Causality-based LTL Model Checking without Automata

joint work with Bernd Finkbeiner

Andrey Kupriyanov

Saarland University
Reactive Systems Group

October 23, 2014

Overview: evolution of the causality-based method

Overview: evolution of the causality-based method

Safety/Reachability

- $\pm$ $\forall\pi.\Box\Phi$ / $\exists\pi.\Diamond\Phi$
- ✓ Infinite-state
- ✓ Multi-threading

$$T_1 \parallel \ldots \parallel T_n \models \Box\neg(at_{l_3} \wedge at_{m_3})$$

[CONCUR 2013]

Overview: evolution of the causality-based method

Safety/Reachability

$\pm$ $\forall\pi.\Box\Phi$ / $\exists\pi.\Diamond\Phi$

✓ Infinite-state

✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models$
$\Box\neg(at_{l_3} \wedge at_{m_3})$

[CONCUR 2013]

Safety: the class of multi-threaded programs with binary locks and arbitrary control flow is analyzable in PTIME.

Overview: evolution of the causality-based method

Safety/Reachability

$\pm$ $\forall\pi.\Box\Phi$ / $\exists\pi.\Diamond\Phi$
✓ Infinite-state
✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models \Box\neg(at_{l_3} \wedge at_{m_3})$

[CONCUR 2013]

Liveness/Termination

$\pm$ $\forall\pi.\Diamond\Phi$ / $\exists\pi.\Box\Phi$
✓ Infinite-state
✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models at_{l_2} \implies \Diamond at_{l_3}$

[CAV 2014]

Safety: the class of multi-threaded programs with binary locks and arbitrary control flow is analyzable in PTIME.

Overview: evolution of the causality-based method

Safety/Reachability

$\pm$ $\forall\pi.\Box\Phi$ / $\exists\pi.\Diamond\Phi$
✓ Infinite-state
✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models$
$\Box\neg(at_{l_3} \wedge at_{m_3})$

[CONCUR 2013]

Liveness/Termination

$\pm$ $\forall\pi.\Diamond\Phi$ / $\exists\pi.\Box\Phi$
✓ Infinite-state
✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models$
$at_{l_2} \implies \Diamond at_{l_3}$

[CAV 2014]

Safety: the class of multi-threaded programs with binary locks and arbitrary control flow is analyzable in PTIME.

Termination: the first termination prover that scales to a large number of non-trivial concurrent threads.

	Terminator		T2		AProVE		**Arctor** [1]	
Threads	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)
1	3.37	26	2.42	38	3.17	237	**0.002**	**2.3**
2	1397	1394	3.25	44	6.79	523	**0.002**	**2.6**
3	×	MO	U(29.2)	253	U(26.6)	1439	**0.002**	**2.6**
4	×	MO	U(36.6)	316	U(71.2)	1455	**0.003**	**2.7**
5	×	MO	U(30.7)	400	U(312)	1536	**0.007**	**2.7**
10	×	MO	Z3-TO	×	×	MO	**0.027**	**3.0**
20	×	MO	Z3-TO	×	×	MO	**0.30**	**4.2**
40	×	MO	Z3-TO	×	×	MO	**4.30**	**12.7**
60	×	MO	Z3-TO	×	×	MO	**20.8**	**35**
80	×	MO	Z3-TO	×	×	MO	**67.7**	**145**
100	×	MO	Z3-TO	×	×	MO	**172**	**231**

[1] **Arctor** : **A**bstraction **R**efinement of **C**oncurrent **T**emporal **Or**derings (react.uni-saarland.de/tools/arctor/)

Overview: evolution of the causality-based method

Safety/Reachability		LTL Model Checking		Liveness/Termination
$\pm\ \forall\pi.\Box\Phi\ /\ \exists\pi.\Diamond\Phi$		✓ Full LTL		$\pm\ \forall\pi.\Diamond\Phi\ /\ \exists\pi.\Box\Phi$
✓ Infinite-state	$\Longrightarrow$	✓ Infinite-state	$\Longleftarrow$	✓ Infinite-state
✓ Multi-threading		✓ Multi-threading		✓ Multi-threading
$T_1 \parallel \ldots \parallel T_n \models \Box\neg(at_{l_3} \wedge at_{m_3})$		$T_1 \parallel \ldots \parallel T_n \models \Box\Diamond(at_{l_2} \wedge r > 0) \Longrightarrow \Box\Diamond at_{l_3}$		$T_1 \parallel \ldots \parallel T_n \models at_{l_2} \Longrightarrow \Diamond at_{l_3}$
[CONCUR 2013]				[CAV 2014]

Safety: the class of multi-threaded programs with binary locks and arbitrary control flow is analyzable in PTIME.

Termination: the first termination prover that scales to a large number of non-trivial concurrent threads.

	Terminator		T2		AProVE		**Arctor** [1]	
Threads	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)	Time(s)	Mem.(MB)
1	3.37	26	2.42	38	3.17	237	**0.002**	**2.3**
2	1397	1394	3.25	44	6.79	523	**0.002**	**2.6**
3	×	MO	U(29.2)	253	U(26.6)	1439	**0.002**	**2.6**
4	×	MO	U(36.6)	316	U(71.2)	1455	**0.003**	**2.7**
5	×	MO	U(30.7)	400	U(312)	1536	**0.007**	**2.7**
10	×	MO	Z3-TO	×	×	MO	**0.027**	**3.0**
20	×	MO	Z3-TO	×	×	MO	**0.30**	**4.2**
40	×	MO	Z3-TO	×	×	MO	**4.30**	**12.7**
60	×	MO	Z3-TO	×	×	MO	**20.8**	**35**
80	×	MO	Z3-TO	×	×	MO	**67.7**	**145**
100	×	MO	Z3-TO	×	×	MO	**172**	**231**

[1] **Arctor** : **A**bstraction **R**efinement of **C**oncurrent **T**emporal **Or**derings (react.uni-saarland.de/tools/arctor/)

LTL model checking

Automata-based LTL Model Checking

The standard way to model check a program P against an LTL property φ:

1. translate $\neg\varphi$ into a Büchi automaton A
2. check for emptiness the synchronized product of A and P

LTL model checking

Automata-based LTL Model Checking

The standard way to model check a program P against an LTL property φ:

1. translate $\neg\varphi$ into a Büchi automaton A
2. check for emptiness the synchronized product of A and P

Main problem: LTL formulas are often not small!

They describe necessary assumptions of fairness, termination, event sequences, ...

LTL model checking

Automata-based LTL Model Checking

The standard way to model check a program P against an LTL property φ:

1. translate $\neg\varphi$ into a Büchi automaton A
2. check for emptiness the synchronized product of A and P

Main problem: LTL formulas are often not small!

They describe necessary assumptions of fairness, termination, event sequences, ...

Example: individual accessibility for semaphores

Fair scheduling: $\Box\Diamond(at_2 \wedge r_{free}) \implies \Box\Diamond at_3$

Termination of critical sections: $\Box(at_3 \implies \Diamond at_1)$

Individual accessibility: $\Box(at_2 \implies \Diamond at_3)$

$$\varphi \equiv \bigwedge_{i \in 1..n}(Scheduling_i \wedge Termination_i) \implies Accessibility_1$$

Translation of $\neg\varphi$ into a Büchi automaton, **ltl3ba**:

Threads	Time (sec)	|Automaton| (MB)
2	0.005	0.002
3	0.09	0.38
4	9.6	8.6
5	1295	185
6	TO	X

Our approach

Causality

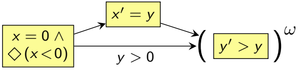
A relationship between two events, when the occurrence of first event is recognized as a necessary prerequisite for the occurrence of the second

Our approach

Causality

A relationship between two events, when the occurrence of first event is recognized as a necessary prerequisite for the occurrence of the second

- **Proof objects: concurrent traces**
  compactly represent sets of program runs, by specifying events that should *necessarily* occur in the run, and the partial order between them



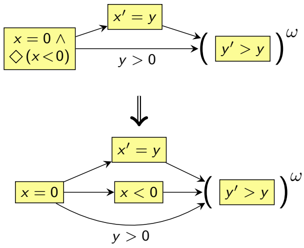
Our approach

Causality

A relationship between two events, when the occurrence of first event is recognized as a necessary prerequisite for the occurrence of the second

- **Proof objects: concurrent traces**
  compactly represent sets of program runs, by specifying events that should *necessarily* occur in the run, and the partial order between them

- **Proof rules based on causality**
  goal-directed, language-preserving trace transformations

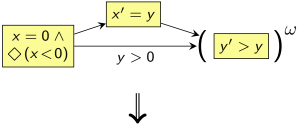


Our approach

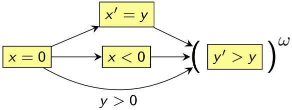
Causality

A relationship between two events, when the occurrence of first event is recognized as a necessary prerequisite for the occurrence of the second

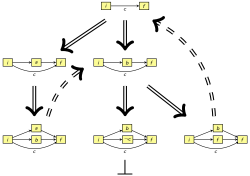
- **Proof objects: concurrent traces**
  compactly represent sets of program runs, by specifying events that should *necessarily* occur in the run, and the partial order between them



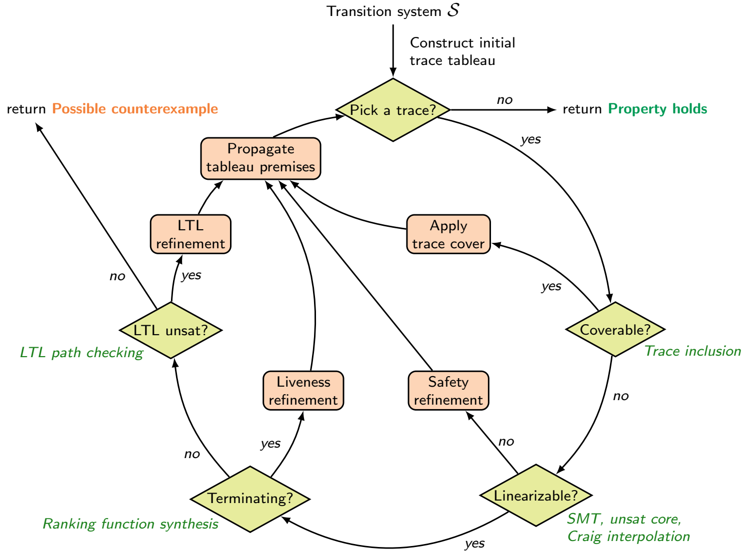
- **Proof rules based on causality**
  goal-directed, language-preserving trace transformations



- **Proof construction: tableau search based on causal loops**
  causal loops $\equiv$ infinitely-looping trace transformations
  - root trace captures all possible counterexamples
  - tableau branches according to applications of proof rules
  - termination when all leaves are contradictory, or covered by causal loops



LTL Model Checking Algorithm



Conclusion

Safety/Reachability

$\pm$ $\forall\pi.\Box\Phi$ / $\exists\pi.\Diamond\Phi$
✓ Infinite-state
✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models$
$\Box\neg(at_{l_3} \wedge at_{m_3})$

[CONCUR 2013]

$\Longrightarrow$

LTL Model Checking

✓ Full LTL
✓ Infinite-state
✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models$
$\Box\Diamond(at_{l_2} \wedge r > 0) \Longrightarrow \Box\Diamond\, at_{l_3}$

$\Longleftarrow$

Liveness/Termination

$\pm$ $\forall\pi.\Diamond\Phi$ / $\exists\pi.\Box\Phi$
✓ Infinite-state
✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models$
$at_{l_2} \Longrightarrow \Diamond\, at_{l_3}$

[CAV 2014]

Result: the class of multi-threaded programs with binary locks is analyzable in PTIME

Result: the first termination prover that scales to a large number of concurrent threads

Conclusion

Safety/Reachability

$\pm$ $\forall \pi . \Box \Phi$ / $\exists \pi . \Diamond \Phi$

✓ Infinite-state

✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models \Box \neg (at_{l_3} \wedge at_{m_3})$

$\Longrightarrow$

LTL Model Checking

✓ Full LTL

✓ Infinite-state

✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models \Box \Diamond (at_{l_2} \wedge r > 0) \Longrightarrow \Box \Diamond at_{l_3}$

$\Longleftarrow$

Liveness/Termination

$\pm$ $\forall \pi . \Diamond \Phi$ / $\exists \pi . \Box \Phi$

✓ Infinite-state

✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models at_{l_2} \Longrightarrow \Diamond at_{l_3}$

[CONCUR 2013]

Result: the class of multi-threaded programs with binary locks is analyzable in PTIME

Preliminary results: exponentially more concise proofs for some classes of programs, compared to standard automata-based methods

[CAV 2014]

Result: the first termination prover that scales to a large number of concurrent threads

Conclusion

Safety/Reachability

$\pm$ $\forall\pi.\Box\Phi$ / $\exists\pi.\Diamond\Phi$
✓ Infinite-state
✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models \Box\neg(at_{l_3} \wedge at_{m_3})$

[CONCUR 2013]

Result: the class of multi-threaded programs with binary locks is analyzable in PTIME

$\Longrightarrow$

LTL Model Checking

✓ Full LTL
✓ Infinite-state
✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models \Box\Diamond(at_{l_2} \wedge r > 0) \implies \Box\Diamond\, at_{l_3}$

Preliminary results: exponentially more concise proofs for some classes of programs, compared to standard automata-based methods

$\Longleftarrow$

Liveness/Termination

$\pm$ $\forall\pi.\Diamond\Phi$ / $\exists\pi.\Box\Phi$
✓ Infinite-state
✓ Multi-threading

$T_1 \parallel \ldots \parallel T_n \models at_{l_2} \implies \Diamond\, at_{l_3}$

[CAV 2014]

Result: the first termination prover that scales to a large number of concurrent threads

check my PhD thesis
(coming soon)

Conclusion

Safety/Reachability		LTL Model Checking		Liveness/Termination
$\pm\ \forall\pi.\Box\Phi\ /\ \exists\pi.\Diamond\Phi$		✓ Full LTL		$\pm\ \forall\pi.\Diamond\Phi\ /\ \exists\pi.\Box\Phi$
✓ Infinite-state		✓ Infinite-state		✓ Infinite-state
✓ Multi-threading	$\Longrightarrow$	✓ Multi-threading	$\Longleftarrow$	✓ Multi-threading
$T_1 \parallel \ldots \parallel T_n \models \Box\neg(at_{l_3} \wedge at_{m_3})$		$T_1 \parallel \ldots \parallel T_n \models \Box\Diamond(at_{l_2} \wedge r > 0) \Longrightarrow \Box\Diamond at_{l_3}$		$T_1 \parallel \ldots \parallel T_n \models at_{l_2} \Longrightarrow \Diamond at_{l_3}$

[CONCUR 2013]

Result: the class of multi-threaded programs with binary locks is analyzable in PTIME

Preliminary results: exponentially more concise proofs for some classes of programs, compared to standard automata-based methods

[CAV 2014]

Result: the first termination prover that scales to a large number of concurrent threads

check my PhD thesis
(coming soon)

Want to learn more? See the poster, and talk to me!