Hybrid Systems
Verification Techniques for Hybrid Systems

גרא וייס
המחלקה למדעי המחשב
אוניברסיטת בן-גוריון
וביינברשטט ב-גוריון
Timed Automata

- We will now focus on a specific type of Hybrid Systems called Timed-Automata

- We will first define formally the set of systems that we will analyze

- And then provide an algorithm for reachability for this type of systems
The set, $\Phi(X)$, of **clock constraints** of a set over $X = \mathbb{R}^n$, is a set of logical expressions defined inductively by $\delta \in \Phi(X)$ if:

$$\delta := (x_i \leq c) \ | \ (x_i \geq c) \ | \ \neg \delta \ | \ \delta_1 \land \delta_2$$

where $x_i \in \{x_1, \ldots, x_n\}$ and $c \geq 0$ is a rational constant.
Examples

Let $X = \mathbb{R}^2$:

- $(x_1 \leq 1) \in \Phi(X)$
- $(0 \leq x_1 \leq 1) \in \Phi(X)$, since $(0 \leq x_1 \leq 1) \iff (x_1 \geq 0) \land (x_1 \leq 1)$
- $(x_1 = 1) \in \Phi(X)$, since $(x_1 = 1) \iff (x_1 \geq 1) \land (x_1 \leq 1)$
- $(x_1 < 1) \in \Phi(X)$, since $(x_1 < 1) \iff (x_1 \leq 1) \land \neg(x_1 \geq 1)$
- $True \in \Phi(X)$, since $True \iff \neg((x_1 = 1) \land \neg(x_1 = 1))$
- $(x_1 \leq x_2) \notin \Phi(X)$

Given $\delta \in \Phi(X)$, we say that $x \in X$ satisfies $\delta$ if $\delta(x) = True$

To $\delta \in \Phi(X)$ we associate a set: $\hat{\delta} = \{x \in X : \delta(x) = True\}$
A **timed automaton** is a hybrid system $H = \langle Q, X, Init, f, Dom, E, G, R \rangle$ where

- $Q$ is a finite set of discrete modes, $Q = \{q_1, ..., q_m\}$
- $X = \mathbb{R}^n$
- $Init = \bigcup_{i=1}^{m} (\{q_i\} \times \widehat{Init}_{q_i})$ where $Init_{q_i} \in \Phi(X)$ is a clock constraint formalizing the initial clock valuations at each mode.
- $f(q, x) = (1, ..., 1)$ for all $\langle q, x \rangle \in Q \times X$
- $Dom(q) = X$ for all $q \in Q$
- $E \subseteq Q \times Q$
- $G(e) = \hat{G}_e$ where $G_e \in \Phi(X)$ is a clock constraint specifying the guard for each transition $e = \langle q, q' \rangle \in E$
- For all $e \in E$, $i \in \{1, ..., n\}$, $R(e, x)$ either leaves $x_i$ unaffected or resets it to 0 (notice that $R$ is single valued)
Example

- \( Q = \{q_1, q_2\} \)
- \( X = \mathbb{R}^2 \)
- \( \text{Init} = \langle q_1, (0, 0)^T \rangle \)
- \( f(q, x) = (1, 1)^T \) for all \( \langle q, x \rangle \)
- \( \text{Dom}(q) = \mathbb{R}^2 \) for all \( q \in Q \)
- \( E = \{\langle q_1, q_2\rangle, \langle q_2, q_1\rangle\} \)
- \( G(q_1, q_2) = \{x \in \mathbb{R}^2: (x_1 \leq 3) \land (x_2 \geq 2)\} \)
- \( G(q_2, q_1) = \{x \in \mathbb{R}^2: (x_1 \leq 1)\} \)
- \( R(q_1, q_2, x) = \{(0, x_2)^T\}, R(q_2, q_1, x) = \{(x_1, 0)^T\} \)
A transition system is a quintuple $T = \langle S, \Sigma, \rightarrow, S_0, S_F \rangle$, where

- $S$ is a set of states
- $\Sigma$ is an alphabet of events
- $\rightarrow \subseteq S \times \Sigma \times S$ is a transition relation
- $S_0 \subseteq S$ is a set of initial states
- $S_F \subseteq S$ is a set of final states
Example

- $S = \{s_1, s_2, s_3\}$
- $\Sigma = \{f, b\}$
- $\rightarrow = \{⟨s_1, f, s_2⟩, ⟨s_2, f, s_3⟩, ⟨s_3, b, s_2⟩, ⟨s_2, b, s_1⟩\}$
- $S_0 = \{s_1\}$
- $S_F = \{s_3\}$
Consider a set of final states of the form $F = \bigcup_{i=1}^{m}(\{q_i\} \times \hat{F}_{q_i})$ where $F_{q_i} \in \Phi(X)$

A timed automaton together with a set of final states, $F$, can be viewed as a transition system, $T = \langle S, \Sigma, \rightarrow, S_0, S_F \rangle$, with:

- $S = Q \times X$
- $\Sigma = E \cup \{\tau\}$, where $\tau$ is a symbol denoting time passage
- $\langle\langle q, x\rangle, e, \langle q', x'\rangle\rangle \in \rightarrow$ iff $e = \langle q, q'\rangle \in E$, $x \in G(e)$ and $x' \in R(e, x)$
- $\langle\langle q, x\rangle, \tau, \langle q', x'\rangle\rangle \in \rightarrow$ iff $q = q'$, and there exists $t \geq 0$ such that $x' = x + t(1, \ldots, 1)^T$
- $S_0 = \text{Init}$
- $S_F = F$
Example

\[
\begin{align*}
\langle q_1, \left( \begin{array}{c} 0 \\ 0 \end{array} \right) \rangle & \xrightarrow{\tau \atop 2.5} \langle q_1, \left( \begin{array}{c} 2.5 \\ 2.5 \end{array} \right) \rangle \\
\langle q_1, \left( \begin{array}{c} 2.5 \\ 2.5 \end{array} \right) \rangle & \xrightarrow{\langle q_1, q_2 \rangle} \langle q_2, \left( \begin{array}{c} 0 \\ 2.5 \end{array} \right) \rangle \\
\langle q_2, \left( \begin{array}{c} 0 \\ 2.5 \end{array} \right) \rangle & \xrightarrow{\tau \atop 0.5} \langle q_2, \left( \begin{array}{c} 0.5 \\ 3 \end{array} \right) \rangle \\
\langle q_2, \left( \begin{array}{c} 0.5 \\ 3 \end{array} \right) \rangle & \xrightarrow{\langle q_2, q_1 \rangle} \langle q_1, \left( \begin{array}{c} 0.5 \\ 0 \end{array} \right) \rangle
\end{align*}
\]
Given a transition system $T$, is any state $s_f \in S_F$ reachable from a state $s_0 \in S_0$ by a sequence of transitions?
Basic algorithm for reachability

\[ \text{Reach}_0 := S_0 \]
\[ \text{Reach}_{-1} := \emptyset \]
\[ i = 0 \]

\textbf{While} \text{Reach}_i \neq \text{Reach}_{i-1}

\[ \text{Reach}_{i+1} := \text{Reach}_i \cup \{ s' \in S : \exists s \in \text{Reach}_i, \sigma \in \Sigma \text{ with } \langle s, \sigma, s' \rangle \} \]
\[ i := i + 1 \]
Equivalence relation

Definition:

A relation $\sim \subseteq S \times S$ is called an equivalence relation if it is:

- **Reflexive**: $\langle s, s \rangle \in \sim$ for all $s \in S$
- **Symmetric**: $\langle s, s' \rangle \in \sim$ implies that $\langle s', s \rangle \in \sim$
- **Transitive**: $\langle s, s' \rangle \in \sim$ and $\langle s', s'' \rangle \in \sim$ implies that also $\langle s, s'' \rangle \in \sim$

For simplicity we write $s \sim s'$ instead of $\langle s, s' \rangle \in \sim$ and say that $s$ is equivalent to $s'$

Examples of equivalence relations:

- "Has the same first name as" on the set of all people
- "Is similar to" or "congruent to" on the set of all triangles
- "Is congruent to modulo $n$" on the integers
- "Has the same image under $f$" on the domain of a function $f$
- "Is parallel to" on the set of subspaces of an affine space.
Examples

- **For the set** \( \{a, b, c\} \)

- \( \sim_1 = \{\langle a, a\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle b, b\rangle, \langle c, c\rangle\} \) is an equivalence relation.

- \( \sim_2 = \{\langle a, a\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle b, b\rangle, \langle b, c\rangle, \langle c, b\rangle, \langle c, c\rangle\} \) is not an equivalence relation because it not transitive: \( a \sim_2 b \land b \sim_2 c \) but \( a \not\sim_2 c \)
**Equivalence classes**

**Definition:** For an equivalence relation \( \sim \) on \( S \), the **equivalence class** of an element \( s \in S \) is the subset of all elements in \( s' \in S \) which are equivalent to \( s \), \( s' \sim s \):

\[
[s] = \{s' \in S: s' \sim s\}
\]

**Definition:** Given an equivalence relation \( \sim \), The set of all equivalence classes in \( S \) is usually denoted as

\[
S/\sim = \{[s]: s \in S\}
\]

and called the **quotient set** of \( S \) by \( \sim \)

**The set of all equivalence classes of \( S \) forms a partition of \( S \)**
Example

- **For the set** \( S = \{aba, ima, ben, bat, kelev, kalba\} \)

- **And the relation**
  \( \sim = \{(aba, ben), (ben, aba), (aba, kelev), (kelev, aba), (ben, kelev), (kelev, ben)\} \)
  \( \cup \{(ima, bat), (bat, ima), (ima, kalba), (kalba, ima), (bat, kalba), (kalba, bat)\} \)
  \( \cup \{(aba, aba), (ima, ima), (ben, ben), (bat, bat), (kelev, kelev), (kalba, kalba)\} \)

- **The set of equivalence classes is:**

\[
S/\sim = \{[ben], [bat]\} = \{\{aba, ben, kelev\}, \{ima, bat, kalba\}\}
\]
For a transition system, \( T = \langle S, \Sigma, \rightarrow, S_0, S_F \rangle \) and an equivalence relation \( \sim \) such that \( S_0 \) and \( S_F \) are both unions of equivalence classes we define the quotient transition system as:

\[
T/\sim = \langle S/\sim, \Sigma, \rightarrow_{\sim}, S_0/\sim, S_F/\sim \rangle
\]

where for \( S_1, S_2 \in S/\sim \)

\[
\langle S_1, \sigma, S_2 \rangle \in \rightarrow_{\sim}
\]

if and only if

\[
\exists s_1 \in S_1, s_2 \in S_2 \text{ such that } \langle s_1, \sigma, s_2 \rangle \in \rightarrow
\]

Notice that the quotient transition system may be non-deterministic, even if the original system is...
Example
Simulation

Given two transitions systems $T_1$ and $T_2$, $T_2$ is said to simulate $T_1$ if for every path of $T_1$, there is a corresponding path in $T_2$.

Example of bisimilar systems:

This concept of simulation is formalized in the following definitions
For $\sigma \in \Sigma$ define the $Pre_\sigma : 2^S \rightarrow 2^S$ operator as:

$$Pre_\sigma(P) = \{ s \in S : \exists s' \in P \text{ such that } \langle s, \sigma, s' \rangle \in \rightarrow \}$$
Bisimulation

Given \( T = (S, \Sigma, \rightarrow, S_0, S_F) \), and an equivalence relation \( \sim \) over \( S \), \( \sim \) is called a **bisimulation** if:

1. \( S_0 \) is a **union of equivalence classes**

2. \( S_F \) is a **union of equivalence classes**

3. For all \( \sigma \in \Sigma \), if \( P \) is a **union of equivalence classes** then \( Pre_\sigma(P) \) is also a **union of equivalence classes**

We say that two systems are bisimilar if one is the quotient transition system of the other with respect to a bisimulation.
Example

\[ p \rightarrow p_1 \rightarrow p_2 \leftarrow p_1' \rightarrow p_2' \]

\[ q \rightarrow q_1 \rightarrow q_2 \leftarrow q_1' \rightarrow q_2' \]
Example
Local characterization of bisimulations

Proposition:

∼ is a bisimulation if and only if:

1. \((s_1 \sim s_2) \land (s_1 \in S_0) \Rightarrow (s_2 \in S_0)\)
2. \((s_1 \sim s_2) \land (s_1 \in S_F) \Rightarrow (s_2 \in S_F)\)
3. \((s_1 \sim s_2) \land \langle s_1, \sigma, s'_1 \rangle \in \rightarrow \Rightarrow \exists s'_2 \text{ such that } (s'_1 \sim s'_2) \land \langle s_2, \sigma, s'_2 \rangle \in \rightarrow\)

Proof:

(3) (⇒)

If \(P\) is an equivalence class and \(s_1 \in Pre_\sigma(P)\), then \(s_2 \in Pre_\sigma(P)\) for all \(s_2 \sim s_1\). Hence \(Pre_\sigma(P)\) must be a union of equivalence classes.

(3) (⇐)

If \(Pre_\sigma(P)\) is a union of equivalence classes and \(s_1 \sim s_2, \langle s_1, \sigma, s'_1 \rangle \in \rightarrow\) then \(s_2 \in [s_1] \Rightarrow s_2 \in Pre_\sigma([s'_1]) \Rightarrow \exists s'_2 \sim s'_1 \text{ such that } \langle s_2, \sigma, s'_2 \rangle \in \rightarrow\)
Computing bisimulations

\[ S / \sim = \{ S_0, S_F, S \setminus (S_0 \cup S_F) \} \]

while \( \exists P, P' \in S / \sim, \sigma \in \Sigma \) such that \((P \cap Pre_{\sigma}(P')) \notin \{ \emptyset, P \}\)

\[ P_1 = P \cap Pre_{\sigma}(P') \]
\[ P_2 = P \setminus Pre_{\sigma}(P') \]

\[ S / \sim = (S / \sim \setminus \{ P \}) \cup \{ P_1, P_2 \} \]

**Proposition:** If the algorithm terminates then \( \sim \) is a bisimulation

**Proof:** exercise…
Example
Example
Example
Example
Example
Computing bisimulations

- This is only a pseudo algorithm
- Implementation and termination for general transition systems are not obvious
- For finite state systems we can implement the algorithm and guarantee that it terminates
- because we can enumerate the states
Why is bisimulation an improvement?

- No need to enumerate all the states, therefore may have a computational advantage
- Extends to systems with infinite states
- If the bisimulation quotient can be computed and is finite, then the reachability computation is decidable
Example: Multiplying all constants in a TA by a rational factor

- Let $T$ be the transition system defined by a timed automaton, $H$

- Consider a positive rational number $\lambda > 0$

- Let $H_\lambda$ denote the timed automaton obtained by replacing all constants, $c$, in $H$ by $\lambda c$

- Let $T_\lambda$ denote the transition system associated with $H_\lambda$

**Proposition:** $T$ and $T_\lambda$ are bisimilar

**Exercise:** Prove the proposition using $q, x \sim q, \lambda x$. Note that, since $\lambda > 0$, $(x_i \leq c) \iff (\lambda x_i \leq \lambda c)$ and $(x_i \geq c) \iff (\lambda x_i \geq \lambda c)$
W.L.O.G., all constants in a TA are integers

- We can therefore assume all constants in the Timed-Automata that we want to analyze are integers.

- If they are not, we let $\lambda$ be a common multiple of their denominators.

- And consider the bisimilar system $T_\lambda$. 
The main trick for analyzing TA: An equivalence relation

- For each clock $x_i$, let $c_i$ denote the largest constant with which $x_i$ is compared.

- Let $[x_i]$ be the integer part of $x_i$ and $\langle x_i \rangle$ its fractional part. In other words, $x_i = [x_i] + \langle x_i \rangle$ where $[x_i] \in \mathbb{Z}$ and $\langle x_i \rangle \in [0,1)$.

- Consider the relation $\sim \subseteq Q \times X$ with $(q,x) \sim (q',x')$ if:
  - $q = q'$
  - for all $x_i$, $[x_i] = [x'_i]$ or $(x_i > c_i) \land (x'_i > c_i)$
  - for all $x_i, x_j$ with $x_i \leq c_i$ and $x_j \leq c_j$
    $$(\langle x_i \rangle \leq \langle x_j \rangle) \iff (\langle x'_i \rangle \leq \langle x'_j \rangle)$$
  - for all $x_i$ with $x_i \leq c_i$,
    $$(\langle x_i \rangle = 0) \iff (\langle x'_i \rangle = 0)$$
The equivalence classes are either open triangles, open line segments, open parallelograms or points.

The number of classes is $2(12\text{ points } + 30\text{ lines } + 18\text{ open sets})$.

Quite a few, but definitely finite!
Properties of the equivalence relation

∼ is an equivalence relation

∼ is a bisimulation

∼ has a finite number of equivalence classes !!!
Conclusion

Reachability is decidable for Timed Automata

This is not trivial: Reachability is undecidable for time automata with even a single stop-watch!