Split Transition Power Abstraction for Unbounded Safety

Martin Blicha[1,3] Grigory Fedyukovich[2] Antti Hyvärinen[1]
Natasha Sharygina[1]

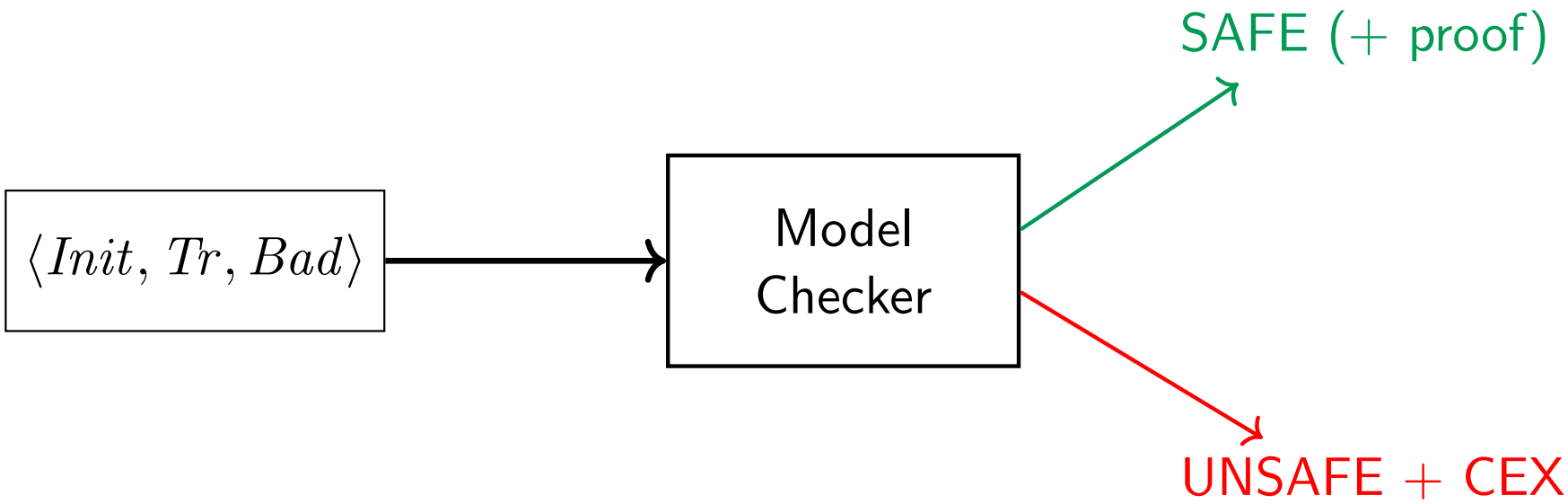
[1]Università della Svizzera italiana, Lugano, Switzerland

[2]Florida State University, Tallahassee, FL, USA

[3]Charles University, Prague, Czech Republic

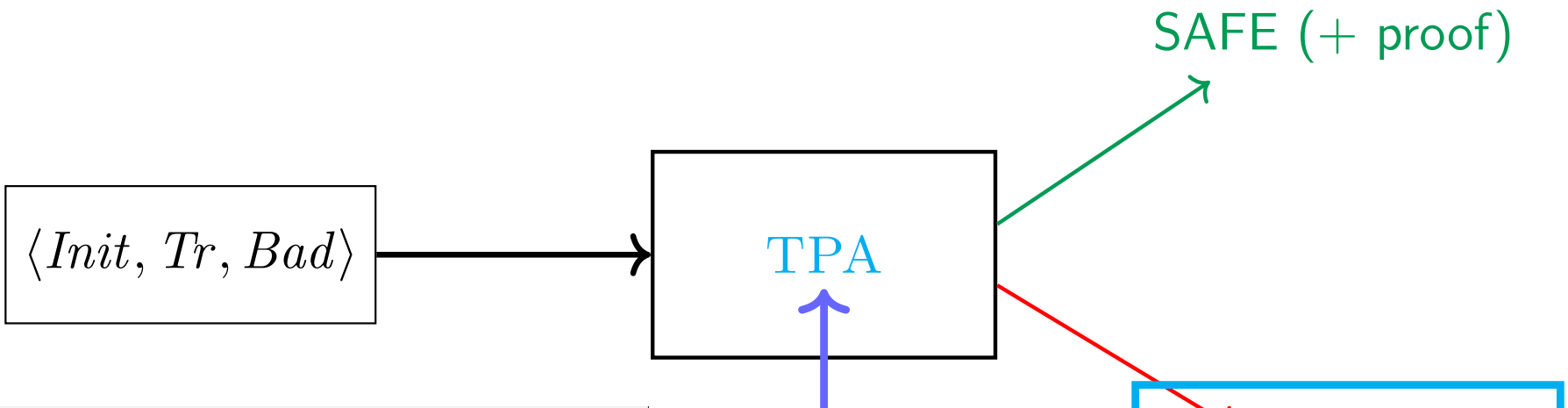
October 21, 2022

Context



SMT-based model checking of transition systems

Context

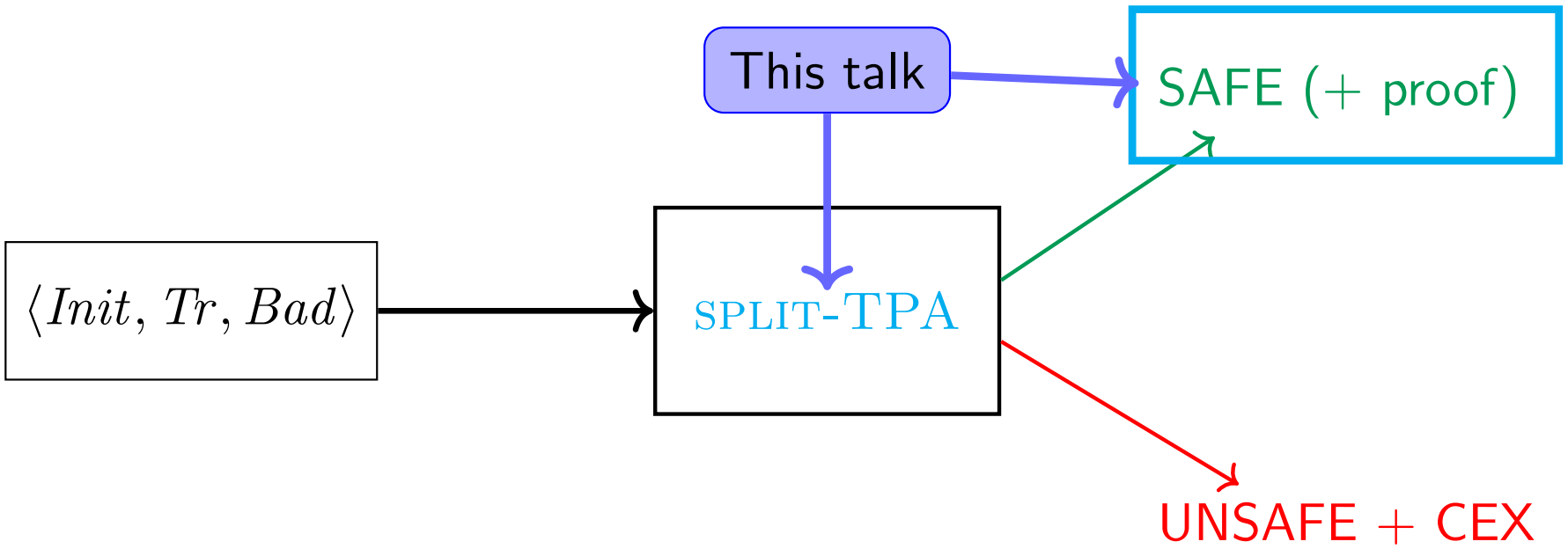


Transition Power Abstractions for Deep Counterexample Detection*

Martin Blicha[1,3](✉), Grigory Fedyukovich[2], Antti E.J. Hyvärinen[1], and Natasha Sharygina[1]

*Transition Power Abstractions for Deep Counterexample Detection, TACAS 2022

Context


This talk
SAFE (+ proof)
$\langle Init, Tr, Bad \rangle$
SPLIT-TPA
UNSAFE + CEX

Outline

Proof of safety - inductive invariant

$$Init(x) \implies Inv(x)$$

$$Inv(x) \wedge Tr(x, x') \implies Inv(x')$$

$$Inv(x) \wedge Bad(x) \implies false$$

safe inductive invariant

Proof of safety - inductive invariant

$$x = 0 \wedge y = 0 \implies Inv(x, y)$$

$$Inv(x, y) \wedge y' = y + 1 \wedge x' = x + y \implies Inv(x', y')$$

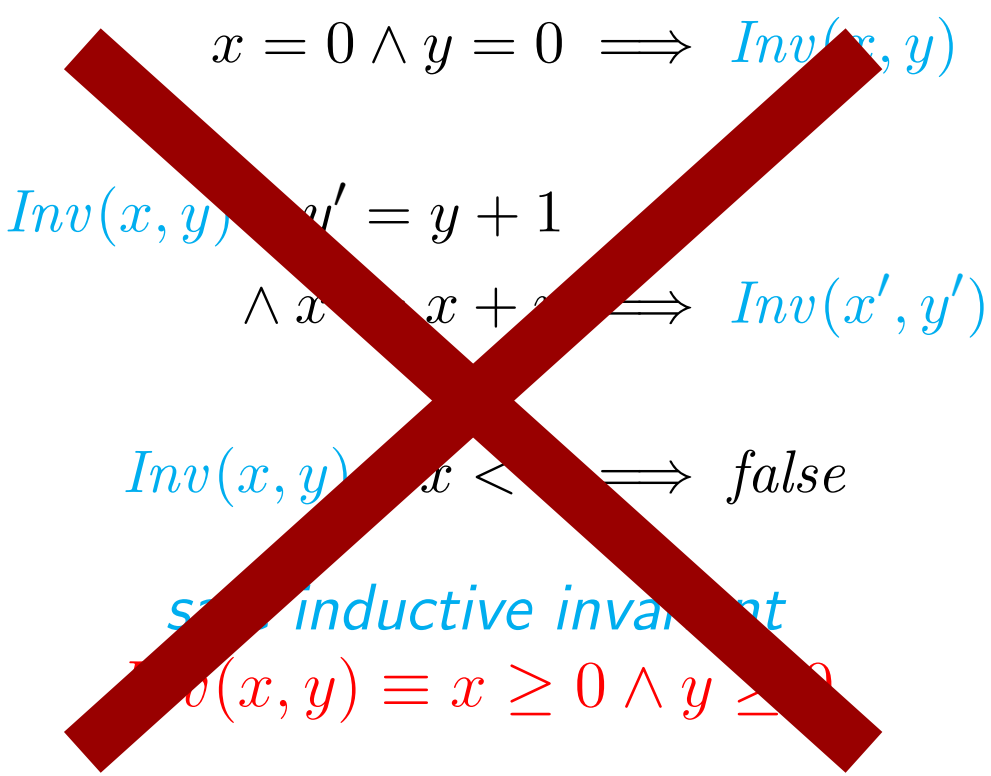
$$Inv(x, y) \wedge x < 0 \implies false$$

safe inductive invariant

$$Inv(x, y) \equiv x \geq 0 \wedge y \geq 0$$

```
x = 0;
y = 0;
while(*)
{
   x = x + y;
   y = y + 1;
}
assert(x >= 0);
```

Proof of safety - inductive invariant




```
x = 0;
y = 0;
while(*)
{
   x = x + y;
   y = y + 1;
}
assert(x >= 0);
```

Proof of safety - inductive transition invariant

$$Id(x, x') \implies TInv(x, x')$$

$$TInv(x, x') \wedge Tr(x', x'') \implies TInv(x, x'')$$

$$Init(x) \wedge TInv(x, x') \wedge Bad(x') \implies false$$

safe inductive transition invariant

Proof of safety - inductive transition invariant

$$x' = x \wedge y' = y \implies TInv(x, y, x', y')$$

$$TInv(x, y, x', y') \wedge y' = y + 1 \wedge x' = x + y \implies TInv(x, y, x'', y'')$$

$$TInv(x, y, x', y') \wedge x = 0 \wedge y = 0 \wedge x' < 0 \implies false$$

```
x = 0;
y = 0;
while(*)
{
   x = x + y;
   y = y + 1;
}
assert(x >= 0);
```

safe inductive transition invariant

$$TInv(x, y, x', y') \equiv y' \geq y \wedge (x' \geq x \vee x' \geq x + y \vee y \neq 0)$$

TPA - Transition Power Abstration

Transition Power Abstraction

Transition Power Abstraction (TPA) sequence
$TPA^{\leq 0}, TPA^{\leq 1}, \ldots, TPA^{\leq n}, \ldots$

$TPA^{\leq n}(x, x')$

Transition Power Abstraction

Transition Power Abstraction (TPA) sequence
$TPA^{\leq 0}, TPA^{\leq 1}, \ldots, TPA^{\leq n}, \ldots$

$TPA^{\leq n}(x, x')$

- overapproximates reachability up to 2^n steps of Tr
  - $Tr^i \subseteq TPA^{\leq n}$ for $0 \leq i \leq 2^n$

Transition Power Abstraction

Transition Power Abstraction (TPA) sequence
$TPA^{\leq 0}, TPA^{\leq 1}, \ldots, TPA^{\leq n}, \ldots$

$TPA^{\leq n}(x, x')$

- overapproximates reachability up to 2^n steps of Tr
  - $Tr^i \subseteq TPA^{\leq n}$ for $0 \leq i \leq 2^n$
- quantifier-free (only 2 copies of state variables)

Transition Power Abstraction

Transition Power Abstraction (TPA) sequence
$TPA^{\leq 0}, TPA^{\leq 1}, \ldots, TPA^{\leq n}, \ldots$

$TPA^{\leq n}(x, x')$

- overapproximates reachability up to 2^n steps of Tr
  - $Tr^i \subseteq TPA^{\leq n}$ for $0 \leq i \leq 2^n$
- quantifier-free (only 2 copies of state variables)

- Construction and refinement of the sequence intertwined with bounded reachability checks

TPA main algorithm

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

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

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

TPA main algorithm

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

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

1: $TPA^{\leq 0} = Id \vee Tr$
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

TPA IsReachable procedure

IsReachable$(Source, Target, n)$

1: $res = Sat?[Source(x) \wedge TPA^{\leq n}(x, x') \wedge TPA^{\leq n}(x', x'') \wedge Target(x'')]$
2: **if** $res == UNSAT$
3: $Source(x) \wedge Target(x'')$
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

TPA IsReachable procedure

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

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

Abstract path exists?

TPA IsReachable procedure

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

1: $res = Sat?[Source(x) \wedge TPA^{\leq n}(x, x') \wedge TPA^{\leq n}(x', x'') \wedge Target(x'')]$
2: **if** $res == UNSAT$
3: $I = Itp(TPA^{\leq n}(x, x') \wedge TPA^{\leq n}(x', x''), Source(x) \wedge Target(x''))$
4: $TPA^{\leq n+1} = TPA^{\leq n+1} \wedge I$
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**

TPA IsReachable procedure

IsReachable($Source, Target, n$)

1: $res = Sat?[Source(x) \wedge TPA^{\leq n}(x, x') \wedge TPA^{\leq n}(x', x'') \wedge Target(x'')]$
2: **if** $res == UNSAT$
3: $I = Itp(TPA^{\leq n}(x, x') \wedge TPA^{\leq n}(x', x''), Source(x) \wedge Target(x''))$
4: $TPA^{\leq n+1} = TPA^{\leq n+1} \wedge I$
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**

TPA IsReachable procedure

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

1: $res = Sat?[Source(x) \wedge TPA^{\leq n}(x, x') \wedge TPA^{\leq n}(x', x'') \wedge Target(x'')]$
2: **if** $res == UNSAT$
3: $I = Itp(TPA^{\leq n}(x, x') \wedge TPA^{\leq n}(x', x''), Source(x) \wedge Target(x''))$
4: $TPA^{\leq n+1} = TPA^{\leq n+1} \wedge I$
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**

TPA - Proving safety

- $TPA^{\leq n}$ proof of bounded safety
  - After `IsReachable`$(Init, Bad, n)$ returns FALSE

TPA - Proving safety

- $TPA^{\leq n}$ proof of bounded safety
  - After `IsReachable`$(Init, Bad, n)$ returns FALSE
- Check for proof of unbounded safety
  - similar to IC3, McMillan's IMC

TPA - Proving safety

- $TPA^{\leq n}$ proof of bounded safety
  - After `IsReachable`$(Init, Bad, n)$ returns FALSE
- Check for proof of unbounded safety
  - similar to IC3, McMillan's IMC

- Safe inductive transition invariant

$$Id(x, x') \implies TPA^{\leq n}(x, x')$$

$$TPA^{\leq n}(x, x') \wedge Tr(x', x'') \implies TPA^{\leq n}(x, x'')$$

$$Init(x) \wedge TPA^{\leq n}(x, x') \wedge Bad(x') \implies false$$

TPA - Proving safety

- $TPA^{\leq n}$ proof of bounded safety
  - After `IsReachable`$(Init, Bad, n)$ returns FALSE
- Check for proof of unbounded safety
  - similar to IC3, McMillan's IMC

- Safe inductive transition invariant

$$Id(x, x') \implies TPA^{\leq n}(x, x') \checkmark$$

$$TPA^{\leq n}(x, x') \wedge Tr(x', x'') \implies TPA^{\leq n}(x, x'')$$

$$Init(x) \wedge TPA^{\leq n}(x, x') \wedge Bad(x') \implies false$$

TPA - Proving safety

- $TPA^{\leq n}$ proof of bounded safety
  - After `IsReachable`$(Init, Bad, n)$ returns FALSE
- Check for proof of unbounded safety
  - similar to IC3, McMillan's IMC

- Safe inductive transition invariant

$$Id(x, x') \implies TPA^{\leq n}(x, x') \checkmark$$

$$TPA^{\leq n}(x, x') \wedge Tr(x', x'') \implies TPA^{\leq n}(x, x'')$$

$$Init(x) \wedge TPA^{\leq n}(x, x') \wedge Bad(x') \implies false \checkmark$$

TPA - Proving safety

- $TPA^{\leq n}$ proof of bounded safety
  - After `IsReachable`$(Init, Bad, n)$ returns FALSE
- Check for proof of unbounded safety
  - similar to IC3, McMillan's IMC

- Safe inductive transition invariant

$$Id(x, x') \implies TPA^{\leq n}(x, x') \checkmark$$

$$TPA^{\leq n}(x, x') \wedge Tr(x', x'') \implies TPA^{\leq n}(x, x'')?$$

$$Init(x) \wedge TPA^{\leq n}(x, x') \wedge Bad(x') \implies false \checkmark$$

TPA - Proving safety

- $TPA^{\leq n}$ proof of bounded safety
  - After `IsReachable`$(Init, Bad, n)$ returns FALSE
- Check for proof of unbounded safety
  - similar to IC3, McMillan's IMC

- Safe inductive transition invariant

$$Id(x, x') \implies TPA^{\leq n}(x, x') \checkmark$$

$$TPA^{\leq n}(x, x') \wedge Tr(x', x'') \implies TPA^{\leq n}(x, x'')?$$

$$Init(x) \wedge TPA^{\leq n}(x, x') \wedge Bad(x') \implies false \checkmark$$

SMT query
Always just 3 copies of state variables

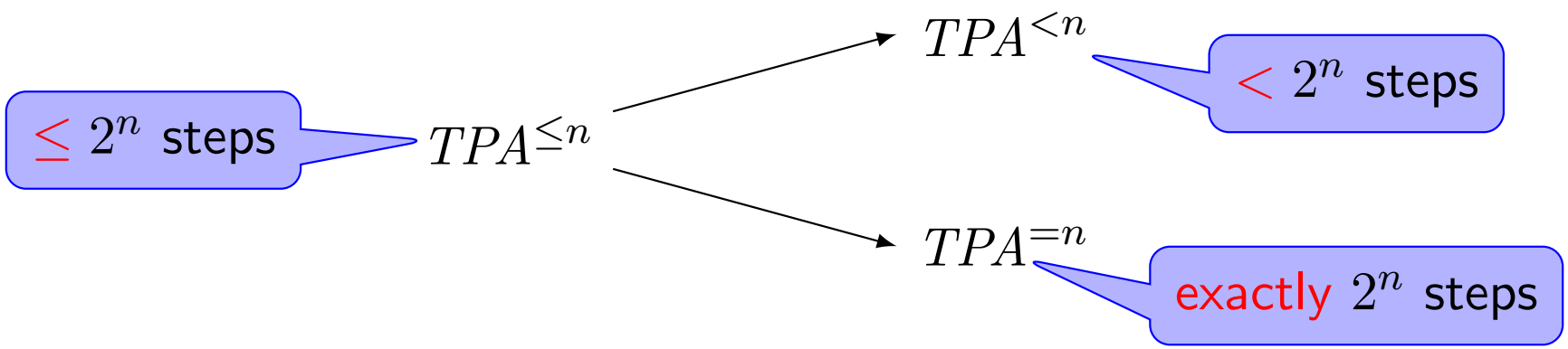
split-TPA

Splitting the TPA sequence

- Problem with TPA: Does not find many inductive transition invariants

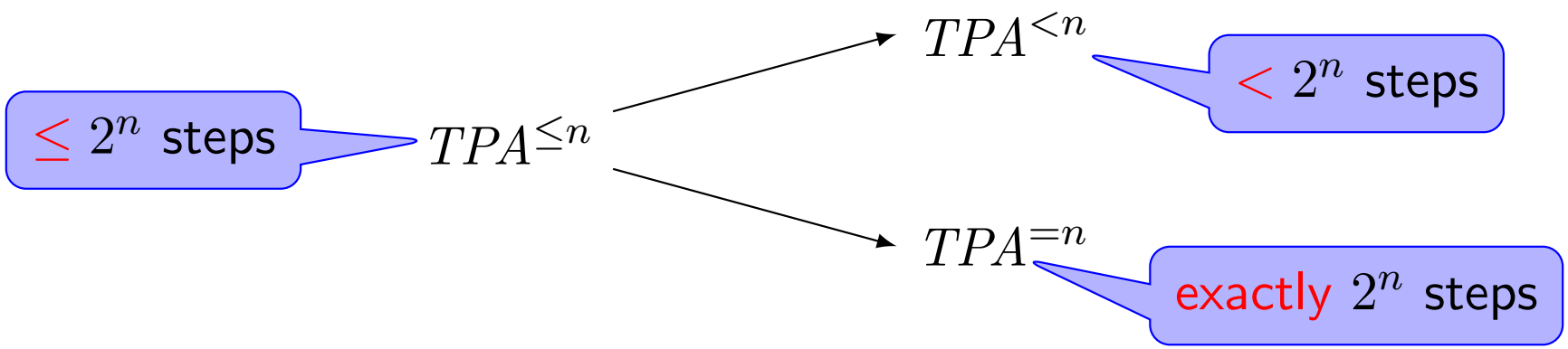
Splitting the TPA sequence

- Problem with TPA: Does not find many inductive transition invariants
- Our solution: Split the TPA sequence!



Splitting the TPA sequence

- Problem with TPA: Does not find many inductive transition invariants
- Our solution: Split the TPA sequence!



- Motivations for splitting
  - Stronger proof rule: k-induction
  - More sources of candidates for transition invariant
  - Symmetries in $TPA^{\leq}$ abstractions

Main algorithm of SPLIT-TPA

Global: $TPA^{<0}, TPA^{<1}, \ldots$ lazy-initialized to $true$
$TPA^{=0}, TPA^{=1}, \ldots$ lazy-initialized to $true$

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

1: $TPA^{<0} = Id$; $TPA^{=0} = Tr$
2: $n = 0$
3: **while** $true$
4: **if** `IsReachableLt`$(Init, Bad, n)$
or `IsReachableEq`$(Init, Bad, n)$
5: **return UNSAFE**
6: **if** `HasInvariant`(n) **return SAFE**
7: $n = n + 1$

Bounded reachability checks in SPLIT-TPA

$TPA^{\leq}$ and `IsReachable`

$$R^{\leq 0} = Id \cup Tr$$

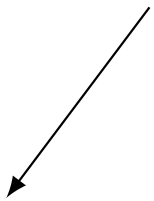
$$R^{\leq n+1} = R^{\leq n} \circ R^{\leq n}$$

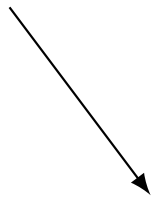
$R^{\leq n}$ – up to 2^n steps of Tr

Bounded reachability checks in SPLIT-TPA

$TPA^{\leq}$ and `IsReachable`

$$R^{\leq 0} = Id \cup Tr$$
$$R^{\leq n+1} = R^{\leq n} \circ R^{\leq n}$$





$TPA^{=}$ and `IsReachableEq`

$$R^{=0} = Tr$$
$$R^{=n+1} = R^{=n} \circ R^{=n}$$

$TPA^{<}$ and `IsReachableLt`

$$R^{<0} = Id$$
$$R^{<n+1} = R^{<n} \cup R^{=n} \circ R^{<n}$$

Discovering safe transition invariants in SPLIT-TPA

$$Id(x, x') \implies TInv(x, x')$$
$$TInv(x, x') \wedge Tr(x', x'') \implies TInv(x, x'')$$
$$Init(x) \wedge TInv(x, x') \wedge Bad(x') \implies false$$

$$Tr^i(x, x') \implies TInv(x, x') \quad \text{for } 0 \leq i < k$$

$$\bigwedge_{i=1}^{k} [TInv(x^{(0)}, x^{(i)}) \wedge Tr(x^{(i)}, x^{(i+1)})] \implies TInv(x^{(0)}, x^{(k+1)})$$

$$Init(x) \wedge TInv(x, x') \wedge Bad(x') \implies false$$

Discovering safe transition invariants in SPLIT-TPA

$$Tr^i(x, x') \implies TPA^{<n}(x, x') \quad \text{for } 0 \leq i < k$$

$$\bigwedge_{i=1}^{k} [TPA^{<n}(x^{(0)}, x^{(i)}) \wedge Tr(x^{(i)}, x^{(i+1)})] \implies TPA^{<n}(x^{(0)}, x^{(k+1)})$$

$$Init(x) \wedge TPA^{<n}(x, x') \wedge Bad(x') \implies false$$

Discovering safe transition invariants in SPLIT-TPA

$$Tr^i(x, x') \implies TPA^{<n}(x, x') \quad \text{for } 0 \leq i < k \checkmark$$

For $k \leq 2^n$

$$\bigwedge_{i=1}^{k} [TPA^{<n}(x^{(0)}, x^{(i)}) \wedge Tr(x^{(i)}, x^{(i+1)})] \implies TPA^{<n}(x^{(0)}, x^{(k+1)})$$

$$Init(x) \wedge TPA^{<n}(x, x') \wedge Bad(x') \implies false \checkmark$$

After `IsReachableLt`$(Init, Bad, n)$ returns FALSE

Discovering safe transition invariants in SPLIT-TPA

$$Tr^i(x, x') \implies TPA^{<n}(x, x') \quad \text{for } 0 \leq i < k \checkmark$$

$$\bigwedge_{i=1}^{k} [TPA^{<n}(x^{(0)}, x^{(i)}) \wedge Tr(x^{(i)}, x^{(i+1)})] \implies TPA^{<n}(x^{(0)}, x^{(k+1)}) \ ?$$

$$Init(x) \wedge TPA^{<n}(x, x') \wedge Bad(x') \implies false \checkmark$$

Check for $k = 2^n$?

Discovering safe transition invariants in SPLIT-TPA

$$\bigwedge_{i=1}^{k} [TPA^{<n}(x^{(0)}, x^{(i)}) \wedge Tr(x^{(i)}, x^{(i+1)})] \implies TPA^{<n}(x^{(0)}, x^{(k+1)})$$

if

$$TPA^{<n} \circ TPA^{=m} \subseteq TPA^{<n} \quad \text{for some } 0 \leq m \leq n$$

then

$TPA^{<n}$ is 2^m-inductive transition invariant

Discovering safe transition invariants in SPLIT-TPA

- Use $TPA^{<n} \cup TPA^{<n} \circ TPA^{=n}$ as the candidate relation instead of $TPA^{<n}$

Discovering safe transition invariants in SPLIT-TPA

- Use $TPA^{<n} \cup TPA^{<n} \circ TPA^{=n}$ as the candidate relation instead of $TPA^{<n}$
- First and third conditions satisfied at some point

Discovering safe transition invariants in SPLIT-TPA

- Use $TPA^{<n} \cup TPA^{<n} \circ TPA^{=n}$ as the candidate relation instead of $TPA^{<n}$
- First and third conditions satisfied at some point
- If $TPA^{=n} \circ TPA^{=n} \subseteq TPA^{=n}$

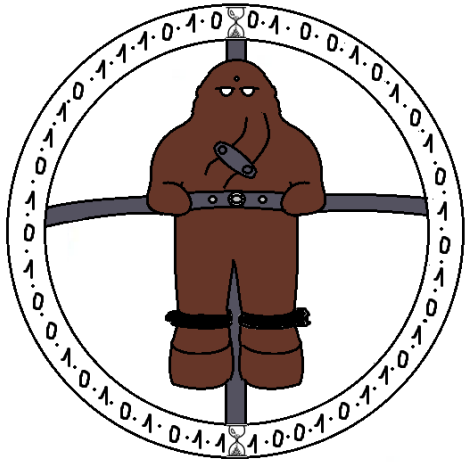
Discovering safe transition invariants in SPLIT-TPA

- Use $TPA^{<n} \cup TPA^{<n} \circ TPA^{=n}$ as the candidate relation instead of $TPA^{<n}$
- First and third conditions satisfied at some point
- If $TPA^{=n} \circ TPA^{=n} \subseteq TPA^{=n}$
- Then $TPA^{<n} \cup TPA^{<n} \circ TPA^{=n}$ is 2^n-inductive transition invariant

Experiments

Experiments

- SPLIT-TPA implemented in Horn solver GOLEM
- 54 benchmarks representing challenging multi-phase loops (from CHC-COMP)
- safe and unsafe version
- timeout 5 minutes
- Artifact: `https://doi.org/10.5281/zenodo.6988735`



Experiments

Benchmark suite	SPLIT-TPA	TPA	Z3SPACER	GSPACER	ELDARICA
multi-phase unsafe	37 (3)	35 (2)	20 (0)	17 (0)	17 (0)
multi-phase safe	19 (7)	12 (0)	6 (0)	24 (3)	26 (4)

Solved (unique) instances.

Future work

- Variants
  - Tradeoff in checking candidate invariants
  - On-the-fly checks for invariants
  - Forward vs backward search during refinement
  - $R^{<n+1} = R^{<n} \cup R^{<n} \circ R^{=n}$ vs $R^{<n+1} = R^{<n} \cup R^{=n} \circ R^{<n}$

Future work

- Variants
  - Tradeoff in checking candidate invariants
  - On-the-fly checks for invariants
  - Forward vs backward search during refinement
  - $R^{<n+1} = R^{<n} \cup R^{<n} \circ R^{=n}$ vs $R^{<n+1} = R^{<n} \cup R^{=n} \circ R^{<n}$
- Extension to general CHCs

Future work

- Variants
  - Tradeoff in checking candidate invariants
  - On-the-fly checks for invariants
  - Forward vs backward search during refinement
  - $R^{<n+1} = R^{<n} \cup R^{<n} \circ R^{=n}$ vs $R^{<n+1} = R^{<n} \cup R^{=n} \circ R^{<n}$
- Extension to general CHCs
  - Modular verification

Conclusion

- SPLIT-TPA: model checking of safety properties

Conclusion

- SPLIT-TPA: model checking of safety properties
- Transition power abstraction sequences
  - Detection of deep counterexamples
  - Safe transition invariants

Conclusion

- SPLIT-TPA: model checking of safety properties
- Transition power abstraction sequences
  - Detection of deep counterexamples
  - Safe transition invariants
- Imlemented in Horn solver GOLEM

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

Thank you!

Questions?

References I

Dejan Jovanović and Bruno Dutertre.

Property-directed k-induction.

In *2016 Formal Methods in Computer-Aided Design (FMCAD)*, pages 85–92, Oct 2016.

Mary Sheeran, Satnam Singh, and Gunnar Stålmarck.

Checking safety properties using induction and a SAT-solver.

In Warren A. Hunt and Steven D. Johnson, editors, *Formal Methods in Computer-Aided Design*, pages 127–144, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

Backup slides

Strengthening proof rule

- Problem with TPA: Does not find many inductive transition invariants

Strengthening proof rule

- Problem with TPA: Does not find many inductive transition invariants
- Idea: Replace induction with k-induction [SSS00, JD16]

Strengthening proof rule

- Problem with TPA: Does not find many inductive transition invariants
- Idea: Replace induction with k-induction [SSS00, JD16]

$$
\begin{aligned}
Id(x, x') &\implies TInv(x, x') \\
TInv(x, x') \wedge Tr(x', x'') &\implies TInv(x, x'') \\
Init(x) \wedge TInv(x, x') \wedge Bad(x') &\implies false
\end{aligned}
$$

$$
Tr^i(x, x') \implies TInv(x, x') \quad \text{for } 0 \leq i < k
$$

$$
\bigwedge_{i=1}^{k} [TInv(x^{(0)}, x^{(i)}) \wedge Tr(x^{(i)}, x^{(i+1)})] \implies TInv(x^{(0)}, x^{(k+1)})
$$

$$
Init(x) \wedge TInv(x, x') \wedge Bad(x') \implies false
$$

k-induction in TPA

$$Tr^i(x, x') \implies TPA^{\leq n}(x, x') \quad \text{for } 0 \leq i < k$$

$$\bigwedge_{i=1}^{k} [TPA^{\leq n}(x^{(0)}, x^{(i)}) \wedge Tr(x^{(i)}, x^{(i+1)})] \implies TPA^{\leq n}(x^{(0)}, x^{(k+1)})$$

$$Init(x) \wedge TPA^{\leq n}(x, x') \wedge Bad(x') \implies false$$

k-induction in TPA

$$Tr^i(x, x') \implies TPA^{\leq n}(x, x') \quad \text{for } 0 \leq i < k \checkmark$$

For $k \leq 2^n$

$$\bigwedge_{i=1}^{k} [TPA^{\leq n}(x^{(0)}, x^{(i)}) \wedge Tr(x^{(i)}, x^{(i+1)})] \implies TPA^{\leq n}(x^{(0)}, x^{(k+1)})$$

$$Init(x) \wedge TPA^{\leq n}(x, x') \wedge Bad(x') \implies false \checkmark$$

After `IsReachable`$(Init, Bad, n)$ returns FALSE

k-induction in TPA

$$Tr^i(x, x') \implies TPA^{\leq n}(x, x') \quad \text{for } 0 \leq i < k \checkmark$$

$$\bigwedge_{i=1}^{k} [TPA^{\leq n}(x^{(0)}, x^{(i)}) \wedge Tr(x^{(i)}, x^{(i+1)})] \implies TPA^{\leq n}(x^{(0)}, x^{(k+1)}) \ ?$$

$$Init(x) \wedge TPA^{\leq n}(x, x') \wedge Bad(x') \implies false \checkmark$$

Check for $k = 2^n$?

Splitting the TPA sequence

$$\bigwedge_{i=1}^{2^n} [TPA^{\leq n}(x^{(0)}, x^{(i)}) \wedge Tr(x^{(i)}, x^{(i+1)})] \implies TPA^{\leq n}(x^{(0)}, x^{(2^n+1)})$$

- Stronger condition easier to check
- $TPA^{\leq n} \circ Tr^{2^n} \subseteq TPA^{\leq n}$

Splitting the TPA sequence

$$\bigwedge_{i=1}^{2^n}[TPA^{\leq n}(x^{(0)}, x^{(i)}) \wedge Tr(x^{(i)}, x^{(i+1)})] \implies TPA^{\leq n}(x^{(0)}, x^{(2^n+1)})$$

- Stronger condition easier to check
- $TPA^{\leq n} \circ Tr^{2^n} \subseteq TPA^{\leq n}$
- Over-approximation of Tr^{2^n} required
- $TPA^{\leq n}$ is too coarse; we need $TPA^{=n}$