

Bottoms Up for CHCs: Novel Transformation of Linear CHCs to Software Verification

Presenting: Márk Somorjai

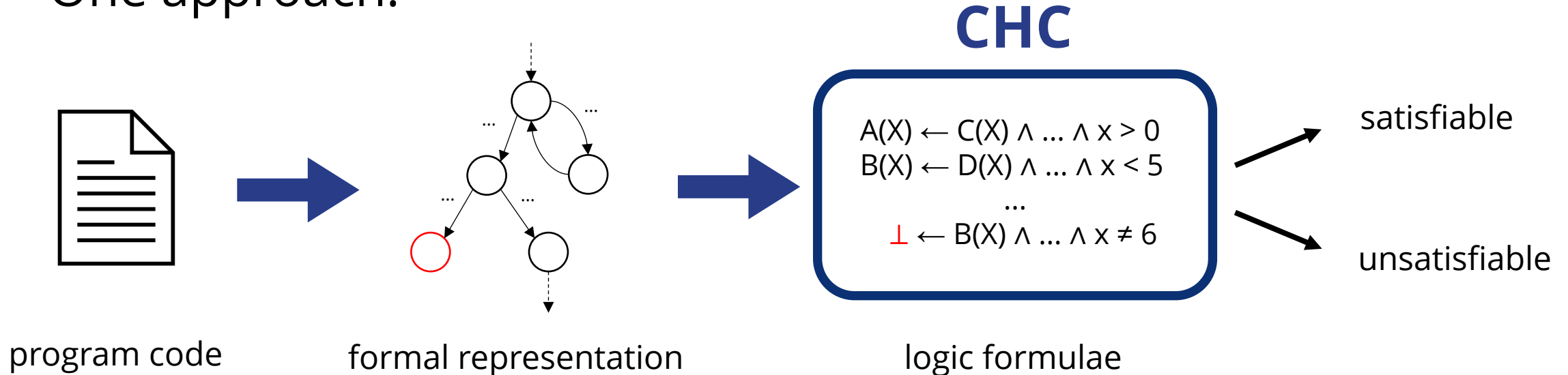
This research was partially funded by the ÚNKP-22-{2,3}-I New National Excellence Program and the 2019-1.3.1-KK-2019-00004 project from the National Research, Development and Innovation Fund of Hungary.



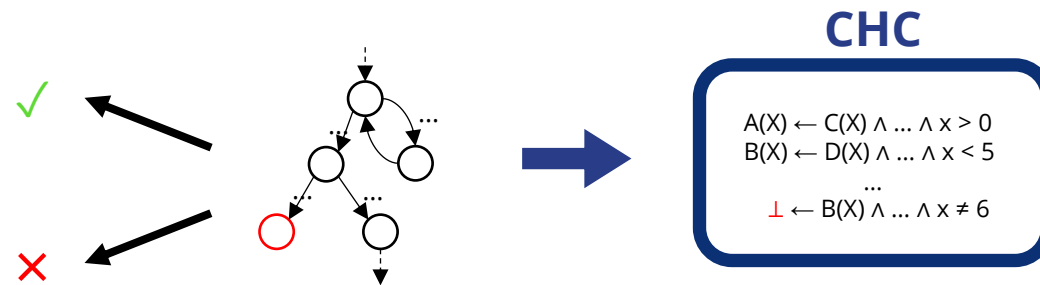
**Critical Systems
Research Group**

Formal Software Verification

- Goal: prove certain properties of software, e.g. error reachability
 - ✓ mathematical proof
 - ✗ refutation
- One approach:



Our Approach

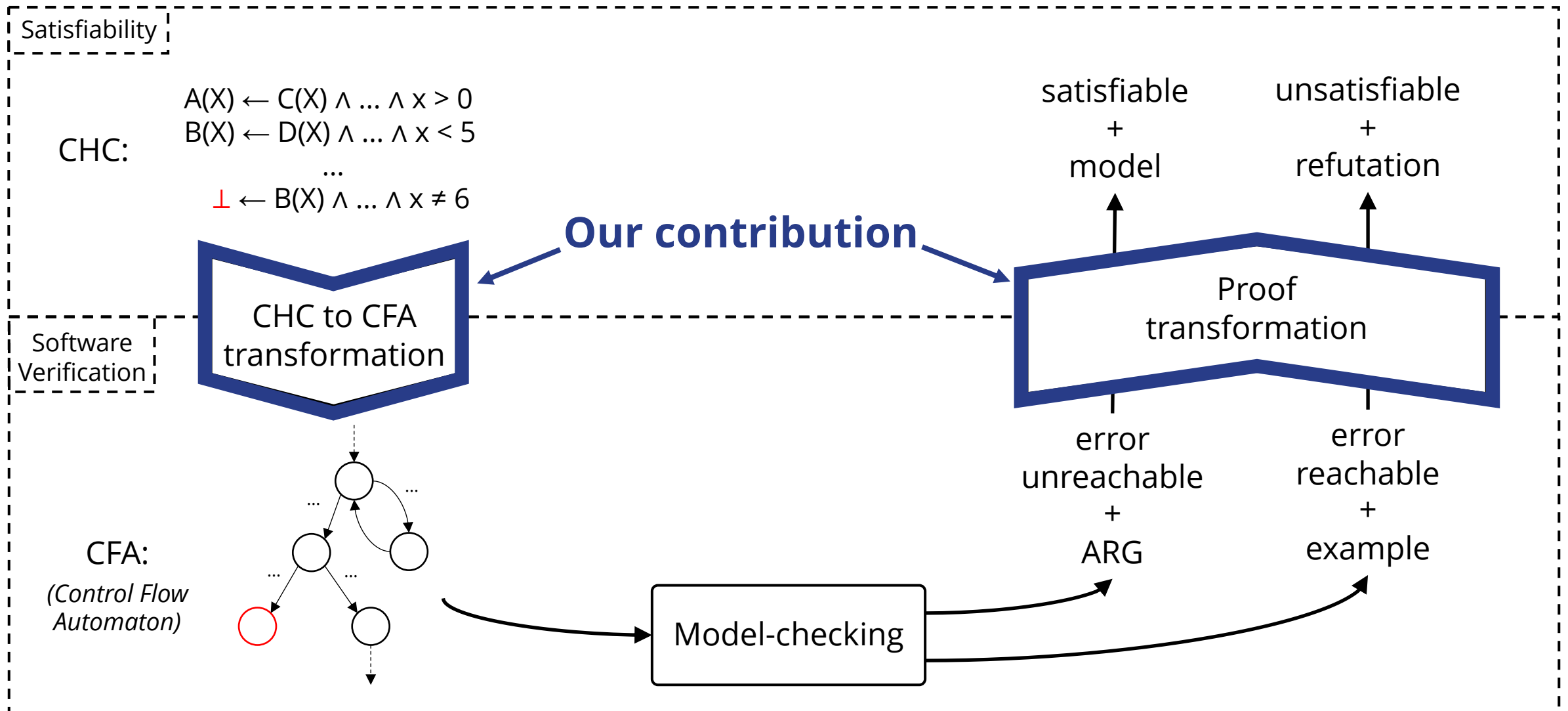


- Advantages:
 - various model-checking techniques
 - powerful

Our goal: efficiently solve CHCs by

1. transforming them to a formal software representation
2. applying powerful abstraction based model-checking techniques

Overview of the Approach



Constrained Horn Clauses

- Linear CHCs: at most 1 uninterpreted function in body

$$\underbrace{H(X) \leftarrow B(X)}_{\text{uninterpreted functions}} \wedge \underbrace{\varphi(X)}_{\text{interpreted formulae}}$$

- CHC types:

- fact: no uninterpreted function in body

$$F(x) \leftarrow x = 0$$

- query: no uninterpreted function in head

$$\perp \leftarrow F(x) \wedge x > 0$$

- deduction: uninterpreted function in both

$$F(y) \leftarrow F(x) \wedge x \leq 1 \wedge y = x + 1$$

Control Flow Automaton

- graph-like representation of programs

Program

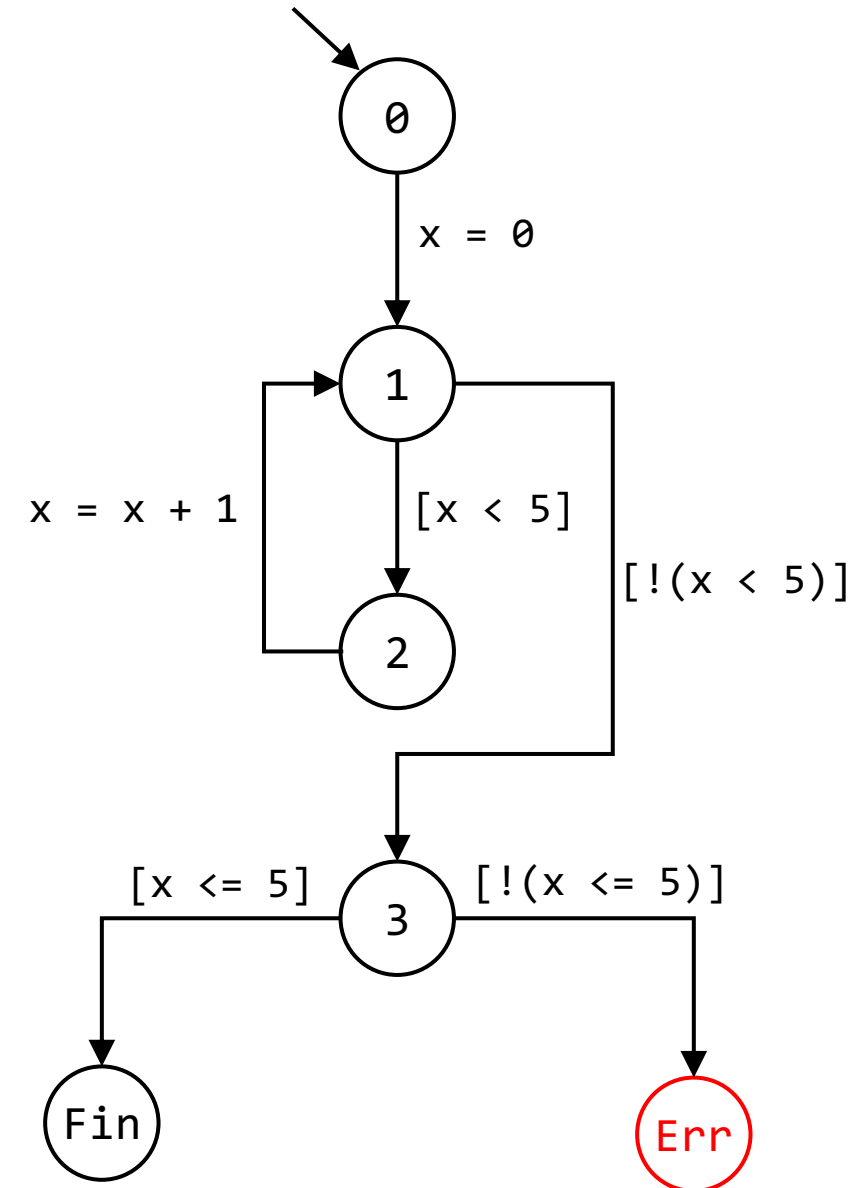
locations $\longleftrightarrow$
statements $\longleftrightarrow$

CFA

nodes
edges

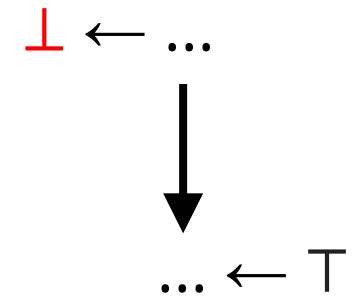
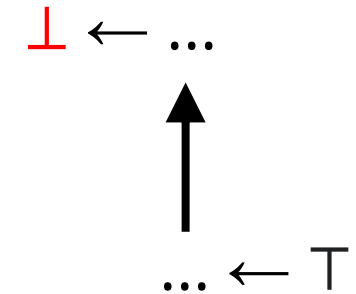
- statement can be:
 - assignment
 - guard
 - procedure call

```
0: x = 0
1: while (x < 5) {
2:     x = x + 1
   }
3: assert(x <= 5)
```



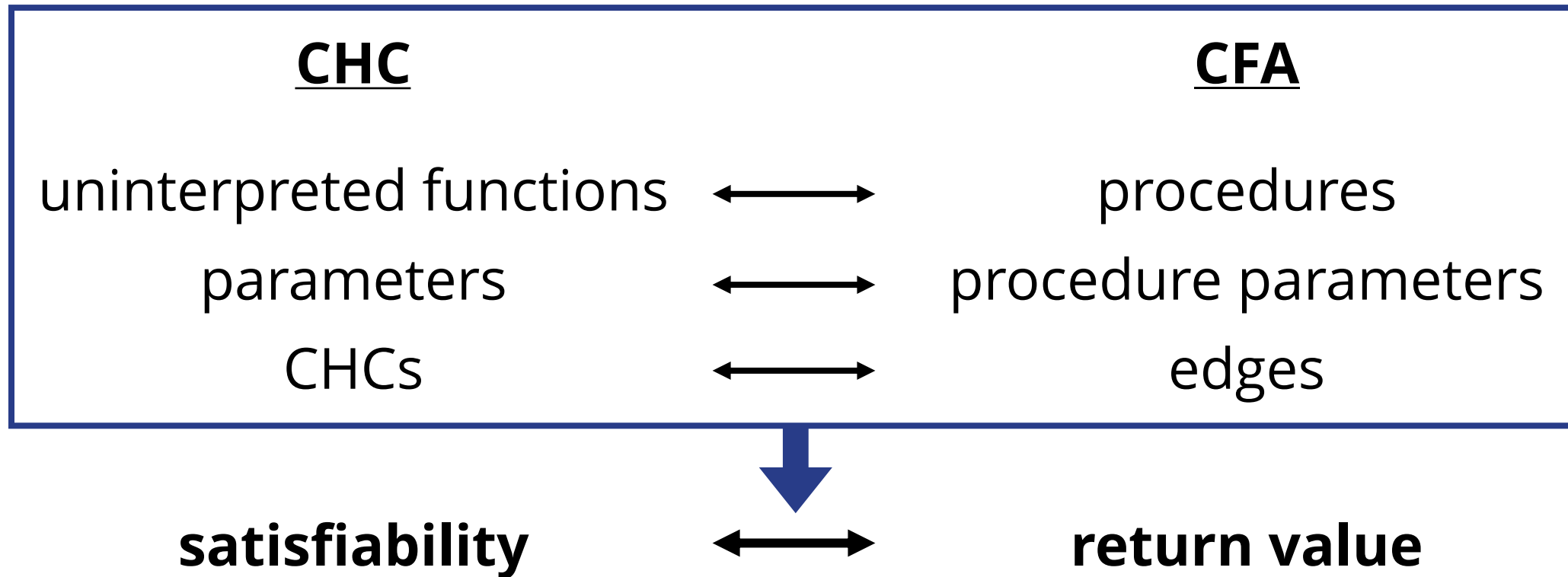
Approaches to CHC to CFA transformation

- Our contribution: *forward/bottom-up* approach
 - repeated application of applicable CHCs to check if $\perp$ is deducible
- Related work: *backward/top-down* approach
 - used by Ultimate Unihorn¹
 - CHC to Boogie code transformation
 - maps uninterpreted functions to procedures
 - recursively checks if body of CHC can be deduced, starting from $\perp$

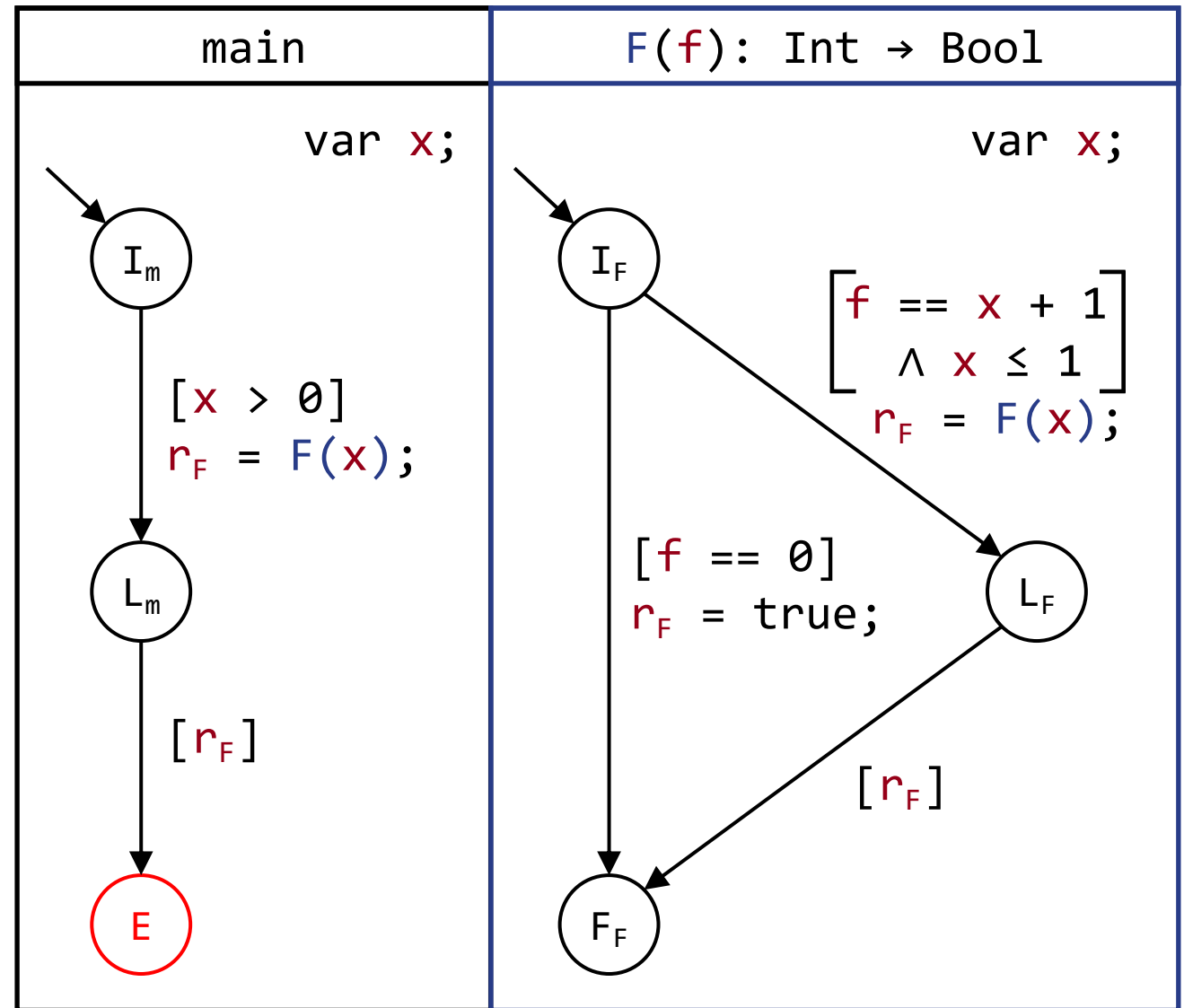
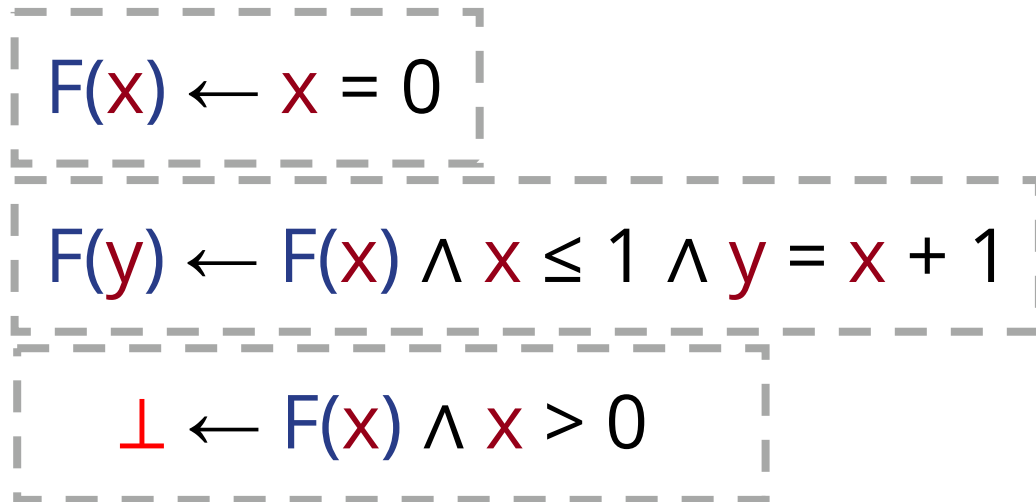


Backward CHC to CFA Transformation

- Unihorn's approach adapted to CFA

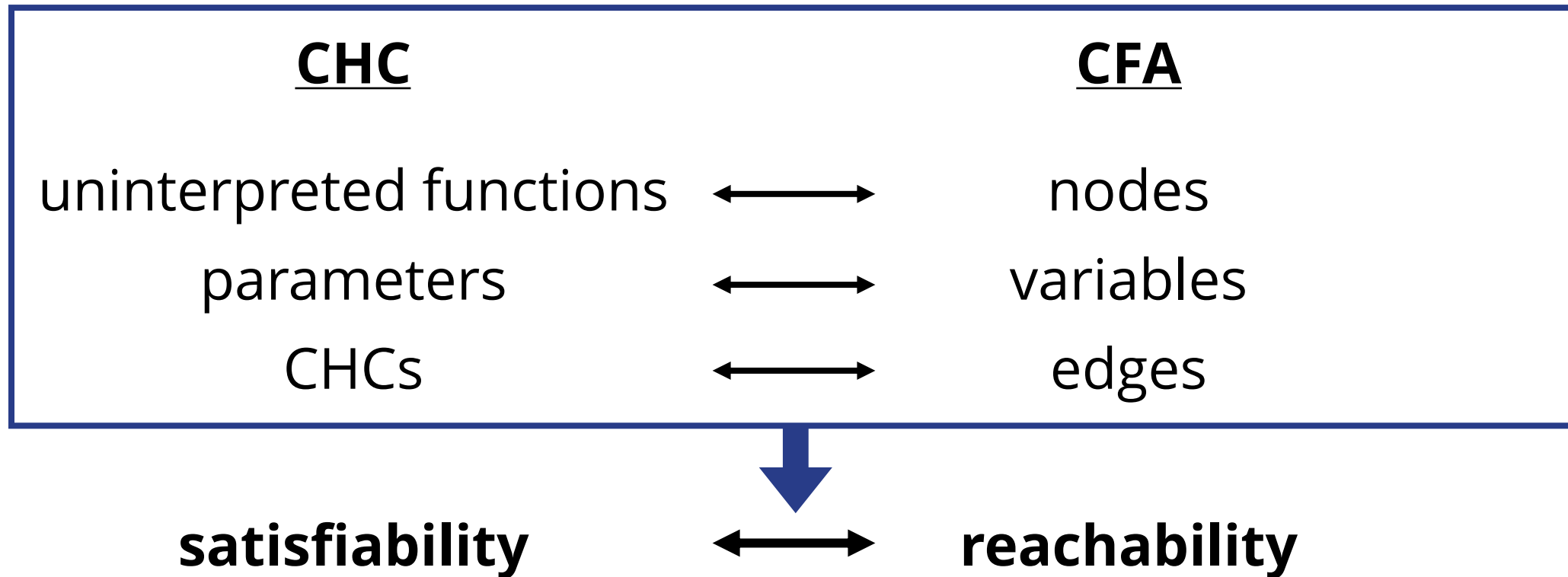


Example

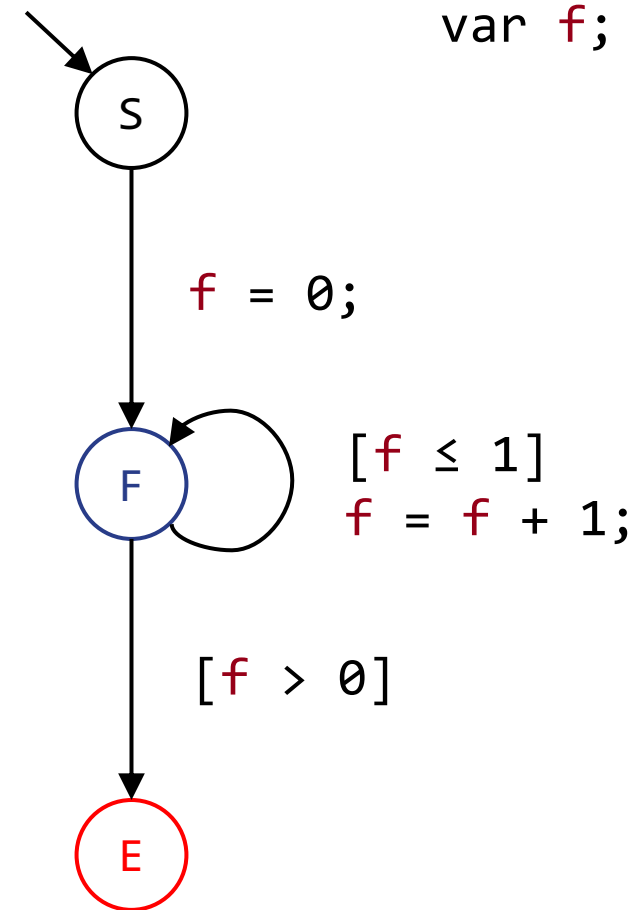
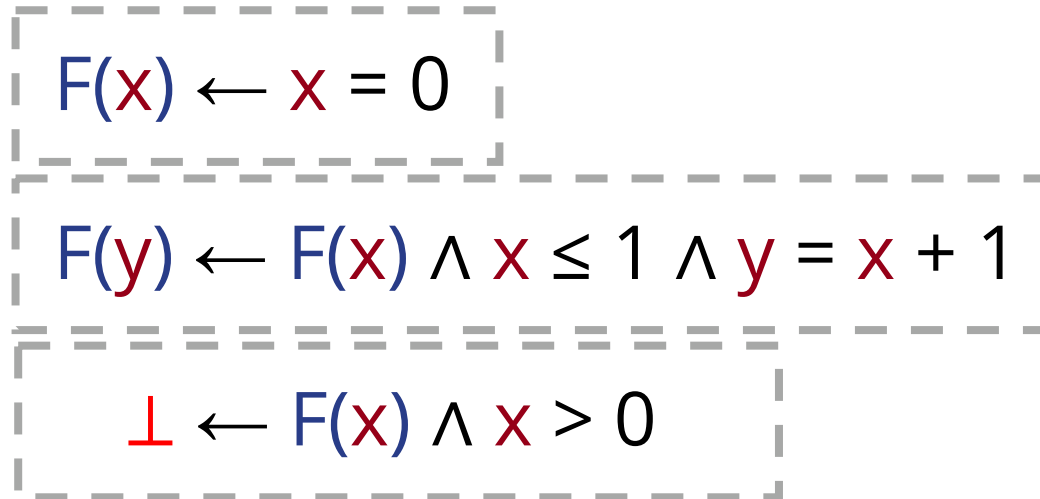


Forward CHC to CFA Transformation

- Our contribution: a novel approach

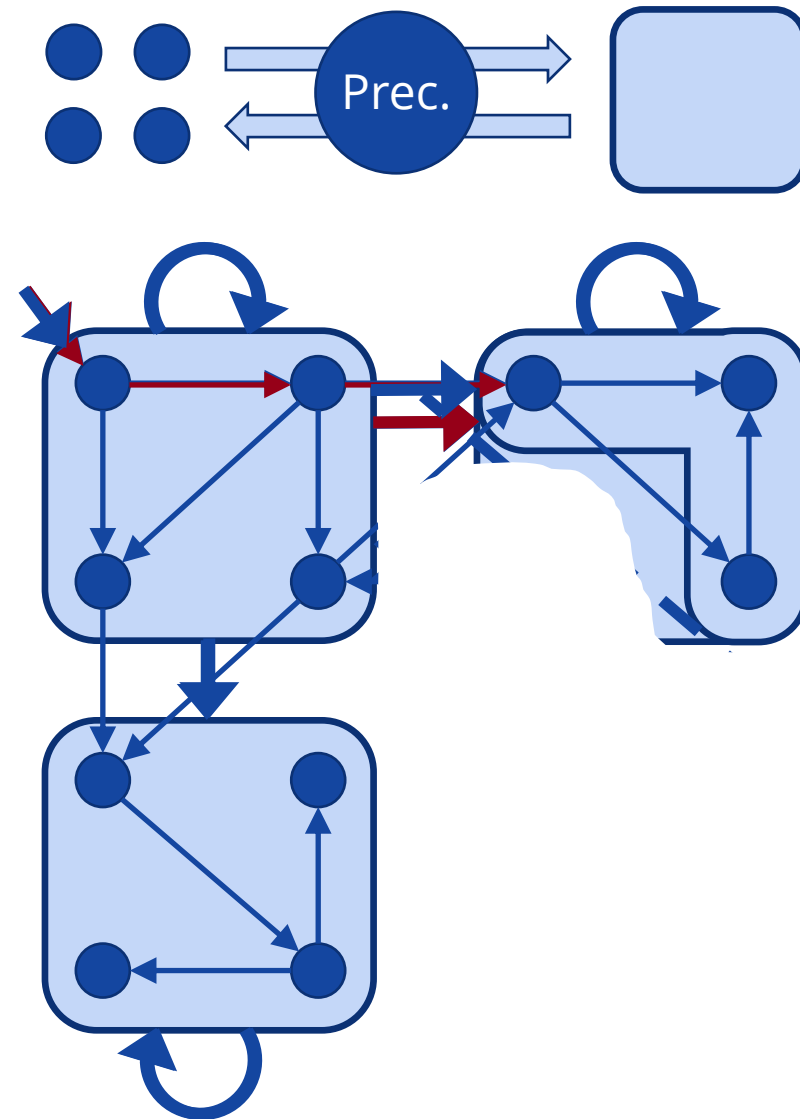


Example



CounterExample-Guided Abstraction Refinement (CEGAR)

- Abstract state space
 - Overapproximates the reachable state space
 - Safe if no error found
 - Precision: Degree of overapproximation
- Abstract counterexample
 - Feasible in abstract state space
 - Feasible in concrete state space?
 - Unsafe if yes
 - Refine if no (abstraction too coarse)
- Refinement
 - Removes unreachable parts from abstract state space



Concrete
state space

Over-
approximate

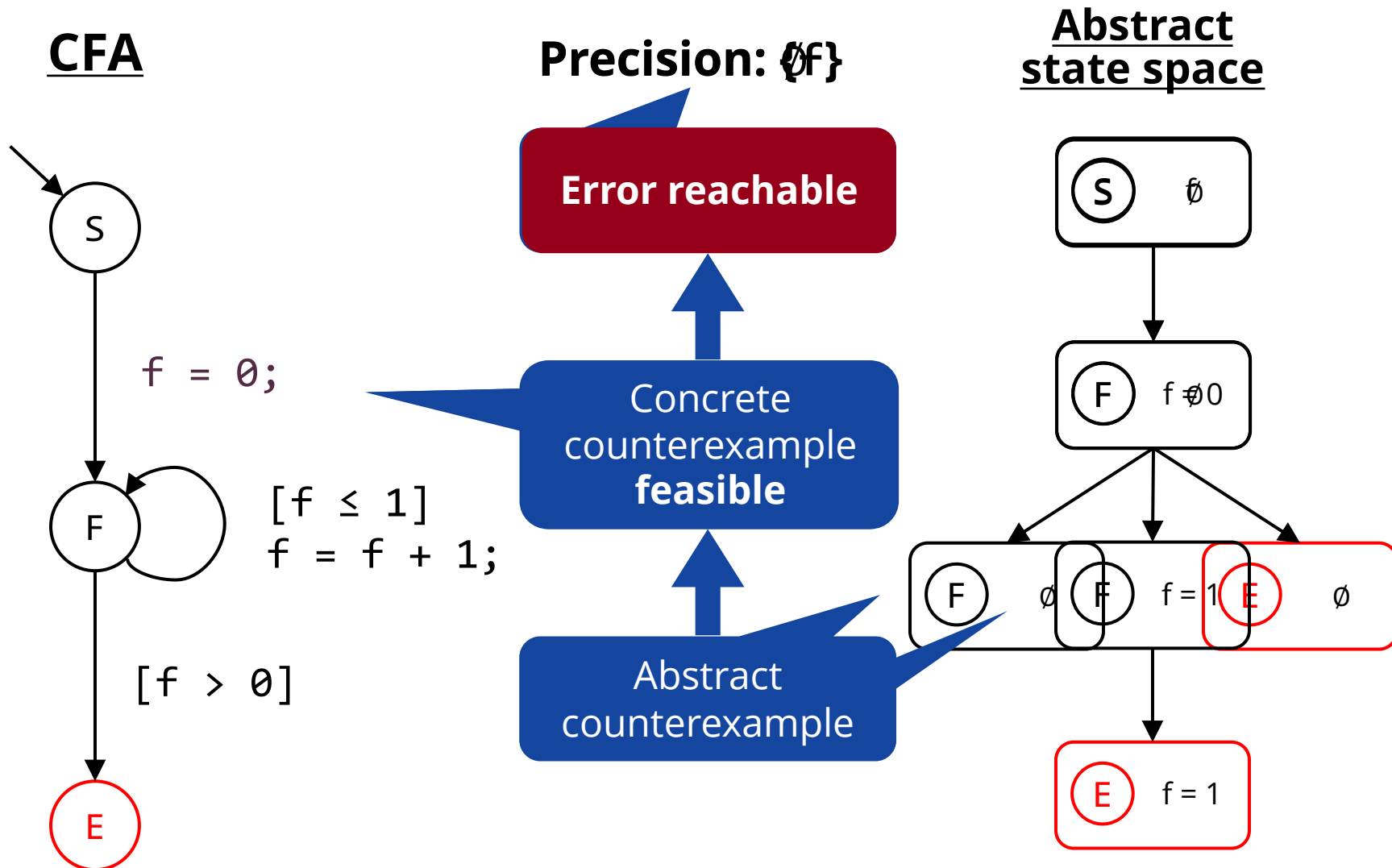
Abstract
state space

Abstract
counterexample

Feasibility
checking

Refinement

Model Checking of Example



Proof Transformation

- Abstraction based model-checking algorithm checks CFA

✓ error unreachable

- all abstract states explored, none contain errors
- reachability $\longleftrightarrow$ satisfiability



- uninterpreted functions should evaluate to true with the explored states' variable values

✗ error reachable

- a concrete path to the erroneous state is given
- edges $\longleftrightarrow$ CHCs

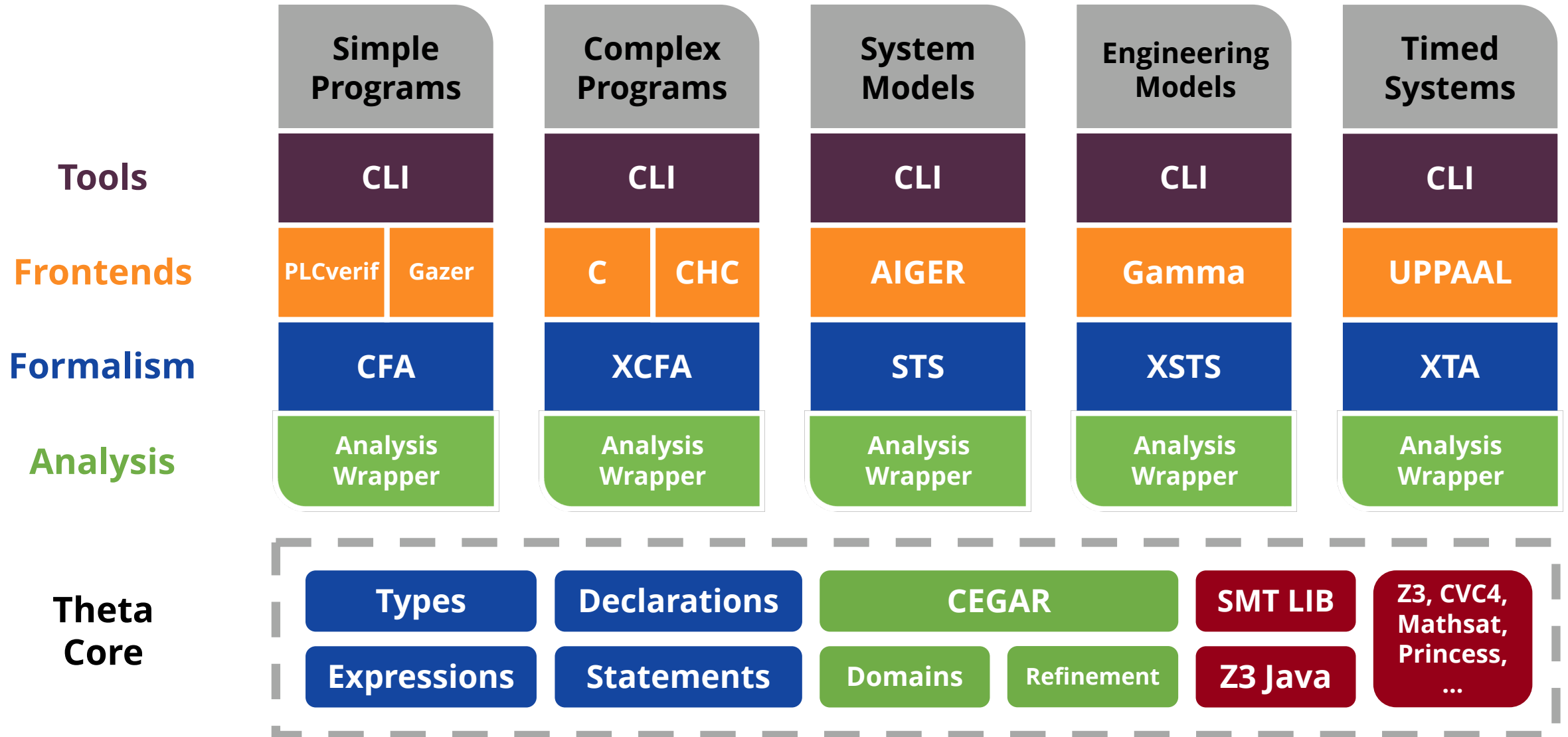


- sequence of CHC applications leading to $\perp$

Implementation



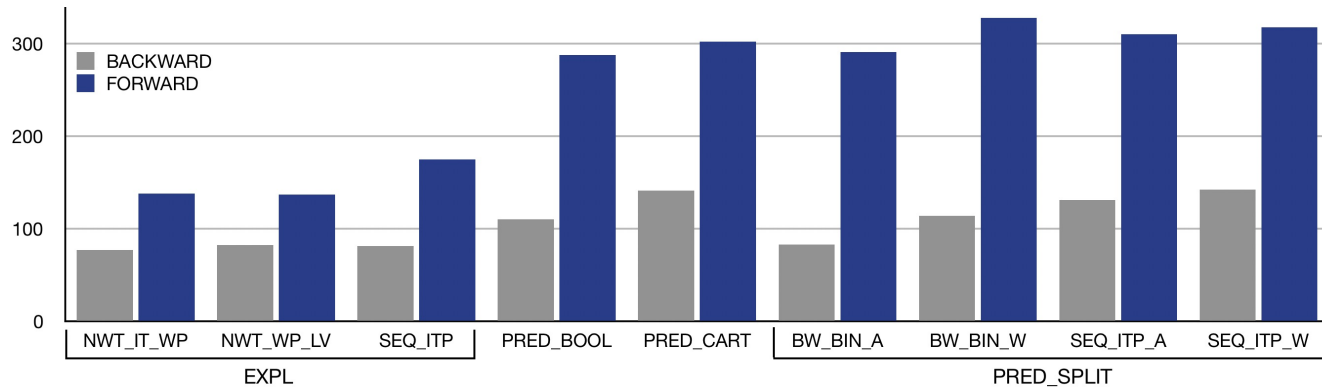
- Transformations implemented in **Theta**¹
 - open-source model-checking framework
 - core algorithm: Counterexample-Guided Abstraction Refinement
- Pros:
 - modular and highly configurable
 - various formalisms, domains, strategies
 - familiarity: developed at BUTE
- Cons:
 - not the most powerful (bottom 20% at SV-COMP23)²



Evaluation

- CHC frontends implemented in Theta
 - backward transformation
 - forward transformation
- Satisfiability is answered, no model/refutation generation
- 585 CHC benchmarks from the LIA-Lin track of CHC-COMP21¹
- Many configurations tested, using BenchExec by SoSyLab²
- No wrong answers, # solved tasks (5 minute timeout) is shown

Benchmark Results



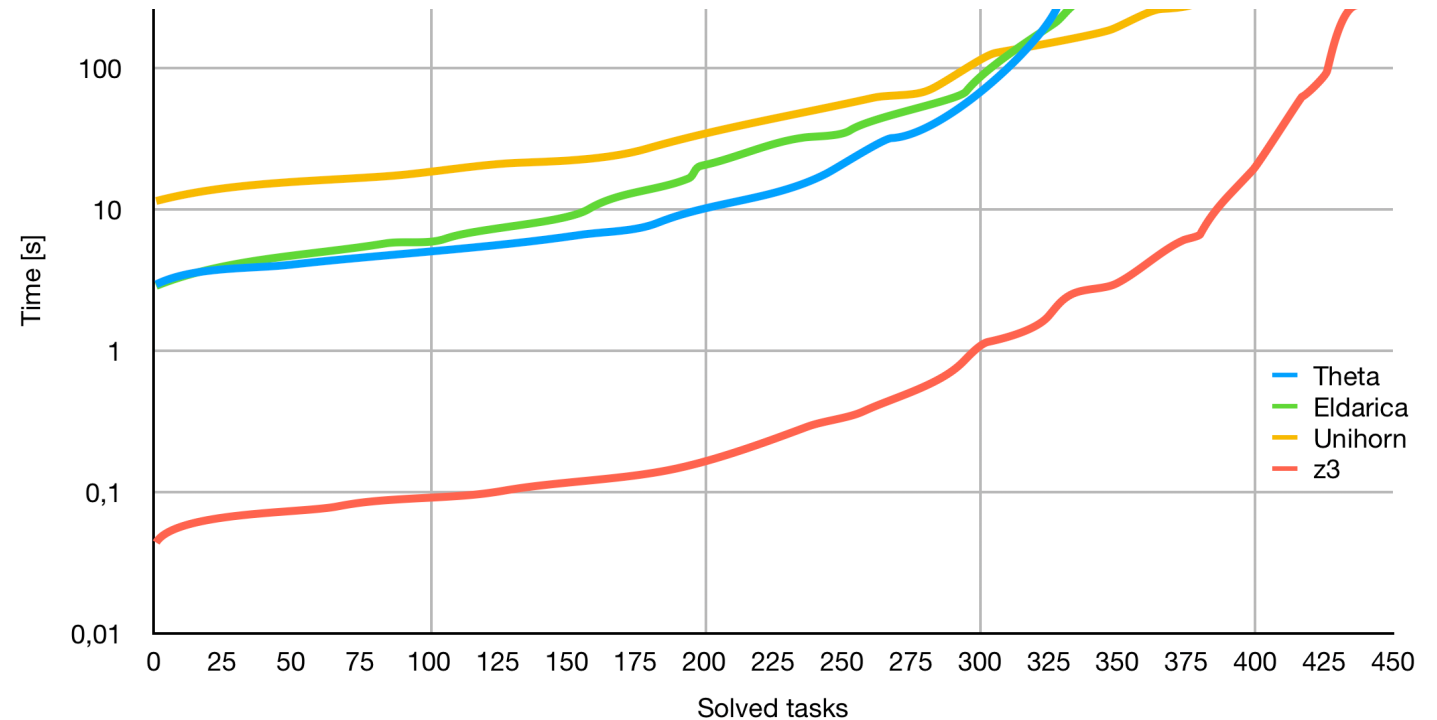
Different Theta configurations

Forward transformation performs better than backward.

Comparison to other tools

(flatter is better)

Theta achieves comparable performance to other CHC solvers.



Summary

- Theoretical results: novel approach to solving linear CHCs
 - forward CHC to CFA transformation
 - proof transformation
- Practical results: implemented frontends in Theta
 - both forward and backward approach
 - Theta can solve CHCs with comparable performance to other solvers
 - Theta was submitted to CHC-COMP '23
- Future work
 - add frontend to more powerful model-checkers
 - implement model/refutation generation