Timed automata
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?
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?
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\}, ...
- 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\}, ...
- 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? \(\text{X} \ldots \text{X} p = \text{X}^{1000} p\). 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 \( \{p\}, \{q\}, \{p, q\}, ... \)
- 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? \( X \ldots X p = X^{1000} p \). 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:** \( \{p\} \xrightarrow{2s} \{q\} \xrightarrow{0.1s} \{p, q\}, ... \)
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**: $\{p\} \xrightarrow{2s} \{q\} \xrightarrow{0.1s} \{p, q\}, ...$
- 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
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
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
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
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 $c$

![Timed Automaton Diagram](image)

- The upper transition resets the clock with $c \leq 5$
- The lower transition is guarded by a clock with $c := 0$
- 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?
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

![Timed Automaton Diagram]

- Transition: c := 0
- Invariants:
  - c <= 5
  - c <= 3
  - c >= 1
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

Can you think of examples of timed behaviors possible for this automaton?
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.

- 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

(b) User.
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?
- Committed states is a more strict version of urgent states
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 $A[], A<>$, $E[]$, $E<>$, $\neg\neg >$
Timed temporal logics: MTL, CTL
Recall this example of a timed automaton

\[
\begin{align*}
&\text{c} \leq 5 \\
&s_1 \\
&\text{c} := 0 \\
&s_2 \\
&\text{c} \leq 3 \\
&\text{c} \geq 1
\end{align*}
\]
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
In LTL, how to express $F$ and $G$ using $U$?

- $F \varphi = \text{true} U \varphi$
- $G \varphi = \neg F \neg \varphi = \neg (\text{true} U \neg \varphi)$

In MTL:
- $F I \varphi = \text{true} U I \varphi$, i.e. $\varphi$ must happen in interval $I$
- $G I \varphi = \neg F I \neg \varphi$, i.e. $\varphi$ holds on the whole interval $I$
In LTL, how to express $F$ and $G$ using $U$?

$F \phi = \text{true } U \phi$
MTL: other operators

- In LTL, how to express \( F \) and \( G \) using \( U \)?
- \( F \phi = \text{true} U \phi \)
- \( G \phi = \neg F \neg \phi = \neg (\text{true} U \neg \phi) \)
MTL: other operators

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

- In MTL:
  - $F_I \phi = \text{true} U_I \phi$, 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 $U$?
  - $F \phi = true \ U \phi$
  - $G \phi = \neg F \neg \phi = \neg (true \ U \neg \phi)$

- In MTL:
  - $F_I \phi = true \ U_I \phi$, i.e. $\phi$ must happen in interval $I$
  - Which $I$ corresponds to the usual $F$? $I = [0, +\infty)$
In LTL, how to express $F$ and $G$ using $U$?

- $F \phi = \text{true } U \phi$
- $G \phi = \neg F \neg \phi = \neg (\text{true } U \neg \phi)$

In MTL:

- $F_I \phi = \text{true } U_I \phi$, i.e. $\phi$ must happen in interval $I$
- Which $I$ corresponds to the usual $F$? $I = [0, +\infty)$
- $G_I \phi = \neg F_I \neg \phi$, i.e. $\phi$ holds on the whole interval $I$
Consider a timed trace: \( \{p\} \xrightarrow{1} \{q\} \xrightarrow{2} \{p, q\} \xrightarrow{3} \text{cycle}\left(\{q\}\xrightarrow{3}\right) \)

Are the following MTL formulae satisfied for this trace (assuming pointwise semantics)?

1. \( F_{[2,5]} q \)
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. \( F_{[2,5]} q \) – Yes: in the third state, the sum of delays so far (duration) equals 3
2. \( G_{[2,5]} (p \land q) \)
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. \( F_{[2,5]} q \) – Yes: in the third state, the sum of delays so far (duration) equals 3
2. \( G_{[2,5]} (p \land q) \) – Yes: the third state is the only state with duration inside [2, 5]
3. \( G_{[0,\infty)} (p \rightarrow F_{[0,1]} q) \)
Consider a timed trace: \( \{p\} \rightarrow^1 \{q\} \rightarrow^2 \{p, q\} \rightarrow^3 \text{cycle}(\{q\}) \)

Are the following MTL formulae satisfied for this trace (assuming pointwise semantics)?

1. \( F_{[2,5]} q \) – Yes: in the third state, the sum of delays so far (duration) equals 3
2. \( G_{[2,5]}(p \land q) \) – Yes: the third state is the only state with duration inside [2, 5]
3. \( G_{[0,\infty]}(p \rightarrow F_{[0,1]} q) \) – 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

\[(s, c) \models \varphi \text{ EU } \psi\] if there exists an infinite path from \((s, c)\) such that \(\varphi \text{ U } \psi\) holds along this path

\[(s, c) \models \varphi \text{ AU } \psi\] if for all infinite paths from \((s, c)\) \(\varphi \text{ U } \psi\) holds
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 TCTL
- \((s, c)\): 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 \(\mathbf{AU}_I\)
Reminder: states have invariants, which must always hold

Let’s recall possible behaviors of this timed automaton

- $s_1$: clock $c \leq 5$
- $s_2$: clock $c \leq 3$
- Transition: $c := 0$ from $s_1$ to $s_2$
- Invariant: $c \geq 1$

Are these TCTL formulae satisfied?

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

