Timed Automata and TCTL
syntax, semantics, and verification problems

Timed Automata =
Finite Automata + Clock Constraints + Clock resets
Timed Automata and TCTL
syntax, semantics, and verification problems

Clock Constraints

For set $C$ of clocks with $x,y \in C$, the set of clock constraints over $C$, $\Psi(C)$, is defined by

$$\alpha ::= x \prec c \mid x - y \prec c \mid \neg \alpha \mid (\alpha \land \alpha)$$

where $c \in \mathbb{N}$ and $\prec \in \{<,\leq\}$. 
Timed Automata

A timed automaton $A$ is a tuple $(L, l_0, E, \text{Label}, C, \text{clocks}, \text{guard}, \text{inv})$ with

- $L$, a non-empty, finite set of locations with initial location $l_0 \in L$
- $E \subseteq L \times L$, a set of edges
- $\text{Label} : L \rightarrow 2^{\mathbb{AP}}$, a function that assigns to each location $l \in L$ a set $\text{Label}(l)$ of atomic propositions
- $C$, a finite set of clocks
- $\text{clocks} : E \rightarrow 2^C$, a function that assigns to each edge $e \in E$ a set of clocks $\text{clocks}(e)$
- $\text{guard} : E \rightarrow \mathbb{AP}(C)$, a function that labels each edge $e \in E$ with a clock constraint $\text{guard}(e)$ over $C$, and
- $\text{inv} : L \rightarrow \mathbb{AP}(C)$, a function that assigns to each location an invariant.

Timed Automata: Syntax

```
   n
```

Clocks: $x, y$
Guard = clock constraint

```
   x := 0
   x <= 5 \& y > 3
   Action used for synchronization
   a
```

Reset
Action performed on clocks

```
   m
```

Action performed on clocks
Timed Automata: Semantics

Clocks: $x, y$

Guard = clock constraint

Reset
Action performed on clocks

State
(location, $x=v$, $y=u$) where $v,u$ are in $\mathbb{R}$

Transitions

Timed Automata with Invariants

Clocks: $x, y$

Transitions

Invariants insure progress!!
Timed Automata: Example

X:=0

X>=2

X:=0

Timed Automata: Example

X:=0

X>=2

X:=0

value of x

2

4

2 4 6 8 10

time

10
Timed Automata: Example

\[2 \leq x \leq 3\]

\[X := 0\]

\[X := 0\]

Timed Automata: Example

\[X \geq 2\]

\[X := 0\]

\[X \leq 3\]
Timed Automata: Example  
(periodic task)

Timed Automata: Example  
(sporadic task)
Timed Automata: Example
(aperiodic task)

Semantics (definition)

- **clock valuations**: $V(C)$  $v: C \rightarrow R \geq 0$
- **state**: $(l, v)$ where $l \in L$ and $v \in V(C)$

- **action transition**  $(l, v) \xrightarrow{a}(l', v')$ if $f$
  
  $g(v)$ and $v' = v[r]$ and $Inv(l')(v')$

- **delay Transition**  $(l, v) \xrightarrow{d}(l, v + d)$ if $f$
  
  $Inv(l)(v + d')$ whenever $d' \leq d \in R \geq 0$
Timed Automata: Example

\[
\begin{align*}
(\text{off}, x = y = 0) & \xrightarrow{3.5} (\text{off}, x = y = 3.5) \xrightarrow{\text{push}} \\
(\text{on}, x = y = 0) & \xrightarrow{\pi} (\text{on}, x = y = \pi) \xrightarrow{\text{push}} \\
(\text{on}, x = 0, y = \pi) & \xrightarrow{3} (\text{on}, x = 3, y = \pi + 3) \xrightarrow{9-(\pi+3)} \\
(\text{on}, x = 9-(\pi+3), y = 9) & \xrightarrow{\text{click}} (\text{off}, x = 0, y = 9) \ldots
\end{align*}
\]

Modeling Concurrency

- Products of automata
- Parallel composition
CCS Parallel Composition (implemented in UPPAAL)

Where \( a \) is an action \( c! \) or \( c? \) or \( \tau \)
\( c \) is a channel name

The UPPAAL Model

\( = \) Networks of Timed Automata + Integer Variables + ....
Verification Problems

Location Reachability (def.)

\( n \) is reachable from \( m \) if there is a sequence of transitions:

\[
(m, u) \rightarrow^* (n, v)
\]
(Timed) Language Inclusion, $L(A) \subseteq L(B)$

\[(a_0, t_0) (a_1, t_1) \ldots \ldots (a_n, t_n) \in L(A)\]

If

"A can perform $a_0$ at $t_0$, $a_1$ at $t_1$ \ldots \ldots $a_n$ at $t_n$"

\[(l_0, u_0) \xrightarrow{t_0} (l_0, u_0+t_0) \xrightarrow{a_0} (l_1, u_1) \ldots \ldots\]

Verification Problems

- Timed Language Inclusion 🎓
  - 1-clock, finite traces, decidable [Ouaknine & Worrell 04]
  - 1-clock, infinite traces & Buchi-conditions, undecidable [Abdulla et al 05]
- Untimed Language Inclusion 😊
- (Un)Timed Bisimulation 😊
- Reachability Analysis 😊
- Optimal Reachability (synthesis problem) 😊
  - If a location is reachable, what is the minimal delay before reaching the location?
Timed CTL = CTL + clock constraints

Note that The semantics of TA defines a transition system where each state has a Computation Tree

Computation Tree Logic, CTL

*Clarke & Emerson 1980*

**Syntax**

$\phi ::= P \mid \neg \phi \mid \phi \lor \phi \mid EX \phi \mid E[\phi U \phi] \mid A[\phi U \phi]$

where $P \in AP$ (atomic propositions)
TCTL
Henzinger, Sifakis et al 1992

Syntax

\[ \phi :: = P | g | \neg \phi | \phi \land \phi | z.\phi | E[\phi U \phi] | A[\phi U \phi] \]

where \( P \in \text{AP} \) (atomic propositions) and \( g \) is a Clock constraint

\( (l,u) \text{ sat } z.\phi \text{ iff } (l,u[z:=0]) \text{ sat } \phi \)

\( \text{AG (P imply } z.(z<10 \text{ or } q)) \)

Timed CTL (a simplified version of TCTL)

Syntax

\[ \phi :: = p | \neg \phi | \phi \land \phi | \text{EX } \phi | E[\phi U \phi] | A[\phi U \phi] \]

where \( p \in \text{AP} \) (atomic propositions) or a Clock constraint
Timed CTL

Syntax

\[ \phi ::= p \mid \neg \phi \mid \phi \lor \phi \mid \text{EX} \phi \mid E[\phi \cup \phi] \mid A[\phi \cup \phi] \]

where \( p \in \text{AP} \) (atomic propositions) or Clock constraint

Derived Operators

Liveness: \( p \rightarrow q \)  \( \text{“} p \text{ leads to } q \text{”} \)
Bounded Liveness/Response

Verify: “whenever p is true, q should be true within 10 sec

AG ((Pb and x>10) imply q)

Use extra clock x and boolean Pb
Add Pb := tt and x:=0 on all edges leading to location P

This is not really correct; "not Pb" should be added as guard

On all edges leaving q
Bounded Liveness

**Verify:** "whenever p is true, q should be true within 10 sec"

\[ P \implies (q \text{ and } x<10) \]

Use extra clock \( x \)
Add \( x:=0 \) on all edges leading to \( P \)

Timed CTL in UPPAAL

\[ EF \, p \mid AG \, p \mid EG \, p \mid AF \, p \mid p \implies P \]

- \( P \) leads to \( q \)
denotes \( AG (p \implies AF q) \)
Problem with Zenoness

A Zeno-automaton may satisfy the formula Without containing a state where \( q \) is true

\[
y \leq 5
\]

EXAMPLE

We want to specify "whenever \( P \) is true, \( Q \) should be true within 10 time units"
EXAMPLE

We want to specify "whenver P is true, Q should be true within 10 time units"

\[ AG \ ((P_b \ and \ x>10) \ imply q) \]

is satisfied  !!!

EXAMPLE

We want to specify "whenver P is true, Q should be true within 10 time units"

\[ AG \ ((P_b \ and \ x>10) \ imply q) \]
Solution with UPPAAL

Check Zeno-freeness by an extra observer
System || ZenoCheck

A
\( x \leq 1 \)

B
\( x = 1 \)
\( x := 0 \)

ZenoCheck

Check

ZenoCheck.A \rightarrow ZenoCheck.B

Committed location!

REACHABILITY ANALYSIS
using Regions
Infinite State Space!

However, the reachability problem is decidable \( \square \) Alur&Dill 1991

Region: From infinite to finite

Concrete State
\( (n, x=2.2, y=1.5) \)

Symbolic state (region)
\( (n, \quad ) \)

An equivalence class (i.e. a region)
There are only finite many such!!
Region equivalence (Intuition)

\[ u \approx v \iff (l, u) \text{ and } (l, v) \text{ may reach the same set of equivalence classes} \]
Region equivalence (Intuition)

$u \cong v$ iff $(l, u)$ and $(l, v)$ may reach the same set of equivalence classes

Region equivalence [Alur and Dill 1990]

- $u, v$ are clock assignments
- $u \equiv v$ iff
  - For all clocks $x$, either (1) $u(x) > C_x$ and $v(x) > C_x$ or (2) $\lfloor u(x) \rfloor = \lfloor v(x) \rfloor$
  - For all clocks $x$, if $u(x) \leq C_x$, $\{u(x)\} = 0$ iff $\{v(x)\} = 0$
  - For all clocks $x, y$, if $u(x) \leq C_x$ and $u(y) \leq C_y$, $\{u(x)\} \leq \{u(y)\}$ iff $\{v(x)\} \leq \{v(y)\}$
Region equivalence (alternatively)

$u \equiv v$ iff $u$ and $v$ satisfy exactly the same set of constraints in the form of $x_i \sim m$ and $x_i \cdot x_j \sim n$ where $\sim$ is in $\{<,>,\leq,\geq\}$ and $m,n < \text{MAX}$

This is not quite correct; we need to consider the MAX more carefully.

Region Graph

*Finite-State Transition System!!*

OBS: there are only finite many regions
Theorem

\[ u \approx v \implies \begin{align*}
    &\bullet u(x:=0) \approx v(x:=0) \\
    &\bullet u+n \approx v+n \text{ for all natural number } n \\
    &\bullet \text{ for all } d<1: \ u+d \approx v+d' \text{ for some } d'<1
\end{align*} \]

"Region equivalence" is preserved by "addition" and reset.
(Also preserved by "subtraction" if clock values are "bounded")

Region graph of a simple timed automata
Fischers again

Untimed case

Timed case

Problems with Region Construction

- Too many ‘regions’
  - Sensitive to the maximal constants
  - e.g. $x>1,000,000$, $y>1,000,000$ as guards in TA
- The number of regions is highly exponential in the number of clocks and the maximal constants.
REACHABILITY ANALYSIS
using ZONES

Zones: From infinite to finite

State
(n, x=3.2, y=2.5)

Symbolic state (zone)
(n, 1≤x≤4, 1≤y≤3)

Zone: conjunction of
x−y = n, x = n

∞
Symbolic Transitions

Thus \((n, 1 \leq x \leq 4, 1 \leq y \leq 3) \Rightarrow (m, 3 < x, y = 0)\)

Fischer’s Protocol

\textit{analysis using zones}

\textbf{Initially} \\
\quad \text{V} \equiv 1

\begin{align*}
A1 & \quad X < 10 \quad V = 1 \quad X = 0 \\
B1 & \quad X > 10 \quad V = 1 \\
B2 & \quad Y < 10 \quad V = 2 \quad Y = 0 \\
A2 & \quad Y > 10 \quad V = 2
\end{align*}

Critical Section
Fischers cont.

Untimed case

\[ A_1, A_2, v=1 \quad \longrightarrow \quad A_1, B_2, v=2 \quad \longrightarrow \quad A_1, CS_2, v=2 \quad \longrightarrow \quad B_1, CS_2, v=1 \quad \longrightarrow \quad CS_1, CS_2, v=1 \]

Fischers cont.

Untimed case

\[ A_1, A_2, v=1 \quad \longrightarrow \quad A_1, B_2, v=2 \quad \longrightarrow \quad A_1, CS_2, v=2 \quad \longrightarrow \quad B_1, CS_2, v=1 \quad \longrightarrow \quad CS_1, CS_2, v=1 \]

Taking time into account

\[ Y \quad \longrightarrow \quad X \]
Fischers cont.

Untimed case

Taking time into account
Fischers cont.

Untimed case

Taking time into account

Fischers cont.

Untimed case

Taking time into account
Zones = Conjuctive constraints

- A zone \( Z \) is a conjunctive formula:
  \[ g_1 \& g_2 \& \ldots \& g_n \]
  where \( g_i \) may be \( x_i \sim b_i \) or \( x_i - x_j \sim b_{ij} \)
- Use a zero-clock \( x_0 \) (constant 0), we have
  \[ \{x_i-x_j \sim b_{ij} | \sim \text{ is } < \text{ or } \leq, \ i,j \leq n\} \]
- This can be represented as a MATRIX, DBM (Difference Bound Matrices)

Solution set as semantics

- Let \( Z \) be a zone (a set of constraints)
- Let \([Z]=\{u | u \text{ is a solution of } Z\}\)

(We shall simply write \( Z \) instead \([Z]\))
### Operations on Zones

- **Strongest post-condition (Delay):** $SP(Z)$ or $Z↑$
  - $[Z↑] = \{u+d| d \in R, u \in [Z]\}$

- **Weakest pre-condition:** $WP(Z)$ or $Z↓$ (the dual of $Z↑$)
  - $[Z↓] = \{u| u+d \in [Z] \text{ for some } d \in R\}$

- **Reset:** $\{x\}Z$ or $Z(x:=0)$
  - $[\{x\}Z] = \{u[0/x] | u \in [Z]\}$

- **Conjunction**
  - $[Z\&g] = [Z] \cap [g]$

---

### Two more operations on Zones

- **Inclusion checking:** $Z_1 \subseteq Z_2$
  - solution sets

- **Emptiness checking:** $Z = \emptyset$
  - no solution
Theorem on Zones

The set of zones is closed under all zone operations

- That is, the result of the operations on a zone is a zone
- Thus, there will be a zone to represent the sets: $[Z\uparrow]$, $[Z\downarrow]$, $\{x\}Z$

One-step reachability: $S_i \rightarrow S_j$

- **Delay**: $(n,Z) \rightarrow (n,Z')$ where $Z' = Z\uparrow \land \text{inv}(n)$
- **Action**: $(n,Z) \rightarrow (m,Z')$ where $Z' = \{x\}(Z \land g)$
  
  ![Diagram](image)
  
  if $n \rightarrow g \rightarrow x:=0 \rightarrow m$

- **Reach**: $(n,Z) \rightarrow (m,Z')$ if $(n,Z) \rightarrow (m,Z')$
- **Successors** $(n,Z) = \{(m,Z') | (n,Z) \rightarrow (m,Z'), Z' \neq \emptyset\}$
Now, we have a search problem