Korn

A C verifier based on Horn-clauses

https://github.com/gernst/korn

SV-COMP 2024, April 8

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

Korn - Background

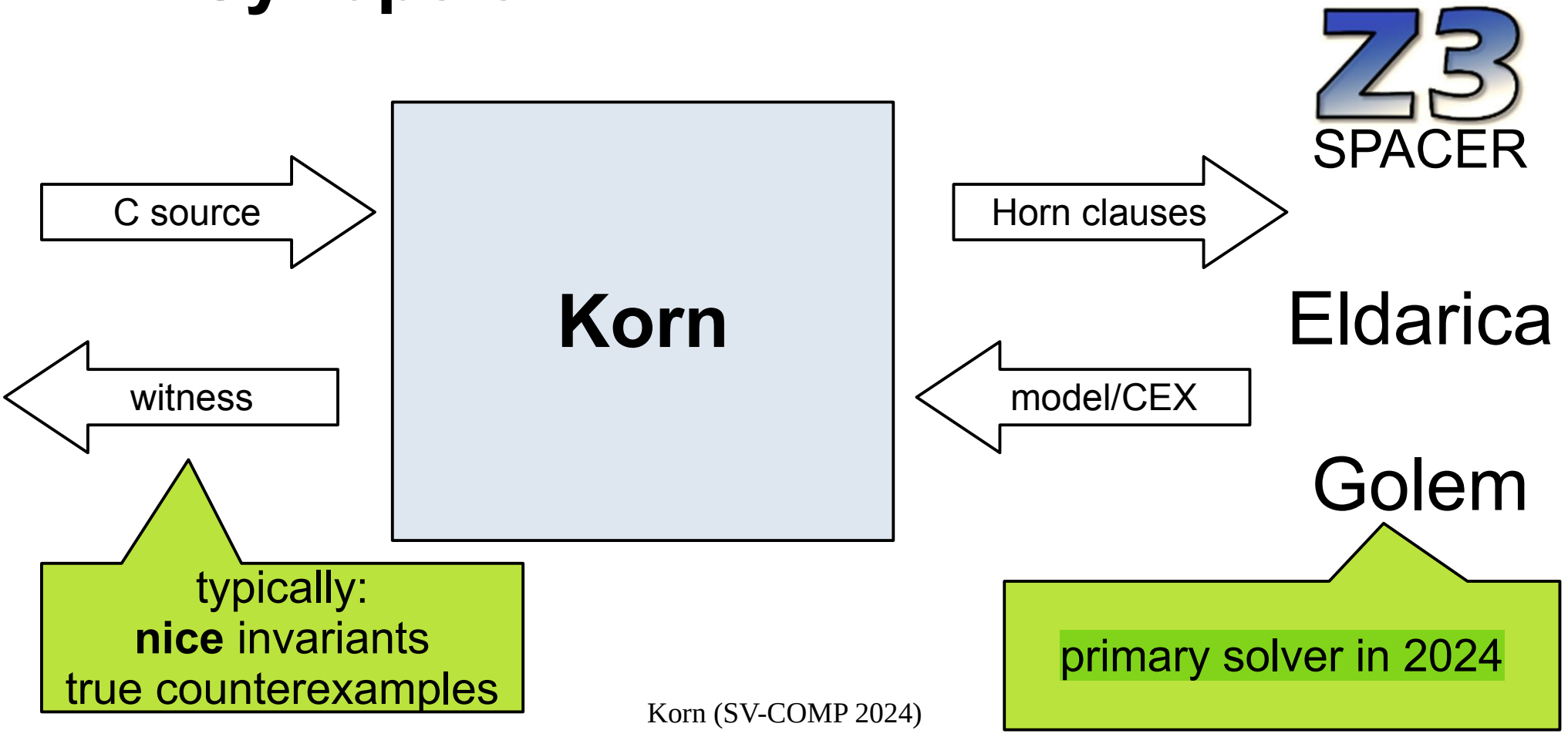
- Goal: investigate different loop encodings (notably: loop contracts, [VMCAI 2022])

  ⊕ easy to hack (Scala) ⊖ many C features missing

- SV-COMP: minor polishing over last year

- Participation: Recursive, Loops, ControlFlow, XCSP

Synopsis


C source
witness
Korn
Horn clauses
model/CEX
Z3
SPACER
Eldarica
Golem
typically:
nice invariants
true counterexamples
primary solver in 2024

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]

⊕ 210 problems solved in ~2s each

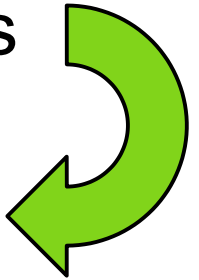
⊕ Avoids 1 unsound verdict (unsigned overflow)

Korn: 2024 updates

- Wrong result: recursive/Primes.c
  Reason: bad handling of effects in shortcut &&
  found and fixed after the competition

Korn: 2024 updates

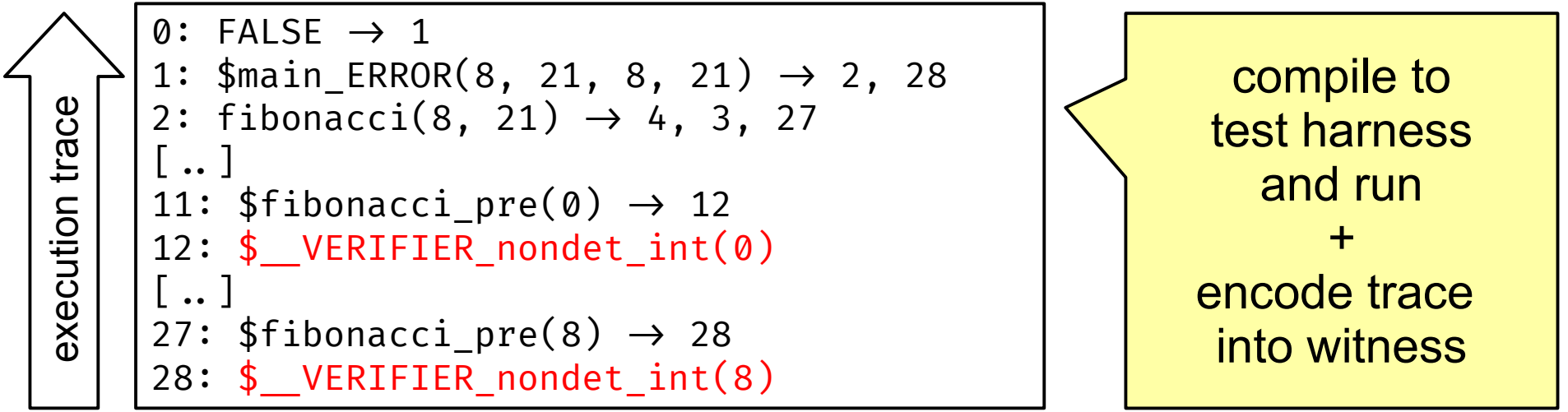
- Wrong result: recursive/Primes.c
  Reason: bad handling of effects in shortcut &&
  found and fixed after the competition
- Two wrong false results: loops-crafted-1
  Reason: unsound encoding of unsigned overflows
- Extend Portfolio: Golem now as first solver:
  similar to Z3 but “easy” access to CEX



Counterexample Validation

don't trust encoding and solvers

- Horn-clauses track `__VERIFIER_nondet_*()`

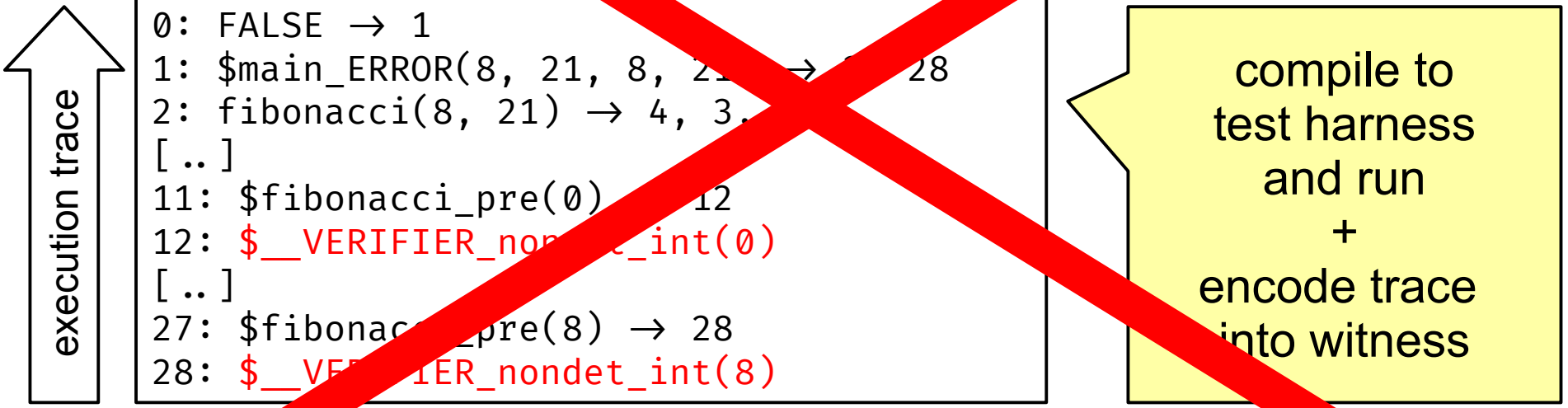


⊕ avoids a handful of incorrect false verdicts

Counterexample Validation

don’t trust encoding and solvers

- Horn-clause track `__VERIFIER_nondet_*()`

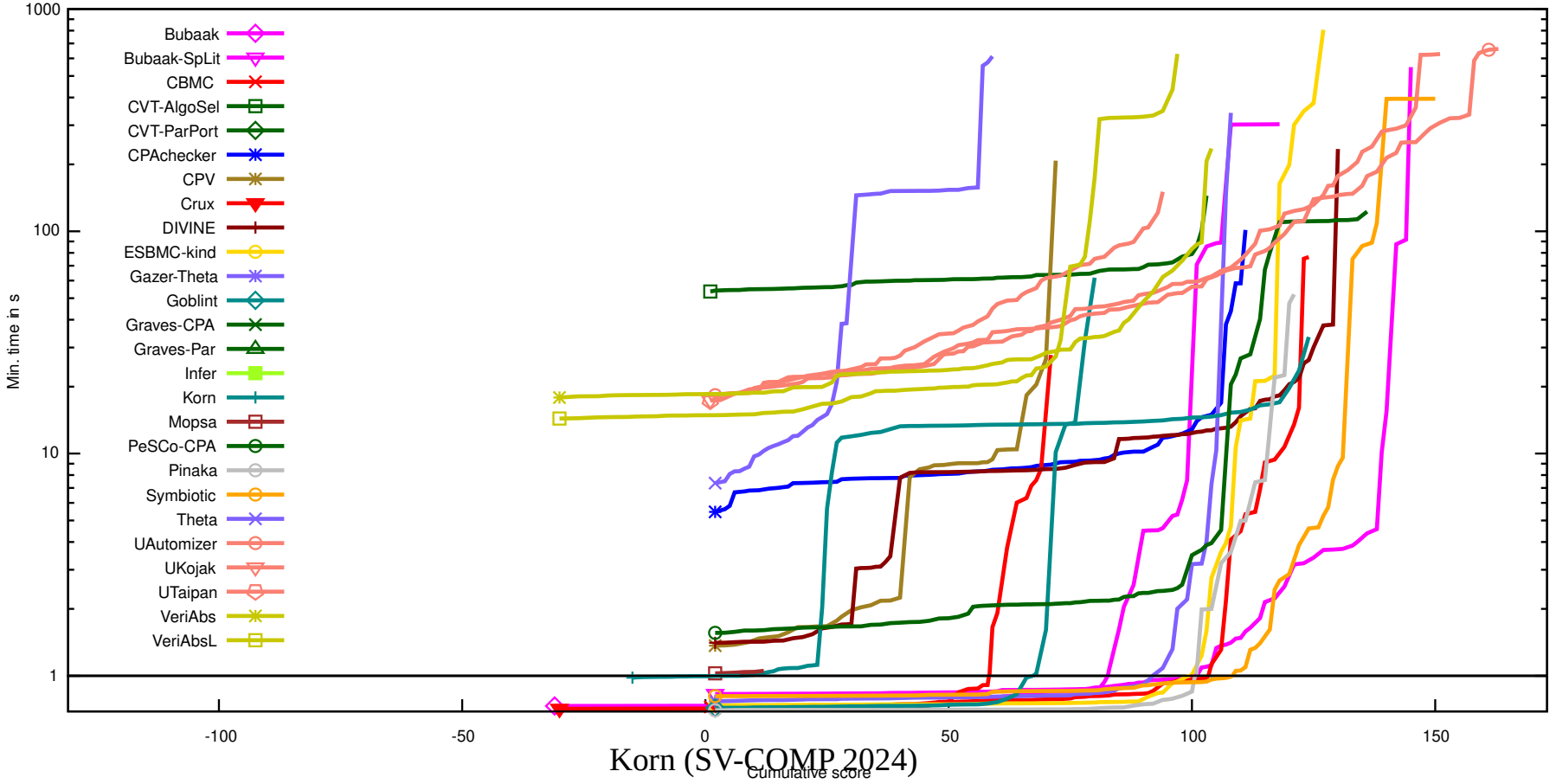


⊕ avoids a handful of incorrect false verdicts

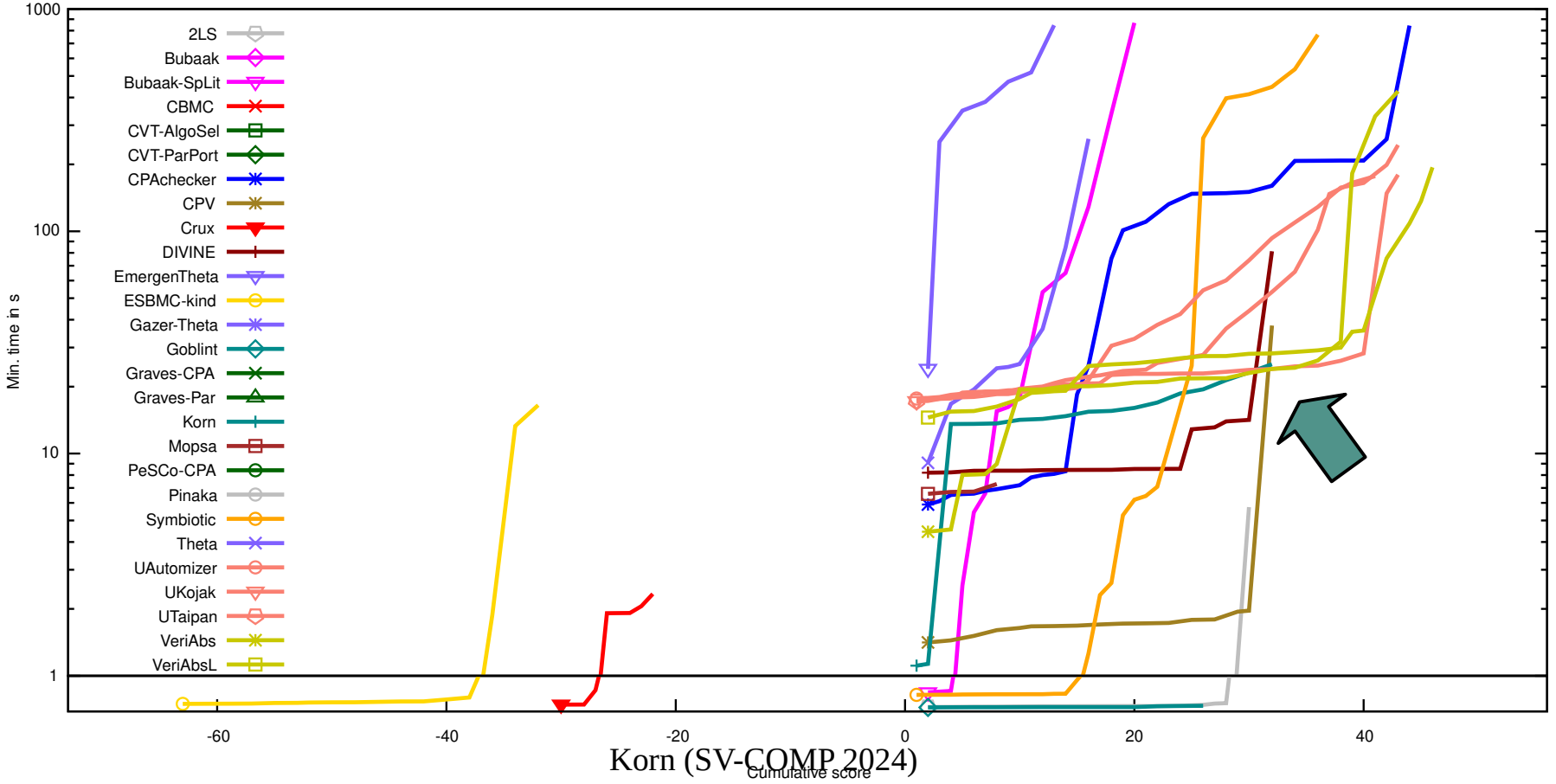
Counterexample Validation

- Insight (for me): Eldarica/Golem CEX are
  - unordered: forgets sequential order of nondet calls
  - shared: equal nondet results are collapsed
- Approach 1: introduce sequential counter
  - ⊖ seems to make verification harder
- Approach 2: use Z3 CEX proofs (more detailed)

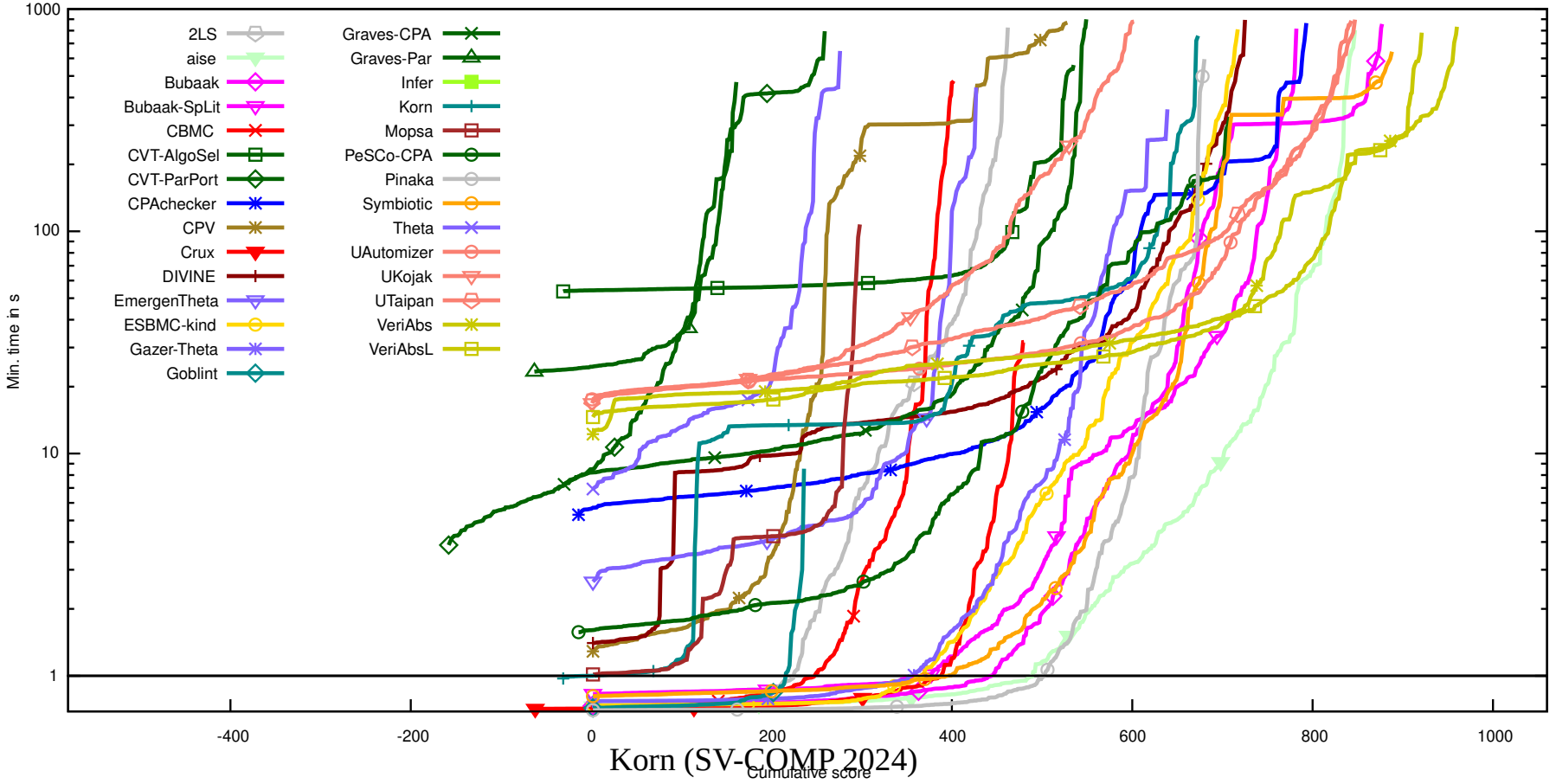
ReachSafety-Recursive



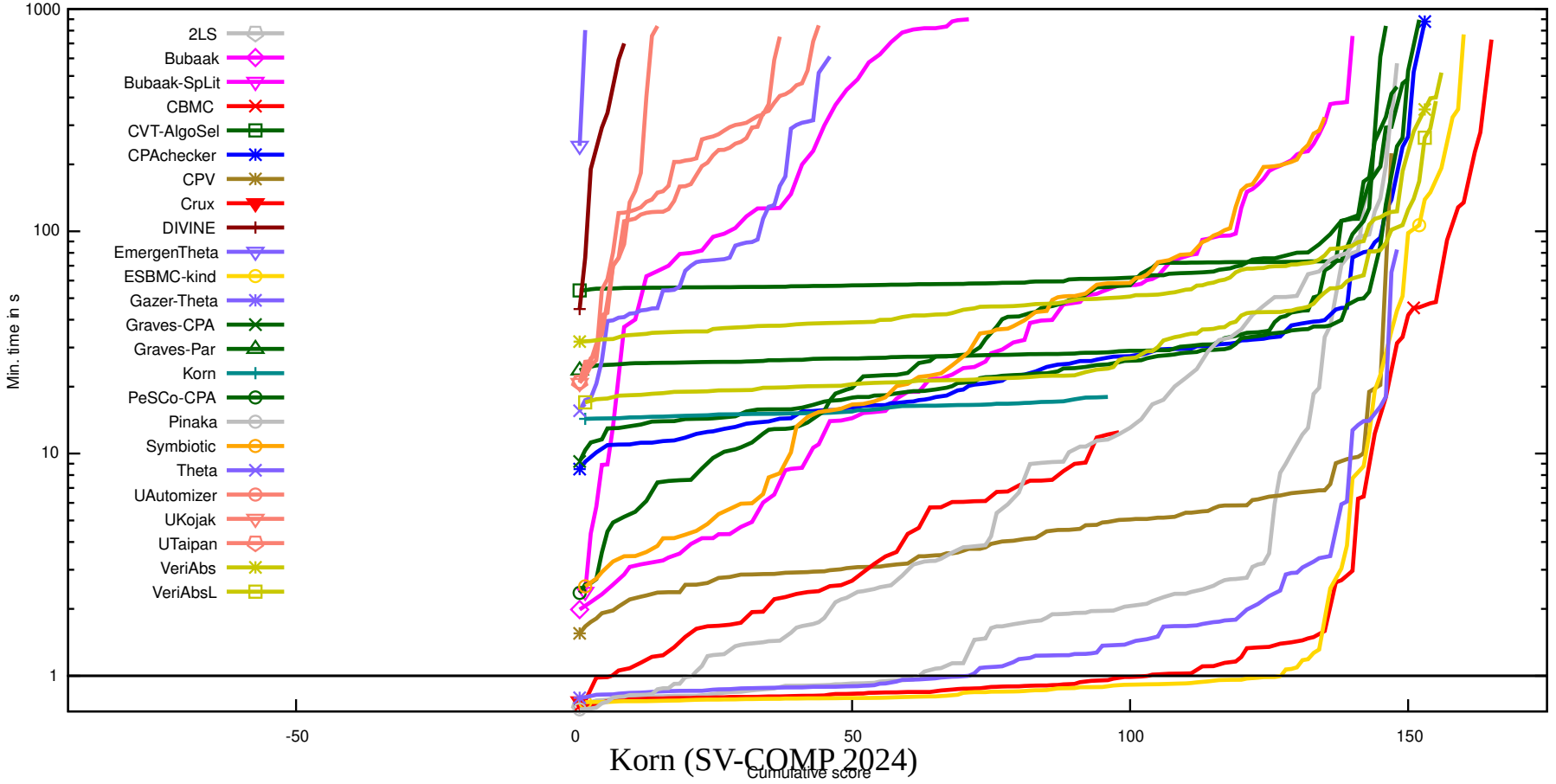
ReachSafety-ControlFlow



ReachSafety-Loops



ReachSafety-XCSP



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