Transition Power Abstraction

Martin Blicha

Università della Svizzera italiana, Lugano, Switzerland

Charles University, Prague, Czech Republic

September 12, 2022
14th Alpine Verification Meeting

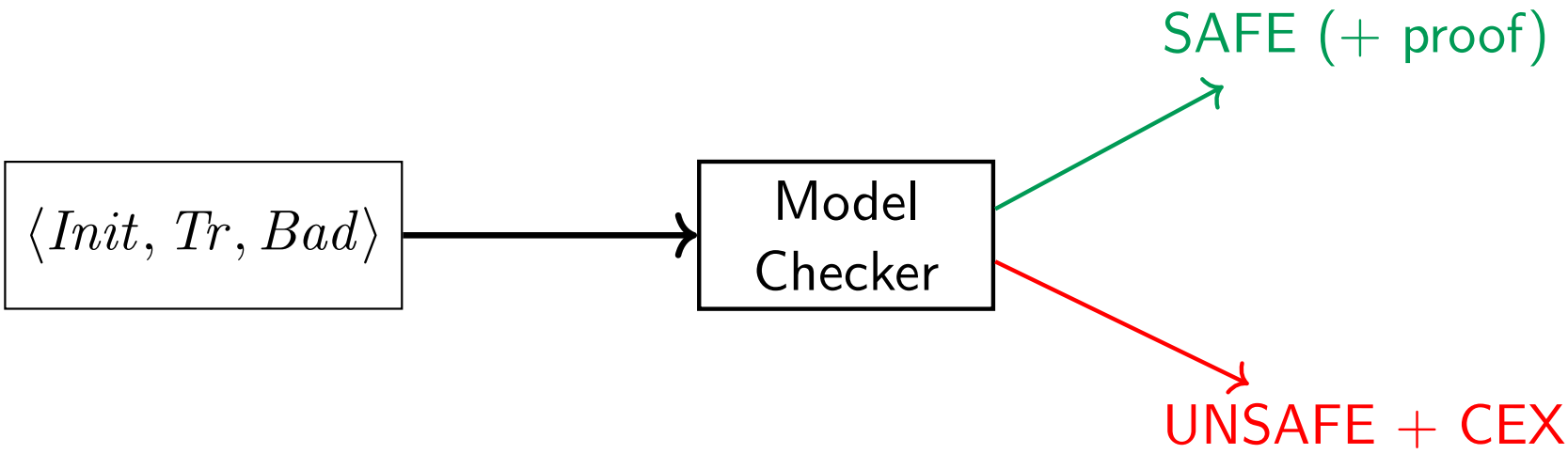
Many thanks to my colleagues!

Antti Hyvärinen

Grigory Fedyukovich

Natasha Sharygina

Context



SMT-based model checking of transition systems (with interpolants)

Motivation

```
x = 0;
y = N;
while(x < 2N)
{
   x = x + 1;
   if(x > N)
      y = y + 1;
}
assert(y != 2N);
```

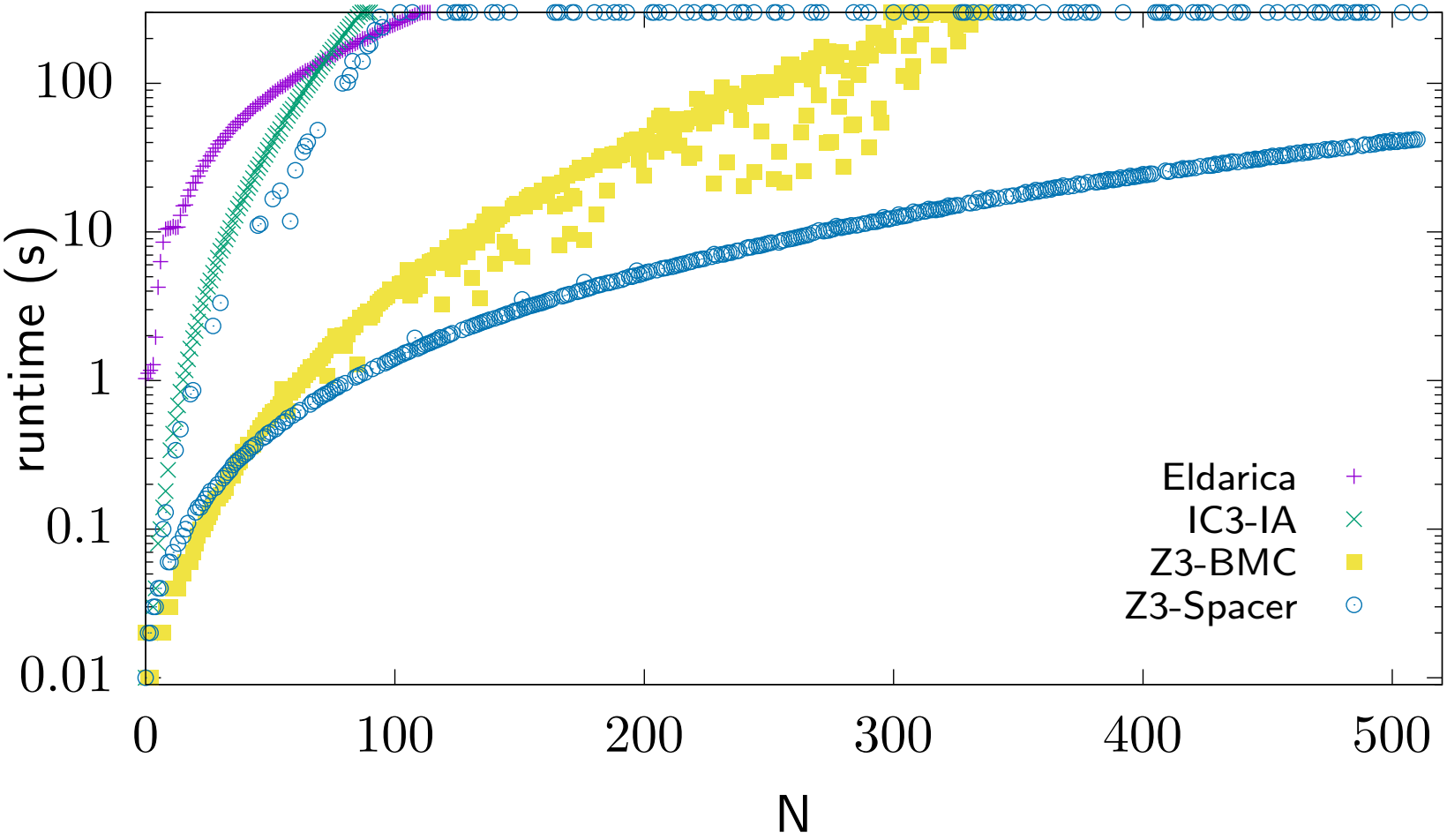
Motivation

```
x = 0;
y = N;
while(x < 2N)
{
   x = x + 1;
   if(x > N)
      y = y + 1;
}
assert(y != 2N);
```

$$Init \equiv x = 0 \wedge y = N$$

$$Tr \equiv x < 2N \wedge x' = x + 1 \wedge y' = ite(x < 2N, y, y + 1)$$

$$Bad \equiv x \geq 2N \wedge y = 2N$$



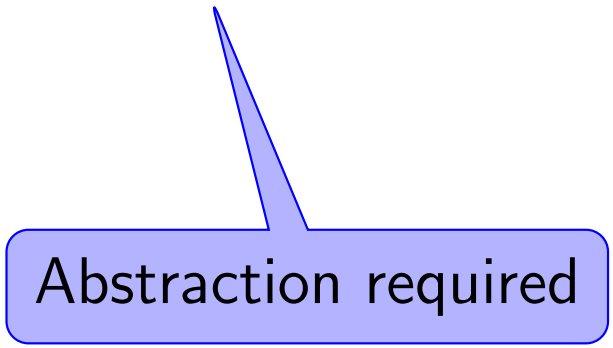
Performance of state-of-the-art tools for increasing counterexample length

Problem: Slow progress in search for longer counterexamples

Solution: Summarize multiple steps

Problem: Slow progress in search for longer counterexamples

Solution: Summarize multiple steps



Transition power abstraction sequence

TPA sequence $TPA^0, TPA^1, \ldots, TPA^n, \ldots$

$TPA^n(x, x')$

- overapproximates reachability up to 2^n steps of Tr
- quantifier-free (only 2 copies of state variables)

Transition power abstraction sequence

TPA sequence $TPA^0, TPA^1, \ldots, TPA^n, \ldots$

$TPA^n(x, x')$

- overapproximates reachability up to 2^n steps of Tr
- quantifier-free (only 2 copies of state variables)

Transition power abstraction sequence

TPA sequence $TPA^0, TPA^1, \ldots, TPA^n, \ldots$

$TPA^n(x, x')$

- overapproximates reachability up to 2^n steps of Tr
- quantifier-free (only 2 copies of state variables)

Transition power abstraction sequence

TPA sequence $TPA^0, TPA^1, \ldots, TPA^n, \ldots$

$TPA^n(x, x')$

- overapproximates reachability up to 2^n steps of Tr
- quantifier-free (only 2 copies of state variables)

- How to construct such sequence?
- How to build a model checking algorithm around it?

Main algorithm

Global: $TPA^0, TPA^1, \ldots$ lazy-initialized to $true$

`CheckSafety`$(Init, Tr, Bad)$

1: $TPA^0 = Id \vee Tr$
2: $n = 0$
3: **while** $true$
4: **if** `IsReachable`$(Init, Bad, n)$
5: **return UNSAFE**
6: $n = n + 1$

Main algorithm

Global: $TPA^0, TPA^1, \ldots$ lazy-initialized to $true$

`CheckSafety`$(Init, Tr, Bad)$

1: $TPA^0 = Id \vee Tr$
2: $n = 0$
3: **while** $true$
4: **if** `IsReachable`$(Init, Bad, n)$
5: **return UNSAFE**
6: $n = n + 1$

up to 2^{n+1} steps of Tr

IsReachable procedure

`IsReachable`($Source, Target, n$)

1: $res = Sat?[Source(x) \wedge TPA^n(x, x') \wedge TPA^n(x', x'') \wedge Target(x'')]$
2: **if** $res == UNSAT$
3: $\wedge\ Target(x'')$
4:
5: **return false**
6: **else** // $res == SAT$
7: **if** $n == 0$
8: **return true**
9: $Intermediate =$ `ExtractIntermediate()`
10: **if not** `IsReachable`($Source, Intermediate, n-1$)
11: goto 1
12: **if not** `IsReachable`($Intermediate, Target, n-1$)
13: goto 1
14: **return true**

Is $Target$ reachable from $Source$ in $\leq 2^{n+1}$ steps

IsReachable procedure

IsReachable($Source, Target, n$)

1: $res = Sat?[Source(x) \wedge TPA^n(x, x') \wedge TPA^n(x', x'') \wedge Target(x'')]$

Abstract path exists?

IsReachable procedure

IsReachable($Source, Target, n$)

1: $res = Sat?[Source(x) \wedge TPA^n(x, x') \wedge TPA^n(x', x'') \wedge Target(x'')]$
2: **if** $res == UNSAT$
3: $I = Itp(TPA^n(x, x') \wedge TPA^n(x', x''), Source(x) \wedge Target(x''))$
4: $TPA^{n+1} = TPA^{n+1} \wedge I$
5: **return false**
6: **else** // $res == SAT$
7: **if** $n == 0$
8: **return true**
9: $Intermediate =$ ExtractIntermediate()
10: **if not** IsReachable($Source, Intermediate, n-1$)
11: goto 1
12: **if not** IsReachable($Intermediate, Target, n-1$)
13: goto 1
14: **return true**

IsReachable procedure

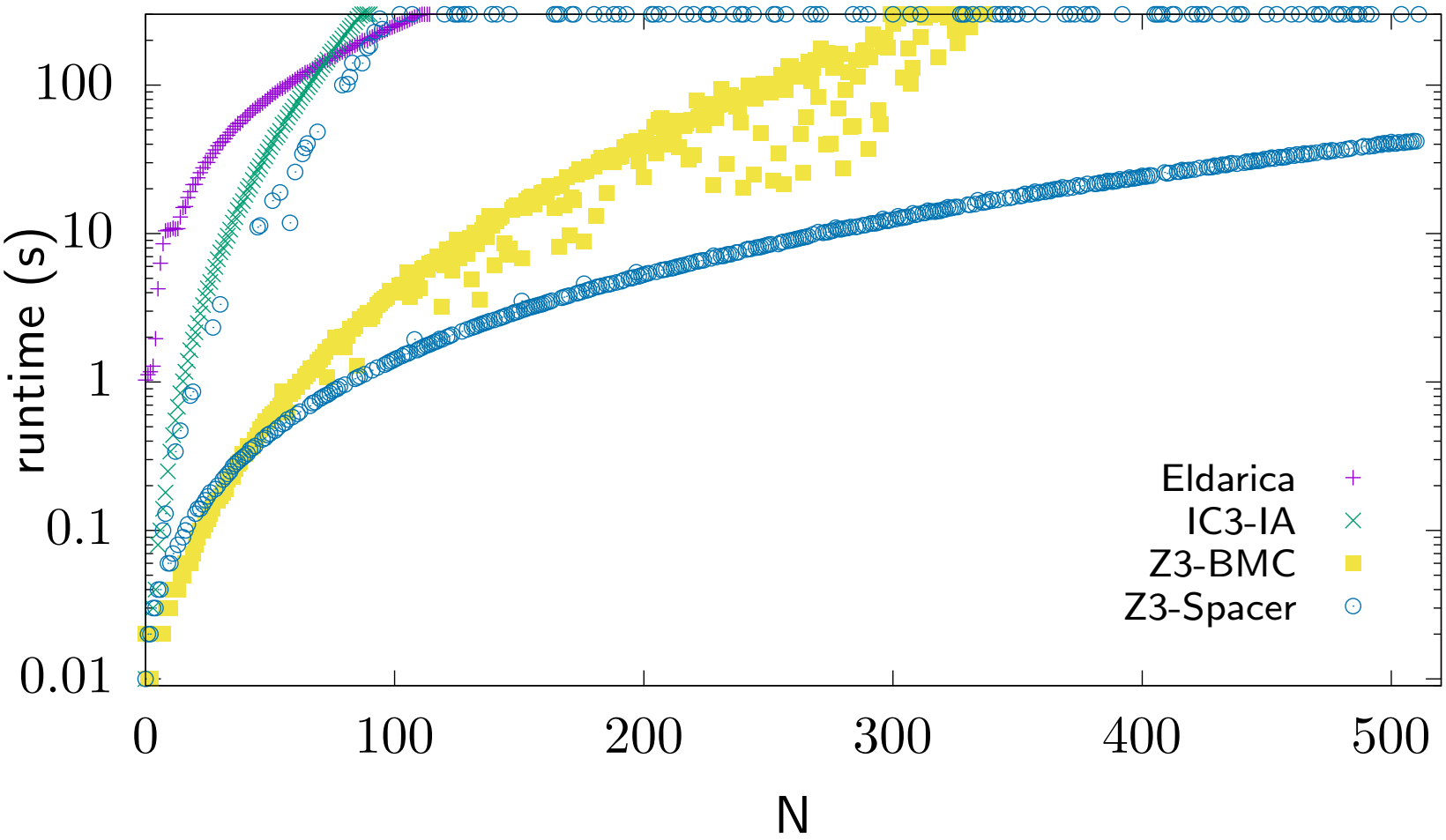
```
IsReachable(Source, Target, n)
1:  res = Sat?[Source(x) ∧ TPA^n(x, x') ∧ TPA^n(x', x'') ∧ Target(x'')]
2:  if res == UNSAT
3:      I = Itp(TPA^n(x, x') ∧ TPA^n(x', x''), Source(x) ∧ Target(x''))
4:      TPA^{n+1} = TPA^{n+1} ∧ I
5:      return false
6:  else // res == SAT
7:      if n == 0
8:          return true
9:      Intermediate = ExtractIntermediate()
10:     if not IsReachable(Source, Intermediate, n−1)
11:         goto 1
12:     if not IsReachable(Intermediate, Target, n−1)
13:         goto 1
14:     return true
```

IsReachable procedure

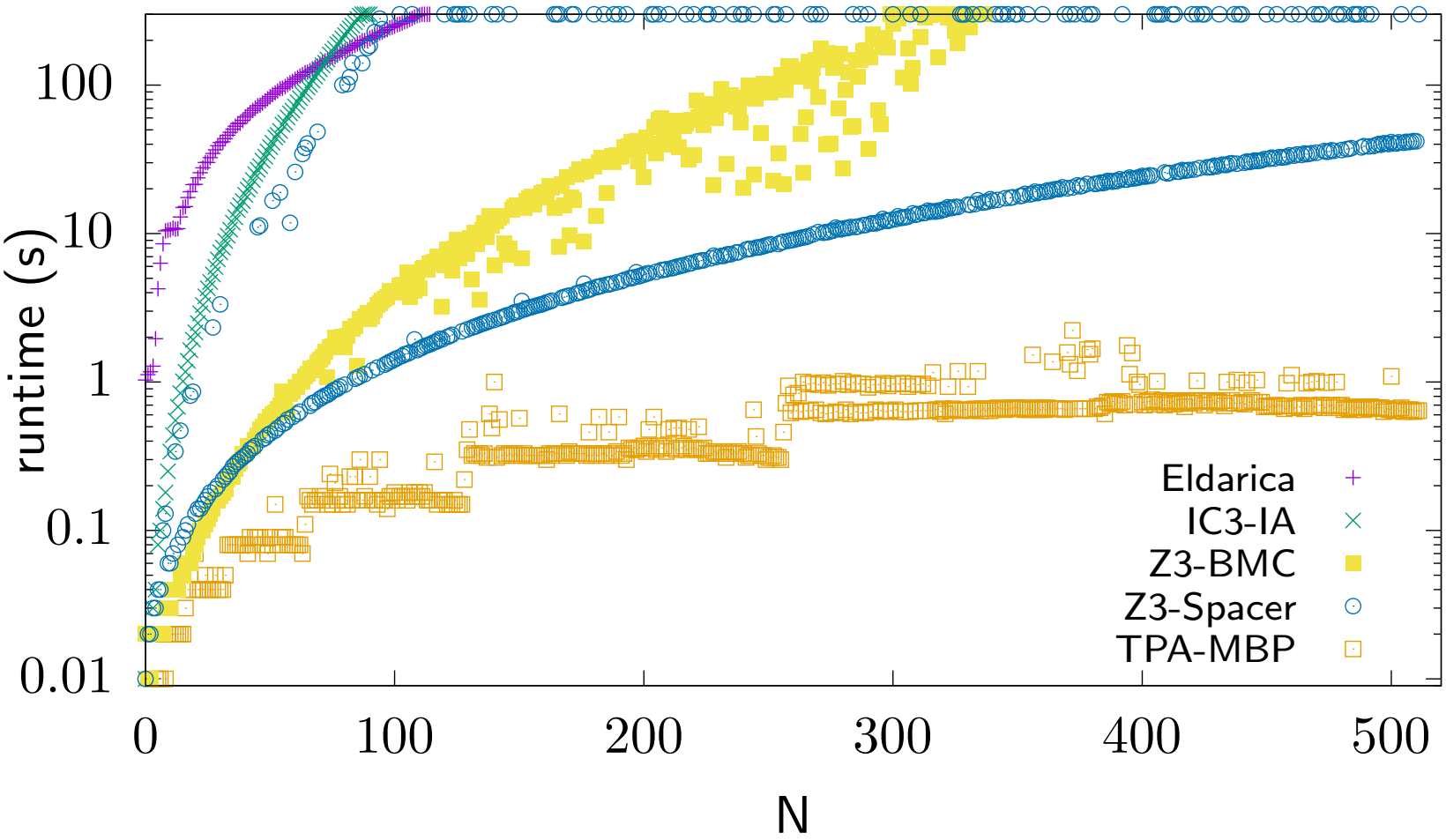
IsReachable($Source, Target, n$)

1: $res = Sat?[Source(x) \wedge TPA^n(x, x') \wedge TPA^n(x', x'') \wedge Target(x'')]$
2: **if** $res == UNSAT$
3: $I = Itp(TPA^n(x, x') \wedge TPA^n(x', x''), Source(x) \wedge Target(x''))$
4: $TPA^{n+1} = TPA^{n+1} \wedge I$
5: **return false**
6: **else** // $res == SAT$
7: **if** $n == 0$
8: **return true**
9: $Intermediate =$ ExtractIntermediate()
10: **if not** IsReachable($Source, Intermediate, n-1$)
11: goto 1
12: **if not** IsReachable($Intermediate, Target, n-1$)
13: goto 1
14: **return true**

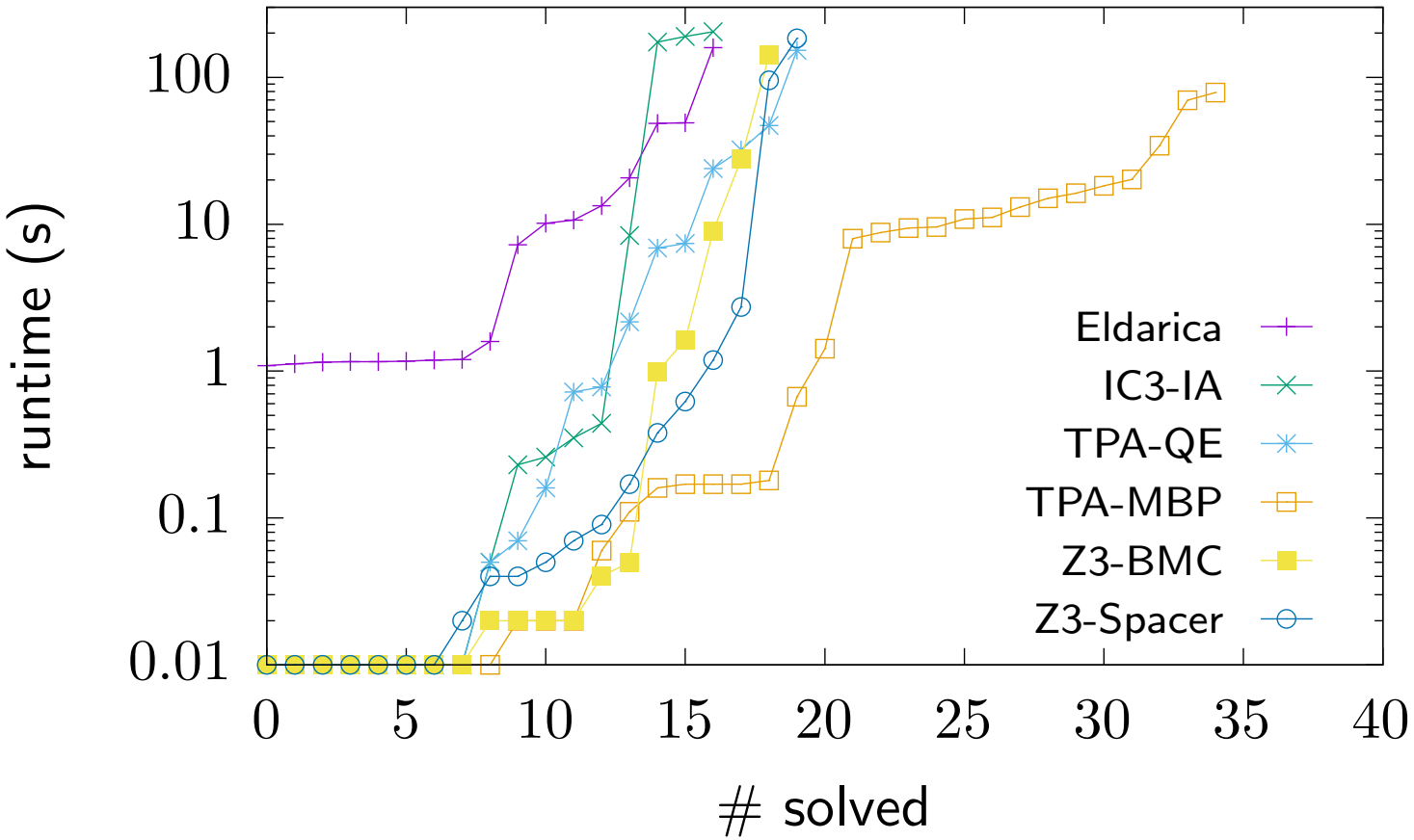
Experiments



Experiments



Experiments



Performance on 54 unsafe multi-phase loops

Proving safety

Proving safety

- Elements of TPA as candidates for safe inductive transition invariants

Proving safety

- Elements of TPA as candidates for safe inductive transition invariants

- Come to FMCAD'22 in Trento!

Summary

- *New* algorithm for safety properties of transition systems
  - Transition power abstraction sequence

Summary

- *New* algorithm for safety properties of transition systems
  - Transition power abstraction sequence
- Deep counterexample detection [TACAS'22]

Summary

- *New* algorithm for safety properties of transition systems
  - Transition power abstraction sequence
- Deep counterexample detection [TACAS'22]
- Transition invariants [TACAS'22 + FMCAD'22]

Summary

- *New* algorithm for safety properties of transition systems
  - Transition power abstraction sequence
- Deep counterexample detection [TACAS'22]
- Transition invariants [TACAS'22 + FMCAD'22]
- Implemented in Horn solver Golem

  `https://github.com/usi-verification-and-security/golem`

Thank you!

Questions?