Korn

A C verifier based on Horn-clauses

https://github.com/gernst/korn

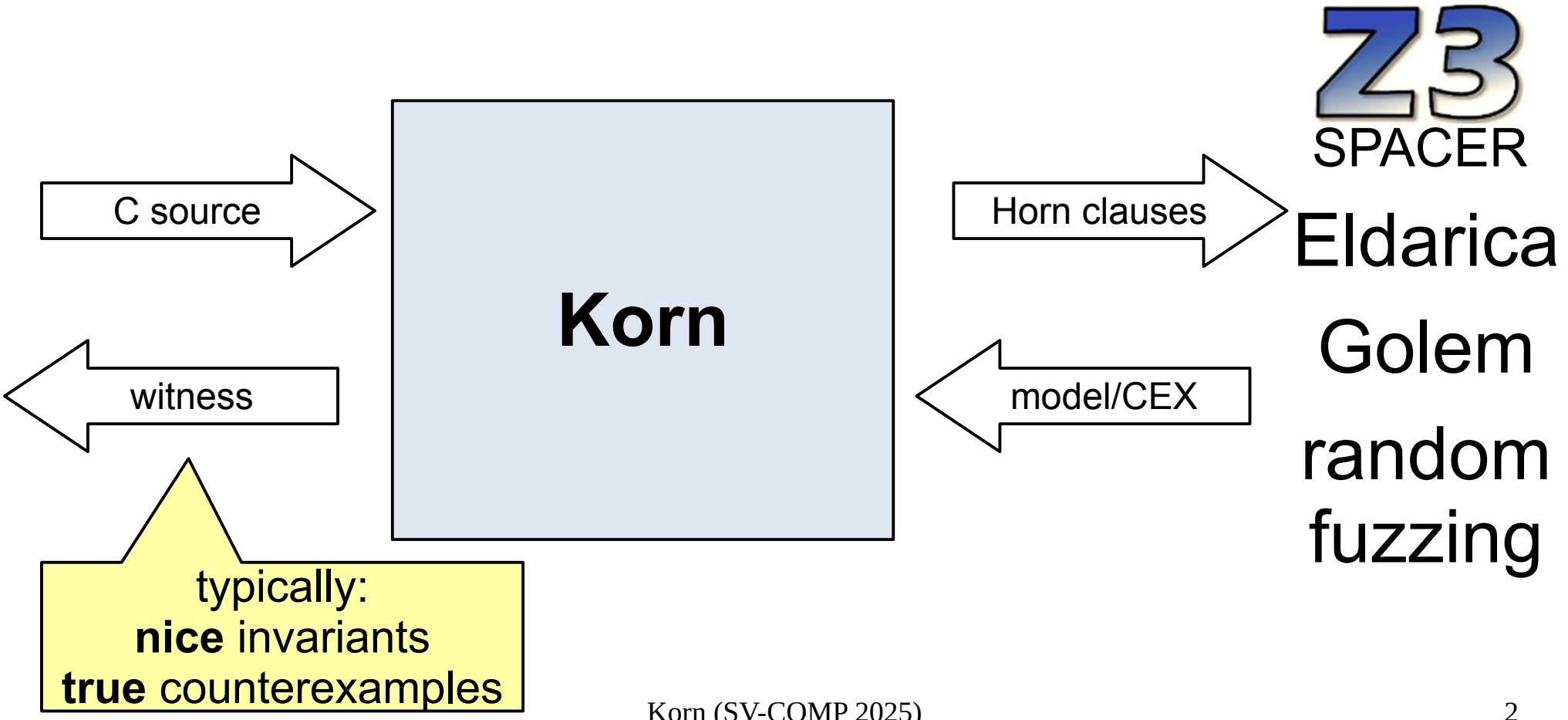
SV-COMP 2025, May 5

Gidon Ernst gidon.ernst@lmu.de
Martin Blicha martin.blicha@usi.ch

Scope: ReachSafety (simple C: integers, arrays, no heap)

Approach: Portfolio with off-the-shelf CHC solvers

Korn: Architecture



Horn-clause based Verification

(well-known, e.g. [Bjørner, Gurfinkel, McMillan, Rybalchenko 2015])

```
assume(i ≤ 0);
int i = 0;

while(i < n) {
  i++;
}

assert(i == n);
```

$$\exists\ inv.$$

second order

$$0 \leqslant n \land i=0 \implies inv(i,n)$$

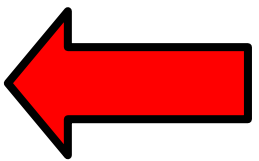
$$i < n \land inv(i,n) \implies inv(i+1,n)$$

$$\neg(i < n) \land inv(i,n) \implies i = n$$

Cheap Random Fuzzing

compile and run for the fun

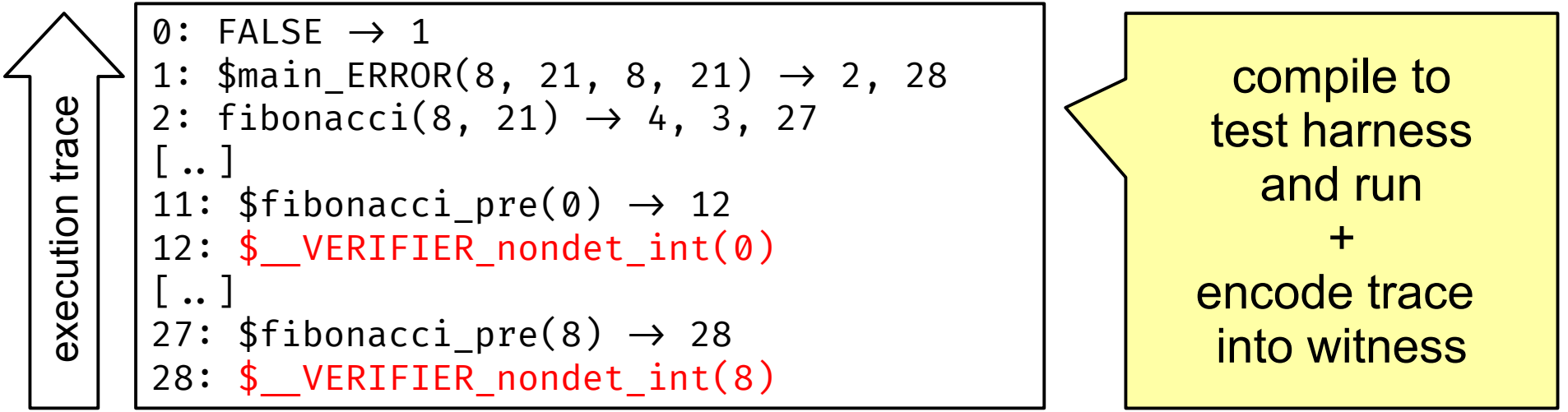
- Many sv-benchmarks falsify with `__VERIFIER_nondet_*()` **small**
- Heuristic: uniform choice between a value in
  0 [0,1] [0,31] [0,1023]
-
  - ⊕ O(100) problems solved in ~2s each
  - ⊖ misses CEX in ReachSafety-XCSP



Counterexample Validation

don't trust encoding and solvers

- Horn-clauses track `__VERIFIER_nondet_*()`

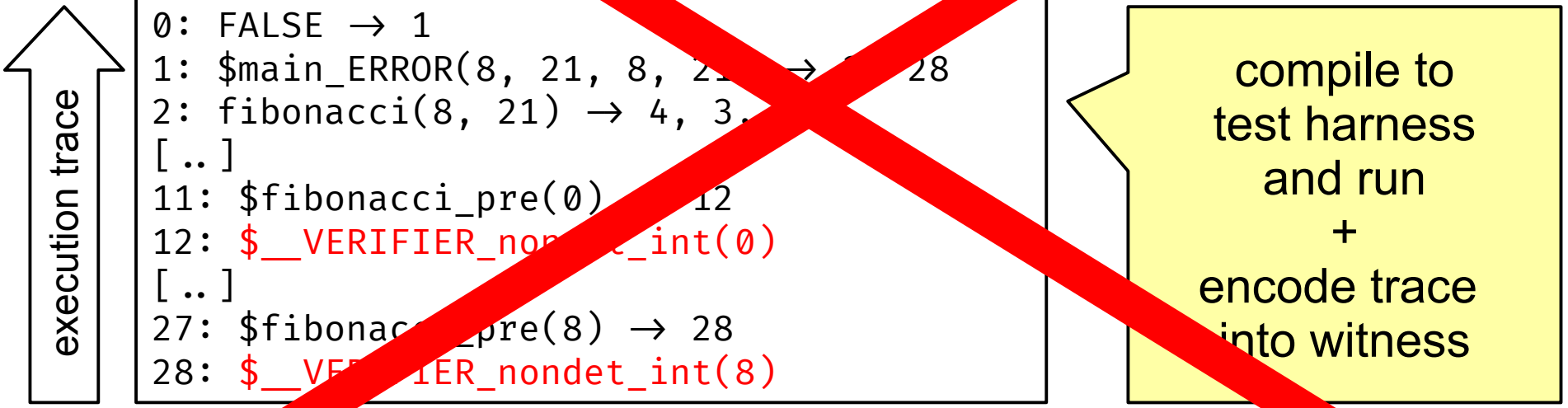


⊕ avoids a handful of incorrect false verdicts

Counterexample Validation

don’t trust encoding and solvers

- Horn-clause track `__VERIFIER_nondet_*()`



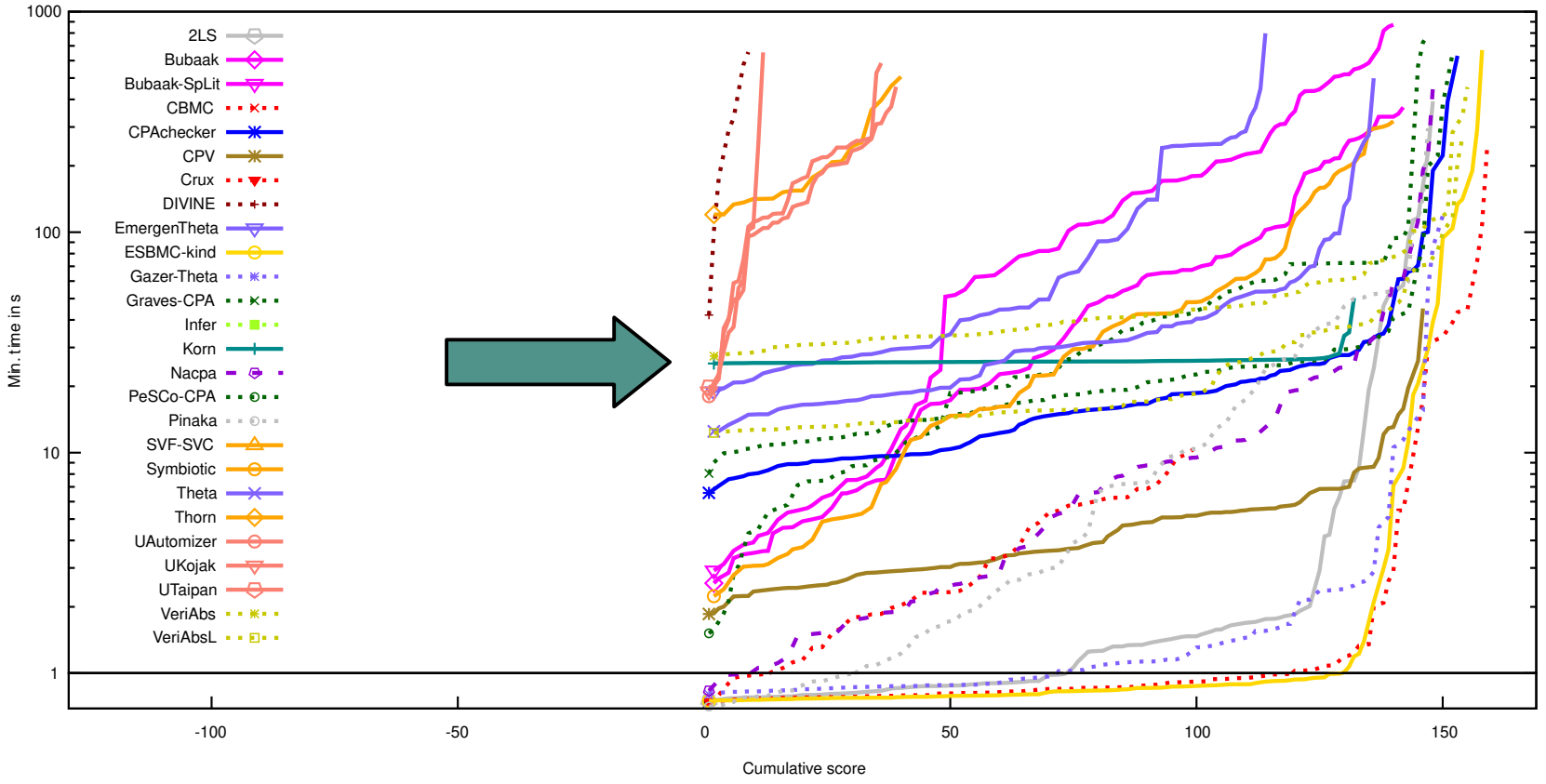
⊕ avoids a handful of incorrect false verdicts

Korn 2025 Update

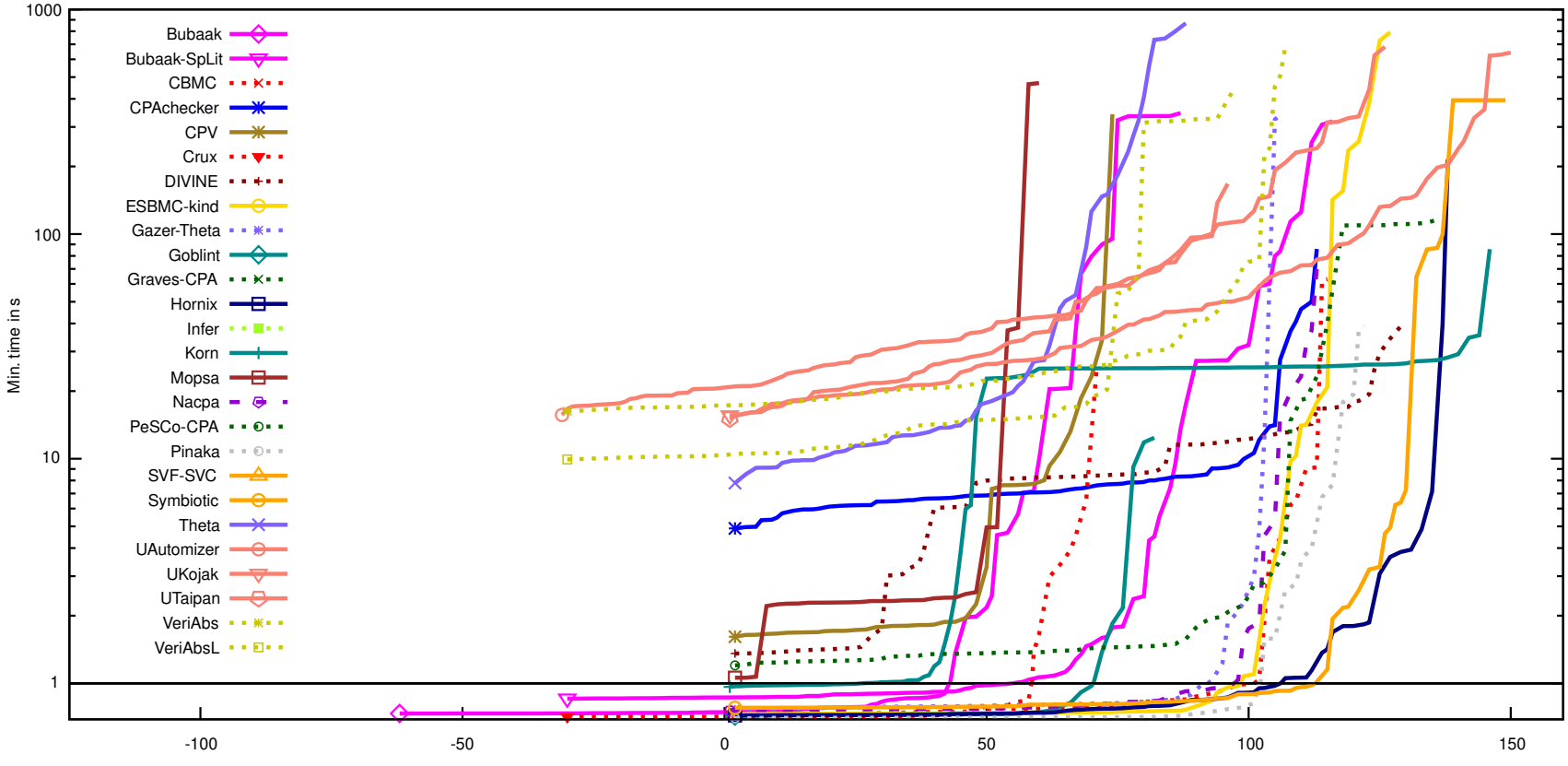
- Motivation: validate all CEX via testing
- CHC solvers give CEX as an **acyclic graph**
  - ⊖ Before: Korn often misinterpreted CEX order

- During ETAPS 2024
  - ⊕ figured out **guarantees** for trace format:
    solver must maintain order of premises, clauses
  - ⊕ figured out **linearization algorithm** for extraction

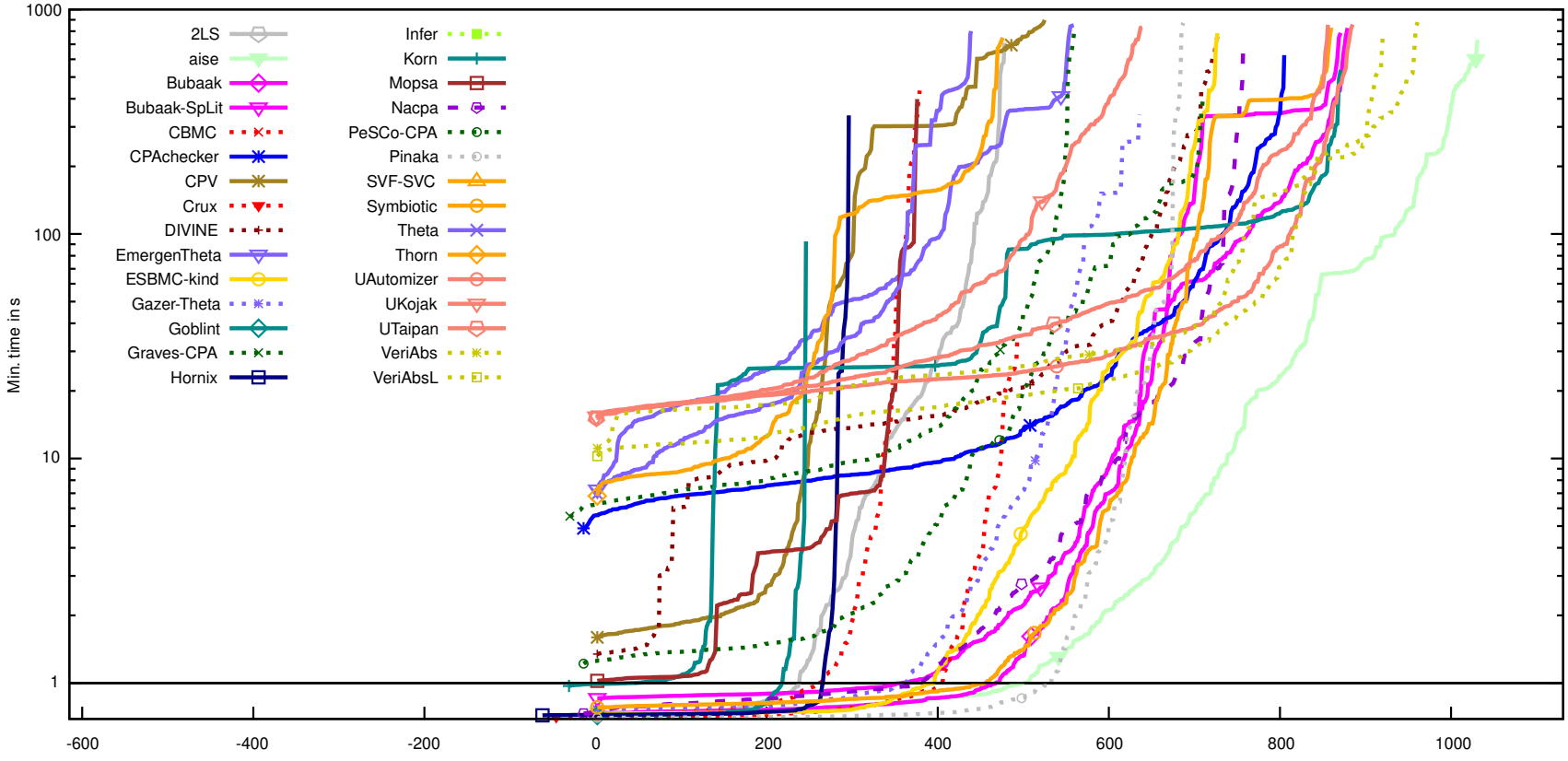
ReachSafety-XCSP



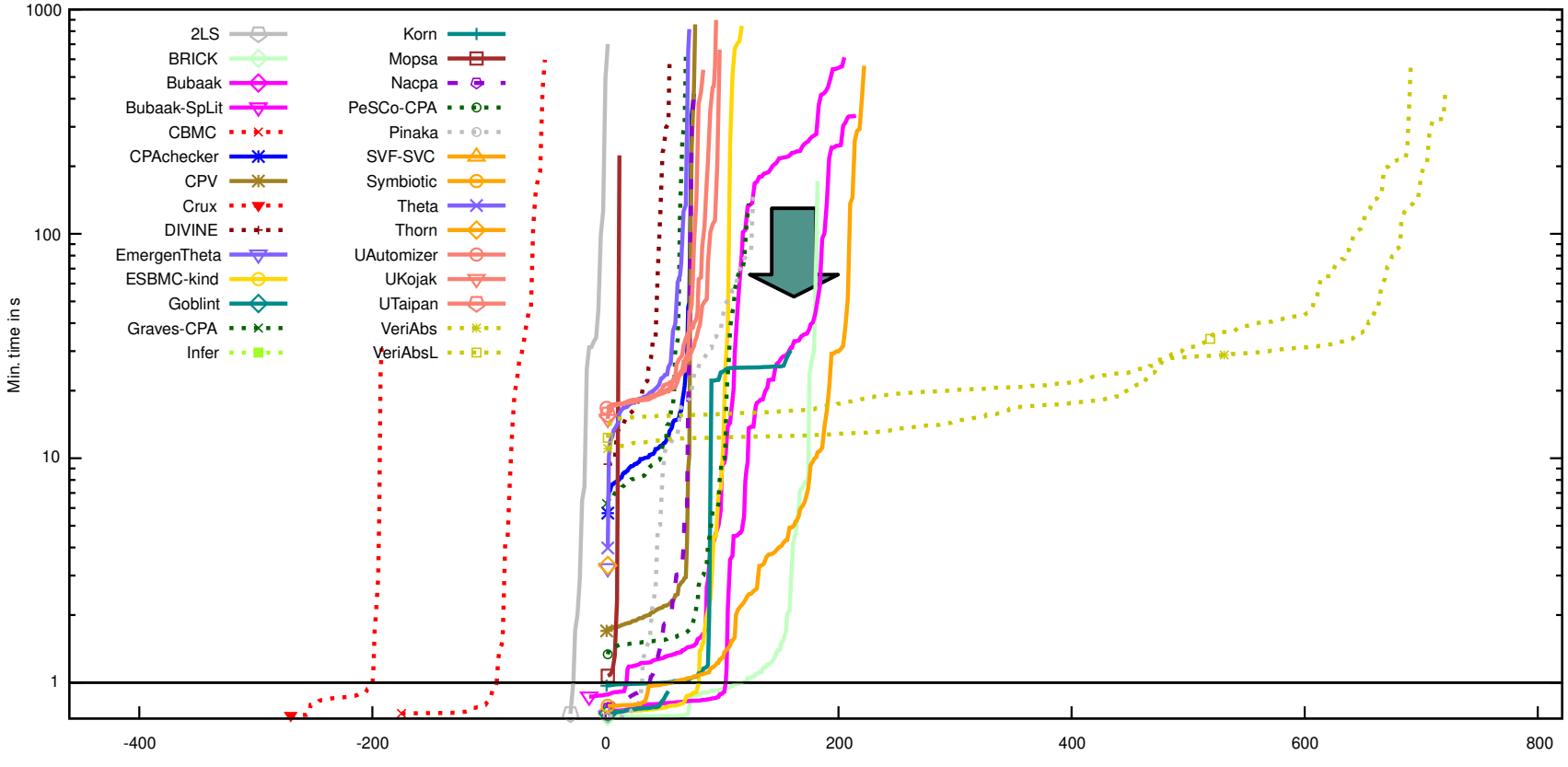
ReachSafety-Recursive



ReachSafety-Loops



ReachSafety-Arrays



Take-Away

https://github.com/gernst/korn

- Horn solvers effective for numeric benchmarks
- Portfolio pays off, including random sampling

- Horn clause encoding enforce **modular** proofs
  - ⊕ procedures + recursion easy
  - ⊖ (typically) doesn't take advantage of finite loop bounds