Back to the Future

Two-Way Timed Automata

Gautham Viswanathan

Chennai Mathematical Institute

2023

Contents

Contents

Timed Automata

We already know what timed automata are:

Timed Automata

We already know what timed automata are:

Timed Automata

We already know what timed automata are:

Goal for today: define and investigate two way timed automata and a related real time logic.

Idea

Idea

- Can move back and forth across the input

Idea

- Can move back and forth across the input
- Clocks don't really 'elapse time', but rather store timestamps of letters

Idea

- Can move back and forth across the input
- Clocks don't really 'elapse time', but rather store timestamps of letters
  - Machine reads through an 'event log'

Idea

- Can move back and forth across the input
- Clocks don't really 'elapse time', but rather store timestamps of letters
  - Machine reads through an 'event log'
- Notion of determinism: one action possible from a given configuration

Definition

Definition (2NTA)

- A 2NTA $\mathcal{A}$ is a 6-tuple $(\Sigma, S, S_0, S_F, C, E)$.

Definition

Definition (2NTA)

- A 2NTA $\mathcal{A}$ is a 6-tuple $(\Sigma, S, S_0, S_F, C, E)$.
- Transitions: (s, s', σ, R, g, d) where d is a direction: forwards or backwards

Definition

Definition ($2NTA$)

- A $2NTA$ $\mathcal{A}$ is a 6-tuple $(\Sigma, S, S_0, S_F, C, E)$.
- Transitions: (s, s', σ, R, g, d) where d is a direction: forwards or backwards
- Constraints are boolean combinations of $x \leq T + c$, for $x \in C$

Definition

Definition (2NTA)

- A 2NTA $\mathcal{A}$ is a 6-tuple $(\Sigma, S, S_0, S_F, C, E)$.
- Transitions: (s, s', σ, R, g, d) where d is a direction: forwards or backwards
- Constraints are boolean combinations of $x \leq T + c$, for $x \in C$
- T is a special variable which takes on the value of the current timestamp

Definition

Definition (2*NTA*)

- A $2NTA$ $\mathcal{A}$ is a 6-tuple $(\Sigma, S, S_0, S_F, C, E)$.
- Transitions: (s, s', σ, R, g, d) where d is a direction: forwards or backwards
- Constraints are boolean combinations of $x \leq T + c$, for $x \in C$
- T is a special variable which takes on the value of the current timestamp
- c is an integer

Runs

- Start at a start state with all clocks set to 0

Runs

- Start at a start state with all clocks set to 0
- To take a transition $e = (s, s', \sigma, R, g, d)$, do the following:

Runs

- Start at a start state with all clocks set to 0
- To take a transition $e = (s, s', \sigma, R, g, d)$, do the following:
  - Set all clocks in R to an arbitrary real number

Runs

- Start at a start state with all clocks set to 0
- To take a transition $e = (s, s', \sigma, R, g, d)$, do the following:
  - Set all clocks in R to an arbitrary real number
  - Now, check if the clocks satisfy the guard g

Runs

- Start at a start state with all clocks set to 0
- To take a transition $e = (s, s', \sigma, R, g, d)$, do the following:
    - Set all clocks in R to an arbitrary real number
    - Now, check if the clocks satisfy the guard g
    - Move the tape head forward or backward as per d

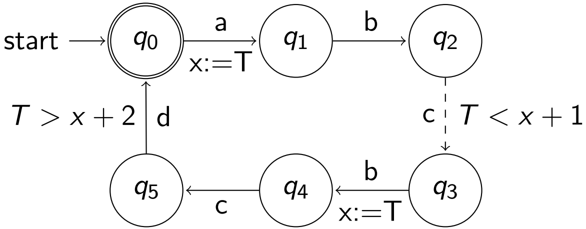
Runs

- Start at a start state with all clocks set to 0
- To take a transition $e = (s, s', \sigma, R, g, d)$, do the following:
  - Set all clocks in R to an arbitrary real number
  - Now, check if the clocks satisfy the guard g
  - Move the tape head forward or backward as per d
- Computation stops when the automaton has no possible transitions to take

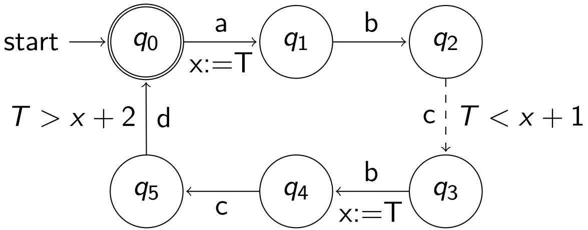
Runs

- Start at a start state with all clocks set to 0
- To take a transition $e = (s, s', \sigma, R, g, d)$, do the following:
  - Set all clocks in R to an arbitrary real number
  - Now, check if the clocks satisfy the guard g
  - Move the tape head forward or backward as per d
- Computation stops when the automaton has no possible transitions to take
- Accept if in a final state, reject otherwise

Example 1

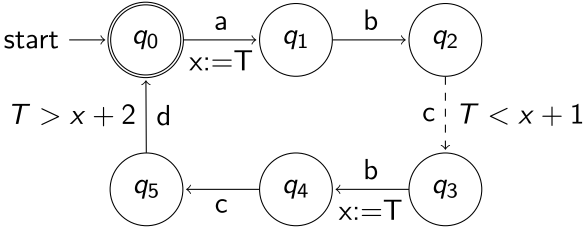


Example 1



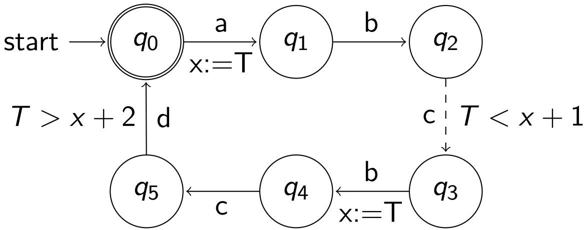
- Notation: $x := T$ stands for $x \in R$ and the guard $x = T$

Example 1



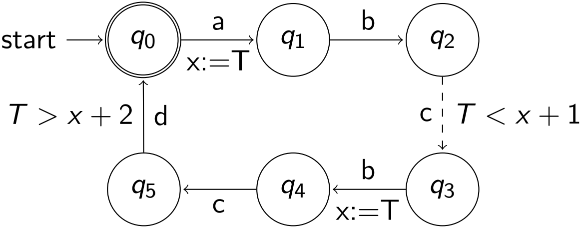
- Notation: $x := T$ stands for $x \in R$ and the guard $x = T$
- The automaton accepts the language $\{abcd \mid t_c - t_a < 1, t_d - t_b > 2\}$

Example 1



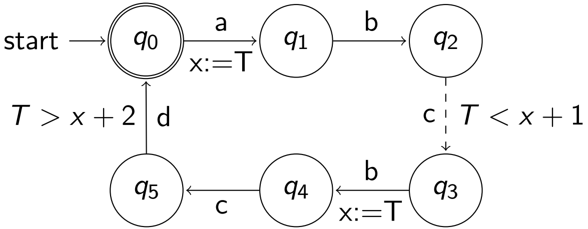
- Notation: $x := T$ stands for $x \in R$ and the guard $x = T$
- The automaton accepts the language $\{abcd \mid t_c - t_a < 1, t_d - t_b > 2\}$
- Does it with only one clock!

Example 1



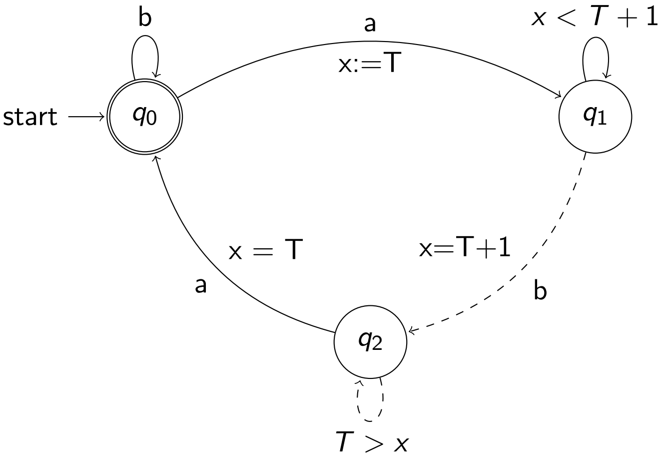
- Notation: $x := T$ stands for $x \in R$ and the guard $x = T$
- The automaton accepts the language $\{abcd \mid t_c - t_a < 1, t_d - t_b > 2\}$
- Does it with only one clock!
- First checks the condition on c and a, then reuses the same clock to check the condition on b and d.

Example 1

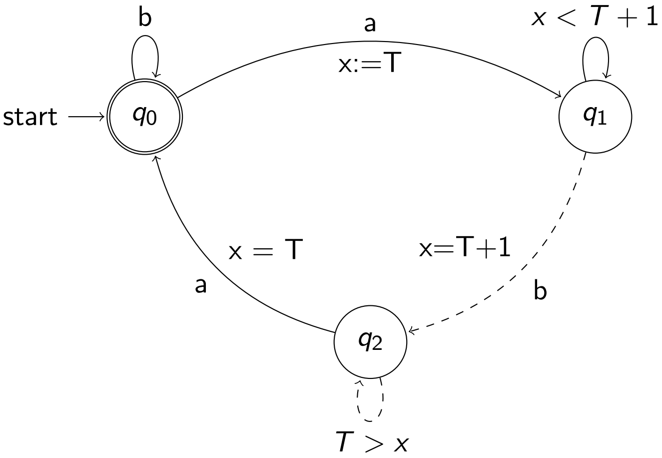


- Notation: $x := T$ stands for $x \in R$ and the guard $x = T$
- The automaton accepts the language $\{abcd \mid t_c - t_a < 1, t_d - t_b > 2\}$
- Does it with only one clock!
- First checks the condition on c and a, then reuses the same clock to check the condition on b and d.
- Overall visits each symbol at most two times

Example 2



Example 2



Example 2 (contd.)

- Accepts the language $\{w \in \{a, b\}^* \mid \forall\ a, \exists\ b \text{ one unit in the future}\}$

Example 2 (contd.)

- Accepts the language $\{w \in \{a, b\}^* \mid \forall\ a, \exists\ b$ one unit in the future$\}$
- Can use more clocks to ensure that no two events happen at the same timestamp

Example 2 (contd.)

- Accepts the language $\{w \in \{a, b\}^* \mid \forall\ a, \exists\ b \text{ one unit in the future}\}$
- Can use more clocks to ensure that no two events happen at the same timestamp
- This language is not accepted by NTAs (its untime is not regular)!

Example 2 (contd.)

- Accepts the language $\{w \in \{a, b\}^* \mid \forall\ a, \exists\ b$ one unit in the future$\}$
- Can use more clocks to ensure that no two events happen at the same timestamp
- This language is not accepted by NTAs (its untime is not regular)!
- Letters of the word are visited *arbitrarily many* times

Determinism

- Resets have to be deterministic; only allow resetting to the current timestamp

Contents

Boundedness

Definition (Reversals, Bounded 2TAs)

- The number of reversals that a 2TA makes in a run is the maximum number of times that it visits any symbol of the input

Boundedness

Definition (Reversals, Bounded 2TAs)

- The number of reversals that a 2TA makes in a run is the maximum number of times that it visits any symbol of the input
- A 2NTA / 2DTA is k-bounded if on all runs, it has at most $2k + 1$ reversals

Boundedness

Definition (Reversals, Bounded 2TAs)

- The number of reversals that a 2TA makes in a run is the maximum number of times that it visits any symbol of the input
- A 2NTA / 2DTA is k-bounded if on all runs, it has at most $2k + 1$ reversals
- The automaton of the first example was of type DTA_1

Boundedness

Definition (Reversals, Bounded 2TAs)

- The number of reversals that a 2TA makes in a run is the maximum number of times that it visits any symbol of the input
- A 2NTA / 2DTA is k-bounded if on all runs, it has at most $2k+1$ reversals
- The automaton of the first example was of type DTA_1
- The automaton of the second example is unbounded

Equivalence with the Standard Model

- This new model should equal our original one when restricted to a single pass

Equivalence with the Standard Model

- This new model should equal our original one when restricted to a single pass
- The point: NTAs store elapsed time information in clocks, $2NTA$s store timestamps

Equivalence with the Standard Model

- This new model should equal our original one when restricted to a single pass
- The point: NTAs store elapsed time information in clocks, $2NTA$s store timestamps
- Can convert one to the other by comparing with the current timestamp

Contents

Expressive Power

The major results:

- $DTA_k \subsetneq DTA_{k+1}$: expressive power increases when we allow more passes

Expressive Power

The major results:

- $DTA_k \subsetneq DTA_{k+1}$: expressive power increases when we allow more passes
- $\cup_k NTA_k = NTA_0$: any bounded number of passes can be simulated by a single nondeterministic pass

Expressive Power

The major results:

- $DTA_k \subsetneq DTA_{k+1}$: expressive power increases when we allow more passes
- $\cup_k NTA_k = NTA_0$: any bounded number of passes can be simulated by a single nondeterministic pass
- $\cup_k DTA_k$ is closed under boolean operations

Expressive Power

The major results:

- $DTA_k \subsetneq DTA_{k+1}$: expressive power increases when we allow more passes
- $\cup_k NTA_k = NTA_0$: any bounded number of passes can be simulated by a single nondeterministic pass
- $\cup_k DTA_k$ is closed under boolean operations
- $NTA_0 \subsetneq 2NTA$: unbounded 2NTAs are stronger than bounded ones

Complexity

- Unbounded two-wayness: very bad
- The emptiness problem for $2DTA$s is undecidable
- Boundedness: desirable and harmless
- Emptiness (and therefore universality and inclusion) for DTA_ks are PSPACE-complete

Contents

Motivation

- Temporal logics: a way to specify properties of a timed system

Motivation

- Temporal logics: a way to specify properties of a timed system
- For example: in the first two seconds of execution, every a is followed by a b 1 to 2 seconds later.

Syntax

Definition (Syntax)

- Usual propositional logic syntax plus ‘time bounded until’

Syntax

Definition (Syntax)

- Usual propositional logic syntax plus ‘time bounded until’
- Fix a set of propositions P

Syntax

Definition (Syntax)

- Usual propositional logic syntax plus ‘time bounded until’
- Fix a set of propositions P
- $\phi := p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \, \mathcal{U}_I \, \phi_2$

Syntax

Definition (Syntax)

- Usual propositional logic syntax plus 'time bounded until'
- Fix a set of propositions P
- $\phi := p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \, \mathcal{U}_I \, \phi_2$
- I is a nonsingular interval with integer endpoints

Semantics

- Interpret formulas over timed words $w = (\bar{\sigma}, \bar{\tau})$

Semantics

- Interpret formulas over timed words $w = (\bar{\sigma}, \bar{\tau})$
- Each letter represents the *world state* at the current time

Semantics

- Interpret formulas over timed words $w = (\bar{\sigma}, \bar{\tau})$
- Each letter represents the *world state* at the current time
- Each letter is a valuation from P to $\{\bot, \top\}$

Semantics

- Interpret formulas over timed words $w = (\bar{\sigma}, \bar{\tau})$
- Each letter represents the *world state* at the current time
- Each letter is a valuation from P to $\{\perp, \top\}$
- If a formula ϕ is true at the i^{th} symbol of a word w, we write $(w, i) \models \phi$

Semantics (contd.)

Define the satisfaction relation as follows:

Semantics (contd.)

Define the satisfaction relation as follows:

- $(w, i) \models p$ iff $p \in \sigma_i$

Semantics (contd.)

Define the satisfaction relation as follows:

- $(w, i) \models p$ iff $p \in \sigma_i$
- $(w, i) \models \neg\phi$ iff $(w, i) \not\models \phi$

Semantics (contd.)

Define the satisfaction relation as follows:

- $(w, i) \models p$ iff $p \in \sigma_i$
- $(w, i) \models \neg\phi$ iff $(w, i) \not\models \phi$
- $(w, i) \models \phi_1 \wedge \phi_2$ iff $(w, i) \models \phi_1$ and $(w, i) \models \phi_2$

Semantics (contd.)

Define the satisfaction relation as follows:

- $(w, i) \models p$ iff $p \in \sigma_i$
- $(w, i) \models \neg\phi$ iff $(w, i) \not\models \phi$
- $(w, i) \models \phi_1 \wedge \phi_2$ iff $(w, i) \models \phi_1$ and $(w, i) \models \phi_2$
- $(w, i) \models \phi_1 \, \mathcal{U}_I \, \phi_2$ iff $(w, j) \models \phi_2$ for some $i \leq j \leq n$ such that:

Semantics (contd.)

Define the satisfaction relation as follows:

- $(w, i) \models p$ iff $p \in \sigma_i$
- $(w, i) \models \neg\phi$ iff $(w, i) \not\models \phi$
- $(w, i) \models \phi_1 \wedge \phi_2$ iff $(w, i) \models \phi_1$ and $(w, i) \models \phi_2$
- $(w, i) \models \phi_1 \, \mathcal{U}_I \, \phi_2$ iff $(w, j) \models \phi_2$ for some $i \leq j \leq n$ such that:
    - $\tau_j \in \tau_i + I$

Semantics (contd.)

Define the satisfaction relation as follows:

- $(w, i) \models p$ iff $p \in \sigma_i$
- $(w, i) \models \neg\phi$ iff $(w, i) \not\models \phi$
- $(w, i) \models \phi_1 \wedge \phi_2$ iff $(w, i) \models \phi_1$ and $(w, i) \models \phi_2$
- $(w, i) \models \phi_1 \, \mathcal{U}_I \, \phi_2$ iff $(w, j) \models \phi_2$ for some $i \leq j \leq n$ such that:
  - $\tau_j \in \tau_i + I$
  - $(w, k) \models \phi_1$ for all $i \leq k < j$

Semantics (contd.)

Define the satisfaction relation as follows:

- $(w, i) \models p$ iff $p \in \sigma_i$
- $(w, i) \models \neg\phi$ iff $(w, i) \not\models \phi$
- $(w, i) \models \phi_1 \wedge \phi_2$ iff $(w, i) \models \phi_1$ and $(w, i) \models \phi_2$
- $(w, i) \models \phi_1 \, \mathcal{U}_I \, \phi_2$ iff $(w, j) \models \phi_2$ for some $i \leq j \leq n$ such that:
    - $\tau_j \in \tau_i + I$
    - $(w, k) \models \phi_1$ for all $i \leq k < j$

A word w over models ϕ if $(w, 1) \models \phi$.

Examples!

- $p \, \mathcal{U}_{[1,2]} \, q$
- $\top \, \mathcal{U}_{[2,3)} \, (p \wedge q)$
- $p \, \mathcal{U}_{[1,2]} \, \left(q \, \mathcal{U}_{[1,2]} \, r\right)$

Into the Past

- Can extend MITL to look into the past as well

Into the Past

- Can extend MITL to look into the past as well
- Add a binary 'time bounded since' operator $\mathcal{S}_I$ with the following semantics:

Into the Past

- Can extend MITL to look into the past as well
- Add a binary 'time bounded since' operator $\mathcal{S}_I$ with the following semantics:
- $(w, i) \models \phi_1 \, \mathcal{S} \, \phi_2$ iff $(w, j) \models \phi_2$ for some $0 \leq j \leq i$ such that:

Into the Past

- Can extend MITL to look into the past as well
- Add a binary 'time bounded since' operator $\mathcal{S}_I$ with the following semantics:
- $(w, i) \models \phi_1 \, \mathcal{S} \, \phi_2$ iff $(w, j) \models \phi_2$ for some $0 \leq j \leq i$ such that:
    - $\tau_j \in \tau_i - I$

Into the Past

- Can extend MITL to look into the past as well
- Add a binary 'time bounded since' operator $\mathcal{S}_I$ with the following semantics:
- $(w, i) \models \phi_1 \, \mathcal{S} \, \phi_2$ iff $(w, j) \models \phi_2$ for some $0 \leq j \leq i$ such that:
  - $\tau_j \in \tau_i - I$
  - $(w, k) \models \phi_1$ for all $j < k \leq i$

Into the Past

- Can extend MITL to look into the past as well
- Add a binary 'time bounded since' operator $\mathcal{S}_I$ with the following semantics:
- $(w, i) \models \phi_1 \, \mathcal{S} \, \phi_2$ iff $(w, j) \models \phi_2$ for some $0 \leq j \leq i$ such that:
  - $\tau_j \in \tau_i - I$
  - $(w, k) \models \phi_1$ for all $j < k \leq i$
- Note: this is simply a reversed 'until'!

Alternations

- We can alternate between ‘until’ and ‘since’

Alternations

- We can alternate between ‘until’ and ‘since’
  - $p \, \mathcal{U}_{[1,2]} \, \left(q \, \mathcal{S}_{[0,1)} \, r\right)$ alternates once

Alternations

- We can alternate between ‘until’ and ‘since’
  - $p \, \mathcal{U}_{[1,2]} \, \left(q \, \mathcal{S}_{[0,1)} \, r\right)$ alternates once
- Corresponds to going back and forth across a word to check conditions

Alternations

- We can alternate between ‘until’ and ‘since’
  - $p\,\mathcal{U}_{[1,2]}\ \left(q\,\mathcal{S}_{[0,1)}\ r\right)$ alternates once
- Corresponds to going back and forth across a word to check conditions
- This intuition is in fact what happens: alternations increase expressive power

Alternations

- We can alternate between 'until' and 'since'
  - $p\,\mathcal{U}_{[1,2]}\ \left(q\,\mathcal{S}_{[0,1)}\ r\right)$ alternates once
- Corresponds to going back and forth across a word to check conditions
- This intuition is in fact what happens: alternations increase expressive power
- Let $MITL_k^P$ be the fragment of this extended logic which allows for k alternations between past and future operators

Alternations

- We can alternate between ‘until’ and ‘since’
  - $p\,\mathcal{U}_{[1,2]}\ \left(q\,\mathcal{S}_{[0,1)}\ r\right)$ alternates once
- Corresponds to going back and forth across a word to check conditions
- This intuition is in fact what happens: alternations increase expressive power
- Let $MITL_k^P$ be the fragment of this extended logic which allows for k alternations between past and future operators
  - Note: This is not the exact definition, but close enough to convey the point

Contents

Connection with 2TAs

The major results:

Connection with 2TAs

The major results:

Theorem (Alur, Henzinger)

- For every MITL formula ϕ, there exists a DTA_1 A_ϕ that accepts precisely the models of ϕ.

Connection with 2TAs

The major results:

Theorem (Alur, Henzinger)

- For every MITL formula ϕ, there exists a DTA_1 A_ϕ that accepts precisely the models of ϕ.
- $MITL_k^P \subset DTA_k$.

Thank You!

Vielen Dank!
Haben Sie Fragen?

Reference

These slides present the work in *Back to the Future: Towards a Theory of Timed Regular Languages* by R. Alur and T. Henzinger, published in 1992.