Certified Timed Automata Model-Checking

Frédéric Herbreteau (LaBRI, Bordeaux INP)

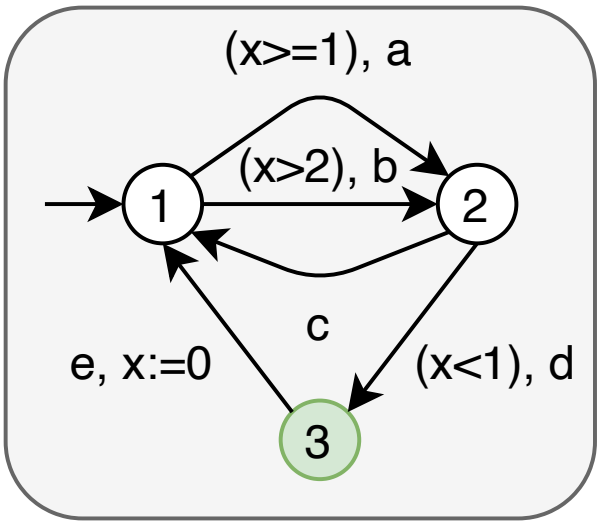
December 3rd, 2020
ANR TickTac on-line meeting

joint work with: Simon Wiemmer (TU München), Jaco Van de Pol (U. Aarhus)

Timed automata model-checking

State reachability in timed automata

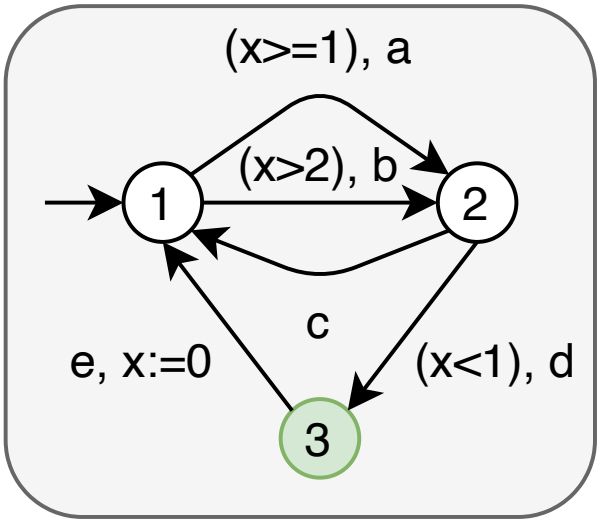
Timed automaton $\mathcal{A}$



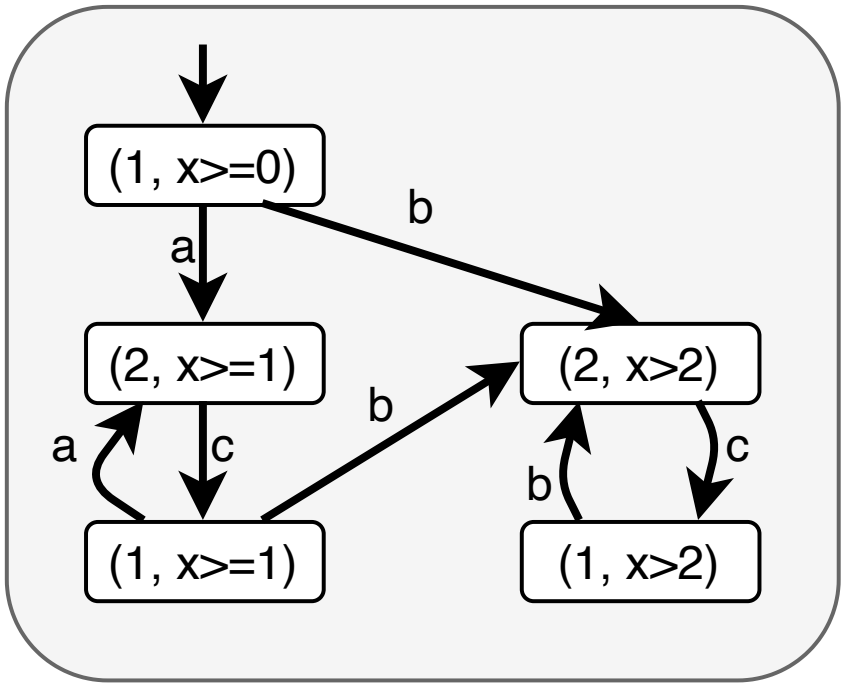
- **Run:** $(1, x = 0) \xrightarrow{1.2,a} (2, x = 1.2) \xrightarrow{0.4,c} (1, x = 1.6) \xrightarrow{0.6,b} (2, x = 2.2)$

State reachability in timed automata

Timed automaton $\mathcal{A}$



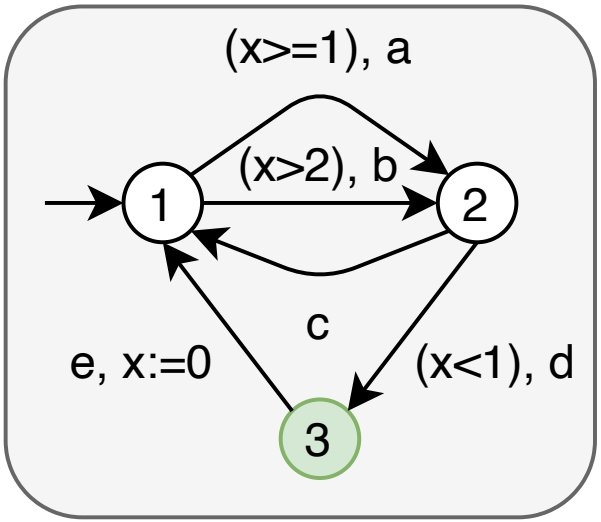
Zone graph $\mathcal{ZG}$



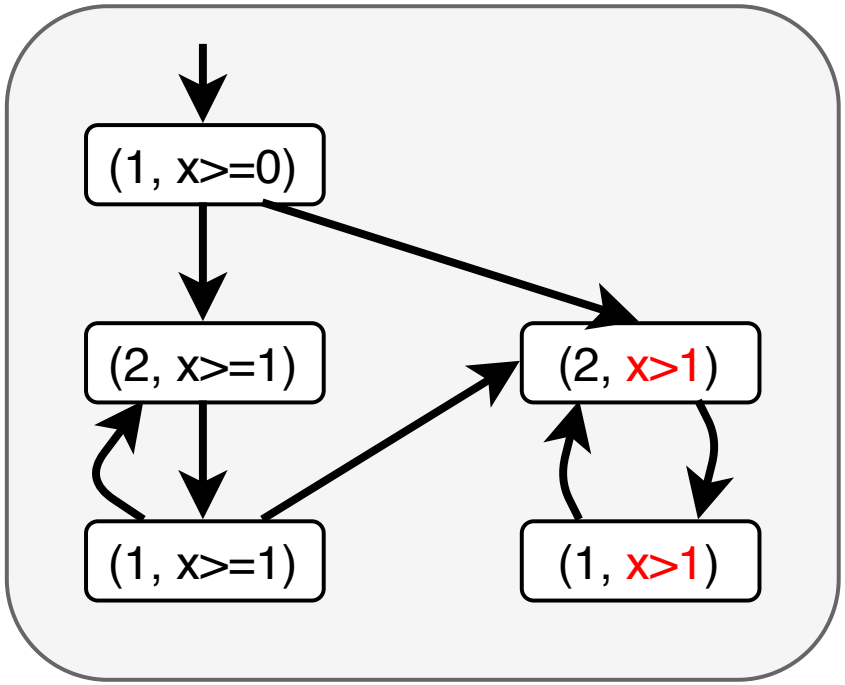
- **Run:** $(1, x = 0) \xrightarrow{1.2,a} (2, x = 1.2) \xrightarrow{0.4,c} (1, x = 1.6) \xrightarrow{0.6,b} (2, x = 2.2)$
- There is a run to 3 in $\mathcal{A}$ iff there is a path to $(3, Z)$ $\mathcal{ZG}$

State reachability in timed automata

Timed automaton $\mathcal{A}$



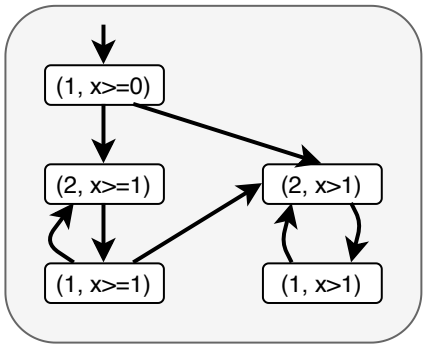
Finite Zone graph $\mathcal{ZG}$



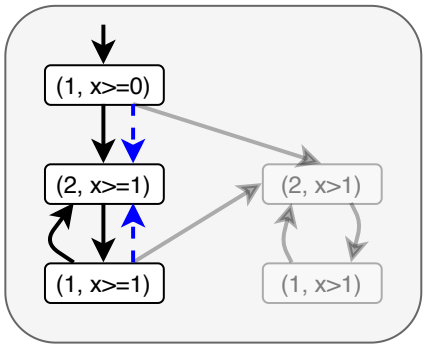
- **Run:** $(1, x = 0) \xrightarrow{1.2,a} (2, x = 1.2) \xrightarrow{0.4,c} (1, x = 1.6) \xrightarrow{0.6,b} (2, x = 2.2)$
- There is a run to 3 in $\mathcal{A}$ iff there is a path to $(3, Z)$ $\mathcal{ZG}$
- **Finiteness** of the zone graph ensured by zone extrapolation

Subsumption graphs, reachability algorithm

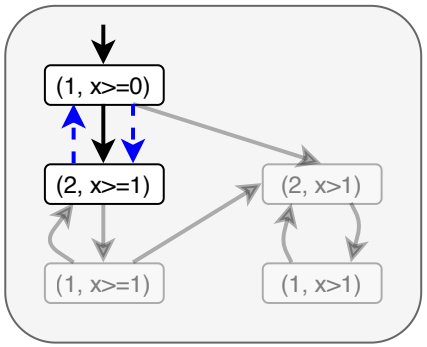
Zone graph $\mathcal{ZG}$



Subsumption graph 1

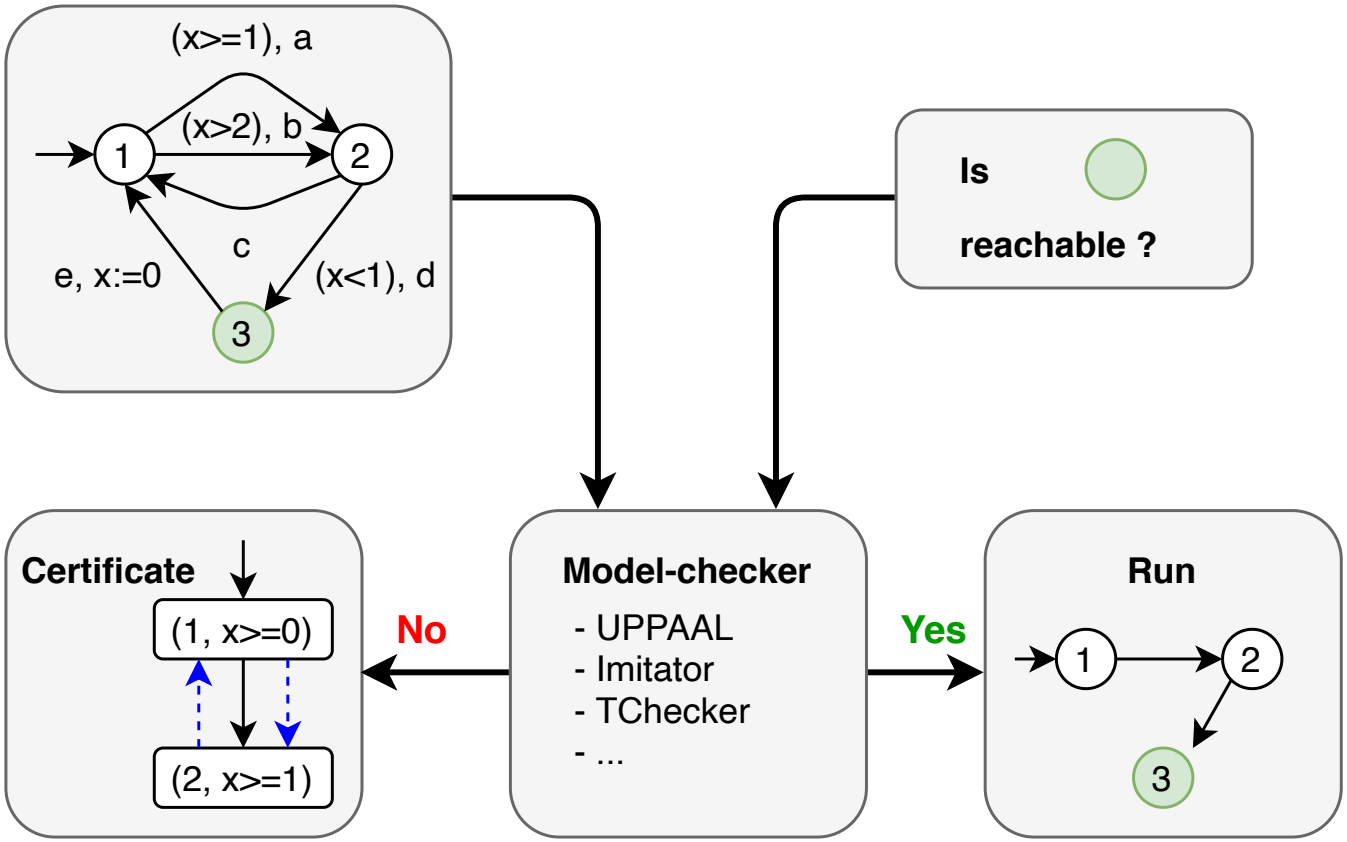


Subsumption graph 2



- **trace inclusion** when $(q, Z) \subseteq (q, Z')$, i.e. $Z \subseteq Z'$
- Standard reachability algorithm: state-space traversal with:
  - Skip (q, Z) if **covered** by some visited node (q, Z')
  - Only keep **maximal nodes**
- The three graphs above are **certificates of unreachability** of 3 in $\mathcal{A}$.

Confidence in model-checkers?

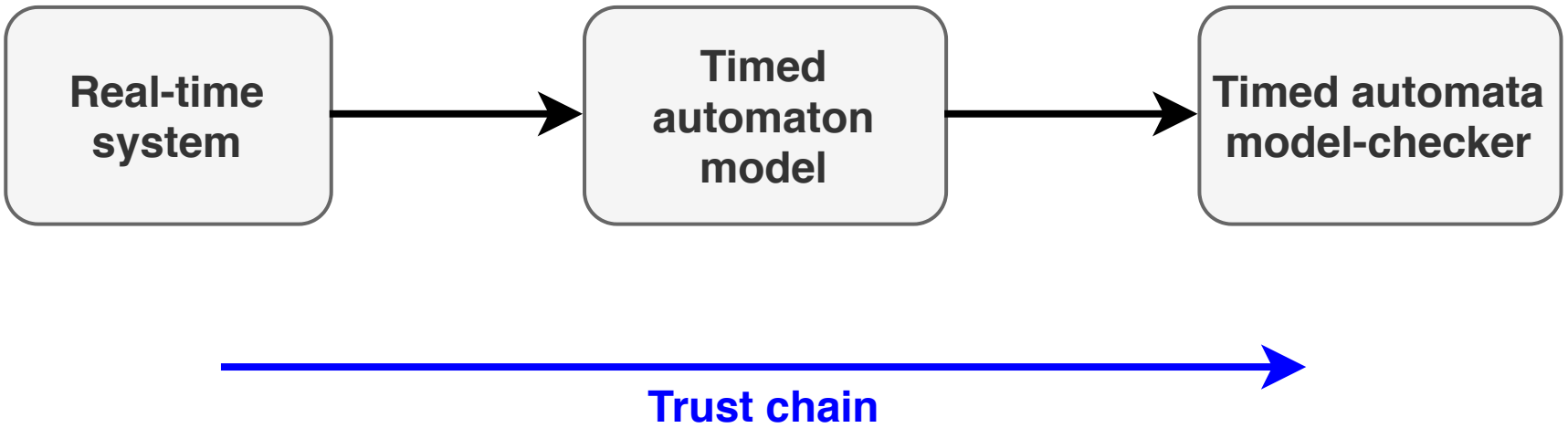


Implementations include:

- unexpected features (bugs)
- optimizations
- misused techniques (extrapolations vs. diagonal constraints)

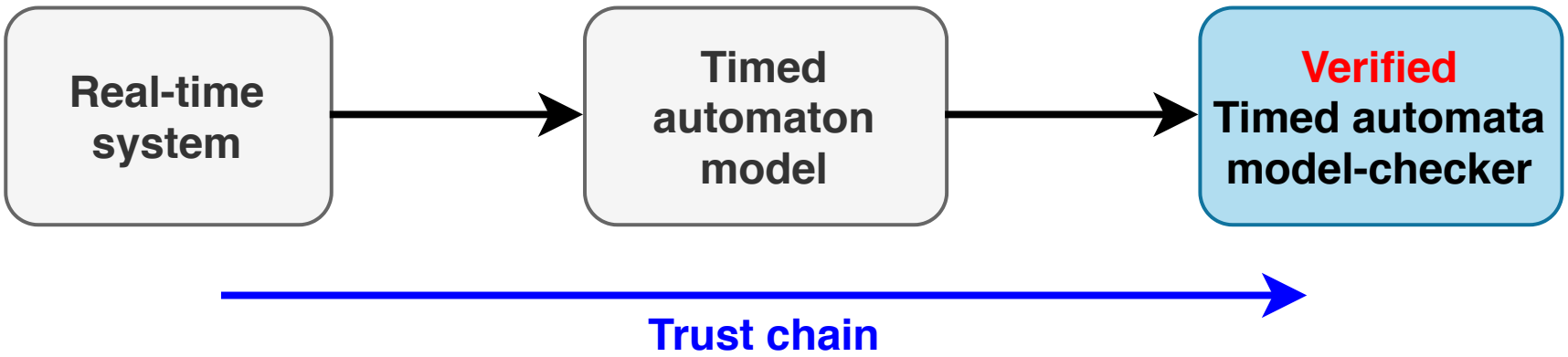
Towards verified model-checking

Trust reduction



Make the model-checker **reliable** using Interactive Theorem Proving

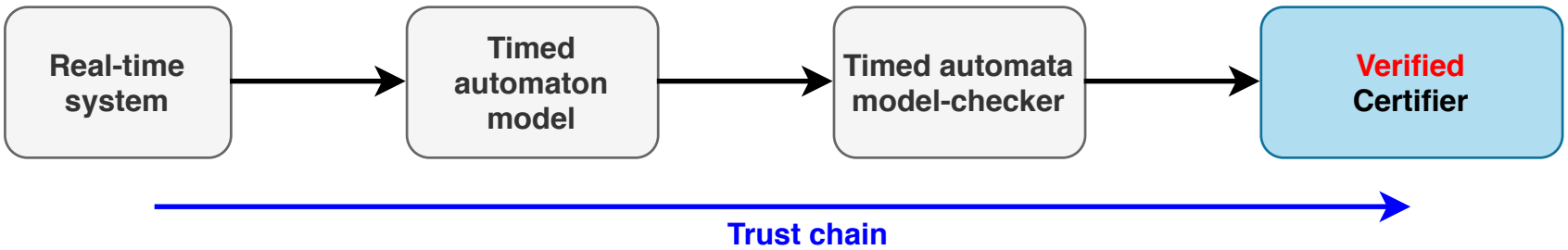
Verified model-checker



- Formalize Timed Automata theory in ITP (regions, zones, post, extrapolations, algorithms, data structures, etc)
- Construct model-checker in ITP
- Prove that the model-checker is correct

But: Optimizations? Performances?

Certified model-checking



Observe: model-checkers output a **certificate**

- Formalize Timed Automata theory in ITP (~~regions~~, zones, post, ~~extrapolations~~, ~~algorithms~~, ~~data structures~~, etc)
- Construct ~~model-checker~~ certifier in ITP
- Prove that the ~~model-checker~~ certifier is correct

Pros and Cons

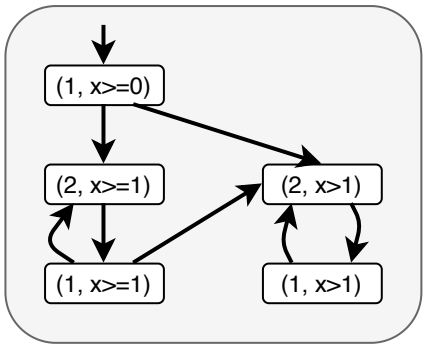
Verified MC	**Certified MC**
+ Soundness & Completeness	- Only soundness
+ Standalone tool	+ Less formalization
- Performances	+ Certifies a class of algorithms
	+ Parallel checking

- Verified C compiler CompCert [LBK+16]
- Verified LTL model-checking of finite state systems [ELN+13]
- Certified SAT solving [HHKW17, Lam17]
- TA: verified reachability model-checker [WL18], unreachability certifier [WvM20]

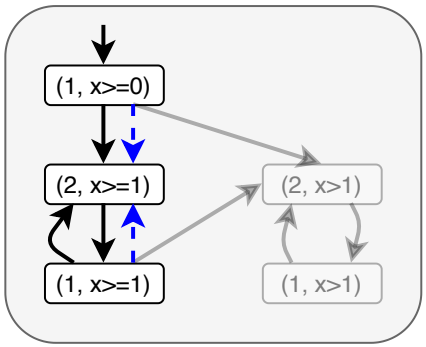
Certified reachability model-checking

Certifying reachability subsumption graphs

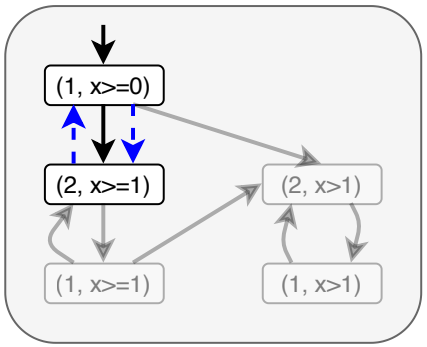
Zone graph $\mathcal{ZG}$



Subsumption graph 1



Subsumption graph 2



Reachability invariants

Finite set I of nodes (q, Z) s.t.

- for any $(q, Z) \in I$ if $(q, Z) \to (q', Z')$ in $\mathcal{ZG}$, there is $(q', Z'') \in I$ s.t. $Z' \subseteq Z''$
- and the initial node (q_0, Z_0) of $\mathcal{ZG}$ is covered by some $(q_0, Z_0') \in I$, i.e. $Z_0 \subseteq Z_0'$

Properties of reachability invariants

Reachability invariants

Finite set I of nodes (q, Z) s.t.

- for any $(q, Z) \in I$ if $(q, Z) \to (q', Z')$ in $\mathcal{ZG}$, there is $(q', Z'') \in I$ s.t. $Z' \subseteq Z''$
- and the initial node (q_0, Z_0) of $\mathcal{ZG}$ is covered by some $(q_0, Z_0') \in I$, i.e. $Z_0 \subseteq Z_0'$

Let $(q, Z) \Rightarrow_I (q', Z')$ if $(q, Z) \to (q', Z'')$ and $Z'' \subseteq Z'$

If there is a run of $\mathcal{A}$: $(q_0, v_0) \to (q_1, v_1) \to \cdots \to (q_n, v_n)$
then, there is a path: $(q_0, Z_0) \to (q_1, Z_1) \to \cdots \to (q_n, Z_n)$ in $\mathcal{ZG}$
then, there is a path: $(q_0, Z_0') \Rightarrow_I (q_1, Z_1') \Rightarrow_I \cdots \Rightarrow_I (q_n, Z_n')$ in I

I is a certificate of **unreachability** if I does not contain 

Certification algorithm (unreachability) [WvM20]

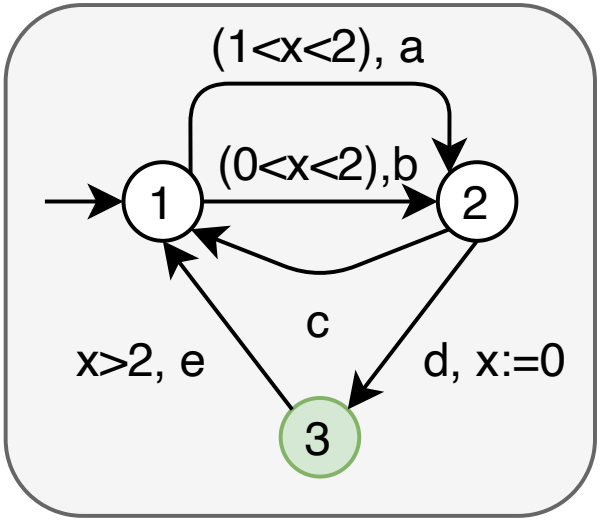
```
1  procedure certify_unreachability(A, C):
2    initial_state_covered := false;
3    for each (q, Z) ∈ C:
4      if Z = ∅ or Z is not canonical:
5        reject C
6      if q is accepting:
7        reject C
8      if q = q_0 and {0} ⊆ Z:
9        initial_state_covered := true;
10     for each (q, Z) → (q', Z'):
11       if ∄(q', Z'') ∈ C s.t. Z' ⊆ Z'':
12         reject C
13   if initial_state_covered:
14     accept C
15   else
16     reject C
```

- **Efficient**, can easily be **parallelized**
- Formalization in ITP: **zones**, **successor computation** and **simulation theorems**
- Agnostic to **extrapolation** and **reachability algorithm**

Certifying liveness model-checking

Liveness for timed automata

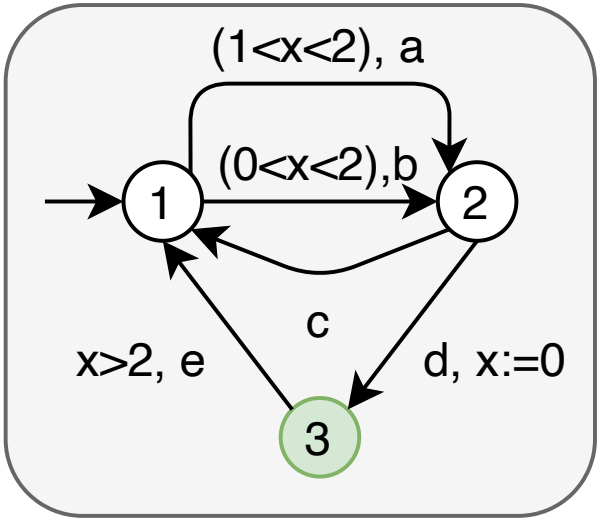
Timed Büchi automaton $\mathcal{A}$



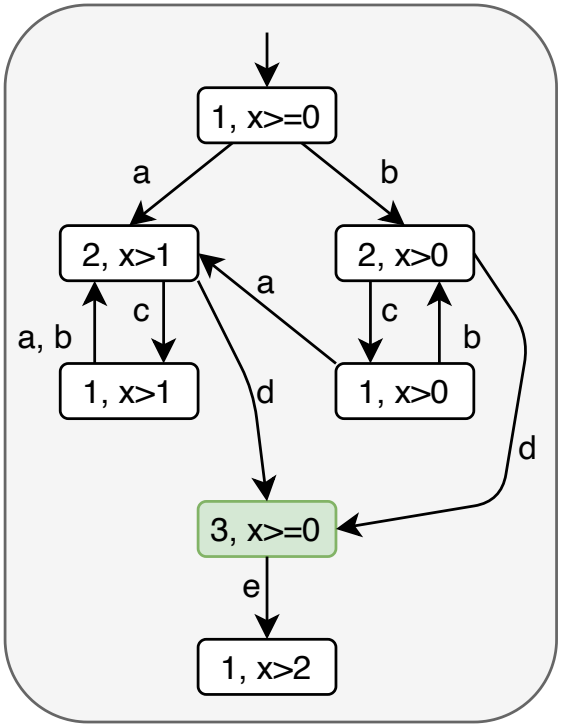
- **Infinite runs:** $(1, x = 0) \xrightarrow{0.1,b} (2, x = 0.1) \xrightarrow{0.3,c} (1, x = 0.4) \cdots$

Liveness for timed automata

Timed Büchi automaton $\mathcal{A}$



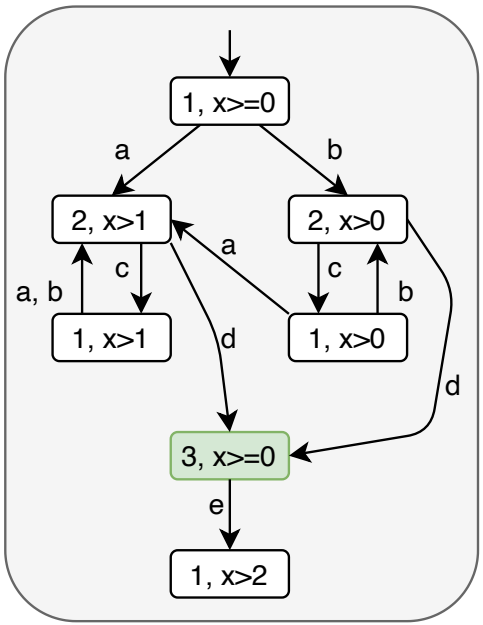
Zone graph $\mathcal{ZG}$



- **Infinite runs:** $(1, x = 0) \xrightarrow{0.1,b} (2, x = 0.1) \xrightarrow{0.3,c} (1, x = 0.4) \cdots$
- **Finiteness** of the zone graph ensured by zone extrapolation
- There is a run that visits 3 **infinitely often** in $\mathcal{A}$ iff there is a path that visits $(3, Z)$ **infinitely often** in $\mathcal{ZG}$

Subsumption graphs and liveness

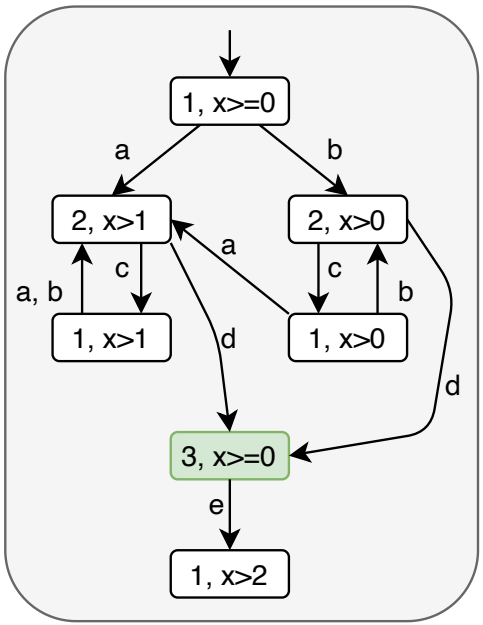
Zone graph $\mathcal{ZG}$ $\surd$



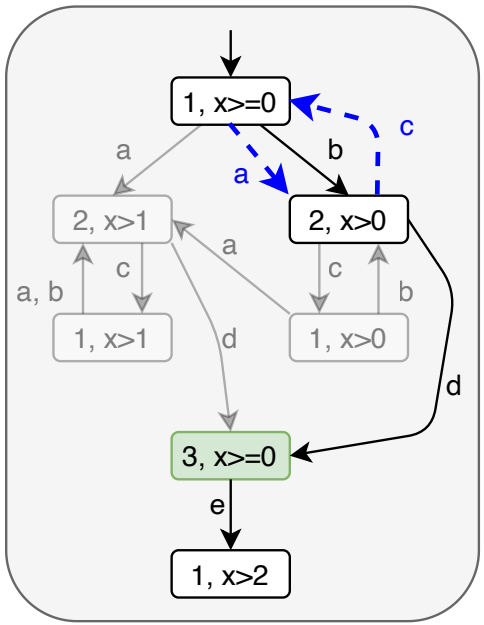
- A subsumption graph with **no accepting cycle** is a liveness **certificate**

Subsumption graphs and liveness

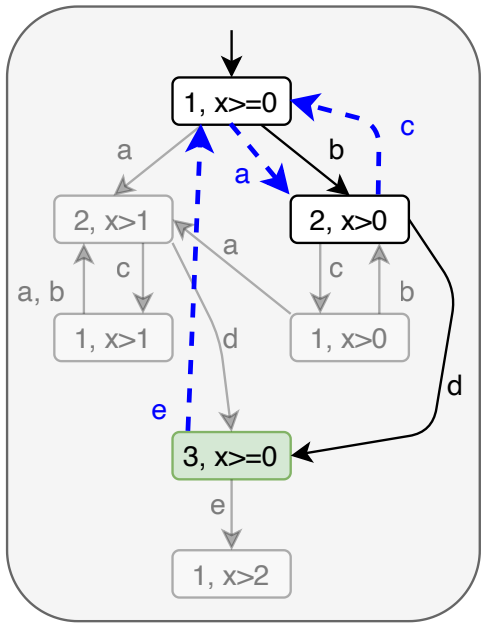
Zone graph $\mathcal{ZG}$ ✓



Subsumption graph 1 ✓



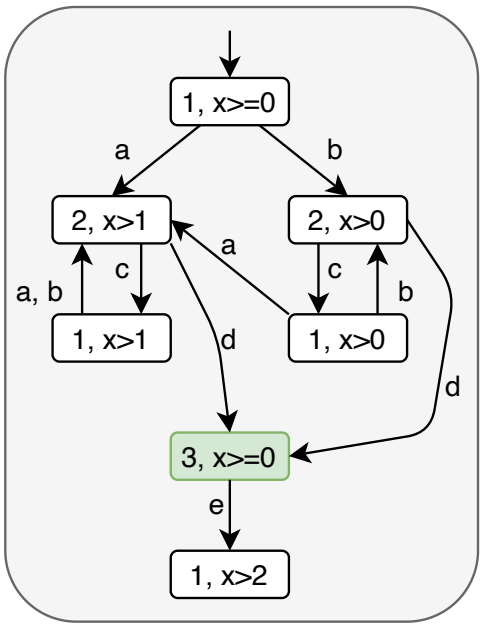
Subsumption graph 2 ×



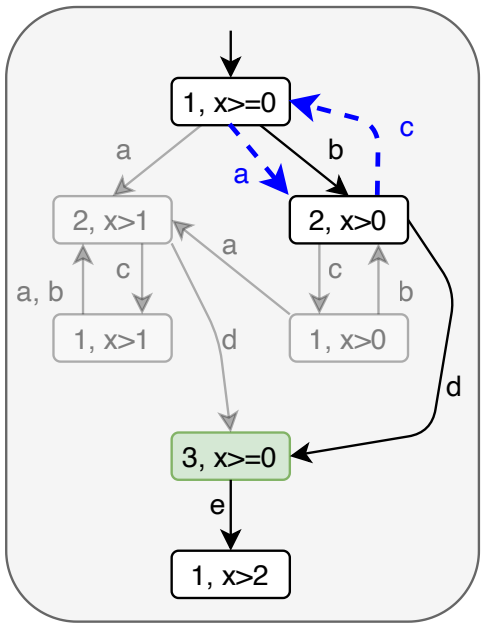
- A subsumption graph with **no accepting cycle** is a liveness **certificate**
- **Not all** subsumptions graphs are liveness certificates

Liveness compatible subsumption graphs

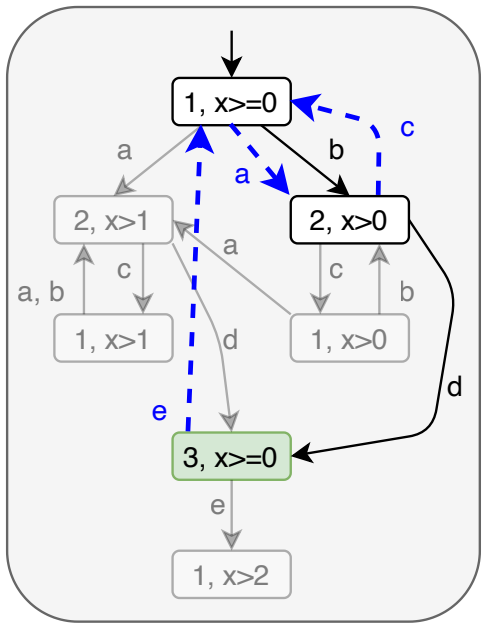
Zone graph $\mathcal{ZG}$ ✓



Subsumption graph 1 ✓



Subsumption graph 2 ×



- A subsumption graph is **liveness compatible** if it has no cycle with both ◯ and ⇢
- Two main algorithms for computing liveness compatible subsumption graphs: **nested-DFS** [LOD+13] and **SCC-decomposition based refinment algorithm** [HSTW16].

Reachability invariants (revisited)

Reachability invariants

Finite set I of nodes (q, Z) s.t.

- for any $(q, Z) \in I$ if $(q, Z) \to (q', Z')$ in $\mathcal{ZG}$, there is $(q', Z'') \in I$ s.t. $Z' \subseteq Z''$
- and the initial node (q_0, Z_0) of $\mathcal{ZG}$ is covered by some $(q_0, Z_0') \in I$, i.e. $Z_0 \subseteq Z_0'$

Let $(q, Z) \Rightarrow_I (q', Z')$ if $(q, Z) \to (q', Z'')$ and $Z'' \subseteq Z'$

If there is a run of $\mathcal{A}$: $(q_0, v_0) \to (q_1, v_1) \to \cdots (q_n, v_n) \to \cdots$
then, there is a path: $(q_0, Z_0) \to (q_1, Z_1) \to \cdots (q_n, Z_n) \to \cdots$ in $\mathcal{ZG}$
then, there is a path: $(q_0, Z_0') \Rightarrow_I (q_1, Z_1') \Rightarrow_I \cdots (q_n, Z_n') \Rightarrow_I \cdots$ in I

Certifying liveness of timed automata

If $\mathcal{A}$ has an **accepting run**: $(q_0, v_0) \to \cdots (\bigcirc, v_f) \to \cdots (\bigcirc, v'_f) \to \cdots$
then $\mathcal{ZG}$ has **accepting path**: $(q_0, Z_0) \to \cdots (\bigcirc, Z_f) \to \cdots (\bigcirc, Z_f) \to \cdots$
then I has **accepting path**: $(q_0, Z'_0) \Rightarrow_I \cdots (\bigcirc, Z'_f) \Rightarrow_I \cdots (\bigcirc, Z'_f) \Rightarrow_I \cdots$

I is a **certificate of liveness** if it contains no lasso with $\bigcirc$

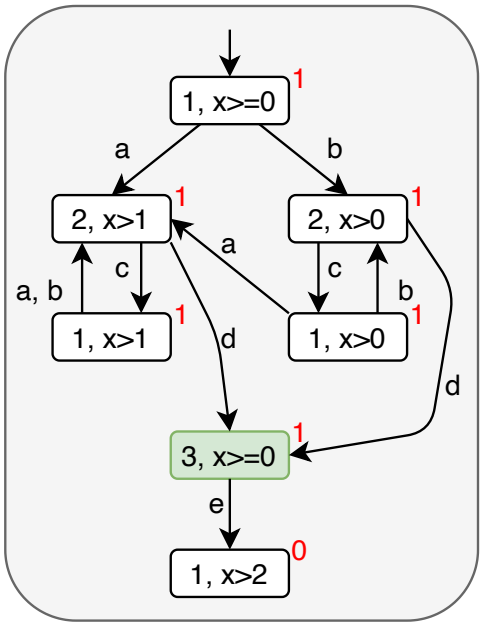
How to check absence of accepting lasso?

- use **topological numberings** $n : I \to \mathbb{N}$
  - if $(\bigcirc, Z) \Rightarrow_I (q', Z')$ then $n(\bigcirc, Z) > n(q', Z')$
  - otherwise $(q, Z) \Rightarrow_I (q', Z')$ then $n(q, Z) \geq n(q', Z')$

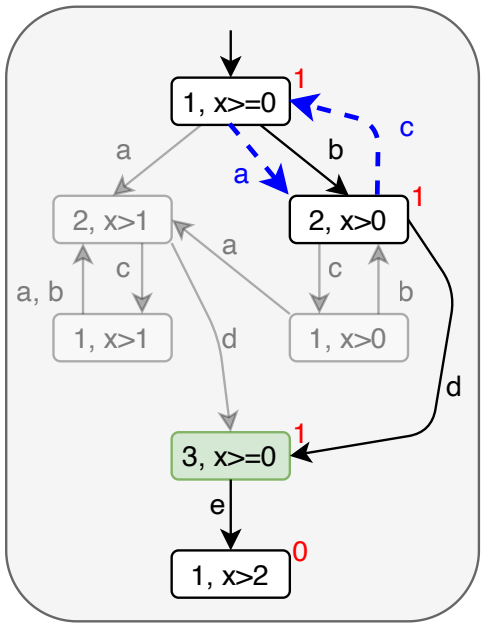
If n is a topological numbering of I, then I contains no cycle on an accepting state, hence no accepting lasso

Topological numberings

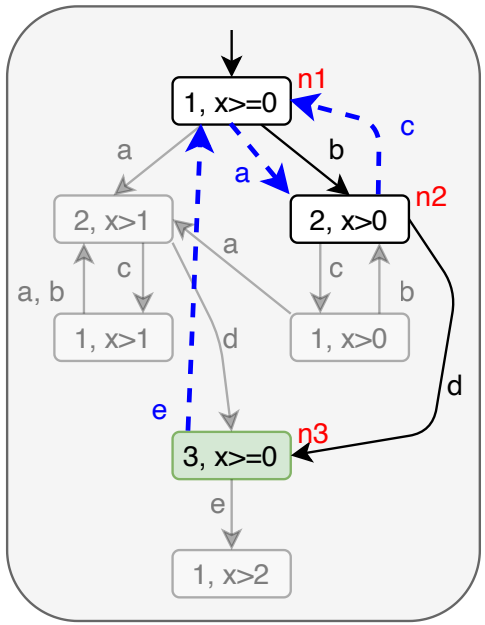
Zone graph $\mathcal{ZG}$ ✓



Subsumption graph 1 ✓



Subsumption graph 2 ×



$n_1 \geq n_2 \geq n_3 > n_1$
unsatisfiable

Topological numberings $n : l \to \mathbb{N}$

- if $(\bigcirc, Z) \Rightarrow_l (q', Z')$ then $n(\bigcirc, Z) > n(q', Z')$
- otherwise $(q, Z) \Rightarrow_l (q', Z')$ then $n(q, Z) \geq n(q', Z')$

Certification algorithm (liveness)

```
1  procedure certify_liveness(A, C, n):
2    initial_state_covered := false;
3    for each (q, Z, n) ∈ C:
4      if Z = ∅ or Z is not canonical:
5        reject C
6      if q is accepting:
7        reject C
8      if q = q0 and {0} ⊆ Z:
9        initial_state_covered := true;
10     for each (q, Z) → (q', Z'):
11       if ∄(q', Z'', n'') ∈ C s.t. Z' ⊆ Z'' and n ≥ n'' and (q = q' ⟹ n > n''):
12         reject C
13   if initial_state_covered:
14     accept C
15   else
16     reject C
```

- Check if C is a **reachability invariant** and if n is a **topological numbering**
- **Efficient**, can easily be **parallelized**
- Formalization in ITP: **zones**, **successor computation** and **simulation theorems**
- Agnostic to **extrapolation** and **liveness algorithm**

Simulation based abstractions

Simulation and reachability invariants

- $\preccurlyeq$ is a **simulation relation** if:

$$\begin{array}{ccccc} \forall & (q_1, v_1) & \preccurlyeq & (q_2, v_2) & \\ & \downarrow & & \downarrow & \\ & (q_1', v_1') & \preccurlyeq & (q_2', v_2') & \exists \end{array}$$

- **Extension to zones:** $Z \preccurlyeq Z'$ if for each $v \in Z$ there exists $v' \in Z'$ such that $v \preccurlyeq v'$

Reachability invariants under simulation $\preccurlyeq$

Finite set I of nodes (q, Z) s.t.

- for any $(q, Z) \in I$ if $(q, Z) \to (q', Z')$ in $\mathcal{ZG}$, there is $(q', Z'') \in I$ s.t. $Z' \preccurlyeq Z''$
- and the initial node (q_0, Z_0) of $\mathcal{ZG}$ is covered by some $(q_0, Z_0') \in I$, i.e. $Z_0 \preccurlyeq Z_0'$

Properties of reachability invariants under simulation

Reachability invariants under simulation $\preccurlyeq$

Finite set I of nodes (q, Z) s.t.

- for any $(q, Z) \in I$ if $(q, Z) \to (q', Z')$ in $\mathcal{ZG}$, there is $(q', Z'') \in I$ s.t. $Z' \preccurlyeq Z''$
- and the initial node (q_0, Z_0) of $\mathcal{ZG}$ is covered by some $(q_0, Z_0') \in I$, i.e. $Z_0 \preccurlyeq Z_0'$

Let $(q, Z) \Rightarrow_{I,\preccurlyeq} (q', Z')$ if $(q, Z) \to (q', Z'')$ and $Z'' \preccurlyeq Z'$

If there is a run of $\mathcal{A}$: $(q_0, v_0) \to (q_1, v_1) \to \cdots (q_n, v_n) \to \cdots$
then, there is a path: $(q_0, Z_0) \to (q_1, Z_1) \to \cdots (q_n, Z_n) \to \cdots$ in $\mathcal{ZG}$
then, there is a path: $(q_0, Z_0') \Rightarrow_{I,\preccurlyeq} (q_1, Z_1') \Rightarrow_{I,\preccurlyeq} \cdots (q_n, Z_n') \Rightarrow_{I,\preccurlyeq} \cdots$ in I

Certification with simulation

```
1  procedure certify_reachability_sim(A, C, ≼):
2    initial_state_covered := false;
3    for each (q, Z) ∈ C:
4      if Z = ∅ or Z is not canonical:
5        reject C
6      if q is accepting:
7        reject C
8      if q = q0 and {0}≼Z:
9        initial_state_covered := true;
10     for each (q, Z) → (q', Z'):
11       if ∄(q', Z'') ∈ C s.t. Z'≼Z'':
12         reject C
13   if initial_state_covered:
14     accept C
15   else
16     reject C
```

- Check if C is a **reachability invariant under simulation** $\preccurlyeq$
- Formalization in ITP: **zones**, **successor computation** and **simulation theorems**, simulation $\preccurlyeq$
- Works for liveness certification as well
- Implemented for $\preccurlyeq_{LU}$ [BBLP06]

Conclusions and perspectives

Experimental results

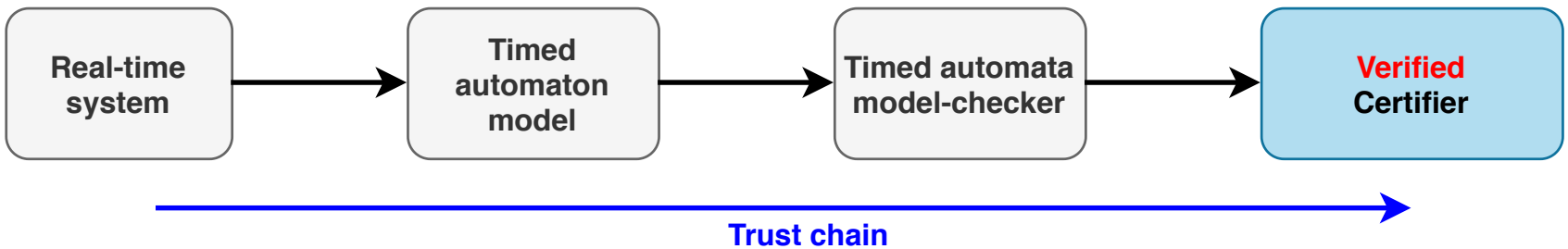
Model	TChecker						Imitator			
	Iterative SCC			NDFS			Merge			NDFS
CC1	✓	57	0.01	✓	3281	0.06	✓	58	0.01	***
CC4	✓	195858	221.56	✓	32575	7.75		***		***
CC5	✓	65639	30.63	✓	143057	218.98		***		***
FD1	✓	214	0.02	✓	677	0.03	✗	294	0.02	✓ 1518 0.11
FI1	✓	65	0.01	✓	71	0.00	✓	136	0.00	***
FI2	✓	314	0.01	✓	344	0.01	✓	589	0.01	***
FI4	✓	204	0.00	✓	224	0.01	✓	793	0.01	***
FI5	✓	3091	0.13	✓	2392	0.09	✗	863	0.03	***

*** timeout 300 sec. (termination not guaranteed with Imitator)

- MUNTA tool: `https://github.com/wimmers/munta`
- Three algorithms: Iterative SCC [HSTW16], nested-DFS with subsumption [LOD+13] and reachability with mergeing [AS11]

Summary

- **Certification of state-of-the-art liveness checking algorithms** for Timed Büchi automata
- MUNTA certifier was fully verified in Isabelle/HOL and still yields reasonable performance → **Trust reduction**



- Independent from extrapolation, and it includes certified **simulation-based abstractions**
- Formalization in terms of: transition systems and simulations → can be **generalized** to other models

Perspectives

- **LTL model-checking**: combine with certifier for LTL $\rightarrow$ NBA construction [ELN+13, BSS19]
- Certify improved Iterative-SCC algorithm [HSTW20] that builds safe **subsumption graphs that are not liveness compatible but still correct**
- Certification of **dynamic abstractions** [HSW13, GMS19]
- Generalize the refinment approch in [HSTW16] to design **new verification algorithms**
- Build a **certifier companion tool** for TChecker

Étienne André and Romain Soulat.

Synthesis of timing parameters satisfying safety properties.
In Giorgio Delzanno and Igor Potapov, editors, *Reachability Problems - 5th International Workshop, RP 2011, Genoa, Italy, September 28-30, 2011. Proceedings*, volume 6945 of *Lecture Notes in Computer Science*, pages 31–44. Springer, 2011.

Gerd Behrmann, Patricia Bouyer, Kim Guldstrand Larsen, and Radek Pelánek.

Lower and upper bounds in zone-based abstractions of timed automata.
Int. J. Softw. Tools Technol. Transf., 8(3):204–215, 2006.

Julian Brunner, Benedikt Seidl, and Salomon Sickert.

A verified and compositional translation of LTL to deterministic rabin automata.
In John Harrison, John O'Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving, ITP 2019, September 9-12, 2019, Portland, OR, USA*, volume 141 of *LIPIcs*, pages 11:1–11:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

Javier Esparza, Peter Lammich, René Neumann, Tobias Nipkow, Alexander Schimpf, and Jan-Georg Smaus.

A fully verified executable LTL model checker.
In *Proc. of CAV'13*, volume 8044 of *LNCS*, pages 463–478. Springer, 2013.

Paul Gastin, Sayan Mukherjee, and B. Srivathsan.

Fast algorithms for handling diagonal constraints in timed automata.
In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I*, volume 11561 of *Lecture Notes in Computer Science*, pages 41–59. Springer, 2019.

Marijn Heule, Warren Hunt, Matt Kaufmann, and Nathan Wetzler.

Efficient, verified checking of propositional proofs.
In Mauricio Ayala-Rincón and César A. Muñoz, editors, *Interactive Theorem Proving*, pages 269–284, Cham, 2017. Springer International Publishing.

Frédéric Herbreteau, B. Srivathsan, Thanh-Tung Tran, and Igor Walukiewicz.

Why liveness for timed automata is hard, and what we can do about it.
In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, December 13-15,*

2016, Chennai, India, volume 65 of *LIPIcs*, pages 48:1–48:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

Frédéric Herbreteau, B. Srivathsan, Thanh-Tung Tran, and Igor Walukiewicz.

Why liveness for timed automata is hard, and what we can do about it.
ACM Trans. Comput. Log., 21(3):17:1–17:28, 2020.

Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz.

Lazy abstractions for timed automata.
In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 990–1005. Springer, 2013.

Peter Lammich.

Efficient verified (UN)SAT certificate checking.
In Leonardo de Moura, editor, *Automated Deduction – CADE 26*, pages 237–254, Cham, 2017. Springer International Publishing.

Xavier Leroy, Sandrine Blazy, Daniel Kästner, Bernhard Schommer, Markus Pister, and Christian Ferdinand.

Compcert – a formally verified optimizing compiler.
In *ERTS 2016: Embedded Real Time Software and Systems*. SEE, 2016.

Alfons Laarman, Mads Chr. Olesen, Andreas Engelbredt Dalsgaard, Kim Guldstrand Larsen, and Jaco van de Pol.

Multi-core emptiness checking of timed büchi automata using inclusion abstraction.
In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 968–983. Springer, 2013.

Simon Wimmer and Peter Lammich.

Verified model checking of timed automata.
In Dirk Beyer and Marieke Huisman, editors, *TACAS 2018*, pages 61–78, Cham, 2018. Springer.

Simon Wimmer and Joshua von Mutius.

Verified certification of reachability checking for timed automata.

In Armin Biere and David Parker, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I*, volume 12078 of *Lecture Notes in Computer Science*, pages 425–443. Springer, 2020.