

Rekurzív programok absztrakció alapú ellenőrzésének támogatása

Formális módszerek



Előadó: Somorjai Márk, MSc hallgató

Témavezető: Dobos-Kovács Mihály, doktorandusz

Villamosmérnöki és Informatikai Kar

Méréstechnika és Információs Rendszerek Tanszék



Formal Verification

- Safety Critical Systems
 - correctness is crucial, e.g. planes, nuclear reactors, cars
- Systems are more and more complex
- Testing cannot be exhaustive
- Formal verification: prove certain properties of software, e.g. safety
 - ✓ mathematical proof
 - ✗ refutation

Abstraction-Based Software Verification

Program

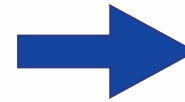
locations
statements



CFA

(Control Flow Automaton)

nodes
edges

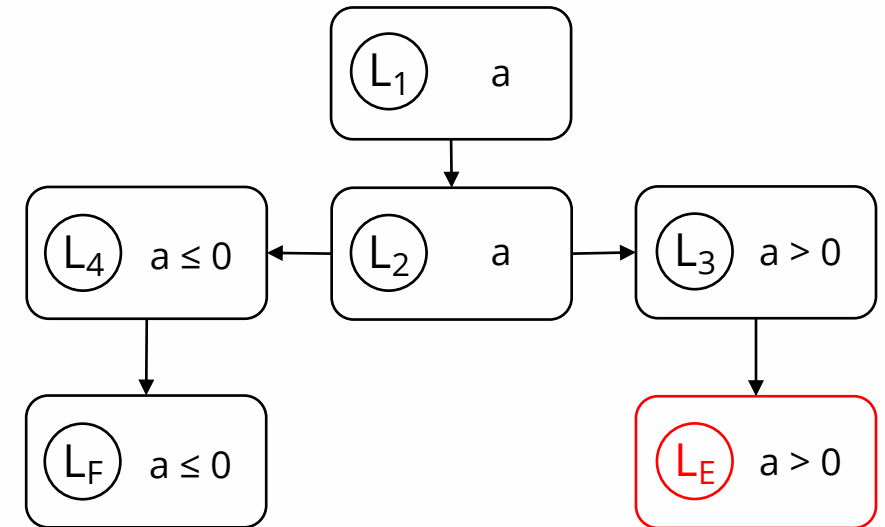
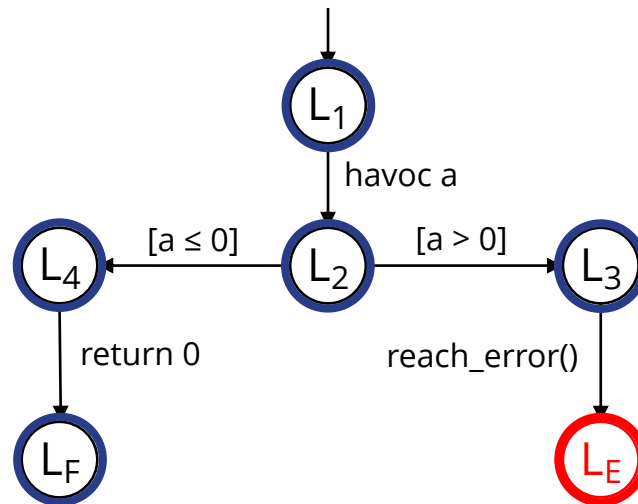


ARG

(Abstract Reachability Graph)

abstract states
transitions

```
int main() {  
  int a = input();  
  if (a > 0) {  
    reach_error();  
  }  
  return 0;  
}
```



Interprocedural Verification

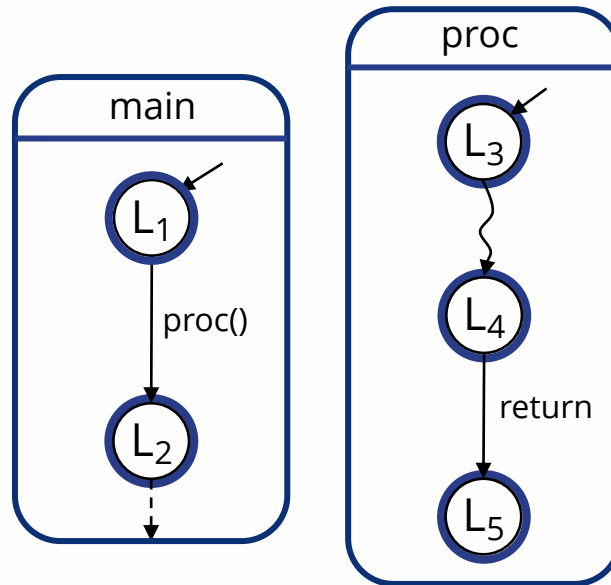
- Verification of programs with procedures

procedure

```
int main() {  
    proc();  
    ...  
}  
  
void proc() {  
    ...  
    return;  
}
```

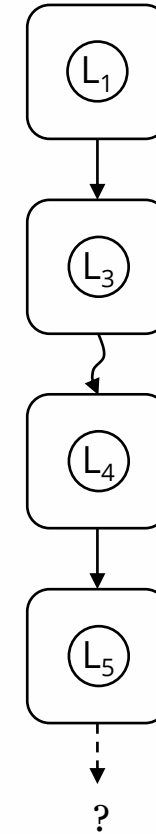


CFA



- Location stack is part of the program's state

ARG

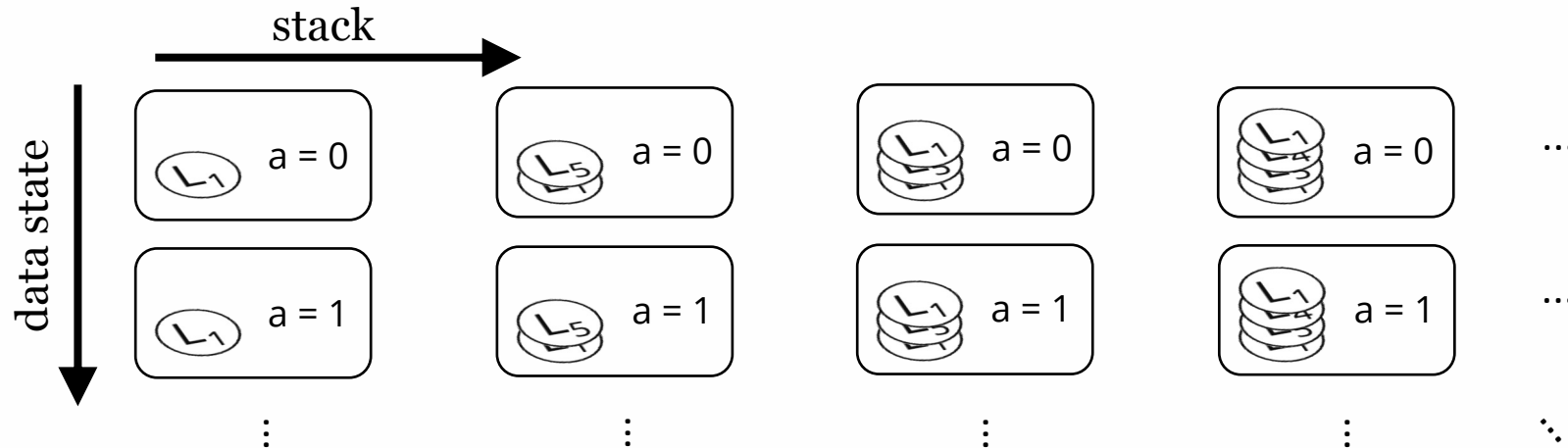


with stacks



Abstraction-Based Interprocedural Verification

- Location stacks leads to further state-space explosion



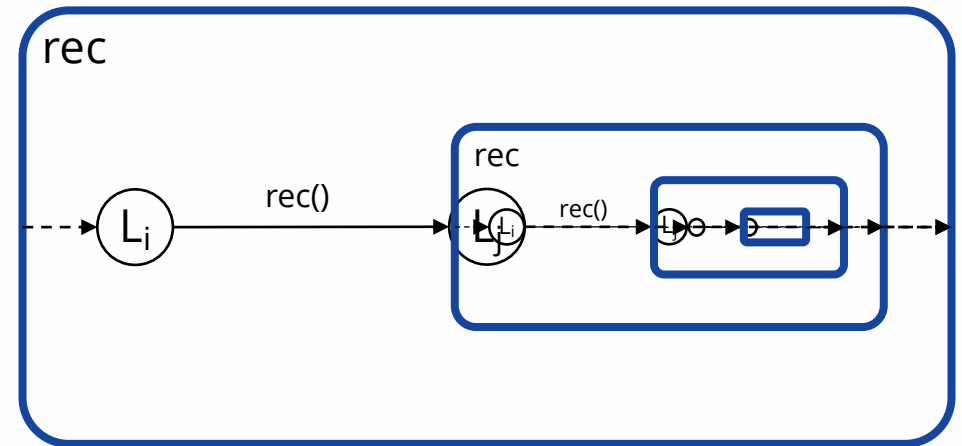
My work:

Reduce the abstract state-space of procedural programs by **applying abstraction to location stacks** in order to **improve the efficiency of verification**.

Procedure Inlining

- Procedure call is replaced with the CFA of the procedure
 - input parameter assignments before the call
 - output parameter assignments after the call

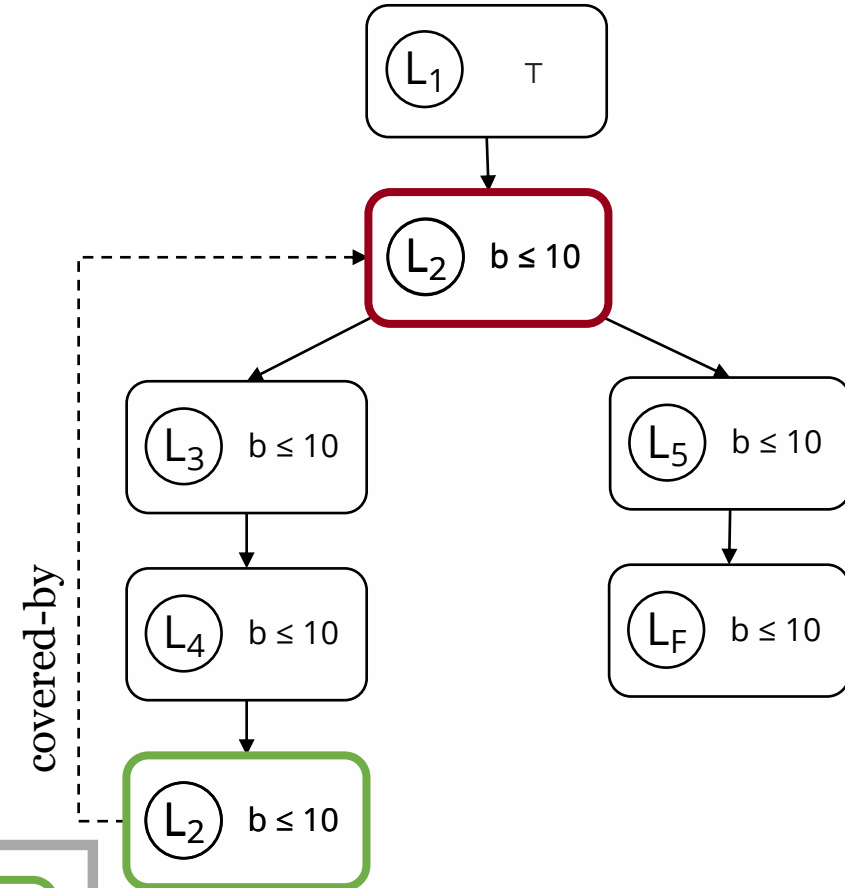
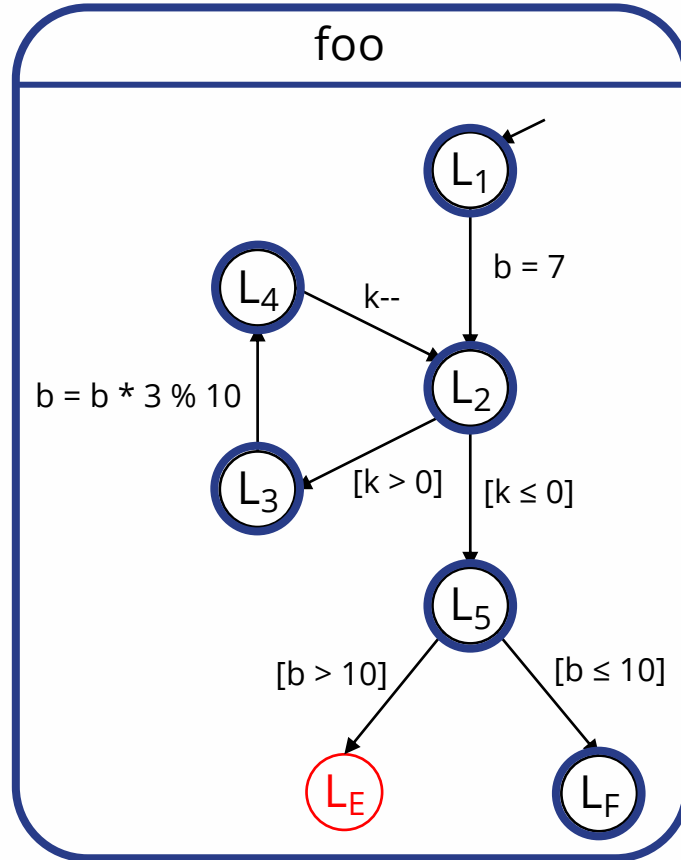
- Problems:
 - larger CFA
 - cannot handle recursive procedures

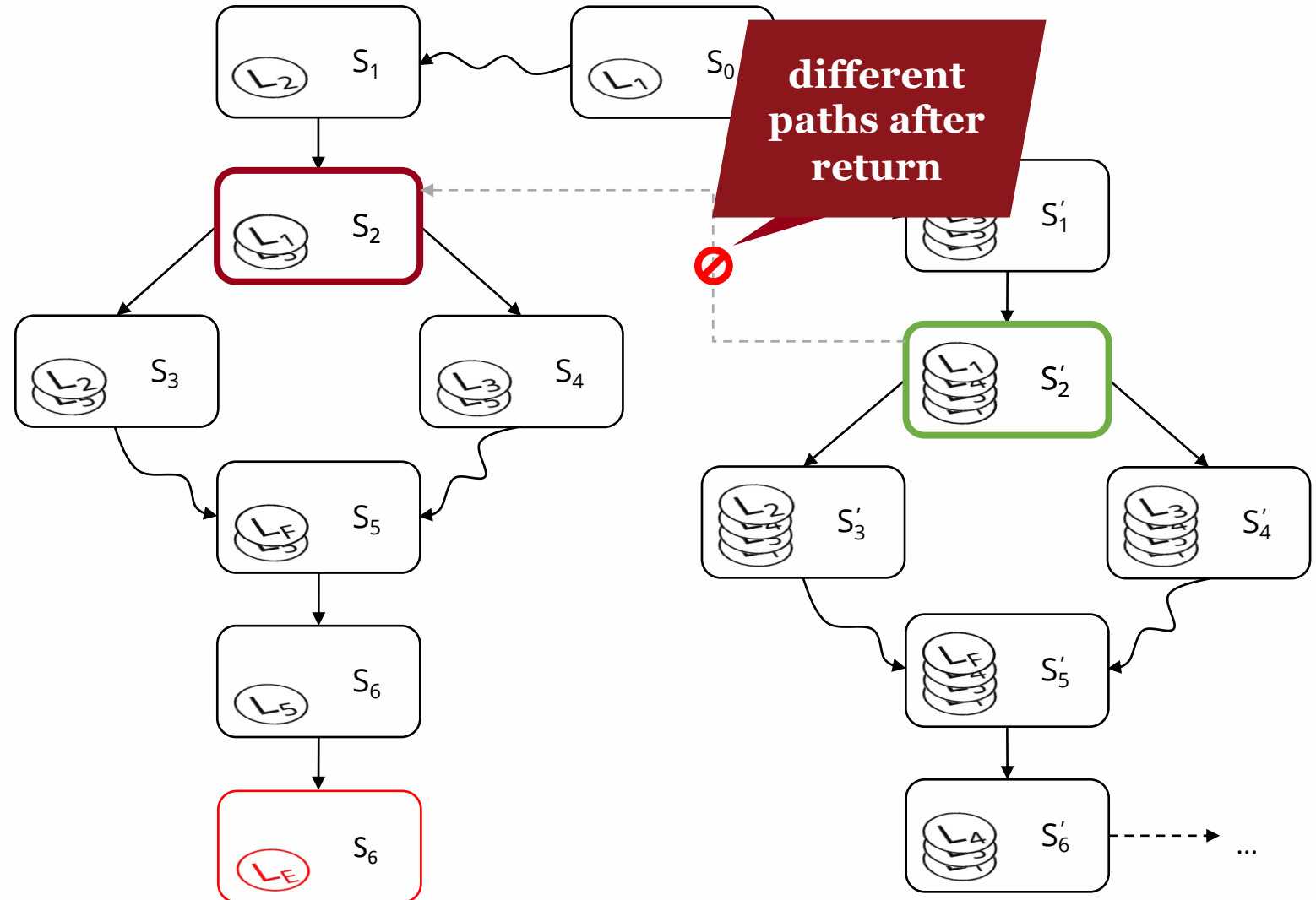


- Verifying recursive programs **requires** a more nuanced approach

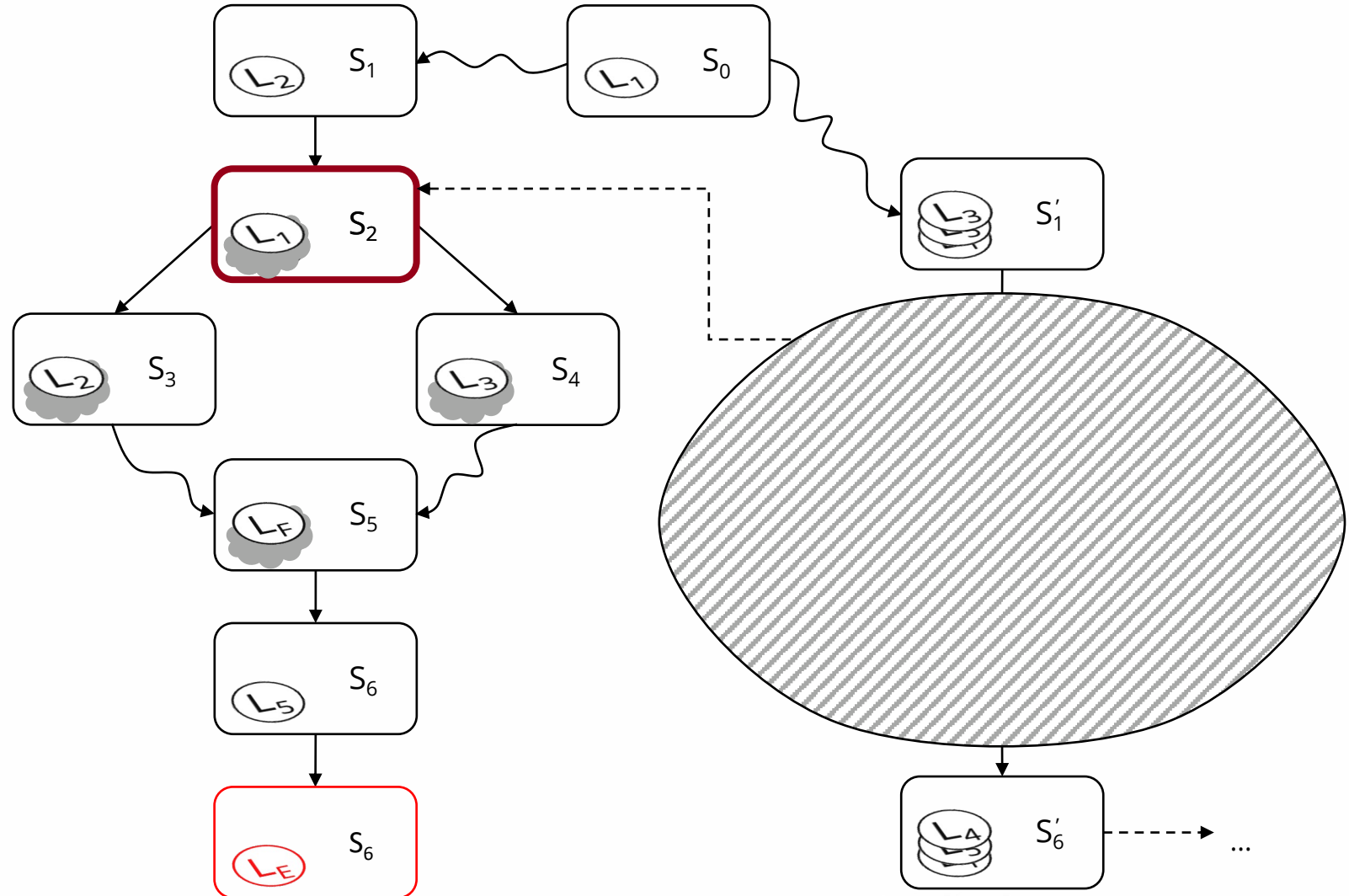
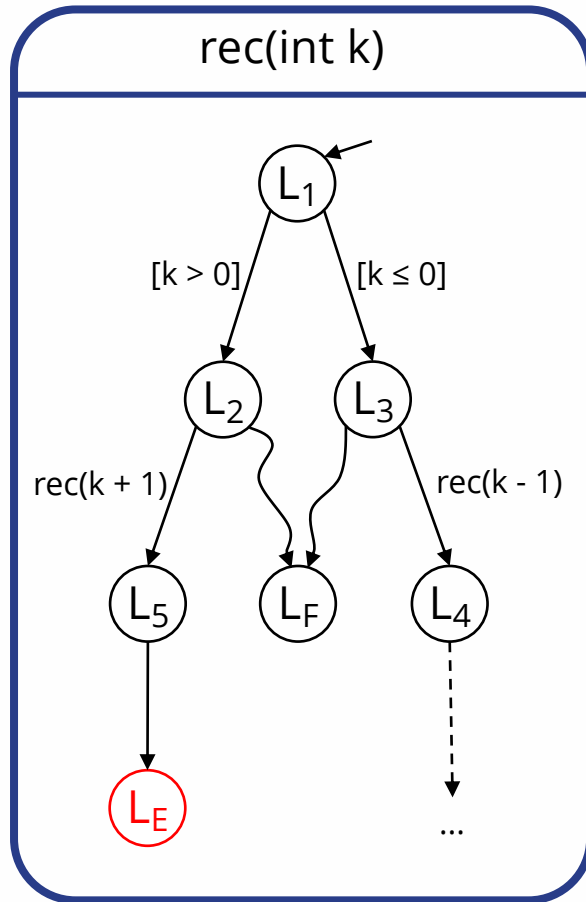
Covering Abstract States

```
void foo(int k) {
  int b = 7;
  while (k > 0) {
    b = b * 3 % 10;
    k--;
  }
  assert(b ≤ 10);
}
```





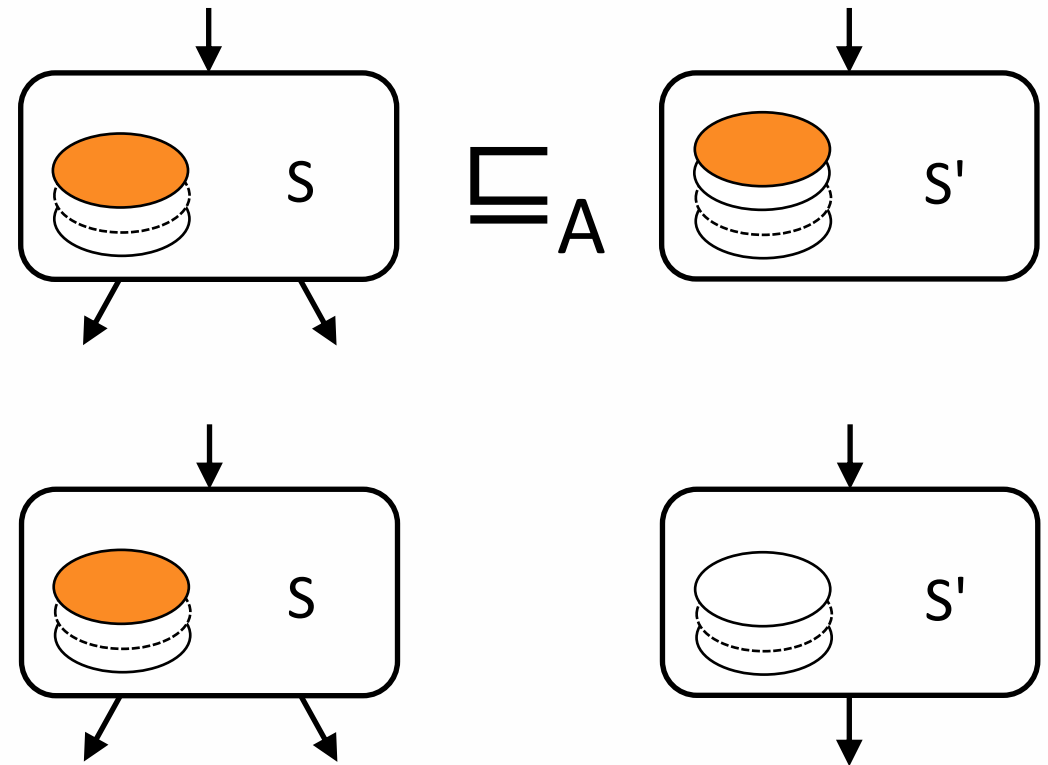
Applying Abstraction to Stacks



Stack Abstraction

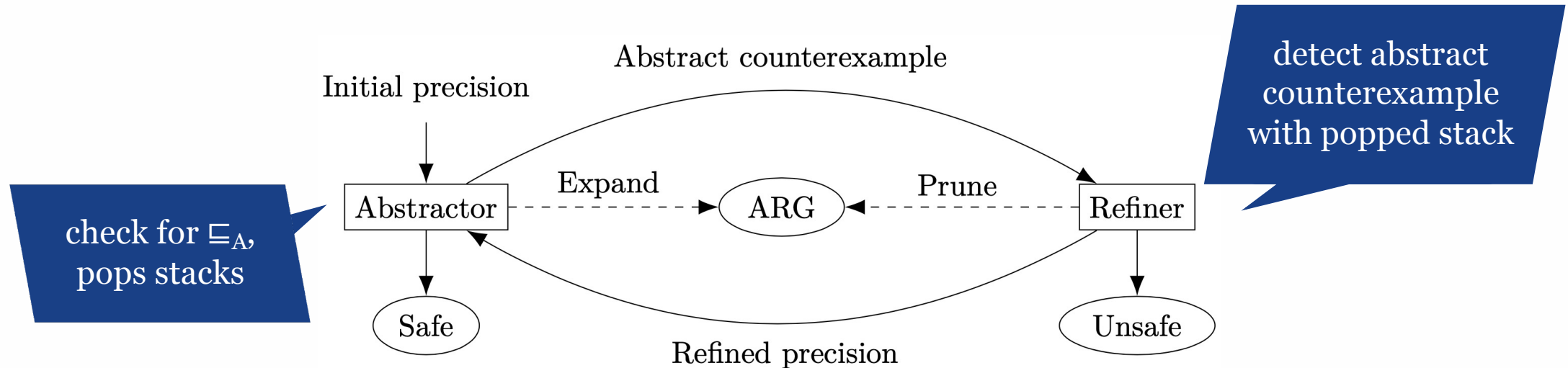
$$(q, S) \sqsubseteq_A (q', S') \iff \text{top}(q) = \text{top}(q') \wedge S \sqsubseteq S'$$

1. Check $(q, S) \sqsubseteq_A (q', S')$:
 - top locations are the same
 - S overapproximates S'
2. If $(q, S) \sqsubseteq_A (q', S')$ found:
 - pop top location of q'
 - continue exploration



Updated Verification Algorithm

- Counterexample-Guided Abstraction Refinement (CEGAR)



- Advantages:
 - reduced state space $\longrightarrow$ more efficient verification
 - can solve some infinitely recursive tasks

Evaluation

- Implementation in Theta¹
 - open-source model-checking framework
 - implementation was merged²
- 1219 C programs from the SV-COMP benchmark repository³
 - tasks without procedures to measure overhead
 - recursive tasks to measure performance gains
- Many configurations tested, using BenchExec by SoSyLab
- No wrong answers, # solved tasks (15-minute timeout) is shown

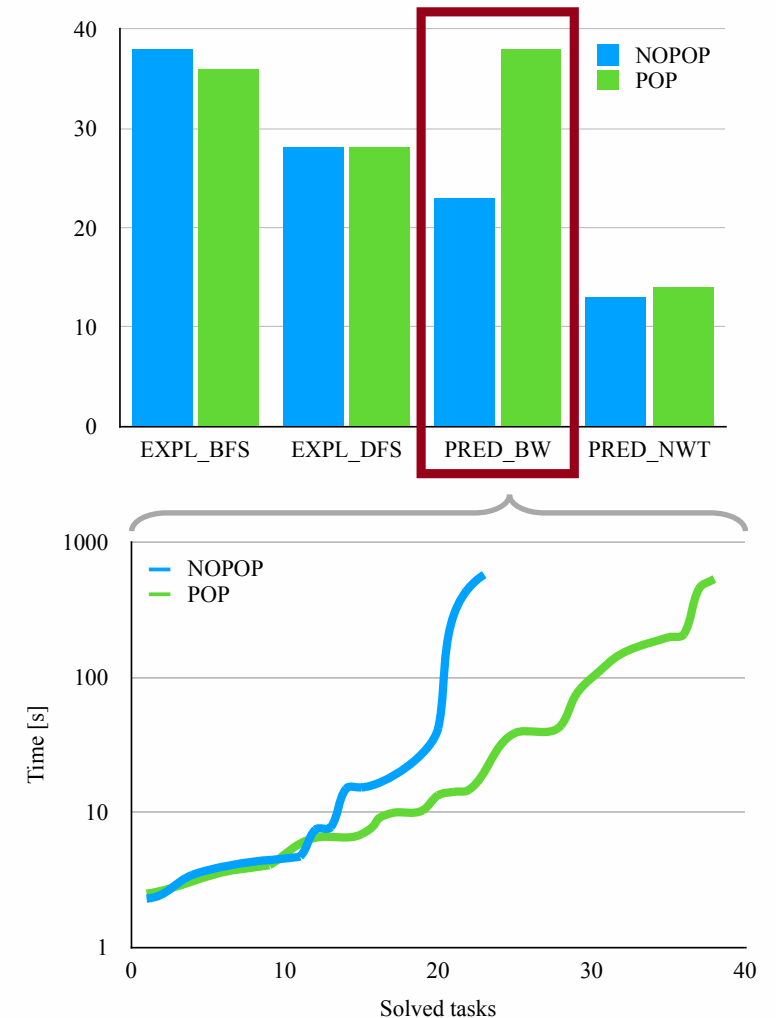
¹ <https://github.com/ftsrg/theta>

² <https://github.com/ftsrg/theta/pull/244>

³ <https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks>

Benchmark Results

- Recursive: Improved performance with predicate abstraction.
 - 65% more programs verified
 - 20% of solved tasks could not be verified by any other configuration
 - solves difficult tasks quicker
- Non-procedural: slight performance overhead
 - but: recursion can be detected before verification



Summary

- Research results: novel interprocedural verification technique
 - stack abstraction → reduced abstract state-space
 - integration into existing abstraction-based algorithm (CEGAR)
 - benchmark campaign: improved performance on recursive programs
- Results available to the institution – ✓ implementation open source¹
- Present work at Scientific Students' Association Conference – ✓ II. place²
- Present work at New Nation Excellence Program Conference – ✓
- Publish work to international community – paper in progress
- Future work: combination with other orthogonal techniques (e.g. summaries)

¹ <https://github.com/ftsrg/theta>

² <https://tdk.bme.hu/VIK/DownloadPaper/Absztrakcialapu-interproceduralis>