

Interpolation-based Model Checking for Proving Program Termination

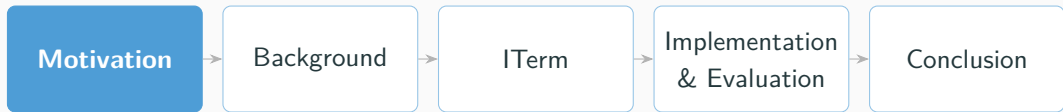
Marek Jankola



October 16. 2026
CPAchecker Workshop 2026



Outline



non-termination
vs. termination

Motivation: Non-Termination

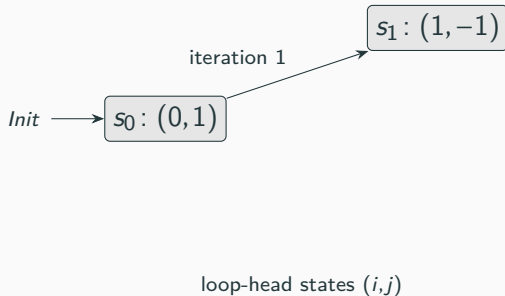
```
1 int main(void) {  
2     int i, j;  
3     j = 1;  
4     i = 0;  
5     while (i-j >= -5) {  
6         j = -i - j;  
7         i = -i - j;  
8     }  
9     return 0;  
10 }
```

Init $\longrightarrow$ $s_0: (0,1)$

loop-head states (i,j)

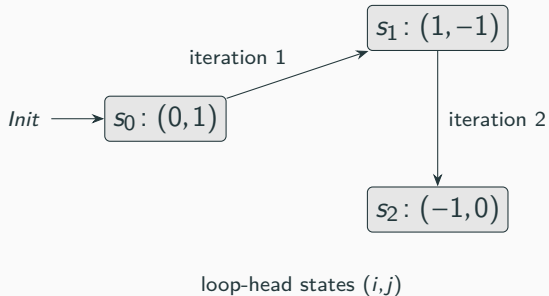
Motivation: Non-Termination

```
1 int main(void) {  
2     int i, j;  
3     j = 1;  
4     i = 0;  
5     while (i-j >= -5) {  
6         j = -i - j;  
7         i = -i - j;  
8     }  
9     return 0;  
10 }
```



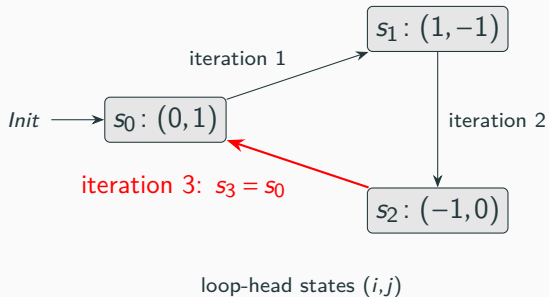
Motivation: Non-Termination

```
1 int main(void) {  
2     int i, j;  
3     j = 1;  
4     i = 0;  
5     while (i-j >= -5) {  
6         j = -i - j;  
7         i = -i - j;  
8     }  
9     return 0;  
10 }
```



Motivation: Non-Termination

```
1 int main(void) {  
2     int i, j;  
3     j = 1;  
4     i = 0;  
5     while (i-j >= -5) {  
6         j = -i - j;  
7         i = -i - j;  
8     }  
9     return 0;  
10 }
```

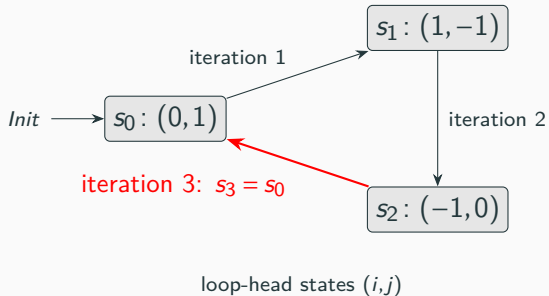


loop-head state repeats $\Rightarrow$ **lasso** $\Rightarrow$ non-termination

- BMC unrolls only **3** iterations

Motivation: Non-Termination

```
1 int main(void) {  
2     int i, j;  
3     j = 1;  
4     i = 0;  
5     while (i-j >= -5) {  
6         j = -i - j;  
7         i = -i - j;  
8     }  
9     return 0;  
10 }
```

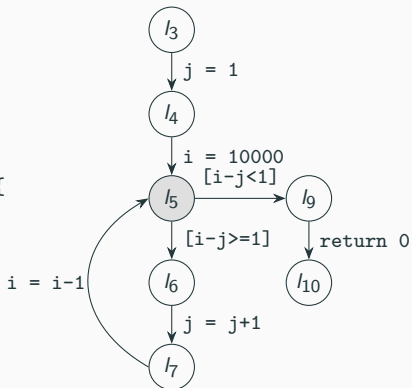


loop-head state repeats $\Rightarrow$ **lasso** $\Rightarrow$ non-termination

- BMC unrolls only 3 iterations
- CPA-BMC: 852 of 1097 non-terminating tasks, more than UAutomizer (835)

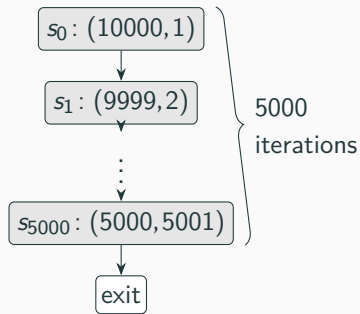
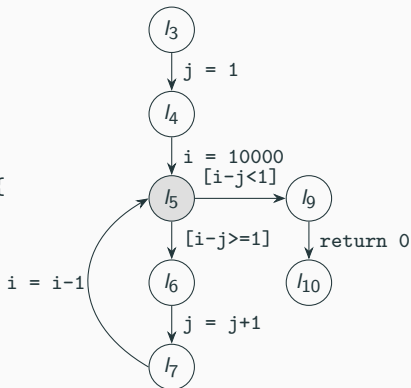
Motivation: Termination

```
1 int main(void) {  
2     int i, j;  
3     j = 1;  
4     i = 10000;  
5     while (i-j >= 1) {  
6         j = j + 1;  
7         i = i - 1;  
8     }  
9     return 0;  
10 }
```



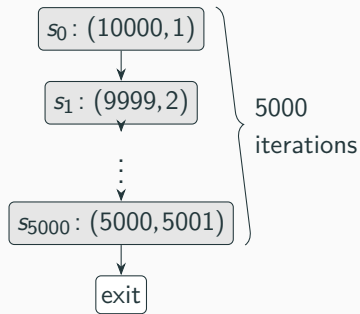
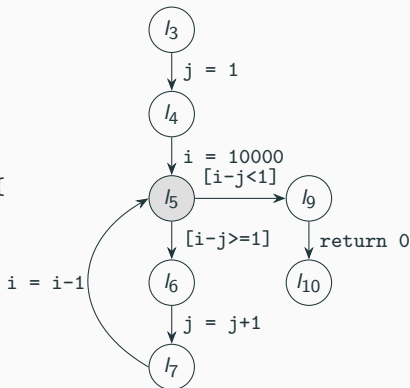
Motivation: Termination

```
1 int main(void) {  
2     int i, j;  
3     j = 1;  
4     i = 10000;  
5     while (i-j >= 1) {  
6         j = j + 1;  
7         i = i - 1;  
8     }  
9     return 0;  
10 }
```



Motivation: Termination

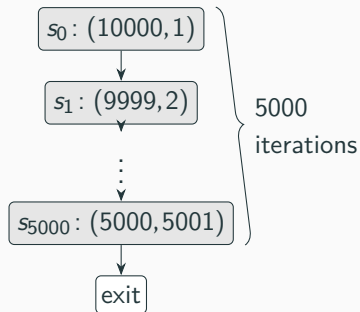
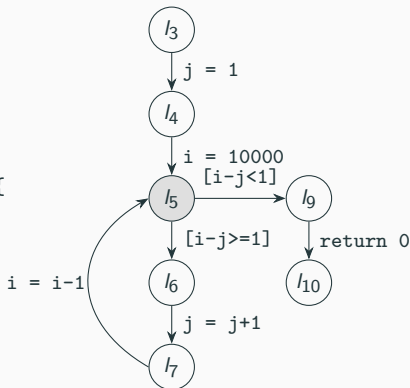
```
1 int main(void) {  
2     int i, j;  
3     j = 1;  
4     i = 10000;  
5     while (i-j >= 1) {  
6         j = j + 1;  
7         i = i - 1;  
8     }  
9     return 0;  
10 }
```



- **X** no state repeats, every BMC query unsat $\Rightarrow$ full unrolling

Motivation: Termination

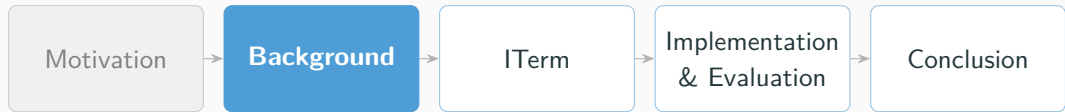
```
1 int main(void) {  
2     int i, j;  
3     j = 1;  
4     i = 10000;  
5     while (i-j >= 1) {  
6         j = j + 1;  
7         i = i - 1;  
8     }  
9     return 0;  
10 }
```



- **X** no state repeats, every BMC query unsat $\Rightarrow$ full unrolling

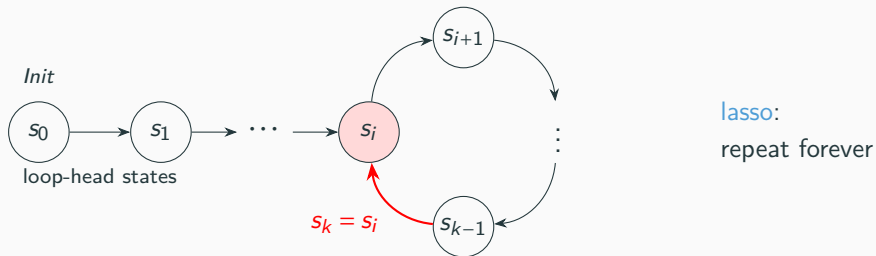
Goal: prove termination without full unrolling

Outline



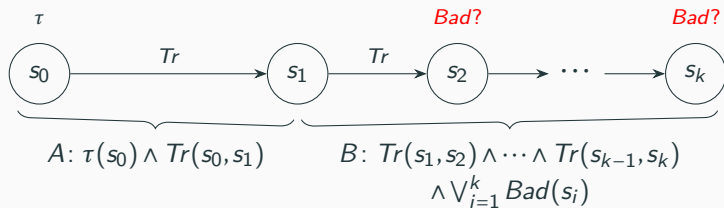
termination as
reachability
IMC

Termination as Reachability [3]

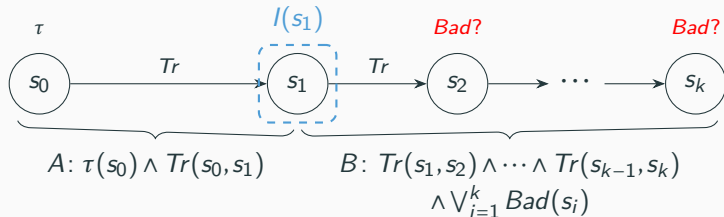


BMC query at bound k : $\underbrace{Init(s_0) \wedge \bigwedge_{i=1}^k Tr(s_{i-1}, s_i)}_{\text{unroll } k \text{ iterations}} \wedge \underbrace{\bigvee_{i < k} s_i = s_k}_{\text{state repeats}}$

Interpolation-based Model Checking (McMillan 2003 [4])

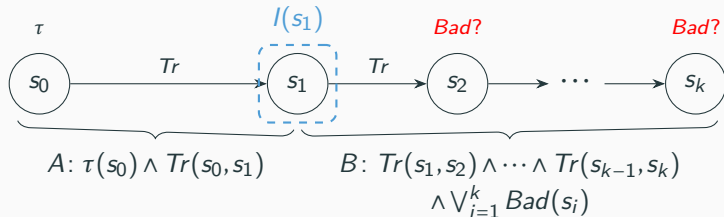


Interpolation-based Model Checking (McMillan 2003 [4])



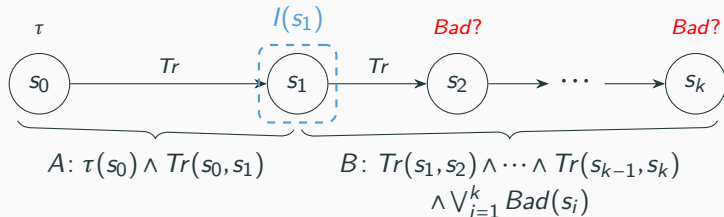
- $\text{unsat} \Rightarrow$ interpolant $I = \text{Itp}(A, B)$:
 - I contains the states reached from τ in one step
 - no bad state reachable from I in $k - 1$ steps

Interpolation-based Model Checking (McMillan 2003 [4])



- $\text{unsat} \Rightarrow$ interpolant $I = \text{Itp}(A, B)$:
 - I contains the states reached from τ in one step
 - no bad state reachable from I in $k - 1$ steps
- $\tau := \tau \vee I$ until $I \Rightarrow \tau$: **inductive invariant** $\Rightarrow$ safe

Interpolation-based Model Checking (McMillan 2003 [4])



- $unsat \Rightarrow$ interpolant $I = ltp(A, B)$:
 - I contains the states reached from τ in one step
 - no bad state reachable from I in $k - 1$ steps
- $\tau := \tau \vee I$ until $I \Rightarrow \tau$: inductive invariant $\Rightarrow$ safe
- sat for $\tau \neq Init \Rightarrow$ increase k (stronger interpolants)

Outline



transition invariants
termination CPA
value analysis

Idea: IMC for Termination

	IMC (reachability)	ITerm (termination)
query checks	$\forall_i \text{Bad}(s_i)$	
interpolant	$I(s)$: set of states	
fixed point	inductive invariant	
proves	no bad state reachable	

Idea: IMC for Termination

	IMC (reachability)	I Term (termination)
query checks	$\forall_i \text{Bad}(s_i)$	$s_{k-1} = s_k$ (state repeats)
interpolant	$I(s)$: set of states	
fixed point	inductive invariant	
proves	no bad state reachable	

Idea: IMC for Termination

	IMC (reachability)	ITerm (termination)
query checks	$\forall_i \text{Bad}(s_i)$	$s_{k-1} = s_k$ (state repeats)
interpolant	$I(s)$: set of states	$\tau(s, s')$: transition predicate
fixed point	inductive invariant	
proves	no bad state reachable	

Idea: IMC for Termination

	IMC (reachability)	ITerm (termination)
query checks	$\forall_i \text{Bad}(s_i)$	$s_{k-1} = s_k$ (state repeats)
interpolant	$I(s)$: set of states	$\tau(s, s')$: transition predicate
fixed point	inductive invariant	inductive transition invariant
proves	no bad state reachable	

Idea: IMC for Termination

	IMC (reachability)	ITerm (termination)
query checks	$\forall_i \text{Bad}(s_i)$	$s_{k-1} = s_k$ (state repeats)
interpolant	$I(s)$: set of states	$\tau(s, s')$: transition predicate
fixed point	inductive invariant	inductive transition invariant
proves	no bad state reachable	irreflexive $\Rightarrow$ termination

Termination-to-Reachability as CPAs

composite CPA

Location CPA

program location

Predicate CPA

path formula (no abstr.)

Termination CPA

builds Φ_e , BMC query

Termination-to-Reachability as CPAs

composite CPA

Location CPA

program location

Predicate CPA

path formula (no abstr.)

Termination CPA

builds Φ_e , BMC query

path formula of loop-head state e after k iterations:

$$\Phi_e \equiv \underbrace{Init(s_0)}_{\text{to the loop head}} \wedge \underbrace{Tr_1(s_0, s_1)}_{\text{iteration 1}} \wedge \cdots \wedge \underbrace{Tr_k(s_{k-1}, s_k)}_{\text{iteration } k}$$

Termination-to-Reachability as CPAs

composite CPA

Location CPA

program location

Predicate CPA

path formula (no abstr.)

Termination CPA

builds Φ_e , BMC query

path formula of loop-head state e after k iterations:

$$\Phi_e \equiv \underbrace{Init(s_0)}_{\text{to the loop head}} \wedge \underbrace{Tr_1(s_0, s_1)}_{\text{iteration 1}} \wedge \cdots \wedge \underbrace{Tr_k(s_{k-1}, s_k)}_{\text{iteration } k}$$

BMC query of e :

$$\Phi_e \wedge \bigvee_{i < k} s_i = s_k$$

sat

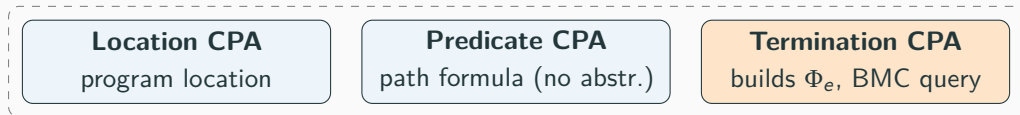
e is target state: **lasso**

unsat

unroll further

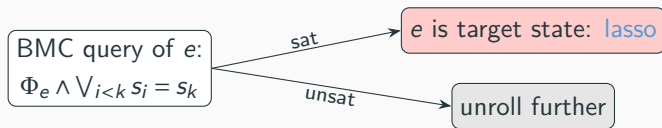
Termination-to-Reachability as CPAs

composite CPA



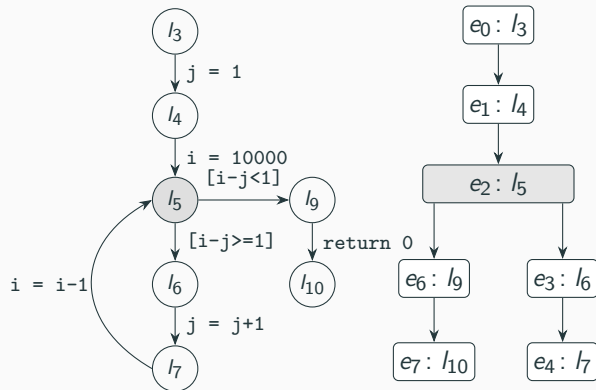
path formula of loop-head state e after k iterations:

$$\Phi_e \equiv \underbrace{Init(s_0)}_{\text{to the loop head}} \wedge \underbrace{Tr_1(s_0, s_1)}_{\text{iteration 1}} \wedge \cdots \wedge \underbrace{Tr_k(s_{k-1}, s_k)}_{\text{iteration } k}$$

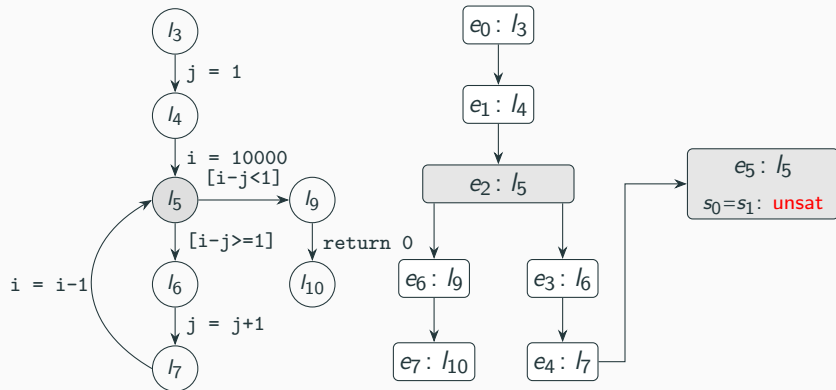


each ARG path has its own Tr_i
 $\Rightarrow$ precise query per path

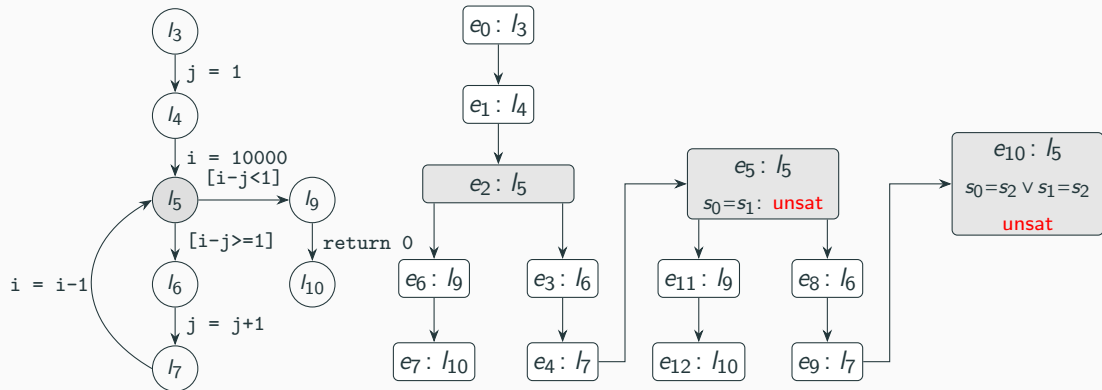
Termination-to-Reachability: ARG



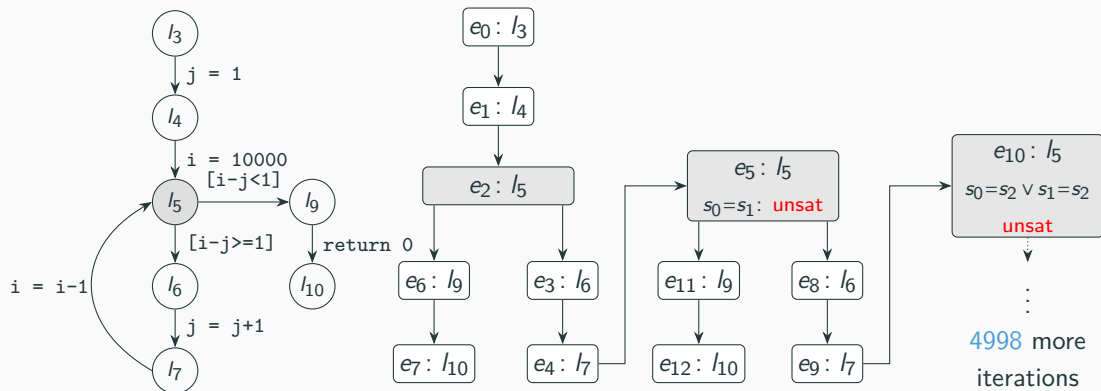
Termination-to-Reachability: ARG



Termination-to-Reachability: ARG

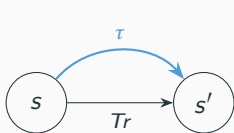


Termination-to-Reachability: ARG

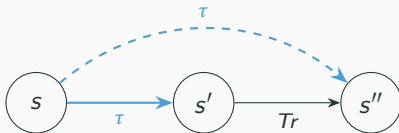


every BMC query unsat $\Rightarrow$ termination proved only after 5000 iterations

Inductive Transition Invariants (Podelski, Rybalchenko 2004 [5])



base
 $Tr \Rightarrow \tau$



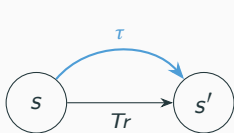
step
 $\tau(s, s') \wedge Tr(s', s'') \Rightarrow \tau(s, s'')$



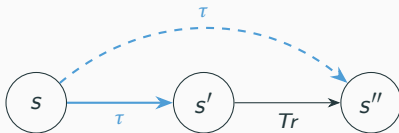
irreflexive
 $\tau(s, s)$ unsat

- transition predicate $\tau(s, s')$: relation over two copies of the variables

Inductive Transition Invariants (Podelski, Rybalchenko 2004 [5])



base
 $Tr \Rightarrow \tau$



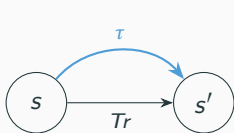
step
 $\tau(s, s') \wedge Tr(s', s'') \Rightarrow \tau(s, s'')$



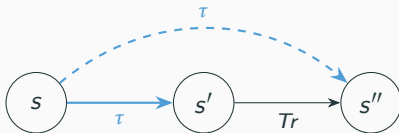
irreflexive
 $\tau(s, s) \text{ unsat}$

- transition predicate $\tau(s, s')$: relation over two copies of the variables
- inductive (base + step) $\Rightarrow Tr^+ \Rightarrow \tau$: **transition invariant**

Inductive Transition Invariants (Podelski, Rybalchenko 2004 [5])



base
 $Tr \Rightarrow \tau$



step
 $\tau(s, s') \wedge Tr(s', s'') \Rightarrow \tau(s, s'')$



irreflexive
 $\tau(s, s)$ unsat

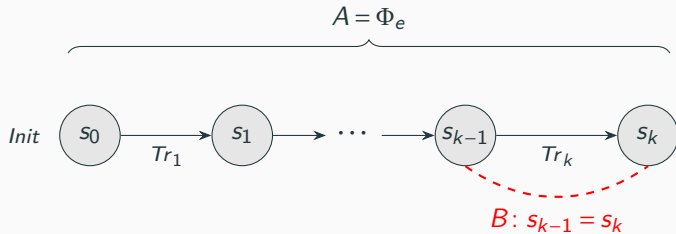
- transition predicate $\tau(s, s')$: relation over two copies of the variables
- inductive (base + step) $\Rightarrow Tr^+ \Rightarrow \tau$: **transition invariant**

Lemma (finite state space)

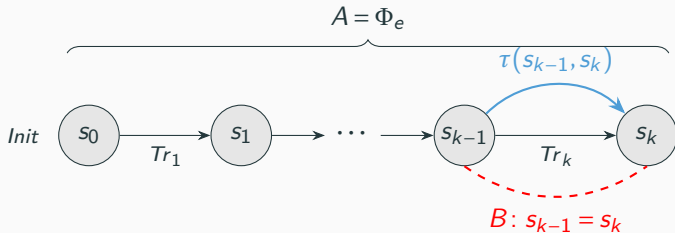
irreflexive transition invariant $\Rightarrow Tr^+$ well-founded $\Rightarrow$ termination

infinite execution repeats a state $s_i = s_j \Rightarrow \tau(s_i, s_i)$ **X**

Interpolants Are Transition Predicates

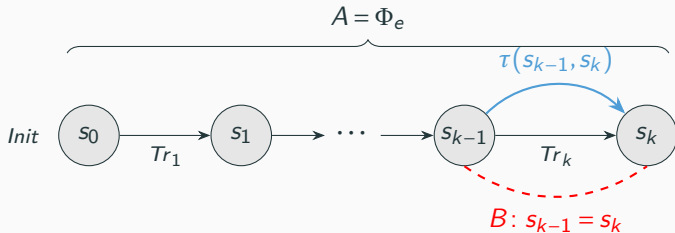


Interpolants Are Transition Predicates



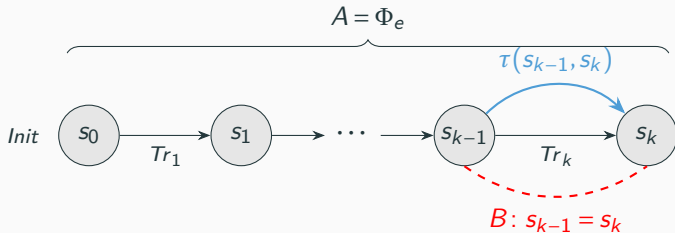
- $\Phi_e \Rightarrow \tau(s_{k-1}, s_k)$: overapproximates the last iteration

Interpolants Are Transition Predicates



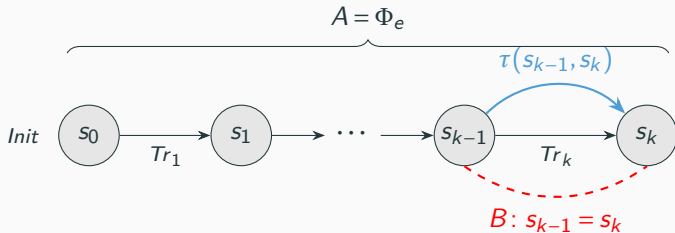
- $\Phi_e \Rightarrow \tau(s_{k-1}, s_k)$: overapproximates the last iteration
- $\tau(s_{k-1}, s_k) \wedge s_{k-1} = s_k$ unsat: **irreflexive** by construction ✓

Interpolants Are Transition Predicates



- $\Phi_e \Rightarrow \tau(s_{k-1}, s_k)$: overapproximates the last iteration
- $\tau(s_{k-1}, s_k) \wedge s_{k-1} = s_k$ unsat: **irreflexive** by construction ✓
- not necessarily inductive $\Rightarrow$ extend τ like IMC

Interpolants Are Transition Predicates

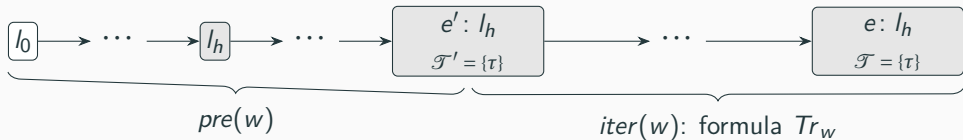


- $\Phi_e \Rightarrow \tau(s_{k-1}, s_k)$: overapproximates the last iteration
- $\tau(s_{k-1}, s_k) \wedge s_{k-1} = s_k$ unsat: **irreflexive** by construction ✓
- not necessarily inductive $\Rightarrow$ extend τ like IMC

Running example at e_5 :

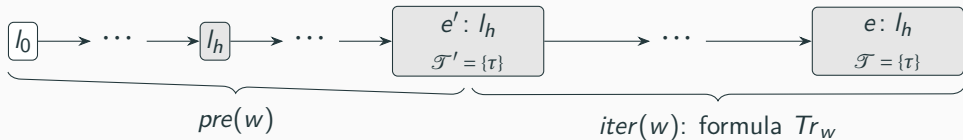
$$\Phi_{e_5} \wedge i_0 = i_1 \wedge j_0 = j_1 \text{ unsat} \longrightarrow ltp: i_1 \leq i_0 - 1 \longrightarrow \tau(s, s') \equiv i' \leq i - 1$$

Termination CPA $\mathbb{T}$



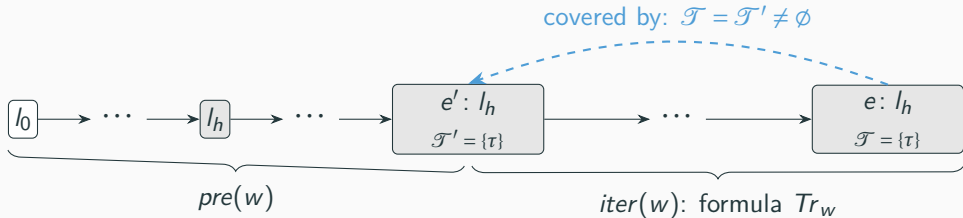
- $e = (w, \mathcal{P}, \mathcal{T})$: ARG-path locations, transition predicates, invariants $\mathcal{T} \subseteq \mathcal{P}$

Termination CPA $\mathbb{T}$



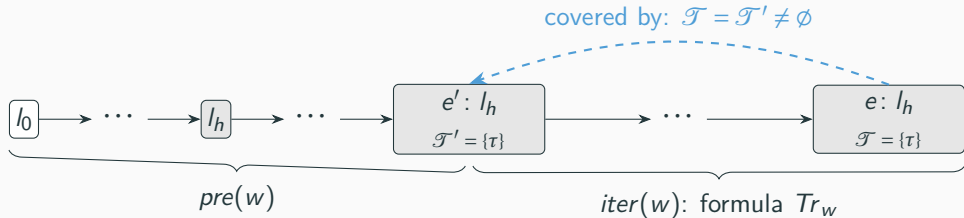
- $e = (w, \mathcal{P}, \mathcal{T})$: ARG-path locations, transition predicates, invariants $\mathcal{T} \subseteq \mathcal{P}$
- $\llbracket e \rrbracket$: pairs (s, s') , s reached via $pre(w)$, satisfying all $\tau \in \mathcal{T}$

Termination CPA $\mathbb{T}$



- $e = (w, \mathcal{P}, \mathcal{T})$: ARG-path locations, transition predicates, invariants $\mathcal{T} \subseteq \mathcal{P}$
- $\llbracket e \rrbracket$: pairs (s, s') , s reached via $pre(w)$, satisfying all $\tau \in \mathcal{T}$
- transfer: $(w, \mathcal{P}, \mathcal{T}) \rightsquigarrow (w \cdot l', \mathcal{P}, \emptyset)$ merge: sep stop: coverage

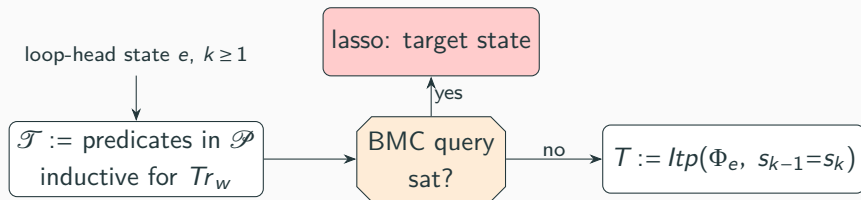
Termination CPA $\mathbb{T}$



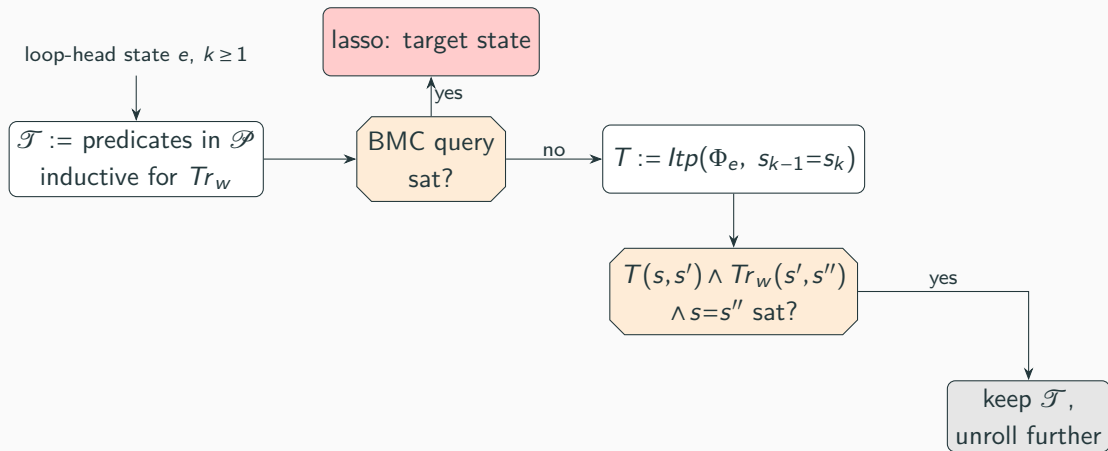
- $e = (w, \mathcal{P}, \mathcal{T})$: ARG-path locations, transition predicates, invariants $\mathcal{T} \subseteq \mathcal{P}$
- $\llbracket e \rrbracket$: pairs (s, s') , s reached via $pre(w)$, satisfying all $\tau \in \mathcal{T}$
- transfer: $(w, \mathcal{P}, \mathcal{T}) \rightsquigarrow (w \cdot l', \mathcal{P}, \emptyset)$ merge: sep stop: **coverage**

Theorem (Soundness): *false* $\Rightarrow$ non-termination, *true* $\Rightarrow$ termination

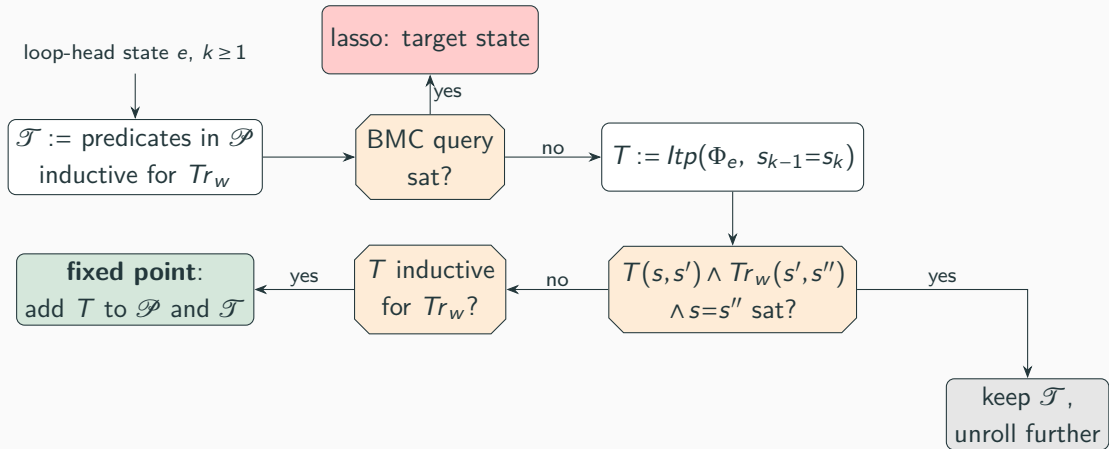
Precision Adjustment of $\mathbb{T}$ at Loop-Head State e



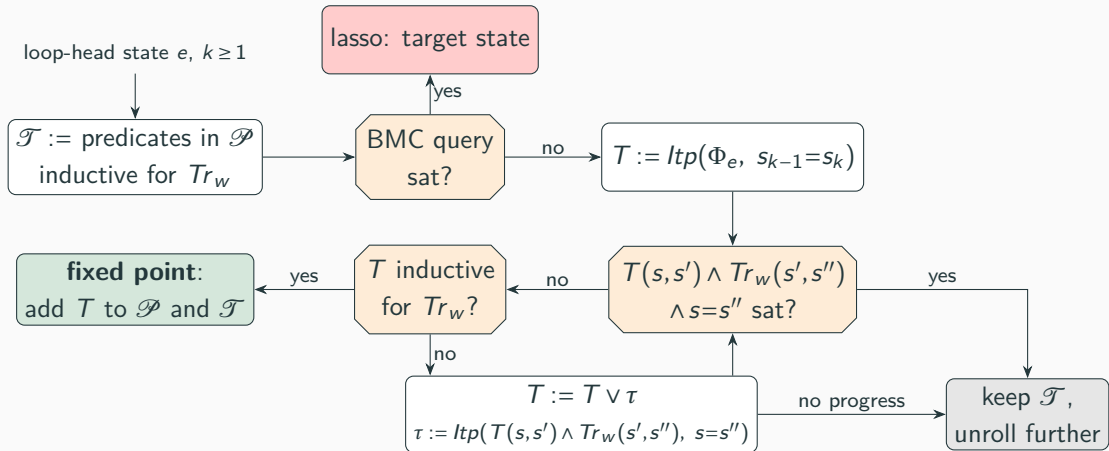
Precision Adjustment of $\mathbb{T}$ at Loop-Head State e



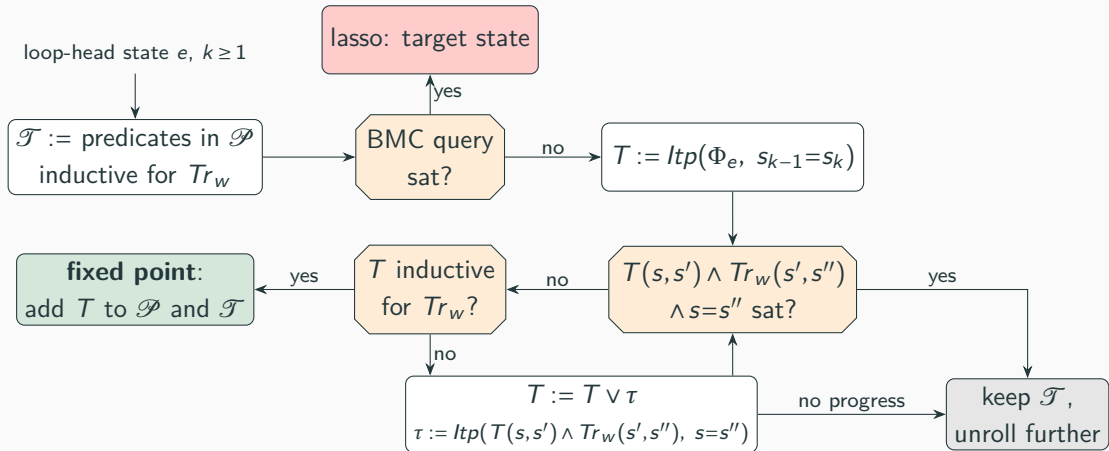
Precision Adjustment of $\mathbb{T}$ at Loop-Head State e



Precision Adjustment of $\mathbb{T}$ at Loop-Head State e



Precision Adjustment of $\mathbb{T}$ at Loop-Head State e



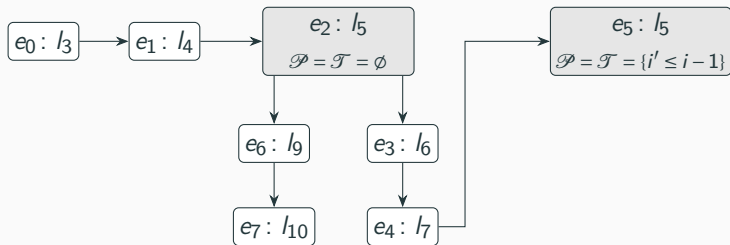
like IMC: reflexivity check $\approx$ “reaches a bad state” extension of $T \approx$ image computation

Running Example



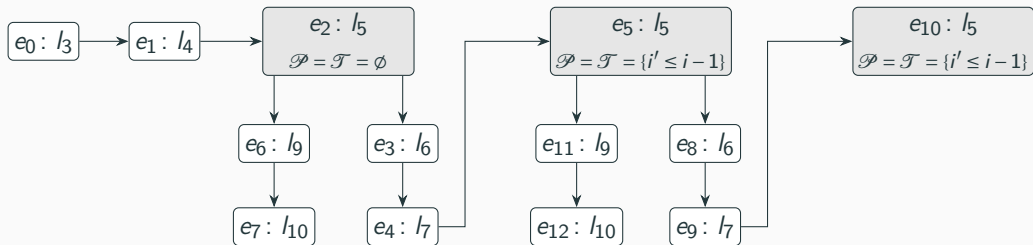
- e_2 : first visit of loop head l_5 , nothing to compare

Running Example



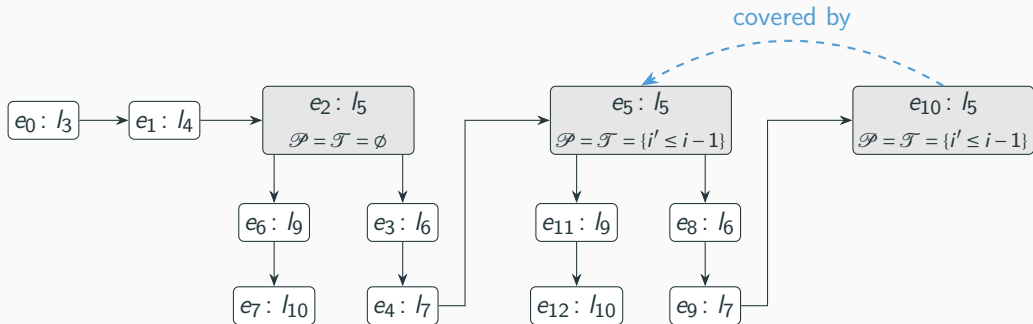
- e_5 : BMC query $\Phi_{e_5} \wedge s_0 = s_1$ unsat $\Rightarrow T \equiv i' \leq i - 1$
- T inductive for $Tr_w \equiv i - j \geq 1 \wedge j' = j + 1 \wedge i' = i - 1$ ✓ $\Rightarrow$ fixed point

Running Example



- $e_{10}: i' \leq i - 1 \in \mathcal{P}$ inductive for $Tr_w \Rightarrow \mathcal{T} = \{i' \leq i - 1\}$
- BMC query $\Phi_{e_{10}} \wedge (s_0 = s_2 \vee s_1 = s_2)$ unsat

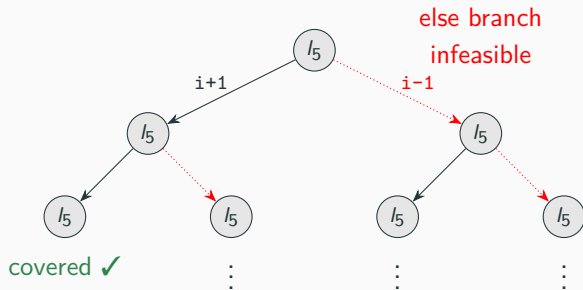
Running Example



- $\mathcal{T}(e_{10}) = \mathcal{T}(e_5)$ and e_5 is the previous iteration $\Rightarrow e_{10}$ covered
- waitlist empty $\Rightarrow$ **true**: 2 iterations instead of 5000

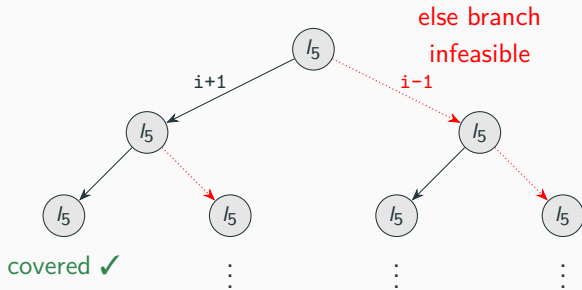
Combination with Value Analysis: Problem

```
1 int main(void) {  
2     int i, up;  
3     i = 0;  
4     up = 1;  
5     while (i < 10000) {  
6         if (up == 1) {  
7             i = i + 1;  
8         } else {  
9             i = i - 1;  
10        }  
11    }  
12    return 0;  
13 }
```



Combination with Value Analysis: Problem

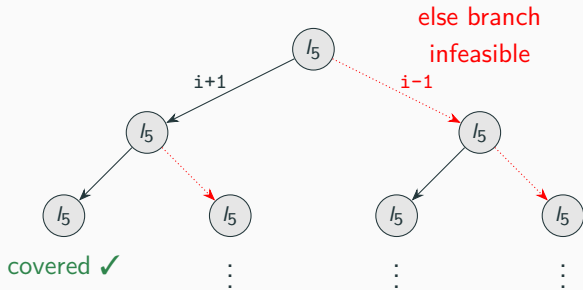
```
1 int main(void) {  
2     int i, up;  
3     i = 0;  
4     up = 1;  
5     while (i < 10000) {  
6         if (up == 1) {  
7             i = i + 1;  
8         } else {  
9             i = i - 1;  
10        }  
11    }  
12    return 0;  
13 }
```



- without value analysis:
 $\mathbb{T}$ explores the else branch

Combination with Value Analysis: Problem

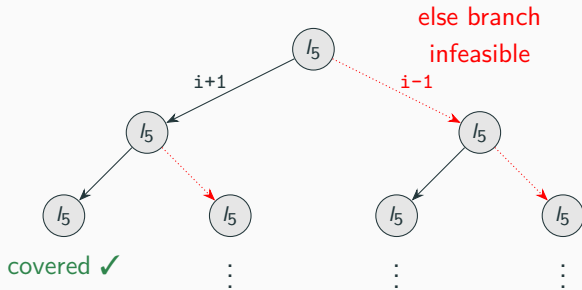
```
1 int main(void) {  
2   int i, up;  
3   i = 0;  
4   up = 1;  
5   while (i < 10000) {  
6     if (up == 1) {  
7       i = i + 1;  
8     } else {  
9       i = i - 1;  
10    }  
11  }  
12  return 0;  
13 }
```



- without value analysis:
 $\mathbb{T}$ explores the else branch
- $i' \geq i + 1$ not inductive there
 $\Rightarrow$ never covered

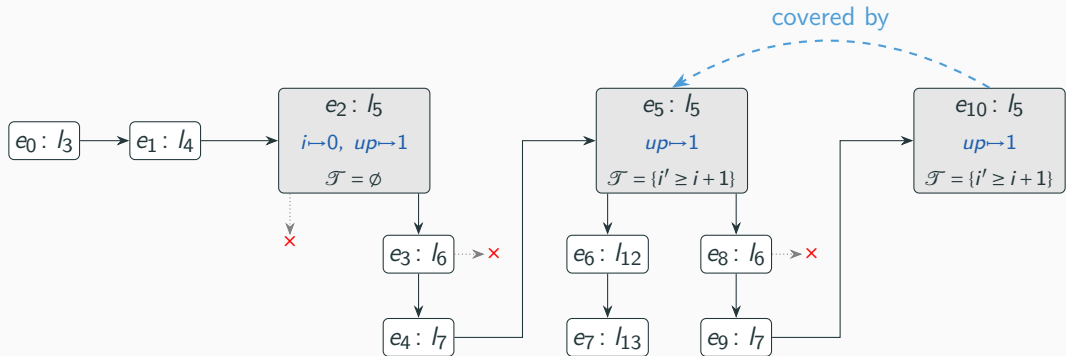
Combination with Value Analysis: Problem

```
1 int main(void) {  
2   int i, up;  
3   i = 0;  
4   up = 1;  
5   while (i < 10000) {  
6     if (up == 1) {  
7       i = i + 1;  
8     } else {  
9       i = i - 1;  
10    }  
11  }  
12  return 0;  
13 }
```



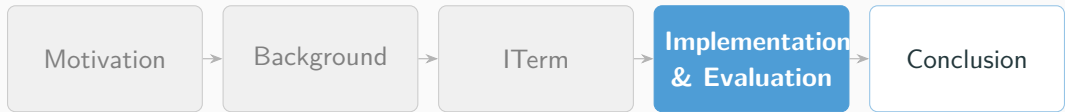
- without value analysis:
 $\top$ explores the else branch
- $i' \geq i + 1$ not inductive there
 $\Rightarrow$ never covered
- exponentially many ARG paths

Combination with Value Analysis $\mathbb{V}$ [1]



- $\mathbb{V}$ tracks concrete values and **prunes infeasible branches** (X)
- i differs between iterations $\Rightarrow \mathbb{V}$ keeps only $up \mapsto 1$
- only 2 iterations: e_{10} covered by e_5

Outline



CPACHECKER

RQ1, RQ2

Implementation in CPAchecker

Integer encoding

- bit vectors:
 $i' = i - 1$ ✗
(not inductive)
- integers:
 $i' \leq i - 1$ ✓

Unsigned overflows

- unsigned x ;
while ($x \geq 0$)
 $x = x - 1$;
- integers: $x' < x$ ✗
- fix: $0 \leq x \leq 2^{32}-1$ in
inductiveness check

Bitwise operations

- uninterpreted
functions
- transition invariants
still sound ✓
- lassos may be spurious
 $\Rightarrow$ no *false* verdict

Location

CallStack

Predicate

Value $\mathbb{V}$

Termination $\mathbb{T}$

breadth-first
exploration

Evaluation Setup

SV-Benchmarks, category Termination: 2533 tasks

1436 terminating

1097 non-terminating

BENCHEXEC [2]: 900 s CPU time, 7 GB memory, 1 core

CPA-BMC

termination-to-reachability with BMC

CPA-ITerm

$\mathbb{T}$ + value analysis

CPA-ITerm_{noVA}

$\mathbb{T}$ without value analysis

CPA-LassoRanker

ranking functions for lassos

CPA-Combination

portfolio: ITerm_{noVA} $\parallel$ BMC $\parallel$ LassoRanker

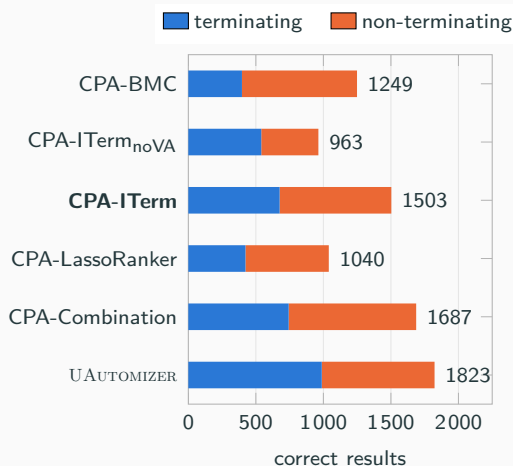
UAUTOMIZER

SV-COMP 2026 version

RQ1: ITerm vs. analyses
in CPAchecker

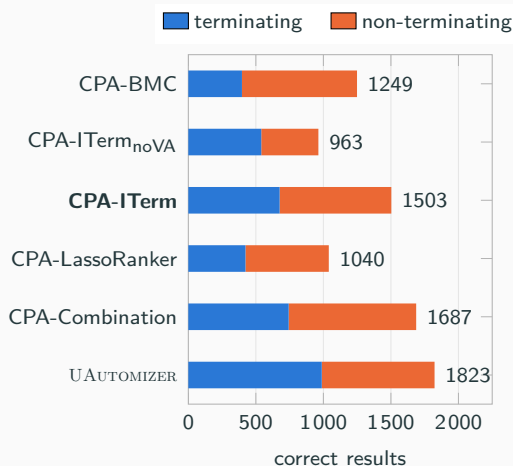
RQ2: ITerm vs.
UAUTOMIZER

Results: Correct Results



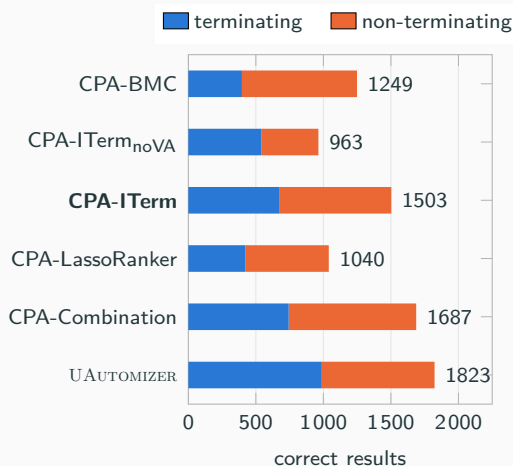
- most correct results in CPAchecker

Results: Correct Results



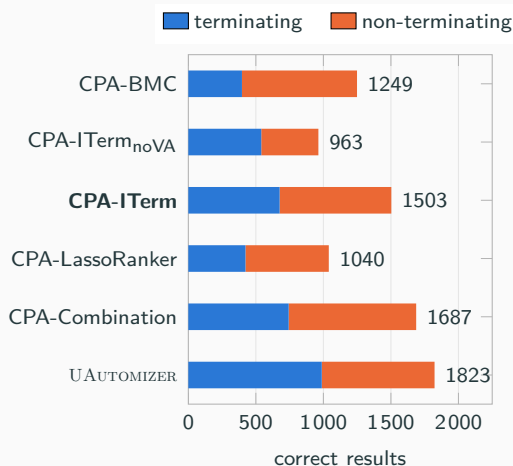
- most correct results in CPACHECKER
- vs. BMC: +286 terminating, -25 non-terminating

Results: Correct Results



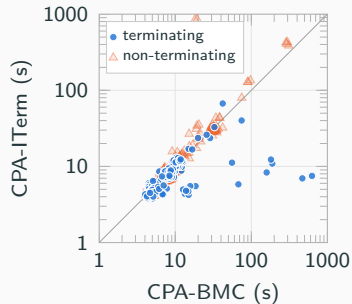
- most correct results in CPACHECKER
- vs. BMC: +286 terminating, -25 non-terminating
- value analysis: out of memory
874 → 114 tasks

Results: Correct Results

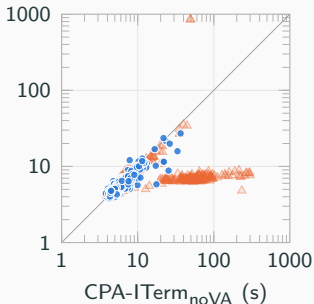


- most correct results in CPACHECKER
- vs. BMC: +286 terminating, -25 non-terminating
- value analysis: out of memory 874 → 114 tasks
- 215 terminating tasks solved by neither BMC nor LassoRanker

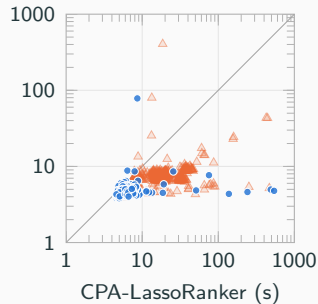
RQ1: CPU Time within CPAchecker



similar speed
(median ratio 0.96)



faster on 700 of 962
(172 twice as fast)

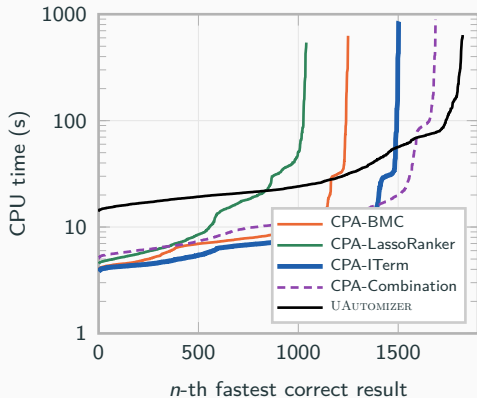


faster on 805 of 821
(median ratio 0.55)

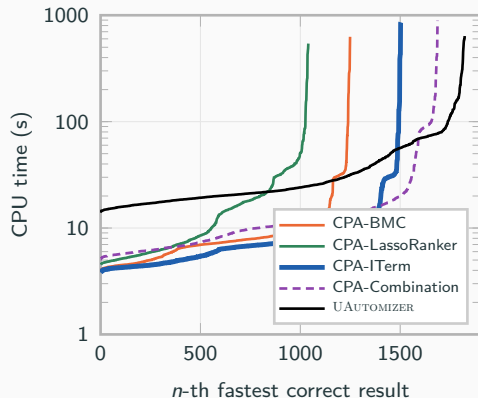
below diagonal: CPA-ITerm faster $\Rightarrow$ interpolation **does not slow down** unrolling

RQ2: Comparison with UAutomizer

1823 (U_{AUTOMIZER}) vs. 1503 (ITerm) correct results
ITerm faster on 1191 of 1205 common tasks



RQ2: Comparison with UAutomizer

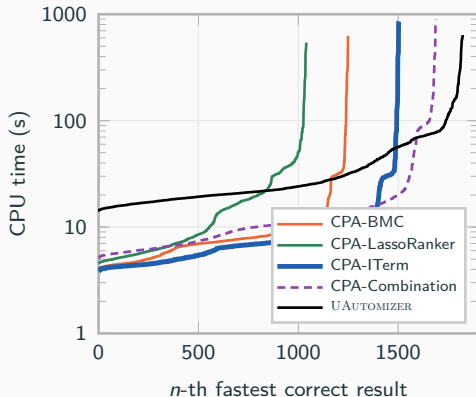


1823 (U_{AUTOMIZER}) vs. 1503 (ITerm) correct results
ITerm faster on 1191 of 1205 common tasks

Only ITerm (298)

- string traversals (104): $t' \geq t + 4$, no bound needed
- lassos in ECA programs (61)

RQ2: Comparison with UAutomizer



1823 (UAutomizer) vs. 1503 (ITerm) correct results
ITerm faster on 1191 of 1205 common tasks

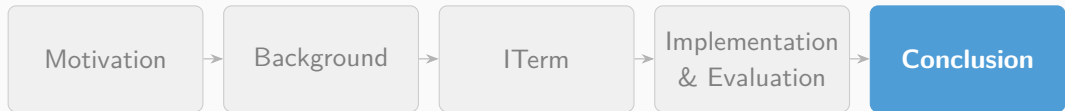
Only ITerm (298)

- string traversals (104): $t' \geq t + 4$, no bound needed
- lassos in ECA programs (61)

Only UAutomizer (618)

- recursion (107), bitwise operations (41)
- large ARGs, many branches (283)
- needs invariant before the loop (92)

Outline



summary
future work

Conclusion

- **ITerm:** IMC transferred from reachability to termination
 - interpolants of BMC queries = irreflexive transition predicates
 - extended to inductive transition invariants

Conclusion

- **ITerm:** IMC transferred from reachability to termination
 - interpolants of BMC queries = irreflexive transition predicates
 - extended to inductive transition invariants
- CPA on the ARG: sound, combines with value analysis

Conclusion

- **ITerm**: IMC transferred from reachability to termination
 - interpolants of BMC queries = irreflexive transition predicates
 - extended to inductive transition invariants
- CPA on the ARG: sound, combines with value analysis
- +286 terminating tasks vs. BMC, no slowdown
- complements LASSORANKER and UAUTOMIZER

Conclusion

- **ITerm**: IMC transferred from reachability to termination
 - interpolants of BMC queries = irreflexive transition predicates
 - extended to inductive transition invariants
- CPA on the ARG: sound, combines with value analysis
- +286 terminating tasks vs. BMC, no slowdown
- complements LASSORANKER and UAUTOMIZER

Future work

- bit-precise transition predicates
- interpolation sequences, IMC with auxiliary invariants for termination

References i

- [1] Beyer, D., Löwe, S.: Explicit-state software model checking based on CEGAR and interpolation. In: Proc. FASE. pp. 146–162. LNCS 7793, Springer (2013).
https://doi.org/10.1007/978-3-642-37057-1_11
- [2] Beyer, D., Löwe, S., Wendler, P.: Reliable benchmarking: Requirements and solutions. Int. J. Softw. Tools Technol. Transfer **21**(1), 1–29 (2019).
<https://doi.org/10.1007/s10009-017-0469-y>
- [3] Biere, A., Artho, C., Schuppan, V.: Liveness checking as safety checking. In: Proc. FMICS. pp. 160–177. No. 2 in ENTSC 66, Elsevier (2002).
[https://doi.org/10.1016/S1571-0661\(04\)80410-9](https://doi.org/10.1016/S1571-0661(04)80410-9)

- [4] McMillan, K.L.: Interpolation and SAT-based model checking. In: Proc. CAV. pp. 1–13. LNCS 2725, Springer (2003).
https://doi.org/10.1007/978-3-540-45069-6_1
- [5] Podelski, A., Rybalchenko, A.: Transition invariants. In: Proc. LICS. pp. 32–41. IEEE (2004). <https://doi.org/10.1109/LICS.2004.1319598>

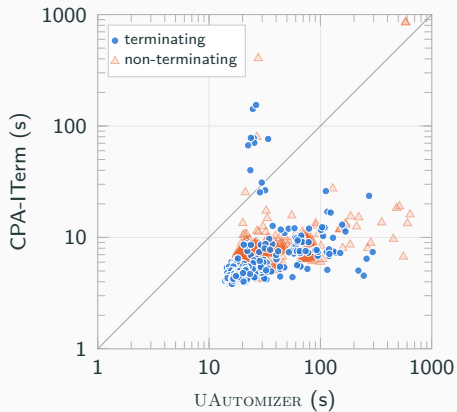
Backup: Precision Adjustment of $\mathbb{T}$

```
1:  $T := \{\tau \in P \mid \tau \text{ is inductive for } Tr_w\}$ 
2: if  $\Phi_e \wedge \bigvee_{i < k} s_i = s_k$  is satisfiable then
3:   return  $e$  as target state    // lasso found
4:  $T(s, s') := ltp(\Phi_e, s_{k-1} = s_k)$  with  $s_{k-1}, s_k$  renamed to  $s, s'$ 
5: loop
6:   if  $T(s, s') \wedge Tr_w(s', s'') \wedge s = s''$  is satisfiable then
7:     return  $(w, \mathcal{P}, \mathcal{T})$     //  $T$  too coarse
8:   if  $T$  is inductive for  $Tr_w$  then
9:     return  $(w, \mathcal{P} \cup \{T\}, \mathcal{T} \cup \{T\})$     // fixed point
10:   $\tau(s, s') := ltp(T(s, s') \wedge Tr_w(s', s''), s = s'')$  with  $s''$  renamed to  $s'$ 
11:  if  $\tau \Rightarrow T$  then
12:    return  $(w, \mathcal{P}, \mathcal{T})$     // no progress
13:   $T := T \vee \tau$ 
```

Backup: Results

Analysis	Correct			Unknown	Unique		
	total	true	false		all	UAUTOMIZER	w/o noVA
CPA-BMC	1249	397	852	1270	0	169	18
CPA-ITerm _{noVA}	963	540	423	1560	0	167	0
CPA-ITerm	1503	676	827	1015	0	298	215
CPA-LassoRanker	1040	423	617	1478	2	67	205
CPA-Combination	1687	743	944	835	0	348	1
UAUTOMIZER	1823	988	835	709	441	—	442

Backup: CPU Time against UAutomizer



CPA-ITerm solves 1315 tasks within 10 s; UAutomizer needs at least 13.9 s per task