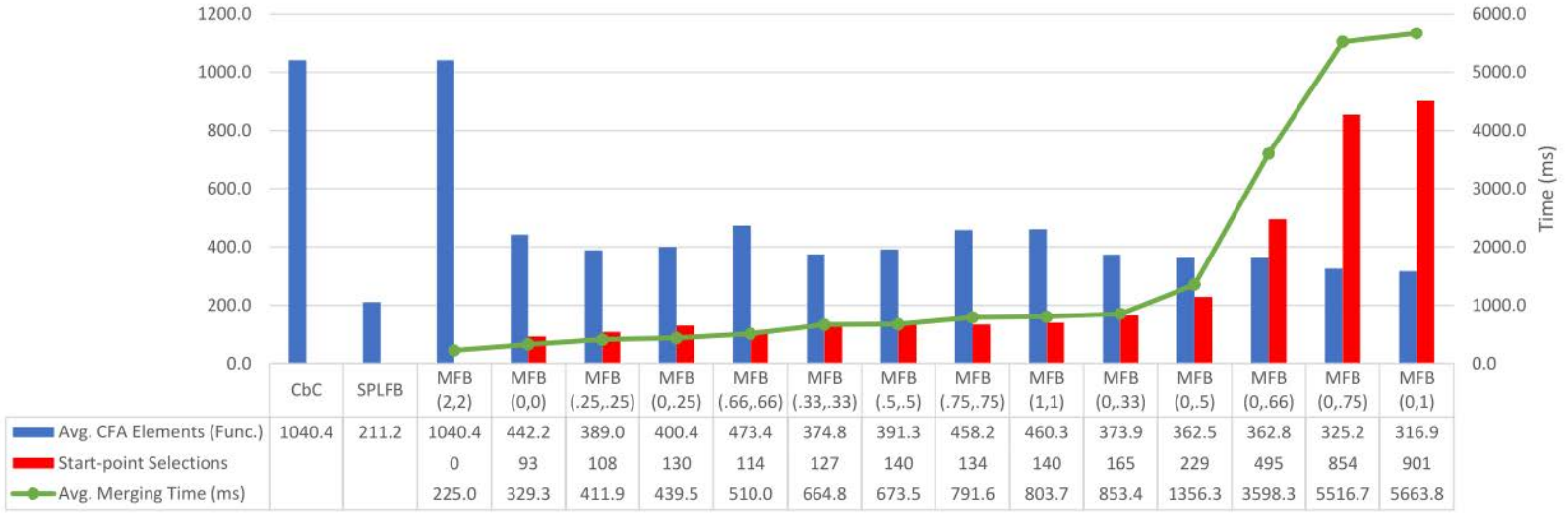
=> Parameter: **Minimum Tuple Size (MTS)**

Merging with Variability Encoding




p1.c
p2.c
pn.c
SiMPOSE
P1:
Prepartitioning
Parameter: MGS
P2:
Incremental
Matching
Parameter: MTS
CPA
CPA
CPA
Test Suite
Test Suite
Test Suite
SPLFB
CbC
MFB
Quality Metrics
Merge Scalability/Precision
Testing Efficiency/Effectiveness

Evaluation Results for BusyBox



	CbC	SPLFB	MFB (2,2)	MFB (0,0)	MFB (.25,.25)	MFB (0,.25)	MFB (.66,.66)	MFB (.33,.33)	MFB (.5,.5)	MFB (.75,.75)	MFB (1,1)	MFB (0,.33)	MFB (0,.5)	MFB (0,.66)	MFB (0,.75)	MFB (0,1)
Avg. CFA Elements (Func.)	1040.4	211.2	1040.4	442.2	389.0	400.4	473.4	374.8	391.3	458.2	460.3	373.9	362.5	362.8	325.2	316.9
Start-point Selections			0	93	108	130	114	127	140	134	140	165	229	495	854	901
Avg. Merging Time (ms)			225.0	329.3	411.9	439.5	510.0	664.8	673.5	791.6	803.7	853.4	1356.3	3598.3	5516.7	5663.8

MFB /	Model Size Reduction Factor	CPU Time Speedup Factor	Test Suite Size Reduction Factor
CbC	2.2 – 3.3	1.3 – 3.3	1.3 – 2.8
SPLFB	0.5 – 0.7	0.5 – 0.8	0.2 – 0.5

Thank You!