HCVS 2025

Theta as a Horn Solver

Levente Bajczi, Milán Mondok, Vince Molnár

Supported by the 2024-2.1.1-EKOP-2024-00003 University Research Scholarship Programme under project numbers **EKOP-24-3-BME-{78,213}**, and the Doctoral Excellence Fellowship Programme under project number **400434/2023**; with the support provided by the Ministry of Culture and Innovation of Hungary from the NRDI Fund.

Critical Systems Research Group

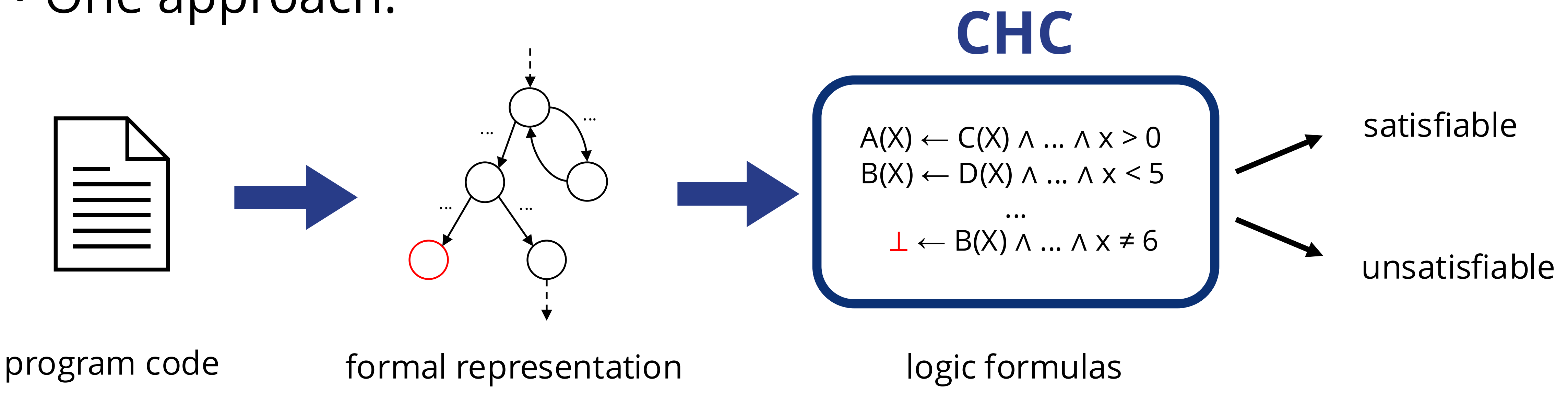
Theta

github.com/ftsrg/theta

- Generic, modular and configurable **model checking framework** developed at Budapest University of Technology
- Originally ARG-based **CEGAR**
- Recently further algorithms:
  - **BMC, K-IND, IMC, Saturation, IC3/PDR**
- Wide array of supported input languages
  - C, Statecharts, PLC, Petri-nets, AIGER HW models, CHC
- First participated in CHC-COMP in 2023

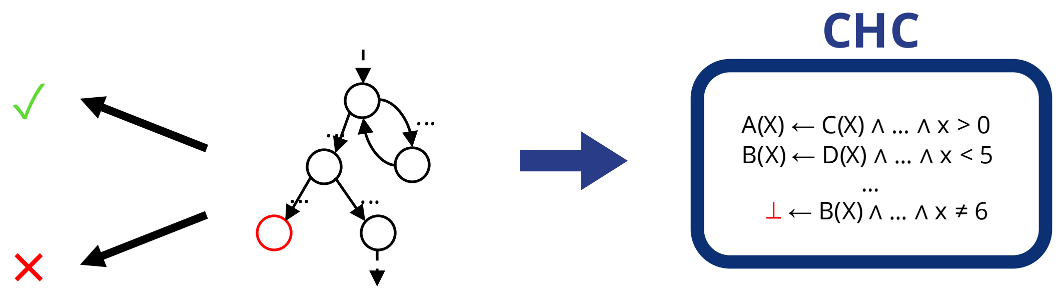
Software Model Checking

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



$A(X) \leftarrow C(X) \land \dots \land x > 0$
$B(X) \leftarrow D(X) \land \dots \land x < 5$
$\dots$
$\bot \leftarrow B(X) \land \dots \land x \neq 6$

Our Approach



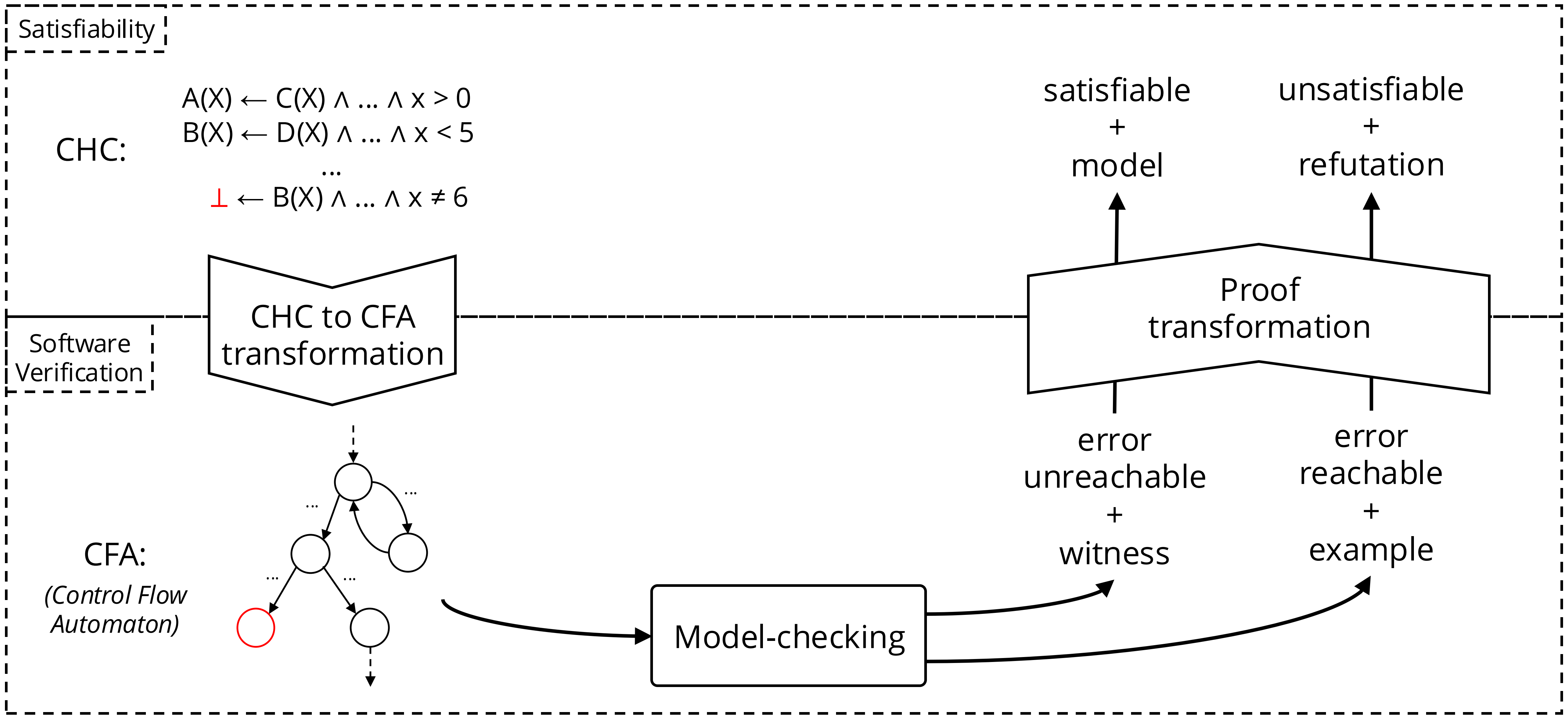
$A(X) \leftarrow C(X) \land \ldots \land x > 0$
$B(X) \leftarrow D(X) \land \ldots \land x < 5$
…
$\bot \leftarrow B(X) \land \ldots \land x \neq 6$

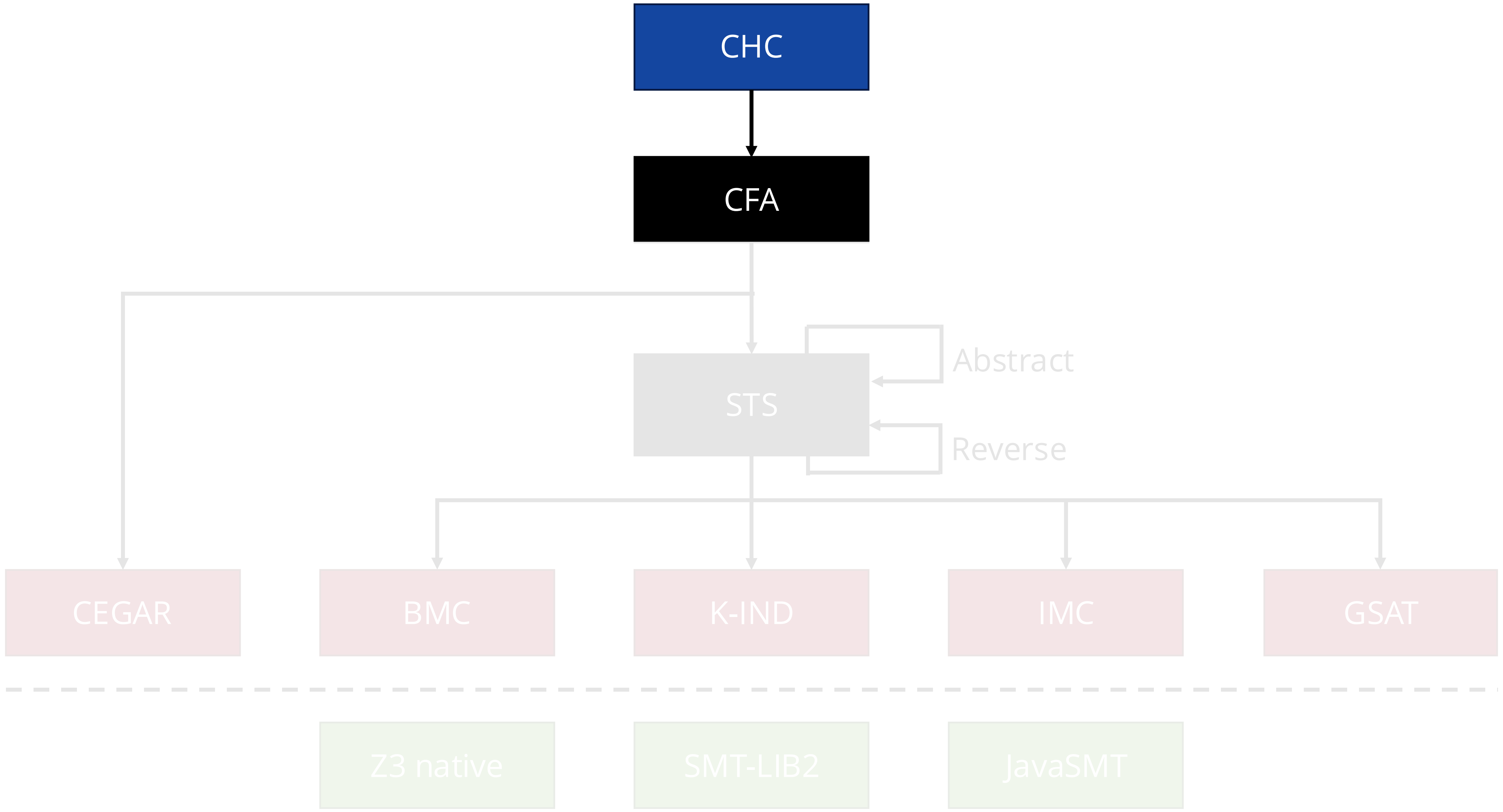
- Advantages:
  - various model-checking techniques
  - powerful model-checking techniques

Our goal: efficiently solve CHCs by

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

Overview of the Approach




CHC
CFA
STS
Abstract
Reverse
CEGAR
BMC
K-IND
IMC
GSAT
Z3 native
SMT-LIB2
JavaSMT

Constrained Horn Clauses

- Linear CHCs: at most 1 uninterpreted function in body

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

- CHC types:
  - fact: no uninterpreted function in body $F(x) \leftarrow x = 0$
  - query: no uninterpreted function in head $\bot \leftarrow F(x) \land x > 0$
  - deduction: uninterpreted function in both $F(y) \leftarrow F(x) \land x \leq 1 \land y = x + 1$

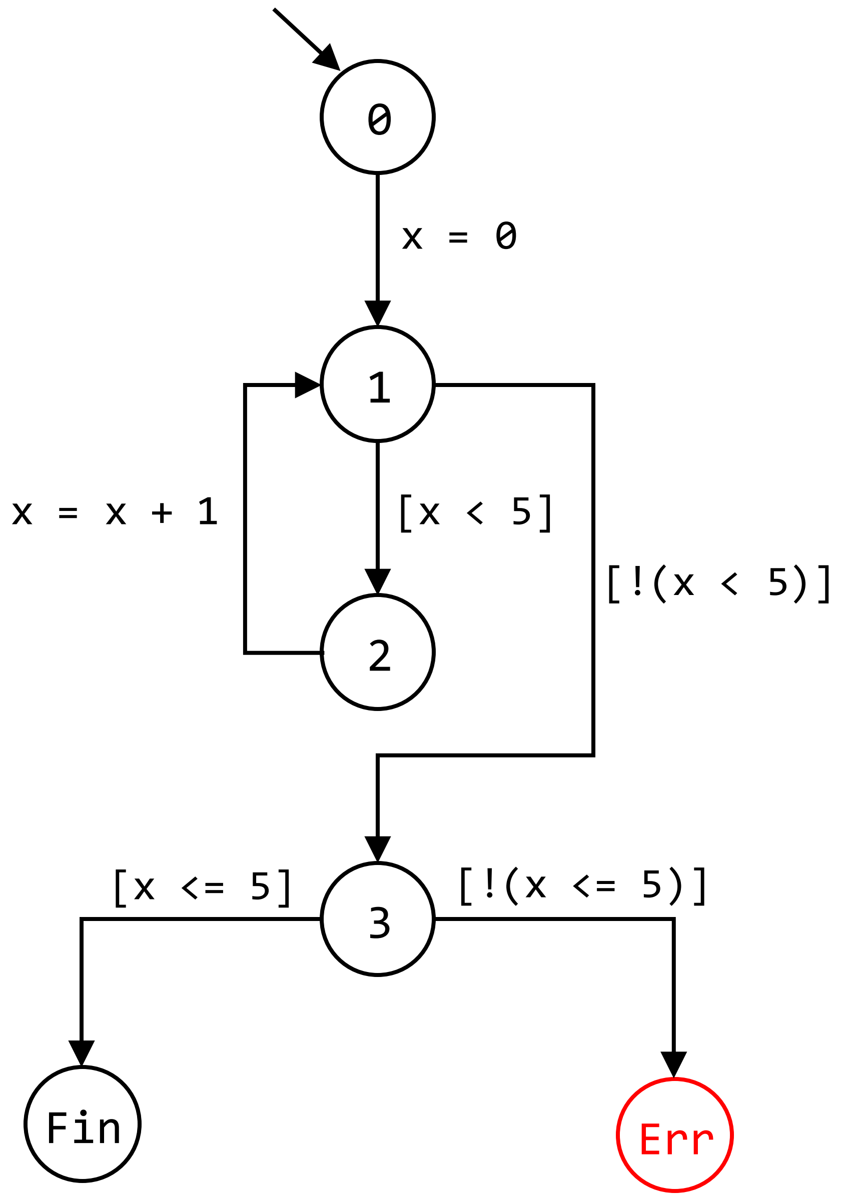
Control Flow Automaton

- graph-like representation of programs

Program		**CFA**
locations	⟷	nodes
statements	⟷	edges

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

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



CHC to CFA transformation options in Theta

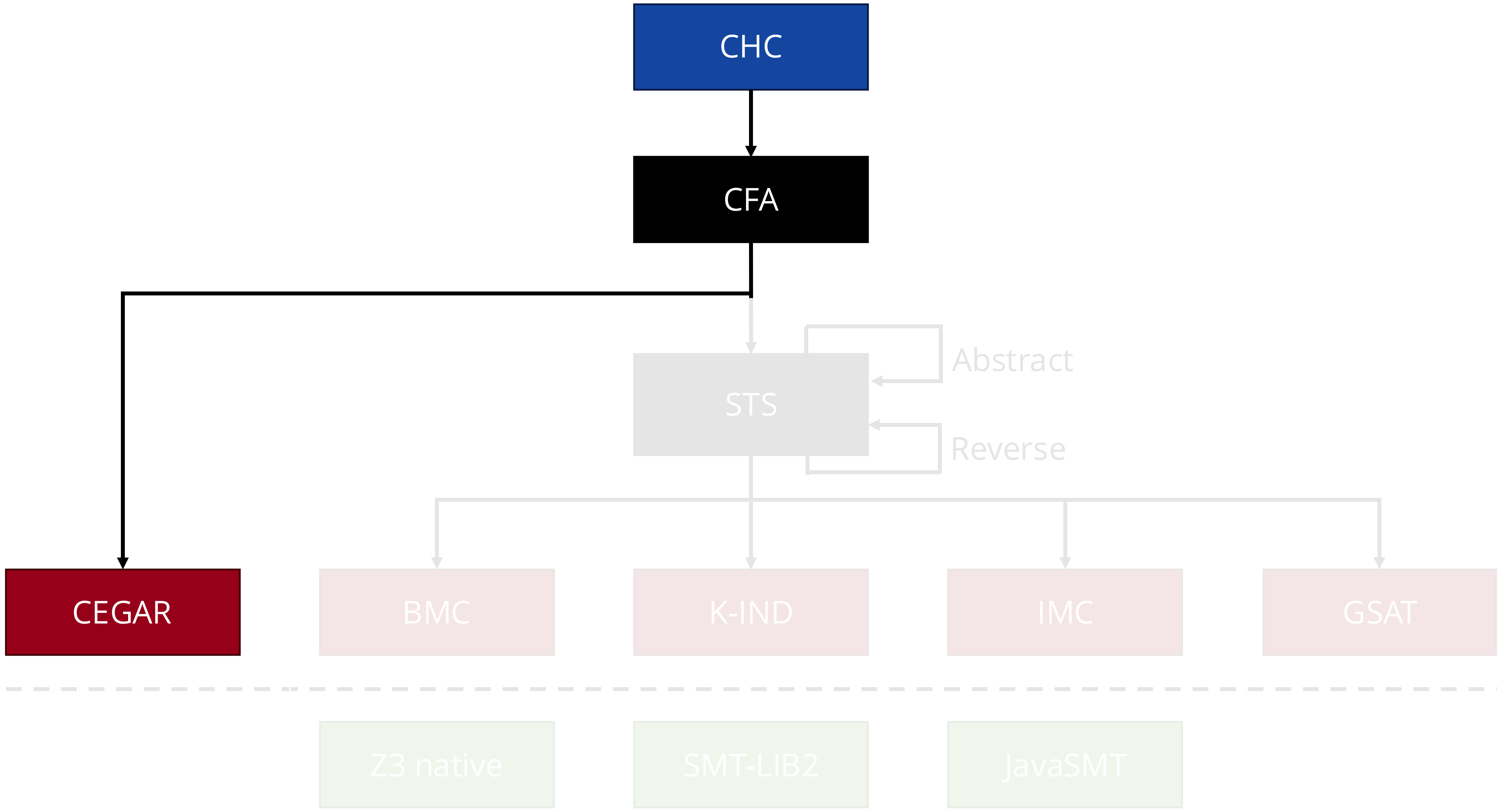
- **Forward/bottom-up** approach
  - unique to Theta, presented at HCVS'23
  - repeated application of applicable CHCs to check if ⊥ is deducible

$$\begin{tikzcd}
\perp & \dots \arrow[l] & \\
& \dots \arrow[u] & \top \arrow[l]
\end{tikzcd}$$

- **Backward/top-down** approach
  - used by Ultimate Unihorn[1]
  - CHC to Boogie code transformation
  - maps uninterpreted functions to procedures
  - recursively checks if body of CHC can be deduced, starting from ⊥

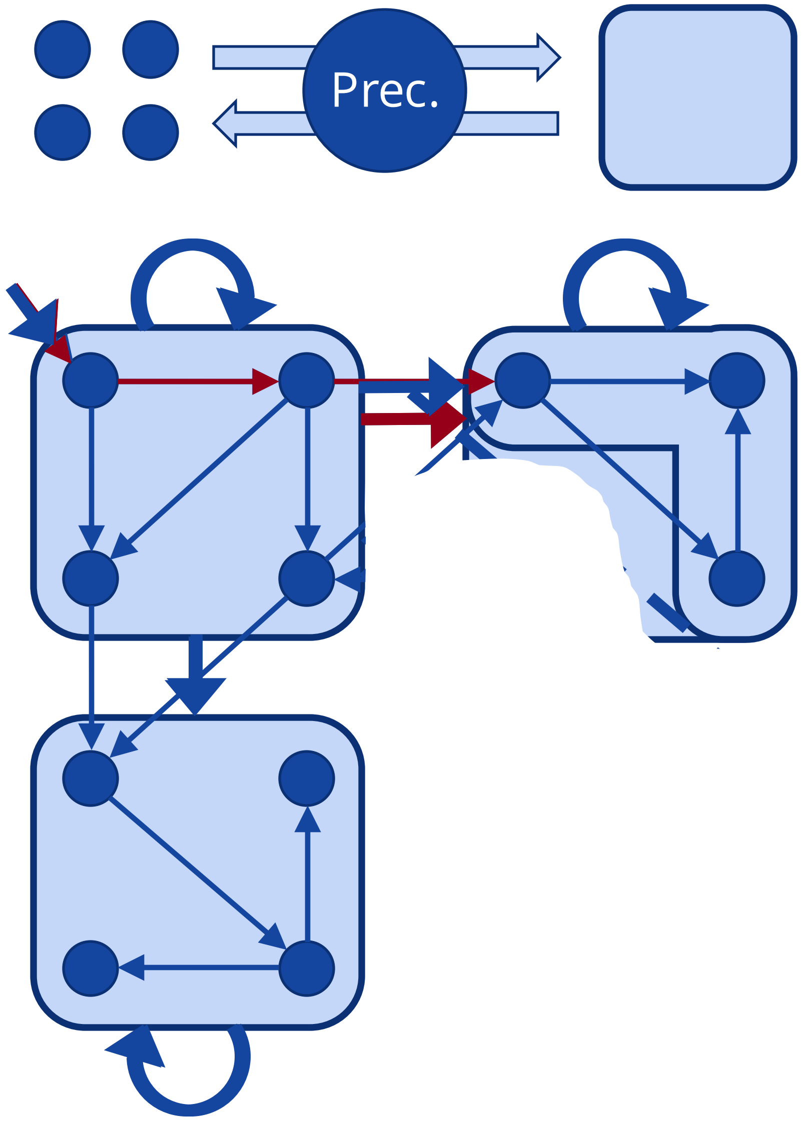
$$\begin{tikzcd}
\perp & \dots \arrow[l] \arrow[d] & \\
& \dots & \top \arrow[l]
\end{tikzcd}$$

[1] https://github.com/ultimate-pa/ultimate


CHC
CFA
STS
Abstract
Reverse
CEGAR
BMC
K-IND
IMC
GSAT
Z3 native
SMT-LIB2
JavaSMT

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

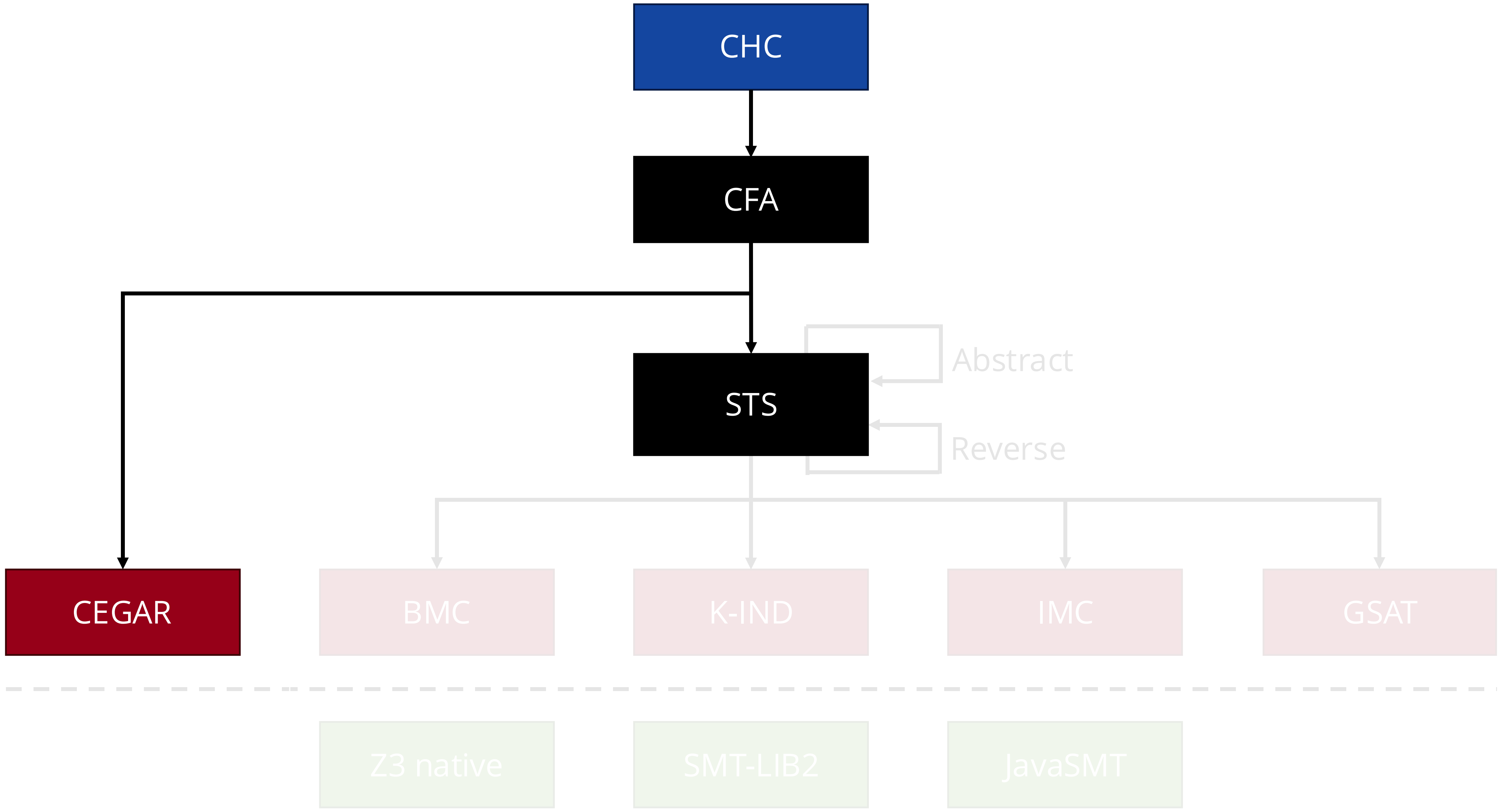
Over-approximate

Abstract state space

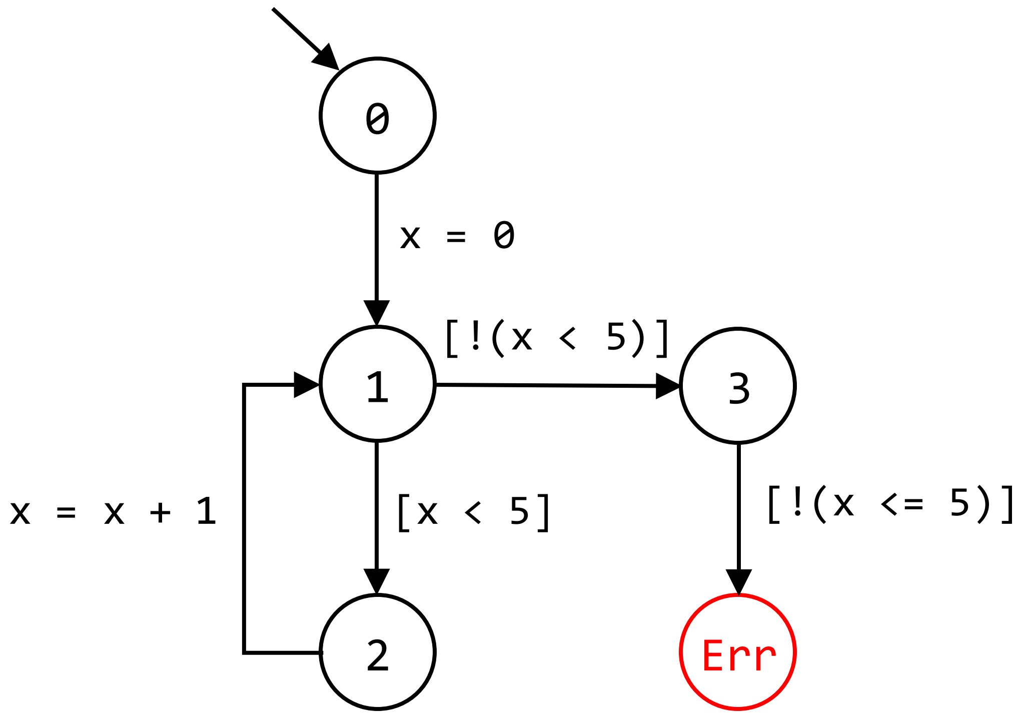
Abstract counterexample

Feasibility checking

Refinement


CHC
CFA
STS
Abstract
Reverse
CEGAR
BMC
K-IND
IMC
GSAT
Z3 native
SMT-LIB2
JavaSMT

CFA to STS Transformation



<u>CFA</u>

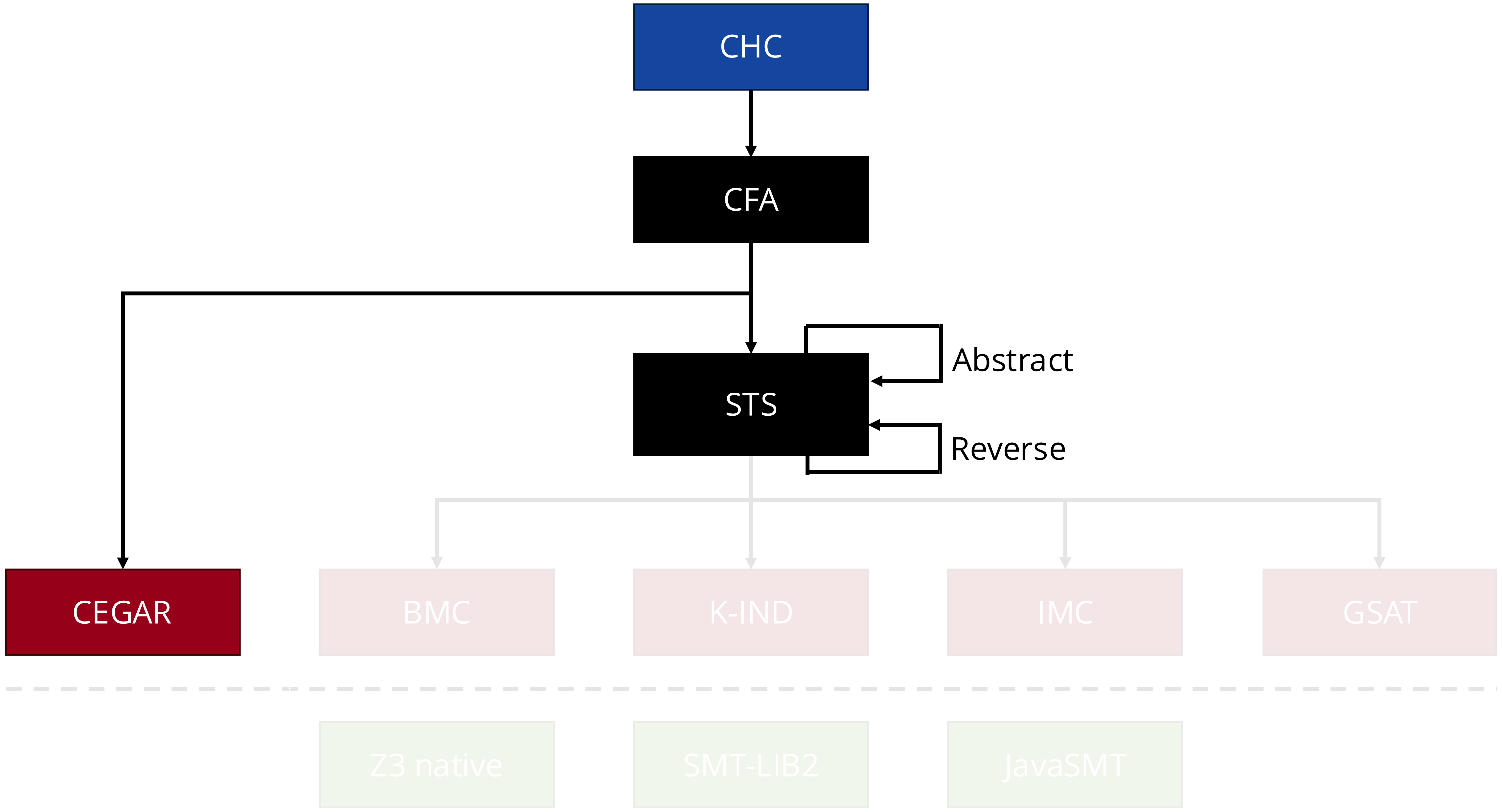
```
I: l=0

T: (l=0 ∧ x'=0 ∧ l'=1)
    ∨ (l=1 ∧ x<5 ∧ x'=x ∧ l'=2)
    ∨ (l=2 ∧ x'=x+1 ∧ l'=1)
    ∨ (l=1 ∧ !(x<5) ∧ x'=x ∧ l'=3)
    ∨ (l=3 ∧ x'=x ∧ l'=Err)

P: !(l=Err)
```

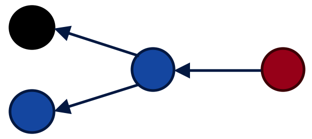
<u>STS</u> : Symbolic Transition System

Currently limited to the FW transformation (single-procedure CFA)


CHC
CFA
STS
Abstract
Reverse
CEGAR
BMC
K-IND
IMC
GSAT
Z3 native
SMT-LIB2
JavaSMT

STS to STS Transformations

- Reversal: Can we reach the **initial state** from the error states with **reversed steps**?



I: $x = 0 \land y < 100$

T: $x' = x + 1 \land y' = y$

P: $x < 50$

STS

I_{rev}: $\neg P$

T_{rev}: $x = x' + 1 \land y = y'$

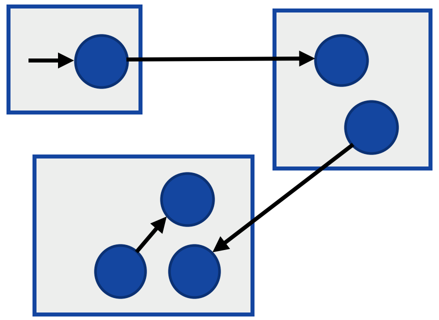
P_{rev}: $\neg I$

Swap **v** with **v'**

Reversed STS

STS to STS Transformations

- Reversal
- Abstraction: Implicit **predicate abstraction** encoded into the model
  - Wrap the analysis in a CEGAR loop



STS

$I: x = 0 \land y < 100$
$T: x' = x + 1 \land y' = y$
$P: x < 50$

+

Precision $\{ (x < 2) \}$

$\Rightarrow$

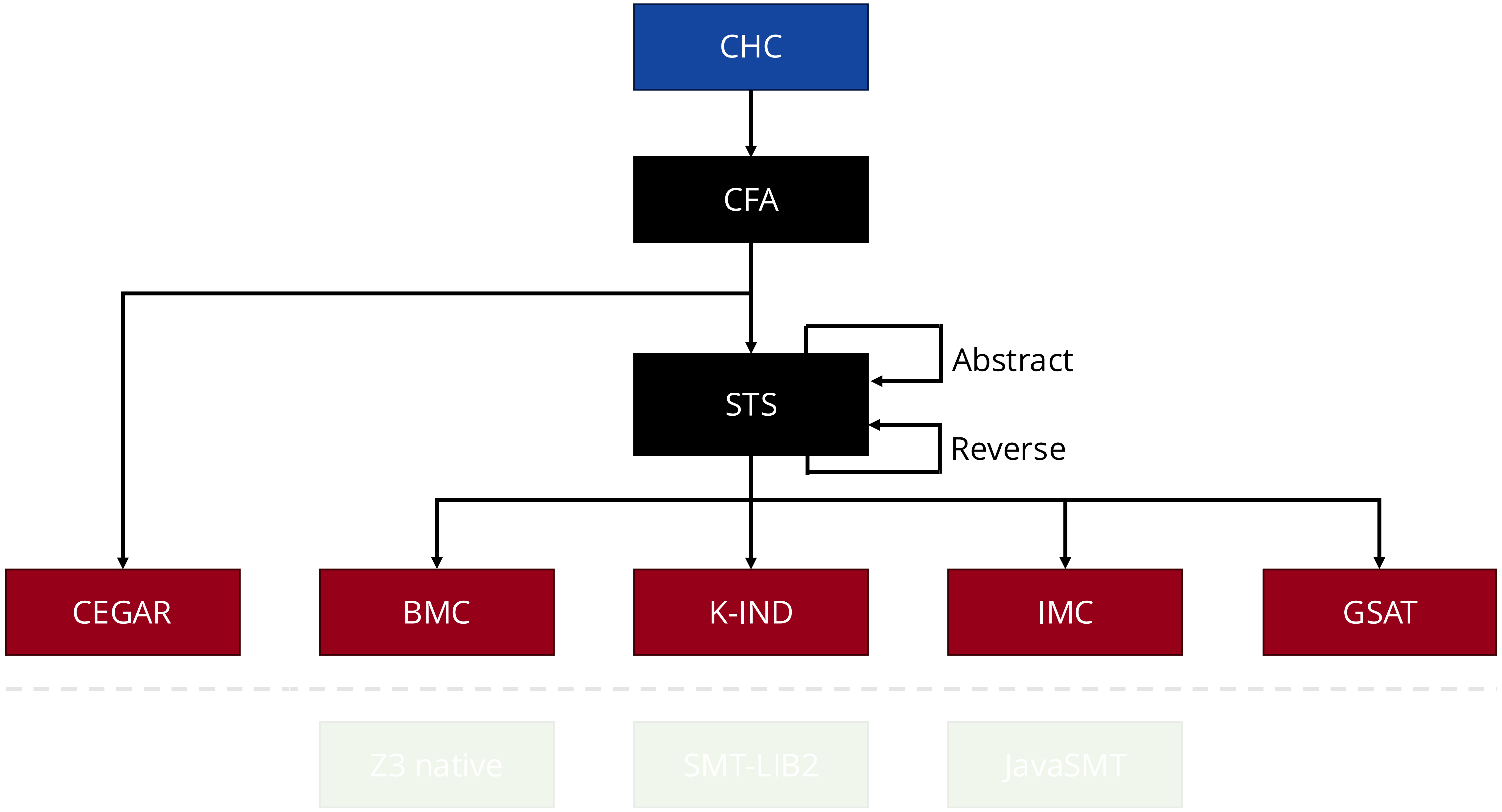
Activation literals to track predicates

$I_{abs}: I \land \mathbf{(p_1 \Leftrightarrow (x < 2))}$

$T_{abs}: T \land \mathbf{(p_1 \Leftrightarrow (x < 2)) \land (p'_1 \Leftrightarrow (x' < 2))}$

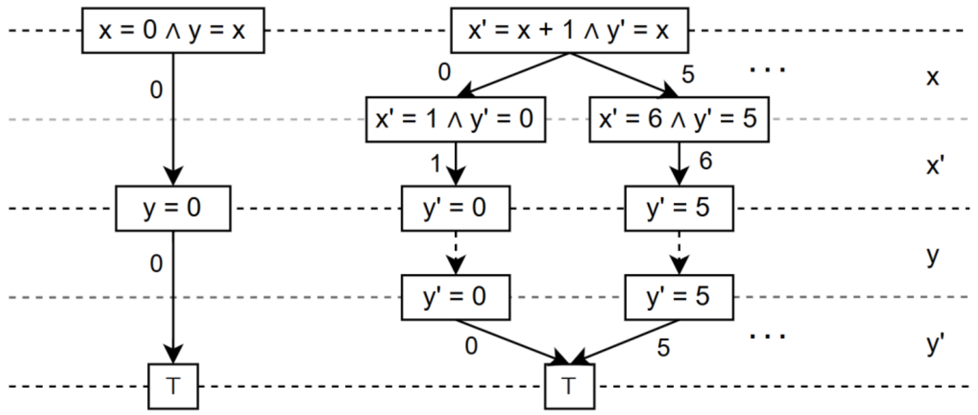
$P_{abs}: \mathbf{\neg(}\neg P \land \mathbf{(p_1 \Leftrightarrow (x < 2)))}$

Abstract STS

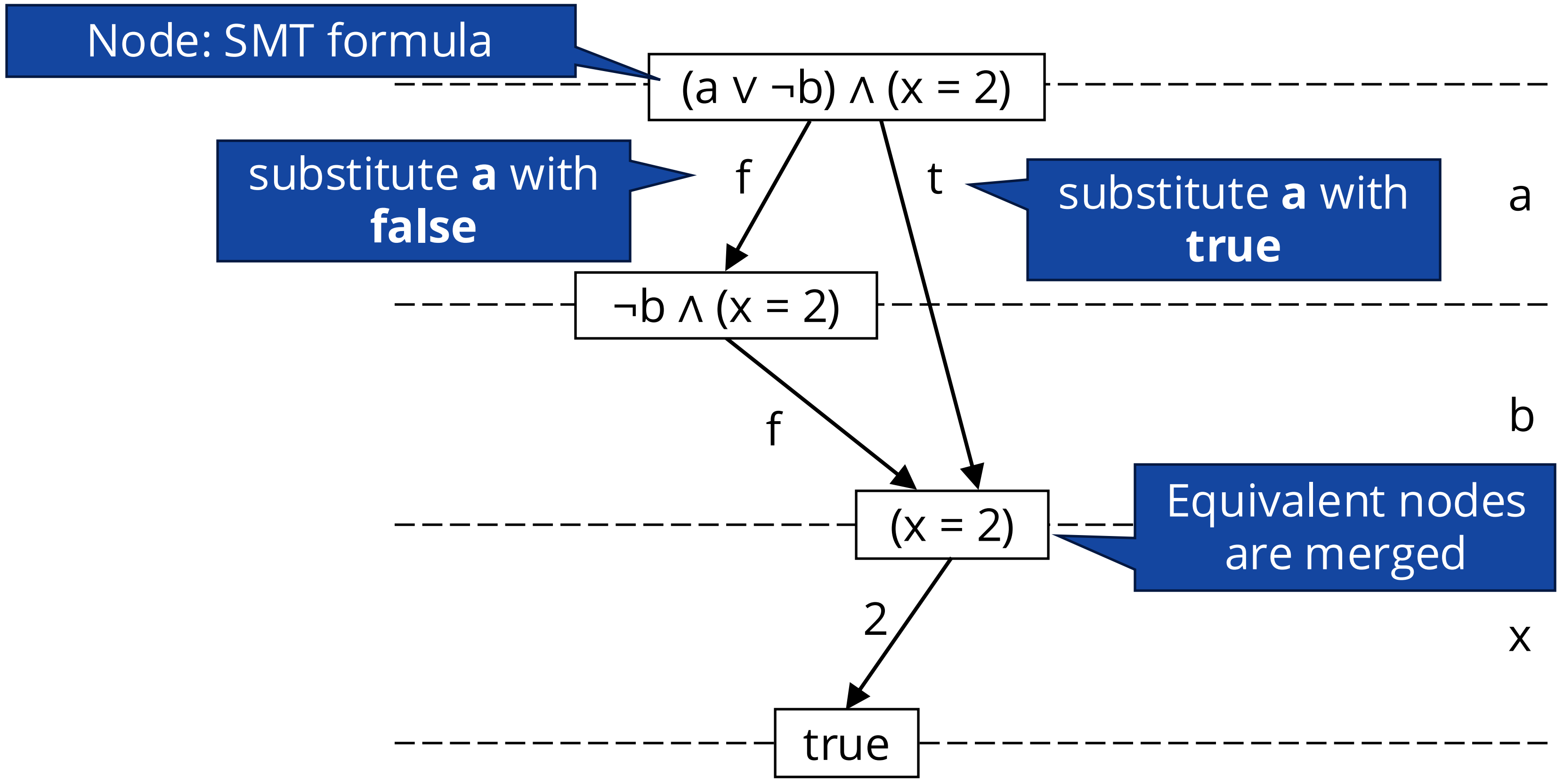

CHC
CFA
STS
Abstract
Reverse
CEGAR
BMC
K-IND
IMC
GSAT
Z3 native
SMT-LIB2
JavaSMT

STS Verification Algorithms

- Bounded model checking (**BMC**)
  - + K-induction (**KIND**)
  - + Interpolation-based model checking (**IMC**)
- Property-directed reachability (**IC3/PDR**) – Not used for CHC yet
- (Generalized) saturation (**GSAT**)
  - – **Substitution diagrams:** Top-down emulation of decision-diagram structure from SMT formulas

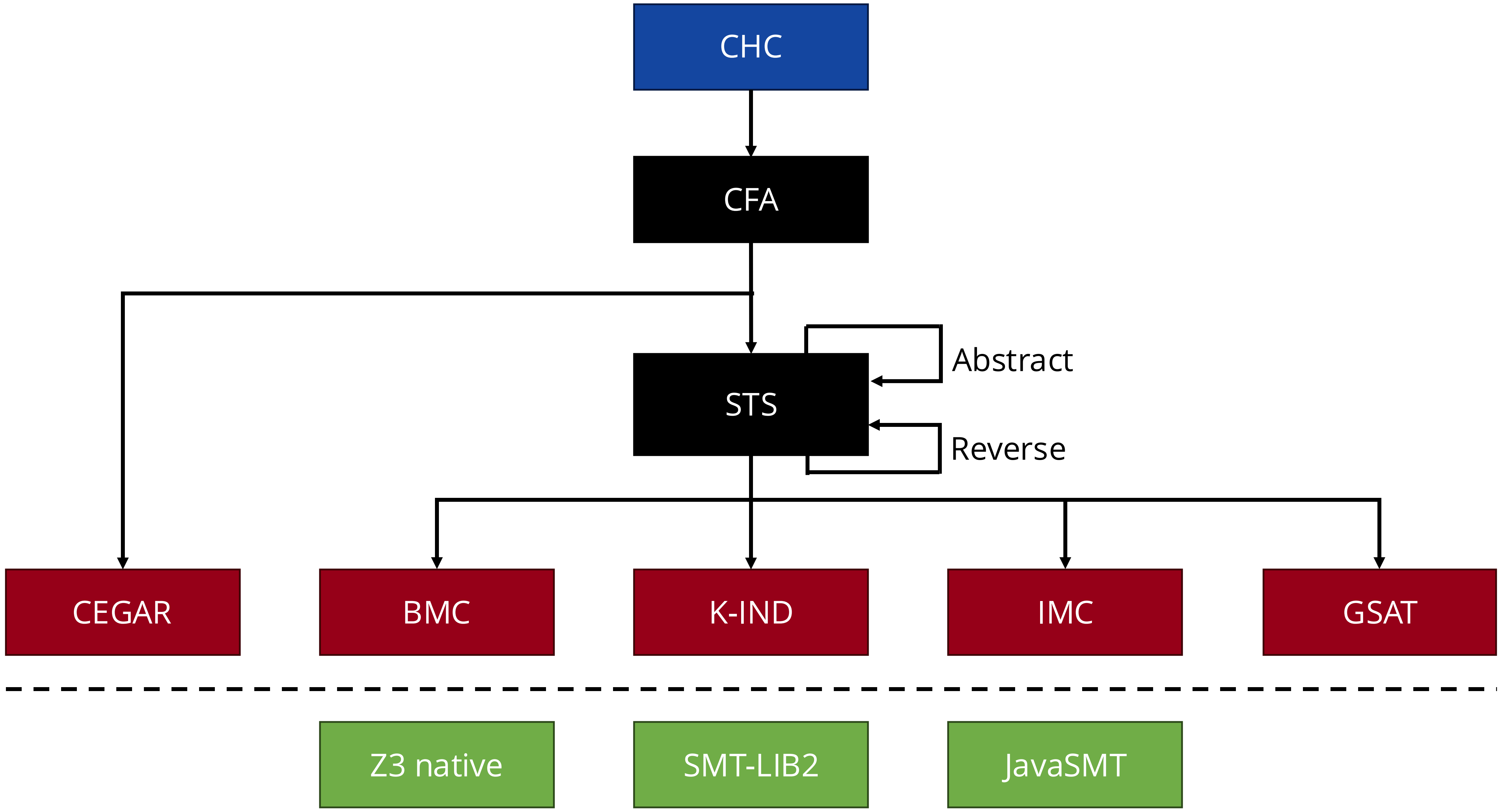


Substitution diagram


Node: SMT formula
(a ∨ ¬b) ∧ (x = 2)
substitute **a** with **false**
f
t
substitute **a** with **true**
a
¬b ∧ (x = 2)
f
b
(x = 2)
Equivalent nodes are merged
2
x
true

Lazy evaluation: presence of edges and children evaluated only when queried!

syntactically or with an **SMT-solver**


CHC
CFA
STS
Abstract
Reverse
CEGAR
BMC
K-IND
IMC
GSAT
Z3 native
SMT-LIB2
JavaSMT

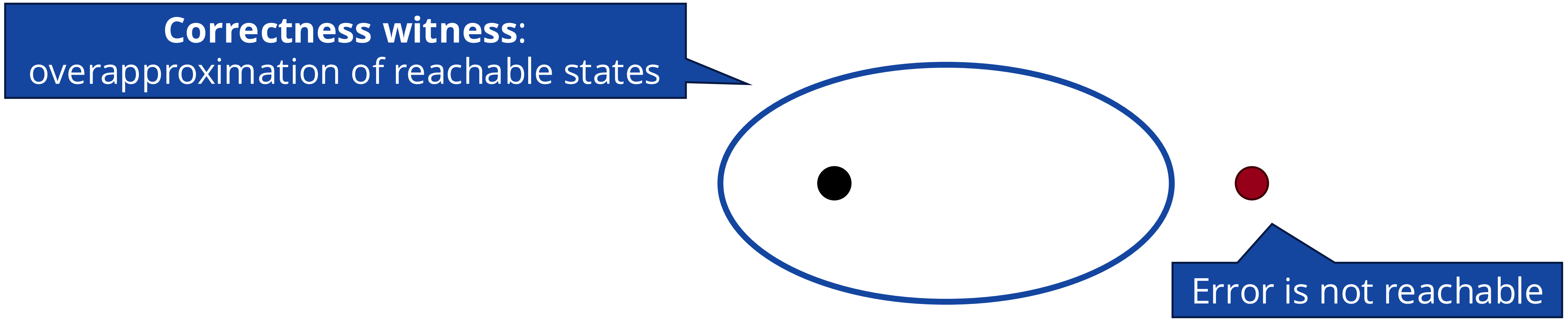
SMT Solver Backends

- Used by all algorithms
- Satisfiability (+ Model generation, UNSAT cores, Interpolation)

- Unified access through a **common interface**
  - **Native Z3** (best performance, strong interpolation)
  - **SMT-LIB2**: cvc5, MathSat, Princess, SMTInterpol, Bitwuzla, Boolector, …
  - **JavaSMT**: common Java API over several 3rd party solvers

CHC Model Generation

- Some of our algorithms provide an **overabstraction** of the reachable **state space**
  - CEGAR: returns the **ARG** (abstract reachability graph)
  - GSAT: returns the precise reachable state space as an **MDD**
  - IMC: provides an **inductive invariant formula**



CHC Model Generation

$F(x)$

UF with parameter x

$loc = F \rightarrow x = y + 2 \wedge y = z + 1$

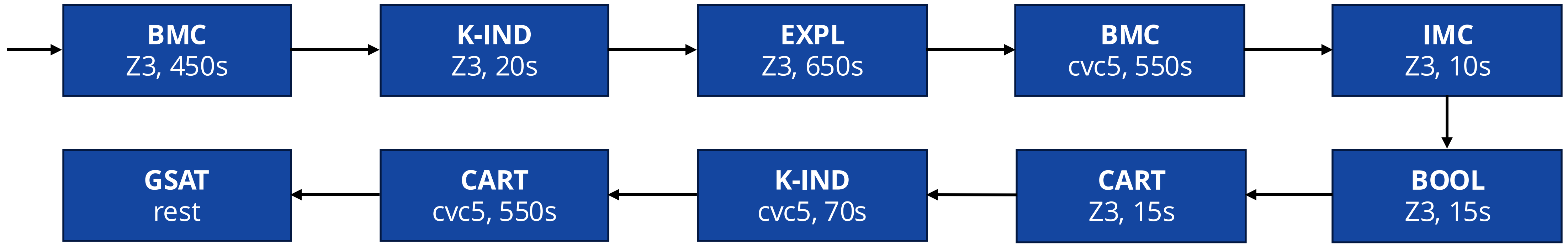
correctness witness formula for F

$\downarrow$

$$\underbrace{\forall x: F(x)}_{\text{universally quantified UF parameters}} \leftrightarrow \overbrace{\exists y,z:}^{\text{existentially quantified other variables}} x = y + 2 \wedge y = z + 1$$

CHC-COMP'25 portfolio

Sequential portfolio, change to next config on timeout or exception



CHC-COMP'25 results

- We fixed several bugs in Theta since
  - Variable name collisions, loop unrolling
- If the competition was rerun, some rankings would change:

		LIA	LIA-Lin	LIA-Arrays	LIA-Lin-Arrays	LRA-Lin	BV
comp	sat	52	114	400	45	73	49
	unsat	140	376	8	4	18	123
	rank	5	8	5	4	3	2
fixed	sat	48	585	440	**63**	76	42
	unsat	136	402	12	18	16	126
	rank	5	5	4	1	3	2

We are currently the best tool for **sat linear array** problems

CHC-COMP'25 model generation results

Category	Eldarica	Golem	ThetaCHC
LIA	378	709	0
LIA-Lin	623	675	565
LIA-Arrays	1000	-	0
LIA-Lin-Arrays	52	-	55
LRA-Lin	0	73	11
BV	17	-	22

We currently do not support non-linear CHC model generation, but have an implementation mind

Summary

github.com/ftsrg/theta

- Forward/backward CHC to CFA transformations
- Diverse algorithm backends with chainable model transformations
- CHC **model generation** from correctness witnesses
- Sequential algorithm **portfolio** for CHC-COMP'25

ThetaCHC

Model checker solving **CHC** through **transformations to CFA**

Forward CHC to CFA Transformation

- Unique to Theta

M. Somorjai et al.: **Bottoms Up for CHCs: Novel Transformation of Linear Constrained Horn Clauses to Software Verification**
In: HCVS 2023.

CHC		**CFA**
uninterpreted functions	⟷	nodes
parameters	⟷	variables
CHCs	⟷	edges

⬇

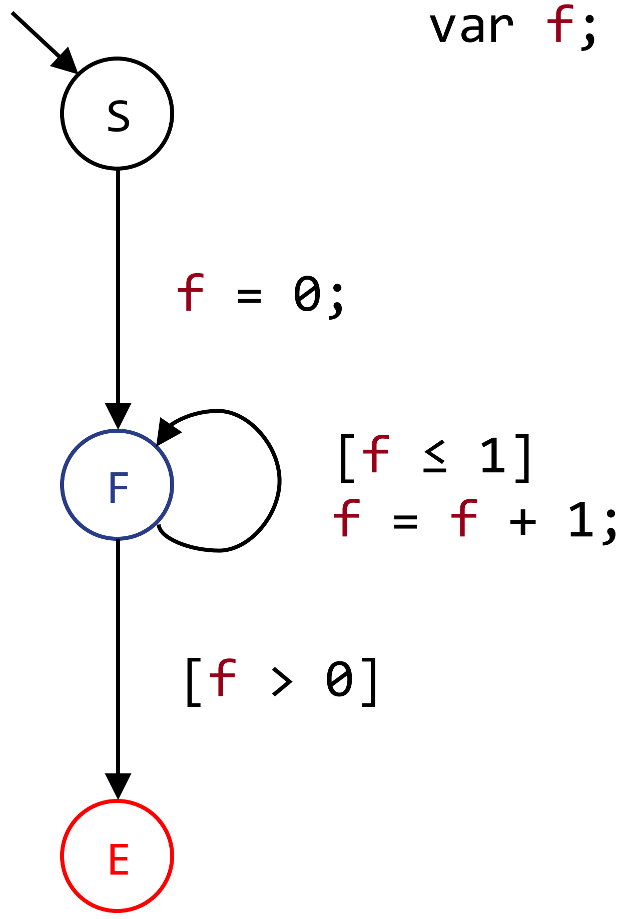
satisfiability ⟷ **reachability**

Example

$F(x) \leftarrow x = 0$

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

$\bot \leftarrow F(x) \land x > 0$



Backward CHC to CFA Transformation

- Unihorn’s approach adapted to CFA

CHC		**CFA**
uninterpreted functions	↔	procedures
parameters	↔	procedure parameters
CHCs	↔	edges

↓

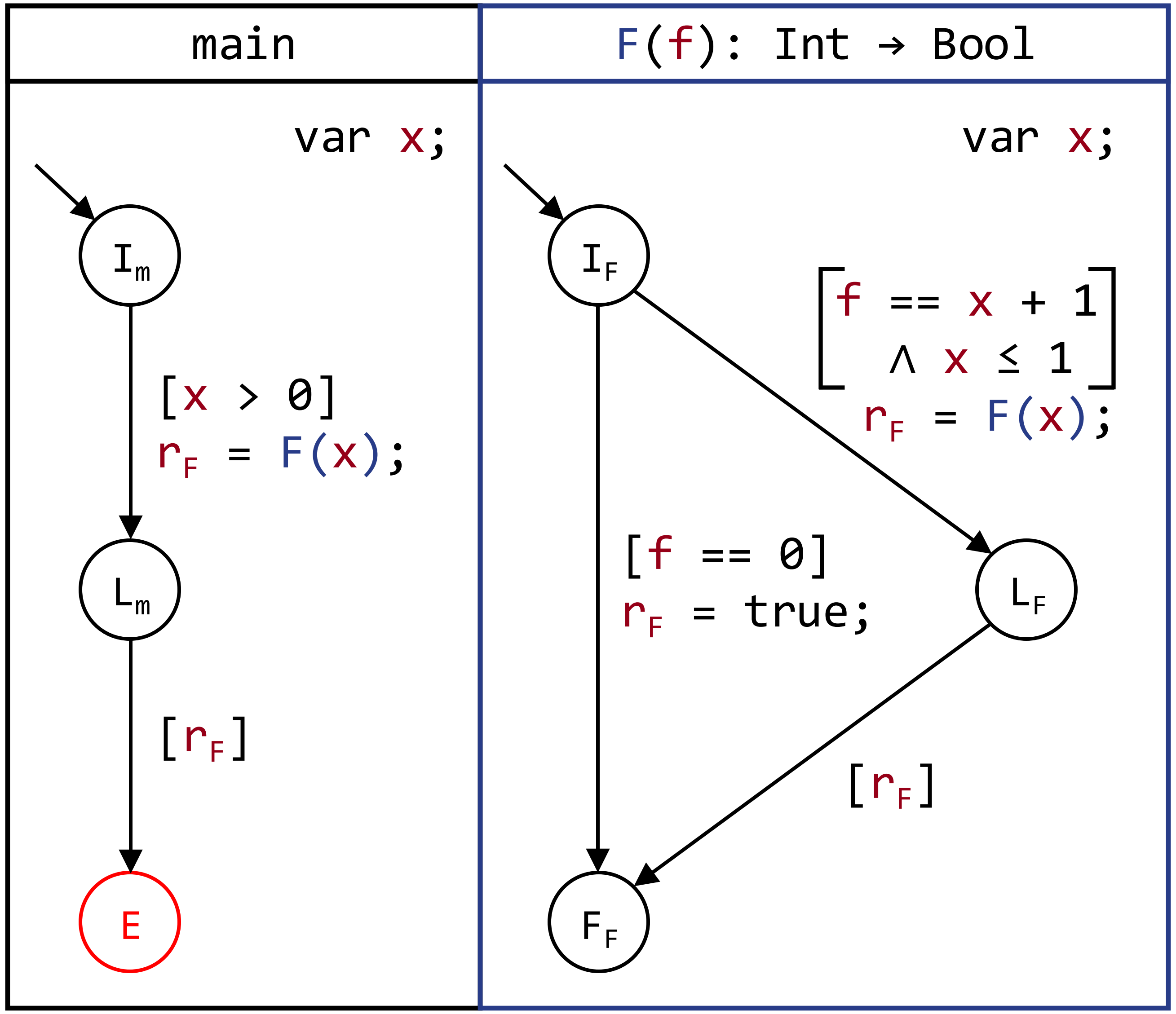
satisfiability ↔ **return value**

Example

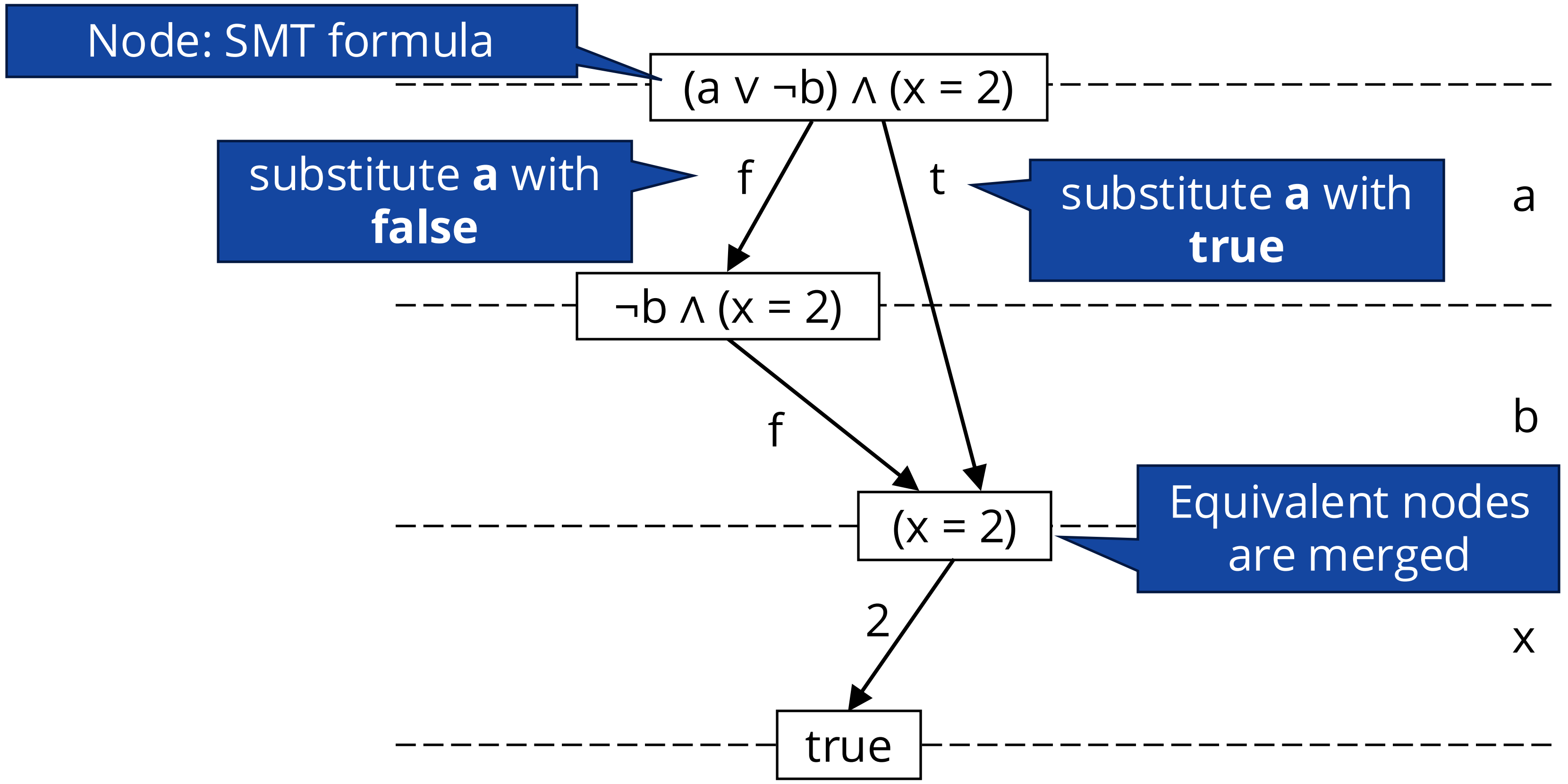
$F(x) \leftarrow x = 0$

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

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



Substitution diagram


Node: SMT formula
(a ∨ ¬b) ∧ (x = 2)
substitute **a** with **false**
f
t
substitute **a** with **true**
a
¬b ∧ (x = 2)
f
b
(x = 2)
Equivalent nodes are merged
2
x
true

Lazy evaluation: presence of edges and children evaluated only when queried!

syntactically or with an **SMT-solver**

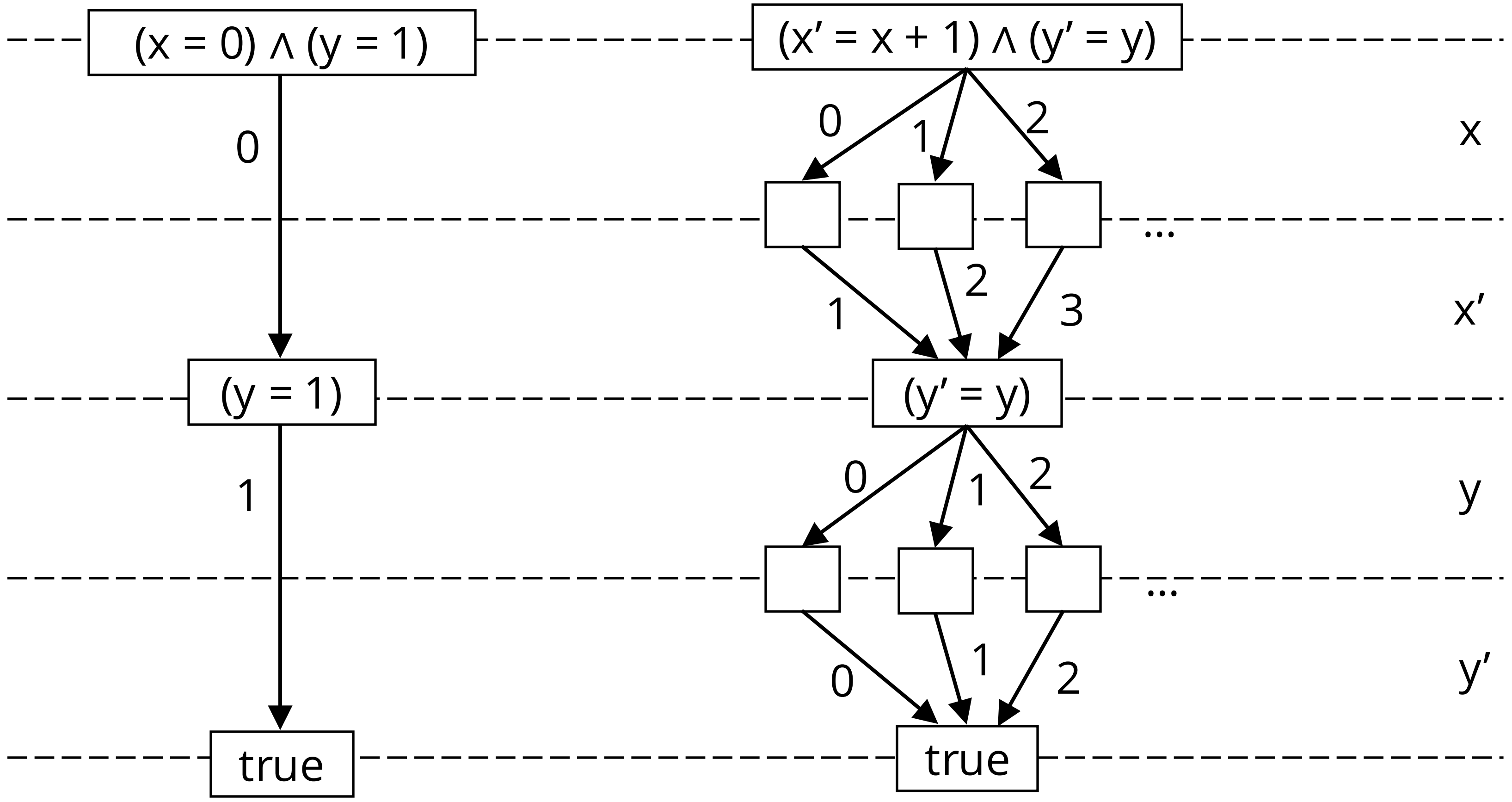
Model checking with substitution diagrams

Initial states

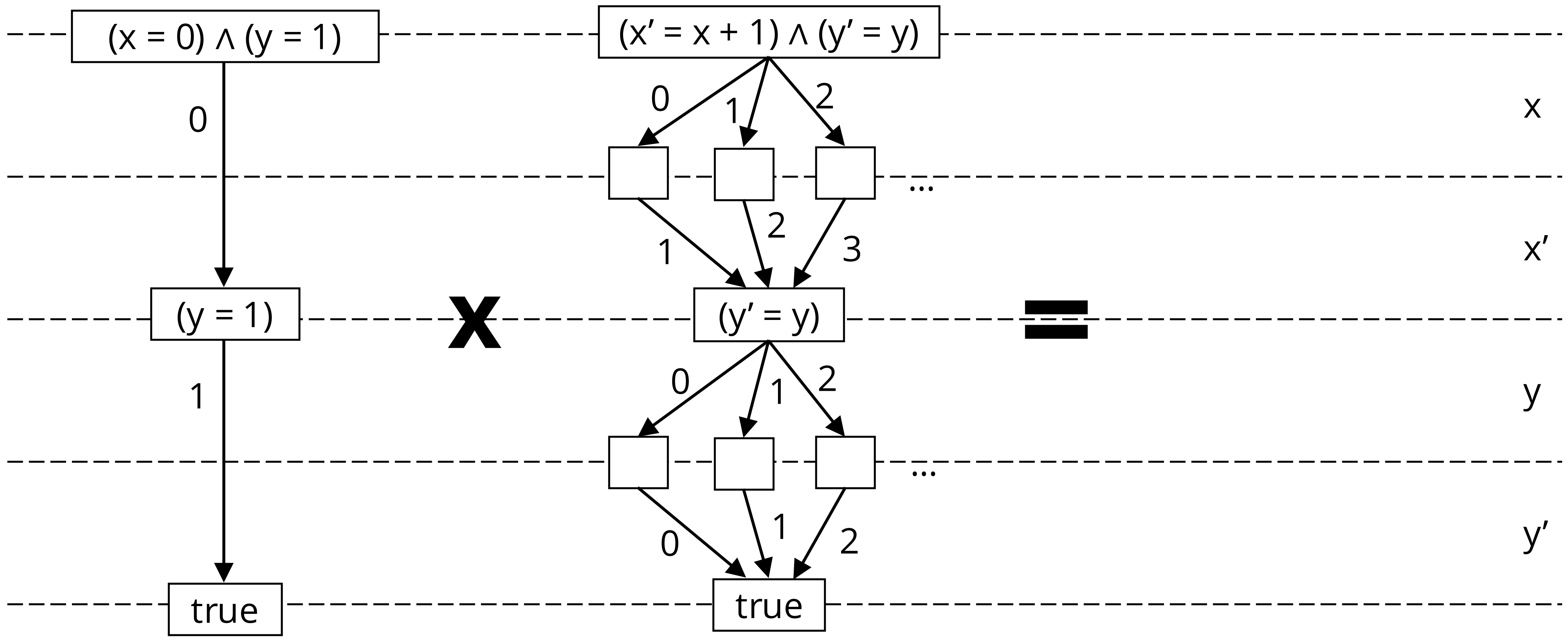
I: $(x = 0) \wedge (y = 1)$

T: $(x' = x + 1) \wedge (y' = y)$

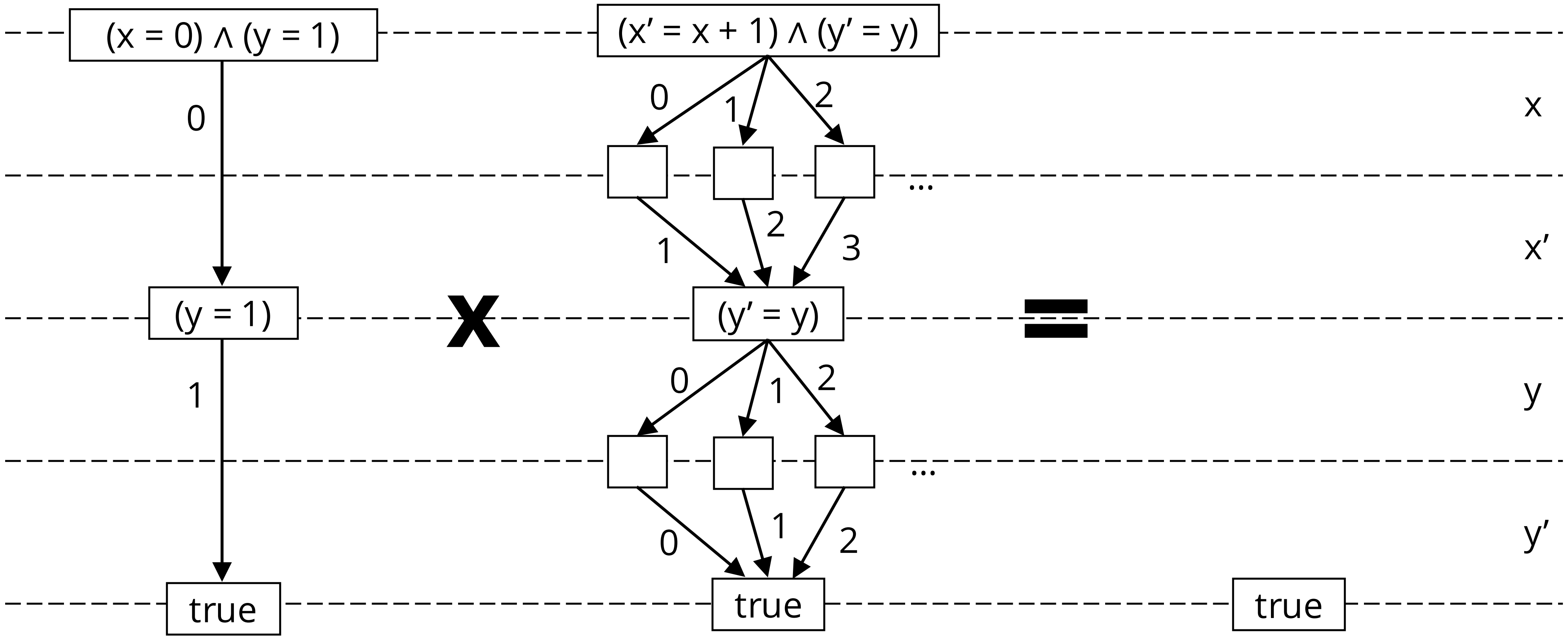
Transition relation



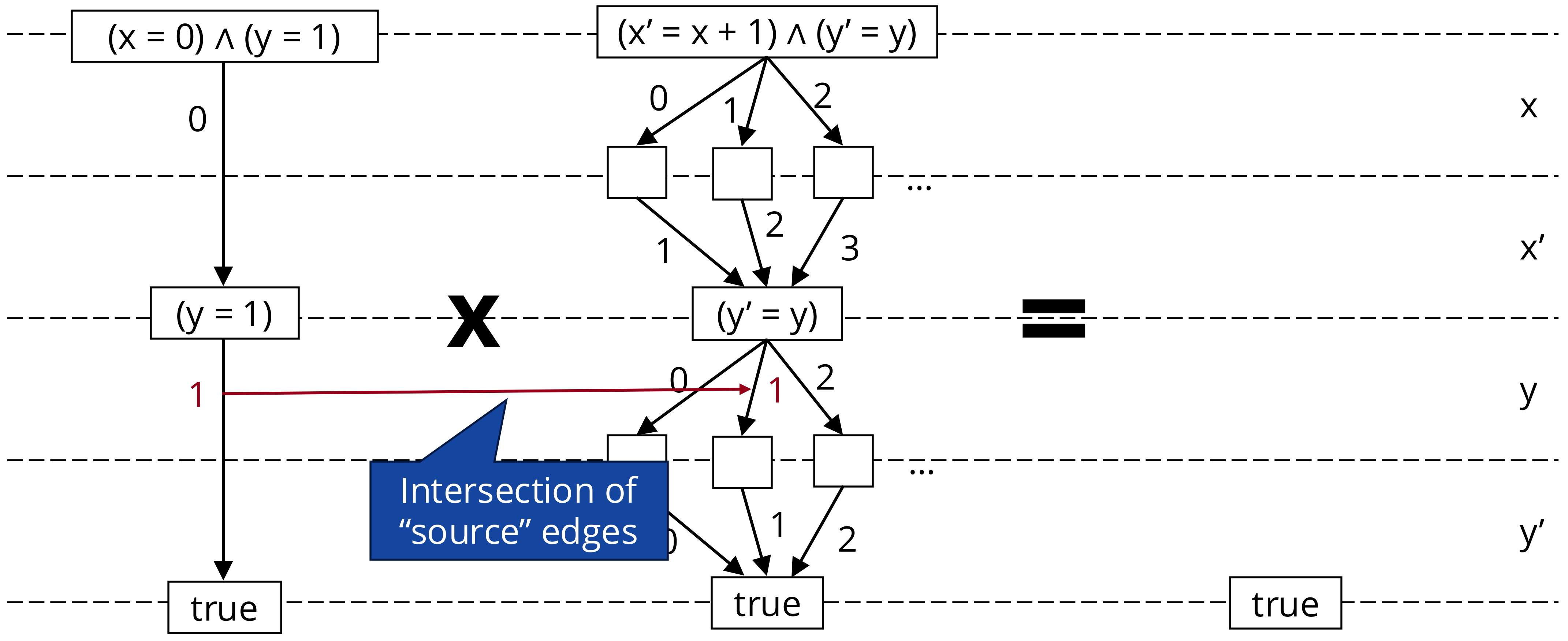
Relational product: model step



Relational product: model step

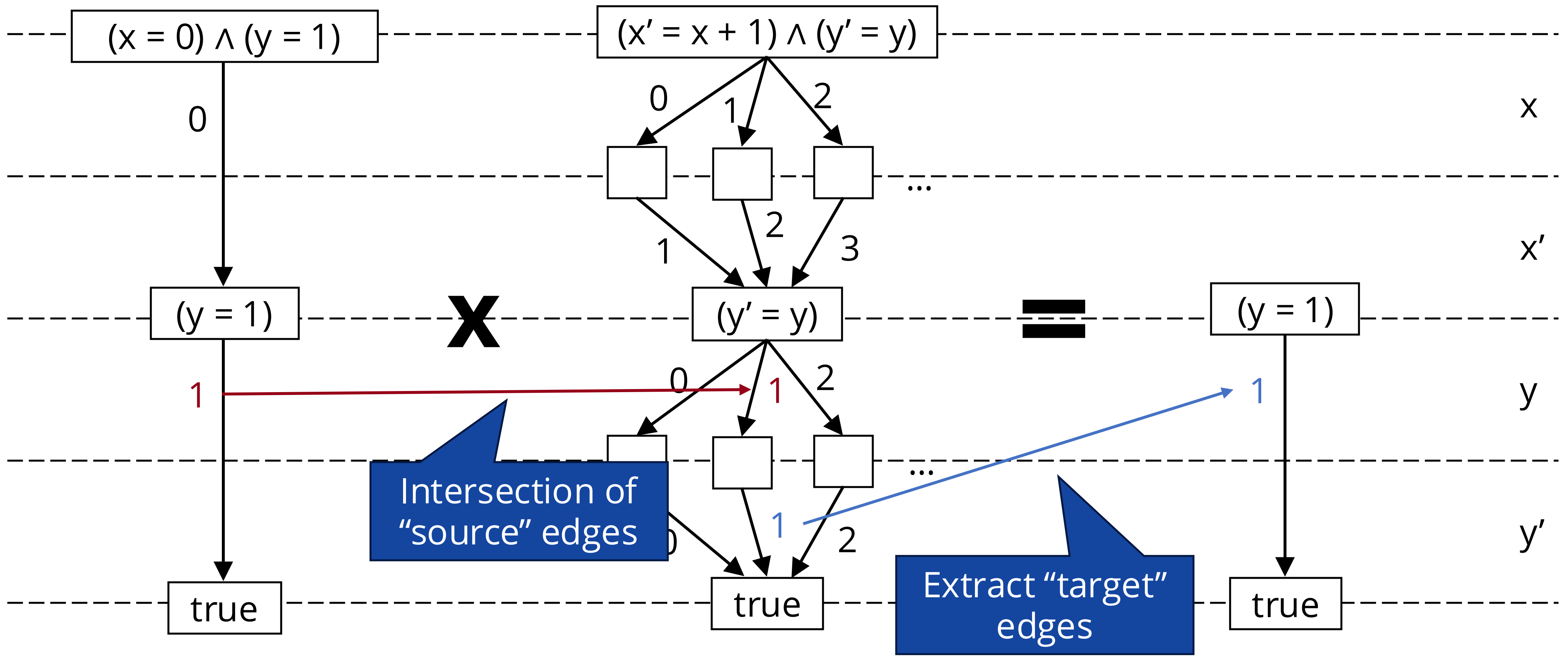


Relational product: model step

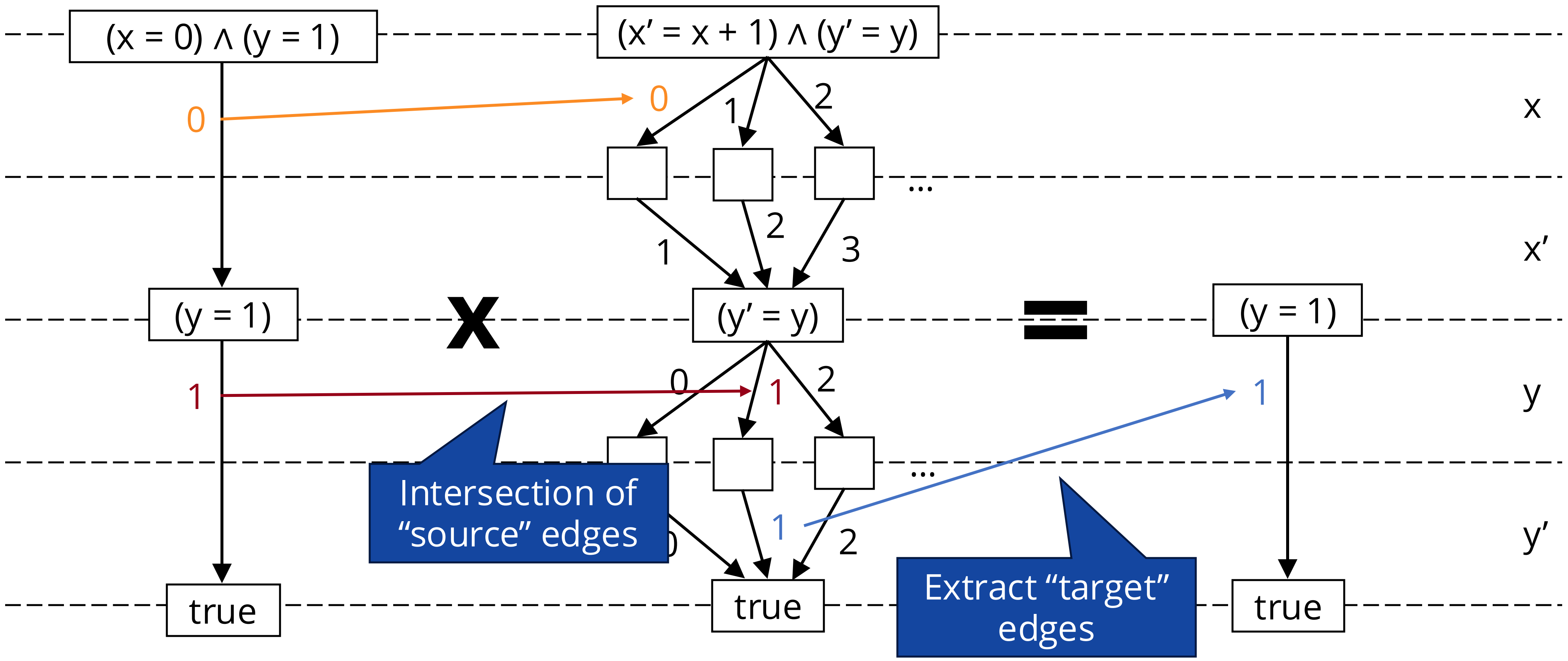


Intersection of "source" edges

Relational product: model step



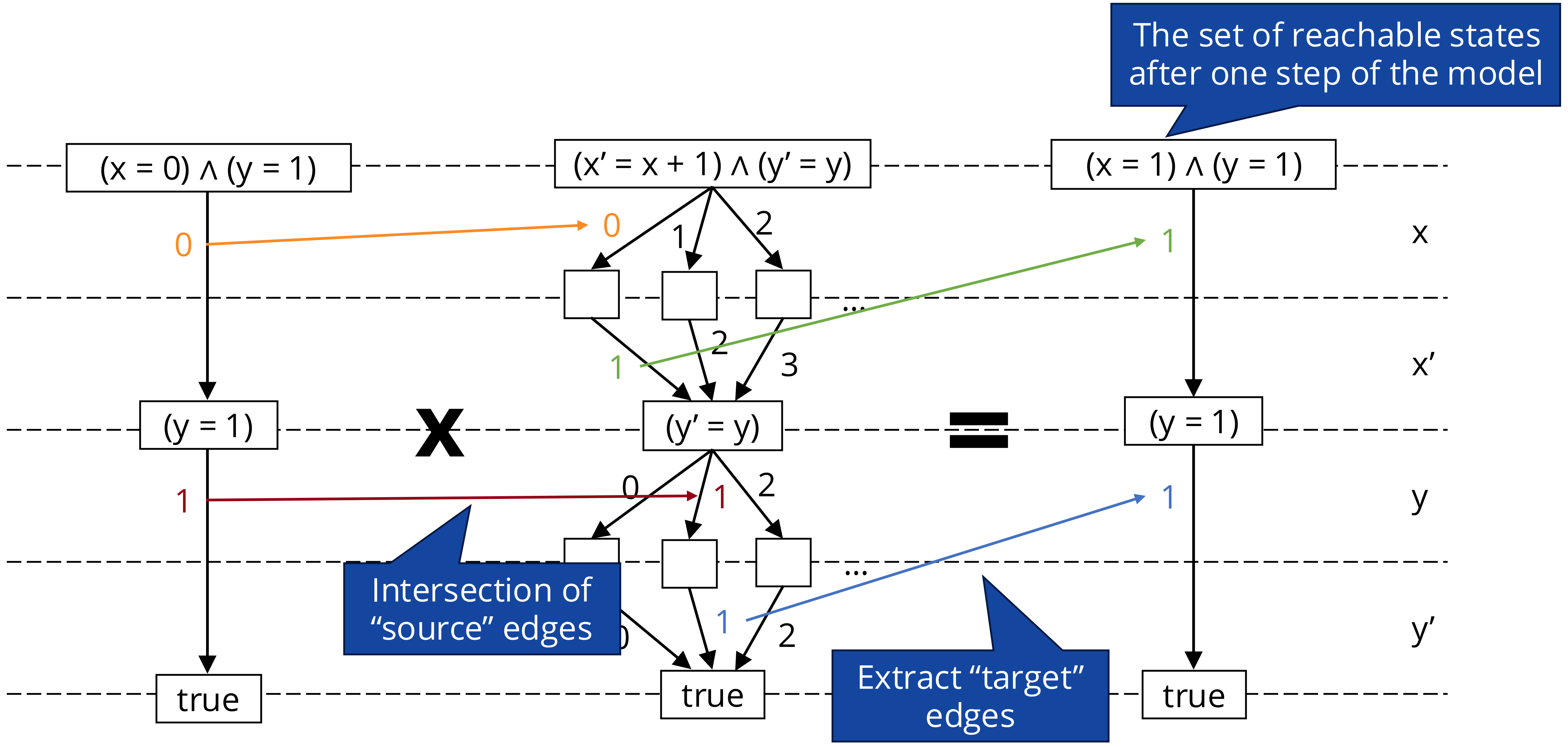
Relational product: model step



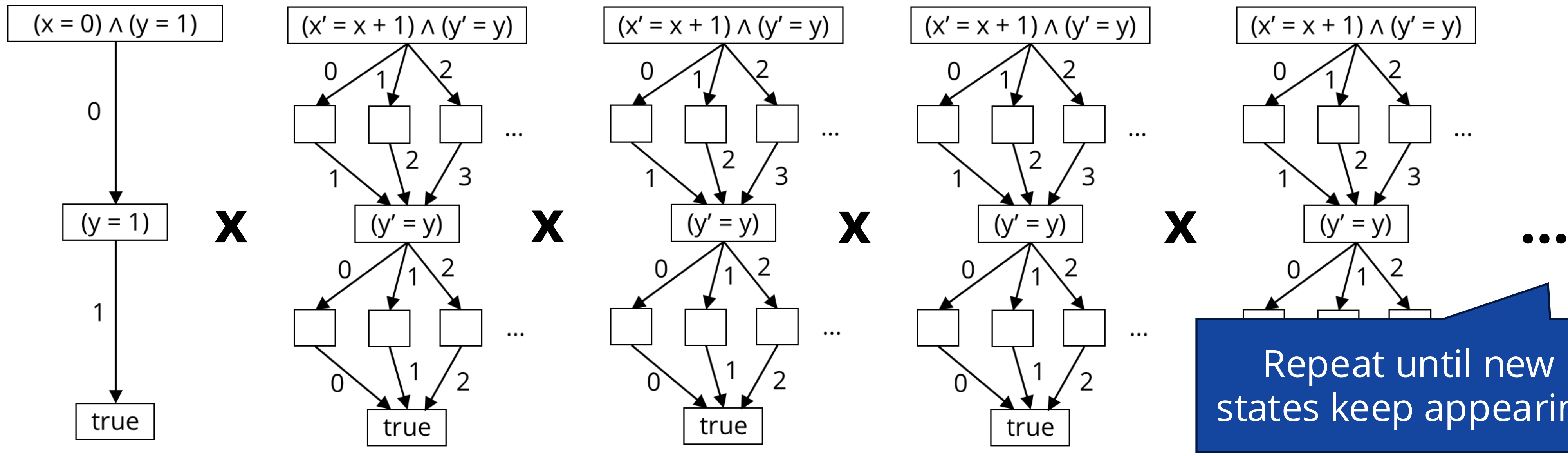
Intersection of “source” edges

Extract “target” edges

Relational product: model step



Fixed point calculation



Repeat until new states keep appearing

Many possible algorithms: BFS, Saturation