# Lecture 6. Timed automata. Timed temporal logics

## ELEC-E8110 Automation Systems Synthesis and Analysis

Igor Buzhinsky

igor.buzhinskii@aalto.fi

Aalto University
School of Electrical
Engineering

2018

# Timed automata

# Logical time

- E.g. infinite trace $\{p\}, \{q\}, \{p, q\}, \ldots$
- Each step can be interpreted as a time unit
- What if we wish to express properties related to physical time?

# Logical time

- E.g. infinite trace $\{p\}, \{q\}, \{p, q\}, ...$
- Each step can be interpreted as a time unit
- What if we wish to express properties related to physical time?
- Assume that each transition has a fixed duration?

# Logical time

- E.g. infinite trace $\{p\}, \{q\}, \{p, q\}, \ldots$
- Each step can be interpreted as a time unit
- What if we wish to express properties related to physical time?
- Assume that each transition has a fixed duration?
- Do you see potential problems?

# Logical time

- E.g. infinite trace $\{p\}, \{q\}, \{p, q\}, ...$
- Each step can be interpreted as a time unit
- What if we wish to express properties related to physical time?
- Assume that each transition has a fixed duration?
- Do you see potential problems?
- Assume that all transitions have the same duration $d$. There can be quick events in the system, e.g. a transition which lasts 10ms, so we must require $d$ to be at least 10ms. How to state that $p$ happens after 10 seconds?

# Logical time

- E.g. infinite trace $\{p\}, \{q\}, \{p, q\}, \ldots$
- Each step can be interpreted as a time unit
- What if we wish to express properties related to physical time?
- Assume that each transition has a fixed duration?
- Do you see potential problems?
- Assume that all transitions have the same duration $d$. There can be quick events in the system, e.g. a transition which lasts 10ms, so we must require $d$ to be at least 10ms. How to state that $p$ happens after 10 seconds? $\mathbf{X} \ldots \mathbf{X}\, p = \mathbf{X}^{1000}\, p$. While CTL model checking is linear of the formula length, LTL model checking is exponential!

# Logical time

- E.g. infinite trace $\{p\}, \{q\}, \{p, q\}, \ldots$
- Each step can be interpreted as a time unit
- What if we wish to express properties related to physical time?
- Assume that each transition has a fixed duration?
- Do you see potential problems?
- Assume that all transitions have the same duration $d$. There can be quick events in the system, e.g. a transition which lasts 10ms, so we must require $d$ to be at least 10ms. How to state that $p$ happens after 10 seconds? $\mathbf{X} \ldots \mathbf{X}\, p = \mathbf{X}^{1000}\, p$. While CTL model checking is linear of the formula length, LTL model checking is exponential!
- This can also bloat the state space

# Logical time

- E.g. infinite trace $\{p\}, \{q\}, \{p, q\}, \ldots$
- Each step can be interpreted as a time unit
- What if we wish to express properties related to physical time?
- Assume that each transition has a fixed duration?
- Do you see potential problems?
- Assume that all transitions have the same duration $d$. There can be quick events in the system, e.g. a transition which lasts 10ms, so we must require $d$ to be at least 10ms. How to state that $p$ happens after 10 seconds? $\mathbf{X} \ldots \mathbf{X}\, p = \mathbf{X}^{1000}\, p$. While CTL model checking is linear of the formula length, LTL model checking is exponential!
- This can also bloat the state space
- What about transitions which should be instantaneous?

# Logical time

- E.g. infinite trace $\{p\}, \{q\}, \{p, q\}, \ldots$
- Each step can be interpreted as a time unit
- What if we wish to express properties related to physical time?
- Assume that each transition has a fixed duration?
- Do you see potential problems?
- Assume that all transitions have the same duration $d$. There can be quick events in the system, e.g. a transition which lasts 10ms, so we must require $d$ to be at least 10ms. How to state that $p$ happens after 10 seconds? $\mathbf{X} \ldots \mathbf{X}\, p = \mathbf{X}^{1000}\, p$. While CTL model checking is linear of the formula length, LTL model checking is exponential!
- This can also bloat the state space
- What about transitions which should be instantaneous? – can merge them and disregard internal states...

# Timed behaviors

- **Pointwise semantics:** each transition between states has a **delay**
- Example of a **timed trace**: $\{p\} \xrightarrow{2s} \{q\} \xrightarrow{0.1s} \{p, q\}, \ldots$

# Timed behaviors

- **Pointwise semantics:** each transition between states has a **delay**
- Example of a **timed trace**: $\{p\} \xrightarrow{2s} \{q\} \xrightarrow{0.1s} \{p, q\}, \ldots$
- We will use this semantics for **timed automata**

# Timed behaviors

- **Pointwise semantics:** each transition between states has a **delay**
- Example of a **timed trace**: $\{p\} \xrightarrow{2s} \{q\} \xrightarrow{0.1s} \{p, q\}, \ldots$
- We will use this semantics for **timed automata**

- **Continuous semantics:** the system has a state at each continuous moment of time

# Timed behaviors

- **Pointwise semantics:** each transition between states has a **delay**
- Example of a **timed trace**: $\{p\} \xrightarrow{2s} \{q\} \xrightarrow{0.1s} \{p, q\}, \ldots$
- We will use this semantics for **timed automata**

- **Continuous semantics:** the system has a state at each continuous moment of time
- In particular, timed traces can be interpreted according to this semantics

# Timed behaviors

- **Pointwise semantics:** each transition between states has a **delay**
- Example of a **timed trace**: $\{p\} \xrightarrow{2s} \{q\} \xrightarrow{0.1s} \{p, q\}, \ldots$
- We will use this semantics for **timed automata**

- **Continuous semantics:** the system has a state at each continuous moment of time
- In particular, timed traces can be interpreted according to this semantics
- In **hybrid automata**, continuous state can evolve according to differential equations. Discretization is required to express this in pointwise semantics

# Timed automata: what's new

- Global **clock variables** which increase with the same rate

# Timed automata: what's new

- Global **clock variables** which increase with the same rate
- Transitions can be guarded by clock constraints and can reset clocks

# Timed automata: what's new

- Global **clock variables** which increase with the same rate
- Transitions can be guarded by clock constraints and can reset clocks
- **Invariants** in states must be satisfied

# Timed automata: what's new

- Global **clock variables** which increase with the same rate
- Transitions can be guarded by clock constraints and can reset clocks
- **Invariants** in states must be satisfied
- State = location (i.e. usual, graphically represented state) + clock values

# Timed automata: what's new

- Global **clock variables** which increase with the same rate
- Transitions can be guarded by clock constraints and can reset clocks
- **Invariants** in states must be satisfied
- State = location (i.e. usual, graphically represented state) + clock values

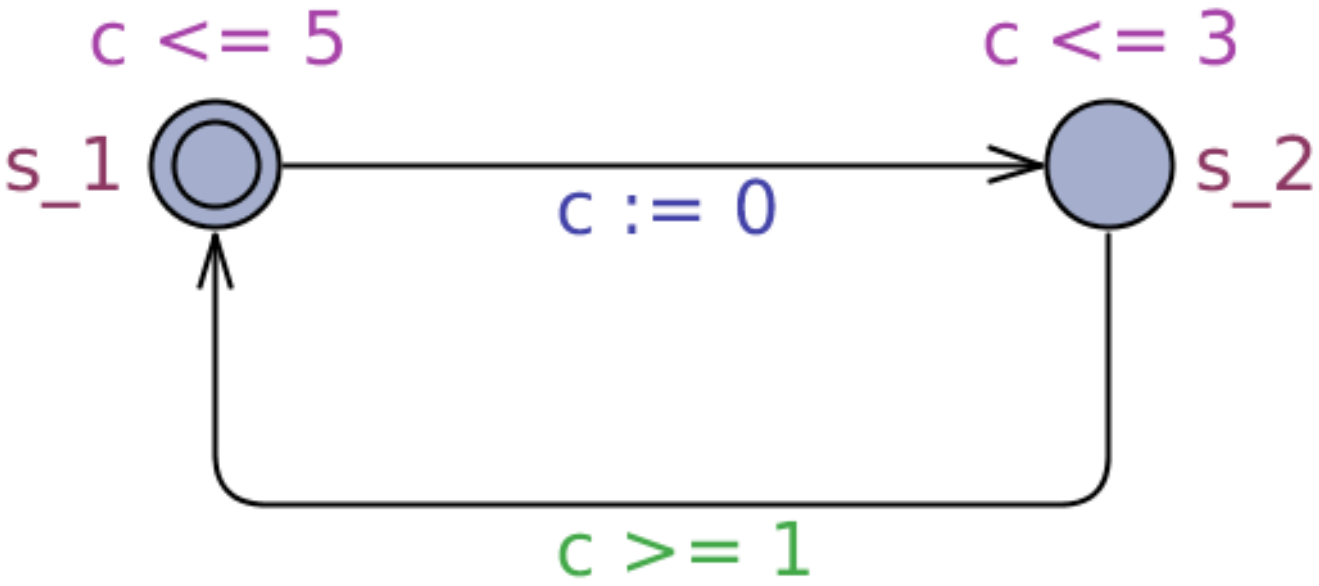
- Two types of transitions:
- Usual transitions ("edges") which adhere to guards – the location changes, clock values remain unchanged

# Timed automata: what's new

- Global **clock variables** which increase with the same rate
- Transitions can be guarded by clock constraints and can reset clocks
- **Invariants** in states must be satisfied
- State = location (i.e. usual, graphically represented state) + clock values

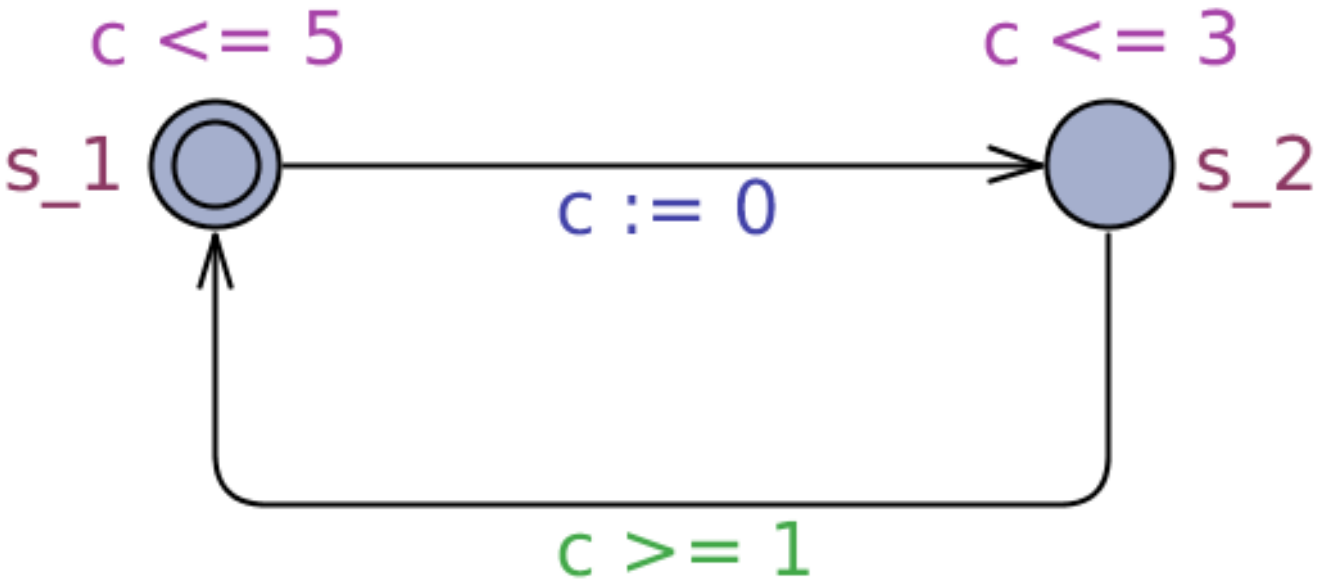
- Two types of transitions:
- Usual transitions ("edges") which adhere to guards – the location changes, clock values remain unchanged
- **Delay transitions** which adhere to invariants – the location does not change, clock values increment by the same value

# An example of a timed automaton



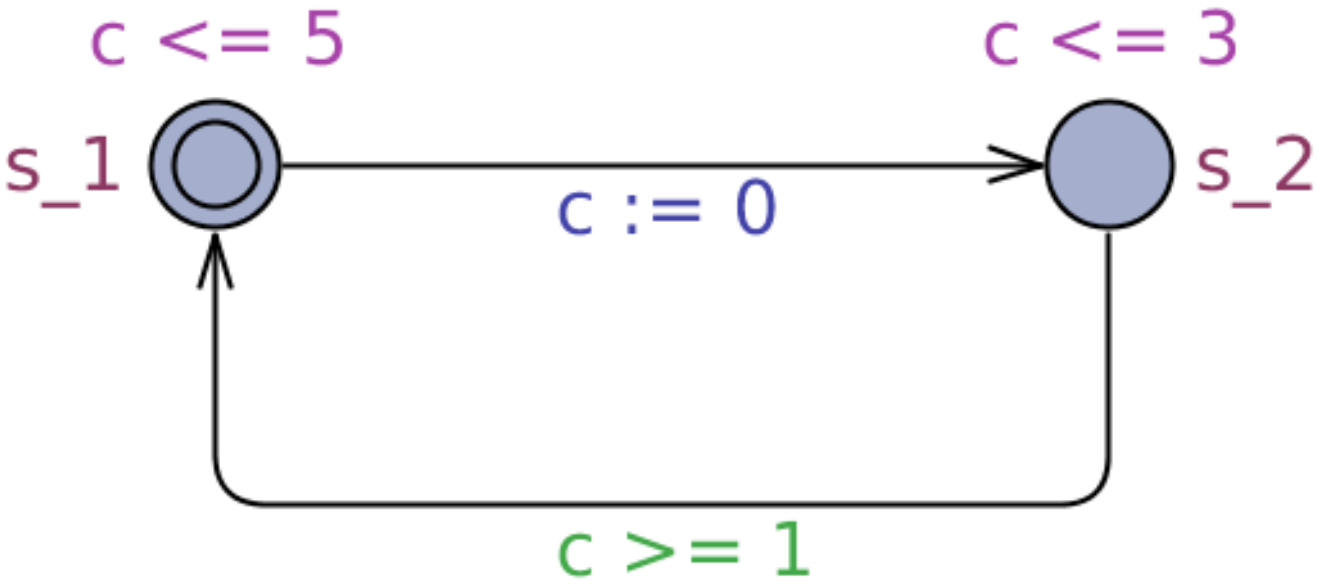

- State = location (i.e. s_1, s_2) + the value of clock c

# An example of a timed automaton



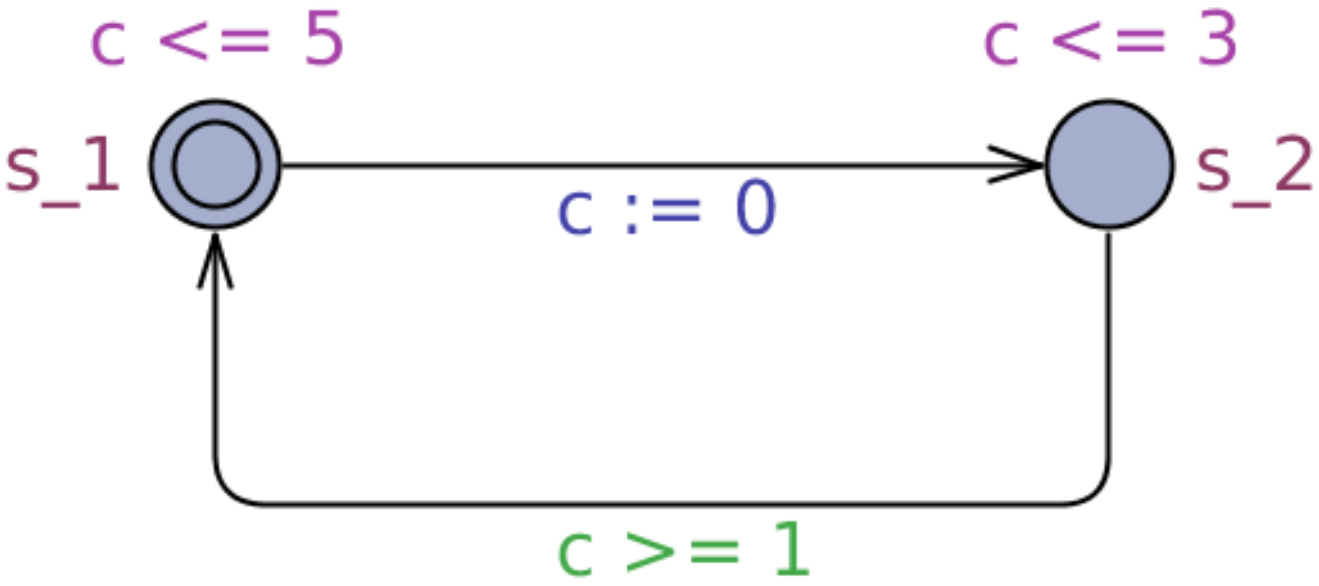

- State = location (i.e. s_1, s_2) + the value of clock c
- The upper transition resets the clock

# An example of a timed automaton



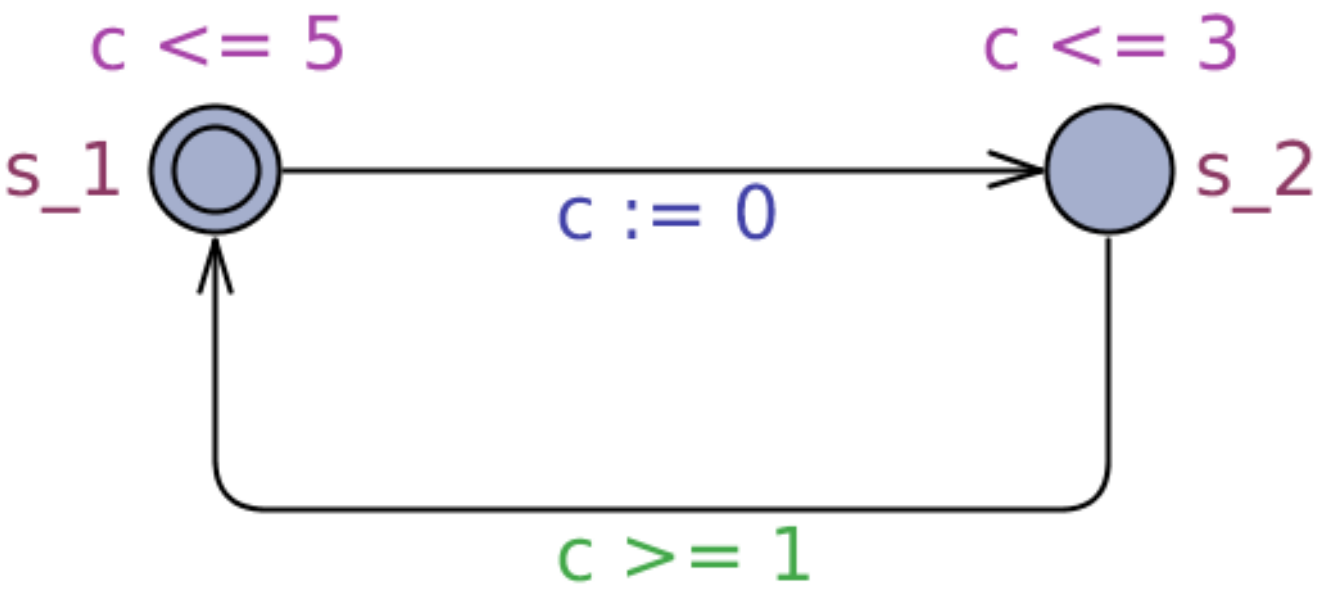

- State = location (i.e. s_1, s_2) + the value of clock c
- The upper transition resets the clock
- The lower transition is guarded by a clock

# An example of a timed automaton



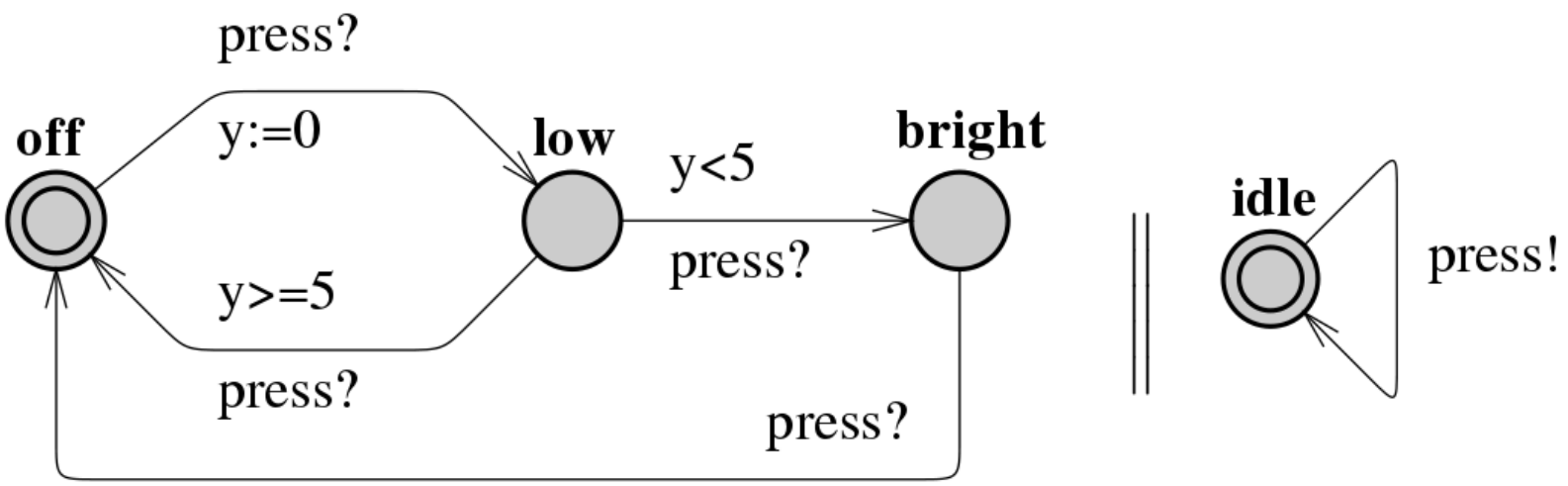

- State = location (i.e. s_1, s_2) + the value of clock c
- The upper transition resets the clock
- The lower transition is guarded by a clock
- Invariants are provided on top; they do not allow c to progress above the specified values in given locations

# An example of a timed automaton




- State = location (i.e. s_1, s_2) + the value of clock c
- The upper transition resets the clock
- The lower transition is guarded by a clock
- Invariants are provided on top; they do not allow c to progress above the specified values in given locations
- Can you think of examples of timed behaviors possible for this automaton?

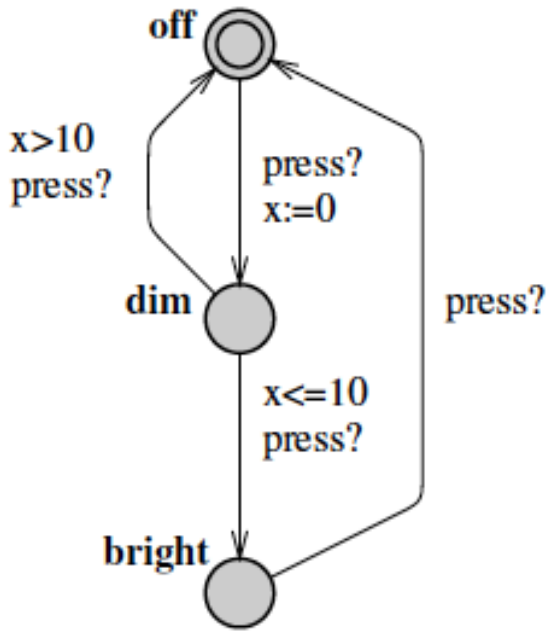
# Systems of timed automata can be modeled in UPPAAL

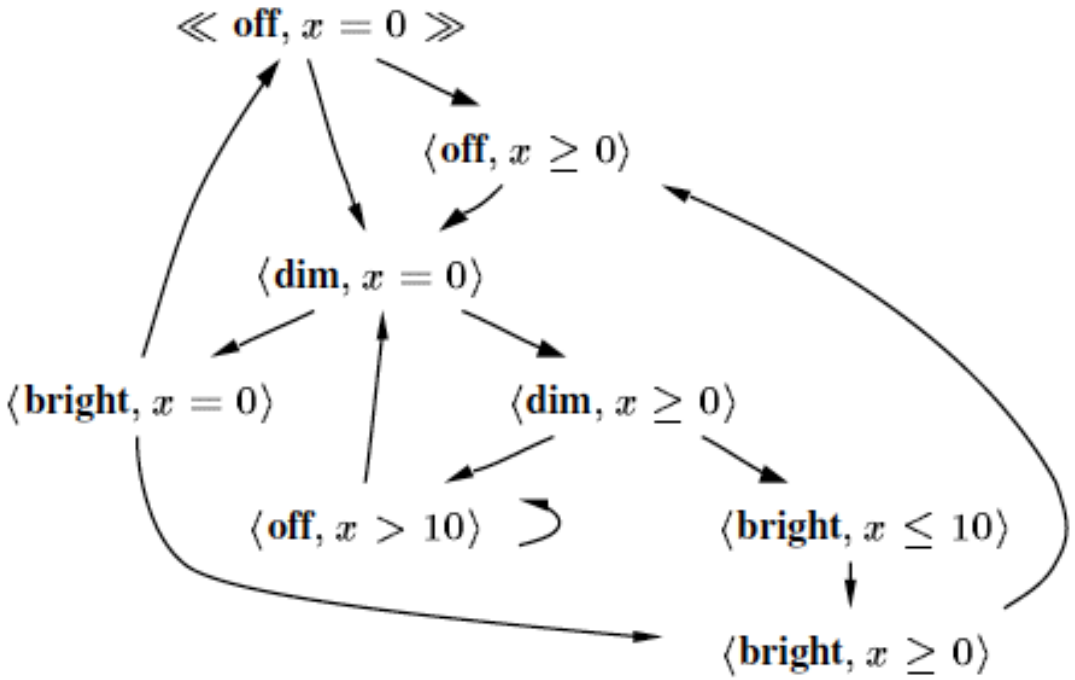



(a) Lamp. (b) User.

- Clock y
- The user can press the button, and the lamp will react
- If the lamp is off and the user presses the button two times within 5 time units, the lamp will become bright
- Otherwise, the light will first become low and then will switch off

# How the state space looks when clocks are presents




- Even though the time is continuous, it is processed using a finite number of intervals
- In the simulator of UPPAAL, the panel with variable values lists current intervals of all clocks

# More features of UPPAAL

- Urgent states: delay transitions are forbidden
- How to model such states with invariants?

# More features of UPPAAL

- Urgent states: delay transitions are forbidden
- How to model such states with invariants?

- Commited states is a more strict version of urgent states

# More features of UPPAAL

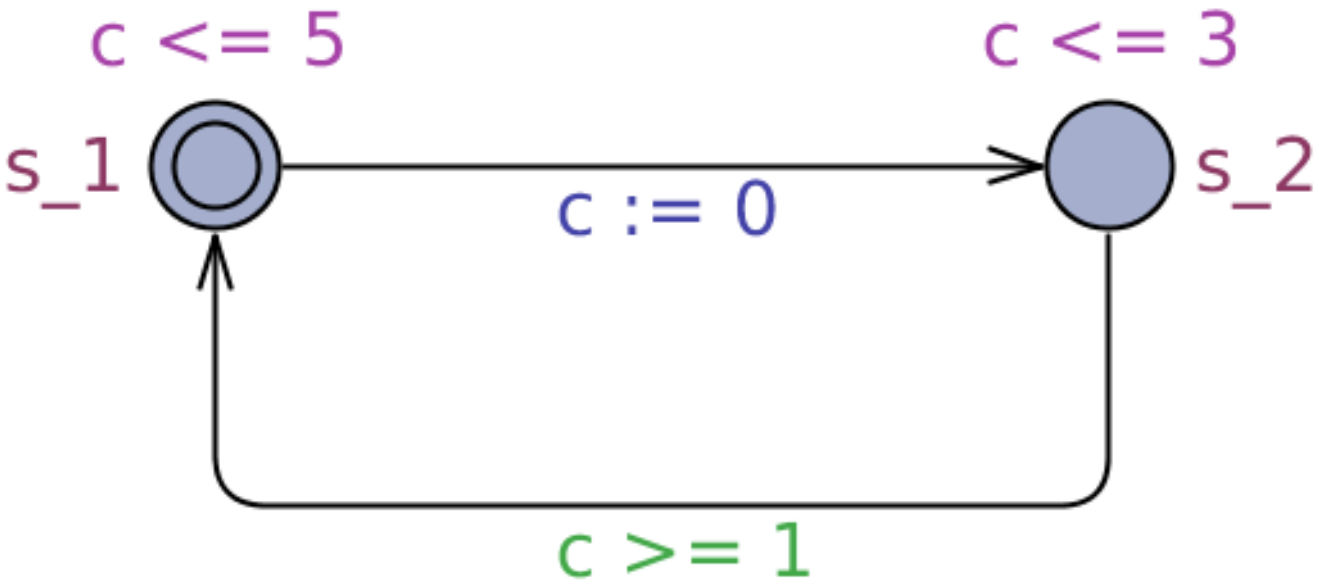
- Urgent states: delay transitions are forbidden
- How to model such states with invariants?

- Commited states is a more strict version of urgent states

- Systems of timed automata can be verified...
- But only with A[], A<>, E[], E<>, $-- >$

# Timed temporal logics: MTL, CTL

# Recall this example of a timed automaton



# Metric temporal logic (MTL)

- Usual Boolean operators are allowed
- Let $I$ be an interval of $\mathbb{R}_+$ with integer bounds, e.g. $[1, 3]$. Open intervals are allowed, e.g. $[2, +\infty)$
- (Timed until) $\phi \, \mathbf{U}_I \, \psi$: there is a position $\pi$ of the timed trace such that $\psi$ holds at this position, $\phi$ holds for each $0 < \pi' < \pi$, and the **duration** of the trace up to position $\pi$ belongs to $I$
- Duration: sum of all delays up to this position
- In pointwise semantics, positions correspond to the elements of the timed trace
- In continuous semantics, there are also intermediate positions

# MTL: other operators

- In LTL, how to express **F** and **G** using **U**?

# MTL: other operators

- In LTL, how to express **F** and **G** using **U**?
- $\mathbf{F}\,\phi = \text{true}\,\mathbf{U}\,\phi$

# MTL: other operators

- In LTL, how to express **F** and **G** using **U**?
- $\mathbf{F}\,\phi = \text{true}\,\mathbf{U}\,\phi$
- $\mathbf{G}\,\phi = \neg\,\mathbf{F}\,\neg\phi = \neg(\text{true}\,\mathbf{U}\,\neg\phi)$

# MTL: other operators

- In LTL, how to express **F** and **G** using **U**?
- $\mathbf{F}\,\phi = \text{true}\,\mathbf{U}\,\phi$
- $\mathbf{G}\,\phi = \neg\,\mathbf{F}\,\neg\phi = \neg(\text{true}\,\mathbf{U}\,\neg\phi)$

- In MTL:
- $\mathbf{F}_I\,\phi = \text{true}\,\mathbf{U}_I\,\phi$, i.e. $\phi$ must happen in interval $I$
- Which $I$ corresponds to the usual **F**?

# MTL: other operators

- In LTL, how to express **F** and **G** using **U**?
- $\mathbf{F}\,\phi = \text{true}\,\mathbf{U}\,\phi$
- $\mathbf{G}\,\phi = \neg\,\mathbf{F}\,\neg\phi = \neg(\text{true}\,\mathbf{U}\,\neg\phi)$

- In MTL:
- $\mathbf{F}_I\,\phi = \text{true}\,\mathbf{U}_I\,\phi$, i.e. $\phi$ must happen in interval $I$
- Which $I$ corresponds to the usual **F**? $I = [0, +\infty)$

# MTL: other operators

- In LTL, how to express **F** and **G** using **U**?
- $\mathbf{F}\,\phi = \text{true}\,\mathbf{U}\,\phi$
- $\mathbf{G}\,\phi = \neg\,\mathbf{F}\,\neg\phi = \neg(\text{true}\,\mathbf{U}\,\neg\phi)$

- In MTL:
- $\mathbf{F}_I\,\phi = \text{true}\,\mathbf{U}_I\,\phi$, i.e. $\phi$ must happen in interval $I$
- Which $I$ corresponds to the usual **F**? $I = [0, +\infty)$
- $\mathbf{G}_I\,\phi = \neg\,\mathbf{F}_I\,\neg\phi$, i.e. $\phi$ holds on the whole interval $I$

# MTL: exercises

- Consider a timed trace: $\{p\} \xrightarrow{1} \{q\} \xrightarrow{2} \{p, q\} \xrightarrow{3} \text{cycle}(\{q\} \xrightarrow{3})$
- Are the following MTL formulae satisfied for this trace (assuming **pointwise** semantics)?

1. $\mathbf{F}_{[2,5]}\, q$

# MTL: exercises

- Consider a timed trace: $\{p\} \xrightarrow{1} \{q\} \xrightarrow{2} \{p, q\} \xrightarrow{3} \text{cycle}(\{q\} \xrightarrow{3})$
- Are the following MTL formulae satisfied for this trace (assuming **pointwise** semantics)?

1. $\mathbf{F}_{[2,5]}\, q$ – Yes: in the third state, the sum of delays so far (duration) equals 3
2. $\mathbf{G}_{[2,5]}(p \wedge q)$

# MTL: exercises

- Consider a timed trace: $\{p\} \xrightarrow{1} \{q\} \xrightarrow{2} \{p, q\} \xrightarrow{3} \text{cycle}(\{q\} \xrightarrow{3})$
- Are the following MTL formulae satisfied for this trace (assuming **pointwise** semantics)?

1. $\mathbf{F}_{[2,5]}\, q$ – Yes: in the third state, the sum of delays so far (duration) equals 3
2. $\mathbf{G}_{[2,5]}(p \wedge q)$ – Yes: the third state is the only state with duration inside $[2, 5]$
3. $\mathbf{G}_{[0,+\infty)}(p \rightarrow \mathbf{F}_{[0,1]}\, q)$

# MTL: exercises

- Consider a timed trace: $\{p\} \xrightarrow{1} \{q\} \xrightarrow{2} \{p, q\} \xrightarrow{3} \text{cycle}(\{q\} \xrightarrow{3})$
- Are the following MTL formulae satisfied for this trace (assuming **pointwise** semantics)?

1. $\mathbf{F}_{[2,5]}\, q$ – Yes: in the third state, the sum of delays so far (duration) equals 3
2. $\mathbf{G}_{[2,5]}(p \wedge q)$ – Yes: the third state is the only state with duration inside $[2, 5]$
3. $\mathbf{G}_{[0,+\infty)}(p \rightarrow \mathbf{F}_{[0,1]}\, q)$ – Yes

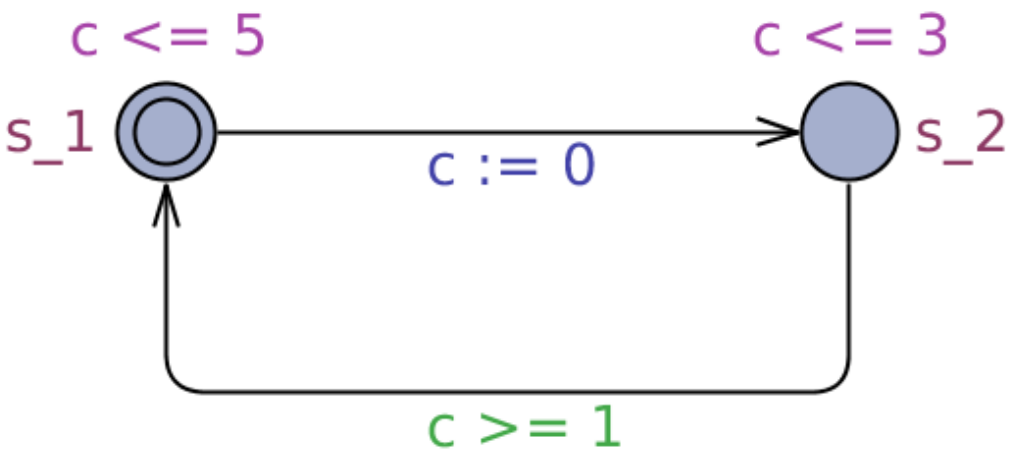
# Timed computation tree logic (TCTL)

- CTL is defined in terms of states / Kripke structures, not paths
- Let's explicitly consider clocks while speaking about states in the context of TCLT
- $(s, c)$: state of the timed automaton and its clock assignment

# Timed computation tree logic (TCTL)

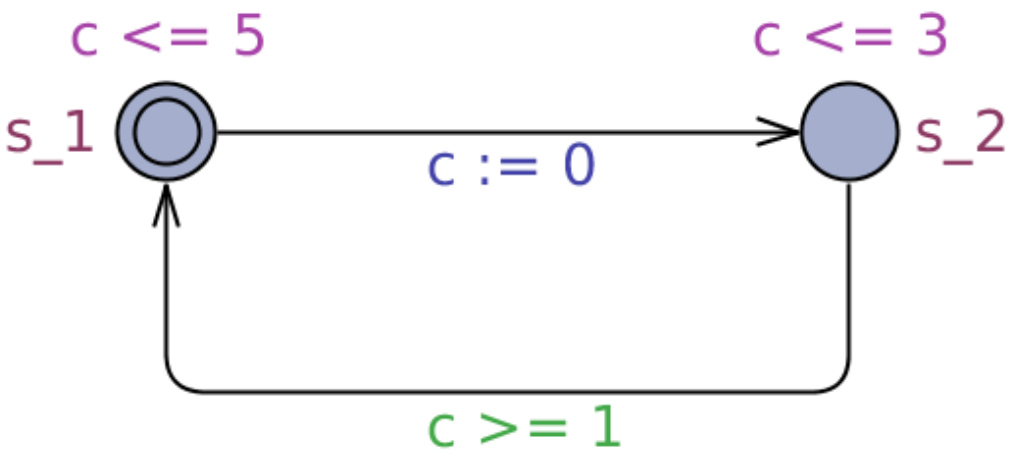
- CTL is defined in terms of states / Kripke structures, not paths
- Let's explicitly consider clocks while speaking about states in the context of TCLT
- $(s, c)$: state of the timed automaton and its clock assignment
- $(s, c) \models \phi\, \mathbf{EU}_I\, \psi$, if there exists an infinite path from $(s, c)$ such that $\phi\, \mathbf{U}_I\, \psi$ holds along this path
- $(s, c) \models \phi\, \mathbf{AU}_I\, \psi$, if for all infinite paths from $(s, c)$ $\phi\, \mathbf{U}_I\, \psi$ holds
- Other operators can be expressed in terms of $\mathbf{EU}_I$ and $\mathbf{AU}_I$

# TCTL: exercises



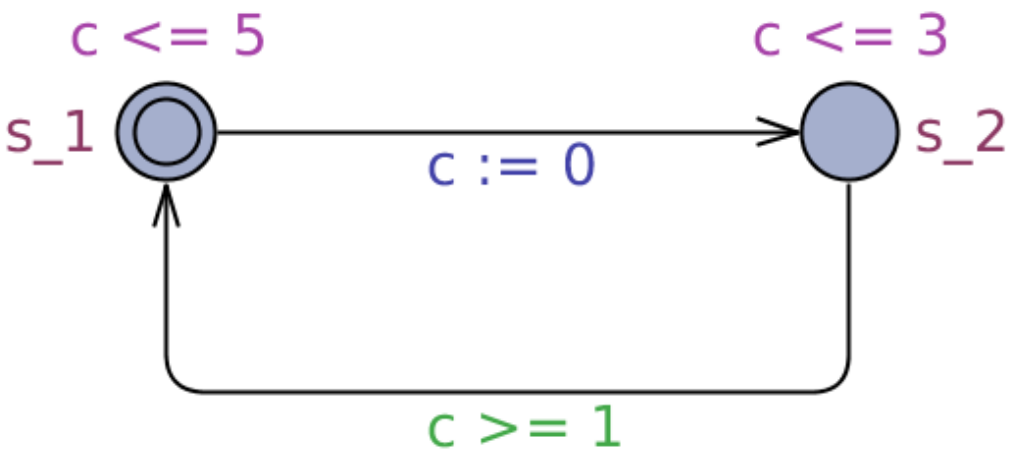

- Reminder: states have invariants, which must always hold
- Let's recall possible behaviors of this timed automaton

# TCTL: exercises




- Reminder: states have invariants, which must always hold
- Let's recall possible behaviors of this timed automaton
- Are these TCTL formulae satisfied? The initial state is $s_1, c = 0$

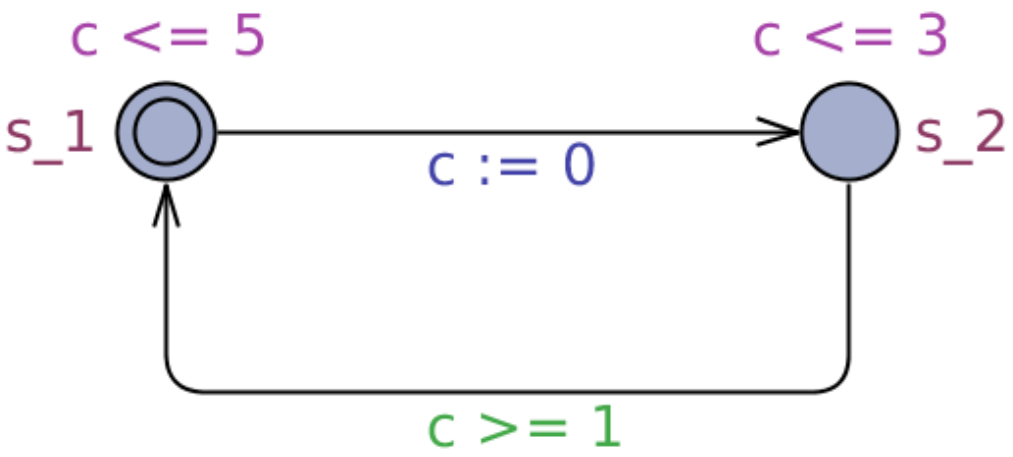
# TCTL: exercises




- Reminder: states have invariants, which must always hold
- Let's recall possible behaviors of this timed automaton
- Are these TCTL formulae satisfied? The initial state is $s_1, c = 0$

1. $\mathbf{EF}_{[0,1]}\, s_2$

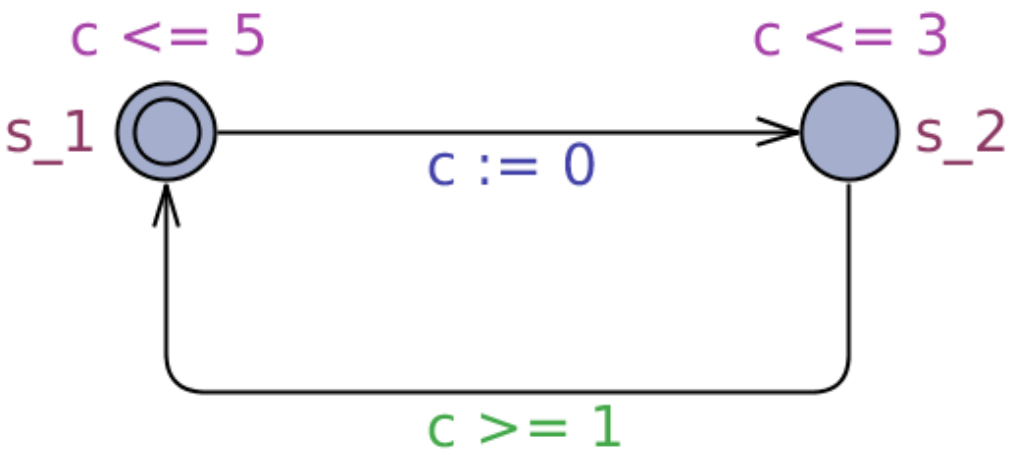
# TCTL: exercises




- Reminder: states have invariants, which must always hold
- Let's recall possible behaviors of this timed automaton
- Are these TCTL formulae satisfied? The initial state is $s_1, c = 0$

1. $\mathbf{EF}_{[0,1]}\, s_2$ – Yes, it is possible to get from $s_1$ to $s_2$ at any time
2. $\mathbf{AF}_{[0,1]}\, s_2$

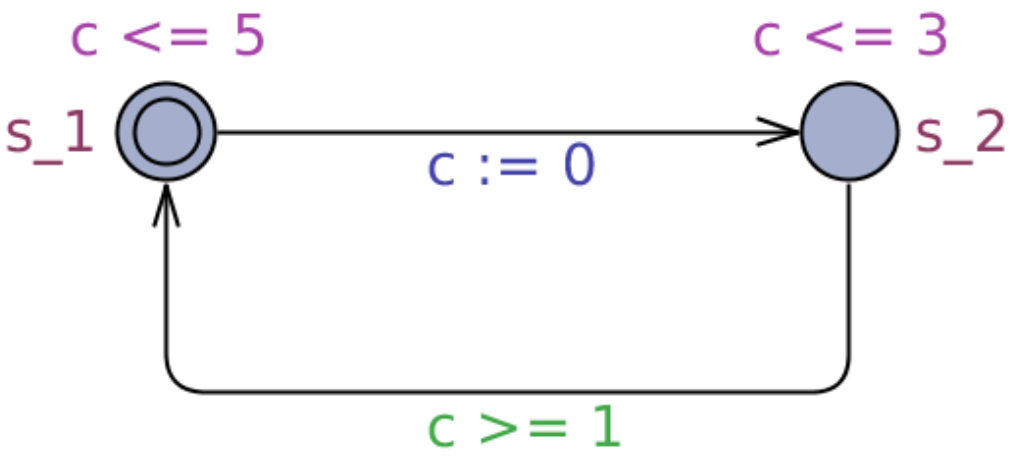
# TCTL: exercises




- Reminder: states have invariants, which must always hold
- Let's recall possible behaviors of this timed automaton
- Are these TCTL formulae satisfied? The initial state is $s_1, c = 0$

1. $\mathbf{EF}_{[0,1]}\, s_2$ – Yes, it is possible to get from $s_1$ to $s_2$ at any time
2. $\mathbf{AF}_{[0,1]}\, s_2$ – No, we can stay in $s_1$ up to $c = 5$
3. $\mathbf{AG}_{[0,+\infty)}(s_2 \to \mathbf{EF}_{[0,0.5]}\, s_1)$

# TCTL: exercises




- Reminder: states have invariants, which must always hold
- Let's recall possible behaviors of this timed automaton
- Are these TCTL formulae satisfied? The initial state is $s_1, c = 0$

1. $\mathbf{EF}_{[0,1]}\, s_2$ – Yes, it is possible to get from $s_1$ to $s_2$ at any time
2. $\mathbf{AF}_{[0,1]}\, s_2$ – No, we can stay in $s_1$ up to $c = 5$
3. $\mathbf{AG}_{[0,+\infty)}(s_2 \to \mathbf{EF}_{[0,0.5]}\, s_1)$ – No, when $s_2$ is entered, the clock is reset, and we need to wait at least for a time unit

# Things are not that easy

- The model checking problem (for infinite words) for MTL under pointwise and continuous semantics is undecidable

# Things are not that easy

- The model checking problem (for infinite words) for MTL under pointwise and continuous semantics is undecidable
- However, there are some subsets of this logic which are

# Things are not that easy

- The model checking problem (for infinite words) for MTL under pointwise and continuous semantics is undecidable
- However, there are some subsets of this logic which are
- Disallowing point intervals (e.g. $I = \{1\}$) helps (MITL temporal logic)

# Things are not that easy

- The model checking problem (for infinite words) for MTL under pointwise and continuous semantics is undecidable
- However, there are some subsets of this logic which are
- Disallowing point intervals (e.g. $I = \{1\}$) helps (MITL temporal logic)
- For finite words, MTL is decidable only under pointwise semantics

# Things are not that easy

- The model checking problem (for infinite words) for MTL under pointwise and continuous semantics is undecidable
- However, there are some subsets of this logic which are
- Disallowing point intervals (e.g. $I = \{1\}$) helps (MITL temporal logic)
- For finite words, MTL is decidable only under pointwise semantics
- Weak (if any) tool support for MTL model-checking

# Things are not that easy

- The model checking problem (for infinite words) for MTL under pointwise and continuous semantics is undecidable
- However, there are some subsets of this logic which are
- Disallowing point intervals (e.g. $I = \{1\}$) helps (MITL temporal logic)
- For finite words, MTL is decidable only under pointwise semantics
- Weak (if any) tool support for MTL model-checking

- Runtime verification: check properties in runtime only for concrete simulations

# Things are not that easy

- The model checking problem (for infinite words) for MTL under pointwise and continuous semantics is undecidable
- However, there are some subsets of this logic which are
- Disallowing point intervals (e.g. $I = \{1\}$) helps (MITL temporal logic)
- For finite words, MTL is decidable only under pointwise semantics
- Weak (if any) tool support for MTL model-checking

- Runtime verification: check properties in runtime only for concrete simulations

- TCTL: PSPACE-complete under any semantics. This is better than decidable fragments of MTL, which are usually EXPSPACE-complete

# Things are not that easy

- The model checking problem (for infinite words) for MTL under pointwise and continuous semantics is undecidable
- However, there are some subsets of this logic which are
- Disallowing point intervals (e.g. $I = \{1\}$) helps (MITL temporal logic)
- For finite words, MTL is decidable only under pointwise semantics
- Weak (if any) tool support for MTL model-checking

- Runtime verification: check properties in runtime only for concrete simulations

- TCTL: PSPACE-complete under any semantics. This is better than decidable fragments of MTL, which are usually EXPSPACE-complete
- Kronos – TCTL model checker for timed automata

# Literature


1. Behrmann, G., David, A., & Larsen, K. G. (2004). A Tutorial on Uppaal. Formal methods for the design of real-time systems, pp. 200–236. Springer Berlin Heidelberg. Available: `http://people.cs.aau.dk/~adavid/publications/21-tutorial.pdf`
2. Bengtsson, J., & Yi, W. (2004). Timed automata: Semantics, algorithms and tools. Lectures on concurrency and Petri nets. Springer Berlin Heidelberg, pp. 87–124.
3. Bouyer, P. (2009). Model-checking timed temporal logics. Electronic Notes in Theoretical Computer Science, 231, 323–341.
4. Alur, R., Courcoubetis, C., & Dill, D. (1990). Model-checking for real-time systems. Fifth Annual IEEE Symposium on Logic in Computer Science (LICS'90), pp. 414–425. IEEE.