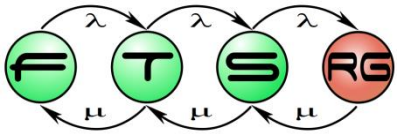
An introduction to the verification of timed automata

Rebeka Farkas

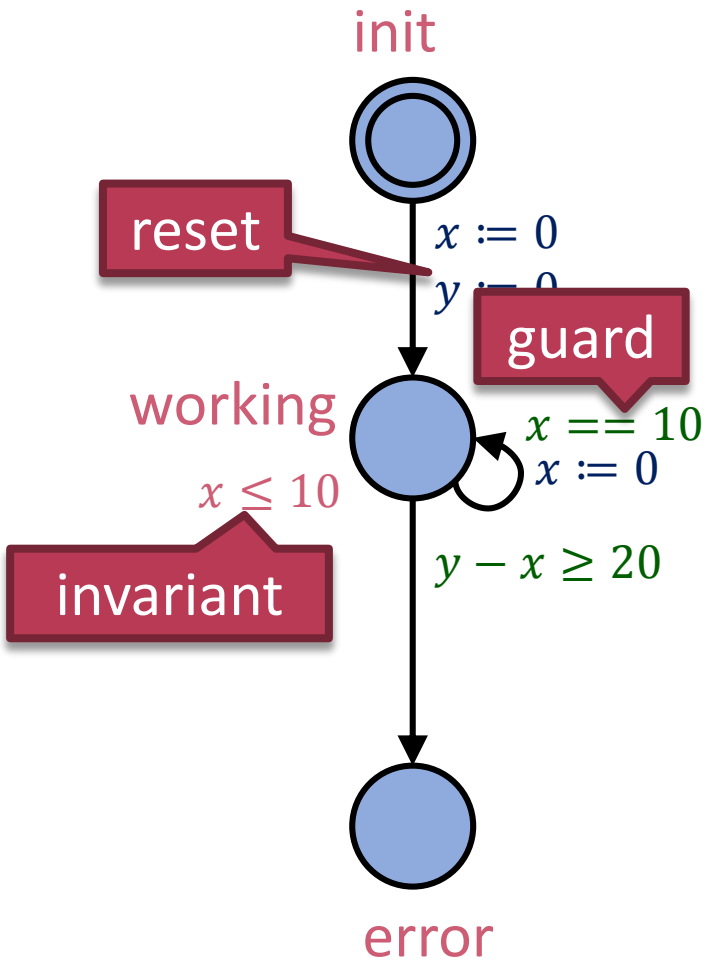
farkasr@mit.bme.hu

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group

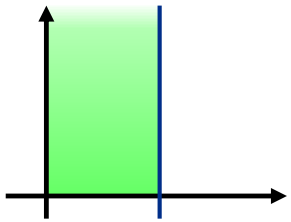
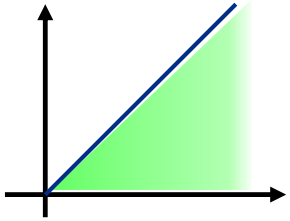


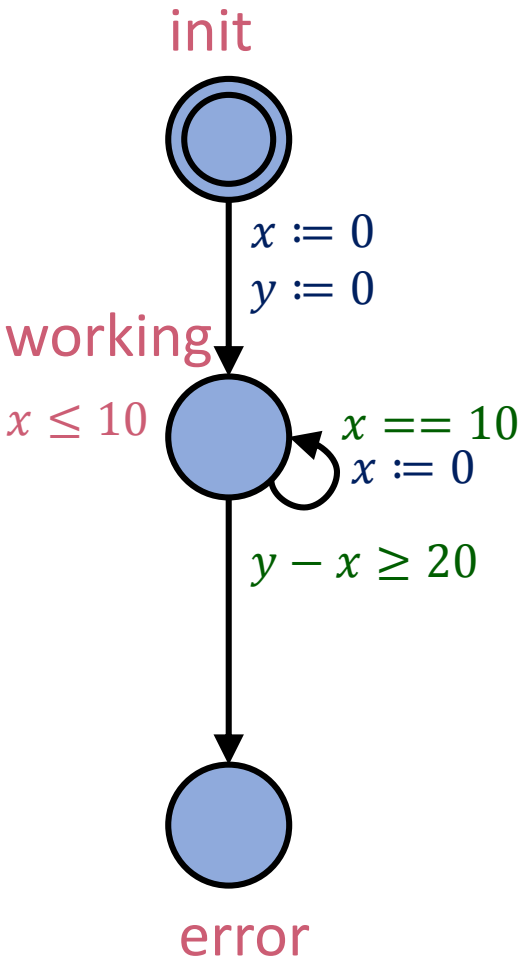
Timed Automaton

- Finite automaton
- Clock variables
  - Time elapses in a location
  - Transition - reset, guard
  - Locations - invariant
- Clock constraints
  - Used in guards and invariants
  - Two types ($x, y \in \mathbb{C}, n \in \mathbb{N}$):
    - Simple constraint: $x \bowtie n$
    - Diagonal constraint: $x - y \bowtie n$
  - + conjunction



Clock constraints

	Simple constraint	Diagonal constraint
Form	$x \leqq n$	$x - y \leqq n$
Shape	 Perpendicular	 Diagonal
Value	Time-dependent	Doesn't change over time
Usage	Guards, invariants	Abstract state space



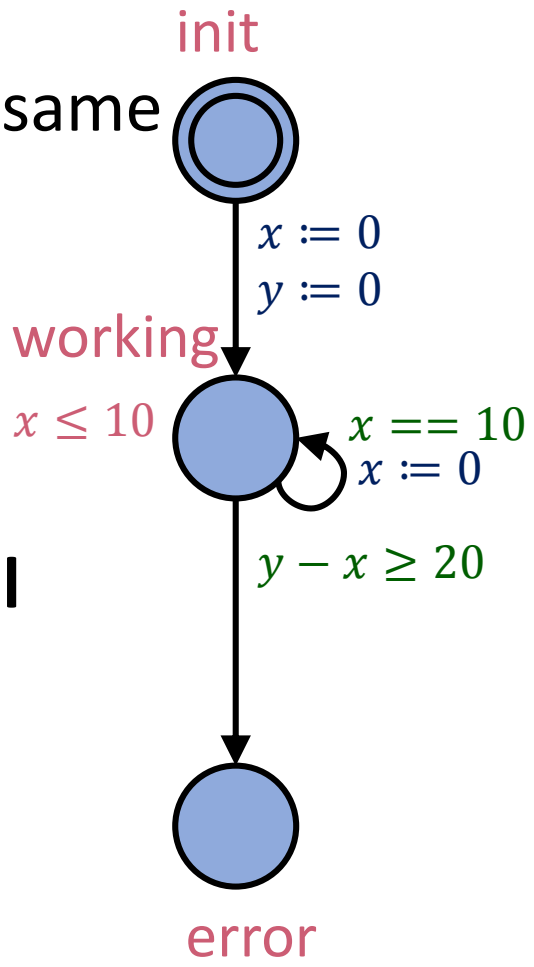
TA definition varies: *diagonal-free* TA

Diagonal-free timed automaton

- DTA constraints: $x \leqq n$ only
  - Expressive power of TA and DTA is the same
- Diagonal constraints:
  - Transformed automaton is **exponential**

$$\mathcal{O}(|\mathcal{L}| * 2^d)$$

  - Difficult to handle in algorithms



ALGORITHMS

Model checking basics

- Input: $\mathcal{A}$ (timed automaton), φ (property to verify)

  Output: $\mathcal{A}$ satisfies φ
- Property:
  - Condition on the statespace
  - Types:
    - Reachability
    - Liveness (e.g. deadlock)
    - Temporal logic
- Methods:
  - Symbolic approach – use a SAT (SMT) solver
  - Statespace exploration – requires efficient abstract domain
  - …and more

Symbolic methods

- General idea: System → Logical constraints
  - SAT – reachable
  - UNSAT – unreachable
  - Applicable for any formalism
- Timed automata
  - Possible[1] – difference logic
  - Inefficient

  →New formalisms (e.g. timeout automaton)

1. Morbé, Georges, Florian Pigorsch, and Christoph Scholl. "Fully Symbolic Model Checking for Timed Automata." *CAV*. Vol. 11. 2011.

Statespace exploration [2]

- Continuous domain → explicit statespace exploration is impossible
- Abstract domain – exact abstraction
- State of TA: location + valuation (values of clocks)
  - Locations can be handled explicitly
  - Valuations: *ZONE*
    - described by **clock constraints**
    - stored in a simple matrix form (Difference Bound Matrix)
    - operations defined (forward and backward exploration)
    - safe abstraction for reachability, and temopral properties

2. J. Bengtsson and W. Yi, “Timed automata: Semantics, algorithms and tools,” in Lectures on Concurrency and Petri Nets, ser. LNCS. Springer Berlin Heidelberg, 2004, vol. 3098, pp. 87–124.

Difference Bound Matrix

- Atomic difference constraints: $x - y \lesssim n$
  - $x - y \gtrsim n$ → $y - x \lesssim -n$
  - $x - y = n$ → $x - y \leq n \wedge x - y \geq n$ → …
  - $x \lessgtr n$ → $x - \mathbf{0} \lessgtr n$ → …

Special clock variable

- Difference Bound Matrix
  - Rows/cols: clock variables (including $\mathbf{0}$)
  - Element: upper bound (+strictness) on the difference
  - Example: $\mathrm{D} = \{x < 5, y - x < 4\}$

$$M(D) = \begin{array}{c} \\ \mathbf{0} \\ x \\ y \end{array} \begin{array}{c} \begin{array}{ccc} \mathbf{0} & x & y \end{array} \\ \begin{pmatrix} 0, \leq & -5, < & 0, \leq \\ \infty & 0, \leq & \infty \\ 9, < & 4, < & 0, \leq \end{pmatrix} \end{array}$$

Non-negativeness

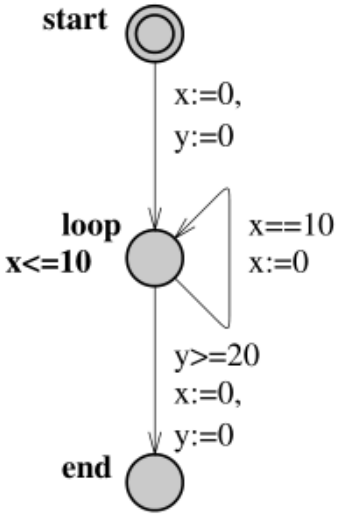
Implied by D

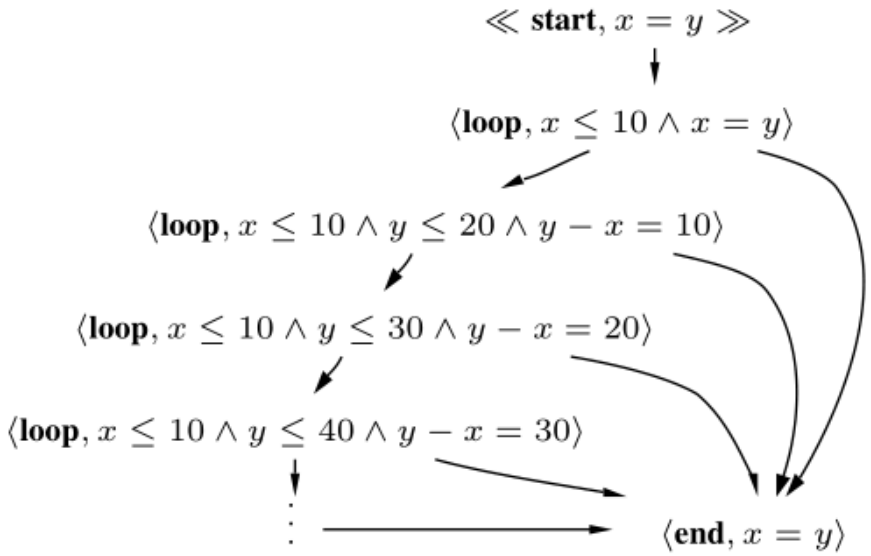
Zone operations

- Operations for statespace-exploration
  - Up() → represents the elapse of time
    - Down() →inverse operation (for backwards exploration)
  - And($x - y \lesssim n$) → adds a constraints (e.g. guard)
  - Sat() → tells if it is consistent
  - Reset(x) → resets a clock variable
    - Free(x) →inverse operation (for backwards exploration)
  - Contains(D) → inclusion

Statespace exploration of TA I.

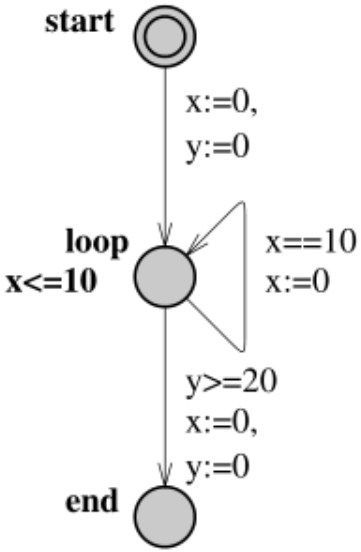
- Algorithm (idea)
  - State space representation: zone graph - finite
    - Node: location + zone
    - Edge: transition
  - Reachability analysis: exploring the zone graph
- Problem: may not terminate

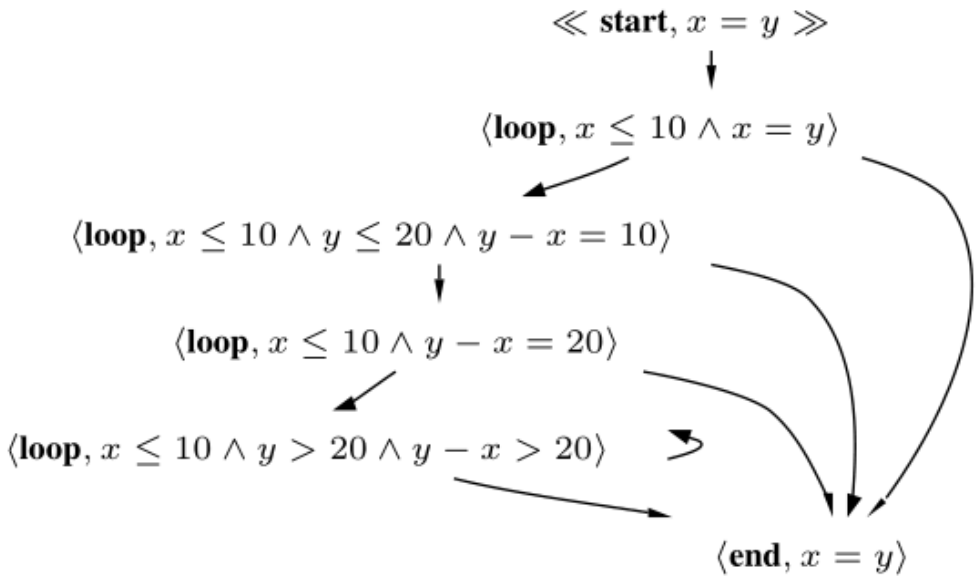




Statespace exploration of TA II.

- Problem: algorithm may not terminate
- Solution: extrapolation
  - Idea:
    - *K(x)* - biggest constant to which clock *x* is compared
    - *x > K(x)* can be represented with the same abstract value

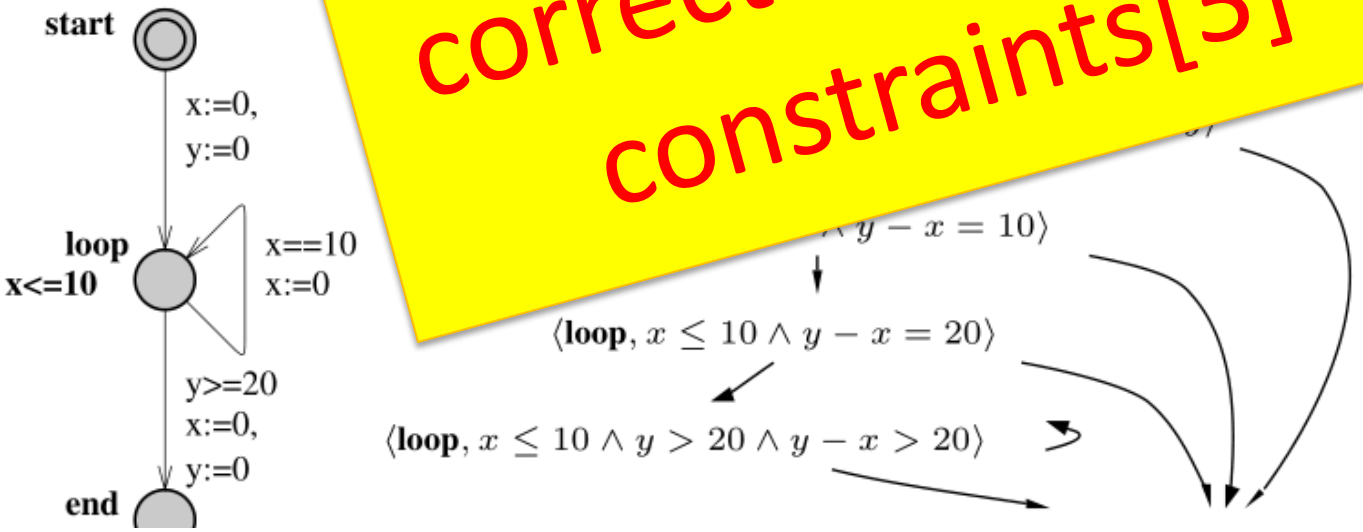




Statespace exploration of TA II.

- Problem: algorithm may not terminate
- Solution: extrapolation
  - Idea:
    - *K(x)* ... pared
    - *x* ... ract value

This algorithm is not correct for diagonal constraints[3]



3. Bouyer, Patricia. "Untameable timed automata!." *STACS*. Vol. 2607. 2003.

Handling diagonal constraints

1. Only consider DTA
   - Encode them in locations
   - New, efficient abstractions, algorithms
     - K(x) →L(x), U(x)
     - Lazy evaluation – CEGAR-based method
     - optimal abstraction for reachability [4]

4. Herbreteau, Frédéric, B. Srivathsan, and Igor Walukiewicz. "Lazy abstractions for timed automata." *International Conference on Computer Aided Verification*. Springer, Berlin, Heidelberg, 2013.

Handling diagonal constraints

1. Only consider DTA
2. Operation *split* [5]
   - Before extrapolation: split zone along diagonal constraints
   - Extrapolate subzones individually
   - Reaplly the original constraints (applied by split)
   - Introduce new nodes to the zone graph for each subzones
   - → Exponential in the number of diagonal constraints

5. Bengtsson, Johan, and Wang Yi. "On Clock Difference Constraints and Termination in Reachability Analysis of Timed Automata." *Formal Methods and Software Engineering* (2003): 491-503.

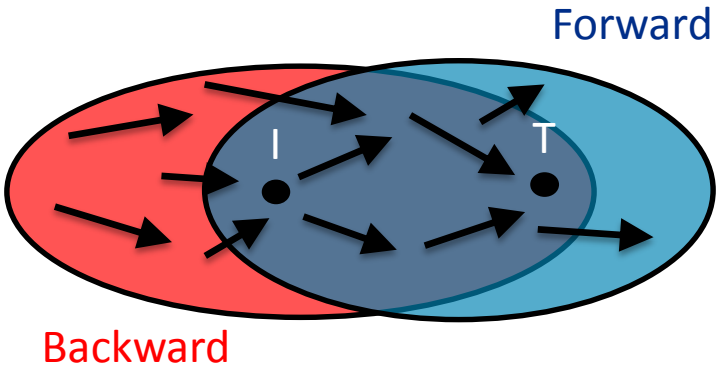
Handling diagonal constraints

1. Only consider DTA
2. Operation *split*
3. Use backward exploration

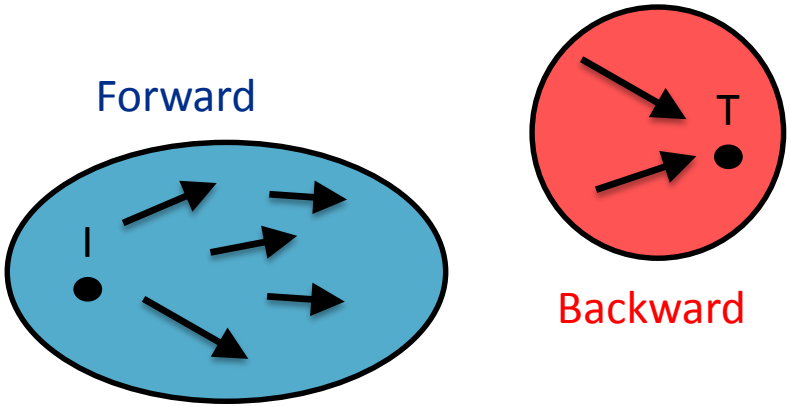
Backward Exploration

	Forward	Backward
Explored states	Reachable from initial state →Target state?	Target state is reachable → Initial state
Advantages	Builds an abstract state space *Explored statespace might be smaller if reachable*	*Explored statespace might be smaller if unreachable* No exponential factor for diagonal timed automata

Reachable



Unreachable



Handling diagonal constraints

1. Only consider DTA
2. Operation *split*
3. Use backward exploration
   - No need for extrapolation → no additional exponential factor
   - Possible: zone operations exist
   - Data variables can't be calculated on the fly
     - Ex: integers a,b,c; state: a=5, b=10, c=15
     - Incoming transition: a=b+c, b=10, c=15 → source state: b+c=5 (???)

My research

Handling diagonal constraints

1. Only consider DTA
2. Operation *split*
3. Use backward exploration
4. CEGAR-based methods [6]
   - Check with the (overapproximating) original algorithm
   - Reachable → check trace (e.g. symbolic method)
   - Invalid → Refine
     - …the automaton based on the diagonal constraints
     - …the statespace (eg. backwards exploration)

6. Bouyer, Patricia, François Laroussinie, and Pierre-Alain Reynier. "Diagonal constraints in timed automata: Forward analysis of timed systems." *FORMATS*. Vol. 5. 2005.

TOOLS

Tools by purpose:

- The Uppaal family:
  - XTA (eXtended with data variables) →Uppaal
  - PTA (probabilistic) → Uppaal Pro
  - Priced TA → Uppaal Cora
  - …
- Similar tools (less formalisms, less efficient): Kronos, RED, …
- Tools for more expressive formalisms (less efficient):
  - PISM – probabilistic models
  - HyTec – linear hybrid automata
  - …

ϴ (Theta) model checker ☺

Which tool to use?

- Simple anwer: Uppaal
- 25 years of improvements and optimization

Same algorithm

Model	nb. of clocks	UPPAAL (-C) nodes	UPPAAL (-C) sec.	$Extra^{+}_{LU}$,sa nodes	$Extra^{+}_{LU}$,sa sec.
$\mathcal{D}''_7$	14	18654	11.60	18654	8.08
$\mathcal{D}''_8$	16				
$\mathcal{D}''_{70}$	140				
CSMA/CD 10	11	120845	1.92	120844	6.26
CSMA/CD 11	12	311310	5.44	311309	16.78
CSMA/CD 12	13	786447	14.77	786446	44.03
FDDI 50	151	12605	52.98	12606	29.41
FDDI 70	211				
FDDI 140	421				
Fischer 9	9	135485	2.45	135485	8.93
Fischer 10	10	447598	10.10	447598	34.02
Fischer 11	11	1464971	40.44	1464971	126.78
Stari 3	10	21897	0.52	6454	0.66
Stari 4	13	193662	7.54	44330	5.60
Stari 5	16	2024821	130.87	359570	57.84

Herbreteau, Frédéric, B. Srivathsan, and Igor Walukiewicz. "Lazy abstractions for timed automata." *International Conference on Computer Aided Verification*. Springer, Berlin, Heidelberg, 2013.

What we've seen today

- Timed automata
- Diagonal constraints
- Symbolic methods
- Zone-based algorithm
- Backwards exploration
- CEGAR-based methods
- Tools → Uppaal