Thorn: CHC-Based Software Verification

Levente Bajczi

April 11, 2026

MŰEGYETEM 1782

BME FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS

ftsrg Critical Systems Research Group

What is Thorn?

Thorn Theta config delegating to CHC solvers

What is Thorn?

Thorn — Theta config delegating to CHC solvers

Constrained Horn Clauses

- Logical formulas of the form
  $\varphi \wedge p_1(\bar{x}_1) \wedge ... \wedge p_n(\bar{x}_n) \implies h(\bar{x})$
- Uninterpreted predicates $\rightarrow$ invariants
- Solvers: **Eldarica**, **Golem**, **Z3**

What is Thorn?

Thorn — Theta config delegating to CHC solvers

Constrained Horn Clauses

- Logical formulas of the form $\varphi \wedge p_1(\bar{x}_1) \wedge ... \wedge p_n(\bar{x}_n) \implies h(\bar{x})$
- Uninterpreted predicates $\rightarrow$ invariants
- Solvers: **Eldarica**, **Golem**, **Z3**

SV-COMP 2026 Categories

- ReachSafety
- MemSafety
- NoOverflows
- **Concurrency** (novel for CHC)
- **Termination** (novel encoding)

CHC Encoding

Program as Control Flow Automaton

- Locations $\rightarrow$ uninterpreted predicates $L(\bar{v})$
- Edges $\rightarrow$ implications:

$$L_{src}(\bar{v}) \wedge \mathit{edge}(\bar{v}, \bar{v}') \implies L_{dst}(\bar{v}')$$

CHC Encoding

Program as Control Flow Automaton

- Locations $\rightarrow$ uninterpreted predicates $L(\bar{v})$
- Edges $\rightarrow$ implications:

$$L_{src}(\bar{v}) \wedge \mathit{edge}(\bar{v}, \bar{v}') \implies L_{dst}(\bar{v}')$$

Boundary conditions

- Init: $\top \implies L_0(\bar{v})$
- Error: $L_{err}(\bar{v}) \implies \bot$

CHC Encoding

Program as Control Flow Automaton

- Locations $\rightarrow$ uninterpreted predicates $L(\bar{v})$
- Edges $\rightarrow$ implications:

$$L_{src}(\bar{v}) \wedge \mathit{edge}(\bar{v}, \bar{v}') \implies L_{dst}(\bar{v}')$$

Boundary conditions

- Init: $\top \implies L_0(\bar{v})$
- Error: $L_{err}(\bar{v}) \implies \bot$

Example

```
int x = 0;          // L0
while (x < 5)       // L1
    x++;             // L2
assert(x == 5);     // L3 (err if !=)
```

CHC system:

$$\begin{aligned}
\top &\implies L_0(0) \\
L_0(x) &\implies L_1(x) \\
L_1(x) \wedge x < 5 &\implies L_2(x) \\
L_2(x) &\implies L_1(x+1) \\
L_1(x) \wedge x \geq 5 &\implies L_3(x) \\
L_3(x) \wedge x \neq 5 &\implies \bot
\end{aligned}$$

Novel Extensions

Concurrency Bounded interl.

Novel Extensions

Concurrency | Bounded interl.

- Threads statically determined at analysis time
- Each thread → dedicated location variable
- Interleavings → nondeterministic choice
- Atomic blocks → mutex variable

→ **Enables CHC-based tools in the Concurrency category**

Novel Extensions

Concurrency Bounded interl.

- Threads statically determined at analysis time
- Each thread → dedicated location variable
- Interleavings → nondeterministic choice
- Atomic blocks → mutex variable

→ **Enables CHC-based tools in the Concurrency category**

Termination Ranking functions

Novel Extensions

Concurrency Bounded interl.

- Threads statically determined at analysis time
- Each thread $\rightarrow$ dedicated location variable
- Interleavings $\rightarrow$ nondeterministic choice
- Atomic blocks $\rightarrow$ mutex variable

$\rightarrow$ **Enables CHC-based tools in the Concurrency category**

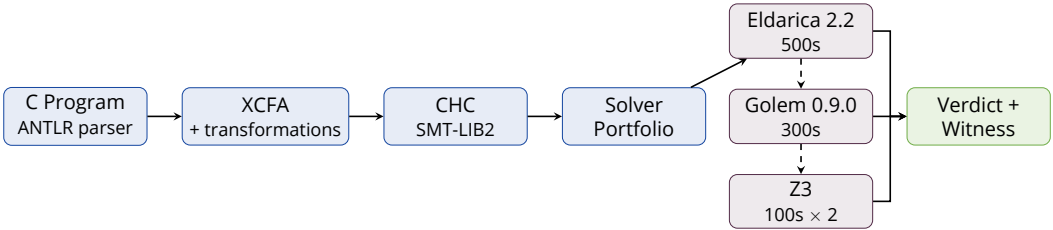
Termination Ranking functions

- Introduce step counter k
- Assert no state revisited:

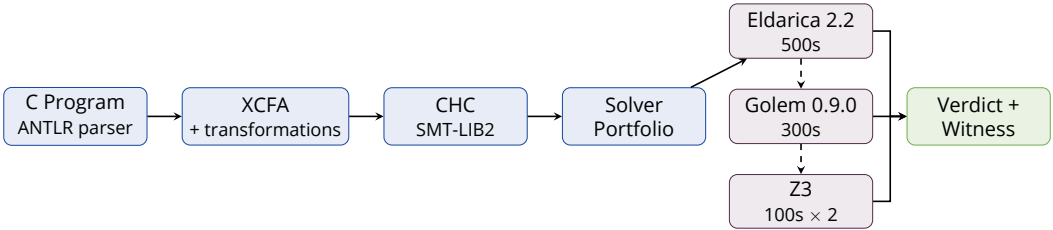
$$L(\bar{v}, k) \wedge L(\bar{v}, k') \wedge k \neq k' \implies \bot$$

- SAT model $\rightarrow$ ranking function $g = k_{\max} - f(\mathit{loc}, \bar{v})$
- UNSAT $\rightarrow$ lasso-shaped counterexample (nontermination)

Architecture



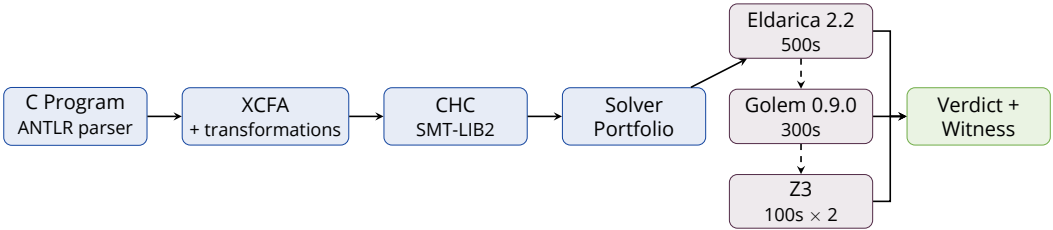
Architecture



Portfolio strategy

- Solvers tried sequentially
- Timeout/error → next solver
- Eldarica & Golem prioritized: cleaner proofs & models

Architecture



Portfolio strategy

- Solvers tried sequentially
- Timeout/error → next solver
- Eldarica & Golem prioritized: cleaner proofs & models

Witness generation

- SAT model → location invariants (correctness witness)
- UNSAT proof → counterexample trace (violation witness)

Summary

Strengths	Proofs, concurrency

Summary

Strengths Proofs, concurrency

- CHC solvers excel at invariant synthesis
- Elegant termination encoding
- Static concurrency support

Summary

Strengths Proofs, concurrency

Weaknesses FP, dyn. threads

- CHC solvers excel at invariant synthesis
- Elegant termination encoding
- Static concurrency support

Summary

Strengths Proofs, concurrency

- CHC solvers excel at invariant synthesis
- Elegant termination encoding
- Static concurrency support

Weaknesses FP, dyn. threads

- No floating-point support
- Unbounded thread creation not supported
- May struggle with large programs/bitvectors
- Transition invariants may improve termination

Summary

Strengths Proofs, concurrency

- CHC solvers excel at invariant synthesis
- Elegant termination encoding
- Static concurrency support

Weaknesses FP, dyn. threads

- No floating-point support
- Unbounded thread creation not supported
- May struggle with large programs/bitvectors
- Transition invariants may improve termination

Part of Theta — open-source, Apache 2.0

github.com/ftsrg/theta theta.mit.bme.hu